X Simpósio Brasileiro em Segurança da Informação
e de Sistemas Computacionais
11 a 15 de outubro de 2010
Fortaleza, CE

Minicursos

Editora
Sociedade Brasileira de Computação (SBC)

Organizadores
Luciano Porto Barreto (UFBA)
André Luiz Moura dos Santos (UECE)
Rossana Maria de Castro Andrade (UFC)

Realização
Universidade Estadual do Ceará (UECE)
Universidade Federal do Ceará (UFC)

Promoção
Sociedade Brasileira de Computação (SBC)

Organizadores
Luciano Porto Barreto
André Luiz Moura dos Santos
Rossana Maria de Castro Andrade

Minicursos do X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2010)

Porto Alegre
Sociedade Brasileira de Computação – SBC
2022

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (10. : 11-15 out. 2010 : Fortaleza)
Minicursos do X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2010) [recurso eletrônico] / organização: Luciano Porto Barreto ; André Luiz Moura dos Santos ; Rossana Maria de Castro Andrade. Dados eletrônicos. – Porto Alegre : Sociedade Brasileira de Computação, 2022.
286 p. ; PDF

Inclui bibliografia
ISBN 978-85-7669-502-8 (e-book)

1. Ciência da Computação. 2. Segurança da informação. 3. Sistemas computacionais. I. Barreto, Luciano Porto. II. Santos, Luiz Moura dos. III. Andrade, Rossana Maria de Castro. IV. Universidade Estadual do Ceará. V. Universidade Federal do Ceará. VI. Sociedade Brasileira de Computação. VII. Título.

CDU 004(063)

Ficha catalográfica elaborada por Jéssica Paola Macedo Müller – CRB-10/2662

Biblioteca Digital da SBC – SBC OpenLib

Índice para catálogo sistemático:

1. Ciência e tecnologia informáticas : Computação : Processamento de dados – Publicação de conferências, congressos, simpósios etc... 004(063)

Sumário

Capítulo 1

Gerenciamento de Identidades Federadas[1]

Michelle S. Wangham[◇], Emerson Ribeiro de Mello[†], Davi da Silva Böger[⋆], Marlon Guerios[‡], Joni da Silva Fraga[⋆]

[⋆] Universidade Federal de Santa Catarina

[◇] Universidade do Vale do Itajaí

[†] Instituto Federal de Santa Catarina

[‡] Inohaus

email:wangham@univali.br, mello@ifsc.edu.br, dsboger@das.ufsc.br, marlonguerios@inohaus.com.br, fraga@das.ufsc.br

Abstract

Identity management is an integrated system of policies, business processes and technologies that enables organizations to provide resources safely, only to their users. This also involves issues related to definition, certification and life cycle management of digital identities. The federated identity model is an approach to optimize the exchange of identities information through federations that share trust relantionships. This chapter analyzes the challenges and proposed solutions to provide federated identity management to collaborative networks.

Resumo

O gerenciamento de identidades consiste de um sistema integrado de políticas, processos de negócios e tecnologias que permite às organizações proverem recursos de forma segura, somente aos seus usuários. Este também envolve aspectos relacionados com a definição, certificação e gerenciamento do ciclo de vida das identidades digitais. O modelo de identidades federadas é uma abordagem para otimizar a troca de informações relacionadas às identidades, através de relações de confiança construídas nas federações. Neste capítulo, são analisados os desafios e as soluções para prover gerenciamento de identidades federadas às redes colaborativas.

[1]Desenvolvido dentro do escopo do projeto "Serviços para Transposição de Credenciais de Autenticação Federadas", GT-STCFed da RNP.

1.1. Introdução

Segundo [Camarinha-Matos et al. 2008], um novo ambiente competitivo para as diversas organizações governamentais e privadas vem se desenvolvendo nos últimos anos e a tendência para negócios colaborativos está forçando uma mudança na forma na qual estas organizações são gerenciadas. Segundo os autores, a participação em redes colaborativas tem sido muito importante para as organizações que anseiam encontrar uma vantagem competitiva diferenciada. Uma rede colaborativa consiste de várias entidades autônomas, heterogêneas e geograficamente distribuídas, que colaboram para encontrar um objetivo comum e compatível e cujas interações são suportadas pelas redes de computadores [Camarinha-Matos et al. 2008]. Três tipos de redes colaborativas serão abordados neste capítulo: as redes colaborativas de organizações (*Collaborative Networks of Organizations* - CNO), as redes nacionais de pesquisa e educação (*National Research and Education Network* - NREN) e as redes colaborativas governamentais (*Collaborative Digital Government* ou *Collaborative E-Government*).

As redes colaborativas possuem uma série de requisitos de interoperabilidade e de segurança. A interoperabilidade é necessária para tratar vários aspectos de heterogeneidade entre os membros da rede, que incluem as diversas plataformas computacionais utilizadas, as várias políticas (administrativas, de segurança, de negócios) às quais esses membros estão sujeitos e as diferentes tecnologias de segurança adotadas. Um suporte a essa heterogeneidade é essencial para garantir que a rede possa atender o maior número possível de participantes. A segurança, por sua vez, é fundamental para que os membros de uma rede colaborativa possam depositar confiança nas interações com outros membros.

Os modelos usuais de autorização se apoiam em uma autoridade de autenticação para mediar a confiança entre partes desconhecidas (terceira parte confiável). Desta forma, as interações entre essas partes são alcançadas pela apresentação de credenciais emitidas por uma autoridade de autenticação confiável. Em ambientes complexos como as redes colaborativas, este modelo de simples intermediação se apresenta como limitado, já que cada domínio possui suas próprias políticas, infraestruturas de segurança e ainda uma forma particular de gerenciar as identidades dos principais [Jøsang e Pope 2005].

O *gerenciamento de identidades* consiste de um sistema integrado de políticas, processos de negócios e tecnologias que permite às organizações o tratamento e manipulação de identidades (atributos de identidade) de seus usuários [Jøsang e Pope 2005, Chadwick 2009]. O gerenciamento de identidades também envolve aspectos relacionados com a definição, certificação e gerenciamento do ciclo de vida das identidades digitais, infraestruturas para troca e validação dessas informações, juntamente com os aspectos legais [Jøsang e Pope 2005].

Nas redes colaborativas, o aumento de provedores de serviços e a crescente necessidade de compartilhar recursos para usuários de diferentes organizações que possuam algum tipo de afinidade são fatores que motivam a constituição de federações. Uma federação é uma forma de associação de parceiros de uma rede colaborativa que usa um conjunto comum de atributos, práticas e políticas para trocar informações e compartilhar serviços, possibilitando a cooperação e transações entre os membros da federação [Carmody et al. 2005]. O gerenciamento de identidades federadas, baseado nestes acordos comerciais, técnicos e políticos, permite que as organizações de uma federação inte-

rajam com base na gestão da identidade compartilhada [IBM 2005]. Como exemplo de soluções que proveem o gerenciamento de identidades federadas, tem-se o *Shibboleth*, os *frameworks* do *Liberty Alliance*, *OpenID* e o *Microsoft CardSpace*.

Uma das funções básicas oferecidas por essas soluções de federação de identidades é a autenticação única (*Single Sign-On* - SSO) [Maler e Reed 2008]. Esta autenticação traz facilidades para os usuários, pois permite que esses passem pelo processo de autenticação uma única vez e usufruam das credenciais obtidas por todos os serviços que desejarem acessar. Garantir tal conceito dentro de um único domínio administrativo e de segurança não é algo complexo, porém garantir o SSO em uma federação com diferentes tecnologias de segurança é algo desafiador [de Mello et al. 2009].

Um problema a ser tratado no gerenciamento de identidades é a privacidade das informações [Hansen et al. 2008]. Em um cenário ideal, os usuários devem exercer o direito de determinar como suas informações serão manipuladas, informando quais informações poderão ser compartilhadas com terceiros, como esse compartilhamento deve ser feito e também indicando o período de tempo o qual essas informações poderão ficar disponíveis nos sistemas.

O objetivo deste capítulo é analisar os desafios e as propostas de soluções para prover gerenciamento de identidades federadas às redes colaborativas, em especial, para as organizações virtuais, para as redes nacionais de pesquisa e educação e para redes colaborativas governamentais. Os conceitos, problemas e soluções propostas para o gerenciamento de identidades apresentados neste capítulo são complementados com a apresentação de cenários de uso em redes colaborativas que demonstram a aplicabilidade das soluções de gerenciamento de identidades apresentadas.

Este capítulo está dividido em sete seções. Nesta primeira seção, foi apresentado o contexto geral em que o trabalho está inserido, destacando os objetivos e a motivação para a escolha do tema. Na Seção 1.2, os conceitos básicos sobre gerenciamento de identidades são introduzidos e os modelos de gerenciamento de identidades presentes na literatura são descritos e comparados. Ainda nesta seção, os requisitos necessários que um sistema de gerenciamento de identidades deve atender, tais como, privacidade, anonimato, interoperabilidade das identidades e gerenciamento de confiança, são apresentados. A Seção 1.3 apresenta os benefícios e funcionalidades oferecidos pelo gerenciamento de identidades federadas e como este modelo tem sido empregado em diferentes redes colaborativas. A Seção 1.4 tem por objetivo apresentar as principais soluções de gerenciamento de identidades federadas e compará-las. Os problemas relacionados à privacidade das identidades federadas e as soluções existentes na literatura para tratar deste importante requisito de segurança são apresentados na Seção 1.5. Na Seção 1.6 são descritas as principais bibliotecas e ferramentas disponíveis para implantar o gerenciamento de identidades em redes colaborativas. Por fim, a Seção 1.7 traz uma síntese dos principais aspectos do gerenciamento de identidades analisados e as tendências de pesquisa nesta área.

1.2. Conceitos e Modelos de Gerenciamento de Identidades

A Internet propiciou uma maior agilidade nas interações entre provedores de serviços e seus usuários. Um provedor de serviços pode ser caracterizado como uma loja virtual, um sistema acadêmico para realizar matrículas, um portal do governo e os usuários destes

sistemas podem ser caracterizados como clientes, alunos e contribuintes. Mecanismos de autorização garantem que somente usuários autorizados poderão obter acesso aos recursos providos, o que imediatamente sugere a necessidade de uma forma para a identificação digital desses usuários e uma forma para gerenciar tais identidades.

A identidade de uma pessoa é composta por uma grande quantidade de informações pessoais que caracteriza essa pessoa em diferentes contextos dos quais essa faz parte [Clauß e Köhntopp 2001]. A identidade é composta pela combinação de subconjuntos, chamados de *identidades parciais*, cujo alguns identificam unicamente uma pessoa (p.ex. cpf) e outros não (p.ex. sexo). Dependendo do contexto e da situação uma pessoa pode ser representada por uma identidade parcial diferente. A identidade parcial de uma pessoa no contexto de uma universidade pode conter informações como seu nome, data de nascimento e as disciplinas que cursa. No contexto de uma empresa, a identidade pode estar associada com funções, privilégios, direitos e responsabilidades. Cabe salientar que uma mesma informação pessoal pode estar presente em diferentes identidades parciais.

Um **sistema de gerenciamento de identidades** provê ferramentas para o gerenciamento dessas identidades parciais em um mundo digital. Segundo [Chadwick 2009], o gerenciamento de identidades consiste em um conjunto de funções e habilidades, como administração, descoberta e troca de informações, usadas para garantir a identidade de uma entidade e as informações contidas nessa identidade, permitindo assim que relações comerciais possam ocorrer de forma segura. Enquanto no mundo real uma pessoa escolhe quais informações revelar de si a outras pessoas, levando em consideração o contexto e a sensibilidade da informação, no mundo digital essa tarefa é desempenhada pelo *sistema de gerenciamento de identidades*.

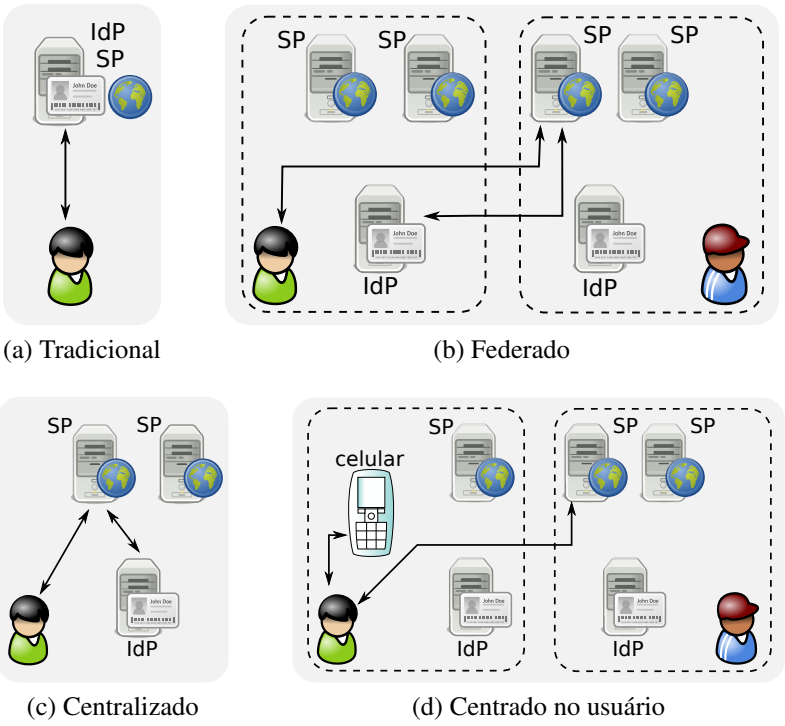
Assim, um *sistema de gerenciamento de identidades* consiste na integração de políticas e processos de negócios, resultando em um sistema de autenticação de usuários aliado a um sistema de gerenciamento de atributos. Em [Bhargav-Spantzel et al. 2007], o *sistema de gerenciamento de identidades* é caracterizado pelos seguintes elementos:

- **Usuário** – aquele que deseja acessar algum serviço;
- **Identidade** – conjunto de atributos de um usuário. Pode ser seu nome, endereço, filiação, data de nascimento, etc;
- **Provedor de Identidades (Identity Provider – IdP)** – responsável por emitir a identidade de um usuário. Após o usuário passar por um processo de autenticação, este recebe uma credencial, dita identidade, que é reconhecida como válida pelos provedores de serviço;
- **Provedor de Serviços (Service Provider – SP)** – oferece recursos a usuários autorizados, após verificar a autenticidade de sua identidade e após comprovar que a mesma carrega todos os atributos necessários para o acesso.

[Chadwick 2009] apresenta a definição de outros elementos presentes em um *sistema de gerenciamento de identidades*. O **identificador** consiste em um conjunto de caracteres e dígitos que são usados para identificar de forma única uma entidade em um

determinado domínio ou sistema. Uma **asserção de atributos** consiste em uma declaração, emitida por um terceiro confiável, indicando que uma determinada entidade possui os referidos atributos.

Em [Jøsang e Pope 2005, Jøsang et al. 2005, Bhargav-Spantzel et al. 2007], os *sistemas de gerenciamento de identidades* seguem modelos classificados como *tradicional*, *centralizado*, *federado* e *centrado no usuário*. Cada um desses modelos apresenta uma forma diferente de interação e disposição dos elementos citados acima. A Figura 1.1 ilustra cada modelo e estes serão descritos em detalhes a seguir.



(a) Tradicional (b) Federado

(c) Centralizado (d) Centrado no usuário

Figura 1.1: Classificação dos modelos de gerenciamento de identidade

O *modelo tradicional* é amplamente utilizado nos atuais sistemas computacionais presentes na Internet. Neste modelo, a identificação do usuário é tratada de forma isolada por cada provedor de serviços, o qual também atua como provedor de identidades (ver Figura 1.1a). Cabe ao usuário criar uma identidade digital para cada provedor de serviços que deseje interagir, não havendo assim o compartilhamento das identidades desses usuários entre diferentes provedores de serviço.

Apesar do *modelo tradicional* ser amplamente adotado, seu uso tende a ser custoso tanto para usuários quanto para provedores de serviços. Atualmente, é comum um usuário possuir múltiplas identidades para interagir com seu provedor de *e-mails*, sítio de notícias, livraria, etc. Cada provedor de serviços pode exigir um conjunto próprio de atributos para compor a identidade digital do usuário. Por outro lado, um conjunto comum de atributos pode ser exigido por diversos provedores de serviços, como nome da conta, senha, endereço, data de nascimento, etc [de Mello 2009].

Para os usuários, gerenciar inúmeras identidades é algo custoso. Primeiro por ter

que fornecer as mesmas informações diversas vezes. Segundo por ter que se preocupar em criar um nome de usuário e senha diferente para cada provedor de serviço, uma vez que usar a mesma senha por diversos provedores não é aconselhado. O custo gerado para os usuários também reflete de diversas formas nos provedores de serviço. A tarefa tediosa de sempre fornecer as mesmas informações no momento da criação de sua identidade, faz com que o usuário não seja tão fiel no preenchimento de atributos que não são cruciais para acessar o recurso oferecido pelo provedor.

O *modelo centralizado* surgiu como uma solução para a inflexibilidade do *modelo tradicional* e está fundamentado no compartilhamento das identidades dos usuários entre provedores de serviços e no conceito de autenticação única (*Single Sign-on* – SSO) [Bhargav-Spantzel et al. 2007]. O serviço *Microsoft Passport Network* foi um percursor deste modelo e visou evitar as inconsistências e redundâncias presentes no *modelo tradicional*, dando aos seus usuários a possibilidade de interagir com diversos provedores de serviços sem que necessitem realizar o processo de autenticação manual em cada um desses [Bhargav-Spantzel et al. 2007].

No *modelo centralizado*, só existe um único provedor de identidades o qual é responsável por autenticar os usuários, fornecer aos provedores de serviços informações sobre estes e todos os provedores de serviços devem confiar plenamente nas informações fornecidas pelo provedor de identidades (ver Figura 1.1c). O conceito de autenticação única (SSO) traz uma grande facilidade para os usuários, pois a estes só é necessário realizar o processo de autenticação uma única vez e usufruir das credenciais obtidas por todos os provedores de serviços que utilizar, até que estas credenciais expirem.

Segundo [Maliki e Seigneur 2007], o ponto fraco deste modelo é que o provedor de identidades possui controle absoluto sobre as informações de seus usuários, podendo assim usá-las da forma que bem entender, sendo este o principal motivo para o *Microsoft Passport Network* não ter tido sucesso.

Visando contornar as dificuldades apresentadas pelo modelo centralizado, o *modelo de identidade federada* está fundamentado sobre a distribuição da tarefa de autenticação dos usuários por múltiplos provedores de identidades, estando estes dispostos em diferentes domínios administrativos (ver Figura 1.1b). Um domínio administrativo pode representar uma empresa, uma universidade, etc e é composto por usuários, diversos provedores de serviços e um único provedor de identidades.

O gerenciamento de identidades federadas é uma abordagem para otimizar a troca de informações relacionadas a identidade através de relações de confiança construídas nas federações [Camenisch e Pfitzmann 2007]. Acordos estabelecidos entre provedores de identidades garantem que identidades emitidas em um domínio sejam reconhecidas por provedores de serviços de outros domínios e o conceito de autenticação única é garantido mesmo diante de diferentes domínios. Dessa forma, o *modelo de identidades federadas* consegue oferecer facilidades para os usuários, pois evita que estes tenham que lidar com diversas identidades e passar diversas vezes pelo processo de autenticação. Para os provedores de serviços, o benefício é que estes poderão lidar com um número menor de usuários temporários. O projeto *Liberty Alliance*, descrito na Seção 1.4.3, e o projeto *Shibboleth*, descrito na Seção 1.4.2, seguem o modelo de gerenciamento de identidades federadas (*Federated Identity Management* – FIM).

O aumento no número de provedores de serviços resultou em um número maior de identidades que o usuário deverá gerenciar. Modelos de gerenciamento como o centralizado e o federado trazem facilidades para esses usuários. A crítica sobre o modelo centralizado se faz principalmente sobre o provedor de identidades, que além de ser um ponto único de falhas possui total poder sobre os dados de seus usuários. [Jøsang e Pope 2005, Bhargav-Spantzel et al. 2007] consideram o gerenciamento de identidades federadas como um modelo centrado nos provedores de identidades. Apesar de haver a distribuição das identidades por diversos provedores, informações desses usuários, uma vez liberadas para esses provedores, podem ser disponibilizadas a terceiros.

O *modelo centrado no usuário* objetiva dar ao usuário o total controle sobre suas identidades digitais, contudo as principais propostas e implementações deste modelo fazem uso de um dos modelos apresentados anteriormente, sendo o modelo de identidade federado o mais usado. Na proposta de [Jøsang e Pope 2005] as identidades de um usuários, destinadas a diferentes provedores de serviços, são armazenadas em um dispositivo físico que fica em poder do usuário, como um *smartcard* ou mesmo um telefone celular (ver Figura 1.1d). O usuário se autentica neste dispositivo físico e cabe a este liberar as informações do usuário para cada provedor de serviços que o usuário acessar, respeitando totalmente as preferências de privacidade do usuário. As soluções OpenID (Seção 1.4.5), Microsoft CardSpace (Seção 1.4.6) e Higgins (Seção 1.4.7) seguem este modelo.

Duas abordagens de identificadores tem sido utilizadas para prover a infraestrutura necessárias para implantação do modelo centrado no usuário [Recordon e Reed 2006]:

- *Identidade Baseada no Endereço* – usa um único endereço digital para identificar o usuário, no contexto de uma relação particular. Este endereço digital é então referenciado para descobrir e invocar serviços de identidades associados. O OpenID segue esta abordagem;
- *Identidade Baseada no Cartão* – usa um *token* digital que contém ou faz referência a um conjunto de atributos que, individualmente ou coletivamente, identifica o usuário e fornece as informações necessárias para realizar uma transação baseada na identidade. Esta é a abordagem empregada por tecnologias que utilizam o protocolo WS-Trust, como *Cardspace* da Microsoft e o Projeto Higgins.

Os modelos apresentados basicamente diferem na forma como as identidades dos usuários são armazenadas e disponibilizadas, sendo que em alguns modelos o compartilhamento da identidade entre provedores de serviços, aliado ao conceito de autenticação única trazem facilidades para os usuários e são hoje essenciais, haja visto o grande número de serviços oferecidos através da Internet [Bhargav-Spantzel et al. 2007].

[Damiani et al. 2003] lista um conjunto de requisitos que um sistema de gerenciamento de identidades deve atender para garantir uma melhor experiência de uso para os usuários sem que isso afete a segurança de suas informações pessoais. Os requisitos listados por [Damiani et al. 2003] são:

- *Interoperabilidade* – As identidades dos usuários devem ser representadas em um formato comum de forma a permitir que a mesma possa ser compreendida e validada mesmo diante de múltiplos domínios administrativos e de segurança;

- *Mecanismo para revogação de identidades* – O sistema deverá prover uma forma para que seus usuários possam gerenciar as informações contidas em suas identidades bem como revogá-las;
- *Gerenciamento de confiança* – Relações de confiança entre provedores de serviços e provedores de identidades de diferentes domínios permitem que as identidades emitidas em um domínio sejam aceitas em outro. É necessário prover uma forma para indicar o nível de confiança associado a cada relação, o que irá influenciar no comportamento dos provedores de serviço;
- *Privacidade* – Usuários devem possuir meios para que possam expressar, e fazer valer, suas preferências de privacidade sobre as informações pessoais presentes em suas identidades;
- *Anonimato* – Aos usuários deve ser garantido o direito de permanecerem anônimos de forma que as informações fornecidas com sua identidade digital não possam ser usadas para descobrir dados de suas outras identidades. O uso de pseudônimos é uma forma para garantir o anonimato.

Segundo [Chadwick 2009], após a falta de sucesso do sistema de gerenciamento de identidades *Microsoft Passport*, Kim Cameron [Cameron 2005] elaborou uma lista com sete leis que ele julga necessárias para que um sistema de identidades obtenha sucesso, são estas:

1. **Consenso e controle do usuário** – Um sistema de identidade só deve revelar informações de um usuário após seus consentimento. A confiança do usuário em seu sistema de identidade é crucial para que o usuário continue a usá-lo;
2. **Revelação mínima para um uso restrito** – Deve-se obter a menor quantidade possível de informações relacionadas às identidades dos usuários. Todo sistema computacional possui falhas e está susceptível a ataques, o que poderia assim revelar informações confidenciais dos usuários a uma parte maliciosa. Dessa forma, deve-se manter a menor quantidade de informação possível e apagá-la assim que essa não for mais necessária [Chadwick 2009];
3. **Justificação das partes** – Sistemas de gerenciamento de identidades devem ser projetados para revelar informações sobre as identidades somente para partes que justifiquem tal necessidade. Somente as partes envolvidas diretamente na transação com o usuário deverão conhecer suas identidades;
4. **Identidades direcionadas** – Deve-se prover suporte a identificadores omnidirecionais, usados por entidades públicas e identificadores unidirecionais, usado por entidades privadas. Dessa forma, um usuário poderia escolher quais identificadores usar de acordo com cada provedor de serviços que for interagir, garantindo seu anonimato, mesmo que provedores de serviços entrem em conluio;
5. **Pluralismo de operadores e tecnologias** – Um sistema de identidades deve operar mesmo diante de diferentes tecnologias presentes em múltiplos provedores de

identidades. É necessário que exista um meta identificador aliado a um protocolo comum para o transporte das identidades;

6. **Integração com o usuário** – O usuário deve ser tratado como uma parte integrante do sistema de identidades para assim se proteger contra ataques de roubo de identidade. A interação humana com os sistemas é o elo mais fraco e a segurança nessa comunicação é algo essencial. Os usuários devem estar familiarizados com sistema de forma a identificar facilmente um ataque tão logo este ocorra;

7. **Experiência consistente através de contextos** – A unificação dos meta sistemas de identificação devem garantir aos usuários uma experiência de uso simples e consistente, mesmo atravessando contextos com diferentes operadores e tecnologias.

1.2.1. Níveis de garantia

Como visto na seção anterior, os sistemas de *gerenciamento de identidades federadas* apresentam facilidades para usuários e para provedores de serviços. Apesar das relações de confiança garantirem que as asserções de segurança emitidas pelos provedores de identidades serão consideradas válidas pelos provedores de serviços, essas não indicam a robustez do processo de autenticação pelo qual o usuário passou junto ao provedor de identidades.

O Instituto Nacional Americano de Padrões e Tecnologias (*National Institute of Standards and Technology* – NIST) lançou um guia sobre o processo de autenticação eletrônico [Burr et al. 2006] no qual foram definidos 4 níveis de garantia, do inglês *Level of Assurance* (LoA), sendo o nível 1 considerado o mais fraco e o nível 4 o mais robusto, indicando os requisitos técnicos para cada nível. Por exemplo, um processo de autenticação que faz uso de nomes de usuário e senha é considerado menos robusto que um processo que faça uso de certificados digitais com chaves públicas.

Provedores de serviços poderiam fazer uso desses níveis de garantia para oferecer diferentes níveis de permissão de acesso [Chadwick 2009]. No caso, usuários que se autenticarem com um nível 1, poderiam somente ter acesso de leitura a um recurso e usuários com um nível 4 poderiam ler e escrever. O SAML (ver Seção 1.4.1), usado na maioria das soluções de gerenciamento de identidades federadas, permite associar um nível de garantia em suas asserções de autenticação, provendo assim uma forma padronizada para troca dessa informação entre provedores de identidades e de serviços.

O documento do NIST também faz relação ao processo de registro dos usuários. Por exemplo, em um sistema que permite os próprios usuários efetuarem seu cadastro através de uma página *web* tem um nível 1, uma vez que não é possível garantir a real identidade deste usuário. Um processo de registro que exige a presença física da pessoa e esta deve fornecer documentos como RG, CPF e declaração de matrícula na instituição de ensino, terá um nível de garantia maior. A federação acadêmica CAFe [RNP 2010] (ver Seção 1.3.1), apresenta em seus procedimentos para ingresso de provedores de identidades, referências aos níveis de garantia, porém ainda não está muito claro como os provedores de serviços irão usufruir dessa informação.

1.3. Uso de Identidades Federadas

De acordo com [Camenisch e Pfitzmann 2007], em muitas áreas da sociedade, inúmeras transações (volumosas e de significativa importância) são realizadas digitalmente pela Internet. Trata-se de transações comerciais entre empresas (B2B), entre empresas e consumidores (B2C), bem como transações com a administração pública direta (G2B e G2C) e interações entre indivíduos (C2C). Na maioria das transações, os principais desafios são: como os parceiros reconhecem uns aos outros (autenticação)? E como estes obtêm as informações, sobre os parceiros de negócios, necessárias para realizar a transação desejada (troca de atributos)?

A colaboração entre diferentes parceiros em uma rede colaborativa passa pela ativação de uma série de funcionalidades dessas organizações. Como limites administrativos precisam ser transpostos, os processos de negócios estarão sob **diversos modelos administrativos** e podem também estar sob diversos mecanismos e tecnologias de segurança. Domínios administrativos distintos podem formar relações de confiança entre si, constituindo federações, permitindo que a autenticação em um possa ser transposta para os domínios associados (autenticação *Single Sign On* - SSO) [Camenisch e Pfitzmann 2007].

Dentre as formas de Redes Colaborativas (RCs), destacam-se as redes colaborativas de organizações (*Collaborative Networks of Organizations* - CNO), que são sistemas constituídos por componentes autônomos, geograficamente distribuídos, que colaboram através da rede para alcançar um objetivo comum [Camarinha-Matos et al. 2008], as redes nacionais de pesquisa e educação (*National Research and Education Network* - NRENs), provedores de serviço de Internet especializados que oferecem serviços de comunicação avançada para comunidade científica e canais dedicados para projetos de pesquisa [TERENA 2008] e as redes colaborativas governamentais (*Collaborative Digital Government* ou *Collaborative E-Government*), constituídas para apoiar as aplicações de Governo Eletrônico e para prover suporte ao trabalho colaborativo entre Instituições Governamentais [Dawes e Pardo 2008]. As seções a seguir descrevem como o gerenciamento de identidades federadas tem sido empregado nestas redes colaborativas.

1.3.1. Redes Nacionais de Pesquisa e Educação

Segundo a associação TERENA (*Trans-European Research and Education Networking Association*)[2] (2008), é crescente, não só na Europa, o número de países interessados em desenvolver e aprimorar suas redes nacionais (NRENs). No Brasil, tem-se a Rede Nacional de Ensino e Pesquisa (RNP), que oferece uma infraestrutura de rede Internet (rede Ipê), que conecta as principais universidades e institutos de pesquisa do país, cerca de 600 instituições, beneficiando-se de um canal de comunicação rápido e com suporte a serviços e aplicações avançadas.

Além do serviço de conectividade de rede com uso de tecnologias avançadas, as NREN oferecem uma diversidade de serviços para os parceiros que participam destas redes colaborativas, entre estes destacam-se: ferramentas avançadas para comunicação instantânea interativa, tais como videoconferência e Telefonia IP; centro de atendimentos a incidentes de segurança; serviços de vídeo digital; ferramentas de planejamento e ope-

[2] `http://www.terena.org`

ração de redes que monitoram o funcionamento de todos os enlaces da NREN; serviços de sincronização de relógios, *Grid Services* e serviços de comércio eletrônico. O portfólio de serviços oferecidos pelas NREN tem crescido a cada ano [TERENA 2008].

Nas atuais redes nacionais de pesquisa, a crescente necessidade de compartilhar recursos para usuários de diferentes instituições que possuam algum tipo de afinidade motivaram a constituição de **federações acadêmicas**. No contexto das NRENs, o *framework Shibboleth* (ver Seção 1.4.2) é a Infraestrutura de Autenticação e Autorização (*Authentication and Authorization Infrastructure* - AAI) mais empregada para constituição de federações acadêmicas. As federações *Incommon*, da rede norte-americana *Internet 2*, e a *CAFe*, da Rede Nacional de Pesquisa (RNP) do Brasil, são exemplos de federações construídas tendo como base este *framework*. Estas federações agrupam pessoas do meio acadêmico como alunos, técnicos administrativos e professores. Tal comunidade é o principal público alvo de muitas empresas, o que torna interessante a essas empresas o ingresso nas federações acadêmicas também como provedores de serviços.

Em operação piloto desde 2008, a Federação CAFe - Comunidade Acadêmica Federada[3] - reúne instituições de ensino e pesquisa brasileiras em uma rede de confiança, na qual cada instituição é responsável por autenticar e prover informações de seus usuários para provedores de serviços autorizados. A rede de confiança constituída pela federação permite a um usuário autenticado em sua instituição de origem acessar, através de um único login, serviços oferecidos via *web* tanto por sua própria instituição como pelos demais membros da federação.

Na CAFe, instituições de ensino e pesquisa podem integrar-se à Federação como provedores de identidades, provedores de serviços ou como ambos. Instituições de ensino podem, por exemplo, prover serviços de ensino a distância e de colaboração a qualquer usuário da federação. A federação está aberta a participação de empresas, podendo essas atuarem somente como provedores de serviços. Editoras, órgãos de fomento e empresas com esquemas de desconto para estudantes ou professores são exemplos de empresas que podem ter interesse no ingresso na federação CAFe [RNP 2010].

De acordo com [RNP 2010], a participação de uma instituição como um provedor de identidade na Federação CAFe envolve, dentre outros, os seguintes benefícios: (1) uso de um único sistema de controle de acesso para serviços internos e externos à instituição; (2) uma única conta (*login*) por usuário para acesso aos serviços e (3) a garantia de privacidade, pois só serão passadas a cada provedor de serviços as informações necessárias ao controle de acesso a esse serviço (configuradas individualmente por cada provedor de identidade). Já para os provedores de serviços, o principal benefício é a simplificação do procedimento de controle de acesso, pois a autenticação e disponibilização de informações sobre os usuários é realizada pelos provedores de identidades, eliminando a necessidade de manutenção dessas informações no provedor de serviços e a autorização para acesso a um recurso por um usuário pode ser realizada através de um mecanismo de controle de acesso baseado em atributos (*Attribute Based Access Control* - ABAC), como por exemplo, o nome da instituição de origem, tipo de vínculo, ou outro atributo disponibilizado pelo provedor de identidades.

[3] `http://www.cafe.rnp.br`

Na literatura, constata-se um forte interesse na formação de federação de federações, chamada de confederação, para prover um gerenciamento de identidades federadas ainda mais globalizado. Um bom exemplo é o *framework* eduGAIN [Lopez et al. 2006], desenvolvido dentro do projeto Europeu GÉANT, que tem a finalidade de ligar redes nacionais de educação e de pesquisa de países dentro da comunidade Europeia. Esta interligação de diferentes federações de identidades pode ser vista como uma confederação de provedores de identidades. No uso destes serviços, membros de instituições associadas a diferentes federações podem trocar informações de forma segura, como se fizessem parte de um mesmo provedor de identidade nacional. Por exemplo, um pesquisador visitante em um dos países da comunidade Europeia, com esta infraestrutura, poder acessar facilmente recursos em sua instituição de origem. Esta interconexão possibilita, portanto, que usuários acadêmicos europeus tenham a visão de que as múltiplas redes nacionais colaborativas formam uma única federação. O eduGAIN tem por finalidade a interoperabilidade entre as soluções existente de Infraestruturas de Autenticação e Autorização (AAI) usadas nas construções das federações de identidades.

1.3.2. Organizações Virtuais

Dentre a grande variedade de redes colaborativas de organizações, a cooperação na forma de organizações virtuais (OVs) é a estratégia que mais se destaca. Esta vem sendo adotada por empresas, profissionais liberais e laboratórios espalhados ao redor do mundo, os quais objetivam atender novas oportunidades de negócios (ONs), bem como ampliar sua participação em novos mercados ou alcançar excelência científica [Kürümlüoglu et al. 2005].

Uma OV corresponde a uma união temporária de organizações independentes que se agregam visando compartilhar recursos e funcionalidades para alcançar objetivos que estas não alcançariam sozinhas. A cooperação destas organizações é garantida pela automação e informatização de grande parte de suas infraestruturas, deixando-as acessíveis via Internet. Nas OVs, as alianças são voláteis e o fluxo de colaboração, algumas vezes, é gerado e conhecido dinamicamente, de acordo com o processo de negócio requerido. Diante deste cenário, as infraestruturas para negócios colaborativos necessitam ser mais flexíveis, transparentes e adaptativas, de acordo com as condições do ambiente de negócio e o nível de autonomia das organizações [Rabelo 2008]. O gerenciamento de identidades federadas facilitam uma abordagem integrada para negócios colaborativos [IBM 2005].

Apesar dos benefícios atraentes do gerenciamento de identidades federadas (ver Seção 1.2), este impõem novos riscos contra a privacidade e segurança devido as valiosas informações compartilhadas entre diferentes domínios administrativos utilizando protocolos de rede fracamente acoplados [Maler e Reed 2008]. Um requisito básico para este tipo de cooperação é a relação de confiança entre as organizações. A infraestrutura de confiança que deve ser construída fornece a representação e implementação técnica das relações de negócio e acordos jurídicos entre os parceiros de negócios [IBM 2005].

Como as OVs são sistemas abertos, os melhores padrões para propósito de autenticação devem ser de tecnologia neutra. Tecnologia neutra significa que regras e processos acomodam diferentes tecnologias desde que estas produzam os mesmos resultados. No setor privado, as principais iniciativas incluem *Liberty Alliance framework*, OpenID, Higgins, Cardspace e *Shibboleth* [Lewis 2008].

1.3.3. Redes Colaborativas Governamentais (E-GOV)

Esta seção descreve o uso de identidades federadas em um sistema específico das redes colaborativas governamentais, chamado de Governo Eletrônico (*e-Government - E-Gov*). Os programas de Governo Eletrônico[4] bem sucedidos dependem do uso adequado das tecnologias de informação e comunicação (TICs) empregadas nas organizações para promover os trabalhos colaborativos em prol de objetivos comuns [Dawes e Pardo 2008]. Nas redes governamentais, as colaborações podem ser de quatro formas: entre organizações públicas (G2G), entre organizações públicas e o terceiro setor, entre organizações públicas e privadas (G2B) e entre o governo e o cidadão (G2C).

Muitos países estão expandido os seus programas de e-Gov aprimorando as suas infraestruturas de colaboração, sendo que o grande desafio está em garantir a interoperabilidade entre estas infraestruturas devido a heterogeneidade dos procedimentos e dados existentes entre a administração pública central e as locais [Baldoni 2010]. Esta diversidade pode tornar difícil a implementação destes programas. O *gerenciamento de identidades federadas* é então um mecanismo necessário para implementar políticas interoperáveis em nível nacional [Gottschalk e Solli-Saether 2008].

A necessidade de autenticação digital está exigindo que os governos melhorem seus processos de emissão de certidões de nascimento, números de serviço social ou carteira de motorista. Os governos estão tendo que decidir como os processos de identificação existentes serão utilizados para a identidade digital e se deverão definir novas leis ou outras medidas para melhorar a emissão de documentos de identidade [Lewis 2008].

Nos programas de e-Gov, o gerenciamento de identidades pode levar a prestação de serviços *online* eficientes [Baldoni 2010]. Como consequência, nos últimos anos, vários governos têm aprovado diretrizes para melhorar os serviços de e-Gov e as medidas de identificação para acesso a informação individual do cidadão e de registros do governo disponíveis em sítios *web*.

A Nova Zelândia, em 2008, lançou um *framework* de interoperabilidade *e-Government Interoperability Framework* - **e-GIF** que define um conjunto de políticas, normas e orientações que abrangem as formas de assegurar a interoperabilidade dos dados do setor público, entre as tecnologias de informação e comunicação (TIC) e entre os processos de negócios eletrônicos. Este *framework* permite que qualquer agência possa juntar suas informações, TICs ou processos com as de outras agências tendo como base um padrão internacional aberto. O problema do gerenciamento de identidades é parte do e-GIF o qual possui sua própria versão da especificação SAML (NZ SAML). A primeira versão do *framework* foi centrada na autenticação e as versões subsequentes terão como foco atributos de identidade e autorização. Quando desejável, a federação de identidades pode incorporar a autenticação única através do serviço *Government Logon Service* (GLS) e utilizar o provedor de identidades e pseudônimos. A identificação é executada através de um *Identities Verification Service* (IVS) que monta uma asserção de identidade para os provedores de serviços da agência do governo e para o usuário de maneira controlada.

[4]O desenvolvimento de programas de Governo Eletrônico tem como princípio a utilização das modernas tecnologias de informação e comunicação (TICs) para democratizar o acesso à informação, ampliar discussões e dinamizar a prestação de serviços públicos com foco na eficiência e efetividade das funções governamentais.

Em 2005, o Reino Unido desenvolveu um sistema de gerenciamento de identidade e um *e-GIF*. O *e-GIF* define as políticas e especificações técnicas que regem os fluxos de informação entre o governo e o setor público. Este *framework* cobre interconectividade, integração de dados e gerenciamento de conteúdo e acesso a *e-services*. O sistema de gerenciamento de identidades oferece serviços através do sítio *web Government Gateway* ou através do *UK Central Government Portal Direct.gov*. Um cidadão, para utilizar qualquer serviço do *Gateway*, precisa primeiro se registrar. Isso geralmente requer que o cidadão forneça o nome completo, endereço de *e-mail* e escolha uma senha. Cada membro irá fornecer fatos conhecidos até mesmo para os serviços individuais que estão sendo inscritos, como por exemplo para o Número de Seguridade Nacional. Ao inscrever-se, este será obrigado a fornecer esses fatos conhecidos para cada serviço, a fim de verificá-los com os dados existentes. Este sistema é baseado em dados existentes nas bases de dados do governo. Cada cidadão registrado receberá em casa um PIN. O *framework* de registro e autenticação fornece diferentes níveis de autenticação, dependendo de cada serviço específico. Existem quatro níveis de autenticação, iniciando em "nenhuma autenticação necessária" e indo até "autenticação biométrica e certificado digitais requeridos".

Na Dinamarca, desde 1986, os cidadãos tem sido registrados digitalmente. O governo tem o objetivo de que até 2012 todas as comunicações escritas relevantes entre as empresas, os cidadãos e o setor público devam ser eletrônicas. Em 2006, o Parlamento decidiu que o uso de padrão aberto deve ser obrigatório em soluções para o setor público. No mesmo ano, foi estabelecido que um portal do cidadão deveria ser criado em substituição aos portais municipais. Um dos pontos principais desta mudança é que os indivíduos, que costumavam ter registros em várias agências e usar várias identidades digitais (ver Seção 1.2), passam a ter acesso a diversos serviços a partir do portal do cidadão, fazendo uso de uma única identidade digital. Para permitir este cenário, aumentou-se a necessidade de uma melhor integração entre os diferentes serviços, o que significa até mesmo uma melhor integração entre os setores público e privado.

Na Itália, em 2001, a reforma da constituição Italiana atribuiu novas possibilidades para ação das autoridades regionais, este processo de descentralização impactou também a infraestrutura de e-gov, devido a interoperabilidade necessária entre as infraestruturas regionais que passaram a ser criadas. Em 2003, foi criado o *National Centre for IT in Public Administration* para promover a integração e cooperação das aplicações em nível nacional. Diversos documentos foram elaborados oferecendo as diretrizes necessárias para prover a interoperabilidade e definindo o *Sistema Pubblico di Connetività - SPC*.

O modelo de cooperação italiana segue uma arquitetura orientada a serviços e o modelo de gerenciamento de identidades federadas é usado para autorização e controle de acesso dos serviços disponíveis no SPC. A federação é necessária para utilizar o sistema de gerenciamento de identidades já existente nas autoridades regionais. A integração é feita através da especificação SAML v2.0.

Visando atender ao requisito de interoperabilidade, o programa de e-Gov do Brasil [GOV.BR 2010] definiu a arquitetura e-PING (Padrões de Interoperabilidade de Governo Eletrônico). Com esta arquitetura busca-se um conjunto mínimo de premissas, políticas e especificações técnicas que regulamentem a utilização da Tecnologia de Informação e Comunicação (TIC) no governo federal, estabelecendo as condições de interação com os

demais Poderes e esferas de governo e com a sociedade em geral. O e-PING ainda está sendo definido e até o momento nenhuma premissa, política ou especificação técnica para o gerenciamento de identidades foi definida.

1.4. Soluções de Gerenciamento de Identidades Federadas

Como as principais soluções de gerenciamento de identidades federadas, são baseadas na especificação SAML, esta seção inicia com a descrição desta especificação. Em seguida, as soluções mais empregadas nas redes colaborativas são descritas e analisadas.

1.4.1. Especificações SAML

A OASIS, um órgão padronizador, lançou um conjunto de especificações para definir uma infraestrutura para troca dinâmica de informações de segurança entre parceiros de negócio. A Linguagem de Marcação para Asserções de Segurança (*Security Assertion Markup Language* – SAML) [OASIS 2005g] tem como núcleo uma gramática XML para representar informações de segurança na forma de asserções, bem como protocolos para requisição e envio dessas asserções [OASIS 2005a].

Nas versões iniciais (1.0 e 1.1), a SAML tinha o objetivo de facilitar a troca de informações de identidade e autorização de usuários para permitir autenticação única (*Single Sign-On* - SSO) na *web*. A versão atual (2.0), por outro lado, estende essa infraestrutura com conceitos e mecanismos derivados de projetos como *Liberty Alliance* (ver Seção 1.4.3) e *Shibboleth* (ver Seção 1.4.2) que têm objetivos mais amplos, como a formação de federações para compartilhamento de informações de segurança e gerenciamento de identidades federadas. A SAML é hoje um padrão de fato e é utilizada, em parte ou por completo, em muitos outros projetos de gerenciamento de identidades, em especial todos os que serão abordados nas seções posteriores deste texto (*Shibboleth*, *Liberty Alliance*, *WS-Federation*, *OpenID* e *CardSpace*).

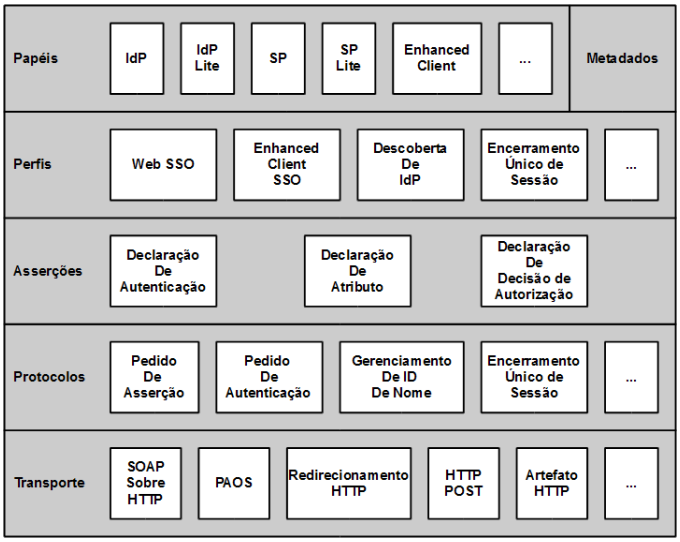


Figura 1.2: Arquitetura da SAML [Maler e Reed 2008]

As especificações SAML definem cinco componentes: papéis, perfis, asserções, protocolos e transporte, ilustrados pela Figura 1.2. No componente *papéis*, além dos

papéis que cada entidade pode desempenhar na infraestrutura SAML, são apresentados também os metadados que descrevem essas entidades. Os *perfis* agregam protocolos e asserções em fluxos de dados específicos para prover funcionalidades como gerenciamento de identidades e autenticação única. As *asserções* são essenciais para o gerenciamento de identidades e de contextos de segurança. Os *protocolos* são usados para requerer e transferir asserções entre as entidades. Esses protocolos podem ser mapeados em diferentes mecanismos de *transporte*. As próximas subseções apresentam estes componentes.

Asserções

As asserções SAML [OASIS 2005a], definidas por uma gramática XML, especificam o formato para representar informações de segurança acerca de um sujeito, que pode ser uma pessoa, organização, computador, etc. Essas informações são atestadas por uma entidade denominada de parte declarante (*asserting party*) ou autoridade SAML (*SAML authority*). A Figura 1.3 apresenta um exemplo de asserção SAML. Essa asserção contém as seguintes informações:

```
1  <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
2      Version="2.0" IssueInstant="2005-01-31T12:00:00Z">
3    <saml:Issuer
4       Format="urn:oasis:names:SAML:2.0:nameid-format:entity">
5      http://idp.example.org
6    </saml:Issuer>
7    <saml:Subject>
8      <saml:NameID
9         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
10       j.doe@example.com
11     </saml:NameID>
12   </saml:Subject>
13   <saml:Conditions NotBefore="2005-01-31T12:00:00Z"
14      NotOnOrAfter="2005-01-31T12:10:00Z" />
15   <saml:AuthnStatement AuthnInstant="2005-01-31T12:00:00Z"
16      SessionIndex="67775277772">
17     <saml:AuthnContext>
18       <saml:AuthnContextClassRef>
19         urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
20       </saml:AuthnContextClassRef>
21     </saml:AuthnContext>
22   </saml:AuthnStatement>
23 </saml:Assertion>
```

Figura 1.3: Exemplo de asserção SAML

- O elemento raiz `saml:Assertion` (linhas 1 e 2) contém a declaração do espaço de nomes da SAML, a versão da SAML usada (atributo `Version`) e o momento em que a asserção foi emitida pela autoridade SAML (atributo `IssueInstant`);
- O elemento `saml:Issuer` (linhas 3 a 6) indica qual autoridade SAML emitiu a asserção, no caso `http://idp.example.org`;
- O elemento `saml:Subject` (linhas 7 a 12) indica o sujeito da asserção, no caso o detentor do e-mail `j.doe@example.com`;

- O elemento `saml:Conditions` (linhas 13 e 14) indica o prazo de validade da asserção, no caso entre 12:00h e 12:10h do dia 31 de janeiro de 2005;
- O elemento `saml:AuthnStatement` (linhas 15 a 22) representa uma declaração de autenticação. Essa declaração indica que o usuário em questão efetuou sua autenticação às 12:00h do dia 31 de janeiro de 2005 (atributo `AuthnInstant`) e que para tal foi usado um mecanismo de transporte protegido por senha (p.e. uma senha enviada sobre um canal SSL) (elemento `AuthnContext`, linhas 17 a 21).

No exemplo acima, a declaração de autenticação indica que o usuário fez uso de nome de usuário e senha como forma de autenticação. No entanto a SAML permite expressar informações de autenticação de diversos outros mecanismos, como certificados X.509 [Housley et al. 2002], *Kerberos* [Kohl e Neuman 1993] e *XML Signature* [Bartel et al. 2002]. Além disso, outros mecanismos podem ser definidos em extensões da especificação.

Apesar do exemplo anterior conter apenas uma declaração de autenticação, asserções podem conter diversas declarações da mesma autoridade sobre o mesmo sujeito. Os tipos de declaração definidos são os seguintes:

- Declaração de autenticação (*Authentication statement*) – criada pela entidade que realizou com sucesso a autenticação do usuário; a declaração indica quando a autenticação foi feita e qual o mecanismo utilizado (usuário/senha, assinatura digital, etc.);
- Declaração de atributo (*Attribute statement*) – contém atributos associados ao sujeito, como nome, filiação, certificado digital, etc. A SAML permite a utilização de gramáticas de atributos quaisquer, no entanto, provê perfis [OASIS 2005f] para a utilização de esquemas importantes como UUID [Leach et al. 2005], X.500/LDAP [ITU-T 2001, Hodges e Morgan 2002] e XACML [OASIS 2005c];
- Declaração de decisão de autorização (*Authorization decision statement*) – indica ações que o sujeito possui o direito de executar sobre determinados recursos.

Protocolos

Além da gramática de asserções, a especificação [OASIS 2005a] define protocolos gerais de pedido e resposta relacionados com troca de asserções SAML, gerenciamento de contextos de segurança (sessões de autenticação) e gerenciamento de identificadores de usuários. Os protocolos SAML são definidos em duas camadas, sendo a camada superior formada pelos esquemas XML das mensagens e a camada inferior composta de especificações de como usar protocolos subjacentes (SOAP, HTTP, etc.) para transportar essas mensagens [OASIS 2005b]. Essa separação garante ao mesmo tempo a interoperabilidade das aplicações e a possibilidade de utilizar os protocolos SAML em diversos cenários distintos nos quais as aplicações possuem restrições específicas (p.e. um navegador *web* provê suporte a protocolos de transporte diferentes de um cliente SOAP). Os seguintes protocolos são definidos pelas especificações [OASIS 2005a, OASIS 2005b]:

- Protocolos da camada superior (esquemas XML)
  - *Protocolo para pedido de autenticação* – define como um sujeito poderá requerer uma nova asserção de autenticação e, opcionalmente, asserções de atributo;
  - *Protocolo de consulta e pedido de asserção* – define consultas para a obtenção de asserções SAML, previamente existentes ou novas, por meio da especificação de um sujeito, dos tipos de asserções desejadas e de parâmetros de busca e filtragem;
  - *Protocolo para encerramento de sessão* – define um mecanismo para permitir que a sessão associada a um sujeito seja terminada (e os recursos associados a esta sejam liberados) simultaneamente em todas as aplicações do sistema;
  - *Protocolo de resolução de artefatos* – provê mecanismos para passagem de mensagens SAML por referência, usando um identificador de tamanho fixo denominado artefato. De posse do artefato, é possível invocar o emissor da mensagem para obter o conteúdo da mensagem;
  - *Protocolo de gerenciamento de identificador de nome* – usado para indicar que o valor ou formato do identificador de nome de algum sujeito será alterado ou que um identificador de nome não será mais usado para se referir ao sujeito;
  - *Protocolo de mapeamento de identificador de nome* – usado para relacionar identificadores diferentes associados a um mesmo sujeito por aplicações diferentes.
- Protocolos da camada inferior (mapeamentos para protocolos de transporte)
  - *SAML sobre SOAP* – define como mensagens dos protocolos SAML são transportadas em envelopes SOAP sobre HTTP;
  - *Redirecionamento HTTP* – define como mensagens dos protocolos SAML são transportadas em mensagens HTTP de redirecionamento;
  - *HTTP POST* – define como mensagens dos protocolos SAML são transportadas codificadas em *base-64* dentro de formulários HTML;
  - *Artefato HTTP* – define como um artefato é transportado usando HTTP, seja em formulário, seja codificado no texto da URL;
  - *SAML URI* – define como recuperar uma asserção SAML a partir de um identificador URI.

Metadados

Nos perfis de uso da especificação SAML, cenários podem conter diversas entidades com papéis distintos. As diferentes entidades devem entrar em acordo quanto a diversos parâmetros como os identificadores das entidades (URI), os protocolos de transportes suportados, os certificados e chaves criptográficas, entre outros. Para garantir que essas informações sejam descritas e obtidas de maneira padronizada, na especificação [OASIS 2005e]

é definido um formato em XML para expressar os metadados das entidades, bem como os perfis para a troca dinâmica dessas informações entre as entidades do sistema.

Os papéis definidos na SAML para as entidades do sistema são [OASIS 2005e]: provedor de identidade (*identity provider* - IdP), uma autoridade SAML responsável por autenticar sujeitos e atestar seus atributos de identidade; provedor de serviço (*service provider* - SP), uma aplicação que consome asserções SAML emitidas por um IdP e, com base nas informações contidas nestas, controla o acesso do sujeito aos recursos; autoridade de autenticação (*authentication authority*), uma autoridade SAML que implementa o protocolo de consulta de asserções contendo declarações de autenticação; ponto de decisão de política (*policy decision point* - PDP), uma autoridade SAML que implementa o protocolo de consulta de asserções contendo declarações de autorização; autoridade de atributos (*attribute authority* - AA), uma autoridade SAML que implementa o protocolo de consulta de asserções contendo declarações de atributos.

Perfil *Web SSO*

Uma das aplicações mais comuns do gerenciamento de identidades federadas é a autenticação única (*Single Sign-On* - SSO) [Maler e Reed 2008], que permite a um sujeito realizar o processo de autenticação uma única vez, junto ao seu provedor de identidades e usufruir das credenciais obtidas para acessar diferentes provedores de serviços. A SAML define o perfil *Web SSO* para prover a funcionalidade SSO na *web*.

No cenário da *Web SSO*, as aplicações envolvidas podem assumir o papel de provedor de identidades (IdP) ou provedor de serviços (SP). Assume-se que o sujeito usa um navegador *web* comum para se comunicar tanto com o SP quanto com o IdP. O perfil define, basicamente, quais ações um SP deve tomar para iniciar um contexto de segurança (sessão) com um sujeito com base em informações de segurança emitidas pelo IdP. Para isso, o perfil faz uso do protocolo de pedido de autenticação juntamente com um dos seguintes mapeamentos: redirecionamento HTTP, HTTP POST e artefato HTTP.

O funcionamento básico do perfil *Web SSO* está ilustrado na Figura 1.4 [OASIS 2005f]. No passo 1, o sujeito, por meio do navegador, tenta acessar um recurso de um SP que não possui um contexto de segurança com esse sujeito. No passo 2, o SP determina qual IdP usar para pedir informações de segurança sobre o sujeito. Essa etapa é dependente da implementação, mas pode fazer uso de documentos de metadados ou do perfil de descoberta de provedores de identidade [OASIS 2005f]. No passo 3, o SP gera uma mensagem `AuthnRequest` (do protocolo de pedido de autenticação) e passa para o navegador, para que este apresente ao IdP. Nessa etapa, qualquer um dos mapeamentos permitidos para o perfil *Web SSO* pode ser usado. No passo 4, o navegador entra em contato com o IdP para que este autentique o sujeito usando a requisição emitida pelo SP. Nessa etapa, o IdP deve usar mecanismos para autenticar o sujeito e gerar as asserções SAML para enviar ao SP. É possível, nessa etapa, que o IdP já possua um contexto de segurança para o sujeito, não sendo necessário que este passe pelo processo de autenticação. No passo 5, o IdP envia a mensagem `Response`, de acordo com o protocolo de pedido de autenticação, para o SP, por intermédio do navegador, usando um dos mapeamentos permitidos. No passo 6, com base nas informações contidas na resposta

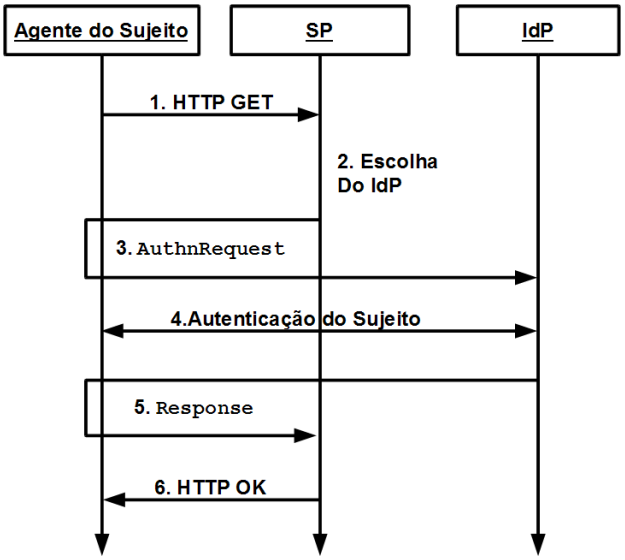


Figura 1.4: Exemplo de funcionamento do perfil *Web SSO*

do IdP entregue pelo sujeito, o SP decide se inicia um contexto de segurança e permite o acesso ao recurso.

Em uma continuação do cenário descrito acima, se o sujeito for acessar um outro SP para o qual também não haja um contexto de segurança, os mesmos passos serão executados, no entanto no passo 4 o IdP poderá responder ao SP sem precisar pedir dados ao sujeito, utilizando o contexto de segurança criado quando o sujeito tentou acessar o primeiro SP.

Uma variação prevista pelo perfil *Web SSO* é a possibilidade de o IdP enviar uma resposta, como no passo 5 do cenário descrito acima, sem a solicitação do SP, para um posterior acesso do sujeito. No cenário anterior, diz-se que ocorreu a autenticação única iniciada pelo SP. Nessa outra variação, ocorreu a autenticação única iniciada pelo IdP.

O perfil *Enhanced Client or Proxy* (ECP) é um perfil para autenticação SSO similar ao *Web SSO*, mas nesse caso o mecanismo de transporte de mensagens usado é o PAOS, ou "SOAP invertido", que permite participação mais ativa do agente do sujeito [OASIS 2005f].

Encerramento Único de Sessão

Quando um sujeito se autentica em um IdP, este último pode estabelecer uma sessão com o sujeito (p.e. por meio de um *cookie*). Além disso, quando o sujeito acessa um SP, este consome asserções emitidas pelo IdP e pode também estabelecer uma sessão com o sujeito. Em um dado momento, o sujeito pode decidir finalizar sua sessão com um SP específico ou pode finalizar todas as sessões de uma só vez. Para o segundo caso, a SAML define o perfil de encerramento único de sessão (*Single log-out* - SLO), baseado no protocolo de SLO [OASIS 2005f].

A Figura 1.5 [OASIS 2005f] ilustra o funcionamento do perfil SLO. Em um dado

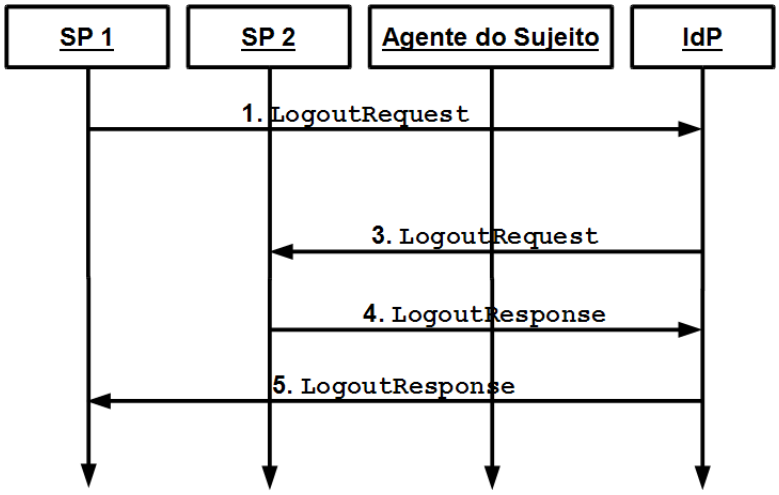


Figura 1.5: Exemplo de funcionamento do perfil SLO

momento, o sujeito indica para o SP que deseja encerrar todas as sessões. O SP em questão enviará uma mensagem `LogoutRequest` (definida no protocolo SLO) para o IdP responsável pela autenticação do sujeito, diretamente ou através do agente do sujeito (p.ex. redirecionamento HTTP no navegador) (etapa 1 da figura). O IdP, então, identifica todos os participantes da sessão do sujeito (etapa 2), que podem ser outros SPs e até outros IdPs que, por sua vez, coordenam sessões de outros SPs. Depois, o IdP envia uma mensagem `LogoutRequest` para esses participantes (etapa 3). Os participantes encerram a sessão localmente e devolvem ao IdP mensagens `LogoutResponse` (etapa 4). O IdP, por sua vez, retorna uma mensagem `LogoutResponse` para o SP que iniciou a desconexão usando o mesmo mecanismo da requisição (diretamente ou via agente do sujeito).

Perfis de atributos

A SAML [OASIS 2005f] define um perfil básico de atributos, que descreve o uso de tipos *XML Schema* como valores de atributos, e perfis que descrevem mapeamentos de formatos e conjuntos de atributos de outras especificações para o formato de atributos do SAML.

Os perfis para mapeamento de atributos permitem o uso de bases de atributos já implantadas para gerar declarações SAML de atributos sem a necessidade de definir os atributos manualmente [OASIS 2005f]. Os mapeamentos incluem transformações de atributos de bases X.500/LDAP [ITU-T 2001, Hodges e Morgan 2002], UUIDs [Leach et al. 2005] e atributos XACML [OASIS 2005c].

SAML e Gerenciamento de Identidades Federadas

A SAML provê suporte a diversas formas de estabelecimento e gerenciamento de identidades federadas [OASIS 2005g]. De maneira geral, a SAML permite que quaisquer mecanismos externos sejam usados para relacionar as diversas identidades de um sujeito.

Por exemplo, duas aplicações podem usar uma sincronização de base de dados para conectar usuários que possuam o mesmo nome de login, ou o mesmo `e-mail`. Esta é a única opção que as versões 1.0 e 1.1 da SAML oferecem suporte.

A SAML 2.0, por outro lado, provê suporte ao uso de pseudônimos, que são identificadores dinâmicos e não relacionados à atributos de identidade do sujeito. Por trás do uso dos pseudônimos estão dois protocolos SAML: o protocolo de gerenciamento de identificador de nome e o protocolo de mapeamento de identificador de nome, bem como os protocolos de transporte e perfis associados. Os pseudônimos servem como identificadores compartilhados entre SP e IdP e podem ser usados de duas formas: persistente e transiente [OASIS 2005g].

O *pseudônimo persistente* é criado uma única vez no IdP e associado permanentemente à identidade do sujeito. Ao receber uma asserção de autenticação contendo o pseudônimo, o SP cria um registro local para o sujeito usando o pseudônimo e outras informações de identidade do sujeito. Em acessos posteriores do mesmo sujeito, o SP receberá o pseudônimo que foi registrado e poderá decidir a autorização com base em dados locais juntamente com informações vindas do IdP e até mesmo poderá requisitar atributos associados ao pseudônimo em uma autoridade de atributos ligada ao IdP. Esse esquema, aliado a políticas de privacidade no acesso a atributos do sujeito, pode garantir a proteção da identidade, no entanto o acesso a diferentes SPs podem ser rastreados, caso esses SP atuem em conluio [OASIS 2005g].

O *pseudônimo transiente* é criado pelo IdP e associado à identidade do sujeito pelo tempo que durar o contexto de segurança. Dessa forma, o SP ainda pode decidir o acesso do sujeito com base em atributos emitidos pelo IdP, mas não pode mais manter informações sobre o sujeito que persistam por mais de uma sessão. Além disso, SPs diferentes não podem correlacionar acessos do mesmo sujeito, garantindo um certo nível de anonimato [OASIS 2005g].

1.4.2. Shibboleth

O projeto *Shibboleth* [Scavo e Cantor 2005] foi uma iniciativa do consórcio americano Internet2 que teve como principal objetivo lançar uma implementação de código aberto, baseada em padrões abertos, para tratar desafios relacionados ao gerenciamento de identidades e controle de acesso em instituições acadêmicas. Em 2003, foi liberada a versão 1.0 e, atualmente, o projeto *Shibboleth* está em sua segunda versão, liberada em 2008. O atual desenvolvimento do *Shibboleth* visa que este seja uma solução genérica para o gerenciamento de identidades federadas, podendo ser adotada por qualquer tipo de organização.

O pacote de *software* desenvolvido está fundamentado sobre padrões abertos como o XML e *Security Assertion Markup Language* (SAML) (ver Seção 1.4.1) e provê uma forma fácil para que aplicações *web* usufruam das facilidades providas pelo modelo de identidades federadas, como o conceito de autenticação única (*Single Sign-On* – SSO) e a troca segura de atributos dos usuários por todos provedores de serviços que compõem a federação. O *Shibboleth* tem como ênfase a privacidade dos atributos dos usuários, sendo que a liberação desses atributos para os provedores de serviços está condicionada a política de privacidade da instituição de origem do usuário e também as preferências

pessoais deste usuário.

Dentro de um domínio *Shibboleth*, existem dois papéis: provedor de identidades (*Identity Provider* – IdP) e provedor de serviços (*Service Provider* – SP). O primeiro é responsável por autenticar seus usuários, antes que estes possam usufruir dos serviços oferecidos pelo segundo. O ponto comum entre estes papéis é que ambos devem implementar toda a pilha de *software* fornecida pelo projeto *Shibboleth*, permitindo assim o transporte das credenciais dos usuários do provedor de identidades até o provedor de serviços [Scavo e Cantor 2005]. No *Shibboleth*, o processo de autenticação sempre é executado na instituição de origem do usuário, através de seu provedor de identidades, fazendo uso dos mecanismos de autenticação presentes nessa instituição. A autenticação de usuários pode ser feita através de nome de usuário e senha, Kerberos, X.509, etc [Chadwick 2009].

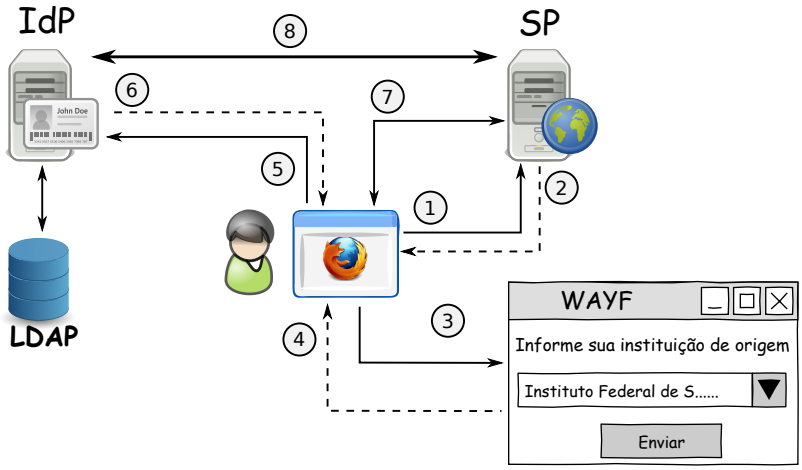


Figura 1.6: Interação de um usuário com um provedor de serviço *Shibboleth*

A Figura 1.6 ilustra os passos para que um usuário, ainda não autenticado, consiga acessar um recurso disponibilizado por um provedor de serviços *Shibboleth*. O usuário, através de seu navegador, aponta para a `URL` da aplicação *web* desejada (passo 1). Este pedido é interceptado pela pilha *Shibboleth*, que verifica que o mesmo não está autenticado e assim deve encaminhá-lo ao provedor de identidades do respectivo usuário para que passe pelo processo de autenticação.

O provedor de serviços desconhece o provedor de identidades do usuário e assim este usuário é encaminhado ao serviço *Where Are You From* (WAYF), através de um redirecionamento HTTP, que permite ao usuário selecionar sua instituição de origem (passo 2). No WAYF são listados somente os provedores de identidades tidos como confiáveis pelos provedores de serviços da federação *Shibboleth*. Após o usuário selecionar sua instituição de origem, este é encaminhado, novamente através de redirecionamentos HTTP, ao provedor de identidade (IdP) de sua instituição (passo 4).

Neste ponto, o usuário realiza o processo de autenticação (passo 5) de acordo com o mecanismo de autenticação presente em sua instituição, podendo tal processo ser realizado através de nome de usuário e senha, estes armazenados em uma base LDAP. Uma vez que o processo de autenticação tenha sido realizado com sucesso, o IdP cria um iden-

tificador aleatório para este usuário o qual persistirá no IdP durante toda a sessão deste usuário. Por fim, o usuário é redirecionado ao recurso *web* desejado (passo 6) e, juntamente com a requisição HTTP ao provedor de serviços (SP), é enviado seu identificador aleatório através de asserções SAML de autenticação.

Cabe ao SP aplicar o controle de acesso, considerando os atributos do usuário que foram fornecidos com essa tentativa de acesso (passo 7). O provedor de serviços pode requisitar outros atributos do usuário junto ao provedor de identidades do usuário para usar no mecanismo de controle de acesso (passo 8), como por exemplo número do registro acadêmico, endereço de *e-mail*, etc. Os provedores de identidades respeitam um conjunto de políticas para revelação de atributos de seus usuários, preservando assim a privacidade dos mesmos. O *Shibboleth* define uma forma padronizada para troca de atributos, através de asserções SAML, porém não especifica como deverão ser tais atributos, deixando tal função livre para os desenvolvedores.

Em um ambiente federado, a padronização destes atributos é fundamental, para que provedores de serviços saibam quais atributos poderão requisitar e para que provedores de identidades saibam quais atributos deverão fornecer. Em [Internet2 2008, Wahl 1997, Smith 2000], foi proposto o `eduPerson`, um conjunto padrão de atributos de identidade comuns para federações acadêmicas, sendo que 6 atributos são altamente recomendados, 10 são sugeridos e 25 são opcionais. A comunidade federada CAFe[5] que reúne instituições acadêmicas brasileiras, além de implementar esse conjunto de atributos, definiu ainda seu próprio conjunto de atributos (chamado `brEduPerson`).

Relações de confiança entre provedores de identidades e de serviços garantem que os provedores de serviços terão certeza que um usuário foi autenticado corretamente junto ao provedor de identidades em questão e que este último fornecerá corretamente os atributos relacionados a este usuário. As asserções emitidas pelo provedor de identidades são assinadas por este, o que permite que o provedor de serviços verifique a autenticidade das mesmas [Chadwick 2009].

Para provedores de serviços cujo público alvo não se restringe exclusivamente à comunidade acadêmica, a implementação da pilha *Shibboleth* pode ser um impeditivo, analisando custos *vs* benefícios. É importante ressaltar ainda que, mesmo partindo do pressuposto de que as relações de confiança já estejam previamente estabelecidas entres os diversos domínios administrativos que representam uma federação, ainda assim, há diversos desafios para transpor as credenciais de autenticação em uma federação, pois provedores de serviços possuem autonomia para decidir quais políticas e tecnologias de segurança utilizar, ou seja, uma federação precisa prover uma infraestrutura que suporte a autenticação SSO mesmo diante de parceiros que usem credenciais de segurança diferentes daquelas usadas no *Shibboleth*.

1.4.3. Projeto Liberty Alliance

O projeto *Liberty Alliance* surgiu como um consórcio formado por empresas de diferentes áreas como telecomunicações, bancos, universidades, etc. com o intuito de criar um conjunto de especificações abertas voltadas para o gerenciamento de identidades federadas

[5] `http://www.cafe.rnp.br`

e sua integração com Serviços *Web* [Liberty 2003]. Atualmente mais de 160 organizações privadas, sem fins lucrativos e governamentais fazem parte do projeto *Liberty Alliance* e um dos principais pontos positivos do projeto é sua influência em padrões como o SAML, cujas extensões propostas pela *Liberty Alliance* são hoje parte do SAML 2.0 [Baldoni 2010]. Os principais objetivos do projeto são [Liberty 2003]:

- Permitir aos usuários garantir a privacidade e a segurança de suas informações pessoais;
- Prover um padrão aberto para permitir uma única autenticação (SSO), o que inclui a autenticação descentralizada e a autorização em múltiplos provedores de serviços;
- Prover especificações compatíveis com uma grande variedade de dispositivos;
- Utilizar em suas especificações, padrões e protocolos existentes e amplamente aceitos;
- Prover meios para que as empresas respeitem os requisitos de segurança e a privacidade dos clientes.

O arcabouço proposto para o gerenciamento de identidades é composto por três componentes principais: *Identity Federation Framework* (ID-FF), *Identity Web Services Framework* (ID-WSF) e *Identity Services Interface Specifications* (ID-SIS).

O ID-FF visa permitir o gerenciamento de identidades federadas através de diversos domínios administrativos, caracterizando o *círculo de confiança*. Um *círculo de confiança* é formado entre organizações através de relações de confiança e acordos comerciais. Dentro de um *círculo de confiança* cada organização pode assumir papéis como provedores de identidades (IdP), como provedores de serviços (SP) ou como ambos [Kallela 2008]. Assim como no *Shibboleth* (ver Seção 1.4.2), o IdP gerencia e autentica os usuários e o SP aceita as informações sobre a identidade de usuários oriundas do IdP de seu *círculo de confiança*.

O ID-FF é formado por três componentes: **redirecionamento HTTP**, que permite que as informações dos usuários, que façam uso de navegadores *web*, sejam trocadas entre o IdP e os SP; **Serviço *Web***, que permite a comunicação entre as entidades do sistema através de trocas de mensagens SOAP; e **metadados e esquemas XML**, os quais indicam como as diversas informações sobre o usuário serão trocadas entre IdPs e SPs [Kallela 2008]. No ID-FF também são descritos mecanismos para ligação entre diferentes contas ou identidades, visando garantir a privacidade e anonimato dos usuários; autenticação única (SSO); e mecanismos para o gerenciamento de sessões de forma simplificada.

O ID-WSF apresenta modelos para a criação de serviços que rodem sobre o arcabouço da *Liberty Alliance*, além de especificar um serviço para a localização de provedores de identidades e mecanismos de segurança. As principais características do ID-WSF são:

- **Permissão baseada no compartilhamento de atributos** – Permite que empresas ofereçam serviços personalizados para os clientes, de acordo com os atributos e preferências que estes clientes escolheram compartilhar;
- **Descoberta do Serviço de Identidades** – Para obter maiores informações sobre a identidade de um cliente, os provedores de serviços podem utilizar do Serviço de Descoberta para encontrar um Serviço de Identidades específico de um cliente;
- **Serviço de Interação** – Um Serviço de Identidades, antes de fornecer as informações de um usuário a um serviço, deve obter a permissão deste usuário. São definidas especificações e protocolos que permitem a interação entre os Serviços de Identidades e os serviços que estão solicitando as informações;
- **Perfis de segurança** – Define perfis e requisitos de segurança para a descoberta e o uso dos Serviços de Identidade;
- **Modelos de Serviços de Identidades** – Provê modelos para a implementação de Serviços de Identidade sobre a ID-WSF.

O componente ID-SIS, que estende o ID-FF e o ID-WSF, provê um conjunto versátil de ferramentas para a construção de serviços interoperáveis sobre o ID-WSF. Tais especificações foram definidas para permitir que organizações possam facilmente, criar ou estender serviços sobre a estrutura do ID-WSF. Uma das especificações propostas foi a *ID-Personal Profile*, que define um serviço para obter informações pessoais de um usuário como nome, endereço, telefone, etc. Tal especificação permite que todas organizações, que estejam de acordo com a *Liberty Alliance*, possuam um conjunto de campos e valores conhecidos, tendo assim um dicionário e uma linguagem padrão para que possam interagir entre si.

Os *identificadores opacos* ou *pseudônimos* foram propostos nas especificações da *Liberty* com o intuito de garantir a privacidade dos usuários dos serviços. Para cada provedor de serviços, o provedor de identidades poderá atribuir diferentes pseudônimos relacionados a um mesmo usuário. Dessa forma, o mesmo usuário será representado por diferentes pseudônimos para cada serviço que for acessar, garantindo assim a proteção contra o rastreamento de suas transações. Identificadores opacos permitem aos provedores de serviços identificar quem são seus clientes, relacionando em suas contas locais, porém não possibilita que os provedores de serviços obtenham informações pessoais dos clientes de forma que possa comprometer a privacidade do mesmo.

1.4.4. Especificações WS-Trust e WS-Federation

A especificação *WS-Security* [OASIS 2004] define mecanismos para garantir a integridade e a confidencialidade de mensagens SOAP, bem como o uso de vários tipos de credenciais de segurança que visam declarar informações de segurança (*claims*). A especificação *WS-Trust* [OASIS 2009b] estende a *WS-Security* com a definição de um protocolo para a troca e disseminação de credenciais de segurança entre diferentes domínios, bem como meios para se verificar se uma credencial é ou não confiável. Dessa forma, as partes envolvidas podem detectar e estender relações de confiança baseadas na emissão e validação de credenciais.

A base do modelo de confiança *WS-Trust* é o serviço de *tokens* de segurança (*security token service* - STS) [OASIS 2009b]. O STS é um Serviço *Web* que implementa uma interface WSDL padrão, que define operações para emissão, renovação, validação e revogação de credenciais. Se duas aplicações de diferentes domínios desejam se comunicar e não possuem uma relação de confiança direta (i.e., as credenciais emitidas em um domínio não são aceitas no outro), estas podem usar um STS de confiança de ambos os domínios para emitir credenciais em nome das aplicações. Dessa forma, o STS serve como mediador de confiança entre domínios de segurança distintos.

De maneira geral, o modelo de confiança da *WS-Trust* [OASIS 2009b] especifica que um serviço *Web* pode exigir que uma mensagem comprove um conjunto de declarações (*claims*) para que seja processada. Essas exigências se traduzem em um conjunto de credenciais de segurança suportadas pelo serviço *Web* e um conjunto de STSs aos quais o serviço confia a emissão dessas credenciais. Para dar dinamismo e interoperabilidade ao serviço, deve-se descrever as exigências de segurança na forma de documentos de políticas em um formato padronizado, como *WS-Policy* [W3C 2007], e disponibilizar essas políticas anexadas à descrição WSDL do serviço.

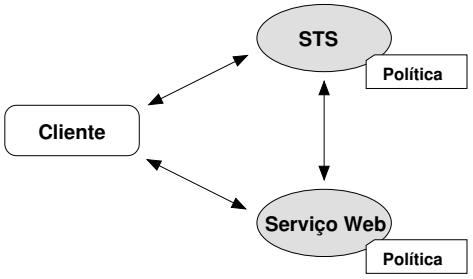


Figura 1.7: Modelo de confiança da *WS-Trust*

A Figura 1.7 ilustra um cenário de mediação de confiança com base em uma terceira parte confiável que implementa um STS [OASIS 2009b]. Quando o cliente deseja acessar o serviço, primeiramente obtém a política desse serviço (passo 1). Por meio dessa política, o cliente descobre o conjunto de exigências do serviço e os STSs de confiança. Caso o cliente não possua credenciais suficientes para satisfazer às exigências, pode pedir as credenciais a um dos STSs listados (passo 2). No entanto, o STS também poderá conter uma política especificando um outro conjunto de exigências e também pode apontar para outros STSs nos quais confie. Caso o cliente possua credenciais suficientes para acessar o STS de confiança do serviço, este pode requerer ao STS que emita as credenciais exigidas. O STS emitirá as credenciais e o cliente apresentará estas juntamente com a requisição ao serviço *Web* (passo 3). O serviço realizará a validação das credenciais passadas pelo cliente, possivelmente invocando o STS que as emitiu (passo 4), para poder permitir o acesso do cliente.

As mensagens trocadas pelo STS são bastante gerais para permitir extensões e composições futuras, cabendo a cada operação (emissão, validação, renovação, revogação, etc.) especificar quais elementos a mensagem deve conter. A mensagem de requisição é composta de um elemento XML `RequestSecurityToken` (RST) e a resposta é formada pelo `RequestSecurityTokenResponse` (RSTR). A RST contém parâmetros gerais, como o tipo de requisição, o tipo de credencial ao qual a requisição se refere, o

serviço ao qual a credencial será apresentada e as exigências desse serviço. Dependendo do tipo da requisição, esta conterá elementos mais específicos, como uma prova de posse de uma chave privada, conteúdo aleatório para geração de chaves, prazo de validade esperado para as credenciais, etc. A RSTR contém, basicamente, a credencial requisitada juntamente com parâmetros da credencial (tipo, validade, contexto de utilização, etc.) descritos de forma independente do tipo de credencial.

WS-Federation

A especificação *WS-Federation* [W3C 2009a] é um padrão OASIS que define mecanismos baseados nos padrões WS-*, em especial *WS-Security*, *WS-Trust* e *WS-Policy*, para a formação de federações. Esses padrões já definem as bases para o gerenciamento federado de identidades, porém a *WS-Federation* propõe extensões que definem como combinar esses modelos de forma a prover funcionalidades mais ricas aos domínios de segurança dentro e entre federações.

Há alguma sobreposição entre as funcionalidades da *WS-Federation* e da tecnologia *SAML* (ver seção 1.4.1) [W3C 2009a]. Ambas soluções permitem gerenciamento federado de identidades e provêm um conjunto de funcionalidades similar. Assim como a SAML, a *WS-Federation* suporta autenticação única, encerramento único de sessão, compartilhamento de atributos (sujeito a políticas de privacidade), pseudônimos permanentes e transientes e documentos de metadados. A principal diferença é que a *WS-Federation* está baseada no modelo de confiança da *WS-Trust* e permite o uso de quaisquer credenciais de segurança, não somente asserções SAML.

A *WS-Federation* define serviços para o gerenciamento de identidades como implementações da interface do STS, definida na *WS-Trust* [W3C 2009a]. Dessa forma, o STS passa a acumular a função de provedor de identidades (*identity provider* - STS/IdP) e a emitir credenciais contendo informações de identidades dos sujeitos do seu domínio de segurança para serem usadas em outros domínios que tiverem uma relação de confiança (i.e. aceitarem credenciais assinadas pelo STS/IdP). As relações de confiança podem ser estabelecidas diretamente, pelo intermédio de um STS/IdP, ou indiretamente por meio de relações de confiança entre STS/IdPs de domínios distintos. A Figura 1.8 apresenta algumas topologias de mediação de confiança previstas na *WS-Federation*, bem como as trocas de mensagens em cenários de acesso a recursos.

No cenário A, o sujeito se autentica no STS/IdP do seu domínio (passo 1, por meio da operação RST) e, em seguida, realiza o acesso aos recursos, enviando junto a credencial emitida (passo 2). O recurso verifica a validade da credencial enviando um pedido de validação para o STS/IdP do seu próprio domínio (passo 3). Como os STS/IdPs do cliente e do serviço possuem uma relação de confiança, a validação terá sucesso e o provedor do recurso poderá decidir sobre o acesso com base na validação das informações contidas na credencial. O cenário B é similar, no entanto, os STS/IdP dos domínios não possuem uma relação direta de confiança, mas uma relação mediada por um terceiro STS/IdP. Além disso, o cliente, antes de tentar acessar o recurso, invoca o STS/IdP do domínio do provedor (passo 2) para que este lhe indique as credenciais aceitas pelo provedor do recurso. O cenário C apresenta uma comunicação direta entre aplicações distribuídas, na qual o

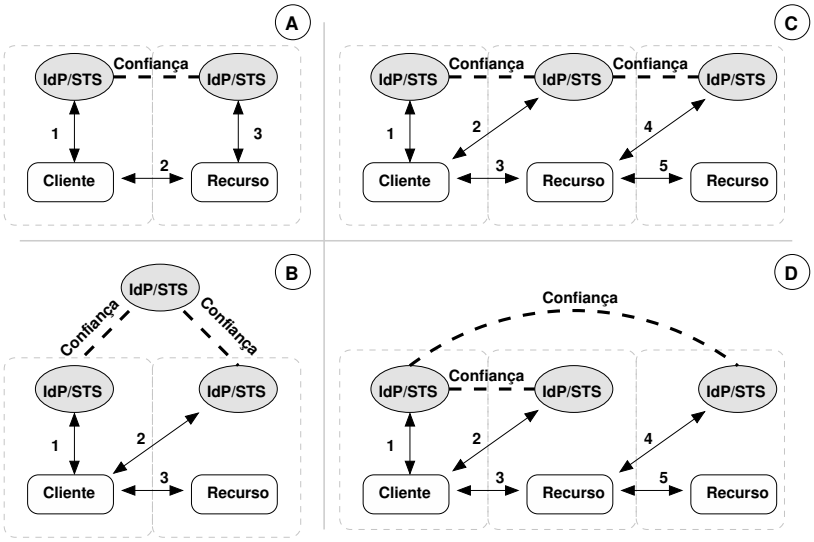


Figura 1.8: Cenários de federações com diferentes topologias de confiança

serviço acessado pelo cliente realiza, por sua vez, o acesso a um segundo recurso de um domínio que possua relação de confiança. O cenário D é similar ao C, no entanto nesse caso o cliente deve enviar ao recurso acessado no passo 3 as credenciais para que esse possa acessar o segundo recurso. A delegação foi necessária pois não há uma relação de confiança entre os domínios dos dois recursos.

Além do provedor de identidades, a *WS-Federation* prevê ainda os papéis de serviço de atributos e serviço de pseudônimos [W3C 2009a]. O serviço de atributos é responsável por emitir atributos de identidade dos sujeitos. A especificação não obriga o uso de nenhuma interface específica para o serviço de atributos, no entanto recomenda para isso o uso do STS, pois o uso do mesmo modelo de confiança e protocolo de comunicação pode facilitar análises de ameaças e a implementação. Além disso, o uso do STS facilita que um mesmo serviço implemente tanto o IdP quanto o servidor de atributos. Os serviços de atributos devem garantir a privacidade das informações dos sujeitos, portanto, a *WS-Federation* prevê que diferentes atributos devam ser associados a diferentes políticas de acesso, de acordo com o consentimento dos sujeitos.

O serviço de pseudônimos é responsável por associar pseudônimos a identidades de sujeitos. Um pseudônimo é um tipo especial de atributo, usado para identificar um sujeito sem revelar diretamente as informações de identidade. Na *WS-Federation*, pseudônimos podem ter diferentes níveis de volatilidade para permitir diferentes níveis de personalização e privacidade [W3C 2009a]. Por exemplo, um sujeito pode ter um pseudônimo diferente para cada provedor de serviço, o que dificulta o seu rastreamento (exceto em caso de conluio entre os serviços) mas ainda permite aos serviços manter informações locais para fins de personalização de acesso. Em outro exemplo, o sujeito pode ter pseudônimos que durem apenas uma sessão de autenticação, aumentando sua privacidade e impedindo que serviços associem qualquer informação persistente. Diferente do IdP e do serviço de atributos, o serviço de pseudônimos usa uma interface diferente, não baseada no STS, mas sim na especificação *WS-Transfer* [W3C 2010], que define métodos para criação, remoção, alteração e obtenção de recursos (no caso, pseudônimos).

Os serviços de atributos, de pseudônimos e IdPs podem trabalhar em conjunto a fim de prover funcionalidades complexas garantindo a privacidade dos sujeitos. Por exemplo, o serviço de atributos pode responder a pedidos de atributos associados a um pseudônimo, para isso invocando o serviço de pseudônimos e obtendo a identidade associada ao pseudônimo. Além disso, esses serviços podem ser implementados no mesmo sistema [W3C 2009a].

Assim como a SAML tem um perfil para clientes capazes de processar mensagens SOAP, a *WS-Federation* tem um perfil para clientes *seb* (i.e. navegadores *web* sem suporte a Serviços *Web*) [W3C 2009a]. O perfil tira a necessidade de o cliente processar mensagens SOAP usando, para isso, mecanismos do protocolo HTTP. Todas as trocas de mensagens são feitas usando somente métodos POST e GET e redirecionamentos são usados para automatizar comunicações entre o STS e a aplicação *web*. Esse perfil permite que navegadores *web* aproveitem os modelos de segurança da *WS-Security* e *WS-Trust*.

Para facilitar a descoberta de serviços e a comunicação entre os participantes da federação, a *WS-Federation* define um formato XML para especificação de metadados da federação [W3C 2009a], que contém descrições de uma ou mais entidades participantes. Essas descrições especificam mecanismos para a descoberta de serviços federados e de metadados (WSDL, políticas, etc.) associados a estes. O formato XML é baseado no modelo de documentos de metadados da especificação SAML [OASIS 2005e] (ver Seção 1.4.1), que associa papéis a entidades. A estrutura do documento é similar, no entanto os papéis que as entidades podem assumir são diferentes e refletem as funcionalidades e os parâmetros dos serviços da *WS-Federation*. A especificação define, ainda, diversos mecanismos para a obtenção de documentos de metadados de maneira segura. Entre os mecanismos estão a incorporação dos documentos em descrições WSDL, uso de URL padrão e resolução de registros DNS.

1.4.5. OpenID

Em 2005, Brad Fitzpatrick, arquiteto-chefe da Six Apart, desenvolveu a tecnologia OpenID [OpenID 2010] para ser utilizada no LiveJournal com o objetivo de autenticar os comentaristas do *blog* da comunidade e evitar o *spam* de comentários. *OpenID Authentication 1.0* começou como um protocolo leve de autenticação de identificadores de usuários (URLs) baseado no HTTP. Conforme [Maler e Reed 2008], a solução OpenID, que fornece ao usuário a possibilidade de ter uma única credencial (um OpenID) para acessar diferentes sítios *web*, está evoluindo rapidamente.

Atualmente, a tecnologia *OpenID Authentication 2.0* é uma plataforma aberta conduzida pela comunidade de desenvolvedores organizada pela *OpenID Foundation*, que permite e incentiva a inovação. Esta suporta tanto URLs[6] quanto XRIs (*Extensible Resource Identifier*)[OASIS 2005d] como identificadores de usuários, acrescenta mais segurança e suporta tanto identificadores públicos quanto privados. Esta última versão, além de adotar o modelo de gerenciamento de identidades centrado no usuário, implementa o conceito de identidade federada [Recordon e Reed 2006].

Uma vasta gama de sítios *web*, especialmente os que tem forte conteúdo gerado

[6]Um exemplo de um identificador do tipo URL é `http://openid-provider.appspot.com/wangham`

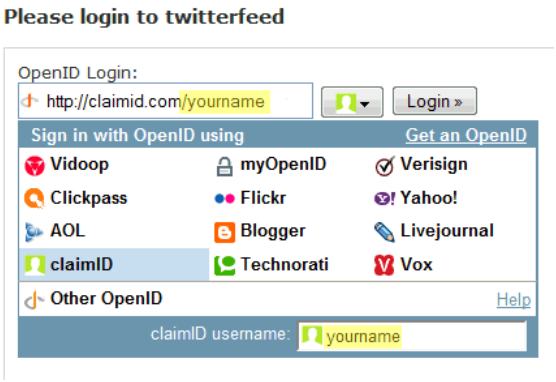


Figura 1.9: Sítio *web* com suporte a *login* OpenID

pelo usuário, adotaram esta tecnologia (usam ou provêm OpenIDs), entre estes destacam-se Google, Six Apart, Yahoo, Flickr, MySpace.com, Facebook, Wordpress, Verisign, AOL e PayPal. Formada em junho de 2007, a *OpenID Foundation* é uma organização internacional sem fins lucrativos constituída de indivíduos e empresas comprometidas com o suporte, promoção e segurança da tecnologia OpenID. A fundação atua como uma organização de confiança do público que representa a comunidade aberta de programadores, fornecedores e usuários [OpenID 2010].

A solução OpenID é descentralizada e nenhuma autoridade central aprova ou registra as partes confiantes (*relying parties*), sítios *web*, ou provedores OpenID (*OpenID providers*). Um usuário pode escolher livremente qual provedor OpenID usará (Google, Yahoo, etc) e pode preservar seu identificador caso deseje futuramente mudar de provedor OpenID (ver Figura 1.9). A distribuição de um identificador OpenID é gratuita e não é necessário qualquer tipo de registro ou aprovação de qualquer organização [Baldoni 2010].

Segundo a especificação [OPENID 2007], o *framework* utiliza apenas pedidos e respostas HTTP, por isso não exige nenhuma capacidade especial do *software* cliente (*User-Agent*), no caso um navegador *web*. O *framework* OpenID não está vinculado à utilização de *cookies*, ou a utilização de qualquer outro mecanismo específico da parte confiante (*relying party*) ou depende da utilização de gerenciamento de sessão do provedor OpenID. Extensões para os navegadores Web podem simplificar a interação com o usuário final, porém não são obrigatórios.

A seguir, uma visão geral das trocas de mensagens do protocolo OpenID 2.0 é descrita e também é ilustrada na Figura 1.10 [OPENID 2007].

1. Através do seu navegador Web, o usuário inicia o processo de autenticação apresentando um identificador para a parte confiante (o sítio *web* que deseja acessar e que suporta *login* OpenID);

2. Com base no identificador fornecido pelo usuário, a parte confiante realiza a descoberta (*Discovery*) da URL do provedor OpenID que o usuário utiliza para autenticação;

3. (Opcional e não ilustrado na Figura) A parte confiante e o provedor OpenID esta-

belecem uma associação - uma chave secreta é estabelecida utilizando o protocolo Diffie-Hellman Key Exchange. Esta associação é estabelecida para facilitar o processo de assinatura e verificação de mensagens trocadas entre o provedor OpenID e o a parte confiante.

4. A parte confiante redireciona o navegador do usuário para o provedor OpenID com um pedido de autenticação OpenID (solicitação de uma asserção ou credencial);

5. O provedor OpenID verifica se o usuário final está autorizado a executar a autenticação OpenID que deseja fazê-lo. A maneira na qual o provedor OpenID autentica o usuário final e as políticas em torno desta autenticação estão fora do escopo da especificação (p.ex: no caso do Google, é utilizada a senha associada a conta google como forma de autenticação);

6. O provedor OpenID redireciona o navegador Web do usuário final de volta para a parte confiante com uma asserção que indica que a autenticação está aprovada (asserções positivas) ou uma mensagem de que a autenticação falhou (asserções negativas);

7. A parte confiante verifica as informações enviadas pelo provedor OpenID, que inclui a verificação da URL retornada, das informações descobertas (provedor OpenId), do *nonce* e da assinatura. Para verificação da assinatura, a parte confiante usa a chave compartilhada estabelecida durante a associação ou a solicita diretamente ao provedor OpenID.

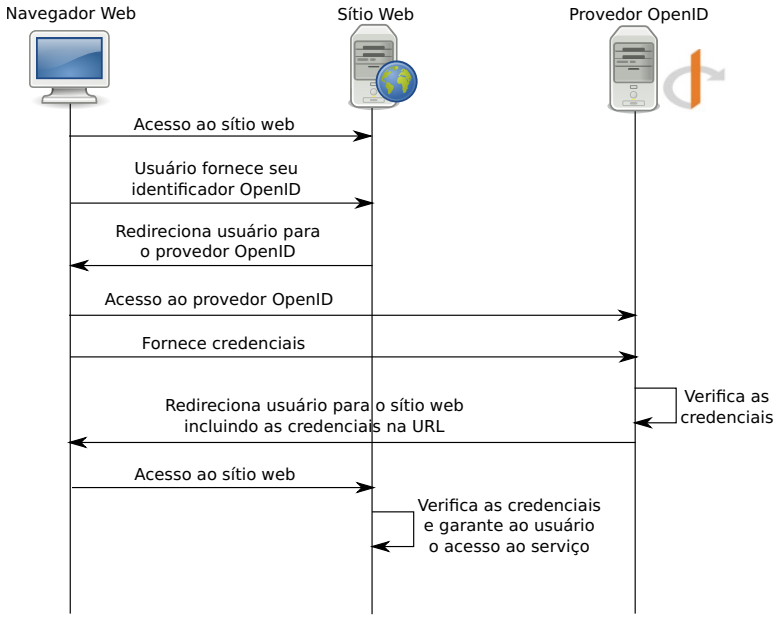


Figura 1.10: Protocolo OpenID Authentication 2.0

Conforme citado no passo 2 do protocolo OpenID 2.0, a descoberta do provedor OpenID se dá quando os usuários fornecem seus identificadores às partes confiantes. Diante disto, provedores OpenID e partes confiantes que não se conhecem podem

se comunicar com sucesso, um tipo de escalabilidade conscientemente modelada dentro dos padrões da própria Web. De acordo com [Maler e Reed 2008], isso pode representar um desafio a privacidade: como uma arquitetura tendenciosa e direcionada ao amplo compartilhamento de informações dos usuários. O OpenID permite e até estimula que diferentes partes confiantes correlacionem atividades de um usuário. No entanto, OpenID 2.0 suporta o login utilizando pseudônimos. O OpenID também permite que o provedor OpenID de um usuário veja todas as partes confiantes (sítios) que o usuário visitou. Para controlar a disseminação dessas informações para um provedor OpenID (de terceiros), a única opção do usuário seria a de executar o seu próprio provedor OpenID. O modelo de descoberta de provedores OpenID também impede uma autenticação única verdadeira, na qual a parte confiante pode visitar diretamente o provedor OpenID sem pedir para que o usuário indique a sua localização [Maler e Reed 2008].

1.4.6. Windows CardSpace (*InfoCard*)

De acordo com [Chappell 2006], nos diferentes tipos de redes colaborativas que utilizam a Internet diferentes tipos de identidades digitais são necessárias e a realidade é que estas identidades são providas por diferentes fontes (IdPs). Isto significa que a solução para o gerenciamento dessas identidades é utilizar sistemas que suportem múltiplas identidades, ou seja, um sistema de sistemas - um meta sistema (*metasystem*) - focado na identidade. O desafio é criar, usar, e gerenciar esta diversidade de identidades de uma forma efetiva.

O meta sistema Windows CardSpace, originalmente chamado de *InfoCard*, é um componente da plataforma .Net da Microsoft projetado para oferecer aos usuários uma experiência consistente do uso de múltiplas identidades digitais, a partir do uso de um agente (*user-agent*) especializado. A Microsoft documentou o protocolo implementado pelo Cardspace na especificação *InfoCard*. A tecnologia CardSpace está disponível por padrão no Windows Vista e no Windows 7 e pode ser incorporada em versões anteriores do sistema operacional Windows. Além disso, a mesma também é suportada no navegador Internet Explorer (desde a versão 7.0).

O Cardspace concentra-se nas coleções de dados do usuário chamados cartões de informação (*InfoCards*), apresentados em uma interface de software, chamado de seletor de identidade (semelhante a uma carteira que contém os cartões que identificam o usuário). Conforme ilustrado na Figura 1.11, cada *InfoCard* representa uma identidade diferente. Quando uma parte confiante (SP) solicita as credenciais do usuário, este escolhe, a partir do seletor, uma de suas identidades [Maler e Reed 2008].

A Figura 1.12 ilustra um cenário no qual um usuário, através de uma aplicação, por exemplo um navegador Web, tenta acessar a parte confiante B que suporta a tecnologia CardSpace. Este usuário também deve ser capaz de escolher, entre um grupo de provedores de identidades, quem será a fonte da identidade digital que este apresentará à parte confiante B. Seja qual for a escolha do usuário, as mensagens trocadas entre as partes serão[Chappell 2006]:

1. A aplicação obtém os requisitos do *token* de segurança da parte confiável que o usuário deseja acessar. Esta informação está contida na política da parte confiável e estas inclui, entre outras informações, qual tipo de *tokens* de segurança que a parte confiável pode aceitar e quais atributos (*claims*) este *token* deve conter;

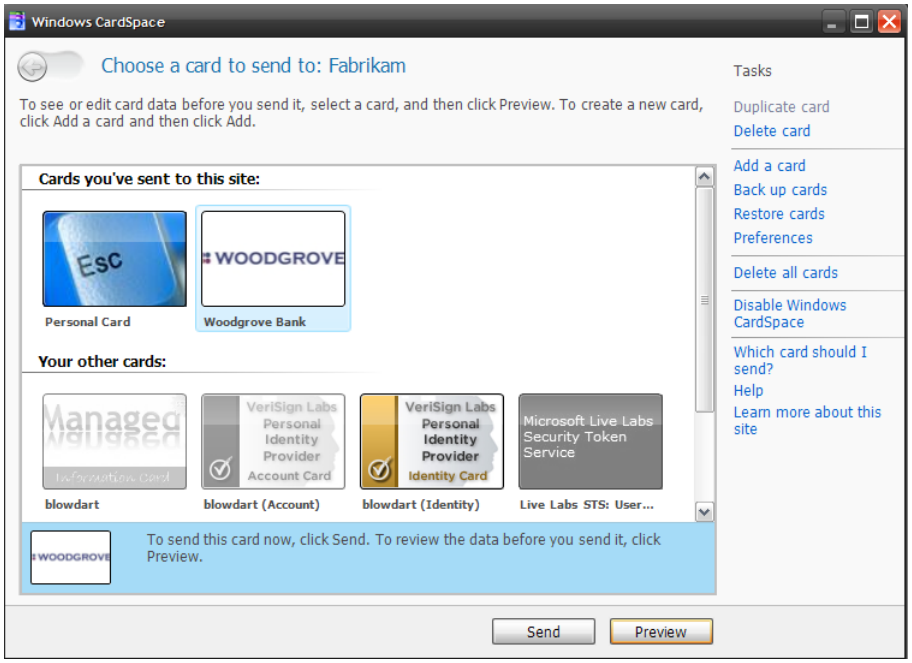


Figura 1.11: Visualização de um Seletor de *InfoCards*

2. Uma vez que estas informações são repassadas para o CardSpace, a aplicação mostra a tela do seletor de identidades, com os cartões de informação que o usuário possui, para que o usuário escolha qual identidade irá usar. Apenas alguns cartões são aplicáveis, os cartões cujos tokens de segurança e os atributos não são compatíveis com os requisitos da parte confiante, estes aparecem na tela esmaecidos e os usuários não podem escolhê-los;

3. Após selecionar o cartão desejado, o CardSpace emite um pedido de *token* de segurança para a o provedor de identidades associado com o cartão. O provedor de identidades então retorna um *token* de segurança;

4. Uma vez que o token é recebido, o CardSpace passa este para aplicação que apresenta para a parte confiante. A parte confiante pode então usar este token para autenticar o usuário ou para outros propósitos.

No passo 3, quando o provedor de identidades retorna o *token* de segurança assinado digitalmente, o conjunto de atributos do *token* corresponde à noção de uma asserção SAML, e, de fato, um dos tipos de token suportados é um *token* SAML. A tecnologia CardSpace suporta dois tipos de cartões [Maler e Reed 2008]:

- *cartões self-asserted* – representam um conjunto de atributos cujos valores são determinados unicamente pelo usuário (semelhantes as identidades OpenID). Na implementação da Microsoft, esses atributos são diretamente armazenados no dispositivo do usuário;

- *cartões administrados* – representam um conjunto (extensível) de atributos dos usuários gerenciado por um provedor de identidade. Tipicamente, cada vez que

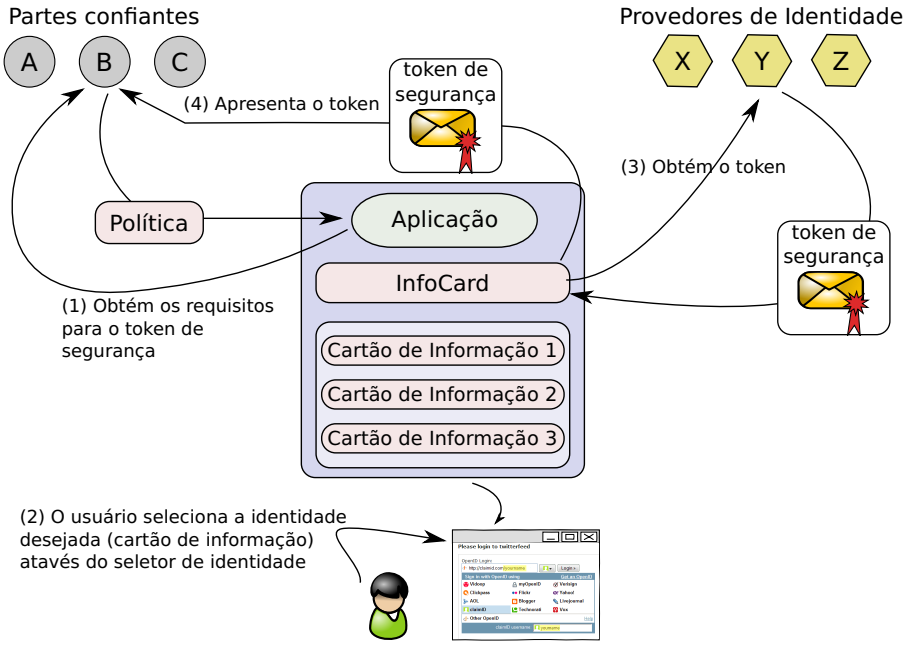


Figura 1.12: Mensagens Trocadas no Windows CardSpace

o usuário seleciona um cartão específico em resposta a um pedido de uma parte confiante, o seletor de identidades recupera os atributos do usuário no IdP que os emitiu. Cartões administrados são típicas identidades SAML-federadas nas quais o IdP rege os seus atributos e validade.

Implementação de tecnologias centradas no usuário impõem um custo, mas também permite soluções mais elegantes para problemas como o da descoberta do IdP. O modelo InfoCard aborda este problema eliminando a necessidade de uma parte confiante se conectar ao IdP. Em cartões *self-asserted*, o dispositivo do cliente pode ser o próprio IdP e, para cartões administrados, os IdPs armazenam seu endereço no cartão para o seletor de identidades usar [Maler e Reed 2008].

Um cartão administrado reflete a estreita relação do usuário com o IdP e um seletor de identidade pode usar isso para aumentar a resistência a ataques de *phishing* de autenticação. Como um intermediário nas comunicações entre IdPs e partes confiantes, um seletor de identidades também permite aos usuários evitar que um IdP identifique os sítios *web* e aplicações *web* visitadas (partes confiantes) pelos usuários. Vale destacar ainda que, um seletor de identidades ao permitir que um usuário selecione a identidade que deseja utilizar possibilita o gerenciamento centrado no usuário [Maler e Reed 2008].

Segundo [Chappell 2006], a tecnologia Windows CardSpace é totalmente agnóstica sobre o formato de token de segurança que é requerido por um provedor de identidade e que é passado para parte confiante. De fato, geralmente o CardSpace não tem consciência do que está dentro do *token* (formato). Por isso, o CardSpace pode trabalhar com qualquer sistema de identidade digital, utilizando qualquer tipo de token de segurança, incluindo simples *usernames tokens*, certificados X.509, *tickets* Kerberos, *tokens* SAML ou qualquer outro *token*. Isso permite que este meta-sistema de identidades possa ser usado junto com qualquer tecnologia de identidades digitais.

Todas as trocas implementadas pelo CardSpace e ilustradas na Figura 1.12, são feitas usando protocolos abertos e padronizados. No cenário mais geral, a política da parte confiante é descrita usando a *WS-SecurityPolicy*[OASIS 2009a], a política é recuperada usando a *WS-MetadataExchange* [W3C 2009b], um token de segurança é adquirido usando *WS-Trust*[OASIS 2009b], e o *token* é transmitido para a parte confiável usando *WS-Security* (todos estes protocolos da família WS são necessários para permitir a troca segura de *tokens* de identidade no meta-sistema de identidades).

1.4.7. Projeto Higgins

A motivação inicial do projeto Higgins foi o desejo de um sistema de gerenciamento de identidades centrado no usuário que permita que este tenha mais controle, comodidade e privacidade sobre a sua identidade e informações de perfil. As pessoas devem ser capazes de decidir quais informações desejam compartilhar e com quais sítios *web*. Higgins trata-se de um *framework* que opera com todos os protocolos de identidade digital, incluindo WS-Trust, OpenID, SAML, XDI, LDAP, entre outros. O objetivo é criar uma identidade única de todas as outras identidades de diferentes domínios [EclipseFoundation 2010].

O *framework* Higgins define uma série de interfaces de programação que os desenvolvedores podem usar para ligar o seu software à função de gerenciamento de identidades Higgins [Le e Bouzefrane 2008]. O projeto Higgins tem recebido contribuições tecnológicas da IBM, Novell, Oracle, CA, Serena, Google, Corisecio, bem como de várias outras empresas e indivíduos. O projeto Higgins aborda cinco áreas [EclipseFoundation 2010]:

1. Proporciona uma experiência consistente ao usuário, baseado em cartões de informação chamados de *i-Card*, para o gerenciamento e divulgação dos seus dados de identidade;

2. Permite aos usuários um maior controle sobre a revelação de suas informações pessoais com os sítios *web* que interagir;

3. Fornece uma API e um modelo de dados para prover identidades federadas, a partir de uma ampla variedade de fontes. Através desta API, o projeto Higgins incentiva os desenvolvedores a criar *plugins* para operar com protocolos e *tokens* de segurança de sistemas legados;

4. Fornece *(plugins*) para que fontes de dados existentes, incluindo diretórios, sistemas de comunicações, sistemas de colaboração e bases de dados, possam ser integradas ao *framework*;

5. Fornece um *framework* para integração de dados de relacionamentos sociais que permite que essas relações sejam persistentes e reutilizáveis, para além das fronteiras da aplicação. Este *framework* organiza as relações em um conjunto de contextos sociais distintos dentro do qual uma pessoa pode expressar diferentes papéis e personalidades.

O modelo de identidades do Higgins segue a abordagem baseada em cliente ativo, ou seja, uma aplicação precisa auxiliar o usuário a controlar as suas múltiplas identidades

e preferências. O Higgins oferece aos usuários três aplicações que atuam como seletores de identidades para a criação, seleção, compartilhamento e gerenciamento de diversos *i-cards* que representam a identidade do usuário em diferentes contextos e relacionamentos. Assim como no CardSpace, os benefícios do *login* centrado em cartões são também válidos para o Higgins. Neste modelo, é possível cruzar contextos e gerenciar qualquer tipo de informação do usuário como canções favoritas, números de identificação do empregado, carteira de habilitação, sua filiação, seu plano de saúde, entre outras que possam ser armazenadas em um cartão. É importante ressaltar que estes seletores são interoperáveis com o Microsoft CardSapce. Estes seletores de identidades estão disponíveis para alguns sistemas operacionais (Mac OSX, Linux e Windows), bem como para os navegadores Firefox e Internet Explorer [EclipseFoundation 2010].

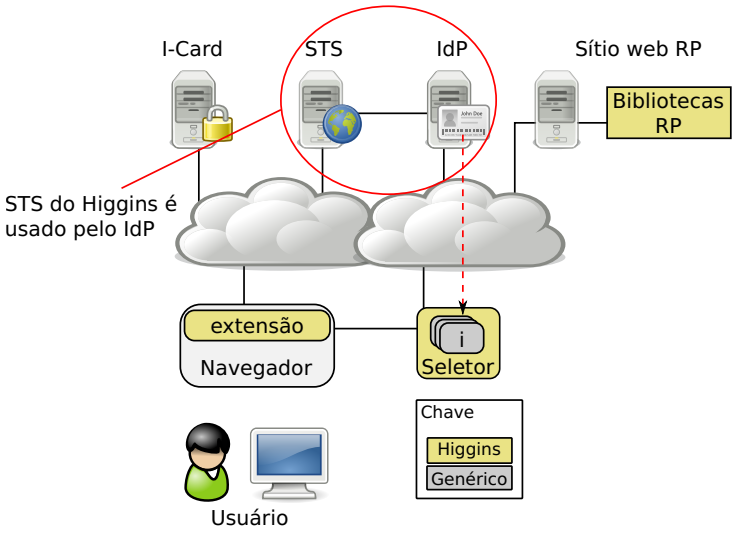


Figura 1.13

Para os desenvolvedores, o Higgins fornece dois provedores de identidades - IdP (que são Serviços Web). O primeiro é um STS (*Security Token Service*), baseado no WS-Trust, e o segundo suporta o padrão SAML 2.0. Higgins também provê bibliotecas para as partes confiantes necessárias para permitir que sítios *web* e sistemas possam solicitar e aceitar cartões de informação (*i-cards*). Os desenvolvedores podem incorporar este código da parte confiante dentro de suas aplicações e sítios *web* para tornar mais fácil para os usuários o processo de autenticação nestes sistemas. As partes confiantes podem prover autenticação OpenID e autenticação através de cartões de informação (*i-card*). A Figura 1.13 apresenta as entidades que participam do processo de autenticação, quando um usuário tenta acessar um parte confiante.

Abaixo das aplicações seletoras e dos Serviços *Web* mencionados anteriormente, encontra-se uma camada de abstração para o gerenciamento de identidades. Esta camada consiste em um *framework* que pode ser estendido através de *plugins*. A camada mais baixa deste *framework* é o serviço de atributos de identidade (*Identity Attribute Service* – IdAS), que fornece interoperabilidade e portabilidade através de federações de dados de identidade (ver Figura 1.14). O IdAS fornece acesso de leitura e escrita a uma ampla variedade de fontes de dados, incluindo diretórios LDAP e arquivos XML, e pode ser estendido usando *plugins* chamados *provedores de contexto*. Ou seja, este serviço torna

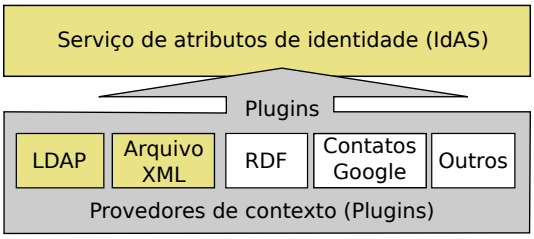


Figura 1.14: Serviço de Atributos de Identidades

possível combinar dados de redes sociais e de identidade através de fontes de dados altamente heterogêneos, incluindo diretórios, bancos de dados relacionais e as redes sociais.

1.4.8. Considerações sobre as soluções de gerenciamento

As soluções para gerenciamento de identidades apresentadas nessa seção compartilham pontos em comum, como a facilidade de autenticação única (SSO), distribuição das tarefas de autenticação e controle, preocupações com a privacidade e anonimato dos usuários. Todas consideram que o usuário pode fazer uso de um navegador *web*, sendo que a troca de informações entre provedores de identidades e de serviços se dá através de redirecionamentos HTTP.

As especificações do SAML apresentam um arcabouço genérico para lidar com identidades federadas, no qual são definidos metadados para representar informações de segurança, protocolos para troca de asserções de segurança, além de casos de uso, com usuários atrás de navegadores *web* ou não. Por se tratar de uma solução genérica, o SAML foi empregado por várias outras soluções para o gerenciamento de identidades, inclusive por todas as soluções apresentadas nessa seção.

O *Shibboleth* surgiu como uma solução aberta para permitir que instituições de ensino e pesquisa ofereçam uma gama maior de serviços para seus usuários, através do compartilhamento de recursos, esses oferecidos através de provedores de serviços. Hoje as principais federações acadêmicas do mundo fazem uso do *Shibboleth* e, em muitas dessas federações, cujo público alvo é formado por alunos e professores, empresas estão se filiando para oferecer serviços personalizados.

O projeto *Liberty Alliance* teve como objetivo facilitar as interações comerciais, usufruindo da Arquitetura Orientada a Serviços (AOS), através do conceito de federações, que em suas especificações são caracterizadas pelos círculos de confiança. Um dos pontos fortes do projeto foi sua participação direta nas especificações do SAML, sendo que muitas sugestões do *Liberty Alliance* são hoje parte do SAML 2.0. A WS-Federation fornece funcionalidades semelhantes àquelas do projeto *Liberty Alliance*, porém está fundamentada sobre uma pilha de especificações para Serviços *Web*, como WS-Trust e WS-Policy [Kallela 2008].

OpenID, CardSpace e Higgins são soluções que seguem a abordagem do gerenciamento de identidades federadas centrado no usuário. Dentre estas soluções, a solução que vem sendo amplamente usada, em especial devido a parceria com empresas que oferecem aplicações Web 2.0, é o OpenID. Uma das vantagens do OpenID é que este não requer software no lado do cliente. O CardSpace e o Higgins adotam o mo-

delo de cliente ativo (chamado de seletor de identidades), sendo que o cliente ativo do Higgins está disponível para diferentes plataformas. Outra diferença é que o OpenID adota a abordagem de identidade baseada no endereço e CardSpace e Higgins adotam a abordagem baseada em cartão (*tokens*). A literatura mostra que esta última abordagem é considerada mais flexível pois suporta identidades providas de diferentes fontes e também por oferecer um experiência consistente ao usuário e por impedir que um provedor de identidade rastreie os provedores de serviços (aplicações) acessados pelo usuário [Maler e Reed 2008, Akram e Hoffmann 2008].

Apesar de prover suporte a qualquer tipo de *token* de segurança, os protocolos adotados no *CardSpace* seguem somente os protocolos de Serviços *Web* da família WS-*, centrados na especificação WS-Trust. No entanto, o projeto Higgins da Fundação Eclipse está focado em uma solução mais independente pois além de oferecer suporte a provedores de identidade baseados na especificação WS-Trust, também oferece suporte a provedores de identidades baseados no SAML 2.0.

Atualmente, soluções de e-gov, como as do Governo dos Estados Unidos, começam a construir suas infraestruturas com suporte a tecnologias de identidades abertas, tais como o OpenID e o CardSpace. O padrão SAML é considerado uma tecnologia de identidade aberta, porém a necessidade de relações de confiança prévias entre IdPs e SPs fazem com que esta tecnologia não seja escalável para aplicações Web 2.0. Diante disto, *frameworks* abertos de confiança (terceiras partes confiáveis) estão sendo desenvolvidos para habilitar sítios *web* do Governo e aplicações a aceitarem credenciais emitidas por diferentes provedores de identidades, comerciais e acadêmicos [Thibeau e Reed 2009].

Segundo [Chadwick 2009], a maior limitação do *Shibboleth* e do CardSpace[7] é que o usuário pode selecionar apenas um provedor de identidades e apresentar apenas uma credencial a um provedor de serviços. Uma solução para este problema, proposto em [Chadwick e Inman 2009], é usar um componente chamado de Serviço de Ligação (*Linking Service*). Este serviço permite aos usuários agregar vários atributos de diferentes IdP preservando a privacidade desses usuários. O projeto Higgins também pretende trabalhar este problema, porém a versão atual ainda não oferece uma solução.

1.5. Privacidade no Gerenciamento de Identidades Federadas

O conceito de identidades federadas fornece aos usuários um modo conveniente para criar identidades e mover-se por vários SPs. Mas não se pode negar que, além de toda a simplicidade e conveniência oferecidas, a gestão dessas identidades federadas torna-se uma tarefa crucial e é necessário levar em consideração as várias ameaças contra a segurança e a privacidade dos dados do usuário. Qualquer infraestrutura de gerenciamento de identidades deve proteger adequadamente as informações do usuário e deve aderir adequadamente às política de privacidade definidas para os dados do mesmo.

Outras propriedades e características são necessárias para manter a privacidade de usuários em uma federação. Medidas de segurança devem garantir que os atributos do usuário não sejam divulgados de forma involuntária. No entanto, os mecanismos adicionais necessários para oferecer resistência aos diferentes tipos de posse indevida ou roubo

[7]Pode-se incluir também o OpenID.

destas identidades e atributos devem levar em conta as diferenças de contextos e as diferenças de políticas nas federações. Por exemplo, se um sistema particular de identidades federadas for centrado no usuário e permitir ao usuário armazenar suas próprias credenciais em um dispositivo de sua propriedade, então, medidas adicionais são necessárias para proteger as credenciais do mesmo o caso este dispositivo seja perdido. Proteger os dados do usuário onde estes estão armazenados é parte da segurança de um sistema e da privacidade do usuário.

O compartilhamento de informações de identificação pessoal é um grande desafio no que se refere à privacidade, à proteção de dados pessoais e à conformidade com aspectos legais que envolvem os direitos individuais das pessoas. Em sistemas de identidades federadas, verifica-se que entre os principais objetivos está o compartilhamento de tais informações.

Pode-se definir a privacidade como a divulgação mínima em nível funcional; isto é, fornecer somente os identificadores e informações necessários para assegurar a execução ou a continuidade de um serviço. Sistemas de identidades federadas, frequentemente, manipulam diferentes tipos de identificadores em diferentes contextos. Tais identificadores podem assumir um caráter *absoluto*, independentes de contexto, ou *relativo*, dependentes do contexto [Ahn e Lam 2005].

Uma técnica importante para a preservação da privacidade é o uso de pseudônimos, que são identificadores de usuários que não permitem inferências em relação à identidade real, propriedades ou atributos dos usuários a quem fazem referência. Pseudônimos podem ter significado *local*, dependente do contexto entre usuário e SP, ou *global*, independente do contexto e sendo válido por toda a federação. A validade pode também ser temporária ou permanente [Ahn e Lam 2005].

Dentre as soluções de gerenciamento de identidades federadas que oferecem mecanismos para prover privacidade, destaca-se a *Liberty Alliance* (ver Seção 1.4.3). Os pseudônimos usados em asserções SAML são construídos com base em valores pseudo-aleatórios que não têm correspondência discernível com os identificadores dos usuários em IdPs ou SPs. Um pseudônimo tem um significado apenas no contexto da relação entre as duas partes que estão se comunicando. A intenção é criar pseudônimos de forma a dificultar a ligação entre usuários e transações (serviços sendo acessados), mantendo assim a privacidade.

O *Liberty Alliance* provê suporte a uma abordagem de compartilhamento de atributos de usuário com o consentimento do mesmo [Aarts e Madsen 2006]. Isto significa que o usuário deve ser colocado no controle da liberação e uso de suas informações armazenadas em um provedor de atributos, papel que pode ser assumido por um IdP. Nas mensagens a serem trocadas neste protocolo a requisição deve especificar o propósito do uso das informações solicitadas e a resposta pode determinar as preferências do usuário em termos de privacidade ou política para o elemento requisitado.

As especificações do *Liberty Alliance* também abordam questões sobre políticas de privacidade multi-nível [Aarts e Madsen 2006, Ahn e Ko 2007], que se faz através de *rótulos de privacidade*. Rótulos de privacidade são semelhantes aos rótulos de segurança nos controles de acesso obrigatórios (*Mandatory Access Control* – MAC). Em controles

do tipo MAC, cada recurso ou objeto é etiquetado com um rótulo de segurança que representa a sensibilidade do recurso considerado. Um usuário (sujeito) desejando fazer um acesso ao recurso considerado deve possuir um nível de autorização (*clearance level*) adequado ao rótulo de segurança do recurso. Os níveis de privacidade são usados nas políticas de privacidade dos provedores de identidades, que também atuam como repositórios de atributos de usuários, e nas requisições de atributos enviadas pelos provedores de serviços aos IdPs.

Em vez de um grande número de políticas de privacidade, variado e personalizado, foi definido um pequeno número (padronizado) destas aos quais, ambos, requisitante e provedor de atributos devem aderir em favor do usuário. Isto leva a uma simplificação nas verificações da política nas requisições de informações. Para preservar a privacidade da informação do usuário, o IdP deve liberar informações de atributos requisitados somente com o consentimento do usuário. O IdP deve ter armazenado as preferências do usuário no que se refere a liberação de suas informações. Estas preferências são expressas na forma da política multi-nível. Ou seja, o usuário classificou suas informações segundo os níveis disponíveis. O IdP precisa só comparar o nível na requisição com o nível definido nas preferências do usuário. No caso de desacordo nestas comparações, o IdP pode tomar algumas ações definidas pelo próprio usuário. É lógico que a definição destas ações e políticas deve considerar vários aspectos do contexto da interação entre usuário e SP e as intenções do usuário em relação à suas informações pessoais.

Os requisitos que devem atender os níveis de privacidade seguem as seguintes regras conforme discutido em [Ahn e Lam 2005]:

- *Os níveis de privacidade devem ser hierárquicos e comparáveis entre as partes envolvidas* – é necessário avaliar cada requisição de atributo pela comparação dos rótulos do SP requerente com a da política do usuário. A informação requisitada é liberada tomando como base esta avaliação;

- *Rótulos de privacidade devem ser amigáveis* – os usuários tendem a usar rótulos para representar conceitos abstratos como níveis de seriedade ou justiça. As pessoas podem não compreender completamente as necessidades exatas da definição destes níveis. Ou seja, talvez estas não consigam dimensionar a importância das informações nos contextos;

- *Os rótulos devem funcionar com um motor de política* – ao invés de avaliar cada descrição de elemento nas políticas de privacidade este motor necessita apenas comparar níveis de privacidade para cada atributo requisitado. Isto reduz consideravelmente o custo do processamento de todas as requisições.

Na abordagem de gerenciamento de identidades federadas centrado no usuário, as soluções de privacidade devem atender alguns pontos chaves [Ahn e Lam 2005]:

- *Notificação* – usuários devem receber *a priori* notificações sobre determinadas liberações de suas informações.

- *Definição de propósitos* – os usuários devem especificar o propósito do uso e quais de suas informações devem ser coletadas;
- *Disponibilidade de suas informações* – os usuários podem acessar e modificar suas informações pessoais quando necessário;
- *Segurança de suas informações* – usuários devem estar certos de que o sistema de gestão de identidades é capaz de proteger suas informações pessoais.

1.6. Ferramentas Computacionais para Implantação de Gerenciamento de Identidades

Nesta seção são apresentadas algumas das principais ferramentas usadas por soluções de gerenciamento de identidades.

1.6.1. Metro

O *framework Metro*[8], componente do projeto *Glassfish* da *Oracle* [9], possui um extenso conjunto de bibliotecas que podem ser facilmente estendidas e implementa padrões e especificações apropriadas ao gerenciamento de identidades, aprovadas e adotadas por diversas empresas e organizações [10], tais como Microsoft, Oracle e OASIS[11]. Estas características o destaca como uma importante opção a ser considerada no desenvolvimento de aplicativos que incluam o gerenciamento de identidades entre seus recursos.

A importância do *framework Metro* no cenário de Serviços *Web* está ligado ao fato deste conter em seu conjunto de bibliotecas, implementações chamadas "implementações de referência" de diversas especificações da plataforma Java. Dentre estas, destaca-se a especificação JAX-WS - *Java API for XML Web Services*[12] - que define as bases da criação de serviços *Web* com foco na linguagem Java e é parte da plataforma Java EE da *Oracle*.

Além da implementação da especificação JAX-WS e de outras implementações auxiliares, destacam-se as bibliotecas *Policy*, responsável pela implementação da especificação *WS-Policy*; XWSS - *XML and Web Services Security* - responsável pela implementação da especificação *WS-Security*; e WSIT - *Web Services Interoperability Technologies* [13] - responsável pela implementação das seguintes especificações: *WS-Trust*, *WS-SecurityConversation*, *WS-SecurityPolicy*, *WS-ReliableMessaging*, *SOAP over TCP WS-MetadataExchange* e *WS-AtomicTransactions*.

O *framework Metro* fornece toda funcionalidade básica para construção de um STS, definido na WS-Trust, por meio da classe `BaseSTSImpl`. Essa classe, por padrão, permite a emissão de asserções SAML 1.1 e 2.0. A implementação é extensível por meio das seguintes interfaces: `STSTokenProvider`, para suportar outros tipos

[8] `https://metro.dev.java.net/discover`
[9] `https://glassfish.dev.java.net`
[10] `https://metro.dev.java.net/guide/Metro_Specifications.html`
[11] `http://www.oasis-open.org`
[12] `https://jax-ws.dev.java.net`
[13] `https://wsit.dev.java.net`

de credenciais; `STSAttributeProvider`, para obter atributos de fontes variadas; e `STSAuthorizationProvider`, para emitir credenciais e declarações de autorização.

1.6.2. Shibboleth

A implantação de um sistema de controle de acessos unificado baseado no Shibboleth em uma instituição requer a configuração de alguns serviços distintos, citados na seção 1.4.2

O provedor de identidades (IdP) do Shibboleth [14] consiste de uma aplicação *web*, disponível em formato WAR, que é disponibilizada em um contêiner Java como o *Apache Tomcat*. O contêiner Java por sua vez deve ser executado vinculado a um servidor *web*, por exemplo *Apache HTTP*, o qual terá o papel de deixar o IdP disponível na rede.

```
1 <Location /secure>
2  AuthType shibboleth
3  ShibRequireSession On
4  require valid-user
5  Order allow,deny
6  allow from all
7 </Location>
```

Figura 1.15: Exemplo de configuração do *Apache* para autenticação via Shibboleth

Provedores de serviços do Shibboleth são aplicações *web* comuns que permitem que seus usuários sejam autenticados através dos provedores de identidades. O Shibboleth disponibiliza um módulo próprio para ser integrado ao servidor *web Apache*, permitindo assim que aplicações *web*, disponibilizadas através do Apache HTTP, possam facilmente usufruir da infraestrutura do Shibboleth. A Figura 1.15 ilustra um exemplo de configuração do servidor web Apache usufruindo do módulo Shibboleth para realizar a autenticação dos usuários que acessarem o recurso `/secure`.

1.6.3. WSO2 Identity Server

O *WSO2 Entity Provider*[15] é uma solução completa, que abrange desde a conexão com fontes de dados LDAP e JDBC até a interface com o usuário por meio de telas para o processo de autenticação. O *WSO2 Entity Provider* é uma solução de código aberto baseada no *WSO2 Carbon*, uma plataforma com diversas funcionalidades para a construção de aplicações para *web*. As principais funcionalidades oferecidas pelo *WSO2 Identity Server* são: controle de acesso com XACML 2.0, suporte a asserções SAML 1.1/2.0, bem como implementações para STS WS-Trust (ver Seção 1.4.4), IdP SAML 2.0 (*Web SSO*), provedor OpenID, provedor de *Information Cards* e um serviço XKMS [Hallam-Baker e Mysore 2005].

[14] O Shibboleth foi a tecnologia adotada pela Federação CAFe. Diversos documentos para instalação e configuração dos serviços estão disponíveis na página da federação (`http://cafe.rnp.br`)

[15] `http://wso2.com/products/identity-server`

1.6.4. Lasso

Lasso[16] consiste em um conjunto de bibliotecas para linguagem C que implementam os conceitos definidos pelas especificações *Liberty Alliance* (ver Seção 1.4.3). A versão atual implementa a ID-FF 1.2, SAML 2.0 e boa parte da ID-WSF. Apesar de ser desenvolvido em C, *Lasso* possui diversos mapeamentos para outras linguagens, como Python e Java.

1.6.5. Windows CardSpace

Conforme visto na Seção 1.4.6, O *Windows CardSpace* é um componente da plataforma .NET. Um provedor de identidades compatível com o *CardSpace* pode ser um STS padrão. Portanto, qualquer implementação da *WS-Trust* pode ser usada para construir um provedor *CardSpace*. Os provedores de serviços, por outro lado, são construídos por meio de extensões HTML ou XHTML que indicam para o navegador quando acionar o seletor de identidades.

```
<OBJECT type="application/x-informationCard" name="xmlToken">
  <PARAM Name="tokenType" Value="urn:oasis:names:tc:SAML:1.0:assertion" />
  <PARAM Name="issuer"
     Value="http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self" />
  <PARAM Name="requiredClaims"
     Value=
     "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
     http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname
     http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname" />
</OBJECT>
```

Figura 1.16: Exemplo de uma configuração para invocar o seletor de identidade do *CardSpace*

A Figura 1.16 apresenta um exemplo de elemento HTML que indica o uso do seletor de identidades. Esse elemento contém parâmetros que indicam que a credencial exigida é uma asserção SAML, emitida pelo IdP (elemento `issuer`) e que deve conter os atributos e-mail, nome e sobrenome.

No lado do usuário, se um navegador habilitado a reconhecer um pedido de autenticação por *InfoCard*, receber uma página com o elemento acima, este poderá invocar uma interface para o usuário selecionar seu *InfoCard* apropriado. A ferramenta *CardSpace* possui um módulo responsável por realizar essa interface com o usuário e é distribuída como parte do *framework .Net*[17].

1.6.6. Eclipse Higgins

O projeto *Eclipse Higgins* (ver Seção 1.4.7) é uma solução que visa aumentar o controle das pessoas sobre suas identidades digitais. O desenvolvimento do projeto é dividido em três partes, *Active Client*, *Personal Data Store* e *Identity Services*. O *Active Client* é um módulo integrado ao navegador *web* que automatiza o processo de autenticação do usuário usando credenciais como *InfoCard*, *OpenID* e nome de usuário e senha. Essa

[16] `http://lasso.entrouvert.org`

[17] `http://www.microsoft.com/windows/products/winfamily/cardspace/getitnow.mspx`

funcionalidade apresenta alguma sobreposição com o *CardSpace*, no entanto o *Higgins* provê suporte a mais tecnologias de autenticação e um número maior de plataformas e navegadores.

O *Personal Data Store* (PDS) é um serviço de armazenamento, sincronização e compartilhamento de atributos de identidade do usuário. Esse serviço ainda não é um produto completo, mas um plano para a próxima versão (2.0). O PDS deverá prover um ponto central de controle sobre informações acerca de um usuário e uma maneira para compartilhar dados entre PDSs de diferentes usuários para formar uma espécie de rede social.

O *Identity Services* implementa IdPs de acordo com a *WS-Trust* e a SAML 2.0, bem como bibliotecas Java para implementação de partes confiantes *InfoCard*. Os IdPs são distribuídos como pacotes independentes e a biblioteca para implementação de partes confiantes é distribuída acoplada a um exemplo de aplicação *web* que ilustra o uso de *InfoCards* para controlar o acesso a recursos.

1.7. Considerações finais

As redes colaborativas, que utilizam computadores e dispositivos móveis, oferecem novas possibilidades de conexões, oportunidades e aplicações. No entanto, a forma como as pessoas e as organizações (privadas e públicas) farão uso dessas oportunidades e aplicações, dependerá do progresso da autenticação de identidades digitais e do gerenciamento destas identidades [Lewis 2008].

Conforme visto neste Capítulo, o modelo de gerenciamento de identidades federadas beneficia tanto usuários quanto provedores de serviços. Constata-se que promover identidades federadas apresenta desafios complexos em termos de questões técnicas e necessidades humanas [Maler e Reed 2008]. Requisitos importantes muitas vezes parecem ser mutuamente exclusivos. Alguns aspectos de segurança, tais como auditoria do acesso a recursos do sistema, pode entrar em conflito com questões de privacidade do usuário. Ao mesmo tempo, a capacitação do usuário, tais como possibilitar que este atue como intermediador de fluxos de dados, pode entrar em conflito com as conveniências do usuário, tais como a realização de autenticação única totalmente "silenciosa".

As principais soluções para prover o gerenciamento de identidades federadas foram descritas e analisadas neste texto. Constatou-se que o SAML 2.0 é base para estas soluções, o *Shibboleth* se tornou um padrão de fato nas redes acadêmicas e a solução *Liberty Alliance* está sendo adotada por uma grande comunidade de empresas privadas e também por empresas públicas. As soluções mais recentes do modelo centrado no usuário, em especial OpenID e CardSpace tem despertado muito interesse, em especial dos provedores de serviços que seguem a abordagem Web 2.0 e por governos que desejam incluir ativamente seus cidadãos através das redes sociais e de seus programas de E-Gov.

Para [Maler e Reed 2008], a interoperabilidade é um desafio contínuo para prover identidades federadas. No entanto, muitos desenvolvedores estão começando a combinar diferentes soluções, de acordo como estas crescem em popularidade. Por exemplo, promover a autenticação em um IdP OpenID ou SAML, usando um cartão de informação (*i-Card* ou *InfoCard*) ou usando um OpenID ao invés de uma asserção SAML para então

acessar um Serviço *Web* com suporte a *Liberty Identity*.

O primeiro passo para para resolver os problemas de interoperabilidade é o entendimento das distâncias entre as tecnologias. Alguns projetos estão sendo desenvolvidos visando promover a interoperabilidade em sistemas de gerenciamento de identidades federadas. Entre estes destacam-se: a iniciativa Kantara, que engloba o projeto Concordia e o grupo de trabalho OSIS (*Open Source Identity System*) do *Identity Commons*. A iniciativa *Kantara*[18] constituiu uma organização para resolver os desafios de interoperabilidade e de harmonização que existem entre as empresa que oferecem serviços e aplicações Web. Criada de forma colaborativa, a iniciativa visa promover a inovação necessária para a ampla adoção de soluções interoperáveis de identidades federadas adequadas para todas as indústrias e regiões e alinhada as necessidades das redes móveis. Já o projeto OSIS [19], reúne muitos projetos de gerenciamento de identidades e tem objetivo contribuir com a construção de uma camada de identidade interoperáveis a partir do soluções de código-fonte aberto e soluções comerciais. Os projetos atuais incluem esforços para promover a interoperabilidade entre o Information Card (CardSpace) e OpenID.

Segundo [Chadwick 2009], gerenciamento de identidades federadas é um tema de pesquisa ativo e, provavelmente, diante da sua complexidade e relevância, continuará assim por muito mais anos. Esta constatação decorre das inúmeras questões que os sistemas de identidades federadas devem considerar, tais como: facilidade de uso, privacidade do usuário, segurança forte, autenticação única diante de diferentes tecnologias, custo dos sistemas (*total cost of ownership*), escalabilidade, controle de acesso de granularidade fina (baseado em atributos), personalização dos serviços e anonimato.

Referências

[Aarts e Madsen 2006] Aarts, R. e Madsen, P. (2006). *Liberty ID-WSF Interaction Service Specification v.2*. Liberty Alliance Project. `http://www.projectliberty.org/liberty/content/download/`.

[Ahn e Ko 2007] Ahn, G.-J. e Ko, M. (2007). User-centric privacy management for federated identity management. *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 0:187–195.

[Ahn e Lam 2005] Ahn, G.-J. e Lam, J. (2005). Managing privacy preferences for federated identity management. In *DIM '05: Proceedings of the 2005 workshop on Digital identity management*, pages 28–36, New York, NY, USA. ACM.

[Akram e Hoffmann 2008] Akram, H. e Hoffmann, M. (2008). Supports for identity management in ambient environments - the hydra approach. In *ICSNC '08: Proceedings of the 2008 Third International Conference on Systems and Networks Communications*, pages 371–377, Washington, DC, USA. IEEE Computer Society.

[Baldoni 2010] Baldoni, R. (2010). Federated Identity Management Systems in e-Government: the Case of Italy. *Electronic Government: An International Journal*, 8(1).

[18] `http://kantarainitiative.org`
[19] `http://osis.idcommons.net`

[Bartel et al. 2002] Bartel, M., Boyer, J., e Fox, B. (2002). *XML-Signature Syntax and Processing*. W3C. http://www.w3.org/TR/xmldsig-core.

[Bhargav-Spantzel et al. 2007] Bhargav-Spantzel, A., Camenisch, J., Gross, T., e Sommer, D. (2007). User centricity: a taxonomy and open issues. *Journal of Computer Security*, 15(5):493–527.

[Burr et al. 2006] Burr, W. E., Dodson, D. F., e Polk, W. T. (2006). Electronic authentication guideline. *NIST Special Publication*, 800:63.

[Camarinha-Matos et al. 2008] Camarinha-Matos, L. M., Afsarmanesh, H., e Ollus, M. (2008). *Methods and Tools for Collaborative Networked Organizations*, chapter Ecolead And Cno Base Concepts, pages 3–32. Springer.

[Camenisch e Pfitzmann 2007] Camenisch, J. e Pfitzmann, B. (2007). *Security, Privacy, and Trust in Modern Data Management*, chapter Federated Identity Management, pages 213–238. Springer Verlag.

[Cameron 2005] Cameron, K. (2005). The laws of identity. http://www.identityblog.com/?p=352/#lawsofiden_topic3.

[Carmody et al. 2005] Carmody, S., Erdos, M., Hazelton, K., Hoehn, W., Morgan, B., Scavo, T., e Wasley, D. (2005). Incommon technical requirements and information. vol. 2005.

[Chadwick 2009] Chadwick, D. (2009). Federated identity management. *Foundations of Security Analysis and Design V*, pages 96–120.

[Chadwick e Inman 2009] Chadwick, D. e Inman, G. (2009). Attribute aggregation in federated identity. *IEEE Computer*, pages 44–53.

[Chappell 2006] Chappell, D. (2006). Introducing windows cardspace. Msnd technical articles, Microsoft Corporation. http://msdn.microsoft.com/en-us/library/aa480189.aspx.

[Clauß e Köhntopp 2001] Clauß, S. e Köhntopp, M. (2001). Identity management and its support of multilateral security. *Computer Networks*, 37(2):205–219.

[Damiani et al. 2003] Damiani, E., di Vimercati, S. D. C., e Samarati, P. (2003). Managing multiple and dependable identities. In *IEEE Internet Computing*, pages 29–37. IEEE.

[Dawes e Pardo 2008] Dawes, S. S. e Pardo, T. A. (2008). *Advances in Digital Government Technology, Human Factors, and Policy*, chapter Building Collaborative Digital Government Systems Systemic: constraints and effective practices, pages 259–273. Springer US.

[de Mello 2009] de Mello, E. R. (2009). *Um modelo para confiança dinâmica em ambientes orientados a serviço*. PhD thesis, Universidade Federal de Santa Catarina.

[de Mello et al. 2009] de Mello, E. R., Wangham, M. S., da Silva Fraga, J., Camargo, E., e da Silva Böger, D. (2009). Model for authentication credentials translation in service oriented architecture. *Transactions on Computational Sciences Journal*, 5430:68–86.

[EclipseFoundation 2010] EclipseFoundation (2010). Higgins open source identity framework. `http://www.eclipse.org/higgins/`.

[Gottschalk e Solli-Saether 2008] Gottschalk, P. e Solli-Saether, H. (2008). Stages of e-government interoperability. *Electronic Government: An International Journal*, 5(3):310–320.

[GOV.BR 2010] GOV.BR (2010). Programa de governo eletrônico brasileiro (gov.br). `http://www.governoeletronico.gov.br`.

[Hallam-Baker e Mysore 2005] Hallam-Baker, P. e Mysore, S. H. (2005). *XML Key Management Specification (XKMS 2.0)*. W3C – Proposed Recommendation.

[Hansen et al. 2008] Hansen, M., Schwartz, A., e Cooper, A. (2008). Privacy and identity management. *Security Privacy, IEEE*, 6(2):38 –45.

[Hodges e Morgan 2002] Hodges, J. e Morgan, R. (2002). *Lightweight Directory Access Protocol (v3): Technical Specification. RFC3377*. IETF.

[Housley et al. 2002] Housley, R., Polk, W., Ford, W., e Solo, D. (2002). *Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. IETF RFC 3280.

[IBM 2005] IBM (2005). *Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions*. IBM, second edition.

[Internet2 2008] Internet2 (2008). eduPerson & eduOrg Object Classes. `http://middleware.internet2.edu/eduperson/`.

[ITU-T 2001] ITU-T (2001). *Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services*. ITU-T Recommendation X.500. `http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.500`.

[Jøsang et al. 2005] Jøsang, A., Fabre, J., Hay, B., Dalziel, J., e Pope, S. (2005). Trust requirements in identity management. In *CRPIT '44: Proceedings of the 2005 Australasian workshop on Grid computing and e-research*, pages 99–108, Darlinghurst, Australia. Australian Computer Society, Inc.

[Jøsang e Pope 2005] Jøsang, A. e Pope, S. (2005). User centric identity management. In *AusCERT Asia Pacific Information Technology Security Conference 2005*.

[Kallela 2008] Kallela, J. (2008). Federated identity management solutions. Technical report, Helsinki University of Technology. `http://www.cse.tkk.fi/en/publications/B/1/papers/Kallela_final.pdf`.

[Kohl e Neuman 1993] Kohl, J. e Neuman, C. (1993). The kerberos network authentication requestor (v5). rfc1510. Technical report, IETF.

[Kürümlüoglu et al. 2005] Kürümlüoglu, M., Nostdal, R., e Karvonen, I. (2005). *Base concepts*, chapter Virtual organisations: Systems and practices, pages 11–28. Springer.

[Le e Bouzefrane 2008] Le, H.-B. e Bouzefrane, S. (2008). Identity management systems and interoperability in a heterogeneous environment. pages 239 –242.

[Leach et al. 2005] Leach, P., Mealling, M., e Salz, R. (2005). *A UUID URN Namespace*. IETF RFC 4122. `http://www.ietf.org/rfc/rfc4122.txt`.

[Lewis 2008] Lewis, J. A. (2008). Authentication 2.0 - new opportunities for online identification. Technical report, Center for Strategic and International Studies.

[Liberty 2003] Liberty (2003). *Introduction to the Liberty Alliance Identity Architecture*. Liberty Alliance.

[Lopez et al. 2006] Lopez, D., Solberg, A., e Stanica, M. (2006). eduGAIN Profiles and Implementation Guidelines.

[Maler e Reed 2008] Maler, E. e Reed, D. (2008). The venn of identity: Options and issues in federated identity management. *Security Privacy, IEEE*, 6(2):16–23.

[Maliki e Seigneur 2007] Maliki, T. E. e Seigneur, J.-M. (2007). A survey of user-centric identity management technologies. In *The International Conference on Emerging Security Information, Systems, and Technologies, 2007. SecureWare 2007*, pages 12–17.

[OASIS 2004] OASIS (2004). *Web Services Security: SOAP Message Security 1.0*. OASIS. `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf`.

[OASIS 2005a] OASIS (2005a). *Assertions and Protocols for the SAML 2.0*. OASIS.

[OASIS 2005b] OASIS (2005b). *Bindings for the OASIS SAML V2.0*. Organization for the Advancement of Structured Information Standards (OASIS).

[OASIS 2005c] OASIS (2005c). *eXtensible Access Control Markup Language (XACML) version 2.0*. Organization for the Advancement of Structured Information Standards (OASIS). `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`.

[OASIS 2005d] OASIS (2005d). *Extensible Resource Identifier (XRI) Syntax V2.0*. OASIS. `http://www.oasis-open.org/committees/download.php/15377/xri-syntax-V2.0-cs.pdf`.

[OASIS 2005e] OASIS (2005e). *Metadata for the OASIS SAML V2.0*. Organization for the Advancement of Structured Information Standards (OASIS).

[OASIS 2005f] OASIS (2005f). *Profiles for the OASIS SAML V2.0*. Organization for the Advancement of Structured Information Standards (OASIS).

[OASIS 2005g] OASIS (2005g). *Security Assertion Markup Language (SAML) 2.0 Technical Overview*. OASIS.

[OASIS 2009a] OASIS (2009a). *WS-SecurityPolicy 1.3*. OASIS. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/ws-securitypolicy.html.

[OASIS 2009b] OASIS (2009b). *WS-Trust 1.4*.

[OPENID 2007] OPENID (2007). Openid authentication 2.0. OPENID. http://openid.net/specs/openid-authentication-2_0.html.

[OpenID 2010] OpenID (2010). Openid. http://openid.net.

[Rabelo 2008] Rabelo, R. J. (2008). *Methods and Tools for Collaborative Networked Organizations*, chapter Advanced Collaborative Business ICT Infrastructures, pages 337–365. Springer.

[Recordon e Reed 2006] Recordon, D. e Reed, D. (2006). Openid 2.0: a platform for user-centric identity management. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 11–16, New York, NY, USA. ACM.

[RNP 2010] RNP (2010). Federação cafe. http://www.cafe.rnp.br.

[Scavo e Cantor 2005] Scavo, T. e Cantor, S. (2005). *Shibboleth Architecture*. http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf.

[Smith 2000] Smith, M. (2000). *Definition of the inetOrgPerson LDAP Object Class*. IETF RFC 2798.

[TERENA 2008] TERENA (2008). *TERENA Compendium of National Research and Education Networks In Europe*. TERENA.

[Thibeau e Reed 2009] Thibeau, D. e Reed, D. (2009). Open trust frameworks for open government: Enabling citizen involvement through open identity technologies. White paper, OpenID Foudation and Information Card Foudation.

[W3C 2007] W3C (2007). *Web Services Policy 1.5 - Framework*. http://www.w3.org/TR/2007/REC-ws-policy-20070904.

[W3C 2009a] W3C (2009a). *Web Services Federation Language – WS-Federation*.

[W3C 2009b] W3C (2009b). *Web Services Metadata Exchange (WS-MetadataExchange)*. W3C. http://www.w3.org/TR/2009/WD-ws-metadata-exchange-20090317.

[W3C 2010] W3C (2010). *Web Services Transfer (WS-Transfer)*. http://www.w3.org/TR/2010/WD-ws-transfer-20100805.

[Wahl 1997] Wahl, M. (1997). *A Summary of the X.500(96) User Schema for use with LDAPv3*. IETF RFC 2256.

Capítulo 2

Aspectos de segurança e privacidade em ambientes de Computação em Nuvem

Arlindo Marcon Jr[1,2], Marcos Laureano[1,2], Altair Santin[1], Carlos Maziero[1]

[1]Programa de Pós-Graduação em Informática – PPGIa
Pontifícia Universidade Católica do Paraná – PUCPR

[2]Instituto Federal de Educação, Ciência e Tecnologia do Paraná – IFPR

Abstract

Cloud computing aims to provide on-demand access to a pool of computing resources, in a dynamic and easily scalable environment. The use of services provided by third parties allows to minimize efforts in local information technology management, and gives benefits in mobility, scalability, and availability. This short course intends to explore the state of the art in the security and privacy areas of cloud computing environments. Initially, the fundamental aspects of cloud computing will be presented. After a short review of the main concepts in security and privacy, it presents a deeper analysis of the relevant risks and threats in cloud environments, as well as the most known approaches to deter them. The text also discusses some open problems and tentative solutions proposed in the litterature.

Resumo

A computação em nuvem visa prover acesso sob demanda a um pool *de recursos computacionais em um ambiente dinâmico e facilmente escalável. A partir de serviços prestados por terceiros se minimizam as preocupações de gerenciamento de tecnologia de informação local, trazendo vantagens de mobilidade, escalabilidade e disponibilidade. Este minicurso visa explorar o estado da arte nas áreas de segurança e privacidade no contexto de computação em nuvem. Inicialmente serão apresentados os aspectos fundamentais de computação em nuvem. Após uma revisão dos principais conceitos de segurança e privacidade, serão discutidos em maior profundidade os riscos e ameaças relevantes nos ambientes de nuvem, bem como as abordagens mais conhecidas para mitigá-los. Ao longo do texto serão discutidos problemas em aberto e tentativas de solução propostas na literatura.*

2.1. Introdução

Computação em nuvem (*Cloud Computing*) visa prover acesso sob-demanda para um *pool* de recursos computacionais (e.g. rede, armazenamento, serviços). Estes recursos podem ser rapidamente providos/liberados com pouco esforço de gerenciamento, pois o ambiente é nativamente dinâmico e facilmente escalável [Mell e Grance 2009]. A principal motivação para computação em nuvem é que a partir de serviços prestados por terceiros se eliminam as preocupações de gerenciamento de tecnologia de informação local, i.e., instalação, configuração e atualização de sistemas, e manutenção da infraestrutura computacional física [Hayes 2008]. Ou seja, a computação em nuvem oferece vantagens relacionadas a mobilidade, escalabilidade, disponibilidade, e implantação de sistemas computacionais.

A definição mais aceita para descrever a computação em nuvem é composta de sete características fundamentais: três modelos de serviço e quatro abordagens de implantação [Mell e Grance 2009]. Os modelos de serviço são compostos pela (a) *Infraestrutura como Serviço*/IaaS – fornece recursos computacionais como processamento, armazenamento, rede etc., (b) *Plataforma como Serviço*/PaaS – permite utilizar a infraestrutura de nuvem para criar e implantar novas aplicações próprias ou para prover suporte para nível de SaaS e (c) *Software como Serviço*/SaaS – provê aplicações à nuvem para serem consumidas sob-demanda. A implantação dos modelos podem seguir uma abordagem (i) *pública* – com acesso disponibilizado para o público em geral, (ii) *privada* – de uso exclusivo de uma organização, (iii) *comunitária* – compartilhada por organizações com interesses comuns ou (iv) *híbrida* – qualquer tipo de combinação entre as categorias anteriores.

Para a infraestrutura o mecanismo de virtualização é uma das principais abordagens. Este permite a flexibilização do uso da camada de hardware. As máquinas virtuais proveem ambientes de processamento independentes e isolados, podendo ser instanciadas e destruídas sob demanda. Dessa forma, o ambiente de máquinas virtuais constitui uma base bastante adequada para a construção de infraestruturas de computação em nuvem [Grobauer et al. 2010].

O modelo de serviço, operacional e as tecnologias utilizadas para prover os serviços do ambiente de computação em nuvem apresentam diferentes níveis de riscos se comparado ao ambiente tradicional de tecnologia de informação [CSA 2009]. O provimento de recursos sob demanda para o processamento e armazenamento massivo de dados está sujeito a falhas de segurança, abusos com relação à privacidade, violação de direitos autorais, etc. Preocupações este aspectos de segurança computacional estão impedindo a ampla adoção da computação em nuvem.

Este minicurso tem como objetivo explorar o estado da arte nas áreas de segurança e privacidade no contexto de computação em nuvem (*cloud computing*). Inicialmente serão apresentados os aspectos fundamentais de computação em nuvem. Após uma breve revisão dos principais conceitos de segurança e privacidade, serão discutidos em maior profundidade os riscos e ameaças relevantes nos ambientes de nuvem, bem como as abordagens conhecidas para mitigá-los. Ao longo do texto serão discutidos problemas em aberto e tentativas de solução propostas na literatura.

2.2. Computação em nuvem

2.2.1. Introdução

Há quase 50 anos foi criado o sistema de compartilhamento de tempo – *time-sharing*. Este sistema fornecia acesso a poder computacional para usuários que não possuíam seu próprio *mainframe*, usando sistemas do tipo *hub-and-spoke*. No começo dos anos 1980 com a chegada dos computadores pessoais, programas e dados não dependiam mais de um centro computacional para executar. Cada indivíduo passou a controlar seu próprio ambiente de trabalho, customizando-o de acordo com suas necessidades. O modelo cliente-servidor, introduzido na mesma época, oferecia um serviço que podia ser invocado através da rede para atender uma necessidade do cliente [Bhattacharjee 2009].

Atualmente, softwares de prateleira ainda dominam o mercado, porém este paradigma está mudando para os ambientes de computação em nuvem – *Cloud Computing*. Tarefas computacionais podem ser migradas dos computadores de mesa e servidores corporativos para a nuvem computacional [Erickson et al. 2009]. A mudança de paradigma marca a inversão de uma tendência que perdurou por muitos anos, afetando todos os níveis do ecossistema computacional, incluindo usuários, desenvolvedores, gerentes de Tecnologia da Informação e os fabricantes de hardware [Hayes 2008].

A nuvem computacional pode ser utilizada para hospedar softwares em centros computacionais disponíveis e acessíveis via Internet. Na topologia atual não existe um ponto central – *hub*, um terminal cliente pode se comunicar com muitos servidores ao mesmo tempo, sendo que estes podem estar trocando informações entre si. A nuvem computacional pode ter uma ou mais centrais de gerenciamento – formada por vários provedores administrando diferentes domínios. Novas funcionalidades para aplicações e serviços podem ser disseminadas a partir de uma central administrativa, sem que o consumidor tenha que se preocupar com a complexidade de gerenciamento do ambiente.

As principais entidades que fazem parte do modelo de computação em nuvem, fornecendo ou interagindo com os serviços podem ser brevemente descritas como: (i) *provedor* – entidade que gerencia e fornece os serviços hospedados nas infraestruturas físicas. O provedor tem controle sobre os recursos computacionais – processador, memória etc.; (ii) *consumidor* – empresa ou organização que contrata e utiliza os serviços de um provedor; (iii) *usuário* – entidade que pode estar vinculada a um consumidor ou agindo por conta própria – o usuário final do serviço.

A migração de sistemas tradicionais para os serviços fornecidos pela nuvem pretende reduzir os custos de manutenção da infraestrutura de TI (Tecnologia da Informação)do consumidor, oferecendo as seguintes vantagens [Zhang et al. 2010]: economia em servidores, armazenamento, rede, licenças de software, energia, resfriamento e bens materiais; redução de trabalho na administração de sistemas; redução do tempo de configuração; diminuição de equipes de trabalho; desenvolvimento de aplicações com ciclo de vida mais curto e consequente redução do tempo de disponibilização de novos produtos e serviços no mercado; maior confiabilidade com custos menores e redução de gastos com manutenção, redução de custos com atualizações de hardware/infraestrutura.

A computação em nuvem também oferece vantagens relacionadas a mobilidade, escalabilidade e disponibilidade de sistemas. A escalabilidade é uma das maiores vantagens

para quem pretende implantar um serviço usando o modelo de computação em nuvem, pois é possível demandar recursos adicionais mesmo se o número de usuários crescer de forma imprevisível. Empresas consumidoras que necessitam de serviços de TI estão considerando a possibilidade de utilizar provedores de serviços terceirizados (*off-premise*), tirando proveito das vantagens oferecidas pela computação em nuvem.

Para facilitar a entrada de novas empresas no mercado, a computação em nuvem aplica o modelo de negócio *pay-as-you-go*. Ou seja, a cobrança é feita de acordo com a utilização de recursos e serviços. Na computação em nuvem, os gastos com capital são convertidos em gastos operacionais [Armbrust et al. 2010]. Os serviços fornecidos pelo provedor de computação em nuvem podem ser distribuídos e utilizados de maneira não uniforme, de acordo com a necessidade do consumidor. Na comunidade de rede há uma situação análoga, onde a largura de banda tem preço baseado em uso (*usage-based pricing*). A computação em nuvem é utilizada pelos consumidores e contabilizada pelo provedor para que o uso de recursos e serviços possa ser faturado [Bhattacharjee 2009].

A computação em nuvem permite que os consumidores utilizem a quantidade de recursos necessários para realizar testes com novos sistemas. Se um projeto falhar durante sua fase inicial, por exemplo, o consumidor do serviço da nuvem investiu pouco no negócio, podendo facilmente alterar seu tipo de negócio ou seus provedores de serviço. No modelo tradicional (*on-premise*) o contratante precisa gastar previamente em licenças, infraestrutura, consultoria etc.; se o projeto falhar esse investimento prévio vira prejuízo.

Em grandes *data centers* as máquinas físicas estão totalmente utilizadas em torno de 20% a 30% do tempo. Com a aplicação da computação em nuvem e das respectivas tecnologias de suporte (e.g. virtualização), a utilização dos recursos é maximizada, reduzindo o tempo ocioso de cada máquina. A camada de virtualização abstrai o hardware, intermediando o acesso de várias aplicações aos mesmos recursos físicos: processador, memória etc. Esta tecnologia permite realizar a consolidação (agrupamento) de cargas de trabalho em poucos servidores físicos, reduzindo os gastos com energia e resfriamento e consequentemente levando a economia em hardware, manutenção etc. A utilização adequada do ambiente de computação em nuvem agrega valor ao negócio de seus consumidores [Bhattacharjee 2009].

A técnica da virtualização permite que sistemas hospedados em um servidor físico sejam transferidos para outros servidores, executando o balanceamento de carga ou cópias de segurança dos sistemas. Com está abordagem, a execução ou restauração de cópias de segurança é concluída em uma pequena fração do tempo que levaria com os servidores físicos tradicionais. No caso de falhas na aplicação ou serviço, uma instância de *backup* (*hot backup*) pode assumir imediatamente o lugar da faltosa. Neste caso, a interrupção no serviço pode passar despercebida para os usuários. As garantias fornecidas pelo provedor para seus consumidores podem ser definidas através de contratos em nível de serviço – SLAs (*Service Level Agreements*) [Kandukuri et al. 2009].

A maioria dos provedores de nuvem oferecem garantias de tempo de atividade (*uptime*) em seus SLAs. Um dos problemas relacionados ao SLA é a dificuldade para expressar e implementar o contrato em nível computacional – e.g. como fornecer garantias de tempo de atividade para uma transação se esta envolve um fluxo de dados através da Internet. As empresas tradicionais ainda possuem dúvidas com relação a transferência de

seus dados internos para fora dos limites de seu filtro de pacotes (*firewall*). Certos países têm suas próprias regras quanto ao lugar onde as empresas podem armazenar seus dados, por exemplo, Canadá e Estados Unidos. Para tentar contornar essas barreiras, provedores de nuvem estão implantando *data centers* em várias partes do mundo.

Empresas que optarem por utilizar um determinado provedor de computação em nuvem podem basear suas escolhas em vários fatores: preço, confiabilidade, disponibilidade, abrangência, suporte etc. Mesmo que um único provedor atenda estes requisitos, outros provedores podem ser utilizados, fornecendo diferentes conjuntos de funcionalidades, como por exemplo: armazenamento da *Amazon*, poder computacional do *Google* e CRM (*Customer Relationship Management*) da *Salesforce*. Tudo isto pode ser feito através de APIs de acesso padronizadas, permitindo que as aplicações permaneçam inalteradas, sendo necessário alterar apenas os provedores de computação em nuvem.

Apesar da existência de grandes provedores de computação em nuvem (e.g. *Amazon*, *Google*, *Saleforce*) existem algumas empresas menores participando deste mercado (e.g. *RightScale*, *Hyperic*, *Mosso*, *Elastra*). Porém, algumas destas companhias pode não sobreviver à concorrência, abandonando o mercado ou sendo adquiridas por outras empresas. Neste caso, os dados e aplicações de seus consumidores e usuários precisam ser transferidos para outro provedor, ou retornar para dentro dos limites organizacionais do consumidor. A padronização neste caso é fundamental, fornecendo um nível de garantia adicional aos consumidores – além dos contratos em nível de serviço.

O tráfego de dados entre a nuvem computacional e o consumidor ou entre serviços hospedados em diferentes provedores gera latência, que fica ainda maior se comparada com o acesso local ao *data center* de uma empresa tradicional. A tecnologia para transferências de dados que muitas nuvens computacionais disponibilizam é baseada nos métodos *get/post* do protocolo HTTP. Porém, este protocolo não foi projetado para atender este tipo de demanda. Para auxiliar na resolução deste problema, a conexão entre o consumidor e o provedor deve ser geograficamente a mais próxima possível. Abordagem esta que é contraditória a proposta da computação em nuvem – fornecer serviços de maneira transparente à localização.

No caso de um *data center* corporativo tradicional ser atacado, o efeito do dano será sentido somente neste domínio, ou por algumas entidades que tenham negócios com esta organização. Porém, se um grande domínio for atacado (e.g. *Amazon*, *Google*) o efeito será percebido por todos, possivelmente nas interfaces dos softwares dos consumidores. Adicionalmente, estes hospedam grandes grupos de consumidores e possivelmente os sistemas de empresas inteiras. A ocorrência de alguma falha pode ter um grande impacto nos negócios de várias entidades. Alguns provedores de nuvem tentam minimizar estes problemas replicando seus *data centers* – dados e aplicações de seus usuários. Assim, a falha em um *data center* não deverá paralisar todos os sistemas, mas o desempenho de algumas aplicações poderá ser afetado.

O uso da computação em nuvem também pode ser percebido em meios científicos e acadêmicos, sendo geralmente implantada em centros de computação de alto desempenho, *High Performance Computing* – HPC [Ogrizovic et al. 2010]. HPCs acadêmicos estão expandindo suas infraestruturas através da virtualização dos *clusters* locais. Esta abordagem permite o fornecimento de ambientes personalizados para uma grande variedade de

consumidores, atendendo as necessidades específicas de cada tipo de demanda.

De maneira geral, o modelo de computação em nuvem visa prover acesso sob demanda a diferentes camadas da plataforma computacional (e.g. rede, servidores, armazenamento, aplicações, serviços etc.). Estas funcionalidades podem ser rapidamente fornecidas ou liberadas com pouco esforço de gerenciamento ou interação humana. A nuvem visa fornecer alta disponibilidade e elasticidade, sendo composta de cinco características essenciais [Mell e Grance 2009]:

1. *Auto-Atendimento*: o consumidor configura cada recurso computacional conforme sua necessidade, sem exigir interação humana com os provedores de serviço.
2. *Amplo acesso à rede*: os recursos são disponibilizados na rede e acessados através de mecanismos padronizados. Isto possibilita o uso em diferentes plataformas (e.g. celulares, notebooks etc.).
3. *Pool de recursos*: os recursos computacionais do provedor são agrupados. Isto permite servir múltiplos consumidores em um modelo multi-inquilino (*multi-tenant*). Ou seja, os recursos físicos e virtuais são distribuídos e ou redistribuídos dinamicamente de acordo com a demanda do consumidor.
4. *Elasticidade*: os recursos podem ser fornecidos rapidamente e em alguns casos automaticamente. A quantidade de recursos disponibilizados passa para o consumidor a impressão de que a nuvem possui uma infraestrutura ilimitada.
5. *Medição no uso dos serviços*: a nuvem controla e otimiza o uso de recursos fornecendo métricas de acordo com o tipo de serviço sendo fornecido. Tanto o provedor quanto o consumidor podem monitorar e controlar a utilização dos recursos.

2.2.2. Serviços, plataformas e infraestrutura

Ambientes de computação em nuvem são similares a sistemas distribuídos que executam processamento de dados. Os critérios utilizados para definir as diferentes abordagens de nuvens podem ser [Rimal et al. 2009]: a) *arquitetura da nuvem* – serviços disponíveis na Internet através de um ponto de entrada que fornece acesso aos recursos computacionais para implementá-los, b) *virtualização* – tecnologia que abstrai o acoplamento entre o sistema e o hardware, c) *serviços* – implementados por provedores como: *SalesForce*, *Microsoft Azure*, *Amazon* etc., d) *tolerância a faltas* – técnicas adotadas para fornecer serviços confiáveis, e) *segurança* – proteção dos dados processados e armazenados, f) *balanceamento de carga* – redistribuição de carga de trabalho ou redirecionamento das solicitações de acesso, g) *interoperabilidade* – definição de interfaces para permitir a portabilidade de aplicações entre nuvens, h) *armazenamento escalável de dados* – transferência de dados para a nuvem sem preocupação com a forma de armazenamento ou cópias de segurança, i) *escalabilidade horizontal/vertical*: a escalabilidade horizontal é denotada pelo que a nuvem fornece através do balanceamento de carga e a escalabilidade vertical esta relacionada a quantidade de recursos utilizados.

Basicamente os provedores de computação em nuvem podem ser classificados de acordo com o tipo de serviço oferecido e o respectivo modelo de implantação [Mell e Grance 2009]. Os modelos de serviço são (Figura 2.1):

1. *Software como um Serviço* (SaaS): oferece o produto final de computação em nuvem, o software que o consumidor usa. O consumidor não gerencia ou controla a infraestrutura – rede, servidores, sistema operacional, armazenamento e funcionalidades de aplicações. O consumidor usa as aplicações sendo executadas na infraestrutura da nuvem. As aplicações podem ser acessadas através de um *thin client* – navegador web – por exemplo.

2. *Plataforma como um Serviço* (PaaS): oferece recursos para o consumidor implantar na infraestrutura da nuvem suas próprias aplicações, desde que utilizem linguagens de programação e ferramentas suportadas pelo provedor. Nesta abordagem, o consumidor não gerencia ou controla a infraestrutura subjacente – rede, servidores, armazenamento – mas somente controla sua própria aplicação e o sistema que está hospedando as configurações do ambiente.

3. *Infraestrutura como um Serviço* (IaaS): fornece recursos computacionais básicos como processamento, armazenamento, rede etc. Com estes recursos o consumidor pode implantar e executar uma grande variedade de programas – sistemas operacionais e seus aplicativos. Neste nível o consumidor não controla ou gerencia a infraestrutura física da nuvem, mas tem controle limitado sobre componentes da rede, como por exemplo, filtros de pacotes.

- *Identidade para a Nuvem como um Serviço* (IDaaS): o *Cloud Security Alliance* considera o IDaaS um serviço de gerenciamento de identidades para a nuvem, sendo externo as aplicações e aos provedores que utilizam as identidades [CSA 2010a]. O IDaaS é um serviço que fornece gerenciamento de identidade e do ciclo de vida dos usuários, funções de controle de acesso, *Single Sign-On* etc. Este serviço pode ser utilizado pelos modelos SaaS, PaaS e IaaS.

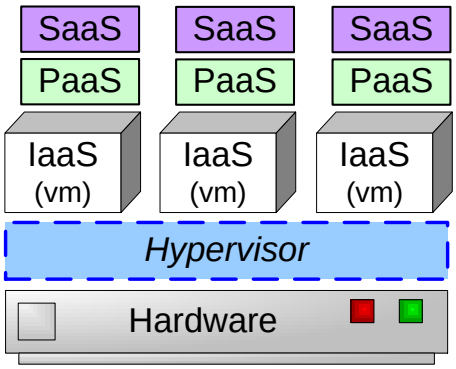


Figura 2.1. Modelos de Serviço.

A computação em nuvem pode ser vista como uma pilha de serviços. Cada camada da pilha oferece serviços construídos com base nas camadas inferiores. Alguns serviços básicos, como medição, contabilização e gerenciamento, se espalham pelas várias camadas, pois são necessários para todos os tipos de serviços oferecidos na nuvem.

A camada IaaS abrange toda a pilha de recursos da infraestrutura – desde as instalações físicas até as plataformas de hardware disponíveis. Essa camada incorpora a

funcionalidade de abstração de recursos, possibilitando a conectividade entre a infraestrutura física e a lógica. A camada PaaS – posicionada acima do IaaS – adiciona um nível de integração com *frameworks* de desenvolvimento de aplicações e funcionalidades de *middleware*. Essa camada provê funções como banco de dados e recursos para troca e enfileiramento de mensagens, permitindo aos desenvolvedores a construção de aplicações sobre o PaaS. As linguagens de programação e as ferramentas utilizadas precisam ser suportadas pelas camadas subjacentes. A camada SaaS – posicionada acima do IaaS e PaaS – provê um ambiente de operação autocontido, utilizado para disponibilizar diferentes serviços para seus consumidores, por exemplo, conteúdos, aplicações, funcionalidades de gerenciamento etc.

Os quatro principais modelos de implantação que podem ser aplicados a computação em nuvem são [Mell e Grance 2009]:

1. *Nuvem privada*: a infraestrutura é operada exclusivamente para atender as necessidades de uma organização, sendo que essa pode ser gerenciada pela organização ou por um terceiro, e sua implementação pode ser local ou remota.

2. *Nuvem baseada em comunidade*: compartilhada por várias organizações que possuem interesses comuns – requisitos de segurança, políticas etc. Esta pode ser gerenciada pelas organizações participantes da comunidade ou por um terceiro, em implementação local ou remota.

3. *Nuvem pública*: a infraestrutura é disponibilizada para o público em geral, podendo pertencer a alguma organização que vende serviços de computação.

4. *Nuvem híbrida*: é uma composição entre dois ou mais modelos de nuvens, por exemplo, privado e público. Estes permanecem como entidades únicas, porém, são ligados por alguma tecnologia especifica – padronizada e aberta ou proprietária. Uma composição de nuvem híbrida pode viabilizar o balanceamento de carga, ou seja, quando a parte privada não consegue mais atender a demanda a parte pública pode fazer esta tarefa – se os dados não forem sensíveis.

Os provedores oferecem seus serviços de forma distinta, diferenciando-se na forma como o consumidor pode obter e como este é capaz de acessar os serviços [Zhang et al. 2010]. A *Amazon* oferece instâncias de máquinas virtuais – que podem ser *Linux*, *Solaris* ou *Windows* – sendo o consumidor livre para instalar suas próprias aplicações. Serviços de armazenamento, banco de dados e gerenciamento de conteúdo também são disponibilizados pela *Amazon*. O *Google* disponibiliza algumas funcionalidades de seu sistema através de uma interface baseada em *Python*, não fornecendo acesso a imagens de um sistema operacional ou qualquer tipo de banco de dados padrão. A *Google* fornece APIs para que os consumidores escrevam aplicações e acessem serviços como correio eletrônico, armazenamento de dados proprietário etc. O provedor *Force.com* é comparável ao *Google* no modelo de negócio, permitindo que o desenvolvedor construa uma aplicação usando um mecanismo de *workflow* e deixando a implementação subjacente da aplicação para a nuvem computacional.

A Arquitetura Orientada a Serviço juntamente com as tecnologias de virtualização podem ser utilizadas para criar um modelo de computação em nuvem reutilizável e configurável – uma arquitetura aberta para nuvens computacionais (*Cloud Computing Open Architecture* – CCOA) [Zhang e Zhou 2009].

Os principais aspectos a serem considerados na definição de uma arquitetura para computação em nuvem podem ser brevemente descritos como: criação de um projeto que possa ser reutilizado e que aplique plataformas de configuração escaláveis; utilização da orientação a serviço e da virtualização para agregar valores práticos e de negócios para as aplicações, sistemas e processos de negócios; modelagem de um conjunto de serviços que seja comuns para as plataformas de nuvens; maximização do valor dos negócios – aplicação de infraestruturas de tecnologia e sistemas de gerenciamento extensíveis.

Para a infraestrutura o mecanismo de virtualização é uma das principais abordagens, por permitir a flexibilização do uso da camada de hardware [Laureano e Maziero 2008]. As máquinas virtuais proveem ambientes de processamento independentes e isolados, podendo ser instanciadas e destruídas sob demanda. Dessa forma, o ambiente de máquinas virtuais constitui uma base bastante adequada para a construção de infraestruturas de computação em nuvem [Grobauer et al. 2010].

2.2.3. Ambientes de computação em nuvem

O consumidor que necessitar de uma grande quantidade de recursos computacionais poderia ter que conectar-se a vários provedores de recursos diferentes para poder satisfazer sua demanda. Assim, o *pool* de recursos fornecido pode ser bastante heterogêneo, tornando a tarefa de utilização complexa, pois envolve o gerenciamento de diferentes contratos, políticas, interfaces, usuários etc. A maioria dos consumidores preferem um ambiente onde os recursos de hardware, o ambiente de programação e o conjunto de programas e protocolos sejam homogêneos. Um ambiente homogêneo torna o desenvolvimento de aplicações de grande escala mais fácil e acessível. A seguir são apresentados dois ambientes para a computação em nuvem – *VMWare VSphere*, que é proprietário e o *Eucalyptus*, que é *open-source* – ambos tentam oferecer a homogeneidade e as facilidades esperadas pelo consumidor.

2.2.3.1. Eucalyptus

O *framework Eucalyptus* oferece um IaaS para computação em nuvem, composto de vários componentes que interagem entre si através de interfaces bem definidas [Eucalyptus 2010]. A nuvem implementada pelo *Eucalyptus* aborda: agendamento e instanciação de máquinas virtuais (*Virtual Machine* - VM), armazenamento de dados e imagens de VMs, interfaces de administração e de consumidor para a nuvem, construção de redes virtuais, e definição e execução de SLAs.

O *Eucalyptus* implementa ambiente operacional para a nuvem que é independente do tipo de hipervisor – que atualmente pode ser Xen ou KVM. *Eucalyptus* foi projetado para ser o menos intrusivo possível, sendo bastante modular, baseado em padrões da indústria e com mecanismos de comunicação independentes de linguagem. A interface externa do *framework* é baseada em uma API desenvolvida pela *Amazon*. Adicionalmente,

a nuvem computacional fornece uma rede virtual sobreposta (*overlay*) que isola o tráfego de diferentes consumidores e permite que dois ou mais *clusters* (agrupamentos de máquinas físicas) pareçam pertencer à mesma rede local.

O *framework* utiliza emulação das interfaces SOAP (*Simple Object Access Protocol*) e *Query* do *Amazon EC2* (*Amazon Elastic Compute Cloud*), permitindo que os consumidores iniciem, controlem, acessem e finalizem VMs. Os consumidores interagem com o *Eucalyptus* utilizando as mesmas ferramentas e interfaces utilizadas para efetuar a interação com o *Amazon EC2*. Cada componente do sistema é implementado como um serviço *web* independente. Esta abordagem possui os seguintes benefícios: cada serviço *web* expõe um documento WSDL (*Web Services Description Language*) bem definido que permite a geração de uma API para qualquer linguagem; características já implantadas em serviços *web* podem ser utilizadas para prover comunicação segura entre os componentes.

Em uma nuvem *Eucalyptus* existem quatro componentes de alto nível (Figura 2.2), cada um com sua interface de serviço *web* [Nurmi et al. 2009]: 1) *Node Controller* – NC; 2) *Cluster Controller* – CC; 3) *Storage Controller* – *Walrus* e 4) *Cloud Controller* – CLC.

Node Controller

O *Node Controller* (NC) é executado em todo nó que hospeda uma VM, pesquisando e gerenciando o sistema hospedeiro e o hipervisor do nó, e respondendo às solicitações do *cluster controler* (evento *ger*, Figura 2.2). O NC consulta os recursos físicos do nó (e.g. número de núcleos, tamanho da memória, espaço disponível em disco, etc.) assim como informações sobre o estado das instâncias das VMs. As informações coletadas são propagadas para o *cluster controler* em resposta a solicitações de informações. Mediante a verificação da autorização e depois da confirmação da disponibilidade dos recursos, o NC executa as solicitações com o auxilio do hipervisor. Para iniciar uma instância de uma VM o NC executa uma cópia dos arquivos referente a imagem da mesma para o nó local – a partir de um repositório de imagens remoto ou de um *cache* local – criando um novo *endpoint* na rede virtual sobreposta e informando o hipervisor para inicializar tal instância. Para finalizar a instância, o NC informa o hipervisor para finalizar a VM, desfazer a rede e limpar os arquivos associados com tal instância.

Cluster Controller

O *Cluster Controller* (CC) geralmente é executado em uma máquina que é a porta de entrada (*front-end*) para o *cluster*, ou qualquer máquina que possua conectividade de rede com ambos os nós NC e CLC. Muitas das operações do CC são similares as operações do NC, porém, o volume de operações é maior. As funções primárias do CC são (evento *ad*, Figura 2.2): agendar solicitações para a execução de instâncias em NCs específicos, controlar a rede virtual sobreposta e recuperar/enviar informações sobre um conjunto de NCs. Quando o CC recebe uma solicitação para executar um conjunto de instâncias, este verifica cada NC (evento *ger*) e envia solicitações de execução de instâncias para o primeiro NC que tiver recursos livres suficientes para hospedar a instância. Quando o CC recebe uma solicitação para descrever os recursos (evento *ad*), este recebe uma lista

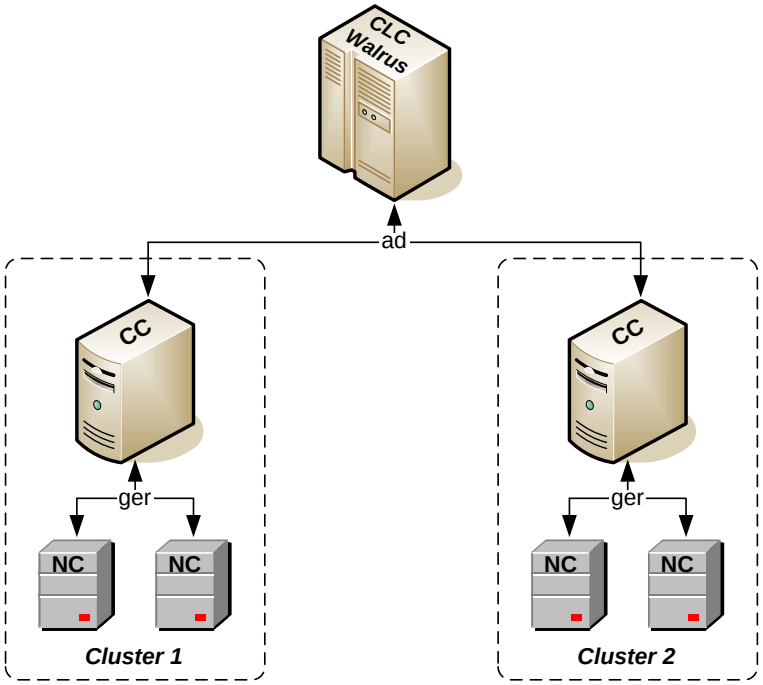


Figura 2.2. Visão geral do ambiente Eucalyptus.

das características do recurso desejado (e.g. núcleos, memória, disco) descrevendo os requisitos necessários para uma determinada instância. Com esta informação o CC calcula quantas instâncias simultâneas de um tipo específico podem ser executadas em sua coleção de NCs, reportando este número para o CLC.

Cloud Controller

Os recursos virtualizados que compõem a nuvem *Eucalyptus* são expostos e gerenciados pelo *Cloud Controller* (CLC). O CLC (evento *ad*, Figura 2.2) possui uma coleção de serviços web que são agrupados de acordo com seus papeis em três categorias: 1) *Resource Services*: executa a arbitragem do sistema para a alocação de recursos, permite que usuários manipulem propriedades de VMs e redes, e monitora os componentes do sistema e os recursos virtuais; 2) *Data Services*: administra dados do consumidor e do sistema *Eucalyptus*, fornece um ambiente consumidor personalizável para o estabelecimento das propriedades de alocação dos recursos; 3) *Interface Services*: fornece interfaces para os usuários, tratamento de autenticação e tradução de protocolos, e expõe as ferramentas de gerenciamento do sistema.

Storage Controller

O *Storage Controller* (*Walrus*) é um serviço de armazenamento de dados que estende as tecnologias padrão para serviços *web* (*Axis2*) tendo compatibilidade de interface com o *Amazon S3* (*Simple Storage Service*). *Walrus* implementa a interface REST (*Representatio-*

nal State Transfer, algumas vezes chamada de *Query*) assim como interfaces SOAP, ambas compatíveis com o S3. O armazenamento possui dois tipos de funcionalidades: 1) criar *streams* de dados fluindo para dentro/fora da nuvem, assim como a partir de instâncias inicializadas em nós; 2) funcionar como um serviço de armazenamento para imagens de VMs. Os arquivos que são utilizados para instanciar VMs em nós podem ser enviados para o *Walrus* e acessados a partir dos nós.

No projeto *Eucalyptus* a solução de rede utilizada para as instâncias de VMs aborda: conectividade, isolamento e desempenho. Toda VM controlada pelo *Eucalyptus* deve ter conectividade de rede com as outras e pelo menos uma das instâncias que fazem parte de um conjunto de VMs deve ter conectividade com a Internet. Com isto, o proprietário do conjunto pode efetuar *login* na instância conectada com a Internet e controlar as demais VMs. Os usuários possuem acesso de administrador nas respectivas VMs instanciadas e nas interfaces de rede. Esta funcionalidade pode causar preocupações à segurança, sendo que sem os devidos cuidados, a VM de um usuário pode adquirir qualquer endereço IP do sistema, causando interferência na rede do sistema ou em outras VMs que estiverem compartilhando o mesmo recurso físico. Assim, em uma nuvem compartilhada por diferentes usuários, VMs pertencentes a um único domínio devem ser capazes de se comunicar, porém VMs pertencentes a outros domínios devem ficar isoladas.

O *Eucalyptus Cluster Controller* (CC) é responsável por configurar e desfazer instâncias das interfaces de redes virtuais em três modos distintos: 1) neste modo o CC incorpora a interface de rede das VMs diretamente numa ponte *Ethernet* implementada em software – que está conectada a rede da máquina física. Isto permite ao administrador tratar pedidos DHCP da rede de VMs do mesmo modo que são tratados os pedidos DHCP que não fazem parte do *Eucalyptus*; 2) neste modo o administrador define endereços estáticos para o par IP/MAC – cada nova instância criada pelo sistema recebe um par IP/MAC livre, que é liberado quando a instância é finalizada. Nos modos 1 e 2, o desempenho de comunicações entre VMs é similar ao nativo – quando as VMs estão sendo executadas no mesmo *cluster* (*overhead* no desempenho pode ser imposto pelo hipervisor), porém neste caso não é provido o isolamento de rede entre VMs; 3) neste modo o *Eucalyptus* gerencia e controla as redes das VMs, fornecendo isolamento de tráfego entre VMs, definição das regras de entrada entre conjuntos lógicos de VMs e a atribuição dinâmica de endereços de IPs públicos para VMs durante o *boot* ou em tempo de execução.

2.2.3.2. VMware VSphere

O *VSphere* é vendido pela empresa *WMware* como um sistema operacional para computação em nuvem [VMWare Inc 2010]. *VSphere* é baseado no *VMware ESX/ESXi* que consiste em dois componentes interagindo entre si para fornecer um ambiente de virtualização robusto e dinâmico: o *Service Console* e o *VMkernel*.

O *Service Console* possui as funções de sistema operacional, usado para interagir com o *VMware ESX* e as máquinas virtuais que estão sendo executadas no servidor físico. O *Service Console* é derivado do Linux e inclui serviços encontrados em SOs tradicionais – *firewall*, agentes *Simple Network Management Protocol* (SNMP) e servidor *web*. Esse componente, apesar de definido pela *VMWare* como um SO de nuvem, apenas inclui os

serviços de SO para suportar a virtualização. O *Service Console* fornece acesso ao segundo componente, o *VMkernel* – base real do processo de virtualização. O *VMkernel* gerencia o acesso das máquinas virtuais ao hardware subjacente, escalonando o acesso ao processador e gerenciando a memória.

O *VMware ESX/ESXi* é um hipervisor do tipo nativo (*bare-metal*), instalado diretamente sobre o hardware e dividindo este com VMs que podem ser executadas simultaneamente, compartilhando os recursos físicos do servidor. Cada VM representa um sistema completo com processador, memória, rede, armazenamento, BIOS etc., podendo executar um SO tradicional com suas respectivas aplicações.

O *VMware vCenter Server* (*vCS* na Figura 2.3) é um utilitário para o gerenciamento centralizado de todos os *hosts ESX/ESXi* e suas respectivas VMs (evento *ger*). O aplicativo *vCenter Server* é baseado em *Windows* e permite aos administradores de TI: implantar, gerenciar, monitorar, automatizar e proteger a infraestrutura virtual. O banco de dados de suporte utilizado pelo *vCenter Server* – que pode ser o *Microsoft SQL Server* ou *Oracle* – armazena todos os dados sobre os *hosts* e as VMs.

O *VMware vSphere Client* (*vSC* na Figura 2.3) é uma aplicação baseada em *Windows* que permite o gerenciamento direto de *hosts ESX/ESXi* (evento *dir*) ou via *vCenter Server* (evento *ad*). O *vSphere Client* é uma interface gráfica usada para o gerenciamento de tarefas cotidianas e para a configuração da infraestrutura virtual. Se o *vSphere Client* for utilizado para se conectar diretamente a um *host ESX/ESXi*, esse exige o uso de uma conta de usuário daquele *host*, enquanto se for usado o *vSphere Client* para se conectar a um *vCenter Server* é exigida uma conta no servidor *Windows*.

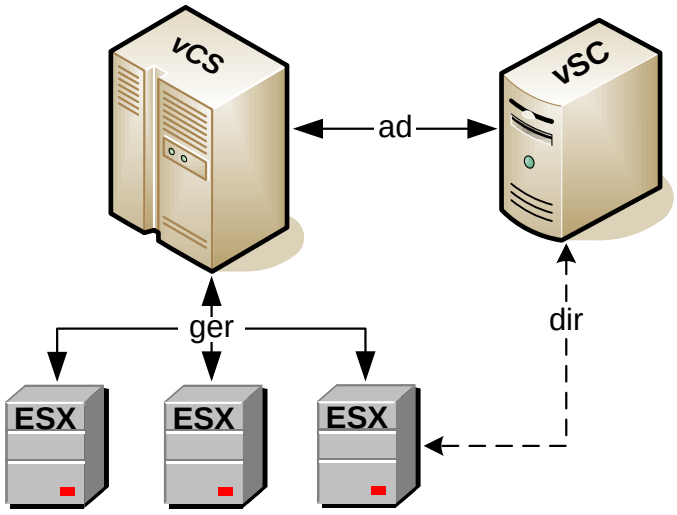


Figura 2.3. O ambiente VMware VSphere.

Todas as tarefas de gerenciamento disponíveis na conexão direta com um *host ESX/ESXi* estão disponíveis quando é feita a conexão com um *vCenter Server*, porém, o oposto não é verdadeiro. As funcionalidades de gerenciamento disponíveis através de um *vCenter Server* são mais significativas e superam as funcionalidades de se conectar diretamente a um *host ESX/ESXi*.

2.2.4. Principais diferenças entre computação em nuvem e em grid

A computação em nuvem não é um conceito completamente novo, estando relacionada com *grid*, *utility computing*, *cluster* e sistemas distribuídos [Foster et al. 2008, Zhang et al. 2010]. O modelo de *grid* está relacionado ao modelo de nuvem, porém, não necessariamente como seu antecessor [Nurmi et al. 2009]. Computação em nuvem e *grid* compartilham uma visão semelhante nos seguintes aspectos: a orientação a serviço, redução de custos computacionais, aumento da confiabilidade e da flexibilidade, e terceirização de tarefas de processamento. Ambas as abordagens podem ser úteis ao mesmo grupo de consumidores (e.g. pesquisadores executando computação paralela fracamente acoplada).

Apesar das semelhanças, *grid* e computação em nuvem diferem principalmente na arquitetura, pois em *grid* os pedidos de consumidores individuais podem (e deveriam) consumir grandes frações do *pool* total de recursos. Em computação em nuvem, frequentemente, se limita o tamanho de uma solicitação individual para ser uma pequena fração da capacidade total disponível no *pool* de recursos, porém tendo como objetivo ser escalável para atender um grande número de consumidores simultâneos. Nuvem e *grid* também compartilham problemas semelhantes, tendo dificuldades para: gerenciar grandes instalações, definir métodos para que consumidores possam procurar e interagir com novos recursos e serviços, e implementar computação paralela capaz de utilizar os recursos e serviços [Foster et al. 2008].

Os principais fatores que contribuem para a ampliação do interesse na computação em nuvem são: 1) aumento do poder computacional e capacidade de armazenamento; 2) crescimento exponencial na quantidade de dados gerados em pesquisas científicas e disponíveis na Internet, sendo constantemente publicados e recuperados; e 3) ampla adoção de *services computing* e aplicações Web 2.0.

Em ambientes de computação em nuvem temos a possibilidade de comprar acesso sob demanda para centenas de computadores em dezenas de *data centers* espalhados pelo mundo (e.g. *Amazon*, *Google*, *Microsoft*). Computação em nuvem é escalável, podendo ser vista como uma entidade que entrega diferentes níveis de serviço para seus consumidores. Essa é impulsionada pelo crescimento econômico, fornecendo serviços que podem ser dinamicamente configurados e entregues sob demanda.

A abordagem de *grid* surgiu com o intuito de resolver problemas computacionais de grande escala, usando uma rede de máquinas comuns que compartilham recursos. Ou seja, uma infraestrutura distribuída que fornece recursos de armazenamento e processamento. Para suportar a criação de organizações virtuais, por exemplo, *grids* fornecem *middlewares*, ferramentas e um conjunto de protocolos padrão que permitem a construção de serviços. Interoperabilidade e segurança são as principais preocupações em *grid*, visto que os recursos podem vir de diferentes domínios administrativos, possuírem políticas de uso globais/locais dedicadas, diversas configurações para o *hardware*, *software* e plataforma, com disponibilidade e capacidade variadas [Pinheiro Jr e Kon 2005].

A computação em nuvem implementa a abordagem proposta pelo modelo de *utility computing*: um modelo de negócio em que recursos computacionais – processamento, armazenamento – são empacotados e contabilizados como um serviço público que está

sendo consumido em nossas residências – e.g. energia elétrica ou água [Zhang et al. 2010]. No modelo de negócios utilizado pela computação em nuvem o consumidor paga ao provedor pelo consumo de determinado software e não pela licença do mesmo. O modelo confia no crescimento econômico para baixar os preços e aumentar os lucros.

A *Amazon* fornece o *Compute Cloud EC2* (cobrado com base no tempo de consumo) e o *Data Cloud S3* (cobrado por consumo de Gigabytes por mês). A transferência de dados é cobrada por consumo de Terabytes por mês. O modelo de negócios utilizado em *grid* é geralmente orientado a projetos, no qual usuários possuem unidades de serviço que podem ser consumidos – horas de processador, por exemplo. Quando uma instituição se associa ao *Teragrid* com seus recursos, por exemplo, ela tem conhecimento de que outros integrantes da comunidade podem utilizar os seus recursos.

No que diz respeito a federação de infraestruturas distintas, *grid* utiliza uma abordagem baseada em *middleware* como maneira de fornecer federações de recursos entre domínios administrativos cooperativos, porém separados geograficamente. Em nuvem, os serviços são distintos e não federados. Um provedor de computação em nuvem geralmente é operado por uma única entidade, com autoridade administrativa suficiente para estabelecer configuração uniforme e políticas adequadas.

A utilização de um *middleware* de *grid* juntamente com o modelo de computação em nuvem também pode ser feita, realizando assim tarefas de propósitos gerais em ambientes virtualizados [Caron et al. 2009]. Ou seja, a nuvem oferece recursos computacionais sob demanda enquanto o *middleware* de *grid* pode ser utilizado para gerenciar os recursos.

Muitos *grids* utilizam um modelo baseado em agendamento (*batch-scheduled*) em que um gerente de recursos locais (e.g. PBS, *Condor*) administra os recursos computacionais num determinado ambiente. Os usuários submetem trabalhos (e.g. *batch jobs*) solicitando algum recurso. Os trabalhos, geralmente, são agendados e enfileirados para processamento posterior, visto que alguns *grids* não suportam aplicações interativas. Para certos tipos de trabalho as decisões a serem tomadas a cerca do agendamento são muito caras em termos computacionais ou de tempo. Em computação em nuvem, os recursos computacionais são compartilhados por todos os usuários simultaneamente.

Um dos principais desafios para ganhar escalabilidade de maneira eficiente é a localização dos dados em relação aos recursos computacionais disponíveis. Para ter uma boa escalabilidade em computação em *grid* e nuvem, os dados devem ser distribuídos entre muitos computadores, sendo que o processamento deve ser executado visando reduzir custos de comunicação. Porém, *grid* geralmente utiliza sistemas de arquivos compartilhados (e.g. NFS, GPFS), que não consideram a questão de proximidade para fazer o armazenamento de dados.

A virtualização é um item indispensável para a computação em nuvem, fornecendo abstração e encapsulamento de dados e aplicações em um determinado domínio. Em geral, *grids* possuem um modelo de confiança diferente, no qual é utilizado delegação de identidade, para poder acessar recursos em diferentes domínios de um *grid* (e.g. *Ganglia*). Os recursos em *grid* não são abstraídos e virtualizados. Em nuvem, diferentes níveis de serviço são oferecidos, neste ambiente o usuário tem acesso a uma API, sendo que os recursos de baixo nível ficam escondidos (principalmente nos modelos SaaS e PaaS). As

informações retornadas para o usuário são limitadas, não fornecendo muitos detalhes sobre o estado do recurso [Foster et al. 2008].

Em computação em nuvem, trilhas de auditoria deixadas pelos processos podem ser utilizadas para rastrear a execução de um serviço desde sua fonte de dados, utilização de dados intermediários e procedimentos aplicados. Essa informação é vital para o entendimento, a descoberta, a validação de dados e processos. Estas informações também são úteis para encontrar erros na execução de fluxos de trabalho, validar ou invalidar resultados e servir de guia para projetos futuros. Em *grids*, o gerenciamento deste item, geralmente, está embutido em sistemas de fluxo de trabalho (e.g. *Chimera*, *Swift*) ou é provido como um serviço autônomo (e.g. *PreServ*). A aplicação deste tipo de gerenciamento em nuvem é de suma importância, visto que esta pode se expandir entre vários provedores de serviço, diferentes plataformas, políticas de acesso e camadas de hardware e software.

Grids têm por objetivo o processamento científico de grande escala, abrangendo e gerenciando uma grande quantidade de recursos (recursos que podem ser heterogêneos e instáveis). O modelo de programação MPI (*Message Passing Interface*) é o mais comumente utilizado – tarefas que usam a memória local da máquina durante o processamento e se comunicam com outras tarefas através do envio/recebimento de mensagens. Linguagens de coordenação permitem que componentes heterogêneos troquem dados, oferecendo facilidades para a estruturação dinâmica de componentes distribuídos (e.g. *Linda*). O sistema de fluxo de trabalho permite a composição dos passos individuais em um gráfico de dependência (e.g. *Swift*).

MapReduce é um modelo de programação paralela que fornece um sistema em tempo de execução para o processamento de grandes conjuntos de dados. Este é baseado nas funções "map" (aplica operações de mapeamento em um conjunto de itens) e "reduce" (executa a divisão de um conjunto de itens) [Grossman 2009]. O *MapReduce* particiona automaticamente dados de entrada e agenda a execução de programas em *clusters* de máquinas. Essa abordagem pode utilizar as máquinas virtuais fornecidas pela computação em nuvem.

Grids geralmente suportam aplicações do tipo HPC (*high performance computing*) e HTC (*high throughput computing*). Esta infraestrutura também suporta *gateways* científicos, que são *front-ends* para uma variedade de aplicações que podem ser fraca ou altamente acopladas. As aplicações para a nuvem caracterizam-se por serem fracamente acopladas, orientadas a transação e interativas (*grids* geralmente usam processamento em *batch*). As tecnologias Web 2.0 e os navegadores *web* têm um papel central na interação dos usuários com as nuvens computacionais.

A maioria das nuvens computacionais abrange *data centers* dedicados, que fazem parte da mesma organização, sendo que as configurações de hardware/software e as plataformas de suporte são homogêneas se comparadas aos ambientes de *grid*. Assim, a interoperabilidade pode se tornar um sério problema para comunicação entre *data centers*, interações que cruzam domínios administrativos etc.

Grids já são construídos considerando a heterogeneidade e o dinamismo dos recursos, onde cada domínio possui sua própria administração, operando de forma autônoma. Assim, aspectos de segurança já foram planejados desde a infraestrutura básica para supor-

tar: *single-sign-on*, delegação, privacidade, integridade, alocação coordenada de recursos, e reserva e compartilhamento de recursos. Ambientes de computação em nuvem ainda estão desenvolvendo estes aspectos.

Computação em *grid* fornece protocolos e serviços em cinco camadas diferentes [Foster et al. 2008]: 1) *fabric layer*: fornece acesso a diferentes tipos de recursos – processamento, rede, armazenamento. Geralmente conta com um gerente de recursos locais (e.g. PBS, Condor), componentes de propósito geral (e.g. GARA), e serviços especializados para o gerenciamento de recursos (e.g. Falkon); 2) *connectivity layer*: define os protocolos de comunicação e autenticação; 3) *resource layer*: define os protocolos para publicação, descoberta, negociação, monitoramento, contabilização e pagamento das operações compartilhadas pelos recursos (e.g. GRAM, gridFTP); 4) *collective layer*: captura interações entre coleções de recursos, serviços de diretório (e.g. MDS), agendamento e intermediação de serviços (e.g. Condor-G, Nimrod-G), sistemas de programação para grid (e.g. MPICH), e políticas para recursos (e.g. CAS); 5) *application layer*: engloba aplicações do usuário construídas em cima dos protocolos e APIs, operando em ambientes de organizações virtuais (e.g. *National Virtual Observatory*, *Teragrid Science gateway*).

Nuvens computacionais são normalmente vistas como um grande *pool* de recursos computacionais, podendo ser acessada via protocolos padrão através de uma interface abstrata – recursos e serviços podem ser fornecidos através de interfaces de Serviços Web. O ambiente de nuvem é projetado para resolver problemas computacionais na escala da Internet, sendo composta basicamente de quatro camadas [Foster et al. 2008]: 1) *fabric*: em nível de hardware – IaaS; 2) *unified resource*: recursos abstraídos ou encapsulados (e.g. virtualização) – IaaS; 3) *platform*: ferramentas especializadas, *middleware* e serviços – PaaS; 4) *application*: softwares executados na nuvem – SaaS.

2.2.5. Problemas em Aberto

Um dos obstáculos para a adoção massiva da computação em nuvem como um modelo de negócio é o capital que empresas tradicionais já investiram no passado na aquisição de hardware e licenças de software. Combinando estes fatores com os sistemas que foram desenvolvidos para serem executados nas infraestruturas já estabelecidas – sistemas legados – se chega a uma realidade onde parte ou a totalidade dos sistemas precisa ser re-projetado/reescrito para poder utilizar as APIs fornecidas pela computação em nuvem. Este problema existe, por exemplo, no dias atuais em grandes organizações do setor bancário, que tem sistemas do século XX desenvolvidos com tecnologias legadas (como ADABAS ou NATURAL, por exemplo) – dificilmente estes sistemas serão portados para outra plataforma. Adicionalmente, pode não haver uma reposição um-para-um para todos os sistemas sendo executados dentro dos limites organizacionais, portanto há casos em que as aplicações teriam que ser repensadas/desenvolvidas.

Andando na contra-mão da ampla adoção de computação em nuvem há o custo do hardware que diminuiu consideravelmente nos últimos anos, o que pode tornar interessante a utilização de computação em nuvem, mas em ambiente privado. Atualmente, empresas tradicionais têm gerado grandes quantidades de dados, hospedados em bancos de dados proprietários como o *MS-SQL Server* ou *Oracle* ou ainda estes dados estão armazenados em um sistema integrado de gestão empresarial (ERP) como o *SAP ERP*, por exemplo.

Empresas líderes de mercado em segmentos importantes ainda não possuem produtos suficientemente desenvolvidos para serem implantados na nuvem computacional ou não fornecem meios para portar os dados do ambiente tradicional para a nuvem computacional. Aparentemente, aspectos importantes da migração do ambiente tradicional para o ambiente de computação em nuvem ainda não estão bem definidos.

À medida que mais aplicações são transferidas para a nuvem computacional, mais largura de banda é necessária para transportar dados entre os provedores e os consumidores dos serviços fornecidos pela nuvem. Em vários lugares do mundo a infraestrutura de Internet disponível ainda é muito deficitária, porém quanto mais perto do provedor o consumidor estiver melhor é a qualidade de serviço que pode ser obtida. Assim, investimentos e estudos em novas tecnologias de rede são necessários, pois se não houver QoS na conexão de Internet para melhorar a taxa de transferência de dados e tempos de resposta aceitáveis para os consumidores, a computação em nuvem poderá ser prejudicada pela infraestrutura da Internet. Adicionalmente, muitas tarefas não podem ser executadas de maneira fácil devido as limitações do modelo pedido-resposta *stateless* (sem estado) do protocolo HTTP. Assim, para tratar todos os tipos de interações na nuvem – como é feito em aplicações de uma empresa tradicional – melhorias no protocolo de comunicação são necessárias.

Frameworks como o *MapReduce* [Grossman 2009] e suas implementações (e.g. *Hadoop*, *Dryad*) são projetados para o processamento distribuído de tarefas intensivas – altamente dependente dos dados. Estes *frameworks* geralmente operam em sistemas de arquivos sobre a Internet (e.g. GFS, HDFS). Estes sistemas de arquivos são diferentes dos sistemas de arquivos distribuídos tradicionais na sua estrutura de armazenamento, padrão de acesso e interface de programação. Assim, introduzem problemas de compatibilidade com sistemas de arquivos e aplicações legados. O desempenho e o consumo de recursos de uma tarefa *MapReduce* é altamente dependente do tipo da aplicação. Um dos desafios inclui a modelagem de desempenho de tarefas *Hadoop* (*online* ou *offline*) e o agendamento adaptativo em condições dinâmicas.

A implantação de métodos e ferramentas que sejam fáceis e eficientes para executar o procedimento de obtenção de imagens instantâneas, congelamento ou re-inicialização (*snapshot/restart*) de instâncias de VMs incentiva a conservação dos recursos computacionais. O ideal é que seja possível automatizar este processo para alcançar o máximo de economia, por exemplo, se uma VM está executando atividades não interativas e com pouca prioridade deve ser possível congelá-la automaticamente com base em parâmetros de configuração. Tais ferramentas, além de economia de energia e aluguel da infraestrutura de hardware devem proporcionar economias no uso de softwares, pois o paradigma de controle de licenças de softwares está mudando para um modelo de pagamento de acordo com o consumo. Algo como *pay-as-you-go* oferecido pela *Amazon* no EC2 para uso de *Windows Server*, *Windows SQL Server*, *IBM DB2 Express* e *IBM WebSphere*, por exemplo [Wang et al. 2010b].

A virtualização pode proporcionar benefícios significativos para a computação em nuvem, permitindo a migração de VMs para balanceamento de carga, por exemplo. Adicionalmente, migração de VMs permite um esquema de provisionamento robusto e sensível as características do ambiente [VMWare Inc 2010]. Porém, detectar carga de trabalho intensa e inicializar o processo de migração e reconfiguração das conexões

do consumidor com a VM necessita de mecanismos complexos. Ou seja, o sistema de gerenciamento deve ser capaz de responder rapidamente as mudanças bruscas de cargas de trabalho das VMs no IaaS.

A consolidação de servidores é uma abordagem efetiva para maximizar a utilização de recursos, enquanto minimizando o consumo de energia no ambiente de nuvem. A migração de VM em execução (*Live VM migration*) é utilizada frequentemente para consolidar (agrupar) vários servidores pouco utilizados em um único servidor com boa utilização. Com isto, os servidores que ficam sem utilização podem ser colocados em estado de espera, economizando energia [Etsion et al. 2009]. Porém, a consolidação de servidores de maneira próxima do ideal é frequentemente considerada um problema de otimização *NP-hard*.

Muitos ambientes de computação em nuvem comerciais são implementados em grandes *data centers*, administrados de modo centralizado (e.g. *Amazon*). Apesar deste projeto obter uma certa economia de escala e facilidade de gerenciamento, também introduz limitações como altos gastos com energia e um alto investimento inicial para construir o *data center*. A criação de pequenos *data centers* pode ser mais vantajosa se comparado aos grandes, pois o consumo de energia é menor, os sistemas de resfriamento são mais simples, a distribuição geográfica pode atender melhor os consumidores e o custo de construção dos mesmos é mais barato. As nuvens computacionais construídas com recursos voluntários ou uma mistura de recursos voluntários e dedicados – modelo híbrido – são mais baratas de se operar e mais adequadas para aplicações sem fins lucrativos, como a computação científica. Porém, esta arquitetura também traz desafios como o gerenciamento de recursos heterogêneos e a disponibilidade dos nós (*churn*).

2.3. Segurança em computação em nuvem

2.3.1. Fundamentos de segurança computacional

Todo o sistema computacional precisa ser protegido, porém é preciso analisar a sensibilidade dos dados que uma aplicação irá manipular para que a segurança seja dimensionada adequadamente [Landwehr 2001]. Ao longo dos anos, a segurança computacional passou por várias fases, inicialmente pretendia-se prevenir as violações de proteção, após esta fase o objetivo foi detectar e limitar as violações que não podiam ser prevenidas. Posteriormente, o foco foi tolerar os ataques, visando manter o fornecimento dos serviços. Estamos indo em direção a segurança comercializada como um serviço ou parte deste e neste caso temos que confiar em quem nos vende o serviço. Analogamente, a maneira como procedemos quando colocamos o nosso dinheiro em um banco – temos que confiar que o banco esteja integro e que os administradores não o levem a falência, porque se isto acontecer estaremos arruinados.

O esquema de segurança computacional necessita preservar as propriedades básicas: confidencialidade, integridade, disponibilidade, autenticidade e não repúdio. Além disto, alguns princípios devem ser considerados: responsabilização dos autores por suas ações, fornecimento do mínimo de privilégios possível para o desempenhar de uma atividade, minimização (da quantidade, do tamanho e da complexidade) dos componentes confiáveis do sistema e priorização do modo de operação seguro durante a implantação e utilização do sistema.

A segurança de sistemas computacionais envolve temas como políticas (conjunto de regras) de segurança e sua utilização em diferentes contextos – comercial, militar, doméstico etc. e a gerência de riscos. As políticas de segurança envolvem, por exemplo, definição de regras para: proteção do nível físico; contenção, recuperação de desastres, *backup*, preservação (durante o uso) e destruição (após o uso) de mídias; operação – envolvendo treinamento do usuário e registro de todas as ações de suporte; uso de criptografia e ciclos de vida de chaves; controle de acesso a sistemas e a recursos; não violação a leis e a ética, etc. A gerência de risco envolve a avaliação sistêmica e continuada dos níveis de segurança computacionais, avaliando os vários sistemas e aplicações de forma integrada para identificar vulnerabilidades, visando eliminá-las, mitigá-las ou tolerá-las [ISO 2005].

De maneira geral os mecanismos utilizados para dar suporte ao esquema citado acima são: definição de domínios de segurança vinculando os usuários a seus respectivos domínios, implantação de operações de autenticação, autorização, controle de acesso e auditoria, e a utilização de criptografia.

2.3.2. Gerenciamento de políticas de identificação e controle de acesso

Os serviços fornecidos pela nuvem computacional podem ser disponibilizados em qualquer local físico de abrangência da mesma, ou utilizando componentes de infraestrutura incompatíveis com o ambiente do consumidor. A gerência de um grande número de serviços (SaaS, PaaS, IaaS) e recursos físicos pode gerar um volume considerável de dados a ser administrada de maneira centralizada, pois será necessário coletar, armazenar, analisar e processar estes dados. Assim, a administração centralizada pode ser considerada impraticável, e portanto faz-se necessário instanciar serviços de gerenciamento distribuídos e fracamente acoplados (com baixa dependência funcional).

Para que as organizações consumidoras utilizem os serviços oferecidos pela nuvem é necessário a implantação de um modelo de gerenciamento seguro e confiável. O esquema a ser utilizado deve facilitar a inserção e remoção de usuários dos serviços oferecidos pela nuvem. A implantação de mecanismos de autenticação robustos e esquemas de delegação de direitos funcionando de maneira confiável são fundamentais para o correto gerenciamento de identidades e para a prestação de serviços em nuvens computacionais.

Serviços de identidade utilizados pela nuvem devem suportar a delegação de direitos administrativos, com isso o gerenciamento pode ser repassado aos administradores individuais de cada ambiente – SaaS, PaaS, IaaS – então cada administrador pode gerenciar contas dentro de seu próprio domínio.

Para fornecer acesso aos diferentes níveis de serviço, a organização consumidora pode utilizar um serviço de SSO (*Single Sign-On*) que faça parte de uma federação para autenticar os usuários das aplicações disponíveis na nuvem (Figura 2.4; evento *aut*). O provedor de SSO pode ser terceirizado, instanciado externamente a organização consumidora (Domínio B). O *OpenID* é uma opção quando a organização consumidora deseja ter o processo de identificação terceirizado [OpenID 2010]. No ambiente de computação em nuvem, a federação de identidades tem um papel fundamental para permitir que organizações consumidoras associadas se autentiquem a partir de um único ou simples *sign-on* (evento *ac*). Então, poderá acontecer a troca de atributos de identidades entre o provedor de serviço e o de identidade (evento *atr*). Padrões como a *WS-Federation* podem auxiliar

na federação de identidades para diferentes domínios administrativos [OASIS 2009b].

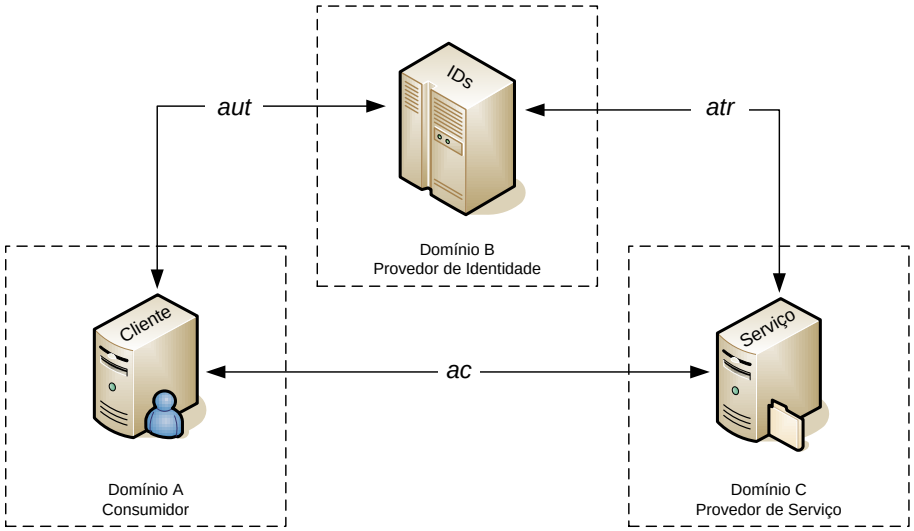


Figura 2.4. Federação de Identidades.

O nível de IaaS atende tipicamente administradores de tecnologia da informação. Assim, a divisão de atribuições dos usuários segue os requisitos de gerenciamento de privilégio de um *data center* comum – e.g. administrador do sistema, engenheiro de rede etc. O nível IaaS basicamente é voltado ao gerenciamento do ciclo de vida de servidores virtuais – interação com máquinas virtuais, envolvendo sua criação, destruição, inicialização, parada, exportação e importação.

No nível de IaaS, o provedor de computação em nuvem não tem controle sobre os serviços instanciados nas máquinas virtuais que autenticam seus usuários. Ou seja, é a organização consumidora (contratante) que decide como o serviço vai executar a autenticação dos seus usuários. Assim, o serviço pode aceitar credenciais de autenticação em vários formatos (e.g. SAML – *Security Assertion Markup Language* [OASIS 2005a], certificados X.509 [ITU-T 2000]). Certificados digitais são mais apropriados para transportar informações que não são alteradas frequentemente. Adicionalmente, esta abordagem necessita de uma infraestrutura de gestão de certificados (por exemplo, o serviço XKMS – *XML Key Management Specification* [W3C 2001]) e procedimentos para manter a integridade da informação.

Provedores de serviço nos níveis SaaS e PaaS geralmente oferecem serviços de autenticação acoplados as suas aplicações e plataformas. Alternativamente os provedores podem delegar a autenticação dos usuários para a organização contratante dos serviços. Com esta abordagem, o contratante pode autenticar seus usuários localmente, utilizando um serviço de identificação interno a organização e estabelecendo confiança com o fornecedor de serviço através da federação de identidades, por exemplo. Caso o usuário esteja agindo em seu próprio nome, o processo de autenticação pode ser centrado no usuário, utilizando algum tipo de identidade válida para a nuvem (e.g. *LiveID*).

Para o ambiente de nuvem o gerenciamento da autenticação deve fornecer suporte aos processos de criação e emissão das credenciais (e.g. senhas, certificados digitais,

credenciais dinâmicas) utilizadas pelos usuários da organização. Procedimentos de autenticação robustos (e.g. baseada em múltiplos fatores) podem não ser compatíveis com determinados serviços fornecidos pela nuvem. Consequentemente, a utilização de uma grande variedade de métodos de autenticação gerará carga administrativa adicional. O usuário dos serviços também precisa ter a usabilidade considerada, pois esse pode necessitar utilizar um conjunto de métodos para as aplicações internas a organização, e outro conjunto para acessar os serviços na nuvem. O mesmo desafio aplica-se aos provedores de computação em nuvem, pois o custo para suportar vários mecanismos de autenticação, acomodando as necessidades de consumidores usando mecanismos heterogêneos pode se tornar pouco atrativo para a entidade que mantém a nuvem. Neste caso, o ideal é a padronização dos mecanismos de autenticação para resolver estas limitações impostas pelas características de computação em nuvem.

O ambiente de computação em nuvem está sendo utilizado para hospedar vários tipos de serviços, e todos exigem garantias de segurança dos dados sendo processados e armazenados. Para tirar o máximo proveito de todo o poder oferecido pela nuvem computacional, as diferentes entidades – provedores e consumidores de serviços – que interagem com o ambiente necessitam de abordagens de segurança abrangentes e confiáveis. O particionamento do ambiente de computação em nuvem em diferentes domínios cria escopos de proteção reduzidos, regrando e limitando as interações entre as partes, classificando os tipos de serviços e recursos, facilitando as operações de gerenciamento, efetuando o balanceamento e a distribuição de carga etc. [Goyal e Mikkilineni 2009].

A definição de um sistema de segurança baseado em políticas é uma necessidade administrativa e de uso da nuvem computacional, pois é possível controlar o acesso e uso individual de cada usuário do ambiente [Yildiz et al. 2009]. Cada organização consumidora de serviços fornecidos pela nuvem precisa definir políticas para seus usuários. Os seguintes tipos de usuários devem ser considerados nas políticas: administrador, desenvolvedor e usuários finais da organização. Adicionalmente, práticas, processos e procedimentos de gerenciamento de identidade e acesso devem englobar os serviços oferecidos pela nuvem. O gerenciamento deve ser, preferencialmente, escalável, efetivo e eficiente para ambos, provedores e consumidores dos serviços [CSA 2010a].

Os perfis e as políticas de controle de acesso podem variar de acordo com o tipo de consumidor – organização ou usuário – de serviços da nuvem. Quando o consumidor é uma organização as políticas envolvem a definição de regras para a totalidade do domínio da organização consumidora, enquanto para usuários da organização as políticas precisam ser individualizadas. O perfil do usuário pode ser descrito como um conjunto de atributos utilizados pela nuvem para customizar o serviço e possivelmente restringir o acesso a outros serviços. O procedimento de controle de acesso utiliza informações sobre o perfil do consumidor para tomar decisões na escolha das políticas. Quando o consumidor é um usuário seus atributos são a única fonte de informações para o monitor de referência, sendo que as políticas foram definidas pela organização consumidora de computação em nuvem. Quando o consumidor representa uma organização, os atributos do perfil são obtidos de SLAs ou contratos – podendo envolver requisitos de QoS (*Quality of Service*).

A utilização de serviços dentro da nuvem cria a possibilidade das políticas de controle de acesso serem definidas em um lugar – por exemplo, internamente a organi-

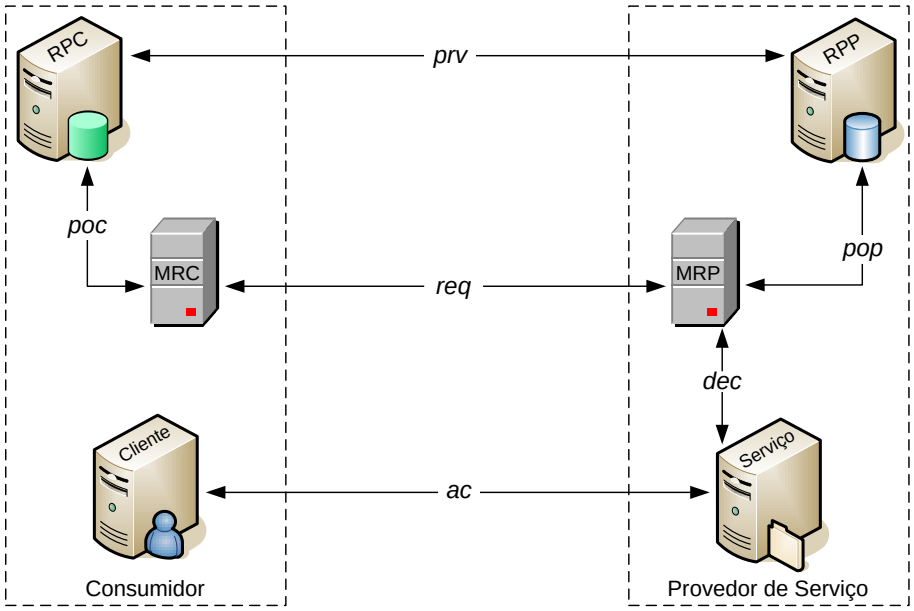


Figura 2.5. Provisionamento de políticas no provedor.

zação (Figura 2.5; evento *poc*; Repositório de Política do Consumidor – RPC) – e serem executas em outro (evento *pop*; Repositório de Política do Provedor – RPP). Isto é, as políticas definidas pela organização consumidora podem ser transferidas do repositório de políticas para o provedor de computação em nuvem que controla o uso dos serviços. Padrões de serviços web como a XACML – *eXtensible Access Control Markup Language* [OASIS 2005b] e a *WS-Policy* ([W3C 2010] são úteis nestes casos. Porém, mesmo com a utilização de especificações padronizadas, o consumidor e o provedor de computação em nuvem precisam utilizar a mesma semântica para que a interação entre as entidades dos diferentes domínios possa ocorrer de maneira transparente e segura – e.g. solicitações de acesso aos serviços (evento *ac*).

A transferência ou configuração de políticas pode ocorrer periodicamente, no modo *batch* ou sob demanda (*just-in-time*), quando será enviada concomitantemente com a solicitação de configuração vinda do Monitor de Referência do Provedor (MRP; evento *prv*). Para o modo *batch* a especificação SPML (*Service Provisioning Markup Language*) pode ser utilizada. Porém, se o modo *single sign-on* estiver ativo e o provedor de serviço de computação em nuvem for capaz de receber informações de políticas em assertivas SAML, sob demanda, então as informações podem ser transmitidas utilizando o *SAML profile of XACML* [OASIS 2005c]. Adicionalmente, se as decisões de autorização forem avaliadas externamente aos serviços hospedados na nuvem (i.e. no Monitor de Referência do Consumidor – MRC), o padrão XACML pode ser utilizado para expressar solicitações e respostas de avaliações de políticas (evento *req*).

O provisionamento (pré-configuração) de políticas para consumidores pode ser efetuado com a utilização da especificação SPML [OASIS 2006]. As políticas podem conter papeis de acesso pré-definidos: administrador, desenvolvedor e usuário final. O ambiente no nível PaaS tipicamente atende desenvolvedores de serviços. Os desenvolvedores, por exemplo, podem necessitar contas para teste de suas aplicações – necessitando de vários

usuários temporários para depurar os serviços. Observe que neste caso os usuários de teste teriam o perfil de usuário final, mas estariam sob o controle de desenvolvedores.

Os consumidores precisam ter certeza que os provedores de nuvem suportam suas necessidades e fornecem mecanismos de controle de acordo com a dinâmica exigida pelo ambiente. Basicamente, o provedor de computação em nuvem precisa: controlar o acesso de usuários aos serviços fornecidos pela nuvem – de acordo com as políticas definidas pelo consumidor; honrar os SLAs ou contratos de QoS estabelecidos com organização consumidora; controlar o acesso aos dados dos usuários; controlar acesso a área do sistema no nível de usuário e administrativo (privilegiado); manter as informações do perfil do usuário e as políticas de controle de acesso atualizadas; permitir a coleta de informações do perfil do usuário e das políticas de controle de acesso implantadas no provedor (para um determinado consumidor); fornecer meios de notificação para alterações em contas de usuários (e.g. criação, remoção, concessões de acesso), visando coibir a configuração de contas falsas ou modificação de direitos de acesso no provedor sem que o consumidor saiba; e fornecer trilhas de auditoria para o ambiente de cada consumidor – identificando atividades de gerenciamento e acesso, assim como a utilização de qualquer recurso para qual foram estabelecidas cotas de uso.

2.3.3. Riscos e ameaças em ambientes de nuvem computacional

As organizações devem avaliar o risco e as opções de segurança antes de mover seus sistemas e aplicativos para o ambiente de computação em nuvem. É necessário avaliar quais dados e serviços podem ser transferidos para o ambiente externo a organização se a nuvem for pública. Os principais tipos de ativos (*assets*) suportados pela nuvem são: dados, aplicações, processos e serviços. Estes ativos/recursos devem ser analisados para que se possa determinar sua importância para o negócio da organização. No processo de análise busca-se avaliar os impactos gerados caso algum requisito de segurança (confidencialidade, integridade ou disponibilidade) seja comprometido. As organizações podem mover integral ou parcialmente seus processos ou dados para o ambiente de computação em nuvem. Parte das transações e informações podem ser mantidas dentro do perímetro da organização – em ambiente privado [CSA 2009].

Entender os relacionamentos e dependências entre as diferentes camadas da computação em nuvem é fundamental para entender os riscos de segurança dos três principais modelos (Saas, PaaS e IaaS) de implantação de nuvem computacional. SaaS provê o conjunto mais integrado de funcionalidades de computação em nuvem, construídas de acordo com as necessidades dos usuários – possuindo o mínimo de extensibilidade (porque o serviço já está definido) para o consumidor e com um nível relativamente alto de segurança integrado ao serviço. PaaS destina-se a permitir que os desenvolvedores construam suas próprias aplicações em cima da plataforma – é mais flexível que o modelo SaaS – oferece ao consumidor funcionalidades pré-configuradas para que este possa escolher e usar. Como o conjunto de funcionalidades não é completo ou plenamente integrado, existe uma maior flexibilidade para se inserir camadas de segurança adicionais. IaaS provê poucas funcionalidades específicas para as aplicações, porém é flexível e extensível. Este provê poucas funcionalidades de segurança integradas – além da proteção da própria infraestrutura da nuvem. IaaS necessita que o sistema operacional, aplicações e conteúdo seja gerenciado e protegido pelo consumidor da nuvem computacional.

Os controles de segurança na computação em nuvem são, em sua maioria, iguais aos controles de qualquer ambiente de tecnologia da informação. Porém, de acordo com os modelos de serviço (SaaS, PaaS, IaaS), modo de operação – administração do ambiente – e tecnologias utilizadas para prover os serviços na nuvem, estes pode apresentar diferentes riscos para a organização quando comparado com as abordagens tradicionais. Na computação em nuvem se abre mão de alguns controles (físico, por exemplo) enquanto se mantêm as responsabilidades sobre o gerenciamento operacional.

Os domínios de segurança de um ambiente de computação em nuvem podem ser administrativo ou operacional. O domínio administrativo aborda assuntos relacionados a política e estratégia de negócio. O domínio operacional preocupa-se com a segurança dentro da arquitetura [CSA 2009]:

- **Domínio Administrativo**: aborda aspectos como: ações legais por violação de contrato, gestão de proteção de dados sensíveis quando o usuário ou o provedor podem ser os responsáveis por erros ou falhas e a influência que a localização física da nuvem pode trazer para os aspectos citados anteriormente – devido a diversidade de tratamento do assunto de acordo com as leis de diferentes países. O gerenciamento administrativo de risco pode ser descrito como a capacidade da organização gerenciar e medir o risco introduzido pela adoção do modelo de computação em nuvem [Kaliski Jr e Pauley 2010].

  Assuntos legais relacionados a utilização da nuvem envolvem a proteção da informação e sistemas computacionais e contramedidas para tratar as violações de segurança e privacidade – por intrusão, vazamento, revelação ou divulgação de dados protegidos. Os aspectos relativos a privacidade serão abordados na Seção 2.4.

  Gerenciar o ciclo de vida dos dados colocados na nuvem computacional está relacionado a identificação e controle de dados. Estes envolvem a definição de controles compensatórios que possam ser utilizados para tratar a perda do controle físico sobre os dados quando estes forem transferidos para a nuvem, e a definição de quem é a entidade responsável pela confidencialidade, integridade e disponibilidade dos dados.

  A utilização de serviços terceirizados precisa considerar a capacidade de mover dados e serviços de um provedor de computação em nuvem para outro, ou trazer os dados novamente para dentro da organização quando necessário. A interoperabilidade entre provedores e consumidores deve ser cuidadosamente avaliada para que a migração de dados não se torne um problema.

- **Domínio Operacional**: a computação em nuvem afeta o domínio operacional – os procedimentos utilizados para implementar a segurança, políticas de continuidade do negócio e recuperação de desastres [ISO 2005]. Assim, é necessário avaliar os possíveis riscos com a intenção de implementação de modelos de gerenciamento de risco adequados ao perfil de consumidores de serviços da nuvem computacional.

  A correta operação da tecnologia de virtualização na computação em nuvem é um item fundamental para mitigação do risco operacional. A utilização de máquinas virtuais possui os seguintes desafios associados [Ristenpart et al. 2009]: garantir que

o arrendamento compartilhado da máquina física não traga problemas de segurança para o consumidor (multilocação segura); garantir o isolamento (não interferência) entre VMs alocadas sobre uma mesma plataforma física (*co-residence*) e garantir monitoramento de vulnerabilidades em nível de hipervisor.

É indispensável definir métodos para detectar incidentes e fornecer notificações ou respostas com as respectivas contramedidas – abordando mecanismos que deveriam estar implantados tanto no nível do provedor quanto no nível de consumidor – possibilitando o correto tratamento e avaliação de incidentes, por exemplo, pode-se utilizar métodos de forense digital nestes casos.

2.3.3.1. Principais Ameaças

Com o modelo de computação em nuvem a informática passa a ser um utilitário que pode ser contratado e utilizado de acordo com a necessidade, sem que o consumidor (contratante) tenha que se preocupar com o gerenciamento da infraestrutura. Apesar dos benefícios que a computação em nuvem pode trazer, os consumidores estão preocupados com os riscos que a utilização de um ambiente novo pode representar para os ativos (*assets*) da organização. Em outras palavras, uma organização estará terceirizando um sistema pelo qual é responsável, por exemplo, sem ter total controle sobre o mesmo. Assim, a decisão de adotar a computação em nuvem precisa ser bem estudada para que o risco possa ser calculado e a computação em nuvem traga benefícios efetivos.

A segurança e a privacidade (Seção 2.4) são os principais desafios que podem impedir a ampla adoção da abordagem de computação em nuvem. Pois, falhas de segurança em qualquer um dos componentes podem impactar os demais componentes de segurança e consequentemente a segurança de todo o sistema poderá entrar em colapso [CSA 2010b].

Provedores de IaaS, por exemplo, oferecem para seus consumidores a ilusão de uma capacidade computacional ilimitada de processamento, largura de banda e armazenamento. O processo de registro para compra desse serviço é simples e pode ser feito por qualquer portador de um cartão de crédito válido. Como os registros podem ser anonimizados (tornados anônimos através do uso de logins - que identificam um usuário do sistema e não entidades no mundo real) e o uso do serviço é imediato após a contratação, malfeitores como *spammers*, programadores de *botnets* etc. são capazes de realizar atividades maliciosas com relativa impunidade [Provos et al. 2009, Zhao et al. 2009]. As soluções possíveis para mitigar os casos citados devem envolver processos de registro e validação de usuários mais rigorosos, aprimoramento da monitoração de fraudes no uso de cartão de crédito e monitoração do tráfego de rede do consumidor sem violar sua privacidade – monitoramento do trafego de informações de origem e destino públicos, por exemplo.

O sequestro de contas ou serviços não é um problema de segurança desconhecido, pois acontece em *phishing*, fraudes, exploração de vulnerabilidades de sistemas e aplicações etc., sendo que é uma prática comum usuários reutilizarem credenciais e senhas, amplificando o impacto deste tipo de ataque. Uma vez obtidas as credenciais de seu alvo, um indivíduo mal intencionado pode acompanhar as atividades e transações efetuadas pela conta de acesso, gerando uma diversidade de problemas (e.g. leitura, alteração e inserção de dados; redirecionamento de clientes para domínios falsos; subversão de instâncias de

serviços legítimos etc.). O compartilhamento ou a delegação de credenciais entre entidades deve ser evitado ou muito bem monitorado. O sistema deve implantar técnicas robustas de autenticação, preferencialmente baseadas em vários fatores (combinação de técnicas que exploram, por exemplo: algo que se é – uma característica física como a impressão digital; algo que se tem – um *token*; ou algo que se sabe – uma senha) e monitoramento para detectar atividades não autorizadas ou suspeitas – através da implantação de um sistema de detecção de intrusão, por exemplo.

Diferente das abordagens tradicionais de tecnologia da informação, a computação em nuvem oferece grande flexibilidade para os usuários visto que estes não precisam se preocupar com a complexidade de gerenciamento inerente a cada sistema (e.g. os banco de dados podem ser transferidos para *data centers* de grandes empresas especializadas). Porém, o gerenciamento dos dados em ambientes terceirizados nem sempre é confiável. Os usuários acabam ficando a mercê da disponibilidade e integridade provida pelos provedores de serviço de armazenamento (e.g. *Amazon Simple Storage Service* – S3). Assim, é necessária a utilização de modelos de armazenamento de dados seguros visando garantir a integridade dos dados dos consumidores [Wang et al. 2009].

Os dados de usuários e consumidores podem ser comprometidos de várias maneiras, por exemplo, informações que não possuem cópia de segurança podem ser eliminadas ou alteradas, os registros de um contexto podem ser desvinculados, o armazenamento pode ser feito em mídias não confiáveis, a chave de codificação pode ser perdida etc. O risco de comprometimento de dados aumenta na nuvem devido ao grande número de desafios inerentes as características arquiteturais e operacionais desse ambiente – implementação de controles de autenticação, autorização, auditoria e cifragem; falhas operacionais; problemas de jurisdição; confiabilidade do *data center* etc. Ambientes terceirizados ou externos necessitam de práticas e mecanismos de segurança rigorosos para garantir a segurança dos dados em trânsito, proteção dos dados utilizados em processos, gerenciamento do ciclo de vida de chaves, estabelecimento de regras contratuais com os provedores exigindo a correta destruição de dados que estão armazenados em mídias antes desta ser liberada para uso, estratégias para efetuar a cópia de segurança etc.

A arquitetura orientada a serviço (*Service Oriented Architecture* – SOA) e o modelo de computação em nuvem podem ser considerados serviços complementares [Zhang e Zhou 2009]. SOA compreende um conjunto de princípios e metodologias projetadas para facilitar a integração de sistemas e a comunicação independentemente da linguagem de desenvolvimento ou plataforma. Enquanto que a computação em nuvem foi projetada para permitir uso instantâneo e massivo da capacidade de processamento e armazenamento, por exemplo, sem que seja necessário investir em infraestrutura, treinamento de equipes, licenciamento de softwares etc.

Independentemente das diferenças nos propósitos de projeto de cada uma, a abordagem de computação em nuvem pode se relacionar com a arquitetura SOA na utilização de componentes como um serviço (*Components as a Service*, CaaS – SOA implementada através de padrões para serviços web). Assim, a computação em nuvem e SOA podem ser desenvolvidas independentemente, ou concomitantemente como atividades complementares visando provimento de um negócio que atenda a uma vasta gama de consumidores [Dawoud et al. 2010]. Para acessar recursos que disponibilizam interfaces

de serviços web, o protocolo SOAP (*Simple Object Access Protocol*) é o mais utilizado, junto com a *WS-Security* – uma das extensões padrão para proteger as mensagens em trânsito. Os ataques que visam afetar a segurança da XML (*Extensible Markup Language*) [Gruschka e Iacono 2009] podem ser mitigados com a utilização de especificações fornecidas por organizações como OASIS e W3C [Jensen et al. 2009].

A grande maioria dos provedores de computação em nuvem expõe um conjunto de interfaces para gerenciar e interagir com os serviços oferecidos (e.g. provisionamento, gerenciamento, orquestração, monitoramento etc.). A segurança e a disponibilidade de alguns serviços da nuvem dependem da segurança destas APIs, por exemplo, autenticação, controle de acesso, cifragem e monitoramento. As interfaces devem ser projetadas para se proteger de tentativas acidentais ou maliciosas de violação de políticas. O modelo de segurança das interfaces disponibilizadas pelo provedor deve ser cuidadosamente analisado e avaliado, assegurando-se que mecanismos consistentes de autenticação e controle de acesso estejam implementados.

O nível de acesso concedido aos funcionários do provedor de computação em nuvem poderia permitir que um indivíduo mal intencionado obtivesse dados confidenciais ou ganhasse controle sobre os serviços disponibilizados no provedor. A utilização de um único domínio de gerenciamento combinado com a falta de transparência dos processos e procedimentos aplicados pelo provedor da nuvem (e.g. ausência de políticas que envolvem os funcionários no procedimento de concessão de direitos e monitoramento de acessos a ativos físicos e virtuais) podem fazer deste ambiente um lugar hostil para processos e informações. Procedimentos de gestão rigorosos devem ser aplicados a toda a cadeia de provimento de serviços (*supply chain*) avaliando cuidadosamente todas as entidades que interagem direta ou indiretamente com o serviço (e.g. transparência na definição de políticas segurança da informação, boas práticas de gerenciamento, relatórios de conformidade, notificação de falhas etc.).

O nível IaaS serve de base para os demais modelos de fornecimento de serviço (PaaS e SaaS), sendo que a falta de segurança neste nível certamente afetará os modelos construídos sobre a mesma. O uso da virtualização permite que os provedores de computação em nuvem maximizem a utilização do hardware, comutando várias VMs (*Virtual Machine*) de consumidores em uma mesma infraestrutura física. Esta abordagem pode introduzir vulnerabilidades, pois geralmente executa-se a multilocação (*multi-tenancy*) – comutação das máquinas virtuais de consumidores distintos sobre o mesmo hardware. Assim, a VM de um consumidor poderia ser alocada no mesmo servidor físico de seu adversário ou concorrente. Esta abordagem traz um novo tipo de ameaça, pois o adversário poderia violar o isolamento entre as VMs – explorando vulnerabilidades que permitam se infiltrar no hipervisor, ou usar canais secundários (*side-channels*) visando obter acesso não autorizado a dados ou processos [Ristenpart et al. 2009, Dawoud et al. 2010].

Os fornecedores de IaaS prestam serviços compartilhando uma mesma infraestrutura física. Porém, componentes físicos do hardware como a memória *cache on die* da CPU não foram projetados para oferecer propriedades de isolamento para uma arquitetura com vários usuários (*multi-tenant* – multi-inquilinos). Para contornar essa limitação, o hipervisor faz a mediação do acesso entre o sistema operacional convidado e os recursos computacionais físicos. Porém, hipervisores podem apresentar falhas que permitam ao

sistema convidado obter níveis impróprios de controle e ou influência na plataforma subjacente. O isolamento (confinamento) das VMs deve ser assegurado para que um consumidor não viole o *address space* de outros consumidores sendo executados no mesmo provedor de computação em nuvem. Todo o processo de implementação, instalação e configuração do ambiente de nuvem deve seguir boas práticas de segurança, sendo que após esta fase o ambiente deve ser monitorado visando detectar mudanças ou atividades não autorizadas.

Um dos princípios da computação em nuvem é a redução dos custos com ativos/recursos de hardware/software e a respectiva manutenção associada aos mesmos, permitindo que as empresas concentrem-se em seus negócios. Esta abordagem tem benefícios financeiros e operacionais, mas deve ser cuidadosamente avaliada devido as preocupações com a segurança – principalmente, atualizações e a versão de um sistema, boas práticas e políticas de segurança. Modelos de segurança para a camada de IaaS podem ser utilizados como um guia para avaliar e reforçar a segurança do ambiente [Dawoud et al. 2010].

2.4. Privacidade e computação em nuvem

A segurança da informação se refere à proteção sobre as informações de uma determinada empresa ou indivíduo, isto é, aplica-se tanto às informações corporativas quanto às pessoais. Todavia, existe uma relação inversa quando se trata de privacidade e segurança, pois quanto maior a segurança coletiva, geralmente menor é a privacidade individual [Fischer-Hübner 2001]. Um exemplo dessa relação são os sistemas de monitoramento com vídeo em prédios e ambientes públicos.

Para [Sweeney 2002], a segurança computacional não implica em proteção à privacidade, pois embora mecanismos de controle de acesso e autenticação possam proteger as informações contra a divulgação, eles não tratam da propagação indireta da informação, nem de divulgações com base em inferências e correlações sobre informações extraídas de outras fontes. Esta seção apresenta os principais conceitos relacionados à privacidade e alguns mecanismos usados (ou propostos) para protegê-la em ambientes computacionais, com ênfase em computação em nuvem.

2.4.1. O conceito de privacidade

De acordo com [Shirey 2000], a privacidade pode ser definida como o direito de uma determinada entidade (normalmente um indivíduo), agindo em seu próprio nome, de determinar o grau de interação de suas informações com contexto onde se encontra inserida, incluindo o grau de comprometimento/disposição em divulgar essas informações para outras entidades. Existem basicamente três elementos na privacidade: o *sigilo*, o *anonimato* e o *isolamento* (ou solidão, o direito de ficar sozinho) [Fischer-Hübner 2001, Wright 2004]. O sigilo é um problema fortemente ligado à confidencialidade, o anonimato está relacionado à proteção da identidade do sujeito e o isolamento é o direito de ficar indisponível para outros indivíduos.

O direito à privacidade é um conceito consolidado em algumas áreas, como a médica, a jurídica e a fiscal. No contexto médico [Beaver e Harold 2004], a privacidade consiste na limitação do acesso às informações de um indivíduo, ao acesso ao próprio indivíduo ou à sua intimidade. Em outras palavras, é o direito do indivíduo não ter sua vida ou seus dados observados sem autorização. Para os juristas, privacidade é o direito

de ficar sozinho [Staples 2007]. A privacidade está atrelada à questão do anonimato, ou seja, à condição de um indivíduo ter suas informações pessoais protegidas [Shirey 2000]. A privacidade também pode ser vista como a capacidade de um usuário realizar ações em um sistema sem ser identificado.

Em geral, a noção de privacidade abrange três diferentes escopos [Fischer-Hübner 2001]: *Privacidade territorial*: proteção da região próxima a um indivíduo, como seu ambiente doméstico, local de trabalho ou espaços públicos; *Privacidade do indivíduo*: proteção contra interferências indesejadas, como buscas físicas (revistas), testes com drogas ou informações que possam violar aspectos morais do indivíduo; e *Privacidade da informação*: designa quando e como dados pessoais de um indivíduo podem ser coletados, armazenados, processados e propagados a terceiros.

A definição mais comum e aceita no mundo da informática diz que *a privacidade consiste nos direitos e obrigações dos indivíduos e organizações com relação à coleta, uso, conservação e divulgação de informações pessoais* [Mather et al. 2009]. A privacidade pode ser vista como um aspecto da confidencialidade. A confidencialidade define que uma informação não deve estar disponível ou divulgada a indivíduos, entidades ou processos não autorizados pela política de acesso [Shirey 2000]. Por sua vez, a privacidade é a proteção contra a exposição indevida de informações pessoais ou o desejo de controlar o nível de exposição e uso dessas informações.

2.4.2. Privacidade e anonimato

O anonimato é a qualidade ou condição do que é anônimo, isto é, sem identificação ou autenticação. Com o advento das mensagens por telecomunicações e, em particular, pela Internet, designa o ato de manter uma identidade escondida de terceiros [Wright 2004, Staples 2007]. O anonimato oferece algumas vantagens, como expressar opiniões polêmicas sem receio de represálias. Contudo, também pode ser usado de forma maliciosa, sendo por essa razão regrado pelas leis de vários países (como é o caso da Constituição brasileira, segundo a qual "É livre a manifestação do pensamento, sendo vedado o anonimato"). O valor do anonimato não reside na capacidade de ser anônimo, mas na possibilidade de atuação ou participação permanecendo inacessível a terceiros. [Staples 2007] comenta que indivíduos com capacidade de desvincular suas identidades de suas atividades têm melhores condições de controlar a divulgação de suas informações pessoais a terceiros.

Deve-se diferenciar o *anonimato total* do *anonimato parcial* (ou pseudo-anonimato) [Wright 2004]. No anonimato total, não é possível rastrear a origem da comunicação, por exemplo, uma carta sem assinatura e sem um endereço de retorno. Já o anonimato parcial faz uso de pseudônimos para mascarar uma identidade verdadeira, permitindo receber respostas sem a possibilidade de associar a origem da comunicação à identidade real. O anonimato parcial é uma alternativa a considerar quando o anonimato total não for permitido, por exemplo, em situações onde a identidade do usuário deva ser mantida oculta, mas este possa ser responsabilizado por seus atos. Todavia, o anonimato total na Internet é difícil de alcançar, pois a infraestrutura que mantém a Internet permite identificar as origens e destinos das comunicações.

O anonimato pode ser considerado uma ação extrema para manter a privacidade.

Sem divulgação alguma de informação pessoal, o controle sobre estas é completo, a não ser que elas sejam obtidas através de ações maliciosas. Todavia, ao empregar o anonimato, um indivíduo pode ter restrito seu acesso a serviços que exijam sua identificação, pode tornar-se menos acessível a comunicações iniciadas por outros indivíduos.

2.4.3. Informações privadas

Informação pessoal é um termo utilizado de forma genérica para identificar informações de diferentes indivíduos. Neste trabalho será utilizada a caracterização de informações privadas proposta por [Pearson et al. 2009], resumida a seguir:

- *Informações que Identificam o Indivíduo*: qualquer informação que pode ser usada para identificar ou localizar um indivíduo (nome ou endereço, por exemplo) ou informações que podem ser correlacionadas com outras informações para identificação do indivíduo (número de cartão de crédito ou de telefone).
- *Informações sensíveis*: informações sobre religião, raça, saúde, orientação sexual, partidária ou outras informações similares consideradas privadas. Estas informações exigem garantias adicionais, pois podem ser utilizadas para constranger o indivíduo. Outras informações consideradas sensíveis incluem dados financeiros ou sobre seu desempenho profissional.
- *Outras informações consideradas sensíveis que possam identificar o indivíduo*: tais como informações biométricas ou imagens de câmeras de vigilância em locais públicos.
- *Dados comportamentais*: dados coletados a partir do uso do computador ou outros dispositivos, como históricos de navegação na Internet ou contatos de amigos virtuais.
- *Dispositivos de identificação única*: outros tipos de informação que possam ser identificadas pelo uso de um dispositivo exclusivo do usuário, tais como endereço IP, dispositivos RFID ou *tokens* de gerência de identidade baseados em hardware.

A privacidade dessas informações possui três aspectos [Pfleeger e Pfleeger 2006]: sua *divulgação*, sua *sensibilidade* e as *partes afetadas*. Sob a ótica da divulgação, a privacidade pode ser definida de acordo com o nível de controle que se deseja dar a uma informação, ou seja, para quem o dono da informação deseja permitir o acesso. A partir do momento que um indivíduo divulga uma informação considerada privada a outro indivíduo (como um número de telefone celular), esta terá poder sobre a mesma, podendo divulgá-la a terceiros se o desejar.

Em relação à sensibilidade, é considerada uma informação sensível qualquer informação que possibilite uma violação de segurança, ou seja, uma violação da integridade, disponibilidade ou confidencialidade. Em [Stahl 2008] a sensibilidade da informação é exemplificada como dados pessoais sobre opiniões religiosas, políticas, condições de saúde ou origem étnica e portanto passíveis de proteção. A definição inequívoca do nível de sensibilidade de uma informação pessoal pode ser complexa. Por exemplo, para muitas

pessoas o valor de seus salários é um dado importante que poucos deveriam conhecer; já para outras pessoas é indiferente se seus colegas conhecem o valor de seu salário.

As entidades afetadas podem ser indivíduos, empresas, organizações, governos, etc. Todas as entidades atribuem níveis de sensibilidade às suas informações (ou às informações de seus elementos constituintes). Um hospital, por exemplo, considera as informações sobre seus pacientes como privadas [Beaver e Harold 2004], enquanto um governo considera privadas muitas informações militares ou diplomáticas.

2.4.4. Princípios da privacidade

Um dos primeiros trabalhos a discutir questões relativas à privacidade em ambiente computacional foi [Ware 1973], que definiu vários princípios (ou critérios a observar) relativos à privacidade de dados pessoais mantidos em um sistema informático. Mais recentemente, outros trabalhos retomaram a definição desses princípios de forma mais ampla, como [Fischer-Hübner 2001, Hinde 2003, Rezgui et al. 2003, Yee e Korba 2009]. Desses trabalhos é possível sintetizar o seguinte conjunto de princípios para a privacidade em um contexto mais amplo, não exclusivamente computacional:

1. *Responsabilidade*: uma organização é responsável pelas informações pessoais sob o seu controle e deve designar indivíduos responsáveis pela organização e conformidade dessas com a legislação e políticas internas. Devem existir termos sobre responsabilidade de uso, com sanções legais em caso do mau uso.

2. *Identificação de objetivos*: os objetivos para os quais as informações pessoais são coletadas devem ser identificados pela organização previamente ou enquanto a informação é coletada.

3. *Consentimento*: o consentimento dos indivíduos é necessário para a coleta, uso e/ou divulgação de informações pessoais. As informações não podem ser transferidas para outras entidades, salvo se autorizado e com um nível de proteção adequado.

4. *Limite de coleta*: a coleta de informação pessoal será limitada ao necessário para os fins identificados pela organização. As informações devem ser coletadas por meios conhecidos e com amparo legal.

5. *Limite de uso, divulgação e retenção*: as informações pessoais não devem ser utilizadas ou divulgadas para outros fins que não aqueles para os quais foram coletadas, exceto com o consentimento do indivíduo ou se exigido por lei. Além disso, as informações pessoais serão mantidas apenas o tempo necessário para o cumprimento desses propósitos.

6. *Precisão*: as informações pessoais deverão ser tão precisas, completas e atualizadas quanto for necessário para os propósitos definidos.

7. *Salvaguardas*: medidas de segurança adequadas à sensibilidade das informações devem ser usadas para protegê-las, tanto em relação à sua confidencialidade quanto à sua integridade.

8. *Transparência*: a organização deve tornar disponíveis aos indivíduos informações específicas sobre suas políticas e práticas relativas à gestão das informações privadas.

9. *Acesso individual*: a pedido, todo indivíduo deve ser informado da existência, uso e divulgação de suas informações pessoais e deve ser permitido o acesso a essa informação.

10. *Crítica à Conformidade*: um indivíduo deve ser capaz de criticar a precisão e integridade de suas informações e modificá-las se necessário. Também deve ser facultado ao indivíduo questionar os princípios anteriores ou os sujeitos responsáveis a respeito da política de privacidade adotada.

2.4.5. Privacidade em computação em nuvem

A privacidade é uma propriedade importante para a computação em nuvem, seja em termos de conformidade legal ou confiança do consumidor, a ponto de ser citada no relatório anual da Ernst & Young [Ernst & Young 2010] como um dos tópicos mais importantes para o ano de 2010. Logo, a proteção da privacidade é uma questão-chave e precisa ser considerada em todas as fases de um projeto para esse ambiente. De forma geral, todos os aspectos apresentados na seção 2.4.3 são levantados nas pesquisas sobre privacidade na nuvem. Ainda assim, vários questionamentos sobre privacidade surgem, na medida em que essa tecnologia evolui [Mather et al. 2009]:

- *Acesso*: os donos dos dados têm o direito de saber quais informações são mantidas e, em alguns casos, de solicitar a remoção dessas informações. Se um usuário exerce o seu direito de solicitar ao provedor a eliminação dos seus dados, será possível garantir que todas as suas informações foram eliminadas da nuvem?

- *Aderência*: quais são os requisitos de conformidade à privacidade neste ambiente? Quais leis, regulamentos, normas ou compromissos contratuais regem o uso das informações, e quem são os respectivos responsáveis? Um ambiente de nuvem pode atravessar várias jurisdições, no caso de dados armazenados em vários países. Qual o foro competente para regular esse ambiente ou as informações armazenadas nele?

- *Armazenamento*: em qual parte da nuvem as informações estão armazenadas? Estão em um centro de dados de outro país? A legislação sobre privacidade pode limitar a capacidade dos provedores em transferir alguns tipos de informações para outros países. Todavia, quando os dados são armazenados na nuvem, essa transferência pode ocorrer sem o conhecimento dos mesmos.

- *Retenção*: por quanto tempo a informação pode ser retida na nuvem? Quais as políticas de retenção e descarte dessas informações? Quem rege essas políticas, o consumidor que armazenou as informações ou o provedor de computação em nuvem? Quais as exceções à política?

- *Destruição*: como deve ocorrer a destruição das informações que identificam o consumidor? Como garantir que não foram conservadas cópias destas? A disponibilidade através de replicação pode ser um problema no momento da destruição

de informações. Será que as réplicas da informação foram destruídas ou apenas tornaram-se inacessíveis?

- *Auditoria*: como as organizações consumidoras podem monitorar e verificar se seus fornecedores estão cumprindo os requisitos de privacidade?
- *Violação da privacidade*: em casos confirmados de violação, quais os responsáveis pela notificação, processos (e custos associados)? Como determinar a culpa de cada entidade envolvida?

Os riscos e ameaças à privacidade diferem de acordo com cenário de uso da computação em nuvem. Para [Pearson et al. 2009], existem alguns cenários de riscos que podem afetar a privacidade: o usuário de serviços em nuvem poderia ser forçado ou persuadido a aceitar o monitoramento das suas atividades ou fornecer informações pessoais contra sua a vontade, ou de alguma outra forma na qual não se sinta confortável; este poderia ver suas informações armazenadas na nuvem perdidas ou divulgadas; os provedores de serviços poderiam utilizar as informações pessoais para outros fins que não os definidos inicialmente. Não existem só riscos para os usuários: em caso de problemas, o implementador do PaaS poderia ser responsabilizado legalmente pela exposição de informações sensíveis, ocasionando prejuízos financeiros e perda de credibilidade.

As normas e recomendações existentes sobre segurança, tais como a *IT Infrastructure Library* (ITIL) [ITIL 2010] ou o padrão ISO 27001:2005 [ISO 2005], não foram criadas pensando nos ambientes virtuais de armazenamento de dados e fornecimento de serviços de nuvem. Neste contexto, o trabalho [Doelitzscher et al. 2010] identifica os seguintes problemas na computação em nuvem:

- *Cumprimento das leis e políticas*: os consumidores são responsáveis pela segurança e integridade de seus dados, inclusive nos casos em que estes estão sendo sob a custódia de terceiros.
- *Privilégios de controle de acesso*: o armazenamento ou tratamento de informações sensíveis na nuvem gera um risco adicional: os serviços de computação em nuvem são controlados por terceiros. É necessário um controle de acesso restrito sobre os administradores para evitar acesso indevido às informações dos consumidores.
- *Segmentação das Informações*: as informações de vários consumidores da nuvem podem ser armazenadas no mesmo disco rígido físico, separadas pelo uso da virtualização. Para a proteção das informações contra acesso não-autorizado, os provedores de serviços na nuvem precisam prover criptografia dos dados. Além disso, é necessário manter registros de auditoria, pois falhas durante o processamento das informações podem acarretar em perda destas.
- *Incidentes de segurança*: os provedores de serviços na nuvem oferecem poucos recursos no caso de incidentes de segurança. Em um ambiente onde máquinas virtuais são continuamente iniciadas e encerradas por vários consumidores, a atualização das informações dos usuários e dos registros de atividade é uma tarefa desafiadora. Normalmente, apenas o registro de atividades de um usuário na nuvem é oferecido em casos de investigação de incidentes.

Alguns dos problemas levantados podem ser resolvidos com ferramentas tradicionais de segurança, como gerência de controle de acesso, ferramentas de auditoria, gerência de identidade, etc. Todavia, ainda restam algumas lacunas relativas à proteção da privacidade. A próxima seção apresenta diversos trabalhos que visam a proteção da privacidade dos usuários em ambientes de computação em nuvem.

2.4.6. Abordagens de proteção da privacidade

Em geral, a proteção da privacidade de dados pode ser realizada através de leis de proteção à privacidade definidas por cada governo, da auto-regulamentação para práticas leais e códigos de conduta promovidas por entidades ao manipular informações, por tecnologias que aumentem a privacidade, e também através da educação sobre privacidade a usuários e profissionais de TI. Infelizmente, a privacidade é muitas vezes mal gerida e, por consequência, ocorrem abusos no uso das informações. Exemplos bem conhecidos são a divulgação de informações pessoais a terceiros sem o consentimento explícito de seus proprietários legítimos, a construção de perfis de usuários a partir do relacionamento de informações heterogêneas [De Capitani di Vimercati e Samarati 2006].

Do ponto de vista tecnológico, existem diversas abordagens que permitem melhorar a proteção da privacidade das informações. Já nos anos 1970, o trabalho [Turn e Ware 1975] sugeria algumas medidas para a proteção de dados pessoais em sistemas computacionais, como reduzir a exposição de informações, limitando a quantidade de dados armazenada e usando amostras aleatórias em vez de coletas completas; reduzir a sensibilidade dos dados ou adicionar erros sutis a eles; modificar ou retirar alguns dados para torná-los anônimos (mascarar o nome, por exemplo); e também criptografar os dados.

Os requisitos de privacidade variam de acordo com as leis de cada país ou a vontade do consumidor. Assim, é importante que os provedores que compartilham e trocam informações adotem e apliquem diretivas de privacidade. O perfil XSPA (*Security and Privacy Authorization* [OASIS 2009a] – uma parte da especificação XACML [OASIS 2005b]) auxilia as entidades na troca e definição de requisitos de privacidade nestes casos.

Na sequência do texto serão discutidos alguns trabalhos específicos recentes para a proteção da privacidade em ambientes de computação em nuvem.

2.4.6.1. Gerência de identidades digitais

Normalmente, usuários novos têm de estabelecer sua identidade antes de utilizar um serviço na nuvem. Isto se dá, geralmente, pelo preenchimento de um formulário *online* solicitando informações sensíveis (nome, endereço, número do cartão de crédito, por exemplo). Este processo deixa rastros de informações pessoais que, se não forem devidamente protegidas, podem ser roubadas. Neste contexto, o trabalho [Bertino et al. 2009] propõe a criação de um serviço para a gestão de identidades digitais (*IdM – Digital Identity Management*) para minimizar o risco de roubo de identidade e fraude.

Os serviços oferecidos na nuvem são heterogêneos e podem utilizar atributos distintos para a identificação dos usuários. Assim, surgem problemas de interoperabilidade que vão desde o uso de *tokens* de identidade diferentes, como os certificados X.509 ou o

uso de informações diferenciadas para identificar o usuário (CPF ou e-mail, por exemplo). O uso de informações diferentes para montar uma identidade cria outro problema: a heterogeneidade de identidades, que ocorre quando usuários e provedores de serviços usam vocabulários diferentes para os atributos de uma identidade. Esse conjunto de identidades diferentes dificulta a utilização da nuvem, pois o usuário pode fornecer informações desnecessárias ou mesmo erradas para um provedor.

Neste contexto, o uso de um gerenciador de identidades para identificar o usuário dificultaria a violação da privacidade do mesmo. A fim de solucionar o problema da gestão de identidades heterogêneas, os autores propõem a utilização do *IdM* combinado com um protocolo de privacidade (utilizando técnicas de criptografia, tabelas de correspondência, dicionários e ontologias), a fim de identificar o usuário nos vários serviços da nuvem. Esse protocolo utiliza um conjunto de provas de conhecimento (*Aggregate Zero Knowledge Proofs of Knowledge - AgZKPK*) que permite ao cliente comprovar seus atributos de identificação sem a necessidade de informá-los de forma explícita.

[Bertino et al. 2009] consideram a adoção do *IdM* em conjunto com outras entidades: Provedores de Identidade (*IdP - Identity Providers*), provedores de serviços na nuvem (*CSP - Cloud Service Providers*), sistemas de registro (*Registrars*) e usuários. O *CSP* provê acesso aos dados e ao software disponível; os *IdPs* fornecem os atributos para a identificação do usuário e o controle no compartilhamento de informações. Nessa proposta, os sistemas de registro são elementos adicionais que armazenam e gerenciam as informações relacionadas à identificação dos atributos utilizados, sendo que cada registrador contém tuplas de atributos de identificação do usuário. As informações armazenadas nos registradores não incluem valores de identificação não-cifrados.

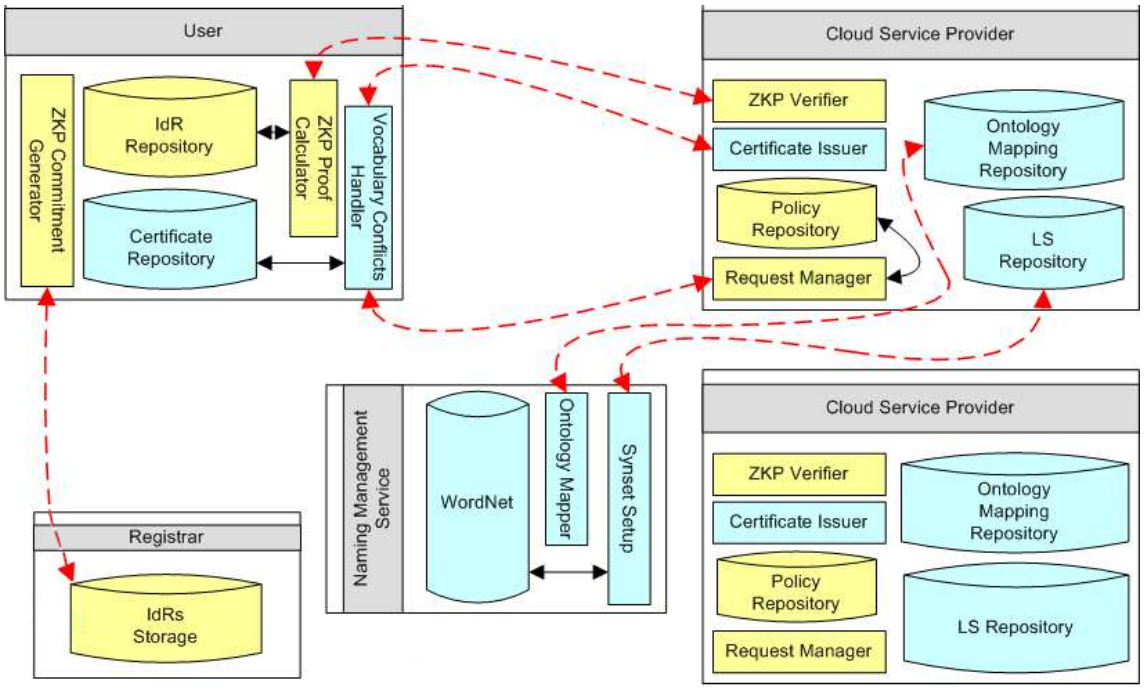


Figura 2.6. Sistema de gerência de identidades digitais [Bertino et al. 2009].

A arquitetura proposta (Figura 2.6) é composta por quatro componentes: o sistema

de registro, o provedor de serviços, o agente do usuário e o gerenciador de identidades heterogêneas. O sistema de registro é o responsável pelo registro de identidades no cliente e fornece mecanismos para recuperação de dados públicos utilizados no protocolo AgZKPK. O agente do usuário é composto por três módulos: gerador de compromissos ZKP[1], calculador da prova ZKP e gerenciador de conflitos de vocabulários. O gerador de compromissos fornece funções para o cálculo de compromissos de Pedersen (*Pedersen commitments* [Pedersen 1992]) sobre os atributos da identidade, o calculador gera a prova AgZKPK a ser fornecida ao CSP. O gerenciador de conflitos controla os nomes de atributos que correspondem aos parâmetros a ser enviados ao CSP e gerencia os certificados de prova de identidade armazenados localmente.

O provedor de serviços é composto por quatro módulos: um gerenciador de pedidos, um gerenciador de mapeamento de caminhos, o certificado do emissor e verificador ZKP e três repositórios responsáveis pelo armazenamento do mapeamento de ontologias, dos conjuntos de sinônimos e das políticas de verificação de identidade. O gerenciador de pedidos manipula as solicitações dos clientes e verifica junto a eles os atributos de identificação necessários para a verificação. O verificador ZPK testa a prova AgZKPK. Finalmente, o serviço de gerência de heterogeneidade provê várias funções que são compartilhadas por todos os CSPs e consiste em dois módulos: *Synset SetUp* e gerenciador de ontologias. O módulo *Synset SetUp* retorna, consultando um dicionário local, um conjunto de sinônimos para um determinado termo e o gerenciador de ontologias fornece as funcionalidades para realizar o mapeamento entre duas ontologias.

2.4.6.2. Gerenciador de privacidade

O artigo [Pearson et al. 2009] propõe um gerenciador de privacidade que usa técnicas de ofuscação de dados (uma técnica na qual algumas características dos dados originais permanecem presentes nos dados cifrados). O método utiliza uma chave de ofuscação fornecida pelo usuário e conhecida do gerenciador, mas que não é fornecida ao provedor de serviços na nuvem. Desta forma, os autores acreditam que não é possível recuperar as informações nem roubar os dados dos usuários. O provedor do serviço tem responsabilidades legais ao tratar informações abertas dos usuários, mas os dados ofuscados não permitem a identificação do indivíduo. Assim, o provedor do serviço não está sujeito às sanções legais no tratamento dos dados ofuscados.

Entretanto, para as aplicações em nuvem não é prático trabalhar somente com dados ofuscados. Neste caso, o gerenciador de privacidade acrescenta duas características: *preferências* e *personalidade*, que possibilitam aos usuários comunicarem aos provedores de serviços como eles desejam que suas informações sejam tratadas (e que indiretamente constituem um consentimento do usuário para o uso das informações pelo provedor). As preferências indicam quais e como os dados podem ser visualizados, enquanto a personalidade possibilita ao usuário personificar diferentes visões da mesma identidade (anônimo, visão parcial, visão total).

[1]ZPK (*zero-knowledge proof*) é um método interativo que permite a uma parte provar para outra que uma determinada declaração é verdadeira sem revelar qualquer informação, a não ser a veracidade da declaração. Normalmente a prova ocorre pela utilização de fórmulas matemáticas.

A proposta sugere o uso de um Módulo de Plataforma Confiável (*Trusted Platform Module* - TPM), um componente de hardware que provê serviços de criptografia, geração e verificação de chaves digitais, cálculo e verificação de *hash* de informações e armazenamento de informações para verificação de identidades [Trusted Computing Group 2010]. O TPM é um módulo inviolável e pode fornecer vários serviços de segurança necessários para soluções na nuvem.

Os autores descrevem três arquiteturas possíveis para o gerenciador de identidade:

- *Gerenciador de privacidade no cliente*: o gerenciador de privacidade executa no lado do cliente e auxilia os usuários na proteção de sua privacidade ao acessar serviços fornecidos pela nuvem (Figura 2.7). O principal componente do gerenciador é provido pelo *serviço de ofuscação e desofuscação*, que reduz a quantidade de informações sensíveis fornecidas para a nuvem. Nesta arquitetura, o TPM é utilizado para proteger as chaves de ofuscação no cliente, garantir a integridade do gerenciador de privacidade e as identidades utilizadas pelo usuário. Como a proposta é baseada no TPM, as informações não estarão disponíveis para acesso mesmo com a violação deste, pois neste caso ocorrerá a perda da chave de ofuscação sob sua responsabilidade.

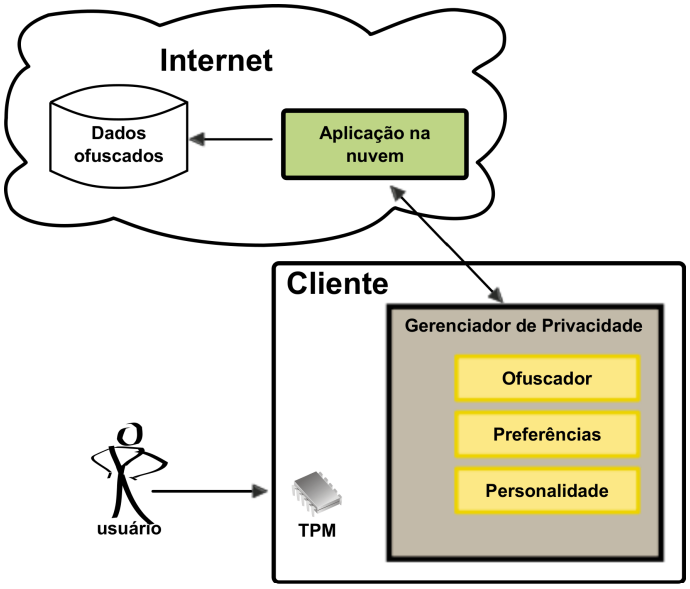


Figura 2.7. Gerenciador de privacidade no cliente [Pearson et al. 2009].

- *Gerenciador de privacidade em uma nuvem híbrida*: uma alternativa à gerência de privacidade no cliente é a possibilidade de gerenciar a privacidade entre nuvens distintas. Esta solução é indicada para ambientes corporativos, nos quais exista uma proteção adequada; sua aplicação principal seria o controle de informações pessoais circulando da nuvem privada para a nuvem pública. A proposta sugere que o gerenciador e o TPM possam executar em ambientes virtualizados [Dalton et al. 2009], com o uso combinado de *proxies* e outras ferramentas tradicionais de segurança. Os autores alertam para o problema da escalabilidade da proposta, que neste caso pode ser afetada em função do nível de tráfego entre as nuvens.

- *Gerenciador de privacidade para infomediários na nuvem*: neste caso, o gerenciador de privacidade atua entre dois infomediários[2]. Ele age em nome do usuário e decide o grau permitido de transferência de informações. O nível de privacidade é baseado nas políticas de transferência de informações especificadas pelo usuário, no contexto do serviço e, se possível, em uma avaliação da confiabilidade do ambiente prestador de serviços. O infomediário poderia ser uma associação de consumidores ou outra entidade qualquer na qual o usuário confie.

Os autores sugerem como exemplo de uso um cenário com um fotógrafo portando uma câmera com GPS embutido, que acrescenta informações geográficas à imagem. Neste cenário, o fotógrafo deseja compartilhar suas fotos com alguns familiares, mas algumas destas também podem ser comercializadas. Para compartilhar apenas com os familiares, basta definir essa intenção nas preferências. Para a comercialização, o fotógrafo deseja retirar as informações de GPS para impedir que outros concorrentes possam tirar fotos idênticas do mesmo local. Neste caso, o gerenciador é utilizado para ofuscar as informações geográficas antes do compartilhamento.

2.4.6.3. Privacidade como um serviço

O trabalho [Itani et al. 2009] apresenta uma proposta para assegurar a privacidade das informações pessoais dos usuários na nuvem, observando a legislação correspondente. Conhecido como *Privacy as a Service* (PasS), o projeto é baseado em um conjunto de protocolos de segurança, com processamento e auditoria das informações sensíveis através de processadores criptográficos. Os processadores criptográficos são indicados por serem invioláveis e à prova de falsificação (de forma análoga ao TPM), de acordo com as especificações definidas em [FIPS 140-2 2001]. Esses processadores são posicionados em um domínio de execução seguro e confiável dentro da infraestrutura da nuvem, devendo estar física e logicamente protegidos contra acessos não-autorizados.

O modelo proposto é baseado em uma solução de nuvem típica composta de duas partes: um *provedor*, que gerencia e opera uma infraestrutura de nuvem para armazenamento e serviços, e um *usuário*, que utiliza o armazenamento na nuvem e os recursos remotos para processamento de dados. O projeto oferece ao usuário o controle completo sobre os mecanismos de privacidade aplicados às informações na nuvem. O serviço de privacidade proposto baseia-se no grau de sensibilidade das informações do usuário para definir os níveis de confiança no provedor do serviço. O serviço de privacidade oferece suporte a três níveis de confiança:

1. *Confiança total*: este nível se aplica às informações não-sensíveis, que podem ser processadas e armazenadas na nuvem sem uso de criptografia. O provedor é considerado totalmente confiável para o armazenamento e processamento de informações deste nível.

[2]Empresas infomediárias são intermediárias de informações, cujo negócio é pesquisar e analisar informação, desenvolvendo análises detalhadas do mercado e caracterização dos clientes para utilização por outras entidades. São especializadas em veiculação de conteúdo via internet.

2. *Confiança parcial*: este nível envolve as informações que precisam ser armazenadas cifradas, seja por questões jurídicas ou por regras de conformidade (como registros médicos ou de transações financeiras, por exemplo). Neste nível o consumidor confia no provedor para armazenar suas informações cifradas utilizado chaves fornecidas por ele.

3. *Sem confiança*: este nível aplica-se a informações sensíveis que devem ser ocultadas do provedor. Este tipo de informação deve ser armazenado cifrado, usando chaves criptográficas especificadas pelo usuário e transformadas em repositórios isolados na nuvem. Esses repositórios são configurados, distribuídos e mantidos por um terceiro confiável compartilhado pelo provedor e pelo usuário. A figura 2.8 apresenta o modelo deste sistema e a interações entre o provedor, o usuário e o terceiro confiável.

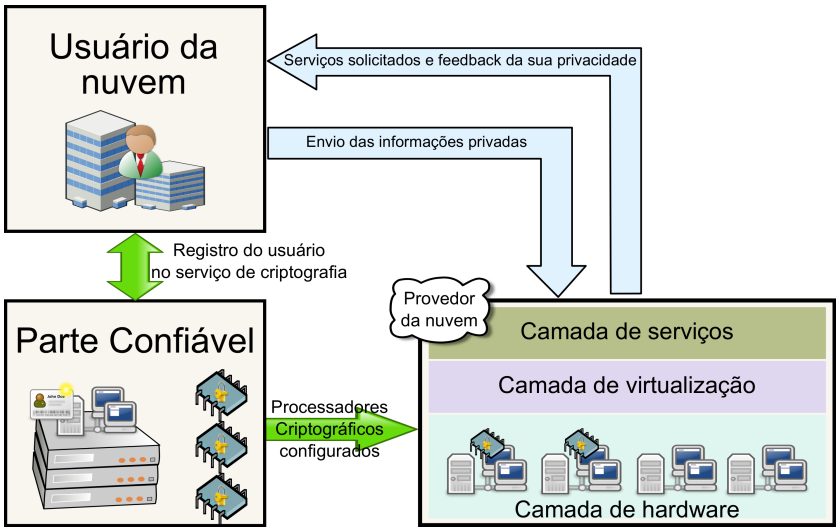


Figura 2.8. Modelo do serviço de privacidade *PasS* (adaptado de [Itani et al. 2009]).

Como pode ser observado na figura 2.8, a proposta faz uso combinado de processadores criptográficos e protocolos de privacidade. Os processadores são posicionados em um terceiro confiável (*trusted third party*) e acessível ao usuário e ao provedor. O uso de uma camada de virtualização é indicado para permitir o compartilhamento dos processadores entre vários usuários simultâneos. A solução prevê a criação de uma camada de software para separar os componentes do serviço a ser executados na nuvem, de acordo com os níveis de confiança definidos e suportados pelo modelo. Finalmente, o modelo especifica um protocolo de *feedback* sobre a privacidade obtida. O objetivo desse protocolo é informar o usuário sobre os níveis de privacidade de suas informações e alertar sobre a possibilidade de vazamento de dados ou outras situações de risco que poderiam comprometer a privacidade de suas informações.

2.4.6.4. Serviços de nuvem em conformidade com a legislação de privacidade

A fim de aderir à legislação governamental alemã sobre privacidade, o trabalho [Doelitzscher et al. 2010] propõe um modelo em camadas, denominado *CloudDataSec*,

para ser implementado em projetos de infraestrutura de nuvens. Este modelo é composto das seguintes camadas (figura 2.9):

- *Análise de risco*: estabelece e gerencia as avaliações de riscos sobre a terceirização de serviços na nuvem, auxiliando na identificação das informações e serviços que devem permanecer dentro dos limites da organização consumidora.
- *Orientações de segurança*: descreve as políticas e restrições legais relativas à privacidade aplicáveis ao ambiente de computação em nuvem.
- *Monitoração de qualidade de serviço (QoS)*: um acordo de nível de serviço (*Service Level Agreement* - SLA) entre clientes e fornecedores especifica os níveis de exigência de segurança e privacidade, além de garantir a segurança jurídica dos contratos sobre os serviços.
- *Criptografia dos dados e registros (logs)*: a criptografia visa proteger a confidencialidade e integridade das informações, enquanto os registros fornecem um histórico completo das atividades do usuário.
- *Comunicação criptografada*: nesta camada são utilizados os protocolos padronizados como SSH, IPSec e suas implementações.

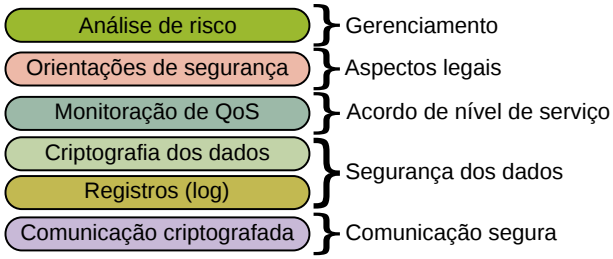


Figura 2.9. Camadas de segurança e privacidade para nuvem [Doelitzscher et al. 2010].

Para os autores, a aplicação do modelo *CloudDataSec* garante a aderência à legislação e a proteção das informações contra alguns ataques, como *man-in-the-middle* (ataque do intermediário). Além de propor um modelo de aderência, o trabalho sugere três níveis de garantia de segurança, sumarizados na tabela 2.1. Para ilustrar a aplicabilidade da proposta, o trabalho descreve um cenário onde uma empresa de desenvolvimento deseja terceirizar serviços na nuvem. Após uma análise de riscos, os seguintes níveis de segurança são identificados:

- *Nível Básico* para o desenvolvimento das páginas Web da empresa; não existem informações sensíveis armazenadas no servidor Web de desenvolvimento.
- *Nível Avançado* para hospedar o site da empresa. Não há restrição sobre o número de máquinas virtuais na mesma máquina física (*host*), mas somente máquinas virtuais desse domínio devem ser permitidas no mesmo *host*. No caso de um incidente de segurança, a máquina virtual comprometida é movida para a quarentena, e uma imagem íntegra da mesma é lançada, para propiciar alta disponibilidade ao serviço.

Tabela 2.1. Níveis de segurança para serviços na nuvem [Doelitzscher et al. 2010]

Serviço	Básico	Avançado	*Premium*
Local da VM	*pool aberto*	*pool* restrito	*host* privado
Identificação	e-mail, cartão	documento	terceiro confiável
Firewall	√	√	√
Administração	*GUI firewall*	*GUI firewall*	*GUI firewall*
Monitor de protocolo	–	√	√
Firewall de aplicação	–	√	√
Quarentena para sistemas comprometidos	–	√	√
Reinício do serviço	–	VMs seguras (até 3)	VMs seguras
Acesso ao sistema em quarentena	–	SSH	SSH, terminal

- *Nível Premium* para hospedar uma loja *on-line*; neste caso, um vazamento de dados dos usuários pode afetar a reputação da empresa. Em caso de incidente, o sistema é movido para a quarentena para evitar o vazamento de informações. O serviço não é relançado automaticamente, para evitar a repetição do ataque e a possibilidade de exposição de dados pessoais.

As implementações descritas nos cenários envolvem o desenvolvimento de um conjunto de módulos, entre os quais o Serviço de Gestão e Monitoramento de Segurança (*Security Management and Monitoring as a Service - SMaaS*). A Figura 2.10 ilustra a arquitetura do *SMaaS* em um ambiente de nuvem simplificado. Por sua vez, o módulo *SMaaS* é constituído dos seguintes componentes: Criptografia, Infraestrutura de Chave Pública (PKI), monitoramento de SLA, módulo de verificação de políticas (configuração, acesso, estado do sistema, etc), prevenção de vazamento dos dados (*Data Leakage Prevention* - DLP), registros (log) e um sistema de detecção de intrusão (IDS).

2.4.6.5. Preservação da privacidade em computação em nuvem

Existem muitos provedores de serviços na nuvem, possibilitando aos usuários acessar serviços diversos, mas quando as informações são trocadas entre os serviços, surge o problema da divulgação das informações e da consequente violação da privacidade [Wang et al. 2009]. A partir dessa afirmação, os autores propõem um novo algoritmo de anonimato a ser aplicados nos micro-dados (partes da informação) antes que estes sejam publicados na nuvem. Segundo os autores, o uso da criptografia não seria suficiente para garantir a privacidade das informações, já que o provedor do serviço precisa decifrar as informações antes do processamento.

Os autores indicam a utilização de uma base de conhecimento externa, não necessariamente armazenada na nuvem, como registros públicos ou outros canais na Internet. A junção dos dados anonimizados com a base externa possibilitaria ao provedor do serviço realizar pesquisas e obter os dados necessários para resolver alguns problemas.

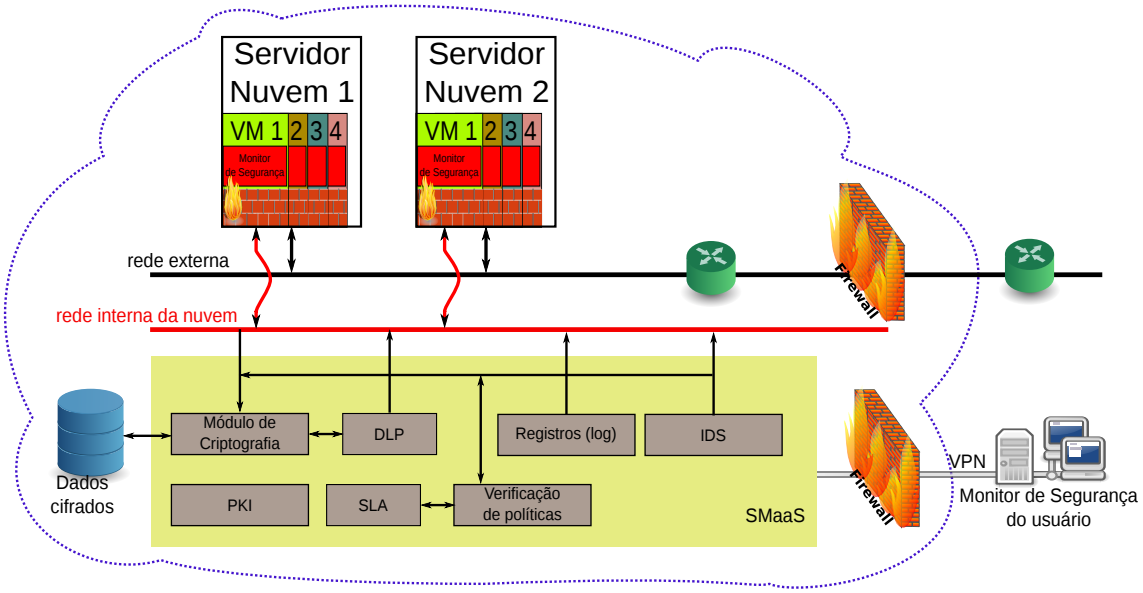


Figura 2.10. *SMaaS* em um ambiente de nuvem [Doelitzscher et al. 2010].

No trabalho os autores utilizam um cenário de acidentes de trânsito, onde informações como cores do veículo são suprimidas e profissão específica é substituída pela grande área (por exemplo: a informação "dentista" é substituída por "área médica", e "diretor de escola" seria substituída por "área de ensino"). Com este exemplo, os autores querem mostrar que é possível montar relatórios estatísticos confiáveis, mesmo sem acesso às informações completas.

2.4.6.6. Privacidade e Criptografia

A circulação e manutenção de dados pessoais na nuvem acaba por exigir o uso da criptografia. Todavia, esta impõe limites estritos para a utilização dos dados ao provedor de serviços, por exemplo, se os dados são armazenado em texto não-cifrado, pode-se procurar um documento especificando uma palavra-chave, o que deixa de ser possível com textos cifrados usando algoritmos tradicionais [Chow et al. 2009]. Contudo, pesquisas recentes em criptografia possibilitaram realizar algumas operações em dados cifrados, como indexação ou pesquisas [Song et al. 2000, Boneh e Waters 2006, Shen et al. 2009, Singh et al. 2010].

Outra possibilidade promissora consiste no uso de algoritmos de criptografia totalmente homomórfica (*fully homomorphic encryption*). Esses algoritmos permitem a aplicação de funções arbitrárias sobre dados criptografados, sem a necessidade de decifrá-los. Formalmente, considerando um algoritmo de criptografia totalmente homomórfica c, dados em claro $a_1, \ldots, a_n$ e seus respectivos dados cifrados $c_1 = c(a_1), \ldots, c_n = c(a_n)$, pode-se rapidamente computar $c(f(a_1, \ldots, a_n))$ a partir de $f(c_1), \ldots, f(c_n)$, para qualquer função computável f. Apesar da noção de cifragem totalmente homomórfica ter sido proposta por Rivest, Adleman e Dertouzos em 1978, só recentemente foram concebidos algoritmos que se enquadram nesta categoria [Gentry 2009]. A possibilidade de realizar computações arbitrárias sobre dados cifrados sem revelar seu conteúdo abre

uma série de novas oportunidades para a proteção da privacidade.

2.5. Problemas em aberto

A segurança das informações é um dos assuntos mais questionados quando está em discussão a utilização ou a migração dos sistemas tradicionais para a nuvem computacional. Os assuntos discutidos sempre recaem em algum aspecto relacionado à segurança, inclusive tópicos a princípio pouco relacionados com segurança, como desempenho [Somani e Chaudhary 2009]. Porém, se um sistema tem problemas de desempenho podem ocorrer condições de corrida ou violações por inconsistência nas políticas. Se a alteração de uma política não foi tornada efetiva devido à lentidão de atualização, então a demora na correção do problema pode causar perdas irreparáveis.

A computação em nuvem traz redução de gastos com recursos locais, por exemplo, porém podem haver riscos associados a essa vantagem porque é necessária a delegação do controle sobre os dados para entidades terceirizadas [Cachin et al. 2009]. Na verdade surge ai a necessidade forte de confiança (*trust*) do consumidor no provedor e vice-versa, pois se o provedor armazena dados cifrados, pode estar fornecendo um repositório para material criminoso, por exemplo, por outro lado se o consumidor não cifra seus dados pode ter sua privacidade (Seção 2.4) violada intencional ou acidentalmente [Birman et al. 2009]. Adicionalmente, países e regiões possuem suas próprias leis regrando a localização física do armazenamento dos dados e suas formas de proteção (e.g. *USA Patriot Act*).

A disponibilidade é uma das preocupações em serviços online, pois podem ocorrer períodos de inatividade ocasionados por falhas de comunicação, de softwares ou por ataques. O armazenamento de dados em locais remotos necessita de garantias de integridade fornecidas pelo provedor da nuvem. O mal funcionamento de um programa no provedor pode liberar o acesso aos dados privados do consumidor, então estes podem ser alterados os copiados. Empresas tradicionais têm receio de armazenar dados fora de seu domínio, porque uma vez que o dado tenha sido comprometido (perdido, danificado ou roubado) nenhum SLA ou contrato poderá reparar tal perda. Existe a preocupação com casos onde um serviço precisa estar conectado simultaneamente a diferentes provedores de nuvem para funcionar sendo que se um destes parar de operar – dependendo do nível de acoplamento funcional entre os provedores, diferentes tipos de clientes podem ser prejudicados.

Quando vários usuários utilizam um mesmo provedor de maneira colaborativa, a consistência durante o acesso concorrente aos recursos também deve ser garantida. Um mesmo sistema de arquivos pode estar sendo acessado por sistemas operacionais diferentes (máquinas virtuais), como se fosse um *storage* (para o qual não há um hardware e um servidor específico para a mediação dos acessos). Em computação em nuvem, se não houver um *data center* para o armazenamento de dados quem terá que fazer o papel de servidor de *storage* é o hipervisor e este não está preparado para tal função. Os desafios são muitos, envolvendo grandes áreas de estudo – abrangendo as diferentes camadas e modelos de implantação de computação em nuvem [Birman et al. 2009].

A grande maioria das APIs de armazenamento ainda são proprietárias, ou não foram totalmente padronizadas, assim os consumidores não podem extrair facilmente seus dados e programas de um domínio para outro. A padronização das APIs permitirá que desenvolvedores implantem serviços com portabilidade ou compatibilidade para

armazenamento de dados em vários provedores de computação em nuvem. Com esta abordagem é possível desenvolver técnicas de replicação ou *backup* para não depender integralmente de um único provedor, pois se este falhar o consumidor poderá acionar seu plano de contingência, recuperando cópias dos dados de outro provedor [Wood et al. 2010]. A padronização de APIs também permitirá que os mesmos softwares sejam utilizados em uma nuvem privada e pública – modelo híbrido (e.g. *Eucalyptus*) – o que mudaria seria apenas a origem e destino (locais/remotos) dos dados, por exemplo.

Atualmente, cada nuvem possui sua própria solução para o gerenciamento de identidades. Várias tentativas estão sendo propostas e adaptadas, porém ainda não existe nenhuma padronização ou abordagem que ofereça fácil portabilidade. A crescente demanda por segurança e conformidade com leis e políticas também pode ser uma boa razão para a utilização de computação em nuvem, pois somente uma entidade – o provedor de computação em nuvem – deverá ser capaz de arcar com os custos para fornecer elevados níveis de segurança e auditabilidade para um grande número de consumidores.

Requisitos de auditabilidade (e.g. *Human Services Health Insurance Portability and Accountability Act* – HIPAA) devem ser fornecidos para que dados corporativos possam ser movidos para nuvens computacionais, permitindo que o acesso aos mesmos possa ser rastreado [Wang et al. 2010a]. Empresas tradicionais enfrentam ameaças/ataques internos e externos, porém, na nuvem computacional as responsabilidades são multi-parte, ou seja, aos consumidores de serviços de nuvem cabe a preocupação com a segurança em nível de aplicação; ao provedor cabe prover segurança física, em nível de *firewall*, IaaS, sistema de arquivos, isolamento de VMs etc.; a entidades terceirizadas contratadas pelo consumidor (como o serviço de identidade, por exemplo) cabe gerenciar configurações e dados sensíveis (incluindo a privacidade).

O usuário da nuvem deve ter meios para se proteger de seu provedor, pois é este que controla a camada de software subjacente. O consumidor deve utilizar contratos com cláusulas claras e em conformidade com as leis do país onde a prestação do serviço de computação em nuvem está sendo oferecida. A auditoria deveria ser acessível como uma camada adicional, fora do domínio do administrador de sistema operacional virtualizado – ficando muito mais segura do que se fosse embutida na aplicação [Wang et al. 2010a].

2.6. Considerações finais

Ao longo deste trabalho, foram discutidas vantagens e limitações que da computação em nuvem. Nós entendemos que a computação em nuvem promete revolucionar o paradigma da computação principalmente por transferir para um provedor - especializado na prestação de serviço e infraestrutura computacional - uma responsabilidade que o consumidor precisa assumir mesmo não estando preparado para isto antes da mesma. Porém, como qualquer prestação de serviço a computação em nuvem, por ser recente, carece de credibilidade, pois aspectos de segurança e privacidade são sempre uma preocupação pelo danos que podem causar as vítimas de violações que possam causar aos mesmos.

O ambiente de computação em nuvem traz uma série de desafios à privacidade, por exemplo: como limitar a coleta das informações? Como garantir o uso correto das informações a partir do momento que estas foram armazenadas na nuvem? Como garantir a destruição das informações na nuvem? A transição da computação pessoal para a

computação em nuvem fornece uma nova gama de aplicações e possibilidades de usos, mas também implica em novos problemas e desafios. A inadequação dos mecanismos convencionais de gestão de privacidade e a falta de ferramentas tecnológicas adequadas para permitir aos usuários controlar a exposição de suas informações sensíveis são limitações que limitam o uso desse ambiente.

Na prática, mesmo antes da computação o ser humano sempre teve que lidar com problemas de segurança e privacidade, a diferença sempre foi o contexto e os mecanismo utilizados para tal. Assim, acreditamos que temas como confiança (*trust*) e privacidade ganhem ainda mais importância porque não se espera, como comentado no texto, que um sujeito que não confia em um banco deposite seu dinheiro lá. Ainda, não se espera que um sujeito que não confia na capacidade de guardar sigilo de um contabilista entregue suas informações privadas ao mesmo. Mas, se espera que a computação em nuvem traga muitas opções de escolha para que os provedores que oferecem o melhor conjunto de propriedades de segurança e de privacidade sejam os mais demandados e possam melhorar cada vez seus serviços, até atingirem níveis satisfatórios de prestação de serviço.

Referências

[Armbrust et al. 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., e Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

[Beaver e Harold 2004] Beaver, K. e Harold, R. (2004). *The Practical Guide to HIPAA Privacy and Security Compliance*. Auerbach Publications.

[Bertino et al. 2009] Bertino, E., Paci, F., Ferrini, R., e Shang, N. (2009). Privacy-preserving digital identity management for cloud computing. *IEEE Data Engineering Bulletin*, 32(1):21–27.

[Bhattacharjee 2009] Bhattacharjee, R. (2009). *An analysis of the cloud computing platform*. MSc thesis, System Design and Management Program, Massachusetts Institute of Technology.

[Birman et al. 2009] Birman, K., Chockler, G., e van Renesse, R. (2009). Toward a cloud computing research agenda. *ACM SIGACT News*, 40(2):68–80.

[Boneh e Waters 2006] Boneh, D. e Waters, B. (2006). Conjunctive, subset, and range queries on encrypted data. Em *Theory of Cryptography Conference (TCC)*, páginas 535–554. Springer.

[Cachin et al. 2009] Cachin, C., Keidar, I., e Shraer, A. (2009). Trusting the cloud. *ACM SIGACT News*, 40(2):81–86.

[Caron et al. 2009] Caron, E., Desprez, F., Loureiro, D., e Muresan, A. (2009). Cloud computing resource management through a grid middleware. Em *IEEE Conference on Cloud Computing*.

[Chow et al. 2009] Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., e Molina, J. (2009). Controlling data in the cloud: outsourcing computation without outsourcing control. Em *ACM workshop on Cloud Computing Security*, páginas 85–90, New York, NY, USA. ACM.

[CSA 2009] CSA (2009). *Security Guidance for Critical Areas of Focus in Cloud Computing – v2.1*. Cloud Security Alliance.

[CSA 2010a] CSA (2010a). Domain 12: Guidance for identity & access management v2.1. http://www.cloudsecurityalliance.org/guidance/csaguide-dom12-v2.10.pdf.

[CSA 2010b] CSA (2010b). *Top Threats to Cloud Computing V1.0*. Cloud Security Alliance.

[Dalton et al. 2009] Dalton, C. I., Plaquin, D., Weidner, W., Kuhlmann, D., Balacheff, B., e Brown, R. (2009). Trusted virtual platforms: A key enabler for converged client devices. *ACM SIGOPS Operating Systems Review*, 43(1):36–43.

[Dawoud et al. 2010] Dawoud, W., Potsdam, G., Takouna, I., e Meinel, C. (2010). Infrastructure as a service security: Challenges and solutions. Em *7th International Conference on Informatics and Systems (INFOS)*.

[De Capitani di Vimercati e Samarati 2006] De Capitani di Vimercati, S. e Samarati, P. (2006). Privacy in the electronic society. Em *International Conference on Information Systems Security (ICISS)*, Kolkata, India. invited talk.

[Doelitzscher et al. 2010] Doelitzscher, F., Reich, C., e Sulistio, A. (2010). Designing cloud services adhering to government privacy laws. Em *International Symposium on Trust, Security and Privacy for Emerging Applications*.

[Erickson et al. 2009] Erickson, J. S., Spencer, S., Rhodes, M., Banks, D., Rutherford, J., Simpson, E., Belrose, G., e R., R. P. (2009). Content-centered collaboration spaces in the cloud. *IEEE Internet Computing*, páginas 34–42.

[Ernst & Young 2010] Ernst & Young (2010). Insights in it risk – top privacy issues for 2010.

[Etsion et al. 2009] Etsion, Y., Ben-Nun, T., e Feitelson, D. (2009). A global scheduling framework for virtualization environments. Em *IEEE Symposium on Parallel and Distributed Processing*.

[Eucalyptus 2010] Eucalyptus (2010). Eucalyptus – the open source cloud plataform. http://open.eucalyptus.com.

[FIPS 140-2 2001] FIPS 140-2 (2001). *Security Requirements for Criptographic Modules - FIPS PUB 140-2*. Computer Security Division.

[Fischer-Hübner 2001] Fischer-Hübner, S. (2001). IT-Security and Privacy: Design and use of privacy-enhancing security mechanisms. Em Goos, G., Hartmanis, J., e van Leeuwen, J., editores, *Lecture Notes in Computer Science*, volume 1958. Springer-Verlang.

[Foster et al. 2008] Foster, I., Zhao, Y., Raicu, I., e Lu, S. (2008). Cloud computing and grid computing 360-degree compared. Em *Grid Computing Environments Workshop*.

[Gentry 2009] Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. Em *Annual ACM Symposium on Theory of computing*, páginas 169–178, New York, NY, USA. ACM.

[Goyal e Mikkilineni 2009] Goyal, P. e Mikkilineni, R. (2009). Policy-based event-driven services-oriented architecture for cloud services operation & management. Em *IEEE International Conference on Cloud Computing*, páginas 135–138. IEEE Computer Society.

[Grobauer et al. 2010] Grobauer, B., Walloschek, T., e Stöcker, E. (2010). Understanding Cloud-Computing Vulnerabilities. *IEEE Security and Privacy*.

[Grossman 2009] Grossman, R. (2009). The case for cloud computing. *IT Professional*, 11(2).

[Gruschka e Iacono 2009] Gruschka, N. e Iacono, L. (2009). Vulnerable cloud: SOAP message security validation revisited. Em *IEEE International Conference on Web Services*.

[Hayes 2008] Hayes, B. (2008). Cloud computing. *Communications of the ACM*, 51(7):9–11.

[Hinde 2003] Hinde, S. (2003). Privacy legislation: a comparison of the US and european approaches. *Computers & Security*, 22(5):378–387.

[ISO 2005] ISO (2005). *ISO/IEC 27001 - Information technology - Security techniques - Information security management systems - Requirements*. International Organization for Standartization.

[Itani et al. 2009] Itani, W., Kayssi, A., e Chehab, A. (2009). Privacy as a service: Privacy-aware data storage and processing in cloud computing architectures. Em *IEEE International Conference on Dependable, Autonomic and Secure Computing*, páginas 711 –716.

[ITIL 2010] ITIL (2010). *IT Infrastructure Library*. Office of Governance Commerce, UK.

[ITU-T 2000] ITU-T (2000). *Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks. Recommendation X.509*. International Telecommunication Union.

[Jensen et al. 2009] Jensen, M., Schwenk, J., Gruschka, N., e Iacono, L. (2009). On technical security issues in cloud computing. Em *IEEE International Conference on Cloud Computing*, páginas 109–116. IEEE Computer Society.

[Kaliski Jr e Pauley 2010] Kaliski Jr, B. e Pauley, W. (2010). Toward risk assessment as a service in cloud environments. Em *USENIX Workshop on Hot Topics in Cloud Computing*.

[Kandukuri et al. 2009] Kandukuri, B., Paturi, V., e Rakshit, A. (2009). Cloud security issues. Em *International Conference on Services Computing*.

[Landwehr 2001] Landwehr, C. (2001). Computer security. *International Journal of Information Security*, 1(1):3–13.

[Laureano e Maziero 2008] Laureano, M. e Maziero, C. (2008). Virtualização: Conceitos e aplicações em segurança. Em Maziero, C., editor, *Livro-Texto de Minicursos SBSeg*, páginas 1–50. Sociedade Brasileira de Computação.

[Mather et al. 2009] Mather, T., Kumaraswamy, S., e Latif, S. (2009). *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O'Reilly Media.

[Mell e Grance 2009] Mell, P. e Grance, T. (2009). The NIST definition of cloud computing. *National Institute of Standards and Technology*.

[Nurmi et al. 2009] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., e Zagorodnov, D. (2009). The eucalyptus open-source cloud computing system. Em *IEEE/ACM International Symposium on Cluster Computing and the Grid*.

[OASIS 2005a] OASIS (2005a). *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS.

[OASIS 2005b] OASIS (2005b). *eXtensible Access Control Markup Language version 2.0*.

[OASIS 2005c] OASIS (2005c). *SAML 2.0 profile of XACML v2.0*. OASIS.

[OASIS 2006] OASIS (2006). *Service Provisioning Markup Language (SPML) Version 2*. OASIS.

[OASIS 2009a] OASIS (2009a). *Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of Security Assertion Markup Language (SAML) for Healthcare Version 1.0*. OASIS.

[OASIS 2009b] OASIS (2009b). *Web Services Federation Language version 1.2*. OASIS.

[Ogrizovic et al. 2010] Ogrizovic, D., Svilicic, B., e Tijan, E. (2010). Open source science clouds. Em *International Convention (MIPRO 2010)*.

[OpenID 2010] OpenID (2010). *OpenID Foundation - OIDF*. OpenID Foundation.

[Pearson et al. 2009] Pearson, S., Shen, Y., e Mowbray, M. (2009). A privacy manager for cloud computing. Em Jaatun, M., Zhao, G., e Rong, C., editores, *Cloud Computing*, volume 5931 of *LNCS*, páginas 90–106. Springer. 10.1007/978-3-642-10665-1-9.

[Pedersen 1992] Pedersen, T. (1992). Non-interactive and information-theoretic secure verifiable secret sharing. Em Feigenbaum, J., editor, *Advances in Cryptology*, volume 576 of *Lecture Notes in Computer Science*, páginas 129–140. Springer Berlin / Heidelberg.

[Pfleeger e Pfleeger 2006] Pfleeger, C. P. e Pfleeger, S. L. (2006). *Security in Computing*. Prentice Hall, fourth edition.

[Pinheiro Jr e Kon 2005] Pinheiro Jr, J. e Kon, F. (2005). Segurança em grades computacionais. Em *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*.

[Provos et al. 2009] Provos, N., Rajab, M., e Mavrommatis, P. (2009). Cybercrime 2.0: when the cloud turns dark. *Communications of the ACM*, 52(4):42–47.

[Rezgui et al. 2003] Rezgui, A., Bouguettaya, A., e Eltoweissy, M. Y. (2003). Privacy on the web: Facts, challenges, and solutions. *IEEE Security and Privacy*, 1(6):40–49.

[Rimal et al. 2009] Rimal, B., Choi, E., e Lumb, I. (2009). A taxonomy and survey of cloud computing systems. Em *International Joint Conference on INC, IMS and IDC*.

[Ristenpart et al. 2009] Ristenpart, T., Tromer, E., Shacham, H., e Savage, S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. Em *ACM conference on Computer and communications security*, páginas 199–212. ACM.

[Shen et al. 2009] Shen, E., Shi, E., e Waters, B. (2009). Predicate privacy in encryption systems. Em Reingold, O., editor, *Theory of Cryptography*, volume 5444 of *Lecture Notes in Computer Science*, páginas 457–473. Springer Berlin / Heidelberg.

[Shirey 2000] Shirey, R. (2000). *RFC 2828 - Internet Security Glossary*. The Internet Society.

[Singh et al. 2010] Singh, M. D., Krishna, P. R., e Saxena, A. (2010). A cryptography based privacy preserving solution to mine cloud data. Em *Annual ACM Bangalore Conference*, páginas 1–4, New York, NY, USA. ACM.

[Somani e Chaudhary 2009] Somani, G. e Chaudhary, S. (2009). Application performance isolation in virtualization. Em *IEEE International Conference on Cloud Computing*, páginas 41–48. IEEE Computer Society.

[Song et al. 2000] Song, D. X., Wagner, D., e Perrig, A. (2000). Practical techniques for searches on encrypted data. Em *IEEE Symposium on Security and Privacy*, páginas 44 –55.

[Stahl 2008] Stahl, B. C. (2008). The impact of the UK Human Rights Act 1998 on privacy protection in the workplace. Em Subramanian, R., editor, *Computer Security, Privacy, and Politics - Current Issues, Challenges, and Solutions*, chapter IV, páginas 55–69. IRM Press.

[Staples 2007] Staples, W. G. (2007). *Encyclopedia of Privacy*. Greenwood Press.

[Sweeney 2002] Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570.

[Trusted Computing Group 2010] Trusted Computing Group (2010). *Trusted Platform Module – Specifications*. Trusted Computing Group.

[Turn e Ware 1975] Turn, R. e Ware, W. H. (1975). Privacy and security in computer systems. Technical Report P5361, Rand Corporation.

[VMWare Inc 2010] VMWare Inc (2010). VMware vSphere. http://www.VMware.com/vSphere.

[W3C 2001] W3C (2001). *XML Key Management Specification (XKMS)*. W3 Consortium.

[W3C 2010] W3C (2010). *Web Services Policy 1.5 – Framework*. W3 Consortium.

[Wang et al. 2010a] Wang, C., Wang, Q., Ren, K., e Lou, W. (2010a). Privacy-preserving public auditing for data storage security in cloud computing. Em *IEEE International Conference on Computer Communications*.

[Wang et al. 2010b] Wang, H., Jing, Q., Chen, R., He, B., Qian, Z., e Zhou, L. (2010b). Distributed systems meet economics: Pricing in the cloud. Em *USENIX Workshop on Hot Topics in Cloud Computing*.

[Wang et al. 2009] Wang, J., Shao, Y., Jiang, S., e Le, J. (2009). Providing privacy preserving in cloud computing. Em *International Conference on Test and Measurement*, páginas 213–216. IEEE Computer Society.

[Ware 1973] Ware, W. H. (1973). Records, computers and the rights of citizens. Technical Report P5077, Rand Corporation.

[Wood et al. 2010] Wood, T., Cecchet, E., Ramakrishnan, K., Shenoy, P., van der Merwe, J., e Venkataramani, A. (2010). Disaster recovery as a cloud service: Economic benefits & deployment challenges. Em *USENIX Workshop on Hot Topics in Cloud Computing*.

[Wright 2004] Wright, T. (2004). *Security, Privacy, and Anonymity*. ACM.

[Yee e Korba 2009] Yee, G. e Korba, L. (2009). Personal privacy policies. Em Vacca, J., editor, *Computer and Information Security Handbook*, páginas 487–505. Morgan Kaufmann.

[Yildiz et al. 2009] Yildiz, M., Abawajy, J., Ercan, T., e Bernoth, A. (2009). A layered security approach for cloud computing infrastructure. Em *International Symposium on Pervasive Systems, Algorithms, and Networks*, páginas 763–767. IEEE.

[Zhang e Zhou 2009] Zhang, L. e Zhou, Q. (2009). CCOA: Cloud computing open architecture. Em *International Conference on Web Services*.

[Zhang et al. 2010] Zhang, Q., Cheng, L., e Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Springer Journal of Internet Services and Applications*, páginas 7–18.

[Zhao et al. 2009] Zhao, Y., Xie, Y., Yu, F., Ke, Q., Yu, Y., Chen, Y., e Gillum, E. (2009). Botgraph: Large scale spamming botnet detection. Em *USENIX Symposium on Networked systems design and implementation*, páginas 321–334. USENIX Association.

Capítulo 3

Transformações de código para proteção de software

Davidson Rodrigo Boccardo, Raphael Carlos Santos Machado e Luiz Fernando Rust da Costa Carmo

Abstract

This chapter presents software transformation techniques for protection of intellectual property and security of sensitive data against 'Man-At-The-End' attacks. Security is normally associated with the confidentiality of the communication channel based on cryptography. Security based on software protection is associated with code obfuscation, watermarking and tamper-proofing techniques. Code obfuscation aims to make harder for a reverse engineer to understand the inner properties of the code. Watermarking is a defense against piracy, in which a mark is embedded inside the code for further tracking of legal authorities. Tamper-proofing aims to ensure that the program executes as expected affront of monitoring and/or changing. The goal of all these techniques is to maintain the security of the software code running at the end user, and its protection is extremely important to ensure that the running software is executing as its purpose.

Resumo

Este capítulo apresenta técnicas de transformação de código para proteção da propriedade intelectual e segurança dos dados contra ataques 'Man-At-The-End'. A segurança convencional é normalmente atrelada com à confidencialidade do canal de comunicação, e esta é baseada em métodos de criptografia. Segurança baseada em proteção de software está associada a técnicas de ofuscação, marca d'água e 'tamper-proofing'. As técnicas de ofuscação de código visam a dificultar o entendimento do código quando neste é aplicado engenharia reversa. As técnicas de marca d'água são defesas contra pirataria, em que uma marca é embarcada no código do programa para rastreabilidade a posteriori em uma ação legal. As técnicas de 'tamper-proofing' garantem que o programa execute como esperado diante de modificação e monitoração. O objetivo de todas estas técnicas é dar maior credibilidade ao software em execução no sistema final assegurando que o software em execução comporte-se de maneira legítima.

3.1. Introdução

Tradicionalmente, segurança de computadores está relacionada ao cenário em que um computador está sob ataque por um adversário ou por um código malicioso criado por este adversário. A pesquisa nesta área envolve o desenvolvimento de técnicas para prevenir que um adversário tenha acesso a um outro computador, ou que emita um alerta em uma situação de ataque. Estas técnicas visam a prevenir, detectar e/ou interromper um ato malicioso. Neste contexto temos *firewalls* que permitem restringir o acesso a um determinado computador; sistemas de detecção de intrusão, que alertam o usuário de um computador diante de um padrão de acesso à rede suspeito; e os varredores de código malicioso, que bloqueiam a execução de código caso apresente comportamento malicioso.

Em ataques *Man-At-The-End*, em que um adversário tem acesso físico ao software/hardware com poder de comprometê-lo, seja por inspeção ou por modificação, modelos tradicionais de segurança não são suficientes. Uma técnica comum para esconder informação dentro de um programa consiste na utilização de técnicas de criptografia. Nesta situação, um desenvolvedor esconderia o segredo de seu software — algoritmo, chave criptográfica ou número serial — usando algum algoritmo criptográfico. Contudo, neste cenário temos que proteger a chave de criptografia, pois caso desvendada o segredo seria revelado. Além disso, para que este código fosse executável, este precisaria ser decriptografado em algum instante, o que possibilitaria de um adversário extrair o código no momento que este estivesse em texto claro. Com acesso ao código, é só uma questão de tempo e motivação para que um adversário consiga extrair o segredo de um programa.

Avanços em análise de programas e tecnologias de engenharia de software têm levado ao aperfeiçoamento das ferramentas para análise de programas e desenvolvimento de software. Estes fornecem recursos para que os softwares sejam mais eficientes e livres (o quanto possível) de vulnerabilidades. No entanto, estes mesmos avanços incrementam o poder dos softwares para engenharia reversa com o objetivo de descobrir vulnerabilidades, realizar alterações indevidas, ou furtar propriedade intelectual. A engenharia reversa exige a reconstrução do código executável para alguma estrutura de alto nível. Por exemplo, para identificação de vulnerabilidades em um software, um adversário teria que primeiro identificá-las para depois atacá-las. Analogamente, para furtar um trecho de código na presença de um direito autoral ou marca d'água, um adversário teria que analisar o código para que fosse possível modificá-lo sem afetar a funcionalidade do programa.

Neste texto, exploramos técnicas de transformação de código para proteção de software com o intuito de dificultar ataques *Man-At-The-End*. Atualmente, existe pouco conhecimento, entre os desenvolvedores de software, sobre a importância do uso destas técnicas. As técnicas de proteção de software diferem das técnicas de segurança de computadores convencional, pois uma vez que um adversário tenha acesso ao programa, não existem limites do que o mesmo pode fazer. Uma técnica de proteção de software pode ser definida como um conjunto de procedimentos para dificultar um adversário de obter ganhos sobre o software, seja por motivos financeiros, sabotagem ou vingança. O conhecimento destas técnicas exerce maior impacto em cenários que podem afetar infra-estruturas nacionais caso atacadas, como *Smart Grids* do setor elétrico e redes de sensores sem fio da área militar. Um cenário de ataque em *Smart Grids* envolve perpetuar ataques na infra-estrutura de medição de forma a causar um *denial of service* na rede elétrica.

Na área militar um adversário poderia modificar os sensores para que estes enviassem informações distorcidas, por exemplo quanto ao movimento das tropas, para a central de comando.

Ataques em software normalmente envolvem duas etapas: análise e modificação. Na etapa de análise, um adversário busca compreensão do software, enquanto na etapa de modificação um adversário subverte o software de acordo com seu objetivo. Um ataque típico envolve fazer a engenharia reversa do software da concorrência para extrair conhecimento de algum módulo ou do software como um todo, para *a posteriori*, modificá-lo para revenda.

Existem diferentes abordagens de engenharia reversa e definir qual abordagem mais apropriada depende da aplicação alvo, da plataforma em que roda a aplicação, onde foi desenvolvida e qual tipo de informação deseja extrair. Existem duas metodologias de engenharia reversa: análise estática e análise dinâmica.

A análise estática de programas é uma técnica utilizada para coletar informações sobre o programa sem a necessidade de executá-lo. A análise envolve o uso de um *disassembler* que faz a conversão do código binário para código *assembly*, cujo formato é mais compreensível por um humano do que uma sequência de bits. A análise estática exige um melhor entendimento do programa (comparado com a análise dinâmica) pois o fluxo da execução é somente baseado no código do programa. Ferramentas de análise de fluxo de controle coletam informações para uma melhor compreensão sobre o fluxo de execução do programa. Como o fluxo pode ser dependente dos valores contidos em posições de memória, análise do fluxo de dados predizem o conjunto de valores que determinada memória pode vir a assumir durante a execução do programa. Outra ferramenta que se enquadra em análise estática são os decompiladores que tentam reverter o processo de compilação para produção de um código de alto nível. Contudo, na maioria das arquiteturas, a recuperação do código de alto nível não é realmente possível, pois existem elementos significativos que são removidos durante a compilação uma vez que o âmbito deste é otimização e não a legibilidade do código.

A análise dinâmica de programas é uma técnica em que a coleta de informações é feita em tempo de execução diante de uma determina entrada. Nesta análise, é possível observar como a entrada interage como o fluxo de execução e os dados do programa. A análise basicamente envolve o uso de um depurador (*debugger*) que permite um analista observar o programa durante sua execução. Um depurador possui tipicamente duas características básicas: habilidade de ajustar pontos de parada (*breakpoints*) no programa e capacidade de verificar o estado atual do programa (registradores, memória, conteúdo da pilha). A análise dinâmica também pode ser feita através de emulação de código. Nesta análise, o código é executado em um emulador que provê funcionalidades similares ao sistema original, assim como um arcabouço para auxiliar um analista no entendimento do código. Técnicas de *profiling* e *tracing* também são consideradas como dinâmicas. A primeira é utilizada para contagem do números de execuções, ou a quantidade de tempo despendida, de diferentes partes do código; a segunda em vez de realizar somente a contagem, registra também quais partes do código do programa foram executadas.

As técnicas de proteção de um software visam a dificultar a engenharia reversa, assegurando que o software executa como esperado e protegendo o software de pira-

taria [18, 19, 16, 17, 30]. Entre as técnicas de proteção de software, destacamos três: ofuscação, marca d'água e *tamper-proofing*. Técnicas de ofuscação de código dificultam engenharia reversa através de um aumento no grau de ininteligibilidade do código. Este aumento se caracteriza por modificações sintáticas que dificultam a compreensão do software mantendo, porém, as funcionalidades originais do programa (sua semântica). Técnicas de *tamper-proofing* asseguram que o software executa como esperado, mesmo na tentativa de modificação ou monitoração por um adversário. Técnicas de marca d'água agem como uma defesa contra pirataria de software cujo papel é determinar a origem de um software. Estas técnicas são baseadas na inclusão de informação no software em modo escondido que possa ser posteriormente reconhecido.

Técnicas de ofuscação de código adicionam confusão em um programa com a finalidade de dificultar o discernimento/extração/modificação de alguma propriedade desse programa, preservando, entretanto, sua funcionalidade. Desenvolvedores de software podem utilizar técnicas de ofuscação para esconder informações a respeito de um algoritmo por questões de propriedade intelectual — incluindo rotinas, chaves criptográficas e números seriais — assim como para incrementar a segurança do código contra a exploração maliciosa de vulnerabilidades, dada a dificuldade de se encontrar vulnerabilidades em um software de difícil discernimento. Por outro lado, ofuscação de software também tem sido utilizado por programadores maliciosos para difundir seus códigos maliciosos. Técnicas de ofuscação também podem ser utilizadas maliciosamente para distorcimento ou corrupção da marca d'água.

As transformações advindas de técnicas de ofuscação de código podem ocorrer tanto no código-fonte — que posteriormente é compilado de maneira a gerar um binário mais complexo — como diretamente no código binário. Alguns trabalhos propõem transformações a serem aplicadas sobre "código intermediário", visando, por exemplo, a proteção de *bytecodes* Java. Visto que sua compreensão é mais simples que a de código binário, e alvo de muitos decompiladores, sua proteção torna-se ainda mais crítica.

A ofuscação de um código busca dificultar a compreensão ou levar o adversário a desistir de analisar um determinado software. A engenharia reversa é o passo anterior à modificação deste software com o objetivo de obter alguma espécie de vantagem. Evitar violações à integridade do software e responder a violações — possivelmente impedindo a execução do software modificado ou redirecionando a execução para um código não funcional — são os objetivos das técnicas de *tamper-proofing*.

Estratégias comuns de *tamper-proofing* envolvem auto-verificação do *hash* de trechos de código ou a inspeção lógica do programa. Por exemplo, conhecendo previamente o *hash* de trechos de código ou o resultado que uma determinada variável deva possuir em um determinado ponto do programa, é possível verificar se os valores durante a execução do programa correspondem com os valores previamente conhecidos. Outra responsabilidade das técnicas de *tamper-proofing* diz respeito à ação a ser tomada diante de uma modificação. Estratégias comuns envolvem desviar o fluxo de execução para um código não funcional ou notificar o desenvolvedor que o código foi alterado. Outras estratégias menos amenas envolvem atacar o próprio sistema do adversário, por exemplo, apagando seus documentos pessoais. Para que as tarefas de verificação e resposta de uma técnica de *tamper-proofing* sejam menos "visíveis" para um adversário, estas costumam estar

dispersas no código do programa, assim como não executarem sequencialmente.

As técnicas de marca d'água de software são importantes para combate à pirataria de software. Embora marca d'água, por si só, não impeça um ato de violação de software, esta, no mínimo, desestimula tal ato e favorece a localização do violador do software. Marca d'água de software é caracterizada por uma informação de identificação codificada no código do programa que posteriormente possa ser reconhecida. Idealmente, uma marca d'água estaria tão atrelada ao software e ao cliente que qualquer região suficientemente grande do código possuiria tal informação o que inviabilizaria um processo de remoção.

Uma estratégia de marca d'água comum e de pouco impacto na execução do software envolve a inclusão do nome do autor em algum ponto do programa. Contudo, esta estratégia é de fácil localização e subversão um vez que um adversário pode modificar o nome do autor ou simplesmente substituir a porção de código do autor por um código não operante.

Dada a impossibilidade de esconder o código de um programa em cenários em que existe a possibilidade de violação física, métodos de ofuscação, *tamper-proofing* e marca d'água visam a tornar o processo de extrair conhecimento, piratear e modificar o código de um programa em um processo proibitivo quanto a custo, tempo e poder computacional. Outra possibilidade para esconder código, dados e execução do programa consiste em métodos baseados em hardwares especializados [20]. Contudo, estes apresentam desvantagens de custo de desenvolvimento, desempenho e perda de flexibilidade. Com estas desvantagens, é difícil um desenvolvedor convencer um cliente da importância destes métodos uma vez que tais métodos visam à proteção contra o próprio cliente (no caso de um cliente mal intencionado). Além disso, sistemas de software baseado em hardware tem sido alvos de ataque [48, 60].

Uma estratégia comum para combate a pirataria baseada em hardware atrela o funcionamento de um programa com alguma característica externa, seja esta dependente do ambiente de trabalho do cliente (número serial do disco rígido, CPU, etc...) ou por um dispositivo externo enviado pelo próprio desenvolvedor (*dongle*). Uma desvantagem de anexar a funcionalidade de um programa com uma característica externa é o transtorno gerado e quantidade de tempo despendida em situações de troca de equipamento (*upgrade* ou defeito) ou perda. Para agravar ainda mais a situação, o desenvolvedor poderia ter decretado falência o que inutilizaria o programa.

Uma estratégia contra engenharia reversa seria atrelar a CPU do usuário com os programas através de um esquema de criptografia de chave pública. Neste caso, a CPU conteria uma chave privada enquanto os programas só executariam se fossem criptografados com a correspondente chave pública. As desvantagens são as mesmas supracitadas, ou seja, custo elevado, desempenho inferior e estresse em caso de troca.

Todos meios de proteção, sejam eles em software ou em hardware, são passíveis de ataques e podem ser subvertidos. Mecanismos baseados em hardware oferecem um nível de proteção mais elevado, porém representam um custo agregado maior. Já as técnicas de transformação de código podem adicionar diferentes camadas de proteção a um custo inferior e, nada impede que ambas formas de proteção coexistam. O balanceamento entre o nível de proteção com o *overhead* em desempenho e custo de desenvolvimento é uma

decisão que cabe ao desenvolvedor do produto.

O restante deste capítulo é estruturado da seguinte forma. Seção 3.2 apresenta técnicas para ofuscação de código. Seção 3.3 apresenta técnicas de *anti-tampering*. Seção 3.4 descreve técnicas de marca d'água. E por fim Seção 3.5 apresenta as conclusões pertinentes.

3.2. Ofuscação de código

Técnicas de ofuscação de código alteram a estrutura de instruções de um programa com a intenção de fazer o conjunto menos aparente, dificultando com isso, o processo de engenharia reversa por um adversário [28, 36]. Estas técnicas tem sido utilizadas para proteger aplicações em situações que um competidor pode utilizar de engenharia reversa para extrair módulos/algoritmos proprietários [14]. Assim, ofuscação de código tem sido provada como uma técnica útil para esconder informação importante dentro de um software. Ofuscação de código tem sido utilizado para prover:

- **Diversidade**, em que a criação de versões distintas de um código sintaticamente dificulta ataques de diferença ("*diffing*") e exploração de vulnerabilidades por agentes maliciosos;
- **Proteção de propriedade intelectual**, em que o entendimento e a extração de códigos proprietários ou dados sigilosos são dificultados;
- **Integridade**, visto que um código mais complicado de entender é mais complicado de ser modificado.

Por outro lado, como qualquer conhecimento, ofuscação de código pode ser utilizada em um contexto malicioso. Especificamente, desenvolvedores de código malicioso frequentemente fazem uso de ofuscação de código para prevenir análise por especialistas ou por ferramentas de análise, assim como para evasão de detecção por detectores de código malicioso. Os trabalhos de Fred Cohen em construção e detecção de códigos maliciosos e ofuscação para criação de diversidade foram pioneiros na área de ofuscação de código [4, 5, 6, 7].

Ofuscar um programa refere-se a produzir um novo programa semanticamente equivalente em que a extração de informações por engenharia reversa é mais difícil de ser realizada neste do que no programa original. O conceito de dificuldade está associado à quantidade de tempo, dinheiro e poder computacional despendidos a mais na análise do programa transformado em relação ao programa original. O impacto de uma determinada transformação é mensurável de acordo com o *overhead*, confusão e dificuldade de um adversário de reverter a ofuscação [14]. As transformações efetivadas por um ofuscador de código são ditadas pela quantidade de *overhead* que um desenvolvedor esta disposto a colocar em sua aplicação. A seguir, detalhamos alguns cenários em que técnicas de ofuscação podem ser utilizadas.

- **Engenharia reversa maliciosa**. Um adversário faz uso de engenharia reversa para obter alguma vantagem pessoal sobre um determinado programa. Este pode, por

exemplo, desvendar o número serial de um programa para que o use sem restrições funcionais. Contudo, o adversário também pode implementar um gerador de número serial ou um *patch* que uma vez distribuído para outros usuários causaria prejuízos para o desenvolvedor, pois um usuário com acesso ao gerador de número serial ou *patch* não necessitaria adquirir o programa. Um outro exemplo de adversário, envolve um desenvolvedor concorrente extrair conhecimento de um algoritmo/módulo proprietário para utilizá-lo em seus próprios produtos.

- **Gerenciamento de direitos digitais (DRM)**. DRM (Digital Rights Managements) é uma tecnologia que visa restringir a cópia de um conteúdo digital por parte dos usuários. Em sistemas DRM de aúdio tipicamente uma mídia é criptografada com uma determinada chave criptográfica que é embarcada em um aparelho reprodutor. Para ouvir a mídia a mesma tem que ser decriptografada pelo aparelho reprodutor. Um adversário faz uso de engenharia reversa para obter a chave criptográfica embarcada no aparelho, possibilitando-o converter uma mídia criptografada em uma mídia em texto claro. Com a mídia em texto claro, um adversário pode escutá-la em qualquer aparelho reprodutor como pode distribuí-la para outros usuários.

- **Infra-estrutura avançada de medição**. O número de dispositivos de medição baseados em software embarcado vem crescendo consideravelmente. Um risco crítico associado à presença de software em dispositivos de medição é o interesse na adulteração do software para se obter vantagens pessoais. Por exemplo, para medidores elétricos um adversário poderia utilizar de engenharia reversa para modificar o software, permitindo-o conectar e desconectar clientes em tempos específicos remotamente, modificar dados de medição ou constantes de calibração, alterar a frequência de comunicação e inutilizar o dispositivo de medição. Em uma rede fortemente interligada como em *Smart Grids*, um adversário poderia causar um colapso na rede elétrica nacional caso estes softwares fossem atacados [59].

- **Redes de sensores sem fio**. Softwares também estão presentes em redes de sensores sem fio. Analogamente ao cenário anterior, é crítico que estes softwares não sofram adulteração. Em um contexto militar, por exemplo, um ataque envolveria modificar os softwares destes sensores de tal forma que estes enviassem informações falsificadas quanto ao movimento das tropas, radioatividade, substâncias químicas, etc... para a central de comando.

Antes de proceder com a descrição de técnicas de ofuscação, descrevemos na Figura 3.1 as etapas de engenharia reversa e compilação. Compilação é o processo de tradução de um programa em uma linguagem fonte (alto nível) para uma linguagem de máquina (baixo nível). Este processo é realizado através de etapas sucessivas que convertem o código a cada etapa para um nível inferior até atingir o nível de linguagem de máquina (*assembly*). Engenharia reversa é o processo oposto, ou seja, visa reconstruir a estrutura semântica de uma linguagem de baixo nível. Em geral, este processo é dividido em duas partes: *disassembly*, que converte uma sequência de bits em uma linguagem *assembly*; e decompilação, que reconstrói a estrutura semântica de um código *assembly* [35].

Técnicas de ofuscação pode dificultar as duas fases da engenharia reversa: a de *disassembly* para que esta retorne uma linguagem *assembly* incorreta [36]; ou a de de-

compilação fazendo com que as ferramentas de análise estática retornem grandes aproximações, dificultando-se assim, a recuperação da semântica do código. O processo de decompilação pode ser dificultado pela inserção de expressões booleanas complexas, ponteiros, saltos indiretos ou espúrios e *aliasing* [22, 16, 37, 21, 26]. *Aliasing* descreve uma situação em que uma posição de um dado na memória pode ser acessado através de diferentes nomes simbólicos em um programa, em que a modificação de um dado por um nome implicitamente modificaria todos os outros nomes apontados.

As técnicas de otimização aplicadas por compiladores geram um código mais ofuscado. Entretanto, estas buscam o aumento de desempenho da aplicação através da reordenação/substituição de instruções e/ou bloco de instruções. Já as técnicas de ofuscação de código visam aplicar transformações de código para dificultar seu discernimento, mesmo que essas degradem o desempenho e/ou aumentem o tamanho do código final.

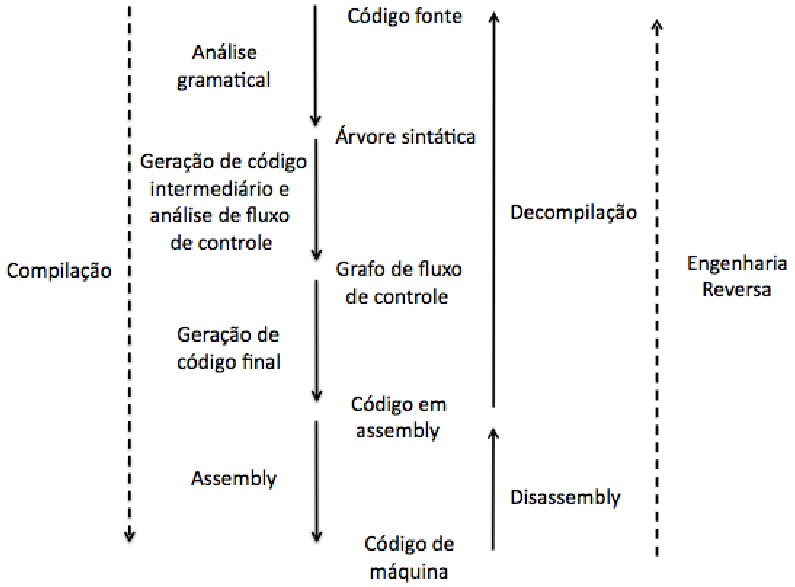


Figura 3.1. Etapas de compilação e engenharia reversa.

A base das ferramentas de análise de programas é o *disassembly* cuja função é a engenharia reversa de um código binário para recuperar o conjunto de instruções *assembly*. Contudo, como o processo de *disassembly* não é ciência exata, ferramentas de análise estática de programas que dependem de sua saída podem gerar resultados incorretos. Estas ferramentas são responsáveis pela abstração dos procedimentos, geração do grafo de fluxo de controle, análise do fluxo de dados e decompilação. Os *disassemblers* podem ser genericamente divididos em dois tipos: varredura linear e transversal recursivo [36].

Os *disassemblers* de varredura linear começam pelo processamento do segmento de texto (.text) da imagem do código binário obtido no cabeçalho do arquivo. O processo de *disassembly* é realizado sequencialmente até atingir o final do segmento de texto. Contudo, diante de dados inseridos no segmento de texto o processo de *disassembly* de varredura linear pode ser comprometido. O código da Figura 3.2 ilustra a situação em que o dado ‘db 0E8h’ seria interpretado como se fosse código. Caso os bytes coincidissem

```
Main:
      ...
      jmp     Func
      db      0E8h
Func:
      ...
```

Figura 3.2. Trecho de código contendo dado no segmento de texto de um binário.

com alguma codificação de alguma instrução *assembly*, o processo de *disassembly* converteria os bytes erroneamente. Em situações de não coincidência, estes *disassemblers* notificam através de pontos de interrogação os bytes não convertidos. Uma propriedade interessante destes *disassemblers* é que eventualmente ocorre a auto sincronização após um determinado número de instruções mal convertidas [36].

Um *disassembler* transversal recursivo visa a superar esta fraqueza fazendo com que o processo de *disassembly* acompanhe do fluxo de controle do programa. Caso houvesse dados no meio do fluxo de instruções (como o exemplo acima), o *disassembler* acompanharia o salto, e consequentemente, não interpretaria erroneamente o dado 'db 0E8h'. Contudo, o processo de *disassembly* é mais complexo, pois este tem que deduzir os possíveis destinos estaticamente, que nem sempre é uma tarefa trivial. Saltos indiretos e *aliasing* são desafios para que esta dedução seja feita corretamente.

Formalmente, dado um código C_σ com um "segredo" σ nele embarcardo, um ofuscador transforma este em um código C'_σ cuja extração do segredo σ é mais difícil de ser realizada em C'_σ do que C_σ. Para esta etapa de transformação temos uma biblioteca de transformações de código que preservam a semântica τ e um conjunto de ferramentas de análise de programas A que servem de auxílio para as transformações provendo informações estáticas e/ou dinâmicas sobre o programa C_σ. Informações dinâmicas são coletadas através da execução do programa diante de um determinado conjunto de entrada I. Figura 3.3 ilustra um processo de ofuscação, em que K é a parametrização de quais ofuscações foram utilizadas, assim como os pontos do código em que foram aplicadas.

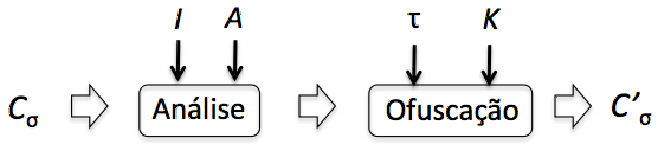


Figura 3.3. Processo de ofuscação.

As técnicas de ofuscação podem ser classificadas em estáticas ou dinâmicas. As estáticas podem ser: transformações de (*layout*) que removem informações de formatação de arquivos ou substituem identificadores; transformações de controle que distorcem o fluxo de controle pela inserção de código espúrio, reordenação do fluxo de execução das instruções através de saltos condicionais/incondicionais ou eliminação das estruturas de controle (laços de repetição e instruções condicionais); e transformações de dados que substituem as instruções ou os dados por novas representações. As dinâmicas transformam o código do programa em tempo de execução e por conseguinte, são mais resilientes

diante de análise puramente estática.

As transformações podem ser feitas tanto no código de alto-nível — que posteriormente é compilado de maneira a gerar um código binário ou um código intermediário mais complexo — como diretamente no código binário. Um código binário possui mais informação que pode ser ofuscada do que um código de alto-nível, porém o processo de modificar um binário é muito mais complexo. Outra vantagem é que a manipulação no binário não precisa se preocupar com o processo de otimização dos compiladores uma vez que a ofuscação é realizada diretamente no código final. A seguir, descreveremos técnicas aplicáveis no código de alto-nível e outras aplicáveis no código de baixo-nível.

3.2.1. Ofuscação estática

3.2.1.1. Transformações de *layout*

Técnicas de transformação de estrutura são as mais simples. Um exemplo é a remoção de informações de formatação de arquivos normalmente presentes em arquivos '.class' da linguagem Java, como é o caso do ofuscador Crema [10]. Outra técnica que se enquadra nas transformações de estrutura consiste na renomeação dos identificadores, o que dificulta a análise do código fonte. A renomeação sendo feita diretamente no código binário, no nível de registradores, é utilizada para criar cópias distintas com o âmbito de dificultar um ataque de diferença ou de assinatura. Esta técnica foi utilizada pelo vírus Win32/Regswap que, a cada replicação, faz a troca dos registradores utilizados. Figura 3.4 ilustra trechos de duas gerações deste vírus, em que a geração mais abaixo substitui o registrador 'edx' por 'eax', 'edi' por 'ebx', 'esi' por 'edx', 'eax' por 'edi' e'ebx' por 'esi'.

```
5A              pop  edx
BF04000000      mov  edi,0004h
8BF5            mov  esi,ebp
B80C000000      mov  eax,000Ch
81C288000000    add  edx,0088h
8B1A            mov  ebx,[edx]
899C8618110000  mov  [esi+eax*4+00001118],ebx

58              pop  eax
BB04000000      mov  ebx,0004h
8BD5            mov  edx,ebp
BF0C000000      mov  edi,000Ch
81C088000000    add  eax,0088h
8B30            mov  esi,[eax]
89B4BA18110000  mov  [edx+edi*4+00001118],esi
```

Figura 3.4. Atribuição de diferentes registradores em duas gerações do Win32/Regswap [25].

3.2.1.2. Transformações de controle

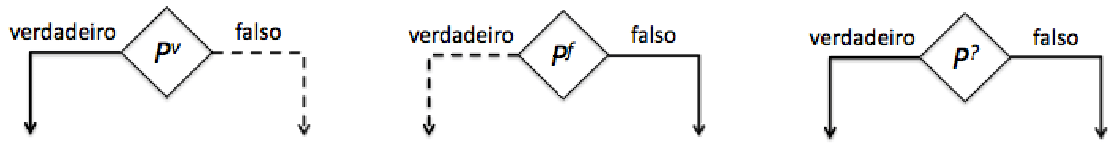
Transformações de controle visam a dificultar a análise do fluxo de controle. A análise do fluxo de controle é dependente do grafo de fluxo de controle que representa o fluxo de execução de todas possíveis execuções de um programa. Técnicas para ofuscar o fluxo de

controle consistem na inserção, reordenação ou eliminação de blocos básicos no fluxo de execução do programa.

A inserção de códigos espúrios no fluxo de controle degrada a precisão de uma análise de fluxo de controle. Esta técnica consiste na adição de saltos condicionais ou incondicionais para que estes códigos sejam incorporados no fluxo de controle do programa. Os saltos condicionais são normalmente vinculados com *predicados opacos*: expressões cujos valores são de conhecimento do programador ou ofuscador, porém de difícil discernimento para um analisador estático ou um analista [14]. Quanto mais difícil determinar o resultado de um predicado opaco, mais robusta será a técnica de ofuscação, pois uma ferramenta de engenharia reversa ou um analista que não consiga determinar estaticamente o valor de um predicado, tem que considerar ambos caminhos do fluxo de controle como possíveis, degradando, portanto, as informações coletadas.

Em outras palavras, predicados opacos provêem um modo para atacar a fase de geração do grafo de fluxo de controle de uma analisador estático. A utilização destas instruções de salto ofuscadas força um analisador adicionar arestas desnecessárias no grafo de fluxo de controle, fazendo com que os resultados sejam menos precisos.

Os predicados opacos podem ser classificados em verdadeiros, falsos ou indeterminados e são denotados respectivamente por P^v, P^f e $P^?$. Figura 3.5 ilustra como estes tipos de predicados opacos são representados graficamente. Para a construção de predicados opacos uma das opções triviais envolve a adição de tautologias para predicados verdadeiros e contradições para predicados falsos.

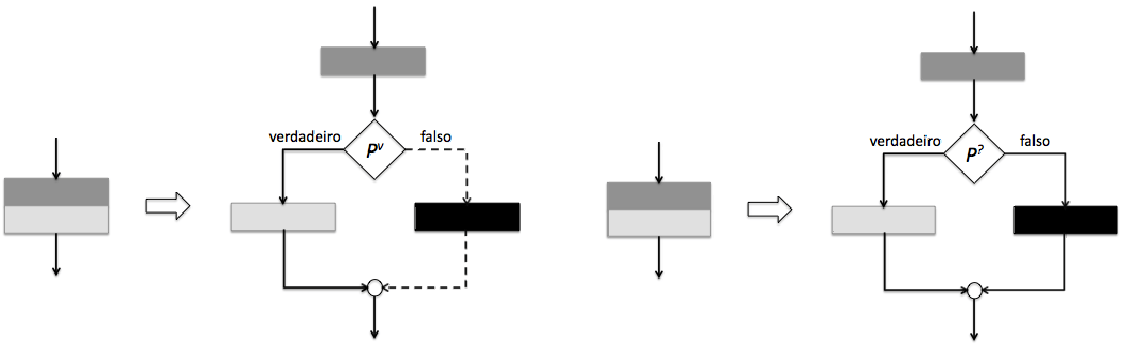


(a) Verdadeiro. (b) Falso. (c) Indeterminado.

Figura 3.5. Tipos de predicados opacos.

Estratégias para inserção de blocos espúrios no fluxo de controle envolvem a utilização de trechos de códigos que podem ser ou não executados [14, 16, 32]. Caso sejam executados, a semântica dos códigos espúrios deve ser equivalente à semântica dos blocos correspondentes. Blocos correspondentes são blocos que não foram executados, pois o fluxo de execução trilhou o caminho dos blocos espúrios. Figura 3.6(a) representa a situação em que o trecho de código em preto nunca será executado, pois o predicado sempre retornará um valor verdadeiro. Figura 3.6(b) representa a situação em que o trecho de código em preto pode ou não ser executado, pois o predicado é indeterminado, contudo a semântica do código preto, neste caso, é equivalente ao código em cinza claro.

A ofuscação por inserção de blocos espúrios no fluxo de controle visa a degradar a precisão de uma análise de fluxo de dados. Uma possível variação, exibida no código da Figura 3.7, seria a inserção de um dado ‘db 0E8h’ através de um predicado opaco ‘cmp eax, eax’ para degradar a etapa de *disassembly*. Nesta situação, *disassembler* poderia considerar a instrução da linha ‘4’ como se fosse código, pois a instrução ‘3’ é um salto incondicional para ela, porém é uma instrução que nunca vai ser executada, pois o salto



(a) Código em preto não executável.

(b) Código em preto executável com semântica equivalente ao código em cinza claro.

Figura 3.6. Inserção de informação espúria no fluxo de controle com predicados opacos.

da instrução ‘2’ sempre será executado dado que os dois operandos sendo comparados (‘eax’ e ‘eax’) na instrução ‘1’ são sempre iguais (predicado opaco).

```
Main:
1     cmp     eax,  eax
2     je      Func+1
3     jmp     Func
Func:
4     db       0E8h
5     push     0
6     call     ExitProcess
```

Figura 3.7. Ofuscação por inserção de dado espúrio por predicado opaco.

Figura 3.8 exibe o código gerado pelo depurador OllyDbg para a sequência de instruções da Figura 3.7. O depurador assumiu que ambos destinos dos saltos são válidos. Observe que o código gerado é muito diferente do código executável original. O depurador interpretou incorretamente o dado posicionado no começo de ‘Func’ como se fosse código. Como o dado ‘E8h’ coincide com o código de operação de uma instrução de chamada ‘call’, o depurador assumiu como se fosse uma instrução de chamada válida e prosseguiu o processo de conversão para instruções subsequentes de forma incorreta.

```
00401000  $ 3BC0          CMP EAX,EAX
00401002  .v74 03         JE SHORT sampleco.00401007
00401004  .vEB 00         JMP SHORT sampleco.00401006
00401006  > E8 6A00E800   CALL 01281075
0040100B  ? 0000          ADD BYTE PTR DS:[EAX],AL
0040100D  ? 00FF          ADD BH,BH
0040100F  ? 25 30304000   AND EAX,403030
```

Figura 3.8. Saída do depurador OllyDbg.

Outra técnica de transformação de controle consiste em achatar o fluxo de execução com o âmbito de dificultar a análise do fluxo de controle [13, 14, 26, 49]. O achatamento pode ser feito colocando cada bloco básico do programa dentro de um ‘*case*’

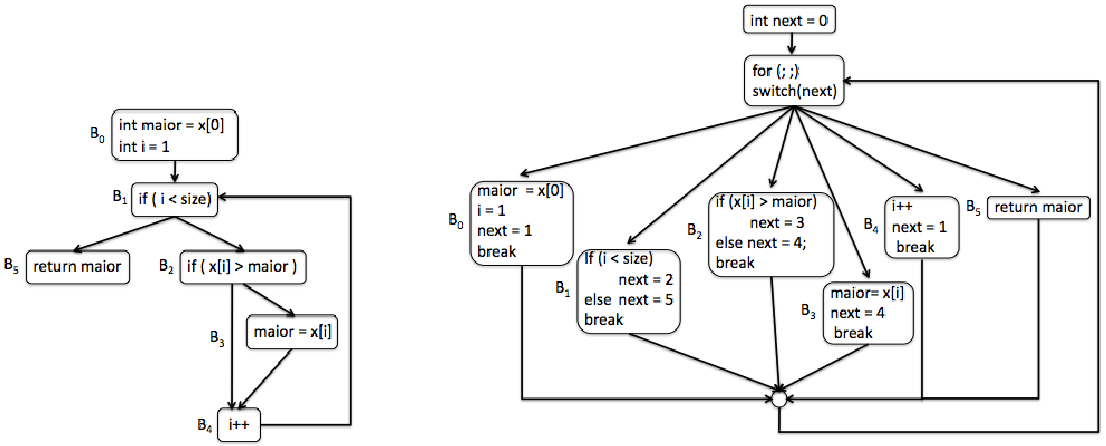
```
/* Código Original */

int maior(int x[], int size) {
     int maior = x[0];
     int i = 1;
     while ( i < size) {
          if ( x[i] > maior )
               maior = x[i];
          i++;
     }
     return maior;
}

/* Código Ofuscado */

int maior(int x[], int size) {
    int next = 0;
    for (;;){
         switch (next) {
              case 0: int maior = x[0]; int i =1; next = 1; break;
              case 1: if  (i < size) next = 2 else next = 5; break;
              case 2: if  (x[i] > maior) next = 3; else next = 4; break;
              case 3: maior = x[i]; next = 4; break;
              case 4: i++; next = 1 ; break;
              case 5: return maior;
         }
     }
}
```

Figura 3.9. Ofuscação por achatamento do fluxo de execução.

de uma estrutura '*switch*', e esta estrutura dentro de um laço infinito. Os códigos da Figura 3.9 exibem esta transformação de ofuscação por achatamento do fluxo de execução. Figura 3.10 ilustra os grafos de fluxo de controle para o código não ofuscado e ofuscado.

O código exemplificado é simples, contudo, oferece subsídios para a compreensão da importância das técnicas de transformação no aumento da dificuldade de análise por um adversário. Pode-se aumentar o grau de dificuldade com o uso de *aliasing* de ponteiros na variável 'next' [22, 16, 37, 21, 26], fazendo com que a análise estática retorne resultados pouco precisos quanto a variável 'next', dificultando-se portanto um analista de acompanhar o fluxo de execução do programa. Apesar de um grande número de trabalhos em algoritmos para análise de *aliasing*, nenhum deles mostrou ser escalável para programas com milhões de linhas de código, concentrando-se a maioria na ordem de cem mil linhas de código [55].

Outra técnica de transformação de controle envolve a reordenação de código cuja idéia é alterar a ordem sintática das instruções de um programa enquanto a ordem de execução do programa é preservada através da inserção de instruções de salto. O código da Figura 3.11 exibe a rotina original da Figura 3.9 ofuscada pela técnica de reordenação de código. Observe que os endereços dos saltos também foram ofuscados por um vetor de ponteiros, o que dificulta ainda mais a análise do código. Vale ressaltar que não é



(a) Código não ofuscado. (b) Código ofuscado por achatamento.

Figura 3.10. Grafo do fluxo de controle da rotina “maior”.

necessário o uso de saltos incondicionais para reordenação de código: uma estratégia mais robusta contra engenharia reversa seria o uso de instruções condicionais com predicados opacos.

```
int maior(int x[], int size) {
     int i, maior;
     char* jtab[ ]= {&&l0, &&l1, &&l3};
     goto *jtab[0];
l2:
     while ( i < size) {
          if ( x[i] > maior )
               maior = x[i];
     i++;
}
     goto *jtab[3];
l0:
     maior = x[0];
     goto *jtab[1];
l1:
     i = 1;
     goto l2;
l3:
     return maior;
}
```

Figura 3.11. Ofuscação por reordenação de código.

3.2.1.3. Transformações de dados

Transformações de dados envolvem a conversão de uma representação de uma estrutura de dados, tipo de dados ou instruções em outra representação que seja mais difícil para um atacante de entendê-la. Funções de criptografar/decriptografar em criptografia são meios

```
CODE:00401000 start           proc near
CODE:00401000                 push    1
CODE:00401002                 push    2
CODE:00401004                 call    sub_401010
CODE:00401009                 push    0
CODE:0040100B                 call    ExitProcess

CODE:00401010 sub_401010      proc near
CODE:00401010                 mov     eax, [esp+arg_0]
CODE:00401014                 mov     ebx, [esp+arg_4]
CODE:00401018                 add     eax, ebx
CODE:0040101A                 retn    8
```

(a) Chamada de procedimento não ofuscada.

```
CODE:00401000 start           proc near
CODE:00401000
CODE:00401000 var_8           = dword ptr -8
CODE:00401000 var_4           = dword ptr -4
CODE:00401000
CODE:00401000                 push    1
CODE:00401002                 push    2
CODE:00401004                 push    offset loc_40100F
CODE:00401009                 push    offset loc_401016
CODE:0040100E                 retn
CODE:0040100F loc_40100F:
CODE:0040100F                 push    0
CODE:00401011                 call    ExitProcess
CODE:00401016 loc_401016:
CODE:00401016                 mov     eax, [esp+4]
CODE:0040101A                 mov     ebx, [esp+10h+var_8]
CODE:0040101E                 add     eax, ebx
CODE:00401020                 retn    8
```

(b) Chamada de procedimento ofuscada.

Figura 3.12. Ofuscação de uma instrução de chamada ‘*call*’.

para transformação de dados. Contudo, para que uma operação seja realizada neste tipo de ofuscação, os dados devem ser primeiramente desofuscados (decriptografados), o que os torna acessíveis para um adversário de engenharia reversa.

Uma técnica de transformação de dados envolve ofuscar convenções relacionadas ao encapsulamento de funções de uma linguagem *assembly*. Este encapsulamento é feito através de convencões relacionadas com instruções de chamada ‘*call*’ e de retorno ‘*ret*’. Na maioria dos códigos gerados por compiladores, uma instrução de chamada ‘*call*’ é emparelhada com uma instrução de retorno ‘*ret*’, e a instrução de retorno exerce a função de transferir o controle para a instrução subsequente a instrução de chamada. Porém, esta convenção não precisa ser seguida, fazendo com que a fase de geração de fluxo de controle na etapa de decompilação retorne resultados imprecisos. A etapa de *disassembly* também pode ser prejudicada caso o *disassembler* seja transversal recursivo, pois o mesmo utiliza do comportamento do fluxo de controle para converter as instruções.

Instruções de chamada e retorno não são atômicas, assim as mesmas podem ser ofuscadas substituindo-as por instruções que exerçam semântica equivalente. Figura 3.12 ilustra esta situação, na coluna a esquerda uma função invoca a outra através de uma instrução normal de chamada de função; na coluna a direita temos um código com semântica equivalente que utiliza de uma sequência de duas instruções ‘*push*’ e uma instrução ‘*ret*’. A primeira instrução ‘*push*’ coloca o endereço da instrução após a instrução de chamada, a segunda instrução ‘*push*’ coloca o alvo da instrução de chamada e a instrução ‘*ret*’ transfere a execução para o endereço alvo da chamada.

A semântica das instruções ‘*call*’ e ‘*ret*’ clássicas pode ser divididas em duas partes: manipulação do endereço de retorno da pilha e transferência do controle do programa. Para ofuscar uma chamada de procedimento (ou um retorno de um procedimento) estas duas partes da semântica das instruções tem que ser separadas e efetuadas usando um outro conjunto de instruções. Este conjunto de instruções não precisa estar contíguo dentro do código do programa, podendo estar distribuído e/ou misturado com outras instruções. Instruções que modificam o ponteiro da pilha, tais como atribuição, incremento e decremento também poderiam ser utilizados para ofuscação. Por outro lado, instruções ‘*call*’ (‘*ret*’) podem ser empregadas com outras finalidades do que fazer (retornar de) uma chamada de procedimento, como identificadas por Lakhotia *et al.* [45]:

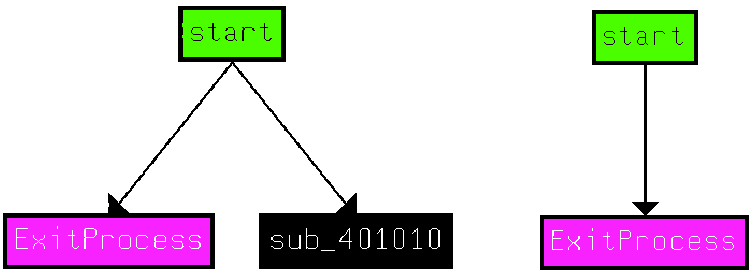
1. Uma chamada ‘*call*’ simulada. A semântica de uma instrução ‘*call end*’ é a seguinte: o endereço da instrução seguinte da instrução de chamada é colocada na pilha e o fluxo de controle é transferido para o endereço ‘*end*’. Esta semântica pode ser simulada através de uma combinação de duas instruções ‘*push*’ e uma instrução ‘*ret*’ como ilustrado na Figura 3.12. O código desta Figura exibe duas rotinas, uma que coloca dois valores na pilha (no caso ‘1’ e ‘2’) e faz a chamada para uma função ‘sub_401010’ que adiciona estes dois argumentos e retorna para a função principal. O mesmo efeito poderia ser adquirido através de uma instrução ‘*push*’ que colocaria o endereço da instrução seguinte da instrução de chamada e uma instrução ‘*jmp*’ que transferiria o fluxo de execução para o alvo da instrução de chamada. Esta situação está ilustrada na Figura 3.14(a).

2. Uma chamada ‘*call*’ para transferir fluxo de controle. Aqui, a instrução de chamada seria utilizada para transferência de controle em que o endereço de retorno seria simplesmente descartado através da remoção do valor do topo da pilha.

3. Um retorno ‘*ret*’ simulado. Uma instrução ‘*ret*’ é complementar a uma instrução ‘*call*’. Esta retira o endereço de retorno (normalmente colocada por uma instrução ‘*call*’) do topo da pilha e transfere o fluxo de controle para este endereço. Esta semântica pode ser adquirida através de uma instrução ‘*pop reg*’ que retira o valor do topo da pilha colocando-o no registrador ‘*reg*’ e uma instrução ‘*jmp reg*’ que transfere o fluxo de controle para o endereço contido no registrador.

4. Um retorno ‘*ret*’ para transferir fluxo de controle. Aqui, a instrução não retorna para um local de chamada. Uma instrução ‘*ret*’ pode ser utilizada para transferir fluxo de controle para outra instrução sem que esta esteja relacionada com uma instrução de chamada. Por exemplo, uma instrução ‘*ret*’ pode ser usada para simular uma instrução ‘*call*’, como descrito anteriormente.

Estas ofuscações distorcem parâmetros importantes que são utilizados para análise automática e manual. Ainda que um programador experiente possa descobrir as ofuscações, o tempo que o mesmo leva pode ser muito precioso quando se trata de um código malicioso na Internet. A substituição de instruções de chamada, em particular, faz com que uma grande parte dos métodos automáticos para detecção de códigos maliciosos falhem uma vez que que esses dependem do reconhecimento de instruções de chamadas para identificar funções *kernel* utilizadas pelo programa. Por exemplo, o *disassembler*



(a) Código não ofuscado. (b) Código ofuscado.

Figura 3.13. Grafo de chamadas para os códigos da Figura 3.12.

IDA Pro [61], vastamente utilizado pela indústria de antivírus, retorna resultados incorretos na geração do grafo de chamadas na presença desta técnica de ofuscação [45]. Figura 3.13 ilustra os grafos de chamadas gerados para o código não ofuscado à esquerda e o código ofuscado à direita. Nota-se que diante de ofuscação o *disassembler* não foi capaz de delimitar o procedimento 'sub_401016'.

Estas ofuscações de chamada e retorno de procedimento atingem duas fases importantes da análise de controle de fluxo interprocedural: identificação dos procedimentos e criação do grafo de chamadas. A maioria das linguagens *assembly* não fornecem mecanismos para encapsular procedimentos. Assim, *disassemblers* utilizam das instruções *call* e *ret* para determinar os limitantes das funções e para criar o grafo de chamadas [61]. Quando estas instruções são ofuscadas, as funções identificadas e o grafo de chamadas gerados podem ser questionáveis e qualquer análise subsequente duvidosa. Um método para análise sensível ao contexto de binários com instruções de chamada e retorno ofuscadas foi recentemente proposto por Boccardo [62]. O método é baseado na noção de contextos baseado na pilha que conseguem rastrear mudanças de contexto na pilha, e por conseguinte, gerar resultados mais precisos do que os métodos clássicos [3].

Outra transformação de dados em código *assembly* envolve instruções de salto incondicionais '*jmp*' [36]. A idéia consiste em substituir um salto incondicional por uma chamada de função e manipular o endereço de retorno da função para ser o endereço de destino do salto incondicional. Figura 3.14 exibe um ofuscamento de um salto incondicional. Figura 3.14(a) mostra um ofuscamento de chamada através do uso de uma instrução '*401004: push*' e uma instrução '*401009: jmp*'. Figura 3.14(b) ilustra o ofuscamento da instrução '*401009: jmp*' substituindo-a pela chamada de função '*40100B: call sub_-401024*'. Instrução '*401024: xchg*' troca o conteúdo do registrador '*eax*' com o valor de retorno contido no topo da pilha '*esp+0*'. Instrução '*401027: add*' soma este valor com o deslocamento para o alvo do salto que foi colocado na pilha antes da chamada da função ('*401009: push*'). Instrução '*40102B: pop*' restaura o valor do registrador '*eax*' e instrução '*40102C: ret*' faz com que a execução seja transferida para o alvo do salto incondicional ('*401017: mov eax, [esp+4]*') .

As transformações de dados não são restritas as linguagens *assembly* como as técnicas de ofuscação de chamada e retorno de procedimento e de saltos incondicionais. Transformações podem ser feitas em uma linguagem de alto-nível para alterar o tipo de

```
CODE:00401000 start           proc near
CODE:00401000 var_8           = dword ptr -8
CODE:00401000 var_4           = dword ptr -4
CODE:00401000                 push    4
CODE:00401002                 push    2
CODE:00401004                 push    offset loc_40100B
CODE:00401009                 jmp     short loc_401012
CODE:0040100B loc_40100B:
CODE:0040100B                 push    0
CODE:0040100D                 call    ExitProcess
CODE:00401012 loc_401012:
CODE:00401012                 mov     eax, [esp+0Ch+var_8]
CODE:00401016                 mov     ebx, [esp+0Ch+var_4]
CODE:0040101A                 add     eax, ebx
CODE:0040101C                 retn    8
```

(a) Salto incondicional não ofuscado.

```
CODE:00401000 start           proc near
CODE:00401000                 push    4
CODE:00401002                 push    2
CODE:00401004                 push    offset loc_401010
CODE:00401009                 push    7
CODE:0040100B                 call    sub_401024
CODE:00401010 loc_401010:
CODE:00401010                 push    0
CODE:00401012                 call    ExitProcess
CODE:00401012 start           endp
CODE:00401017                 mov     eax, [esp+4]
CODE:0040101B                 mov     ebx, [esp+8]
CODE:0040101F                 add     eax, ebx
CODE:00401021                 retn    8
CODE:00401024 sub_401024      proc near
CODE:00401024                 xchg    eax, [esp+0]
CODE:00401027                 add     [esp+arg_0], eax
CODE:0040102B                 pop     eax
CODE:0040102C                 retn
```

(b) Salto condicional ofuscado.

Figura 3.14. Ofuscação de um salto incondicional.

dados como inteiros, booleanos, *strings* ou estrutura de dados como vetores para que o código gerado seja mais complicado de ser entendido [17]. O tipo inteiro pode ser transformado alterando-se seu intervalo de valoração. Por exemplo, um inteiro 'i' pode ser representado por '$i' = c_1.i + c_2$', na qual 'c_1' e 'c_2' são constantes inteiras. Figura 3.15 ilustra este tipo de transformação utilizando $c_1 = 8$ e $c_2 = 3$. O único cuidado necessário ao utilizar esta técnica refere-se ao *overflow* uma vez que o intervalo de valores para o programa transformado é menor do que o intervalo de valores do programa original. O tipo booleano pode ser transformado utilizando-se valores múltiplos para representar os valores 'verdadeiro' e 'falso'.

Transformar *strings* em uma forma menos visível para um adversário é muito importante dado que *strings* tipicamente fornecem informações relevantes quanto o comportamento de um programa para um analista. Uma forma de mudar a representação de uma *string* consiste na implementação de uma rotina que produza a *string* desejada. Digamos que queremos ofuscar a string 'ofusca'. Figura 3.16 ilustra uma rotina para gerar esta *string*, passando a entrada de valor '1' para a função 'gera'.

Técnicas de transformação de vetores são baseadas no particionamento, fusão ou redimensionamento do número de dimensões. Figura 3.17 exibe uma transformação por particionamento de um vetor em dois sub-vetores. Figura 3.18 exibe uma transformação por fusão de dois vetores em um único vetor. Figura 3.19 exibe a transformação de

```
/* Código Original */

int i = 1;
while (i < 1000) {
   ... A[i] ...;
   i++;
}

/* Código Ofuscado */

int i = 11;
while (i < 8003) {
   ... A[(i-3)/8] ...;
   i+=8;
}
```

Figura 3.15. Transformação de inteiros.

```
String gera (int n) {
     String S;
     l1: if (n==1) {S[i++] = "o"; goto l2; }
     l2: if (n==2) {S[i++] = "f"; goto l3;}
     l3: if (n==3) {S[i++] = "u"; goto l4;}
     l4: if (n==4) {S[i++] = "s"; goto l5;}
     l5: if (n==5) {S[i++] = "c"; goto l6;}
     l6: if (n==6) {S[i++] = "a"; return S;}
}
```

Figura 3.16. Transformação de *strings*.

aumento do número de dimensões de um vetor. Figura 3.20 exibe a transformação de diminuição do número de dimensões de um vetor.

3.2.2. Ofuscação dinâmica

As técnicas de ofuscação de código até então abordadas são estáticas, na qual o código é transformado antes de sua execução. Apesar de estas aumentarem o grau de dificuldade de um adversário em descobrir alguma propriedade do código, as mesmas não são tão robustas diante de ferramentas de análise dinâmica. Técnicas de ofuscação dinâmica transformam o código do programa em tempo de execução e por conseguinte, tornam a tarefa de desvendar um segredo mais complicada para ambos tipos de análise: estática ou dinâmica. Em contrapartida, estas degradam desempenho, pois uma vez que o segmento de código é alterado, o *pipeline*[1] de instruções tem que ser esvaziado, o conteúdo do cache de dados tem que ser escrito na memória e o cache de instruções tem que ser invalidado.

Fred Cohen [6] aponta duas estratégias para ofuscação dinâmica. Ambas estratégias consistem de códigos auto modificáveis, porém a primeira utiliza-se de uma parte constante do código criptografada (ou compactada) que é decriptografada (ou descom-

[1]Técnica de hardware que possibilita a CPU buscar uma ou mais instruções além da próxima a ser executada.

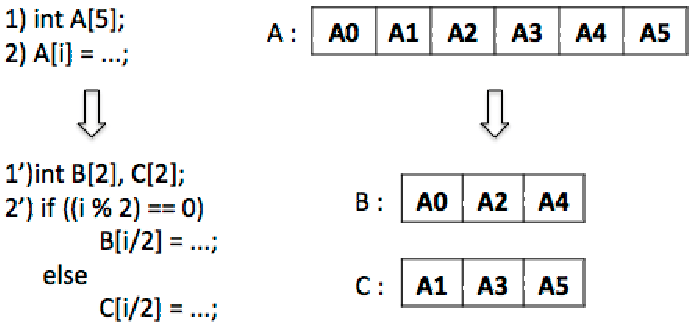


Figura 3.17. Transformação por particionamento de um vetor.

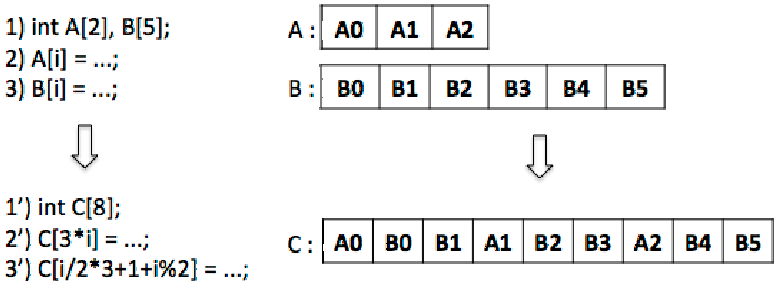


Figura 3.18. Transformação por fusão de um vetor.

pactada) por uma rotina embarcada no código que é executada durante a execução do programa; a segunda visa a transformar o fluxo de execução do programa constantemente.

A primeira estratégia tipicamente está atrelada a aplicação de criptografia (ou compactação) para esconder um trecho de código. Esta técnica de proteção, apesar de ser tradicional, não é considerada resiliente diante de ataques '*Man-At-The-End*, pois em algum momento de execução, o código torna-se visível para um adversário. Um adversário através de um depurador pode estabelecer critérios de parada e ter acesso ao código em texto claro. Outra possibilidade, ainda mais simples, é um adversário usar de emulação para ter acesso a todas instruções que foram executadas pelo programa. O código da Figura 3.21 ilustra esta técnica em que o trecho de código das linhas 3-9 encontram-se criptografados por um ou exclusivo (XOR) com a chave '34'. Durante a execução do programa, o código entre as linhas 'l3' e 'l9' é decriptografado pelo código das linhas 'l1' e 'l2' para posteriormente ser executado.

Kanzaki *et al.* [33] explora a idéia de mudar o fluxo de execução do programa constantemente e não somente durante algum ponto de execução como a técnica discutida anteriormente. Para isso, a técnica alterna uma sequência de instruções originais por uma sequência de instruções falsas e vice versa. O objetivo é deixar instruções originais o menor tempo possível na memória e disfarçá-las com instruções falsas para evitar que um analista descubra propriedades importantes do programa. Antes de executar uma instrução falsa, a instrução verdadeira é colocada no endereço da instrução falsa, e após executar a instrução verdadeira, a instrução falsa é recolocada no endereço da instrução verdadeira. Este procedimento é sucintamente ilustrado na Figura 3.22 em que a instrução falsa 'instf' do endereço '2' é substituída por uma instrução verdadeira 'instv' após exe-

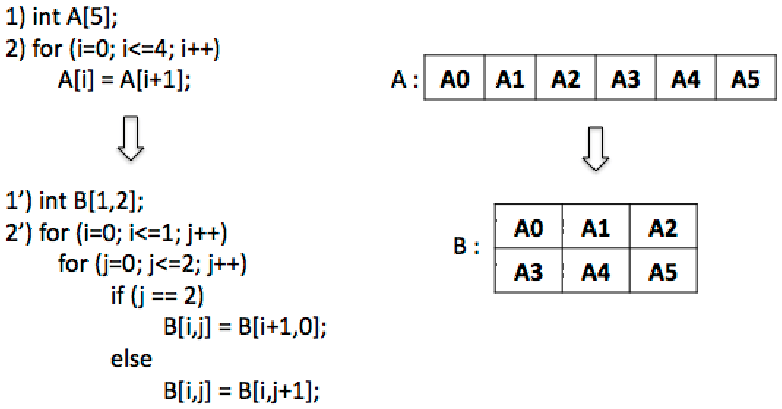


Figura 3.19. Transformação por aumento do número de dimensões de um vetor.

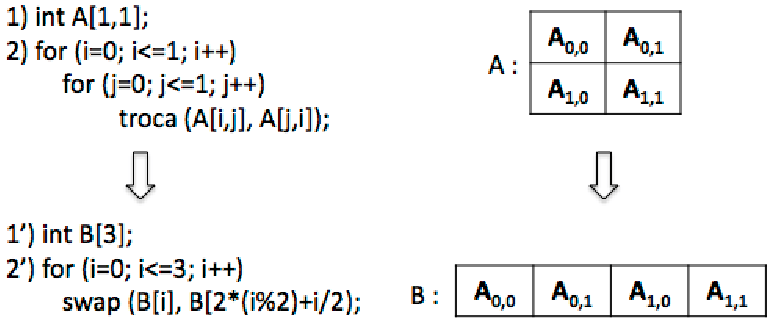


Figura 3.20. Transformação por diminuição do número de dimensões de um vetor.

cução da instrução do endereço '1'. Após a execução de 'instv', a instrução do endereço '3' substitui 'instv' do endereço '2' pela instrução falsa 'instf'. É claro que a técnica de criptografia (ou compactação) também poderia ser utilizada para constantemente alterar o fluxo de execução, ou seja, em um curto espaço de tempo o código estaria decriptografado (ou descompactado) na memória enquanto que na maior parte do tempo o código estaria inteligível.

Madou *et al.* [46] implementou um ofuscador dinâmico para fusão de código. A técnica envolve o compartilhamento de endereços de memória por duas ou mais funções distintas através da criação de um padrão de bytes comum a essas funções e da criação de *scripts* encarregados de substituir o padrão pela função a ser executada. Figura 3.23 ilustra essa técnica na qual é criado um padrão 'p' para duas funções 'f_1' e 'f_2' cujos bytes diferem nas posições '2' e '3', e portanto, são representados por '?' (qualquer *byte*) no padrão 'p'. Os *scripts* 's_1' e 's_2' são encarregados de substituir os bytes correspondentes das funções em tempo de execução.

Vale ressaltar que um adversário pode interceptar estes *scripts* uma vez que movimentações de código em posições constantes (no padrão 'p') despertam a atenção de um adversário, auxiliando-o a desvendar propriedades do programa. Os autores desta técnica propõem o uso de criptografia nos *scripts* para aumentar a resiliência do método.

Aucsmith [9] desenvolveu uma técnica de ofuscação dinâmica que divide o pro-

```
int maior(int x[], int size) {
l1:     char* p=&&comeco;
l2:     while (p < (char *)&&fim)    *p++ ^=34; // xor com a chave 34
comeco:
l3:     int maior = x[0];
l4:     int i = 1;
l5:     while ( i < size) {
l6:          if ( x[i] > maior )
l7:               maior = x[i];
l8:          i++;     }
l9:    return maior;
fim:
}
```

Figura 3.21. Técnica de criptografia utilizada para proteger o código.

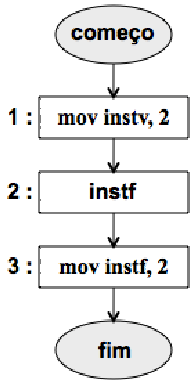


Figura 3.22. Ofuscação dinâmica através da intercalação de instruções falsas com instruções originais.

grama em blocos e permuta-os ciclicamente através de ou exclusivos (XORs). Antes da execução, os blocos são separados em duas regiões de memória: superior e inferior. A execução dos blocos é feita de maneira alternada e de forma que a cada iteração é feito um XOR de cada bloco da memória superior com cada bloco correspondente da memória inferior de modo que sempre exista um bloco em texto claro que vai ser executado na iteração subsequente. Nas iterações pares, partes da memória superior são mescladas com as partes da memória inferior, e nas iterações ímpares partes da memória inferior são mescladas com as partes da memória superior.

Figura 3.24 ilustra a permutação de dois trechos de código 'A' e 'B', lembrando que $A \oplus B = C$ e $C \oplus B = A$. Nota-se que a permutação de dois trechos de código envolve três etapas 'I', 'I_1' e 'I_2' e para que os trechos retornem a configuração inicial 'I_0' necessita-se de seis etapas: de 'I_0' até 'I_5'.

Figura 3.25 exibe como é a proteção de uma função constituída pelos blocos 'A' até 'F' pela técnica de Aucsmith. Observa-se que a função foi dividida em seis blocos e que foi configurado um estado inicial 'I_0' para os blocos do programa de modo que em todas iterações, o próximo bloco a ser executado esteja em texto claro. A sequência de estados configura-se de acordo com a intercalação dos XORs dos blocos da memória

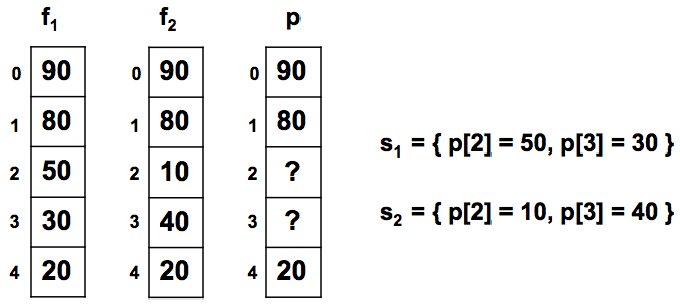


Figura 3.23. Ofuscação dinâmica por fusão de código.

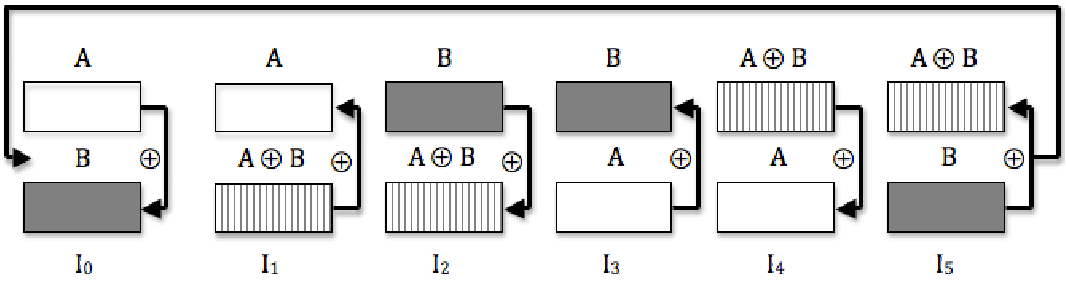


Figura 3.24. Permutação de trechos de código utilizando XOR.

superior com os blocos da memória inferior. O autor também propõe o uso de uma chave criptográfica para aumentar a robustez do método.

3.2.3. Ofuscação maliciosa

Programadores de código malicioso também aplicam técnicas de ofuscação de código e possuem a disposição deles um arsenal de ferramentas de ofuscação, tais como Mistfall [64] da comunidade *BlackHat*, assim como ferramentas comercialmente disponíveis como PECompact [65]. Estas modificações visam evadir detecção pelos varredores de código malicioso, pois uma boa parte da "inteligência" destes varredores é baseada no padrão de bits, chamado de assinatura[39]; e ofuscações alteram o padrão de bits e, consequentemente a assinatura.

Dois conceitos de ofuscação surgiram da comunidade Blackhat: polimorfismo e metamorfismo [57, 25]. Ambos conceitos se enquadram em ofuscação dinâmica. Um código polimórfico é constituído de duas partes: rotinas de criptografia e uma parte constante de código. A natureza polimórfica se dá através de transformações aplicadas no par de rotinas de criptografia, podendo gerar diferentes métodos para criptografar a parte constante do código a cada replicação do código do programa. Figura 3.26(a) exibe a representação de diferentes gerações de um vírus polimórfico e o correspondente código malicioso decriptografado. Observa-se que o corpo decriptografado do código malicioso permanece inalterado durante as gerações.

Um código metamórfico é constituído de uma rotina que transforma todo o código do programa, incluindo a rotina de transformação. Assim, não existem especificadamente rotinas de criptografia e uma parte constante de código como nos códigos polimórficos. Cada replicação de um código metamórfico pode gerar versões totalmente diferentes da

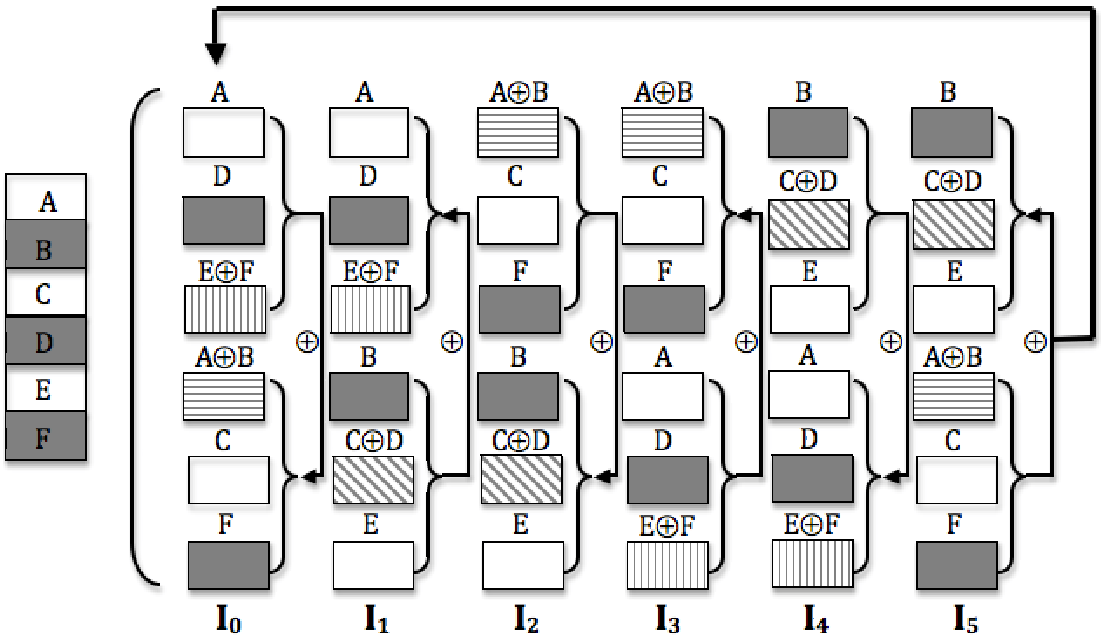


Figura 3.25. Técnica de Aucsmith.

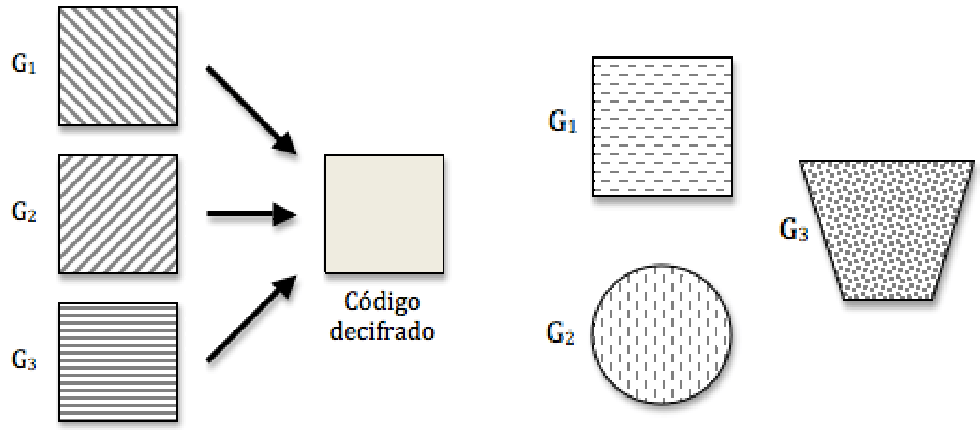
versão anterior, podendo inclusive, adicionar funcionalidades [25]. Figura 3.26(b) exibe a representação de diferentes gerações de um código metamórfico, na qual todo código do programa é transformado. O código malicioso W32/Evol é um exemplo de código metamórfico. Figura 3.27 exibe trechos de código extraídos dos pontos de entrada de três gerações deste código malicioso. Nota-se que da primeira geração para a segunda geração a instrução do endereço ‘401006’ foi substituída pelos códigos dos endereços de ‘1003acf’ até ‘1003ad8’. Em outro exemplo de código de segunda geração a instrução do endereço ‘40100f’ da primeira geração foi substituída pelas instruções dos endereços de ‘1002601’ até ‘100260a’.

3.3. Tamper-proofing

As técnicas de *tamper-proofing* visam a assegurar que um determinado programa executa como esperado, mesmo em situações de modificação ou monitoração. Estas técnicas, além de estarem atreladas à detecção violações de integridade, também possuem funcionalidades de resposta em situações de modificação ou monitoração.

Estas técnicas estão correlacionadas com as técnicas de ofuscação de código dado que para que um código seja modificado, o mesmo precisa ser entendido. As técnicas de ofuscação realizam transformações no código para que a engenharia reversa do mesmo seja mais complicada quanto ao tempo, custo ou poder computacional. Assim, pode-se dizer que a aplicação de ofuscação é uma etapa que garante maior resiliência ao método de *tamper-proofing*. O exemplo a seguir demonstra a importância da utilização de técnicas de ofuscação para dificultar violações a integridade em um programa.

Uma prática comum utilizada por desenvolvedores de software para atrair clientes consiste na distribuição de programas de avaliação com restrições na funcionalidade e/ou no tempo de uso. Programas comerciais frequentemente incluem uma verificação de



(a) Código polimórfico. (b) Código metamórfico.

Figura 3.26. Representação gráfica de diferentes gerações de códigos polimórficos e metamórficos.

tempo de uso, como o trecho de código da Figura 3.28.

Este código permite o usuário executar o programa até quinze vezes, após esta quantidade o programa terminaria sua execução assim que fosse verificada a condição exibida no código. Enquanto estas restrições resguardam o direito do desenvolvedor ao seu software, as mesmas motivam um adversário a modificar o software para remover as restrições. No exemplo acima, um adversário através de engenharia reversa poderia localizar o trecho de código representado e modificá-lo para que o mesmo possa ser executado a quantidade de vezes que ele achasse conveniente ou simplesmente nulificar a condição para que o programa executasse por tempo indeterminado.

As técnicas de *tamperproofing* também estão associadas as técnicas de marca d'água. Para elucidar esta questão, assuma que exista um sistema que um adversário nunca consiga modificar, assim, as técnicas de marca d'água poderiam ser as mais simples possíveis visto que um adversário não conseguiria remover/adulterar a marca d'água. Como este sistema não existe, temos que considerar técnicas para tornar as marcas d'água menos suscetíveis a remoção ou modificação.

Além de as técnicas de *tamper-proofing* estarem atreladas a modificação, qualquer ato de monitoração para extração de código ou dados de um programa também é considerado como uma forma de modificação. Um cenário comum para aplicação das técnicas *tamper-proofing* envolve sistemas de gerenciamento de direitos digitais (DRM) que possuem tipicamente dispositivos reprodutores com uma chave criptográfica embarcada e mídias criptografadas por esta chave criptográfica. Um adversário com acesso ao código do dispositivo reprodutor poderia modificá-lo para reproduzir mídias de outras regiões, assim como ter acesso a mídia em texto claro.

A modificação do código — por inserção e/ou por remoção — pode ser feita *offline*, ou seja, antes de executar o programa ou *online* (em tempo de execução) com a utilização de ferramentas externas como depuradores ou emuladores. Assumindo que o sistema possa ser violável fisicamente, três funções são atribuídas às técnicas de *tamper-proofing*: detecção de violações a integridade do código ou dos dados, verificação do

```
;primeira geração
  401000:       55                      push   ebp
  401001:       8b ec                   mov    ebp,esp
  401003:       83 ec 04                sub    esp,4
  401006:       8b 45 04                mov    eax,DWORD PTR [ebp+4]
  401009:       89 45 08                mov    DWORD PTR [ebp+8],eax
  40100c:       8b 45 00                mov    eax,DWORD PTR [ebp]
  40100f:       89 45 fc                mov    DWORD PTR [ebp-4],eax
  401012:       e8 aa 01 00 00          call   0x4011c1
;segunda geração
  1003ac9:       55                     push   ebp
  1003aca:       8b ec                  mov    ebp,esp
  1003acc:       83 ec 04               sub    esp,4
  1003acf:       56                     push   esi
  1003ad0:       89 ee                  mov    esi,ebp
  1003ad2:       83 c6 53               add    esi,83
  1003ad5:       8b 46 b1               mov    eax,DWORD PTR [esi-79]
  1003ad8:       5e                     pop    esi
  1003ad9:       89 45 08               mov    DWORD PTR [ebp+8],eax
  1003adc:       8b 45 00               mov    eax,DWORD PTR [ebp]
  1003adf:       89 45 fc               mov    DWORD PTR [ebp-4],eax
  1003ae2:       e8 73 02 00 00         call   0x1003d5a
;outra segunda geração
  10025f2:       55                     push   ebp
  10025f3:       8b ec                  mov    ebp,esp
  10025f5:       83 ec 04               sub    esp,4
  10025f8:       8b 45 04               mov    eax,DWORD PTR [ebp+4]
  10025fb:       89 45 08               mov    DWORD PTR [ebp+8],eax
  10025fe:       8b 45 00               mov    eax,DWORD PTR [ebp]
  1002601:       56                     push   esi
  1002602:       89 ee                  mov    esi,ebp
  1002604:       83 ee 1f               sub    esi,31
  1002607:       89 46 1b               mov    DWORD PTR [esi+27],eax
  100260a:       5e                     pop    esi
  100260b:       e8 e9 03 00 00         call   0x10029f9
```

Figura 3.27. Trechos de códigos de três gerações do código malicioso Win32/Evol.

```
if (numero_execucoes > 15) {
    printf ("Período de avaliação expirado");
    exit();
}
```

Figura 3.28. Trecho de código para verificar o número de execuções de um programa.

ambiente em que o código está sendo executado e tomada de ação, caso haja violações a integridade ou caso o ambiente de execução esteja hostilizado (programa executando em um sistema operacional corrompido ou sobre um emulador ou com um depurador anexado). Existem diversos métodos para monitorar se um ambiente está hostilizado, porém estes são específicos para cada ambiente. Uma abordagem ampla destes métodos podem ser encontrada em [44, 58].

Estratégias de *tamper-proofing* para detectar violações a integridade consistem na auto-verificação de trechos de código do programa ou na inspeção lógica do programa. A verificação pode ser classificada em estática em que a verificação é feita em tempo de carregamento do programa ou dinâmica em que a verificação é feita durante tempo de execução. A inspeção lógica pode ser feita nos dados do programa ou no fluxo de execução. Dado que um desenvolvedor tem conhecimento *a priori* dos possíveis estados que seu programa pode atingir, o desenvolvedor pode inserir rotinas para verificar se uma variável ou o resultado de uma função está em conformidade [8].

Entre as estratégias comuns para a ação (resposta) de um sistema de *tamper-proofing* temos: terminar a execução do programa, restaurar o programa substituindo os trechos de código adulterados por trechos originais, retornar resultados incorretos, degradar o desempenho da aplicação, contactar o desenvolver [52] ou até mesmo punir o sistema do adversário, por exemplo, apagando seus documentos pessoais [54]. Sejam quais forem a forma de verificação e a resposta a ser tomada, é importante que as mesmas estejam distantes em tempo (com execução não simultânea) e espaço (não próximas no código do programa) para aumentar a dificuldade de um adversário em subvertê-las.

3.3.1. Auto-verificação de código

Os métodos de detecção a violações a integridade são baseados na auto-verificação do código através do uso de *hash* (resumo criptográfico). O programa durante ou antes de sua execução calcula o *hash* de uma determinada região no segmento de código do programa e compara-o com um valor de *hash* esperado. Caso o resultado da comparação divirja, o programa pode acionar a resposta que lhe foi programada.

A técnica proposta por Chang e Atallah [27] explora uma metodologia distribuída através de uma rede de verificadores e respondedores, diferenciando-a de outras formas de um único ponto de verificação (facilmente subvertido). A criação desta rede permite proteção mútua entre os verificadores e respondedores, em que os verificadores além de verificar blocos de código do programa, também verificam verificadores e respondedores. Caso o resultado do verificador divirja do original, o respondedor tem a função de substituir o trecho de código adulterado por um trecho de código original.

Figura 3.29 exibe a estrutura de memória em que dois trechos de código, ‘C1’ e

‘C2’ estão protegidos por verificadores de *hash* e respondedores, que substituem o código adulterado pelo original caso haja modificação. Os trechos de código ‘C1’ e ‘C2’ encontram-se protegidos mutuamente por ‘G1’, ..., ‘G5’. O trecho de código ‘C1’ é corrigido por ‘G3’ antes de ser executado, e este é verificado por ‘G1’ e ‘G5’. A guarda ‘G5’ é verificada por ‘G2’ e reparada caso esteja modificada.

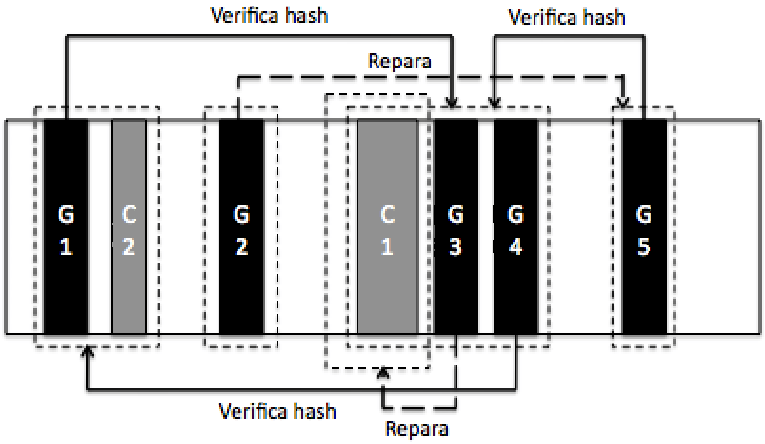


Figura 3.29. Estrutura de memória de um programa protegido.

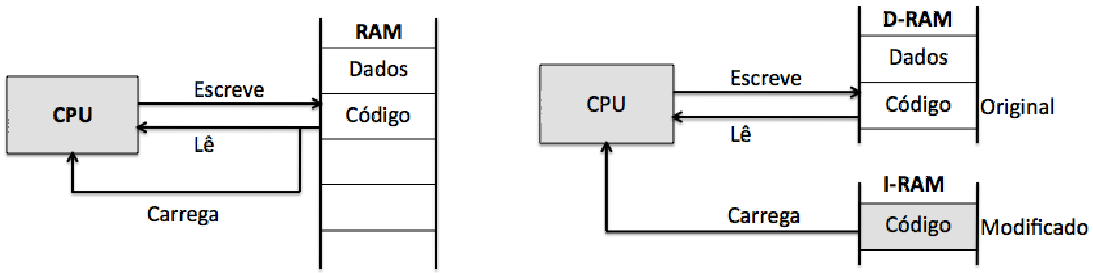
Uma técnica de ataque contra auto-verificadores de *hash* consiste em varrer o código a procura de instruções que comparam um cálculo de *hash* com um valor aleatório. Em programas que são distribuídos de forma distinta, um ataque de diferença poderia ser utilizado para localizar as exatas posições dos cálculos de *hash*. Uma estratégia para conter este problema consiste na duplicação de cada região que está sendo efetuado o cálculo de *hash* para que ao invés de comparar o resultado com um valor aleatório, comparar o resultado do cálculo de uma região com o cálculo da região duplicada. Uma desvantagem desta estratégia é a duplicação do código do programa.

As funções que calculam o *hash* também podem ser alvos de ataque de varredura por padrão, assim, é extremamente importante que estas funções estejam escondidas (o quanto possível) no código. Por exemplo, funções de criptografia tais como SHA-1 e MD5 apresentam assinaturas padrões que podem ser exploradas por um atacante. Horne *et al.* descreveu uma estratégia para dificultar estes ataques através da utilização de uma gama variada de funções de *hash*. Os autores utilizaram-se técnicas de otimização e ofuscação para gerar aproximadamente três milhões destas funções.

O trabalho de Horne *et al.* [24] envolve esconder os valores de *hash* para que estes sejam mais robustos diante de varredores de código que buscam instruções de comparação de um cálculo de *hash* com um valor aleatório. A idéia de seu trabalho consiste em diminuir o grau de visibilidade na instrução de comparação, fazendo com que as funções de cálculo de *hash* sempre retornem o valor zero, fazendo com que a comparação seja feita sobre um valor booleano (no caso 0), que é uma comparação típica na maioria dos programas. O algoritmo insere verificadores na forma “*if (hash(start_address, end_address)) respond();*” em que a resposta dada pela função “*respond()*” só executará se o cálculo do *hash* for diferente de zero em situações de modificação. Mais especificamente, o algoritmo faz uso de pontos para inserção de bits que fazem com que o cálculo do *hash*

seja nulo.

Os ataques aos métodos de auto-verificação de *hash*, como dito anteriormente, tipicamente envolvem análise para localização dos pontos de verificação para posterior modificação/anulação da verificação. Wurster *et al.* [50] desenvolveu um ataque aos métodos de verificação de *hash* baseado em modificações no ambiente, chamado de ataque de replicação de páginas. O ataque foca em uma deficiência dos algoritmos de auto-verificação de *hash* que assumem que instruções e dados devam compartilhar o mesmo espaço de endereçamento de memória (arquitetura von Neumann [1]) [43], ou seja, o código lido pelas rotinas de auto-verificação de *hash* é o mesmo que é carregado pelo processador para execução (vide Figura 3.30(a)). Contudo, um adversário pode violar esta suposição pela criação de uma arquitetura de memória virtual Harvard [2] em que memórias para instruções e dados são distintas (vide Figura 3.30(b)). Isso é feito através da modificação do sistema operacional para que este replique as páginas de memória das instruções tal que a leitura e o carregamento de instruções busquem valores de diferentes páginas de memória. Com isso, um adversário pode alterar o código na memória de instruções sem ser perceptível para as rotinas de verificação de *hash* que lêem a memória de dados.

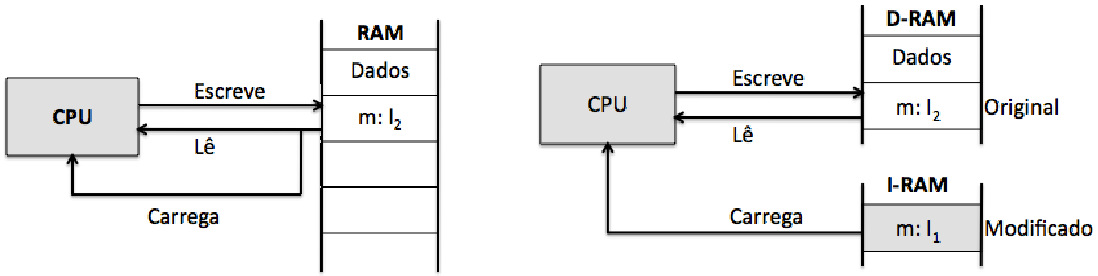


(a) Arquitetura von Neumann. (b) Arquitetura Harvard.

Figura 3.30. Ataque aos algoritmos de verificação de *hash*.

O trabalho de Giffin *et al.* [43], ao contrário do trabalho de Wurster *et al.* [50], apresenta uma técnica para detectar ataques de replicação de páginas. A técnica é baseada em códigos auto-modificáveis e envolve as seguintes etapas: sobreescrever uma instrução 'I_1' do endereço *m* com uma instrução 'I_2'; ler o valor contido no endereço *m*; executar a instrução do endereço *m*. Para que o ataque seja perceptível ao usuário é necessário que a instrução 'I_2' altere o fluxo de execução do programa.

Durante a execução do programa, se a instrução lida for 'I_2' e a execução do programa seguir o fluxo de execução de 'I_2' significa que o ambiente é regular ou que não está sobre ataque de replicação de páginas (vide Figura 3.31(a)). Caso contrário, estamos diante de um ataque de replicação de página em que a instrução lida é 'I_2' e a execução do programa não seguiu o fluxo de execução ditado pela instrução 'I_2', ou seja, a instrução carregada foi 'I_1' que significa que o ambiente está alterado ou sob ataque por replicação de páginas (vide Figura 3.31(b)).



(a) Arquitetura von Neumann. (b) Arquitetura Harvard.

Figura 3.31. Detecção de ataques de replicação de páginas.

3.3.2. Inspeção da lógica de execução

Os métodos de auto-verificação de *hash* possuem alguns desvantagens tais como a facilidade de detectar uma rotina de verificação dada a sua natureza atípica de operação, desde que a maioria dos programas não lêem seu próprio segmento de texto (código) e o fato de estes métodos não tratarem alterações no comportamento ou dados do programa. Por exemplo, um adversário pode temporariamente alterar o valor de um registrador que contém o resultado de retorno de uma função antes mesmo de a função retornar sem precisar alterar o código do programa. Métodos que inspecionam a corretude dos dados e do fluxo de controle visam a atender as deficiências dos métodos de auto-verificação de *hash*.

Chen *et al.* [31] definiu o conceito de *Oblivious Hashing* em que um adversário não tem a percepção que o código está sendo verificado. A idéia de Chen *et al.* é calcular o *hash* do traço de execução de um trecho de código, permitindo probabilisticamente e deterministicamente verificar o comportamento do programa em tempo de execução. Códigos que calculam o *hash* são inseridos no código fonte do programa que implicitamente calculam o *hash* no contexto de execução do código do programa. O objetivo é que os códigos para o cálculo do *hash* estejam misturados com o código original com a finalidade de dificultar a distinção destes por um adversário. Os *hashes* são basicamente calculados sobre valores de variáveis e no resultado de predicados de fluxo de controle. Outro trabalho que seguiu a idéia de *Oblivious Hashing* é o de Jacob *et al.* [56] que vincula a sobreposição de blocos básicos de instruções x86 com o cálculo de *hash*. Este cálculo é efetuado sem a necessidade de ler o código, e por isso, invulnerável a ataques de separação de memória.

Outra técnica de *tamper-proofing* consiste em remotamente detectar e responder a violações de integridade. Este cenário visa à proteção de um programa que executa em um local não confiável (lado do cliente) monitorado por comunicação com um programa que executa em um local confiável (lado do servidor). O servidor além de fornecer recursos aos pedidos do cliente, oferece mecanismos para verificar a integridade do programa e responder à violações. Em um sistema cliente-servidor em que o cliente espera recursos advindos do servidor, uma maneira fácil de responder a uma violação de integridade consiste na interrupção da comunicação. A tarefa de verificação não é básica quanto a tarefa de resposta pois o servidor não acessa diretamente o código do lado do cliente. O servidor poderia requisitar o *hash* de um determinado trecho de código através do canal de comunicação, porém nada garante que a resposta do cliente seria legítima. A resposta

pode ser forjada pela modificação do código cliente ou do ambiente de execução ou pela interceptação e substituição das mensagens de rede.

Soluções para verificar integridade de modo remoto são baseadas no compartilhamento de código entre o cliente e do servidor, levando em consideração que códigos mais críticos estejam no servidor. O balanceamento do compartilhamento reflete na carga computacional e na latência comunicação. Em situações que grande parte do código está contida no servidor significaria uma carga computacional alta no servidor e uma latência de comunicação alta para o cliente. Caso contrário, ou seja, a maior parte de código contida no cliente refletiria uma latência de comunicação baixa para o cliente e uma carga computacional baixa para o servidor. Contudo, nesta situação é difícil o servidor garantir que o código do cliente não foi modificado. As técnicas para *tamper-proofing* remoto são niveladas em um balanceamento intermediário entre carga computacional e latência de comunicação.

A técnica de Zhang e Gupta [38] automaticamente divide um programa em duas partes: uma aberta que executará no cliente e outra fechada que executará no servidor, podendo assim, proteger partes sensíveis do código. Esta técnica trata problemas como latência e largura de banda mantendo estruturas de dados grandes no lado do cliente e restringindo a execução do servidor e do cliente em uma rede local.

Outra técnica consiste na requisição de um valor de *hash* pelo servidor de um determinado trecho do código em execução no cliente, com o adicional de avaliar a legitimidade da resposta do cliente pela medição de determinadas propriedades do ambiente do cliente, como por exemplo, o tempo gasto para o cliente retornar o *hash* [51]. Esta técnica é baseada em algumas suposições para garantir que o código em execução no cliente não foi modificado. Estas suposições incluem: conhecimento da configuração de hardware do cliente, latência entre cliente e servidor, restringir a comunicação do cliente para que durante a verificação o cliente só mantenha comunicação com o servidor e o código do cliente tem que executar independentemente de outros códigos presentes no cliente.

A função para cálculo de *hash* no cliente é implementada de tal forma que se um adversário modificar o código, o mesmo retornará um valor de *hash* incorreto ou levará um tempo notável para efetuar o cálculo. A notabilidade de tempo justifica as suposições consideradas anteriormente, pois caso o servidor desconhecesse o poder computacional do cliente ou a velocidade de comunicação, o servidor não poderia estimar o tempo considerado “legítimo” para calcular o *hash*.

Outra idéia é de utilizar ofuscação dinâmica como um método de *tamper-proofing* remoto, fazendo com que o servidor envie versões modificadas (ofuscadas) para o cliente de modo que seja difícil para um adversário analisar o código. A técnica mantém uma representação do código do cliente tanto no lado do cliente como no lado do servidor. Este código é dividido em blocos de tal forma que a execução de cada bloco é feito através de requisições ao servidor. Paralelamente, o servidor contém um mecanismo de mutação que modifica os blocos do cliente e compartilha os blocos modificados com o cliente. Quando o cliente requisita ao servidor um bloco, este recebe um novo bloco do servidor caso o bloco esteja modificado. A técnica também pode enviar blocos para o cliente assim que modificados. Este processo de atender a requisições e enviar blocos é uma tarefa responsável por um processo escalonador executando no lado do servidor.

A robustez desta técnica diante de violações de integridade é relacionada com a taxa em que o servidor gera e envia blocos modificados para o cliente e a capacidade que um adversário tem de analisar um código que é constantemente alterado. É interessante que as transformações gerem códigos ofuscados diferentes continuamente e que o tempo de análise de um código por um adversário seja inferior a criação e envio dos blocos pelo servidor. Uma possibilidade de ataque consiste em ignorar os blocos enviados pelo servidor e copiar um estado fixo do programa contido na memória para análise fora do tempo de execução.

3.4. Marca d'água

Dizemos que um código C possui a propriedade σ como *marca d'água* se a propriedade σ pode ser verificada em C e em qualquer código C' obtido de C através de uma "pequena modificação". O conceito de "pequena modificação" pode ser definido de maneira rigorosa — por exemplo, o número de bits que devem ser alterados em C para se obter C' — de todo modo, uma modificação sobre C deveria ser considerada pequena se pudesse ser facilmente conduzida por um adversário. A propriedade σ pode ser definida tanto de maneira estática — por exemplo, um padrão de bits encontrado ao longo do código binário — quanto de maneira dinâmica — padrões somente verificáveis em tempo de execução.

A marca d'água fornece subsídios para se identificar o uso indevido de um software. Neste sentido, a marca d'água pode ser interpretada como o último estágio na defesa de um software: uma vez que a ofuscação falha em seu objetivo de **dificultar a análise** e as ferramentas de tamper-proofing falham em seu objetivo de **imperdir a execução indevida**, ainda assim, é possível identificar que um determinado software, em execução em um dispositivo, está sendo utilizado indevidamente. Embora a marca d'àgua não impeça o uso indevido de um software, ela, ao menos, fornece meios de que outras ações — por exemplo, ações legais — sejam tomadas.

Exemplo 1. Um software possui uma área que contém o nome do seu desenvolvedor. O nome do desenvolvedor poderia ser inserido em uma área do programa jamais executada, por exemplo, depois de um salto incondicional. A verificação da marca d'água seria feita através da busca, no software, do nome do desenvolvedor, cuja presença indicaria sua autoria. Uma forma de remoção da marca d'água por um atacante seria simplesmente remover a área do código que indica sua autoria.

Exemplo 2. A técnica apresentada no Exemplo 1 é bastante singela e passível de ataque de remoção bastante rudimentar. Uma maneira de se dificultar o ataque de remoção descrito anteriormente seria inserir a informação de autoria em uma quantidade maior de posições do código. Além disso, para dificultar a "remoção automática", em que o atacante simplesmente busca e remove um padrão fixo de bits, a identificação de autoria do software poderia ser inserida através de identificadores variáveis — por exemplo, dependentes de posição de memória ou de instruções próximas. A remoção da marca d'água, neste caso, demandaria a localização e remoção de todas as instruções não executadas no código. Técnicas de ofuscação podem ser usadas para dificultar ainda mais a remoção da marca d'água.

Vimos, nos simples Exemplos 1 e 2, que duas operações estão associadas a uma técnica de marca d'água: a operação de *inserção* da marca d'água e a operação da *extra-*

ção da marca d'água para posterior verificação. A operação de inserção é a técnica de transformação de software que irá receber um software qualquer e dotá-lo de uma propriedade σ, posteriormente verificável. A operação de extração é aquela que irá tomar um software qualquer e nele buscar a existência da propriedade σ que irá evidenciar o uso devido ou indevido daquele software.

Os extratores podem, ainda, ser classificados como *blind* ou *informed*. Um detector é classificado como *informed* se, para extrair a marca d'água, é neessário estar de posse da "obra original", ou seja, do objeto sobre o qual foi inserido a marca d'água. Já um extrator *blind* é capaz de extrair a marca d'água de qualquer objeto, mesmo que a versão "original deste objeto não esteja disponível". Um exemplo de uso de extratores *informed* é o caso do rastreamento de transações em que o proprietário da obra original busca identificar quem distribuiu ilegalmente a obra — como o usuário do extrator é o proprietário da obra, então esta obra estará disponível, *à priori*, durante o processo de extração. Em outros casos, a obra não estará disponível, como é o caso, por exemplo, da extração da marca d'água de um vídeo para fins de verificação de autenticidade — neste caso, o vídeo é uma obra desconhecida, de forma que o extrator deverá ser capaz de obter a marca d'água sem o conhecimento da obra original. Na prática, extratores *informed* apresentam melhores resultados quando comparados a extratores *blinded* [63]. Isso porque a obra cuja marca dágua está sendo verificada pode sofrer uma série de alterações antes do processo de extração — tanto alterações maliciosas, visando à própria corrupção da marca d'água, por exemplo, quanto alterações lícitas, tal como a própria inclusão da marca d'água. Assim, a obra original torna-se uma boa referência para o algoritmo de extração da marca d'água.

Um sistema de marca d'água é dito *privado* quando apenas usuários autorizados são capazes de extrair a marca d'água; caso contrário, o sistema de marca d'água é *público*.

A *transparência* de uma marca d'água está associada ao grau de similaridade entre a obra original e a obra com marca d'água. Quanto menos perceptível for a marca d'água, mais transparente ela será considerada. A *capacidade* de um sistema de marca d'água diz respeito à quantidade de informação que este sistema é capaz de inserir na obra. A *robustez* de uma marca d'água é a sua resistência a manipulações da obra. O grau de robustez de uma marca d'água pode ser classificado como *seguro*, *robusto*, *semi-frágil* ou *frágil*, de acordo com o conjunto de manipulações da obra ao qual a marca d'água é capaz de resistir [63]. Na prática, observa-se um compromisso entre transparência, capacidade e robustez em sistemas de marca d'água: quanto mais informação a obra carrega na forma de marca d'água e quanto maior a sua robustez, menor será a transparência da marca d'água.

A Figura 3.32 ilustra um sistema de marca d'água genérico.

Antes de examinarmos técnicas mais sofisticadas para codificar uma marca d'água no software, vamos estabelecer as diferenças entre marca d'água e *fingerprinting* e explorar alguns possíveis ataques à marca d'água. Em um cenário de marca d'água todas as cópias do mesmo software são distribuídas de forma idêntica, ou seja, as marcas são codificadas igualmente. Em um cenário de *fingerprinting* toda cópia distribuída contém uma marca diferente associada ao cliente, possibilitando assim, um rastreamento do cliente

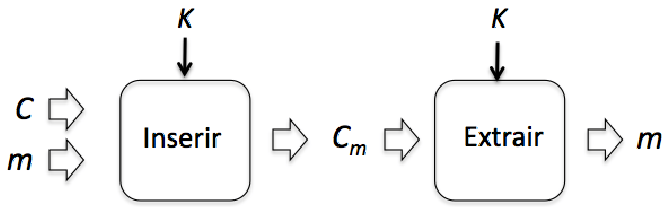


Figura 3.32. Sistema de *watermarking* em software.

diante de posse de uma cópia ilegal. Um problema deste cenário é a possibilidade de ataques de diferença na qual um adversário adquire cópias distintas de um mesmo software para identificar a localização da marca, facilitando sua remoção ou falsificação.

Marcas d'água são também passíveis de outros tipos de ataque. Ataque de distorção consiste na aplicação de técnicas de transformação de código que preservam a semântica do código, como otimização, portabilidade, compressão e ou ofuscação para para impedir que o processo de extração de um sistema de *watermarking* consiga reaver a marca. Ataque de adição envolve a adição de uma outra marca por um adversário com o objetivo de alegar em uma ação judicial que o software é de sua autoria. Ataques à marca d'água são considerados bem sucedidos se o processo de extração não consegue legitimar a marca original e a funcionalidade do software adulterado é a mesma do software original.

As técnicas de marca d'água de software podem ser classificadas em estáticas ou dinâmicas. Nas estáticas, as marcas são armazenadas no próprio código executável e o processo de inserção e extração é feito sem a necessidade do programa ser executado ou interpretado. Nas dinâmicas, o programa ou parte dele precisa ser executado ou interpretado para que a marca d'água seja extraída. O processo de extração de uma marca d'água dinâmica utiliza-se de uma sequência especial de entrada para extrair a marca de um determinado estado do programa. Estas marcas dinâmicas possuem a vantagem de serem mais resilientes diante de ataques de falsificação, remoção ou adição, pois as marcas estão associadas a um estado dinâmico do programa que por conseguinte, não são estaticamente visíveis para um adversário, contudo, são ainda suscetíveis a ataques de distorção.

3.4.1. Marcas d'água estáticas

A técnica de Peter Wayner [29] de inserir uma mensagem secreta em uma lista ordenada pode ser utilizada para inserir uma marca em qualquer lista de uma linguagem de programação que possa ser reordenada. A idéia consiste na codificação de um inteiro (marca d' água) através da reordenação das declarações de um programa ou atribuições desde que estas não sejam dependentes uma das outras. Por exemplo, assuma que a marca é '213', Figura 3.33 mostra como a marca é inserida por reordenação.

Neste caso, o extrator teria que ter acesso ao código original e ao programa com a marca d'água contida para extrair a marca d'água, ou seja, um extrator *informed*.

A reordenação também pode ser feita nos blocos básicos de um grafo de fluxo de controle [47]. A idéia é similar a técnica anterior ao ponto que ao invés de reordenar declarações de um programa reordena seus blocos básicos através de instruções incondi-

```
/* Código Original */
1: a = 5;
2: b = 8;
3: c = 4;

/* Código Reordenado*/
2: b = 8;
1: a = 5;
3: c = 4;
```

Figura 3.33. Reordenação das atribuições para inserção de marca d'água.

cionais ou condicionais através de predicados opacos.

Outra técnica consiste na renumeração dos registradores de um código binário para inserir a marca d'água. Vamos utilizar a mesma marca d'água da técnica anterior, ou seja, '213'. Figura 3.34 ilustra esta técnica.

```
/* Código Original */
1: r1 = 3;
2: r2 = r1 + 3;
3: r2 = r2 * 4;
4: r3 = r2 + r1;

/* Código Renumerado */
1: r2 = 3;
2: r1 = r2 + 3;
3: r1 = r1 * 4;
4: r3 = r1 + r2;
```

Figura 3.34. Renumeração dos registradores para inserção de marca d'água.

As técnicas até agora examinadas são consideradas frágeis, pois basta um adversário reordenar declarações ou blocos básicos ou renumerar registradores aleatoriamente para que as marcas não sejam mais válidas.

Moskowitz e Cooperman [11] desenvolveram uma técnica de marca d'água de software baseada no uso de imagens, vídeos ou áudio embarcados no software. Caso o software não possua um destes meios, é feita a inserção de um no código do programa. A idéia é ao invés de criar métodos para inserir e extrair marca d'água de software, utilizar algoritmos de inserção e extração já existentes para imagens, vídeos e áudio em software. A técnica embarca trechos de código essenciais para a execução do programa nestes meios, de tal forma que estes trechos de código só são executados diante de um processo de extração durante a execução. Isso caracteriza o método como sendo também um método de *tamper-proofing*, pois caso a marca seja distorcida de algum modo, não será possível extrair o código e o programa não apresentará a mesma funcionalidade. A distorção de uma imagem, por exemplo, pode ser feita através da ferramenta Stirmark [66]. Esta possui uma coleção de transformações que fazem grande parte dos algoritmos de marca d'água de imagem falharem no processo de extração.

```
/* Código Original */
...
    if ( x < y ) jmp b
a: bloco_de_instrução_a
b: bloco_de_instrução_b
...

/* Código Alterado */
...
    if ( x >= y ) jmp a
b: bloco_de_instrução_b
a: bloco_de_instrução_a
...
```

Figura 3.35. Inversão da condição em saltos condicionais.

Monden *et al.* [15] apresentou um algoritmo de marca d'água em que a marca é codificada através da modificação dos *opcodes* de uma função espúria. Esta função é adicionada ao programa e sua execução é disfarçada através de predicados opacos, fazendo com que a função nunca seja executada. Um ataque a este algoritmo basicamente envolve a construção do grafo de chamadas do programa e a utilização de *profiling* para efetuar a contagem de quantas vezes as funções estão sendo invocadas. Como a função espúria que contem a marca nunca é executada, um adversário trivialmente pode remover a função, apagando consequentemente a marca d'água do programa.

Venkatesan *et al.* [23] implementou um algoritmo de marca d'água que adiciona arestas redundantes no grafo de de fluxo de controle do programa, fazendo com que um adversário não consiga separar ou contar quais arestas são partes do programa e quais são pertencentes a marca d'água, sendo assim, mais resiliente à ataques de remoção auxiliado por *profiling*. A inserção é baseada na codificação de uma marca em um grafo que seja compatível com um grafo de fluxo de controle, ou seja, que seja redutível e que possua grau de conectividade 'um' ou 'dois' visto que as estruturas de controle ('if', 'for' e 'while') de um programa apresentam estas características. A estrutura 'switch' apesar de apresentar um grau de conectividade maior não é uma estrutura comumente utilizada na maioria dos programas. Uma possibilidade de ataque a esta técnica é alterar os saltos condicionais invertendo sua condição. Os códigos da Figura 3.35 ilustram esta inversão em que o código alterado inverte a condição do salto do código original. Contudo, a técnica propõe o uso de um grafo que seja resistente a estas inversões [40].

Outro algoritmo de marca d'água foi proposto por Cousot e Cousot [42] na qual a idéia é inserir uma marca d'água que possa ser extraída utilizando análise estática. A técnica proposta é baseada na propagação de constantes de um análise de fluxo de dados. Figura 3.36 ilustra como é o processo de inserção de uma marca d'água '2' na rotina maior. Para extração da marca d'água temos uma chave secreta '3' na qual durante a análise de propagação de constantes, os valores de 'w' dentro e fora do laço são constantes e resultam na marca d'água '2' se aplicados com o módulo da chave '3', *i.e.*, '10 mod 3 = 2' e '14 mod 3 = 2'. Os autores propõem o uso de polinômios no lugar de constantes para que a localização da marca d'água seja mais complicada para um adversário.

```
int maior(int x[], int size) {
     int maior = x[0];
     int i = 1;
     int w = 10;
     while ( i < size) {
          if ( x[i] > maior )
               maior = x[i];
          w = 14;
          i++;
     }
     return maior;
}
```

Figura 3.36. Marca d'água '2' contida na rotina maior.

Os algoritmos abordados até o momento basicamente fazem uso ou da codificação de um identificador como permutação do código original, ou da inserção de um novo código não funcional. Estes métodos são passíveis de ataque pelas mesmas transformações aplicadas para inserção da marca d'água, ou seja, um adversário pode aplicar transformações como reordenação e inserção para dificultar a extração da marca d'água. A seguir, examinaremos um algoritmo que extrai a marca da d'água através da execução ou interpretação do programa.

3.4.2. Marcas d'água dinâmicas

Algoritmos de marcas d'água dinâmicas são baseados na inserção da marca d'água em um estado do programa. A extração da marca d'água envolve a saída do programa, ou a estrutura de dados, ou um código dinamicamente gerado pelo programa. Este procedimento pode ser feito por exemplo, examinando a pilha, os *threads* ou os registradores do programa. Como um adversário não tem conhecimento de como o processo de extração é feito, marcas d'águas dinâmicas acabam sendo mais resilientes diante de ataques de distorção, falsificação ou remoção.

Collberg *et al.* [41] implementou um algoritmo para codificar uma marca d'água nos saltos do programa. A idéia envolve a adição de saltos condicionais no programa que diante de uma determinada entrada codifica o bit '1' caso a condição do salto for satisfeita ou codifica o bit '0' caso contrário. Figura 3.37 ilustra este algoritmo que diante de um vetor maior que '1' embarca a marca d'água '10010_2' nos saltos dos endereços da linha '6' até a linha '10'. É claro que que está técnica de inserção é facilmente subvertida através da inversão dos saltos condicionais (vide Figura 3.35).

Este trabalho propõe uma maneira para conter estes ataques de inversão de saltos. A idéia consiste na inserção de uma sequência de predicados e saltos em posições do código que são executadas várias vezes diante de uma determinada entrada. Na primeira execução destes códigos são identificadas as direções dos saltos para que nas execuções subsequentes seja gerada a codificação da marca d'água. Este algoritmo insere predicados que são dependentes das variáveis do programa, e a codificação da marca d'água é feita analisando o traço de execução do programa. Esta análise é feita para determinar predicados que são verdadeiros na primeira execução e falsos nas execuções posteriores.

```
/* Rotina Original */

int maior(int x[], int size) {
1:     int maior = x[0];
2:     int i = 1;
3:     while ( i < size) {
4:          if ( x[i] > maior )
5:               maior = x[i];
6:          i++;
     }
7:     return maior;
}

/* Rotina com marca d'água */

int maior(int x[], int size) {
1:     int maior = x[0];
2:     int i = 1;
3:     int u;
4:     int t = 0;
5:     while ( i < size) {
6:           if (t = 0) u++;
7:           if (t != 0) u++;
8:           if (t != 0) u++;
9:           if (t == 0) u++;
10:          if (t != 0) u++;
11:          if ( x[i] > maior )
12:               maior = x[i];
13:         i++;
     }
14:     return maior;
}
```

Figura 3.37. Inserção da marca d'água '10010_2' na rotina maior.

Figura 3.38 apresenta um código que extrai a marca d'água '10010_2' dado o vetor de entrada '$x = [0105]$'. Observe que dado este vetor, os códigos da linha '6' até a linha '11' entram em execução por duas vezes durante a execução desta rotina. Na primeira execução os valores de 'i = x[i] = maior = 1' e, portanto, todos os predicados são satisfeitos. Na segunda execução, o valor de 'i = 3' e 'x[i] = maior = 5', o que satisfaz os predicados das linhas '8', '9' e '11'. O bit '1' da marca d'água é codificado quando o predicado da segunda execução difere do predicado da primeira execução e a do bit '0' caso contrário. Através de uma análise de fluxo de dados, um adversário pode remover estas condições uma vez que a variável 'u' não altera o comportamento do programa. Um disfarce possível é atribuir 'u' a uma variável utilizada no programa, por exemplo, $if\ (P^f)\ x[i] = u$.

Collberg *et al.* [41] também propõe uma maneira de inserir uma marca d'água baseada na ofuscação de saltos incondicionais (vide Figura 3.14). A idéia é mapear cada bit da marca d'água em um salto incondicional '$a_i \mapsto b_i$' na qual $1 \leq i \leq n$ representa o

```
/* Rotina com marca d'água */

int maior(int x[], int size) {
1:     int maior = x[0];
2:     int i = 1;
3:     int u;
4:     while ( i < size) {
5:          if ( x[i] > maior ) {
6:               maior = x[i];
7:               if (i == x[i]) u++;
8:               if (x[i] == maior) u++;
9:               if (x[i] == maior) u++;
10:              if (i == x[i]) u++;
11:              if (x[i] == maior) u++; }
12:         i++;
      }
7:     return maior;
}
```

Figura 3.38. Codificação da marca d'água '10010_2' na rotina maior.

intervalo de endereços do programa. No exemplo da Figura 3.39, o bit '1' é codificado nos saltos para frente (*i.e.*, $a_i < b_i$) e o bit '0' nos saltos para trás (*i.e.*, $a_i > b_i$). Um salto incondicional $l_{comeco} \mapsto l_{fim}$ é incorporado em uma sequência de saltos incondicionais ofuscados a_0, a_1, a_2 e a_3 inseridos no programa. Este ofuscamento substitue um salto incondicional por uma chamada de função e manipula o endereço de retorno desta função para ser o endereço destino do salto incondicional. Este método também insere um salto incondicional antes das chamadas de função a_1, a_2 e a_3, pois estas não podem ser executadas exceto se advindas da execução de a_0. Estas marcas codificadas no próprio código executável são mais robustas contra ataques de adição e distorção uma vez que estes saltos ofuscados manipulam endereços absolutos e, por isso, qualquer alteração no endereçamento resulta em um programa não funcional.

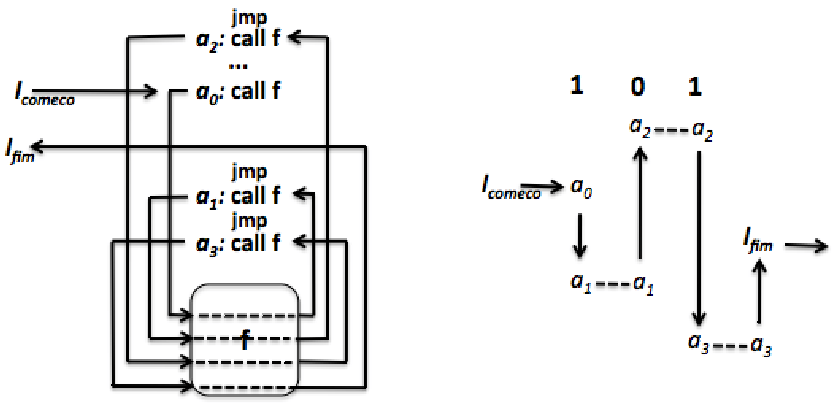


Figura 3.39. Codificação da marca d'água '101_2' através de saltos incondicionais ofuscados.

3.4.3. Marca d'água *versus* esteganografia

O leitor iniciado nas técnicas da esteganografia poderá ter se perguntado qual seria a diferença desta para a marca d'água. De uma maneira bastante concisa, uma técnica esteganográfica permite que se insira uma informação em um objeto de tal maneira que se torna muito difícil "detectar" a existência daquela informação no objeto. Não é objetivo primário da esteganografia que a informação embarcada em um objeto seja altamente resiliente a manipulações do objeto. Já a marca d'água busca associar uma informação a um objeto de tal maneira que seja "muito difícil" remover a informação daquele objeto, não se preocupando, primariamente, se a informação é ou não detectável.

A confusão observada algumas vezes na literatura entre os dois conceitos advém do fato de que as técnicas utilizadas tanto para esteganografia quanto para marca d'água atuam de maneira a "modificar" o objeto responsável pelo transporte da informação, sem que este objeto perca, no entanto, a sua funcionalidade inicial. Dessa forma, é possível o uso de ferramentas esteganográficas "puras" para a criação de marcas d'água. Deve-se ter claro, no entanto, que, uma vez descoberta, a marca d'água poderá ser facilmente removível. Além disso, observa-se que boa parte das ferramentas de marca d'água descritas na literatura inserem informação em um objeto efetuando sobre ele mudanças imperceptíveis, o que torna tais ferramentas passíveis de uso com finalidade esteganográfica.

Descrevemos um exemplo bastante concreto que contrapõe esteganografia e marca d'água. Suponha que se deseja transmitir uma determinada informação através de sua escrita sobre a superfície de um equipamento — por exemplo, um celular. As possíveis maneiras de se inserir a informação distinguem-se, entre outros, quanto ao tipo de tinta utilizado na escrita. Suponha que se utiliza uma tinta visível apenas sob radiação ultravioleta. Neste caso, tem-se um exemplo de esteganografia, já que a mensagem será invisível e "indetectável" sob iluminação convencional. Suponha, por outro lado, que se utiliza uma tinta de alta aderência — ou seja, de difícil remoção. Neste caso, tem-se um exemplo de marca dágua. O primeiro exemplo — da tinta invisível — seria usado, tipicamente, para a transmissão de uma informação sigilosa. No entanto, dado que a informação é de difícil detecção, o método poderia ser usado para inserir uma informação associada ao próprio equipamento, o que seria, em geral, objeto de uma marca d'água.

No mundo digital, técnicas esteganográficas podem aplicadas em áudio, imagem, texto e software. No contexto de áudio, uma técnica de marca d'água consiste em adicionar ecos nos bits menos significantes de som que são pouco perceptíveis para a audição humana. No contexto de imagem, pode-se aleatoriamente escolher dois bits da imagem e incrementar o brilho do primeiro *bit* em uma pequena quantidade e decrementar o segundo *bit* pela mesma quantidade. No contexto de textos, técnicas de *watermarking* envolvem alteração do espaçamento entre parágrafos ou palavras, assim como inserção de palavras sinônimas que codifiquem um padrão de bits. No contexto de software, pode-se inserir bits de informação em regiões do programa que nunca são executadas — por exemplo, após um salto incondicional. Em todos os casos a esteganografia será mais bem sucedida quanto menos perceptível for a modificação das características do objeto onde se insere a informação.

3.5. Conclusões

Todos meios de proteção, seja eles em software ou em hardware, são passíveis de ataques e podem ser subvertidos. Mecanismos baseados em hardware oferecem um nível de proteção mais elevado, porém apresentam desvantagens quanto ao custo, desempenho inferior e estresse para o usuário final em caso de extravio, atualização ou defeito. As técnicas de transformação de código degradam relativamente menos o desempenho e apresentam um custo inferior de desenvolvimento se comparado com os métodos de proteção baseado em hardware. Um outro ponto diz respeito ao fato que os hardwares possuirem um único ponto de falha, por exemplo, a proteção de uma chave em hardware, uma vez esta descoberta todo o sistema pode ser corrompido [48, 60]. As técnicas de transformação de código podem adicionar diferentes camadas de proteção dando maior robustez caso um adversário consiga burlar um nível de proteção. Porém nada impede que proteções de software e hardware coexistam. O balanceamento entre o nível de proteção com o *overhead* em desempenho e custo é uma decisão que cabe ao desenvolvedor do produto/programa.

Referências

[1] Neumann, J. von. . “First draft of a report on the EDVAC”,1945.

[2] Aiken, H.H. . “Proposed automatic calculating machine”. Unpublished manuscript, November, 1937. Also appeared in IEEE Spectrum, 1(8):62–69, Aug. 1964.

[3] Sharir, M.; Pnueli, A. . “Two approaches to interprocedural data flow analysis”. Program Flow Analysis: theory and applications. Englewood Cliffs: Prentice-Hall, 1981.

[4] Cohen F. . “Computer viruses—theory and experiments". In: IFIP-TC11, Computers and Security, pages 22–35, 1987.

[5] Cohen F. . “Current trends in computer viruses". In: International Symposium on Information Security, 1991.

[6] Cohen F. . “Operating system protection through program evolution". Computer Security, 12(6):565–584, 1993.

[7] Cohen F. . “A short course on computer viruses"(2nd ed.). John Wiley & Sons, Inc., New York, 1994.

[8] Blum M., Kannan S. . “Designing programs that check their work”. Journal of the ACM, 42(1):269–291, January 1995.

[9] Aucsmith, D. . “Tamper resistant software: An implementation”. In: Ross J. Anderson, editor, Information Hiding, First International Workshop, pp. 317–333, Cambridge, U.K., May 1996. Lecture Notes in Computer Science, Vol. 1174.

[10] Vliet, H. P. V. . “Crema — The Java obfuscator”. http://web.inter.nl.net/users/H.P.van.Vliet/mocha.html, January 1996.

[11] Moskowitz, S. A., Cooperman, Marc. . “Method for stega-cipher protection of computer code", US Patent 5,745,569, January 1996. Assignee: The Dice Company.

[12] Davidson, R. L., Myhrvold, N. . “Method and system for generating and auditing a signature for a computer program", US Patent 5,559,884, September 1996. Assignee: Microsoft Corporation.

[13] Majumdar, A., Thomborson, C. D., Drape, S. . “A Survey of Control-Flow Obfuscations”. In: Proceedings of the International Conference on Information Systems Security, 2006, pp. 353–356.

[14] Collberg, C.; Thomborson, C.; Low, D. . “A taxonomy of obfuscating transformations". Technical Report 148, Department of Computer Science, The University of Auckland, New Zealand, July 1997.

[15] Monden, A.; Iida, H.; Matsumoto, K.; Torii, K.; Ichisugi, Y. . “Watermarking method for computer programs”. In: Proc. of theSymposium on Cryptography and Information Security, 1998.

[16] Collberg, C.; Thomborson, C; Low, D. . “Manufacturing cheap, resilient, and stealthy opaque constructs”. In: ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL’98, San Diego, January 1998.

[17] Collberg, C.; Thomborson, C; Low, D. . “Breaking abstractions and unstructuring data structures”. In: Proc. 1998 IEEE International Conference on Computer Languages, pages 28–38.

[18] Collberg, C.,Thomborson, C. . “Software watermarking: Models and dynamic embeddings", In: Principles of Programming Languages 1999, POPL’99, San Antonio, TX, January 1999.

[19] Collberg C.; Thomborson C. . “Watermarking, tamper-proofing, and obfuscation — tools for software protection”. Technical Report TR00-03, The Department of Computer Science, University of Arizona, February 2000.

[20] Lie, D.; Thekkath, C.; Mitchell, M.; Lincoln, P.; Boneh, D.; Mitchell, J.; Horowitz, M. . “Architectural support for copy and tamper resistant software”. In: Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX), pp. 168–177, November 2000.

[21] Wang, C.; Hill, J.; Knight, J.; Davidson, J. .“Software tamper resistance: Obstructing static analysis of programs.” Technical Report CS-2000-12, 2000.

[22] Cho, W.; Lee, I.; Park, S. . “Against intelligent tampering: Software tamper resistance by extended control flow obfuscation”. In: Proc. World Multiconference on Systems, Cybernetics, and Informatics. International Institute of Informatics and Systematics, 2001.

[23] Venkatesan, R. , Vazirani, V., Sinha S. . “A graph theoretic approach to software watermarking". In: 4th International Information Hiding Workshop, Pittsburgh, PA, April 2001.

[24] Horne, B.; Matheson, L.; Sheehan, C.; Tarjan, R. E. . “Dynamic self-checking techniques for improved tamper resistance”. In: Security and Privacy in Digital Rights Management, ACM CCS-8 Workshop DRM 2001, Philadelphia, November 2001. Springer-Verlag, LNCS 2320.

[25] Ször, P.; Ferrie, P. “Hunting for metamorphic”. In: Proc. of the 11th Virus Bulletin Conference, Prague, Czech Republic, pp. 123–144, 2001.

[26] Wang, C.; Hill, J.; Knight, J.; Davidson, J. . “Protection of software-based survivability mechanisms”. In: Proc. International Conference of Dependable Systems and Networks, July 2001.

[27] Chang, H.; Atallah, Mikhail J. . “Protecting Software Code by Guards”. In: ACM CCS-8 Workshop on Security and Privacy in Digital Rights Management, 2002, pp 160–175.

[28] Collberg C.; Thomborson C. . “Watermarking, tamper-proofing, and obfuscation — tools for software protection”. In: IEEE Transactions on Software Engineering, New York, v. 28, n. 8, p. 735–746, 2002.

[29] Wayner, P. .“Disappearing Cryptography: Information Hiding: Steganography and Watermarking (2nd Edition)”. Morgan kaufmann Publishers Inc., San Francisco, CA, 2002.

[30] Wroblewski, G. . “General Method of Program Code Obfuscation”. PhD thesis, Wroclaw University of technology, Institute of Engineering Cybernetics, 2002.

[31] Chen, Y.; Venkatesan, R.; Cary, M.; Pang, R.; and Sinha, S.; and Jakubowski, M. H. . “Oblivious Hashing: A Stealthy Software Integrity Verification Primitive”. In: 5th International Workshop on Information Hiding, 2003, pp. 400–414.

[32] Christian Sven Collberg, Clark David Thomborson, and Douglas Wai Kok Low. . “Obfuscation techniques for enhancing software security”. U.S. Patent 6668325, December 2003.

[33] Kanzaki, Y.; Monden, A.; Nakamura, M; Matsumoto K. . “Exploiting self-modification mechanism for program protection”. In: Proc. of the 27th Annual International Conference on Computer Software and Applications, pp 170, Washington, USA, 2003.

[34] Horne, W. G.; Matheson, L. R.; Sheehan, C.; Tarjan, R. E. . “Software self-checking systems and methods”. U.S. Application 20030023856, January 2003. Assigned to InterTrust Technologies Corporation.

[35] Lakhotia, A.; Singh, P. K. . “Challenges in getting ‘formal’ with viruses”. Virus Bulletin, pp. 15–19, September 2003.

[36] Linn, C.; Debray, S. . “Obfuscation of executable code to improve resistance to static disassembly”. In: Proceedings of the 10th Computer and Communications Security (CCS), 2003, pp. 290–299.

[37] Ogiso, T.; Sakabe, Y.; Soshi, M.; Miyaji, A. . “Software obfuscation on a theoretical basis and its implementation”. In: IEEE Trans. Fundamentals, E86-A(1), January 2003.

[38] Zhang, X.; Gupta, R. . “Hiding program slices for software security”. In: Proc. of the International Symposium on Code Generation and Optimization, 2003, pp. 325–336, Washington, DC, 2003. IEEE.

[39] Christodorescu, M.; Somesh, J. . “Testing Malware Detectors”. In: Proc. of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA’04), July 11-14, 2004, Boston, Massachusetts, USA.

[40] Collberg, C.; Huntwork, A.; Carter, E.; Townsend G. . “Graph theoretic software watermarks: Implementation, analysis, and attacks”. In Workshop on Information Hiding, pp. 192-207, 2004. Springer-Verlag.

[41] Collberg, C.; Debray, S.; Carter, E.; Huntwork, A.; Linn, C.; Stepp, M. . “Dynamic Path-Based Software Watermarking”. In: Proc. SIGPLAN ’04 Conf. on Prog. Language Design and Implementation (PLDI 04), June 2004.

[42] Cousot, P.; Cousot, R. . “An abstract interpretation-based framework for software watermarking”. In: Proc. of Principles of Programming Languages, 2004, Venice, Italy. ACM.

[43] Giffin J. T.; Christodorescu M.; Kruger L. “Strengthening software self-checksumming via self-modifying code". In: 21st Annual Computer Security Applications Conference, 2005, pp. 23–32, IEEE Computer Society.

[44] Holz T.; Raynal F. . “Detecting honeypots and other suspicious environments”. In: Proceedings of the 2005 IEEE Workshop on Information Assurance and Security, U.S. Military Academy, West Point, NY, June 2005.

[45] Lakhotia, A.; Kumar, E. U.; Venable, M. . “A method for detecting obfuscated calls in malicious binaries”. In: IEEE Transactions on Software Engineering, Piscataway, v. 31, n. 11, pp. 955–968, 2005.

[46] Madou, M.; Anckaert, B.; Moseley, P.; Debray, S.; De Sutter, B.; De Bosschere, K. . “Software protection through dynamic code mutation”. In: 6th International Workshop in Information Security Applications, August 2005.

[47] Myles, G.; Collberg, C.; Heidepriem, Z.; Navabi, A. . “The evaluation of two software watermarking algorithms”. Software: Practice and Experience, 35(10):923-938,2005.

[48] Skorobogatov, S. P. . “Semi-invasive attacks – A new approach to hardware security analysis”. Technical Report 630 - University of Cambridge, 2005.

[49] Udupa, S. K; Debray, S. K.; Madou, M. . “Deobfuscation: Reverse engineering obfuscated code”. In: WCRE ’05: Proceedings ofthe 12th Working Conference on Reverse Engineering, pages 45–54, Washington, DC, 2005. IEEE.

[50] Wurster, G.; van Oorschot, P. C.; Somayaji, A. . “A generic attack on checksumming-based software tamper resistance”. In: IEEE Symposium on Security and Privacy, Oakland, CA, pp. 127–138, May 2005.

[51] Seshadri, A.; Luk, M.; Perrig, A.; van Doorn, L.; Khosla, P. . “Enternally verifiable code execution”. Commun. ACM, 49(9):45–49, 2006.

[52] Tan, G.; Chen, Y.; Jakubowski, M. H. . “Delayed and controlled failures in tamper-resistant systems”. In: Information Hiding, 2006. Springer-Verlag.

[53] Wong, W.; Stamp, M. . “Hunting for Metamorphic Engines”. Journal in Computer Virology (Department of Computer Science, San Jose State University).

[54] Farrell, N. . “Mac Display Eater kills home files”. The Inquirer (February 27 , 2007), www.theinquirer.net/default. aspx?article=37824.

[55] Hardekopf, B.; Lin, C. . “The ant and the grasshopper: Fast and accurate pointer analysis for millions of lines of code”. SIGPLAN Not., 42(6):290–299, 2007.

[56] Jacob, M.; Jakubowski, M. H.; Venkatesan, R. . “Towards integral binary execution: implementing oblivious hashing using overlapped instruction encodings”. In: Proc. of the 9th Workshop on Multimedia and Security, pp. 129–140, New York, 2007. ACM.

[57] Srinivasan, R. . “Protecting anti-virus software under viral attacks”. M.Sc. Thesis, Arizona State University.

[58] Ferrie, P. . “Anti-Unpacker Tricks”. In: 2nd International CARO Workshop, 2008, The Netherlands.

[59] Baker, D. . “Making a Secure Smart Grid a Reality”. Journal of Energy Security 2009.

[60] Goodspeed *et al.*. “Low level Design Vulnerabilities in Wireless Control System Hardware”, S4 2009 papers.

[61] IDA Pro - Disassembler. 2009. Data Rescue, Liege, Belgium. Available from Internet: <http://www.datarescue.com. Último acesso July 2009>.

[62] Boccardo, D. R. . “Context-Sensitive Analysis of x86 Obfuscated Executables” PhD thesis, Universidade Estadual Paulista, Departamento de Engenharia Elétrica - FEIS, 2009.

[63] Oliveira, L. P. M. . “Marca d’água Frágil e Semi-frágil para Autenticação de Vídeo no Padrão H.264”. Dissertação de mestrado, Universidade Federal do Rio de Janeiro, 2009.

[64] z0mbie. “Automated reverse engineering: Mistfall engine.” Publicado online em http://z0mbie.host.sk/autorev.txt . Último acesso, Julho 2010.

[65] PECompact. “Windows executable compressor”. Bitsum Technologies. http://www.bitsum.com . Último acesso, Julho 2010.

[66] Petitcolas, F. A. P. . “Stirmark Benchmark 4.0”. http://www.petitcolas.net/fabien/watermarking/stirmark . Último acesso, Julho 2010.

Capítulo

4

Aspectos de Segurança na Interconexão de Redes Celulares e WLANs

Silas Leite Albuquerque, Paulo Roberto de Lira Gondim e Cláudio de Castro Monteiro

Departamento de Engenharia Elétrica, Faculdade de Tecnologia, Universidade de Brasília

Abstract

Wireless communication is extremely present in our everyday lives. Among the standards currently on the market two stand out: the cellular networks and WLAN. These standards are considered complementary from the moment that one enables high transmission rates and other large coverage areas. Thus, the integration of these patterns is very attractive. However this is not straightforward and presents many challenges. One in particular regards to security aspects considered in a transition between two networks of different standards (inter-technology handover). In this context, this work explores exactly the aspects of security (more precisely the authentication and authorization) observed during a handover between WLANs and cellular networks (focus on the 2G and 3G).

Resumo

A comunicação sem fio é algo extremamente presente no nosso cotidiano. Dentre os padrões existentes atualmente no mercado destacam-se dois: as redes celulares e as WLAN. Esses padrões são considerados complementares a partir do momento em que um viabiliza altas taxas de transmissão e outro, grandes áreas de cobertura. Dessa forma, a integração desses padrões é algo muito atrativo. Entretanto isso não simples e apresenta muitos desafios. Um em especial diz respeito aos aspectos de segurança considerados em uma transição entre duas redes de padrões distintos (handover inter-tecnologias). Nesse contexto o presente trabalho explora exatamente os aspectos de segurança (mais precisamente a autenticação e a autorização) observados durante um handover entre WLANs e redes celulares (foco nas redes 2G e 3G).

4.1. Aspectos Introdutórios

As redes de comunicação sem fio (*wireless*) são uma realidade inequívoca nos dias de hoje. Seguindo diferentes padrões e técnicas, elas existem em quase todos os lugares por onde passamos no nosso cotidiano. São WWANs (*Wireless Wide Area Network* – Redes Sem Fio de Área Ampla), WMANs (*Wireless Metropolitan Area Networks* - Redes Sem Fio de Área Metropolitana), WLANs (*Wireless Local Area Networks* - Redes Sem Fio de Área Local) e até WPANs (*Wireless Personal Area Network* - Redes Sem Fio de Área Pessoal) espalhadas por todos os lugares. Essas redes englobam residências, ambientes de trabalho, ruas, *shoppings centers*, aeroportos, rodoviárias, estações de metrô, *cyber-cafés*, cidades e áreas de campo, enfim, quase todos os lugares habitados ou pelos quais o ser humano passa.

Do conjunto de redes sem fio existentes destacam-se dois tipos: as redes móveis celulares e as WLAN baseadas no padrão [IEEE 802.11]. Esses tipos são, de certa forma, complementares, pois quando são comparados, percebe-se que o primeiro fornece ampla área de cobertura com taxa de transmissão reduzida e o segundo viabiliza altas taxas de transmissão em pequenas áreas de cobertura.

Dessa forma, a interconexão entre essas tecnologias mostra-se algo promissor e tem sido alvo de amplos estudos na comunidade acadêmica.

Nesse contexto, também a continuidade de conexão e de serviço em redes sem fio móveis vem se tornando uma necessidade latente dos usuários, que cada vez mais exigem ubiqüidade em seus acessos a serviços considerados críticos (voz e vídeo, por exemplo).

As redes celulares tem sido uma opção importante nesse cenário, considerando sua evolução e seu recente suporte a serviços comutados por pacotes e a altas taxas de transmissão.

Por outro lado, como dito anteriormente, as redes sem fio do tipo WLAN têm sido uma alternativa de baixo custo, oferecendo taxas de transmissão bem mais elevadas, típicas desse tipo de rede.

A questão então é integrar essas duas redes, de forma que a conexão do usuário e o serviço oferecido a ele possam ter continuidade independente da rede a qual o usuário esteja enlaçado.

Para isso, questões inerentes ao controle de acesso ao meio, às arquiteturas, tecnologias e protocolos para a integração WLAN-3G vem sendo estudadas. No entanto, todos esses estudos passam por um ponto comum: o suporte à segurança da informação.

Paradoxalmente aliado a esse aspecto, as tentativas de uso de arquiteturas e tecnologias para a integração de redes focam suas atenções na redução do tempo envolvido com os processos de transferência do móvel de uma rede para outra.

Como solucionar esse problema? De um lado a necessidade de diminuir o tempo de um *handover* com o objetivo de viabilizar a manutenção de uma sessão de uma

determinada aplicação. Do outro lado a necessidade de realizar um *handover* de forma segura, pautado em processos de autenticação e autorização confiáveis e que certamente geram um consumo de tempo relevante.

São muitos os aspectos que devem ser considerados para a resolução desse binômio formado por partes aparentemente contraditórias.

Nesse contexto, focalizando especificamente esses desafios, o objetivo geral do curso é apresentar aspectos teóricos e práticos envolvidos no fornecimento de segurança para a interconexão (*handover*) entre as redes do padrão [IEEE 802.11] e as redes móveis celulares (focalizando 2ª e 3ª gerações). Além disso, também têm-se a intenção de explorar algumas formas de diminuir o custo temporal total de processos de segurança com o objetivo de minimizar o impacto sobre as sessões de aplicações em execução.

Cabe salientar que serão priorizados os problemas de segurança que envolvem os mecanismos de autenticação e autorização entre as partes envolvidas na interconexão, além de aspectos ligados ao gerenciamento de chaves criptográficas. Serão apresentados alguns protocolos e propostas aplicados a diversas situações onde a autenticação e a autorização são necessárias.

Para atingir esses objetivos, o restante do texto deste curso está organizado da seguinte forma:

- A seção 4.2 abordará assuntos ligados à garantia de confiança entre as diversas partes envolvidas em um processo de comunicação de dados baseado em redes sem fio. Serão feitas algumas considerações gerais e será descrito um modelo e níveis de confiança que devem existir para que as partes possam ser consideradas confiáveis frente às outras entidades componentes do processo.
- A seção 4.3 versará sobre os serviços de AAA (*Authentication, Authorization and Accounting* – Autenticação, Autorização e Contabilização) e focalizará dois importantes protocolos para AAA que estão relacionados ao problema abordado neste curso: o RADIUS (*Remote Authentication Dial In User Service* – Serviço de Autenticação e Contabilização Remota de Usuários) e o Diameter.
- A seção 4.4 apresentará o protocolo EAP (*Extensible Authentication Protocol* – Protocolo de Autenticação Estensível), que, atuando em nível de enlace, é a base que viabiliza os processos de autenticação que envolvem as WLANs e redes celulares de 2ª e 3ª gerações. Também serão abordados alguns de seus métodos criados posteriormente com o objetivo de resolver problemas específicos de determinados ambientes e tecnologias. Será tratado mais detalhadamente, além do próprio *framework* EAP, o ERP (*EAP Re-authentication Protocol* – Protocolo de re-autenticação EAP), pelo fato de ser um protocolo ligado diretamente ao processo de re-autenticação que é um dos focos de um processo de *handover*.
- Na seção 4.5, semelhante àquilo que foi mostrado na seção anterior, serão apresentados os principais protocolos de autenticação utilizados em redes celulares de 2ª e 3ª gerações, o EAP-SIM (*EAP Method for Global System for Mobile Communications Subscriber Identity Module* – Método EAP para

Módulo de Identidade do Assinante do Sistema Global para Comunicações Móveis) e o EAP-AKA (EAP *Method for 3rd Generation Authentication and Key Agreement* – Método EAP para Autenticação e Acordo de Chave de 3ª Geração), respectivamente.

- A seção 4.6 abordará o gerenciamento de chaves e tratará alguns aspectos gerais com a hierarquia de chaves, distribuição e reuso de material usado para a criação de chaves.
- Na seção 4.7 serão descritas as métricas, as técnicas e as fases do processo de *handover*, caracterizando cada uma delas no contexto das redes envolvidas e destacando os aspectos de segurança observados. Além disso, serão apresentados também os aspectos gerais do padrão IEEE 802.21, que foi proposto para tentar resolver (ou atenuar) o problema dos *handovers* entre redes heterogêneas.
- Na seção 4.8 serão tratados alguns protocolos de autenticação envolvidos nas operações de *handover* inter-tecnologias. Dessa forma, serão apresentados os protocolos MPA (*Media Independent Pre-Authentication* – Pré-autenticação Independente do Meio), que será priorizado tendo em vista sua importância no contexto tratado, o HOKEY (*Handover Keying* – "Chaveamento" de Handover) e o PANA (*Protocol for Carrying Authentication for Network Access* – Protocolo para Transporte de Autenticação para Acesso a Rede).
- A seção 4.9 finalizará o presente trabalho apresentando algumas conclusões

4.2. Garantia de confiança

4.2.1. Premissas e conceitos básicos

Antes de tratar especificamente do modelo adequado ao problema abordado neste trabalho, serão enunciadas algumas definições que viabilizarão um entendimento mais adequado daquilo que está sendo explorado.

A primeira e mais relevante no contexto desta seção diz respeito àquilo que vem a ser **confiança**. A definição clássica encontrada no dicionário Houaiss da Língua Portuguesa é:

> *"...crença na probidade moral, na sinceridade afetiva, nas qualidades profissionais, etc., de outrem, que torna incompatível imaginar um deslize, uma traição, uma demonstração de incompetência de sua parte; crédito, fé..."*

Percebe-se nitidamente que o escritor focaliza, em sua definição, relacionamentos humanos. Assim, apesar de colaborar para o entendimento do termo, o texto ressaltado não é totalmente adequado àquilo que está sendo explorado, afinal as entidades que fazem parte do contexto abordado neste trabalho extrapolam o ser humano, sendo consideradas, de forma resumida, sistemas complexos formados por pessoas, equipamentos (*hardware*) e programas (*software*).

Muitas outras definições existem, e as mais adequadas tratam confiança sob dois enfoques: o da esperança e o da dependência [Josang ET AL, 2007]. No primeiro caso,

confiança está ligada ao fato de uma entidade ter esperança (dar crédito), com certa probabilidade, que outra entidade aja em seu favor (da primeira). No segundo caso, confiança diz respeito às situações nas quais uma entidade está disposta, com certa probabilidade, a depender de outra para atingir algum objetivo. Os dois enfoques são muito próximos e em ambos os casos, quanto maior a probabilidade, maior será a confiança.

Outro conceito importante no contexto desta seção é o de **reputação**. O mesmo dicionário utilizado na definição passada indica:

"... conceito de que alguém ou algo goza num grupo humano..."

Como no caso passado, o escritor focaliza relacionamentos humanos. De forma mais adequada, pode-se entender reputação como o que é geralmente dito ou acreditado (em quantidade e qualidade) sobre uma entidade, ou seja, é algo derivado de um comportamento frente a outras entidades [Josang ET AL, 2007].

Confiança e reputação parecem conceitos proporcionais (boa reputação implica em maior confiabilidade, por exemplo), entretanto isto é um equívoco facilmente percebido quando examinamos as afirmações:

- "Confio em você por causa da sua boa reputação";
- "Confio em você, independente da sua má reputação";
- "Não confio em você, independente da sua boa reputação".

Na primeira afirmação, a confiança deriva diretamente da reputação, entretanto nas duas seguintes, ela deriva de outra coisa que provavelmente é mais relevante que a própria reputação (um conjunto de leis e normas, por exemplo, que previna danos causados a uma entidade - que confia - por outra entidade - em quem se confia – pode ser suficiente para alguém confiar em outrem, independente de sua reputação).

Outro conceito importante para o contexto deste trabalho é o de **modelo de confiança**. Este pode ser entendido como o arcabouço formado por entidades, seus relacionamentos e seus comportamentos, que viabiliza, para as diversas partes interessadas, confiança em níveis adequados para a consecução de seus objetivos. No item 4.2.2 será discutido um modelo de confiança adequado para o problema considerado neste trabalho.

Para que seja possível definir modelos e soluções que forneçam confiabilidade aos processos de comunicações que envolvem, em particular, usuários, redes celulares e WLANs, é necessário preocupar-se com questões do tipo [Koien & Haslestad, 2003]:

- Em que entidades deve-se confiar?
- De que forma deve-se confiar nessas entidades?
- Que tipos de características de segurança são necessárias para justificar confiança?

Essas questões podem ser respondidas de diversas formas, dependendo da situação analisada, e indicarão as premissas básicas que devem ser atendidas por um modelo de confiança que esteja em conformidade com o problema explorado.

4.2.2. Modelo e níveis de confiança

Um modelo de confiança adequado às situações tratadas neste trabalho considera a presença de três entidades básicas [ETSI TS 133 234]: A rede celular, a rede sem fio (WLAN) e o usuário. Salienta-se que para a implementação de soluções específicas, cada caso deve ser analisado separadamente e demanda a geração de modelos de confiança apropriados.

Pode-se definir mais detalhadamente as três entidades básicas da seguinte forma:

- Rede celular – integrante da parte fixa do modelo e formada por todos os componentes da rede de comunicações que provê serviços de telefonia móvel celular, podendo ser composta por uma ou mais operadoras, um ou mais domínios e uma ou mais tecnologias de acesso (GSM e UMTS, por exemplo). Também são consideradas, como parte da rede celular, as outras redes celulares que podem ser acessadas pelo usuário quando em situação de "*roamming*". Note que, apesar de não estarem sendo tratados nesta seção, também deverão existir modelos de confiança internos, entre os diversos integrantes da rede celular (Estações Rádio Base, Centros de Autenticação, Servidores diversos e etc.) para que a rede como um todo seja considerada confiável.
- WLAN – também integrante da parte fixa do modelo, essa entidade é composta por todos os elementos que se encontram "à retaguarda" dos pontos de acesso sem fio (incluindo os próprios) e que fornecem acesso público à rede do padrão [IEEE 802.11]. Tal qual na rede celular, também deverão existir modelos de confiança entre seus elementos para que a WLAN seja considerada confiável.
- Usuário - é elemento móvel do modelo que gera, dessa forma, a necessidade de *handovers*. Considera-se neste trabalho que o usuário é formado pelo próprio assinante das redes celulares e WLANs, pelo equipamento com capacidade de processamento e de comunicação (telefone, PDA, laptop e etc.) e pelos dispositivos de segurança vinculados, particularmente os módulos de identificação do usuário (SIM/USIM/UICC) ligados ao equipamento e que embutem o material criptográfico a ser utilizado na segurança dos processos. Note que para que seja possível a realização de *handovers* inter-tecnologias (usuário saindo de uma WLAN e entrando em uma rede celular, por exemplo), o equipamento com capacidade de comunicação indicado anteriormente deverá ser multi-modo, ou seja, deverá suportar comunicações na rede celular e na WLAN (simultaneamente ou não, dependendo da necessidade).

Cabe salientar que a WLAN e a rede celular podem fazer parte de uma arquitetura muito acoplada, pouco acoplada ou não acoplada, como será visto com mais detalhes na seção 4.7 deste trabalho. Cada tipo de arquitetura gerará peculiaridades à relação de confiança existente entre essas duas entidades.

A Figura 4.1 representa as entidades abordadas e as relações entre elas.

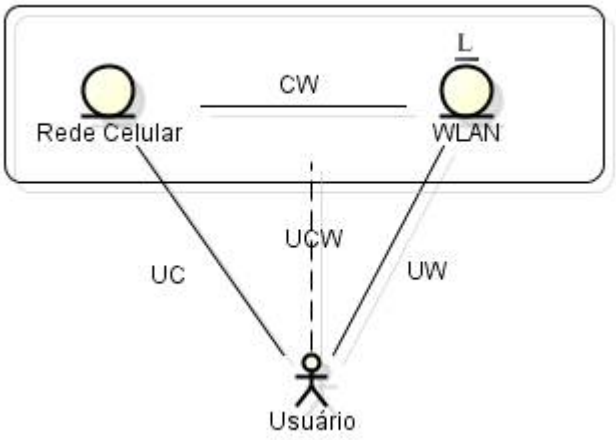


Figura 4.1 – modelo de confiança

As relações de confiança existentes entre as partes envolvidas são representadas, na figura, por CW (relação de confiança existente entre a rede celular e a WLAN), UC (usuário e rede celular), UW (usuário e WLAN) e UCW (usuário, rede celular e WLAN quando estas duas últimas estão muito acopladas). Cada uma possui suas características particulares.

A relação CW está intimamente ligada ao grau de acoplamento entre as entidades. Em uma arquitetura muito acoplada, como a WLAN está praticamente embutida da rede celular (fazer parte da mesma entidade legal), essa relação passa a ser considerada interna à rede celular e é viabilizada por meio dos modelos de confiança internos da rede celular (e WLAN). Nesta situação, as relações UC e UW podem ser resumidas à relação UCW. Já em situações de pouco ou nenhum acoplamento, a relação CW é controlada por meio de tratados formais de parceria entre rede celular e WLAN e é implementada, geralmente, por protocolos de AAA (seção 4.3).

A relação UC é regida formalmente por um acordo de fornecimento de serviços de telefonia móvel celular seguros firmado, no ato da contratação, entre o assinante e a operadora de telefonia (ou operadoras). Entretanto, quem confere confiabilidade prática à relação (particularmente aos processos de comunicação que ocorrem entre as duas partes) são os mecanismos de segurança física e lógica (neste último, ressalta-se o papel da criptografia) desencadeados por ambas as partes e que focalizam certos componentes das entidades (no usuário, o SIM/USIM/UICC, e na rede celular, o centro de autenticação, por exemplo).

A relação UW, semelhante àquilo que ocorre na UC, é criada a partir acordos entre assinantes e provedores de acesso a redes sem fio (um usuário pode ter um contrato com uma empresa que fornece acesso a uma WLAN existente em um aeroporto, por exemplo) e também é viabilizada por meio de processos e protocolos de segurança da informação. Essa relação torna-se bastante complexa a partir do momento em que acessos a WLANs públicas (que não demandam acordos prévios entre o usuário e a rede) passam a ser considerados. Na primeira situação (WLANs contratadas), deve ser papel dos provedores (inerente ao modelo de confiança) viabilizarem segurança inter-usuários, ou seja, evitar, por exemplo, que dados de um usuário possam ser capturados

indevidamente por outros usuários. Já na segunda situação (WLANs públicas), geralmente o provedor abstém-se dessa responsabilidade. Note que essa lacuna no modelo de confiança poderia ser preenchida caso fosse considerada uma relação de confiança entre usuários (UU), o que não será tratado neste trabalho.

Além dos elementos e de suas relações, também é interessante que um modelo de confiança considere os níveis de confiança tolerados entre as diversas partes. No caso em análise, o usuário e a rede celular, desde que autenticados mutuamente, devem ter total confiança um no outro (o que deve ser garantido pelo acordo firmado). Entretanto a WLAN, como única entidade que foi considerada em vários estados (WLANs públicas, contratadas, embutidas na rede celular, etc.), apresenta níveis de confiabilidade distintos. Nesse contexto, e focalizando as relações que consideram a WLAN, percebem-se três níveis de confiança distintos [ETSI TS 133 234]:

1. A WLAN pode ser totalmente não confiável para o usuário e para a rede celular;
2. Alguns elementos da WLAN são confiáveis para o usuário e para a rede celular (alguns servidores em uma arquitetura pouco acoplada, por exemplo), outros elementos não são confiáveis;
3. Todos os elementos da WLAN são totalmente confiáveis para o usuário e a rede celular.

Cada nível de confiança implicará em comportamentos específicos das diversas partes e interferirá diretamente sobre o fornecimento dos serviços para os usuários (determinados serviços que demandam muita segurança não poderão ser viabilizados para os usuários por uma WLAN caso esta seja considerada totalmente não confiável, por exemplo).

4.3.Protocolos Genéricos para AAA

A presente seção versará sobre os serviços de AAA (*Authentication, Authorization and Accounting* - Autenticação, Autorização e Contabilização) e focalizará dois importantes protocolos para AAA que estão relacionados ao problema abordado neste curso: o RADIUS (*Remote Authentication Dial In User Service* – Serviço de Autenticação e Contabilização Remota de Usuários) e o Diameter (diâmetro – passando a idéia de que é duas vezes o raio – RADIUS).

Cabe definir, antes de abordar os protocolos propriamente ditos, o que vem a ser cada um dos serviços de AAA:

- Autenticação – serviço que viabiliza a verificação da autenticidade da identidade de uma entidade, ou seja, constata que ela é quem diz ser;
- Autorização – uma vez identificada a entidade, esse serviço verifica suas prerrogativas (seu direitos em termos de acesso a serviços e recursos) e autoriza o acesso;
- Contabilização – tendo sido autenticada e autorizada, a entidade começa a utilizar o serviço solicitado; essa utilização passa a ser, então, medida ou

contabilizada (em termos de tempo de uso, por exemplo) para fins específicos (cobrança financeira por tempo de uso do serviço, por exemplo).

4.3.1. RADIUS

O RADIUS, caracterizado pela [RFC 2865], foi definido originalmente para prover AAA para sessões SLIP [RFC 1055] e PPP [RFC 1661]. Ele viabiliza uma diminuição de sobrecarga sobre o NAS (*Network Access Server* - Servidor de Acesso à Rede) a partir do momento em que reúne, sob sua responsabilidade (RADIUS), as listas de usuários autorizados e seus respectivos parâmetros de segurança (senhas e etc.) que em situações anteriores eram armazenadas pelos próprios NAS. Esta arquitetura torna possível a criação um banco de dados de usuários centralizado que permite a gerência unificada e o suporte a chamadas vindas de NAS fisicamente distribuídos.

O RADIUS está fundamentado em uma arquitetura cliente/servidor e opera na camada de aplicação utilizando, para troca de mensagens, o protocolo de transporte UDP (*User Datagram Protocol* – Protocolo de Datagramas de Usuário) [RFC 768]. De forma simplificada, o protocolo é utilizado na conferência dos parâmetros "nome de usuário" e "senha". Estas informações, uma vez disponibilizadas para o NAS, são repassadas para o servidor RADIUS (ou servidor de AAA), que verifica a veracidade das informações e libera (ou não) o acesso aos serviços solicitados pelo usuário (caso o acesso seja liberado, também é feita a contabilização do uso dos recursos).

A Figura 4.2 representa uma típica troca de mensagens de autenticação e autorização entre um cliente e um servidor RADIUS (entidades intermediárias são omitidas por questão de simplicidade).

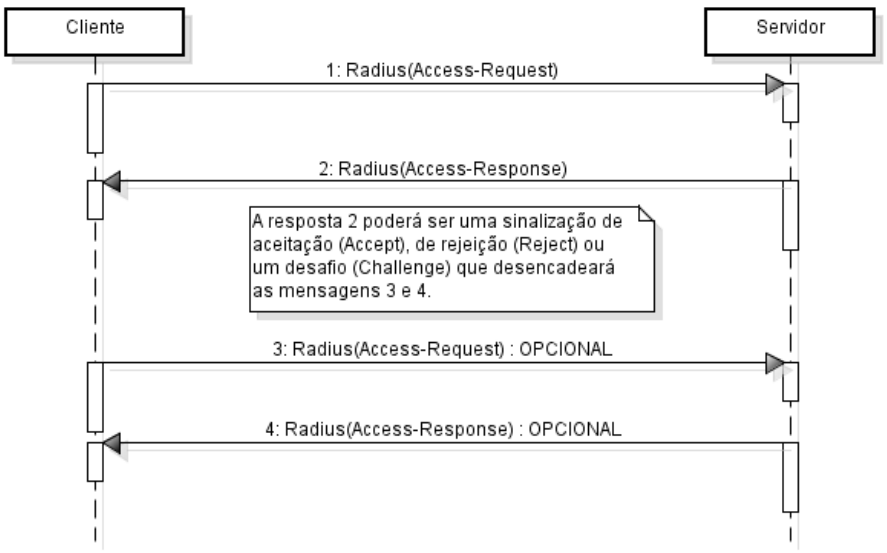


Figura 4.2 – Troca de mensagens de Autenticação e Autorização RADIUS

Quando um usuário solicita uma conexão, o NAS envia uma mensagem de pedido de acesso RADIUS ao servidor de AAA, transmitindo o nome do usuário e a senha ofuscada (um derivado da senha calculado a partir da função *hash* MD5), tipo de conexão (porta), identidade do NAS e o campo autenticador.

Em resposta, o servidor de AAA usa a fonte do pacote, identidade do NAS e o campo autenticador para determinar se o NAS pode enviar requisições de autenticação. Nesse caso, o servidor de AAA tenta achar o nome do usuário em seu banco de dados e aplica a senha ofuscada e outros atributos constantes no pedido de acesso para decidir se deve ser concedido acesso a este usuário.

Dependendo do método de autenticação utilizado, o Servidor de AAA pode responder à solicitação devolvendo uma mensagem-desafio (*access-challenge* – desafio de acesso) que contem, como parâmetro, um valor aleatório. O NAS retransmite o desafio ao usuário remoto que deverá, por sua vez, responder com o valor correto para provar sua identidade (a resposta pode ser, por exemplo, calculada por meio da codificação do desafio utilizando um derivado da senha). Essa resposta é encaminhada para o NAS que retransmite ao servidor de AAA dentro de outra mensagem de solicitação de acesso RADIUS.

Se o servidor de AAA verificar que o usuário é autêntico, ele o autoriza a usar o serviço solicitado (acesso a um recurso de rede, por exemplo) e devolve uma mensagem de permissão de acesso RADIUS. Caso a autenticidade na seja verificada, o Servidor de AAA devolve uma mensagem de rejeição de acesso RADIUS e o NAS desconecta o usuário.

Quando uma mensagem de permissão de acesso é recebida, instantaneamente é iniciado o processo de contabilização RADIUS (não representado na Figura 4.2). Para isso o NAS envia uma mensagem de requisição de contabilização RADIUS para o Servidor de AAA. O Servidor, tendo recebido a requisição, passa a realizar um registro contábil vinculado ao usuário recém autorizado. Ao mesmo tempo, o NAS ativa a sessão desse usuário. Terminando a sessão, uma nova mensagem de requisição de contabilização do RADIUS é enviada pelo NAS para o servidor, o que sinaliza que deverão ser gravadas, nos registros contábeis do servidor, a razão da desconexão e a duração da sessão encerrada. Salienta-se que poderão ser solicitadas alterações antes do término da sessão. Essas solicitações também gerarão mensagens de requisição e de resposta de contabilização.

As mensagens do RADIUS são transportadas por meio de datagramas UDP e possuem os seguintes campos, conforme Figura 4.3: código (tipo da mensagem), identificador (geralmente um número seqüencial), tamanho (quantidade de octetos), autenticador (desafio do servidor RADIUS – *access-challenge* - ou resposta a esse desafio) e atributos (opcionalmente requeridos para tipos específicos de serviços).

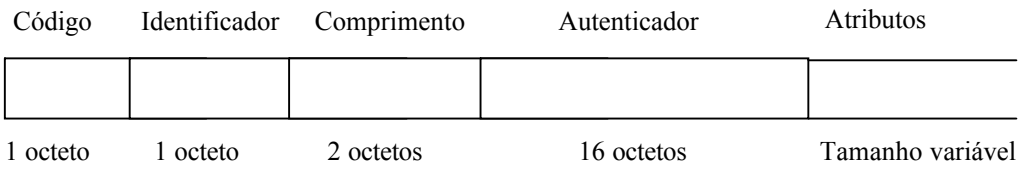


Figura 4.3 – Mensagem RADIUS – formato genérico

Salienta-se que o propósito do campo autenticador é prover segurança às transmissões. Ele pode ser de dois tipos: requisição ou resposta. O autenticador-requisição segue do usuário para o servidor AAA (passando pelo NAS) e contém um valor totalmente aleatório. Já o autenticador-resposta segue no sentido oposto e contém um valor *hash* MD5 calculado a partir do pacote de requisição RADIUS recebido seguido dos atributos de resposta e do segredo compartilhado entre servidor AAA e usuário.

O campo autenticador também fornece ajuda para o servidor AAA descobrir falsificação de respostas RADIUS, além de ser usado para obscurecer a senha do usuário, inibindo revelação ao NAS ou qualquer outra entidade intermediária que poderia bisbilhotar as mensagens RADIUS.

O uso do campo autenticador intimida ataques passivos, mas um intruso que consiga capturar as mensagens de requisição de acesso RADIUS e as mensagens de resposta de acesso pode executar um ataque de dicionário e descobrir a senha. A [RFC 3580] documenta diversas vulnerabilidades, bem como diretrizes de uso para reduzir riscos na utilização do RADIUS.

A filosofia de transmissão, pelo RADIUS, de pares atributo-valor (AVP – *Attribute-Value Pair*) facilita o uso do protocolo com uma grande variedade de tecnologias de acesso e métodos de autenticação. O padrão original definiu vários pares atributo-valor comuns, como usuário-nome, usuário-senha, NAS-IP-endereço, porta NAS e tipo de serviço. Entretanto novos pares atributo-valor podem ser definidos para que sejam criadas extensões proprietárias, o que pode ser observado, por exemplo, na [RFC 2548], que define os pares atributo-valor da Microsoft associados ao protocolo MS-CHAP v1 [RFC 2433].

Além disso, considerando o contexto específico deste trabalho, muitos pares atributo-valor padrão foram definidos para dar suporte ao EAP (seção 4.4.1 deste trabalho), como pode ser constatado na [RFC3579].

4.3.2. Diameter

O Diameter, tal qual o RADIUS, foi criado para prover os serviços de AAA para viabilizar, por exemplo, o controle de acesso a serviços e recursos de redes. Ele está definido na [RFC 3588] e pode ser utilizado tanto em redes locais quanto em ambientes totalmente distribuídos (situações onde o cliente está em situação de *roaming*, por exemplo).

O protocolo é considerado por muitos como o sucessor do RADIUS (apesar de não serem totalmente compatíveis) e foi criado para atender a demandas, no contexto de AAA, que não eram mais atendidas pelo seu antecessor. A tabela a seguir mostra algumas das principais diferenças entre os dois protocolos:

Tabela 4.1– Comparação Diameter X RADIUS (adaptado de [Liu ET AL, 2006]).

Característica analisada	Diameter	RADIUS
Protocolo de transporte	Orientado à conexão (TCP e SCTP)	Não orientado à conexão (UDP)
Segurança	Ponto-a-ponto e fim-a-fim	Ponto-a-ponto
Suporte a agentes	Agentes de retransmissão (*relay*), de procuração (*proxy*), de redirecionamento (*redirect)* e de tradução (*translation*)	Não suporta
Capacidade de negociação	Negocia aplicações suportadas e níveis de segurança	Não suporta
Descobrimento de parceiro	Configuração estática e busca dinâmica	Configuração estática
Mensagem iniciada pelo servidor	Suportada (por exemplo, mensagens de re-autenticação e de término da sessão)	Não suporta
Tamanho máximo dos dados de atributos	16.777.215 octetos	255 octetos

As bases que fundamentaram a criação do Diameter podem ser encontradas na [RFC 2989] que cataloga uma série de requisitos que devem ser atendidos por protocolos de AAA de uma forma geral. Esses requisitos incluem, por exemplo, capacidade de controle de falhas e mecanismos de auditoria que não foram viabilizados por meio da criação do RADIUS mas que foram considerados para o desenvolvimento do Diameter.

O protocolo em si aproveita as idéias fundamentais do RADIUS e acrescenta uma série de mensagens e procedimentos genéricos que permitem o atendimento à maior parte dos requisitos desejados. Uma das grandes evoluções observadas no Diameter está ligada à capacidade de viabilizar interfaces que sejam compatíveis com várias aplicações que necessitem de AAA (que já existem ou que venham a existir). Isso é possível devido à criação de aplicações Diameter que funcionam sobre o protocolo base de AAA e sob as aplicações específicas (camada superior).

O Diameter possui uma arquitetura ponto-a-ponto, o que permite o encadeamento de vários nós para o fornecimento de serviços de AAA. Cada nó (o termo usado é nó Diameter) pode agir, dependendo do contexto, como cliente, servidor ou agente de AAA. Os nós com os dois primeiros papéis agem como seus correspondentes no protocolo antecessor, entretanto último papel (agente) tem características diferentes e pode assumir quatro diferentes configurações:

- retransmissão (*relay*) – agente que recebe requisições e as retransmite para outros nós Diameter;
- procuração (*proxy*) – agente semelhante ao anterior, entretanto, ao invés de retransmitir as próprias requisições recebidas, cria novas mensagens derivadas da requisição e de políticas (controle de admissão, utilização de recursos, etc.) definidas previamente;
- redirecionamento (*redirect*) – parecido com o primeiro, entretanto não retransmite a mensagem recebida, ao invés disso, responde ao remetente com informações suficientes para que este transmita sua mensagem diretamente para o destinatário adequado;
- tradução (*translation*) – agente utilizado na interface com outros protocolos (Diameter-RADIUS, por exemplo).

As mensagens Diameter são síncronas, ou seja, a cada mensagem enviada por um nó deverá ser criada uma correspondente resposta. O formato geral de um pacote Diameter pode ser observado na Figura 4.4:

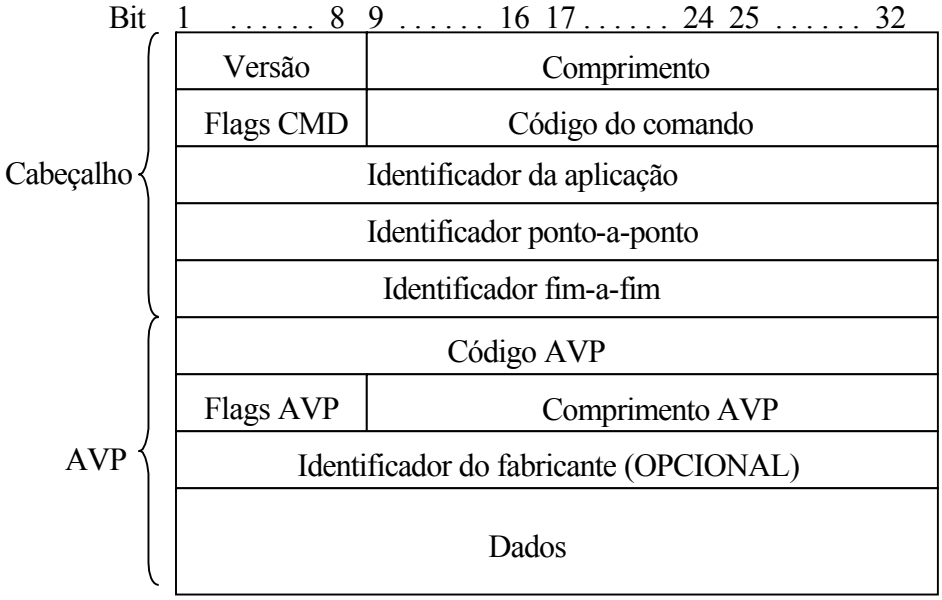


Figura 4.4 – Mensagem Diameter – formato genérico

Os campos são assim definidos:

- Versão – 1 octeto – versão do protocolo Diameter;
- Comprimento – 3 octetos – comprimento de toda a mensagem, em número de octetos, incluindo o próprio tamanho do cabeçalho;

- *Flags* CMD – 1 octeto – 8 bits de controle do comando que podem sinalizar se a mensagem é uma requisição ou uma resposta, se pode ser repassada ou não, se é uma mensagem de erro ou não, etc.;
- Código do comando – 3 octetos – comando associado à mensagem;
- Identificador da aplicação – 4 octetos – identifica a aplicação para qual a mensagem está sendo direcionada;
- Identificador ponto-a-ponto – 4 octetos – auxilia na conferência do sincronismo das mensagens entre os nós (uma resposta deve ter o mesmo identificador que a requisição que a originou, esse identificador, por sua vez, não pode fazer parte de outras respostas ou requisições que transitam entre esses nós);
- Identificador fim-a-fim – 4 octetos – usado para a detecção de mensagens duplicadas entre servidor e cliente (agentes intermediários não podem alterar esse valor);
- Código AVP – 4 octetos – código do par atributo-valor (os códigos de 1 a 255 são reservados para manter certa compatibilidade com o RADIUS);
- *Flags* AVP – 1 octeto – informa ao recebedor como o atributo deve ser manipulado (se há necessidade de cifração fim-a-fim, se é um dado que contem informações do fabricante, etc.);
- Comprimento AVP – 3 octetos – indica o tamanho do par atributo-valor, em número de octetos, incluindo o cabeçalho do AVP (código, *flags*, etc.);
- Identificador do fabricante (OPCIONAL) – 4 octetos – valor que identifica um fabricante específico (necessidade criada pelo fato do Diameter permitir que fabricantes criem seus próprios tipos de AVP, independentes dos definidos pelo próprio padrão);
- Dados – tamanho variável – zero ou mais octetos que contêm as informações específicas do atributo considerado.

Antes de qualquer transmissão ser feita, há a necessidade de se configurar, no NAS, a localização do servidor AAA que deve ser utilizado. Essa configuração pode ser feita manualmente ou pode ser auxiliada por servidores que contêm informações dos diversos nós Diameter existentes em um dado ambiente (domínio, etc.). Esse segundo tipo de configuração é chamado de “descoberta do par”, pois ao ser gerada a requisição, a busca pelo nó que poderá atendê-la da melhor forma é feita automaticamente.

Tendo sido feito o direcionamento coerente (par direcionado para par adequado), é estabelecida, então, a conexão em nível de transporte entre os pares, que representa uma ligação entre dois nós Diameter. Essa ligação, como visto na

Tabela 4.1, deve ser baseada em TCP ou SCTP, que são protocolos de transporte mais confiáveis que o UDP, e viabiliza a troca de mensagens.

Além das conexões, o Diameter estabelece sessões entre nós. Diferente das conexões, as sessões são ligações lógicas entre clientes (usuários) e servidores, são

consideradas em nível de aplicação e podem embutir várias conexões. Exemplificando: em sua situação hipotética na qual o cliente esteja separado do servidor por meio de um agente de retransmissão, este agente fará conexões (uma com o cliente e outra com o servidor) que pertencerão à única sessão criada entre o cliente e o servidor.

A troca de mensagens Diameter, como é possível inferir da complexidade da arquitetura até agora vista, não é tão simples quanto a equivalente no RADIUS. Por esse motivo, a [RFC 3588] define uma série de máquinas de estado que representam os procedimentos e trocas de mensagens que devem ser desencadeados para que os serviços de AAA sejam fornecidos a contento.

De forma simplificada, um fluxo de mensagens entre dois nós Diameter quaisquer (cliente-servidor, cliente-agente, agente-agente e agente-servidor) pode ser representado pela Figura 4.5.

Figura 4.5 – Trocas de mensagens - Diameter

Como pode ser visto, antes de focalizar um serviço de AAA específico, o nó inicial (A) solicita informações sobre as possibilidades do seu parceiro (B). Tendo recebido essa informação, os nós A e B começam a troca de mensagens que realmente viabilizarão o serviço desejado. Após terem sido atingidos os objetivos (ou em situações de encerramento precoce planejado), as partes finalizam a conexão (dependendo do contexto, tanto A quanto B podem solicitar esta finalização).

4.4. Protocolos de Autenticação utilizados nas redes WLAN

Nesta seção será apresentado o protocolo EAP, que é a base que viabiliza os processos de autenticação que envolvem as WLAN. Também serão abordados alguns de seus métodos criados posteriormente com o objetivo de resolver problemas específicos de determinados ambientes e tecnologias. Será tratado mais detalhadamente, além do próprio *framework* EAP, o ERP, pelo fato de ser um protocolo ligado diretamente ao processo de re-autenticação que é um dos focos de um processo de *handover*.

4.4.1. EAP

O EAP (*Extensible Authentication Protocol* – Protocolo de Autenticação Extensível) é um framework que suporta vários métodos de autenticação e que tipicamente é executado diretamente na camada de enlace, não requerendo, por exemplo, conectividade IP. Ele foi projetado para autenticações de acesso a redes e, apesar de fornecer funções que servem de base para diversos métodos de autenticação, não pode ser considerado um mecanismo de autenticação específico.

O EAP é bastante flexível, podendo ser usado em enlaces dedicados ou em circuitos comutados, da mesma forma funcionando em redes com ou sem fio. O processo de encapsulamento para meios com fio (os vinculados especificamente ao padrão IEEE-802) é descrito no padrão [IEEE-802.1X]. Já no caso das redes IEEE sem fio, a descrição do processo encontra-se no padrão [IEEE-802.11-2007], que embute a emenda IEEE-802.11i.

Diferente de muitos protocolos de autenticação existentes, as autenticações baseadas em EAP sempre são iniciadas pelo autenticador (quem autentica), e não pelo par ou suplicante (quem deseja autenticar-se). Além disso, o framework viabiliza a utilização de servidores de autenticação remotos. Em situações nas quais um autenticador não suporta algum dos métodos de autenticação escolhidos para o processo, este poderá delegar ao servidor de autenticação remoto a execução do método. Nesse caso, o autenticador servirá apenas de passagem para o fluxo de informações entre o servidor e o suplicante [RFC3748]. Cabe salientar que essa última situação é a que ocorre mais freqüentemente quando se utiliza o EAP.

Apesar de fornecer suporte à retransmissão de pacotes, o EAP não suporta fragmentação (alguns métodos vinculados, como o EAP-TLS, por exemplo, fornecem esse suporte) nem ordenação de pacotes (utiliza esse serviço de camadas inferiores).

Os pacotes EAP têm, de uma forma geral, o formato observado na Figura 4.6, contendo os seguintes componentes:

- Código – ocupa 1 octeto e identifica o tipo de pacote EAP, que pode ser *Request* (requisição), *Response* (resposta), *Success* (sucesso) e *Failure* (falha);
- Identificador – ocupa 1 octeto e tem a finalidade de sincronizar respostas com suas respectivas requisições;
- Comprimento – ocupa 2 octetos e indica o tamanho (comprimento), em número de octetos, do pacote EAP como um todo (incluindo o "cabeçalho" formado pelo código, identificador e comprimento);

- Dados – tem seu formato definido com base no código do pacote e pode ocupar 0 ou mais octetos.

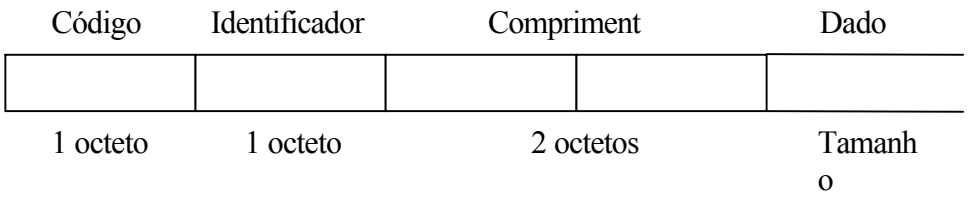


Figura 4.6 – Pacote EAP – formato genérico

Conforme aquilo que é visto na Figura 4.7, o processo de troca de mensagens de autenticação do EAP ocorre seguindo os seguintes passos:

1. O autenticador envia uma solicitação de autenticação de determinado tipo (identidade, desafio-MD5, etc.) para o suplicante;
2. O suplicante responde com um pacote adequado ao tipo de autenticação solicitado;
3. O autenticador envia um pacote de requisição adicional e o suplicante responde com pacotes adequados; esse passo deve ser repetido quantas vezes forem necessárias; salienta-se que uma nova requisição somente poderá ser feita após a resposta adequada ter sido recebida, entretanto, caso não sejam recebidas respostas, poderão ser feitas retransmissões de requisições já enviadas; caso proceda-se dessa forma e, mesmo assim, as respostas continuem sem chegar, o autenticador deverá finalizar a troca de mensagens;
4. Após o número adequado de mensagens trocadas, o autenticador concluirá sobre o sucesso ou a falha da autenticação e, em ambos os casos, enviará para o suplicante o pacote final sinalizando o resultado (sucesso ou falha).

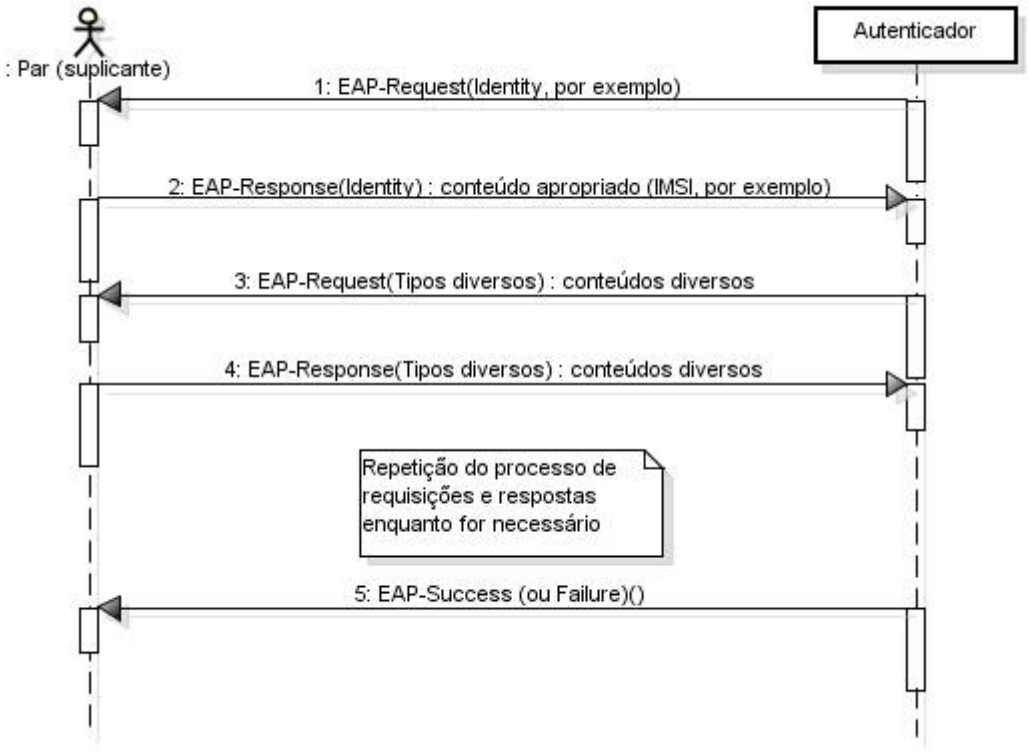


Figura 4.7 – EAP – troca de mensagens genéricas

A hierarquia de chaves do EAP (maior detalhamento na seção4.6) está baseada na criação inicial de duas chaves: a MSK (*Master Session Key* - Chave Mestra da Sessão) e a EMSK (*Extended Master Session Key* - Chave Mestra Estendida da Sessão). Após a troca de mensagens entre o servidor e o suplicante (tendo com intermediário o autenticador) e o sucesso do processo de autenticação, o servidor remete a MSK para o autenticador (utilizando protocolos de AAA como os enunciados na seção 4.3) que, por sua vez, estabelece, em conjunto com o suplicante e utilizando a MSK, as TSKs (*Transient Session Keys* - Chaves Transientes de Sessão), que são utilizadas na proteção dos pacotes de dados [RFC 5296].

Para viabilizar suas funcionalidades, o EAP obedece a um modelo conceitual composto pelas camadas definidas a seguir [RFC 3748]:

- Camada do método EAP – responsável pela execução dos algoritmos de autenticação e pela fragmentação dos dados, visto que esta última tarefa não é suportada pelo framework EAP;
- Camada do suplicante ou do autenticador (conforme o local onde esteja sendo executado o determinado passo do protocolo) – responsável por diferenciar a execução das funções do método EAP (camada acima) entre o suplicante e o autenticador;
- Camada EAP – recebe e transmite os pacotes EAP por meio da camada inferior, implementa a detecção de pacotes duplicados e o processo de retransmissão de

pacotes, além de demultiplexar os pacotes EAP para a camada superior (do suplicante ou do autenticador, conforme o caso);

- Camada inferior – tem a função de receber e transmitir os *frames* EAP entre o suplicante e o autenticador; são exemplos de camadas inferiores o PPP [RFC 1661], o UDP [RFC 768], o TCP [RFC 793], o IKEv2 [RFC 4306], o [IEEE 802.1X] e o [IEEE 802.11], dentre outros; o EAP assume que a camada inferior apresenta características como transporte não confiável (solicitação de retransmissões é papel do EAP), detecção de erros, garantia de ordenação de pacotes, possibilidade de duplicação de pacotes, dentre outras.

Na seção a seguir serão descritos alguns métodos de autenticação baseados no EAP e que podem ser utilizados em redes do padrão [IEEE 802.11]. Esses métodos (e quaisquer outros que venham a ser utilizados com este padrão) devem atender obrigatoriamente a alguns requisitos para que sejam considerados seguros [RFC4017]:

1. Capacidade de geração de material para chaves simétricas;
2. Utilização de chaves fortes (mínimo de 128 bits);
3. Suporte a autenticação mútua;
4. Equivalência de estado compartilhado (estados equivalentes no autenticador e no suplicante);
5. Resistência a ataques de dicionário;
6. Proteção contra ataques MITM (*man-in-the-middle* – homem no meio) [Schneier, 2006];
7. Negociação protegida de suítes criptográficas.

Além desses, há alguns outros requisitos recomendados e características opcionais que devem ser levados em consideração para os métodos EAP a serem utilizados em WLAN: fragmentação, ocultação da identidade dos usuários finais, canal de ligação, re-conexão rápida, etc.

4.4.2. ERP

O ERP (*EAP Re-authentication Protocol* – Protocolo de Re-autenticação EAP) [RFC 5296] será abordado com um nível de detalhamento maior que os demais métodos tendo em vista a sua importância no contexto da diminuição do atraso gerado nos processos de autenticação.

Ele foi criado para minimizar os problemas de re-autenticação para a passagem de suplicantes (para o caso explorado neste curso, equipamentos móveis) entre autenticadores ou para a extensão da autenticação entre um suplicante e um autenticador específico.

Nos métodos que implementam o EAP, quando um suplicante deseja autenticar-se para um novo autenticador, geralmente é executado um novo processo de autenticação EAP completo (há métodos, como o EAP-SIM, o EAP-AKA e o EAP-TTLS que são exceções a essa regra), independente do fato do suplicante já estar autenticado em outro ambiente e haver material para a criação de chaves cuja validade ainda não expirou. Esse novo processo de autenticação envolve uma nova troca de mensagens entre as partes envolvidas (como visto na seção 4.4.1), o que gera um atraso bastante relevante ao processo de passagem (*handover*) do suplicante de um autenticador para outro, quer no contexto da necessidade de aumento do tempo de processamento, quer no tocante ao tempo necessário para o trânsito das novas mensagens [RFC 5169].

Percebendo esse problema, alguns esforços anteriores ao ERP foram feitos no sentido de reduzir esse atraso, entretanto os mecanismos paliativos criados incorriam em problemas relacionados, por exemplo, à dependência de métodos EAP específicos ou à impossibilidade de re-autenticação entre autenticadores diferentes. Dessa forma, permanecia uma lacuna a ser solucionada por um protocolo genérico que pudesse ser executado independentemente do método EAP utilizado.

Nesse contexto surge o ERP, cuja idéia, apesar de poder ser utilizada no âmbito de *handovers* inter-domínios e inter-tecnologias, focaliza as re-autenticações dentro de um mesmo domínio administrativo (*handover* intra-domínio).

Os principais objetivos a serem atingidos pela implementação do ERP são [RFC 5169]:

- Execução de operações com baixa latência (ou pelo menos com latência inferior à observada em um processo de autenticação EAP completo);
- Independência relativa das camadas inferiores, o que viabiliza a execução do protocolo sobre qualquer tecnologia de acesso;
- Independência e compatibilidade com os métodos EAP utilizados para o processo de autenticação, permitindo o uso do ERP vinculado a quaisquer desses métodos;
- Compatibilidade com protocolos de AAA, particularmente com o RADIUS e o Diameter (seção 4.3);
- Agilidade criptográfica, entendida nesse contexto como a capacidade de negociação dinâmica de parâmetros criptográficos sem afetar o desempenho global do processo;
- Minimização das mudanças necessárias nas diversas entidades partícepes dos processos de re-autenticação (suplicante, autenticador e servidor EAP), desde que o desempenho geral do processo não seja afetado.

Além dos descritos acima, especificamente no contexto da segurança, também devem ser seguidos os preceitos definidos para o gerenciamento de chaves utilizadas nos processos de AAA (seção 4.6).

É válido ressaltar que, apesar da independência relativa das camadas inferiores ser um dos objetivos citados, algumas especificações de camadas inferiores, como o [IEEE 802.1X] e o IKEv2 [RFC 4306], por exemplo, devem ser revisadas para permitir o funcionamento coerente do ERP.

O funcionamento do ERP permite que um suplicante autentique-se para um servidor de re-autenticação EAP, e vice-versa, utilizando material criptográfico gerado a partir de execuções prévias de autenticações EAP, não sendo possível qualquer processo de re-autenticação sem uma autenticação completa anterior. No melhor caso, será necessária apenas uma viagem de ida e volta (*round trip*) entre o suplicante e o servidor de re-autenticação EAP. Tal qual nessas execuções prévias, no ERP, geralmente, o autenticador serve apenas de intermediário entre o suplicante e o servidor.

A Figura 4.8 detalha uma típica troca de mensagens ERP.

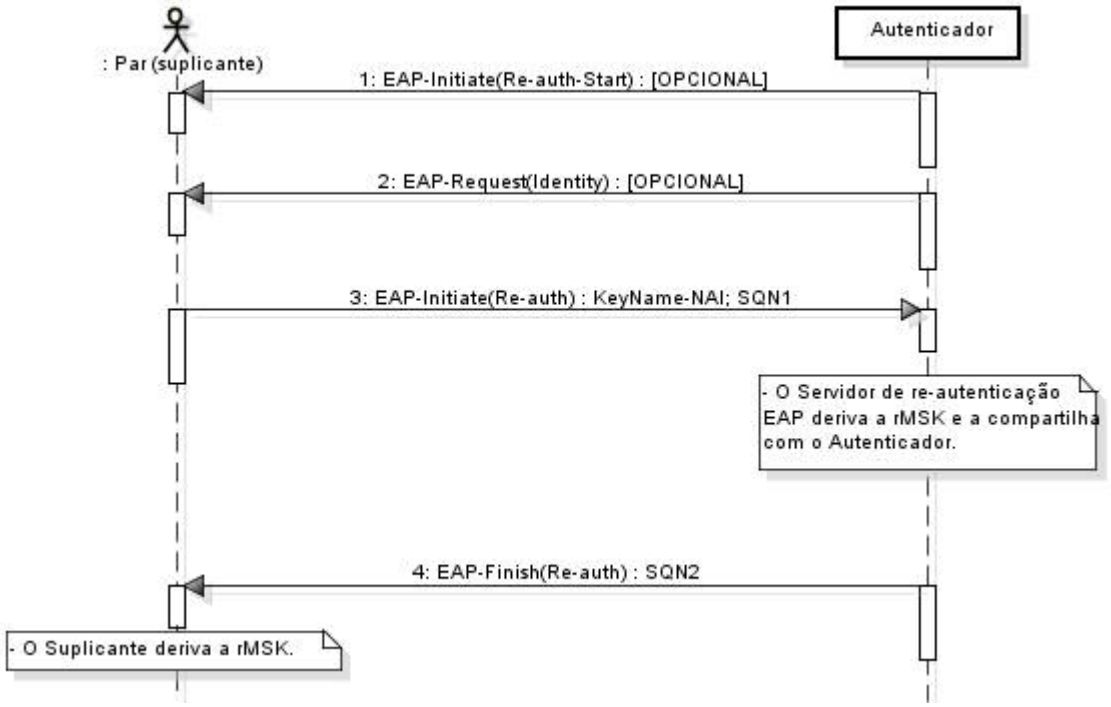


Figura 4.8 – ERP – troca de mensagens genéricas

As quatro situações a seguir são possíveis, e para cada uma delas haverá uma troca de mensagens diferente:

1. O processo é iniciado pelo suplicante (caso normal para o ERP) e o autenticador suporta o ERP e possui material não expirado para criação de chaves:
    - O suplicante, em situações normais, é o responsável por iniciar a troca de mensagens ERP utilizando um novo código EAP, o "EAP-Initiate" (este código e o "EAP-Finish" foram criados especificamente para re-autenticações EAP) – passo 3 da figura. Nesta mensagem, seguem os parâmetros "KeyName-NAI", que identifica o domínio do servidor de re-autenticação EAP e a "rIK" (chave de integridade da re-autenticação) usada para proteger as mensagens de re-autenticação, e um número

seqüencial usado na proteção contra ataques de repetição (*Replay Attacks*).

- O autenticador, tendo recebido a solicitação, encaminha a mensagem ao servidor de re-autenticação apropriado que foi deduzido a partir do "KeyName-NAI". Esse servidor deriva, então, a "rMSK" (Chave Mestra da Sessão de Re-autenticação) a partir da "rRK" (Chave Raiz de Re-autenticação, que foi deduzida a partir da EMSK criada por ocasião do processo de autenticação EAP completo prévio) e do número seqüencial recebido. Após isso, o servidor envia uma mensagem "EAP-Finish" para o autenticador o qual a repassa para o suplicante – passo 4 da figura. Nessa mensagem segue, como parâmetro, o número seqüencial recebido e incrementado pelo servidor. Seguirá também, apenas do servidor até o autenticador, a "rMSK" derivada no servidor.
- Tendo recebido a mensagem do autenticador, o suplicante deriva a "rMSK" a partir da "rRK" (que já possui) e do número seqüencial recebido.
- Salienta-se que as comunicações ocorridas entre o autenticador e o servidor são viabilizadas por meio de protocolos de AAA (seção 4.3) e são omitidas da Figura 4.8 por questões de simplicidade.

2. O processo é iniciado pelo suplicante (caso normal para o ERP) e o autenticador não suporta o ERP ou não possui material não expirado para criação de chaves:
   - O suplicante envia um "EAP-Initiate" – passo 3 da figura – para o autenticador;
   - O autenticador despreza a mensagem do suplicante por não reconhecê-la;
   - O suplicante permanece insistindo no passo 3 da figura até receber um "EAP-Request" do autenticador, o que faz com que o suplicante conclua que o autenticador não suporta o ERP;
   - Desse ponto em diante, tudo ocorrerá como visto na Figura 4.7 – EAP – troca de mensagens genéricas.
3. O processo é iniciado pelo autenticador e o suplicante suporta o ERP e possui material não expirado para criação de chaves:
   - O autenticador envia o "EAP-Initiate" solicitando ao suplicante o início da re-autenticação – passo 1 da figura;
   - O suplicante responde com outro "EAP-Initiate" – passo 3 da figura – e o processo prossegue seguindo os passos da situação 1;
   - Salienta-se que, caso o suplicante demore a responder à solicitação do autenticador, este poderá inferir, de forma equivocada, que o suplicante não suporta o ERP. Caso isso ocorra, o autenticador enviará um "EAP-Request" para o suplicante (passo 2 da figura) que, por sua vez, deverá ignorar essa requisição do autenticador e prosseguir com o passo 3 da figura. O autenticador, tão logo receba o "EAP-Initiate" atrasado do

suplicante, deverá desprezar o "EAP-Request" enviado por ele mesmo e dar prosseguimento ao processo de re-autenticação.

4. O processo é iniciado pelo autenticador e o suplicante não suporta o ERP ou não possui material não expirado para criação de chaves:
   - o O autenticador envia o "EAP-Initiate" solicitando ao suplicante o início da re-autenticação – passo 1 da figura;
   - o O suplicante ignora e descarta as mensagens recebidas por não reconhecê-las;
   - o Após um número configurável de tentativas sem sucesso, o autenticador envia um "EAP-Request" para o suplicante solicitando o início de um processo de autenticação EAP completo – passo 2 da figura;
   - o Desse ponto em diante, tudo ocorrerá como visto na Figura 4.7 – EAP – troca de mensagens genéricas.

4.4.3. Outros métodos EAP

Existem vários outros métodos EAP definidos para a solução de problemas específicos e para atender a demandas particulares de certos fabricantes. Pode-se citar, dentre vários outros, os seguintes:

1. EAP-TLS [RFC 5216]

Esse método considera a utilização do TLS (*Transport Layer Security* – Segurança da Camada de Transporte) [RFC 5246] vinculado ao funcionamento do *framework* EAP representa uma solução bastante segura, apesar de ser de difícil implementação. O TLS pode ser usado, por exemplo, para promover a autenticação mútua dos elementos que participam de uma troca de mensagens EAP. O protocolo oferece um túnel fim-a-fim criptografado para a transferência de dados, inclusive material utilizado na derivação de chaves, baseado na utilização de certificados digitais e na existência de uma ICP (infra-estrutura de chaves públicas).

2. EAP-TTLS [RFC 5281]

É um método EAP que encapsula uma sessão TLS que viabiliza, na fase de aperto de mãos (*handshake*) a autenticação do servidor para o cliente (ou a autenticação mútua) e a geração de material para a criação de chaves criptográficas. Essas chaves serão utilizadas posteriormente, na fase de troca de dados, para o fechamento de um túnel criptográfico que proverá segurança às informações em trânsito entre as partes. Uma vez fechado o túnel seguro, uma nova autenticação será feita (cliente para o servidor ou autenticação mútua) de forma sigilosa utilizando o próprio mecanismo de autenticação do EAP (é possível utilizar-se outros métodos de autenticação).

Uma característica especialmente interessante deste método é o fato dele prever a possibilidade de recomeço de uma sessão já encerrada, o que diminui significativamente o tempo necessário para o processo de autenticação e geralmente é empregado quando um cliente necessita realizar uma re-autenticação para um ponto de acesso de uma rede sem fio.

3. LEAP

O LEAP (*Lightweight* EAP – EAP Leve) consiste em um método EAP proprietário desenvolvido pela Cisco. Por ser um padrão fechado, a sua escolha como método EAP deve ser feita com cautela, pois o seu uso está relacionado ao tipo de equipamento utilizado, ou seja, a utilização do LEAP implica na necessidade de que os equipamentos da infra-estrutura (placas de rede e *Access Points*, por exemplo) sejam compatíveis com o mesmo. O LEAP foi desenvolvido para funcionar no topo da camada da infra-estrutura de autenticação do modelo [IEEE 802.1X] e viabiliza autenticação mútua e autenticação baseado no equipamento.

4. PEAP

Desenvolvido pela Microsoft, O PEAP (*Protected* EAP – EAP Protegido) consiste em um método de autenticação com dois estágios. O primeiro estágio estabelece uma sessão TLS para o servidor e permite que o cliente autentique o servidor usando o certificado digital do servidor. O segundo estágio requer um segundo método EAP encapsulado na sessão PEAP para autenticar o cliente a um servidor RADIUS. Isso permite que o PEAP use uma variedade de métodos de autenticação de clientes, incluindo o uso de senhas no protocolo MS-CHAP v2 [RFC 2759] e certificados usando o EAP-TLS encapsulado no PEAP.

4.5. Principais Protocolos de Autenticação utilizados nas redes móveis celulares

Semelhante àquilo que foi mostrado na seção anterior, aqui serão apresentados os principais protocolos de autenticação utilizados em redes celulares de 2ª e 3ª gerações, o EAP-SIM e o EAP-AKA, respectivamente.

4.5.1. Redes 2G - EAP-SIM

O EAP-SIM (EAP *Method for Global System for Mobile Communications Subscriber Identity Module* – Método EAP para Módulo de Identidade do Assinante do Sistema Global para Comunicações Móveis) [RFC4186] foi desenvolvido pelo 3GPP (*3rd Generation Mobile System* – Sistema Móvel de Terceira Geração) e é usado para autenticação e distribuição de chaves de sessão utilizando o SIM do GSM (*Global System for Mobile Communications* - Sistema Global para Comunicações Móveis).

A autenticação na rede GSM é baseada em mecanismos de Desafio-Resposta (*Challenge-Response*) e utiliza os algoritmos criptográficos A3 e A8, cujas seguranças interferem diretamente na confiabilidade do próprio EAP-SIM.

A Figura 4.9 retrata um processo de autenticação completa executado no contexto do EAP-SIM. Cabe salientar que na figura estão omitidos, por questão de simplicidade, o servidor EAP e outras entidades da arquitetura GSM que serão utilizadas na autenticação (a comunicação entre o Autenticador e as demais entidades é feita por meio de protocolos AAA). Dessa forma o Autenticador, que na maioria das vezes serve apenas como passagem para as mensagens de autenticação, está aglutinando as funções do Servidor EAP e das demais entidades.

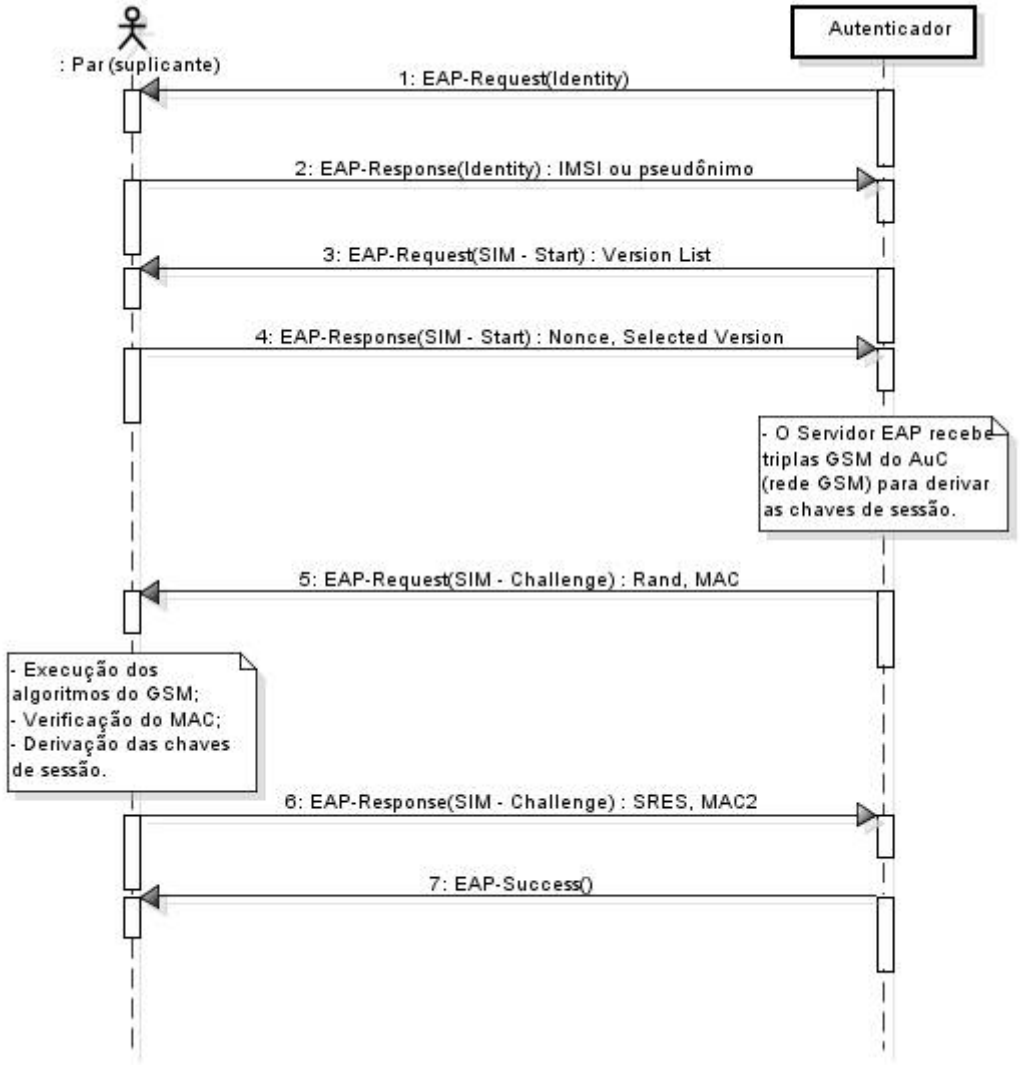


Figura 4.9 – processo de autenticação completa do EAP-SIM

Em uma primeira mensagem, o Autenticador solicita a identidade do Suplicante. Este, por sua vez, responde com parâmetros vinculados ao IMSI (*International Mobile Subscriber Identity* – Identidade Internacional do Assinante Móvel) ou, caso deseje-se preservar a identidade do assinante, com um pseudônimo temporário. Após isso, o Suplicante recebe do Autenticador a solicitação de que as operações em seu SIM sejam iniciadas. Vinculada a essa solicitação, segue a lista de versões do EAP-SIM suportadas pelo Servidor EAP (representado, na figura, pelo Autenticador). O Suplicante responde com a versão EAP-SIM selecionada e com um valor aleatório (Nonce) que será utilizado em operações futuras. Tendo recebido esse conteúdo, o Autenticador entra em contato com o AuC (*Authentication Center* – Centro de Autenticação) da rede GSM e obtém algumas triplas GSM (vetores de autenticação) que contêm material a ser utilizado na autenticação do Suplicante (RAND – número aleatório que servirá de desafio; SRES – valor esperado de resposta do suplicante; Kc – chave de cifração). Cabe salientar que, apesar desse material pertencer à rede GSM, o Autenticador poderá armazená-lo com a

finalidade de utilizar posteriormente em caso de erros. O Autenticador envia, então, para o Suplicante, um Desafio contendo um valor aleatório (RAND) protegido por um MAC (*Message Authentication Code* – Código de Autenticação de Mensagem). Ao receber o Desafio, o Suplicante executa alguns algoritmos do padrão GSM (tal qual o fez o Autenticador) para validar o MAC e para derivar chaves criptográficas que serão utilizadas no restante do processo. Ele responde para o Autenticador, então, com um segundo MAC que protege os valores resposta criados no ambiente do SIM. Ao receber a resposta, o Autenticador valida esse segundo MAC e, caso todo o processo tenha ocorrido dentro do esperado, finaliza o tráfego de mensagens sinalizado para o Suplicante o sucesso do processo de autenticação.

A hierarquia de chaves do EAP-SIM está baseada na geração da "MK" (*Master Key* – Chave mestra) a partir dos parâmetros secretos encontrados na rede GSM (chaves secretas localizadas nos Centros de Autenticação e nos Módulos de Identificação de Assinantes) e dos valores trocados por ocasião do processo de autenticação. A partir daí, a "MK" viabiliza a criação de "TEKs" (*Transient EAP Keys* – Chaves EAP Transientes), que protegerão os pacotes EAP-SIM, da "MSK" e da "EMSK" (comentadas na seção 4.4.1 deste trabalho).

Com o objetivo de diminuir o tempo empregado nos processo de autenticação que utilizam o EAP-SIM, este método viabilizou um mecanismo conhecido como re-autenticação rápida (*fast re-authentication*) que é particularmente interessante em situações nas quais autenticações EAP-SIM são uma necessidade freqüente. Salienta-se que, tal qual na situação do ERP apresentado na seção 4.4.2 deste trabalho, para que haja uma re-autenticação, obrigatoriamente deverá ter ocorrido uma autenticação completa anteriormente. O ganho temporal observado na comparação da autenticação completa com a re-autenticação rápida advém do fato desta não utilizar os algoritmos A3/A8 (do padrão GSM) e não precisar de novas triplas GSM obtidas a partir do Centro de Autenticação (*AuC*). Dessa forma, a re-autenticação rápida pode ser realizada por meio de uma quantidade menor de viagens de ida e volta (*Roundtrips*) do que a autenticação completa.

Apesar de ser algo bastante interessante, esse mecanismo é opcional tanto nas implementações de suplicantes quanto nas de servidores. Assim, caso uma das partes deseje realizar uma re-autenticação rápida e a outra não forneça suporte para tal (ou apenas opte por não fazê-lo), o processo será obrigatoriamente revertido para uma autenticação completa.

As chaves utilizadas no processo de re-autenticação rápida são derivadas daquelas utilizadas na autenticação completa realizada previamente (o que caracteriza a REUTILIZAÇÃO de MATERIAL CRIPTOGRÁFICO). As chaves de autenticação e de cifração utilizadas ("TEKs" derivadas da "MK") são as mesmas do processo de autenticação prévio, entretanto novas "MSK" e "EMSK" serão criadas a partir da "MK" e de novos parâmetros trocados durante a re-autenticação.

4.5.2. Redes 3G - EAP-AKA

O EAP-AKA (EAP *Method for 3rd Generation Authentication and Key Agreement* – Método EAP para Autenticação e Acordo de Chave de 3ª Geração) [RFC4187] é

semelhante ao EAP-SIM, também tendo sido desenvolvido pelo 3GPP e sendo usado para autenticação e distribuição de chaves de sessão. Utiliza, para tal, o mecanismo de Autenticação e Acordo de Chave (AKA) encontrado em redes móveis de 3ª geração UMTS (*Universal Mobile Telecommunications System* – Sistema de Telecomunicações Móveis Universal) e CDMA2000 (*Code Division Multiple Access* 2000 - Acesso Múltiplo por Divisão de Código 2000).

A Figura 4.10 retrata um processo de autenticação completa executado no contexto do EAP-AKA. Também estão omitidos, tal qual na figura relativa ao EAP-SIM, o servidor EAP e outras entidades da arquitetura 3G (UMTS/CDMA2000) que serão utilizadas na autenticação (a comunicação entre o Autenticador e as demais entidades é feita por meio de protocolos AAA). Dessa forma o Autenticador, que na maioria das vezes serve apenas como passagem para as mensagens de autenticação, está aglutinando as funções do Servidor EAP e das demais entidades.

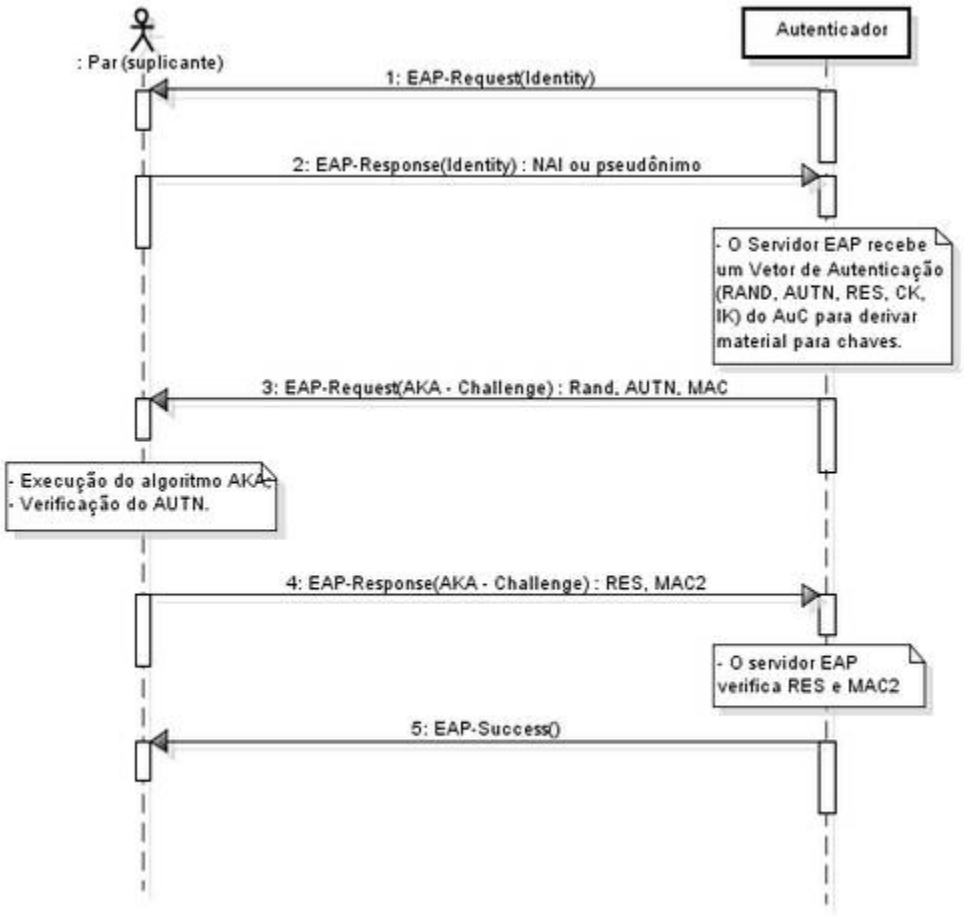


Figura 4.10 – processo de autenticação completa do EAP-AKA

O EAP-AKA utiliza, no mínimo, duas viagens de ida e volta (*round trip*) para a autenticação das partes. Inicialmente, o Autenticador solicita a identidade do Suplicante. Este, por sua vez, responde com o NAI (*Network Access Identifier* - Identificador de

Acesso à Rede), parâmetro vinculado ao IMSI, ou, caso deseje-se preservar a identidade do assinante, com um pseudônimo temporário (ao invés de transmitir a identidade do suplicante neste passo, é possível deixar para comunicá-la em comunicações EAP-AKA futuras). Após isso, o Servidor EAP obtém do Centro de Autenticação (AuC) da rede 3G um vetor de autenticação composto por um número aleatório (RAND), um *token* de autenticação da rede (AUTN), um valor esperado que autenticará o suplicante (RES ou XRES), e duas chaves de sessão, uma de cifração ("CK") e outra para garantia de integridade ("IK"). Salienta-se que esse vetor de autenticação poderá ser armazenado pelo servidor EAP para utilizações posteriores, tornando desnecessárias novas comunicações entre o servidor EAP e o núcleo da rede 3G. A seguir, o servidor EAP inicia o protocolo AKA enviando um Desafio contendo os parâmetros RAND, AUTN e um código de autenticação de mensagem MAC. O suplicante, a partir da execução do algoritmo AKA (utilizando parâmetros secretos contidos em seu *smartcard*), verifica o *token* de autenticação da rede (AUTN) e, em caso de sucesso, responde ao servidor com o parâmetro RES, que permitirá sua autenticação perante esse servidor, e um MAC que viabiliza a garantia de integridade dessa mensagem. Tendo recebido a resposta, o servidor atesta a autenticidade do suplicante por meio da verificação dos parâmetros RES e MAC2. Finalizado o procedimento, o servidor envia uma mensagem para o suplicante sinalizando o sucesso de todo o processo. Cabe salientar que o autenticador poderá receber, do servidor EAP, material para chaves que na será repassado para o suplicante tendo em vista o fato deste já possuir esse material em seu meio.

A hierarquia de chaves do EAP-AKA é parecida à do EAP-SIM e está baseada na geração da "MK" (*Master Key* – Chave mestra) a partir dos parâmetros AKA encontrados na rede 3G ("IK" e "CK" discutidos anteriormente) e da identidade trocada por ocasião do processo de autenticação. A partir daí, a "MK" viabiliza a criação de "TEKs" (*Transient EAP Keys* – Chaves EAP Transientes), que protegerão os pacotes EAP-AKA, da "MSK" e da "EMSK" (comentadas na seção 4.4.1 deste trabalho).

Semelhante àquilo que ocorre no EAP-SIM e com o objetivo de diminuir o tempo empregado nos processo de autenticação que utilizam o EAP-AKA, este método viabilizou um mecanismo conhecido como re-autenticação rápida (*fast re-authentication*) que é particularmente interessante em situações nas quais autenticações EAP-AKA são uma necessidade freqüente. Salienta-se que, tal qual na situação do ERP apresentado na seção 4.4.2 deste trabalho, para que haja uma re-autenticação, obrigatoriamente deverá ter ocorrido uma autenticação completa anteriormente. O ganho temporal observado na comparação da autenticação completa com a re-autenticação rápida advém do fato desta não utilizar os algoritmos AKA (do padrão 3G) e não precisar de novos vetores obtidos a partir do Centro de Autenticação (*AuC*). Dessa forma, a re-autenticação rápida pode ser realizada por meio de uma quantidade menor de viagens de ida e volta (*Roundtrips*) do que a autenticação completa.

Apesar de ser algo bastante interessante, esse mecanismo é opcional tanto nas implementações de suplicantes quanto nas de servidores. Assim, caso uma das partes deseje realizar uma re-autenticação rápida e a outra não forneça suporte para tal (ou apenas opte por não fazê-lo), o processo será obrigatoriamente revertido para uma autenticação completa.

As chaves utilizadas no processo de re-autenticação rápida são derivadas daquelas utilizadas na autenticação completa realizada previamente (o que caracteriza, tal qual no EAP-SIM, a REUTILIZAÇÃO de MATERIAL CRIPTOGRÁFICO). As chaves de autenticação e de cifração utilizadas ("TEKs" derivadas da "MK") são as mesmas do processo de autenticação prévio, entretanto novas "MSK" e "EMSK" serão criadas a partir da "MK" e de novos parâmetros trocados durante a re-autenticação.

4.6. Mecanismos de gerenciamento de chaves

Com o objetivo de apresentar alguns desafios e soluções para o problema do gerenciamento de chaves criptográficas utilizadas nas redes envolvidas, serão abordados nesta seção alguns aspectos gerais do processo, hierarquia de chaves, distribuição e reuso de material usado para a criação de chaves.

4.6.1. Aspectos gerais

A atividade de gerenciamento de chaves é algo complexo e geralmente envolve vários elementos (entidades e procedimentos). Analisando, por exemplo, o gerenciamento de chaves a serem utilizadas em um processo de AAA (seção 4.3), percebe-se que estão envolvidas entidades como servidores de AAA e vários elementos do núcleo das redes consideradas, além de serem utilizados procedimentos para viabilizar os diversos controles necessários e protocolos de comunicação seguros para permitir a troca adequada de informações. Dessa forma, constata-se facilmente que um mecanismo que se propõe a gerenciar chaves deve ser, no mínimo, tão complexo quanto os problemas que ele tenta resolver.

O êxito em operações com sistemas criptográficos depende intimamente de um bom gerenciamento de chaves que engloba atividades como criação de chaves, armazenamento, transmissão segura para as partes autorizadas, utilização (e re-utilização) segura de material criptográfico, troca (reposição) de chaves e sua destruição, dentre outras.

Muitos têm sido os trabalhos realizados sobre essas diversas atividades (separadamente ou em conjunto). Nesse contexto a presente seção não pretende exaurir o assunto, mas tão somente dar noções sobre alguns problemas existentes e algumas de suas possíveis soluções, particularmente no que diz respeito às inter-conexões tratadas neste curso.

Especificamente relacionado às chaves tratadas nas seções passadas (EAP), a [RFC 5247] especifica uma hierarquia de chaves EAP e sugere um *framework* para o transporte e utilização do material a ser utilizado na geração das chaves e dos parâmetros gerados pelos métodos EAP.

Para um entendimento mais detalhando do processo de derivação de chaves EAP, faz-se necessária uma visão mais geral das fases de uma conversação que embute a utilização de uma autenticação EAP. Pode-se, então, considerar a ocorrência de três fases distintas [RFC 5247]:

1. Descobrimento – suplicante (par) encontra o autenticador e descobre suas possibilidades e compatibilidades;

2. Autenticação – tem início após o suplicante e o autenticador terem se reconhecido mutuamente e viabiliza a garantia da identificação das partes envolvidas;

    a. Autenticação EAP – sub-fase obrigatória e que viabiliza a autenticação por meio do protocolo EAP. Nessa sub-fase o material de criação de chaves é derivado tanto no suplicante quanto no servidor EAP envolvido (que pode ou não estar embutido no autenticador) a partir de valores armazenados previamente nessas entidades;

    b. Transporte de chaves por protocolo de AAA (opcional) – essa sub-fase ocorre caso haja a necessidade de comunicação entre o autenticador e um servidor EAP que se encontra à retaguarda. Nesse caso é utilizado um protocolo de AAA (tipicamente o RADIUS ou o Diameter) para viabilizar essa comunicação e permitir que o material de criação de chaves adequado (tipicamente a MSK) seja repassado do servidor para o autenticador (que é quem de fato entrará em contato direto com o suplicante);

3. Protocolo de associação segura - tendo ocorrido um processo de autenticação com êxito, deverá ser estabelecida uma associação de segurança entre as partes extremas (suplicante e autenticador) para viabilizar a comunicação de informações (não mais de dados ligados à autenticação) entre suplicante e autenticador.

É interessante observar que a maioria das fases descritas anteriormente não são executadas pelo protocolo EAP em si. As fases 1, 2.b e 3 são desencadeadas além do próprio escopo do método EAP considerado (1 e 3 dependem das camadas inferiores e 2.b depende de um protocolo de AAA externo).

4.6.2. Hierarquia de chaves

Como percebido no item anterior, as derivações de chaves a serem utilizadas pelos métodos EAP sempre são feitas a partir de conteúdo existente nas entidades envolvidas no processo. Esse conteúdo pode estar no formato de segredos pré-compartilhados ou de chaves assimétricas (públicas e privadas) e dá origem a dois tipos de material para a criação de chaves EAP: material calculado localmente pelo método EAP e não exportado (TEKs, por exemplo) e material calculado e exportado pelo método EAP (MSK e EMSK, por exemplo).

Cada chave criada no âmbito do *framework* de gerenciamento de chaves EAP tem um nome e um escopo bem definidos. Podem ser citadas, por exemplo:

- MSK (*Master Session Key* – Chave Mestra de Sessão) - chave derivada, entre o suplicante e o servidor, gerada a partir de material previamente existente e que é exportada pelo método EAP;

- EMSK (*Extended* MSK – MSK Estendida) - chave adicional criada no mesmo âmbito da MSK e utilizada para outros propósitos; é exportada pelo método EAP, mas é de conhecimento apenas do suplicante e do servidor;
- TEK (*Transient EAP Key* – Chave EAP Transiente) - chaves de sessão que são utilizadas para o estabelecimento de um canal protegido entre suplicante e servidor durante o processo de autenticação EAP.
- TSK (*Transient Session Keys* - Chaves Transientes de Sessão) – chaves utilizadas na proteção dos pacotes de dados (canal protegido) que trafegam entre suplicante e autenticador após o processo de autenticação EAP.

Cabe salientar que a [RFC 5295] define que a EMSK poderá ser utilizada para a derivação de chaves de aplicações específicas, a saber:

- USRK (*Usage Specific Root Key* – Chave Raiz de Uso Específico);
- DSRK (*Domain Specific Root Key* – Chave Raiz de Domínio Específico);
- DSUSRK (*Domain Specific Usage Specific Root Key* – Chave Raiz de Domínio e Uso Específico).

Essas chaves, como indicado pelos próprios nomes, têm aplicações específicas e propõe-se a resolver problemas particularizados de certos métodos (ou melhorias de métodos) e também podem ter seus escopos delimitados para certos domínios de gerenciamento de chaves.

A Figura 4.11 ilustra a dependência existente entre as diversas chaves abordadas.

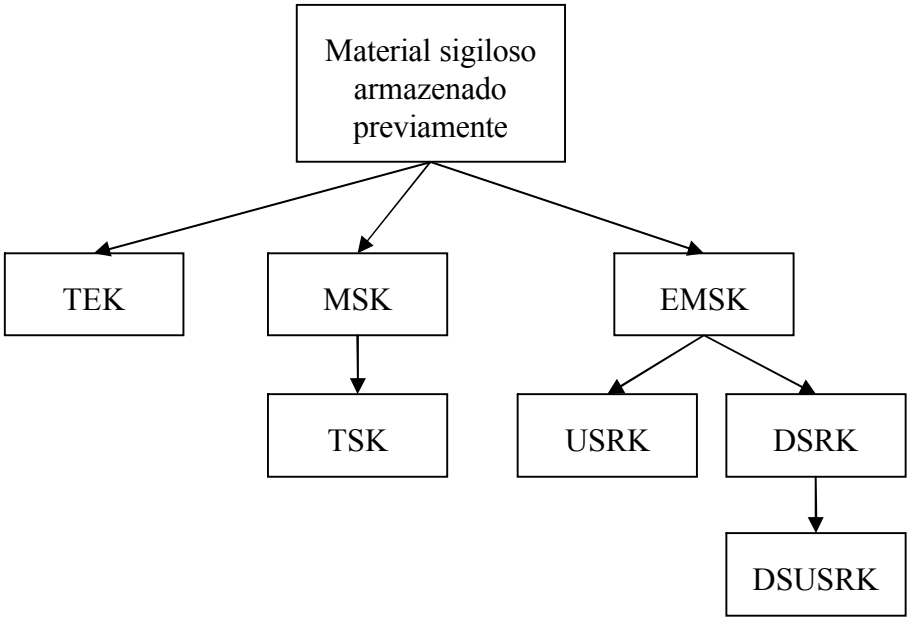


Figura 4.11 – Hierarquia de chaves EAP

4.6.3. Distribuição e reuso de material para chaves

O processo de geração de chaves criptográficas a serem utilizadas em protocolos de autenticação, como pôde ser visto, é algo complexo e demorado. Apesar dessas chaves serem criadas prioritariamente para viabilizar segurança entre um determinado suplicante e um autenticador, tem-se percebido que pelo menos parte do material criptográfico pode ser re-utilizado em novas parcerias (o mesmo suplicante e um novo autenticador), o que certamente permite uma diminuição relevante do computo do tempo global da nova autenticação. Entretanto, além do ganho gerado por esse novo processo, há também o ônus relacionado à necessidade de novos mecanismos que viabilizem a distribuição e o reuso do material de forma segura.

No contexto das chaves EAP que podem ser utilizadas em *handovers*, a [RFC 5749] propõe um mecanismo que viabiliza a distribuição controlada da chave raiz de um servidor EAP para outro servidor de rede que necessita dessa chave (ou de suas derivadas). O documento também propõe um modelo geral para um protocolo de intercâmbio de distribuição de chaves (KDE – *Key Distribution Exchange*) que se propõe a distribuir diferentes tipos de chaves utilizando protocolos de AAA.

O mecanismo de distribuição de chaves comentado anteriormente fundamenta-se na existência de dois servidores: o KDS (*Key Delivering Server* – Servidor de Distribuição de Chave), que é um servidor de rede que possui uma EMSK (ou uma DSRK) e a utiliza para criar e distribuir RKs (*Root Key* – Chave Raiz) para outros servidores, e o KRS (*Key Requesting Server* – Servidor de Requisição de Chave), que se comunica com o KDS solicitando as RKs. Uma possível forma de funcionamento da arquitetura de distribuição de chaves é, por exemplo:

1. Um suplicante solicita conexão a um servidor;
2. O servidor age como um KRS e solicita a um KDS (que de alguma forma já tenha compartilhado material de criação de chaves com o suplicante) uma RK que viabilize o fornecimento do serviço solicitado pelo suplicante;
3. A partir desse momento (e enquanto a RK não expirar) o KRS poderá atender a novas solicitações de mesmo suplicante utilizando a RK obtida e utilizadas na derivação das chaves adequadas.

Salienta-se que as EMSK, por definição, não podem sair do servidor EAP original (que contém os parâmetros adequados compartilhados com os suplicantes considerados), dessa forma os KDS obrigatoriamente serão servidores EAP originais ou então conterão apenas DSRKs derivadas de EMSKs localizadas em servidores EAP originais.

Além da arquitetura ilustrada, a [RFC 5749], como dito anteriormente, propõe o KDE. O papel desse protocolo é viabilizar a distribuição segura e controlada das RKs entre KRSs e KDSs. Para tal utiliza, além de seus próprios mecanismos internos, protocolos de AAA como os vistos na seção 4.3. A troca de mensagens gerenciadas pelo KDE é simples e baseia-se no envio, do KRS para o KDS, de uma "KDE-Request", e do KDS para o KRS, de uma "KDE-Response". Uma "KDE-Request" é um KRT (*Key Request Token – Token* de Requisição de Chave) encapsulado em uma mensagem de

AAA. Já uma "KDE-Response" é um KDT (*Key Delivery Token* – Token de Entrega de Chave) que contém a chave solicitada e também é encapsulado em uma mensagem de AAA.

O KDE é um protocolo que pode ser utilizado, por exemplo, para viabilizar os processos de re-autenticação propostos pelo ERP visto na seção 4.4.2.

4.7. Processos de *Handover* e o padrão IEEE 802.21

Nessa seção serão descritas as métricas, as técnicas e as fases do processo de *handover*, caracterizando cada uma delas no contexto das redes envolvidas e destacando os aspectos de segurança observados. Serão apresentados os tipos de *handover* caracterizados na literatura e as métricas de qualidade de serviço (QoS) que são utilizadas para sinalizar os impactos da execução de processos de segurança para a consecução do *handover*. Além disso, serão apresentados também os aspectos gerais do padrão IEEE 802.21 (*Media Independent Handover – Handover* Independente do Meio) – padrão aplicado aos *handover*s entre redes heterogêneas que são alvo deste trabalho - detalhando sua arquitetura e seu funcionamento a fim de que fiquem claras as suas possibilidades e a sua aplicabilidade dentro do contexto do problema explorado.

4.7.1. Tipos de *handover*

O processo de *handover* possui algumas características de acordo com suas classificações. A primeira e mais básica delas é feita em relação ao seu tipo, podendo ser [IEEE 802.21]:

1. *soft handover* - em que os recursos para suporte a fluxos de tráfego estão continuamente disponíveis durante as transferências de conexão do nó móvel, em nível de enlace, de um ponto de acoplamento origem para um ponto de acoplamento alvo; neste caso, são alocados recursos para o ponto de acoplamento alvo antes da ocorrência do evento de troca de enlace (correspondente a *make-before-break* – faça antes de quebrar);
2. *seamless handover* - *handover* no qual há mudança do ponto de acoplamento, em nível da camada de enlace, no entanto o nó móvel não sofre degradação dos parâmetros de serviço (qualidade do sinal, por exemplo) ou, caso sofra, essa degradação é tolerável tanto para o nó móvel quanto para a própria rede servidora;
3. *hard handover* - onde o *handover* é feito de forma abrupta, ocorrendo a perda da conexão anterior antes da atual ser estabelecida; neste sentido, recursos para suportar fluxos de tráfego são sujeitos a completa indisponibilidade entre a perda da conexão e a sua restauração (correspondente a *break-before-make* – quebre antes de fazer).

Outra classificação do *handover* é feita baseada nos tipos de tecnologia de acesso (RAT - *Radio Access Tecnologies*) das redes envolvidas. Durante a migração de um terminal móvel entre redes com a mesma RAT, o *handover* é dito horizontal (HHO – *Horizontal Handover*) ou intra-tecnologia (*Intratechnology Handover*). Por outro lado, em se tratando de redes heterogêneas, as RATs deverão ser diferentes, caracterizando o

handover dito vertical (VHO – *Vertical Handover*) ou inter-tecnologias (*Intertechnology Handover*) que, sem dúvida necessita de maiores cuidados em sua gerência [Dutta ET AL, 2008].

Há também a classificação que se preocupa com os domínios envolvidos no processo de *handover*. Nesse caso, um *handover* pode ser considerado Intra-domínio (*Intradomain*), para situações nas quais as redes envolvidas fazem parte do mesmo domínio administrativo, ou Inter-domínios (*Interdomain*), situação observada quando o móvel passa de uma rede pertencente a um domínio administrativo para outra rede, de outro domínio administrativo.

De outra maneira, o *handover* pode ser também classificado em relação à sua execução. Nessa classificação, encontramos:

1. o *handover* assistido pelo móvel (MAHO – *Mobile Assisted Handover*), que, embora executado pela rede, é baseado em informações de qualidade da conexão medidas no móvel;
2. o *handover* iniciado pela rede (NIHO – *Network Initiated Handover*), no qual o móvel não possui nenhuma participação quer seja na identificação, quer seja na execução do *handover*, ficando todas as decisões a critério da rede; e
3. o *handover* controlado pelo móvel (MCHO – *Mobile Controlled Handover*), que, por sua vez, é identificado e executado integralmente pelo móvel.
4. o *handover* controlado pela rede (NCHO – *Network Controlled Handover*), em que a rede toma a decisão a respeito do handover e instrui o móvel com relação à sua execução.

Em relação ao momento de execução das diversas atividades componentes de um *handover*, é possível dividir o procedimento em três fases distintas, conforme a Figura 4.12.

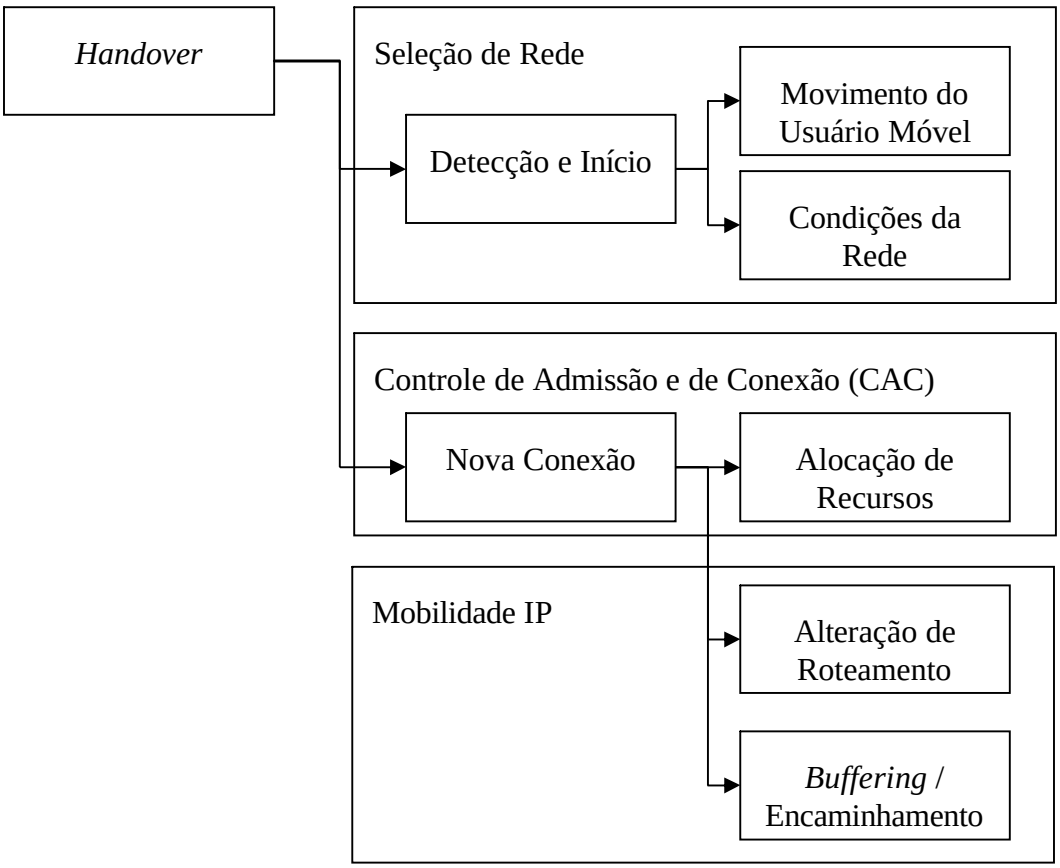


Figura 4.12 – fases do *Handover*

Percebe-se que a integração entre as soluções apresentadas para os problemas de seleção de redes (*Network Selection*), controle de admissão de conexão (CAC) e gerência de mobilidade IP (*Mobility* IP) é essencial para o sucesso da arquitetura de integração entre redes sem fio, visto que juntas elas tentam atender aos requisitos de um *handover* eficiente, a saber:

- A detecção e o início do *handover* devem acontecer antes da perda da conexão atual do móvel (*soft handover*);
- A melhor rede deve ser sempre selecionada antes da execução do *handover*;
- O *handover* não pode deixar de ser finalizado devido à falta de recursos na nova rede;
- A latência do *handover* deve ser a menor possível, visando atender aos requisitos de aplicações com altos requisitos de qualidade de serviço (QoS).

Cabe salientar que esse último requisito é o ponto fundamental no qual os processos de segurança discutidos neste trabalho interferem. Daí a necessidade de prover-se segurança gerando o mínimo de latência possível.

4.7.2. Garantia de QoS durante o *handover* X requisitos de segurança

Os estudos feitos no sentido de integrar redes wireless podem ser categorizados de três formas: os esforços feitos visando à padronização de tecnologias que possam integrar, de forma transparente ao usuário, essas redes; os esforços feitos no sentido de reduzir a latência dos efeitos da mobilidade nessa integração; e os esforços feitos no sentido de garantir ao usuário uma transição transparente entre redes, preservando todos os seus requisitos de qualidade de serviço.

O segundo e o terceiro problemas estão intimamente ligados à execução de processos de segurança durante a passagem de uma rede para outra. A latência, como comentado anteriormente, é o ponto principal aonde a segurança interfere, pois é patente que quanto maior a preocupação com a segurança, maior será o tempo dedicado à execução de processos de segurança, o que poderá gerar prejuízos para o desempenho (segurança X desempenho) geral do processo de *handover*. Além disso, a transição transparente pressupõe a manutenção de sessões de aplicações que porventura estejam sendo realizadas, o que pode ser prejudicado pelo atraso gerado pelos processos de segurança ao processo geral de *handover*.

O problema de garantir qualidade de serviço em redes wireless pode ainda ser sub-dividido em três outros:

1. o problema da seleção de redes;
2. o problema do controle de admissão de conexões; e
3. o problema da reserva e escalonamento de recursos.

Normalmente, estes problemas estão intimamente ligados e suas soluções se fundem quando a intenção é integrar redes sem fio, minimizando os impactos do processo de migração entre as redes envolvidas na integração.

Sendo assim, os aspectos ligados à seleção de redes têm sido abordados, de forma eficiente, levando em consideração as informações medidas diretamente no móvel, o que normalmente fornece uma posição mais real e efetiva sobre a qualidade de cada rede, sob o ponto de vista de quem está efetivamente recebendo o serviço.

No entanto, em uma visão sistêmica do processo de *handover*, não faz muito sentido selecionar uma rede com bons níveis globais de QoS e, ao tentar efetuar o *handover* para a mesma, os mecanismos de CAC e alocação de recursos recusarem tal conexão. Isso implicaria em aumento na latência global do processo, visto que uma nova rede teria que ser selecionada.

Por essa razão, o mais interessante é que a admissão da rede seja um dos parâmetros considerados na função de custo usada pelo mecanismo de seleção de rede, mostrando a interligação necessária entre as soluções para cada uma dessas fases do processo de *handover*.

Por outro lado, considerando os aspectos de segurança, as avaliações mostram-se mais complexas, pois é inviável, por exemplo, mensurar previamente a probabilidade de um móvel obter êxito em um processo de autenticação para permitir acesso à nova rede. Salienta-se que o fracasso nesse processo inviabilizará a nova conexão e gerará a

necessidade de seleção de uma nova rede, aumentando consideravelmente a latência e provavelmente gerando a queda das sessões de aplicações em execução.

Em um contexto onde temos redes sem fio heterogêneas (WLANs e redes celulares, por exemplo), podemos contar com duas situações distintas:

1. o móvel possui interfaces distintas que permitam o mesmo estar conectado às duas redes ao mesmo tempo (multimodo); e
2. o móvel está sempre conectado usando apenas uma de suas interfaces.

No primeiro caso, as redes teriam coberturas sobrepostas, podendo acontecer o *soft handover* com característica vertical, tendo em vista as diferenças entre as tecnologias de acesso. No segundo caso, independentemente das redes possuírem áreas de cobertura sobrepostas, a única possibilidade de passagem é o *hard handover*.

Assim, em ambos os casos, os dispositivos móveis, usando seus procedimentos padrão de seleção de redes, tendem a selecionar àquela cujo nível de sinal é julgado melhor. No entanto, apesar dessa avaliação poder ser eficiente para *handovers* horizontais visto que as grandezas medidas em relação ao nível de sinal são as mesmas, isso não ocorre para situações nas quais são necessários *handovers* verticais.

4.7.3. Arquiteturas e tecnologias de integração de redes heterogêneas

Muitos estudos têm adotado formas de reduzir a latência do *handover* entre redes heterogêneas usando MIH [IEEE 802.21], MIP [RFC 3344] e SIP [RFC 3261], esquemas de Controle de Admissão de Conexão e alocação de recursos.

Esses estudos se baseiam em arquiteturas para a integração categorizadas como muito acoplada, pouco acoplada e não acoplada (também conhecida como *peer-to-peer*) [Munasinghe & Jamalipour, 2008] [Song ET AL., 2007]. Os conceitos que norteiam essa categorização podem ser resumidos como segue.

Na arquitetura muito acoplada, a WLAN está diretamente ligada ao núcleo da rede celular, fazendo com que tanto o tráfego de dados quanto o de sinalização, sejam roteados através do núcleo da rede celular antes de chegar à rede IP externa. Portanto, as técnicas de gerência de mobilidade usadas em redes celulares podem ser aplicadas diretamente sobre a WLAN, considerando que a mesma é parte integrante da rede celular.

Por outro lado, em arquiteturas pouco acopladas, a troca de sinalização entre a WLAN e a rede celular é feita através do núcleo da rede celular, enquanto que os fluxos de dados são encaminhados diretamente à rede IP externa.

Uma vez que o tráfego de dados é encaminhado diretamente a uma rede IP externa, este método pode ajudar a evitar um gargalo de tráfego no núcleo da rede celular, fazendo com que esse método seja mais eficiente nas entregas de dados e, portanto, a mobilidade de sessões de conexões que exijam mais QoS possa ser garantida mais facilmente [Yang and Deng, 2007].

Finalmente, as arquiteturas *peer-to-peer* consideram as redes celulares e WLAN como redes independentes, interligadas normalmente através de gateways. Essa arquitetura pode ser vista como uma variante da arquitetura pouco acoplada [Yusof ET AL., 2007]. Neste caso, a gerência de mobilidade deve ser realizada por um protocolo de camada superior como Mobile IP (MIP) [Munasinghe & Jamalipour, 2007].

4.7.4. O padrão IEEE 802.21

O padrão IEEE 802 não suporta o *handover* vertical entre os diferentes tipos de redes. Ele também não fornece mecanismos para facilitar o *handover* vertical em um ambiente de redes heterogêneas. Portanto, um novo padrão, chamado IEEE 802.21, foi proposto com o objetivo de preencher essa lacuna bastante atual.

Ele possui algoritmos para permitir o *seamless handover* (*handover* suave) entre diferentes tipos de rede, através do fornecimento de informações que permitam a entrega de dados entre redes celulares, GSM, GPRS, WLAN, Bluetooth, e WiMaX através de diferentes mecanismos de transmissão. Este padrão também é chamado de MIH (*Media Independent Handover – Handover* Independente do Meio) [da Silva, 2009].

Os objetivos do padrão IEEE 802.21 podem ser resumidos como [Machan ET AL., 2008]:

- Habilitar mobilidade e *seamless handover* entre redes heterogêneas sem fio;
- Incluir definições de objetos gerenciados que são compatíveis com as normas de gerência do SNMP (*Simple Network Management Protocol* – Protocolo Simples de Gerenciamento de Rede);
- Incluir a definição de um nova abstração de serviço na camada de enlace para fornecer uma tecnologia com interface comum e independente da camada física;
- Definir um conjunto de funções que, interagindo com as camadas superiores possam executar, de forma eficiente, *handover*s verticais;
- Fornecer suporte para autenticação, autorização e detecção/seleção de rede.

Esse último objetivo está totalmente ligado ao tema abordado neste curso e tem sido alvo de estudos recentes realizados por um grupo de trabalho vinculado ao padrão MIH e que preocupa-se exclusivamente com os aspectos de segurança a serem considerado pelo padrão.

Quando um usuário se movimenta entre dois pontos de acessos utilizando a mesma tecnologia, o *handover* geralmente pode ser realizado usando métodos nativos da própria tecnologia sem fio, sem envolver a função MIH (MIHF – *MIH Fuction*). Dessa forma, durante a realização de uma sessão de uma determinada aplicação (uma chamada de voz sobre IP, por exemplo), um nó móvel conectado a uma rede (WLAN ou rede celular) e envolvido nessa sessão pode se movimentar de um ponto de acesso para outro na mesma rede (*handover* intra-tecnologia) e utilizar mecanismos internos ao padrão considerado (IEEE 802.11, GSM, UMTS, etc.) para tratar esse *handover*.

Contudo, se o *handover* ocorre motivado pela movimentação de um equipamento entre um ponto de acesso WLAN localizado em uma rede corporativa e outro ponto de acesso de uma rede celular pública (tipicamente uma Estação Rádio Base – ERB), por exemplo, então o MIH é necessário, pois os dois pontos de acesso pertencem a tecnologias distintas.

As principais funcionalidades fornecidas pelo MIH residem na possibilidade de comunicação entre redes sem fio com tecnologias de enlace diferentes e entre elas e a camada de rede (IP). Este procedimento de *handover* utiliza as informações dos MN (*Mobile Node* - Nó Móvel) e das infra-estruturas das redes.

O padrão IEEE 802.21 informa a disposição das próximas redes ao MN e o ajuda a detectar e selecionar a melhor rede. Esta informação inclui dados sobre a camada de enlace de cada rede. O MIH também pode se comunicar com vários protocolos das camadas superiores que utilizam o IP (SIP e MIP, por exemplo).

A MIHF engloba três tipos de serviços que têm o objetivo de facilitar o *handover* entre redes heterogêneas:

- O MIES (*Media Independent Event Service* – Serviço de Evento Independente do Meio): dá suporte à transferência, filtragem e classificação de alterações dinâmicas da camada de enlace para a camada de rede. Isto gera eventos que informam o estado do enlace (“up/down”) e/ou a disponibilidade de um novo enlace.
- O MICS (*Media Independent Command Service* - Serviço de Comando Independente do Meio): possibilita o controle e gerência das características do enlace da rede que possam contribuir para a decisão do *handover*. O MICS oferece as funções para que a camada de rede possa gerenciar e controlar a camada de enlace.
- O MIIS (*Media Independent Information Service* - Serviço de Informação Independente do Meio): oferece as informações que são necessárias para realizar o *handover*. Ele define um serviço que fornece informações contendo uma lista de redes disponíveis, a versão do protocolo IP utilizado e os dados sobre as operadoras dessas redes, visando à realização de *handover*s mais rápidos. Usando estas informações, o terminal móvel é capaz de tomar uma decisão sobre o *handover*.

As mensagens criadas a partir desses serviços são retransmitidas pela MIHF que está localizada entre a camada de enlace e a camada de rede. O MIHF fornece interfaces homogêneas e independentes de tecnologias de enlace utilizadas pelas redes. Estas interfaces manipulam a comunicação entre as camadas 3 e 2, conforme ilustrado na Figura 4.13.

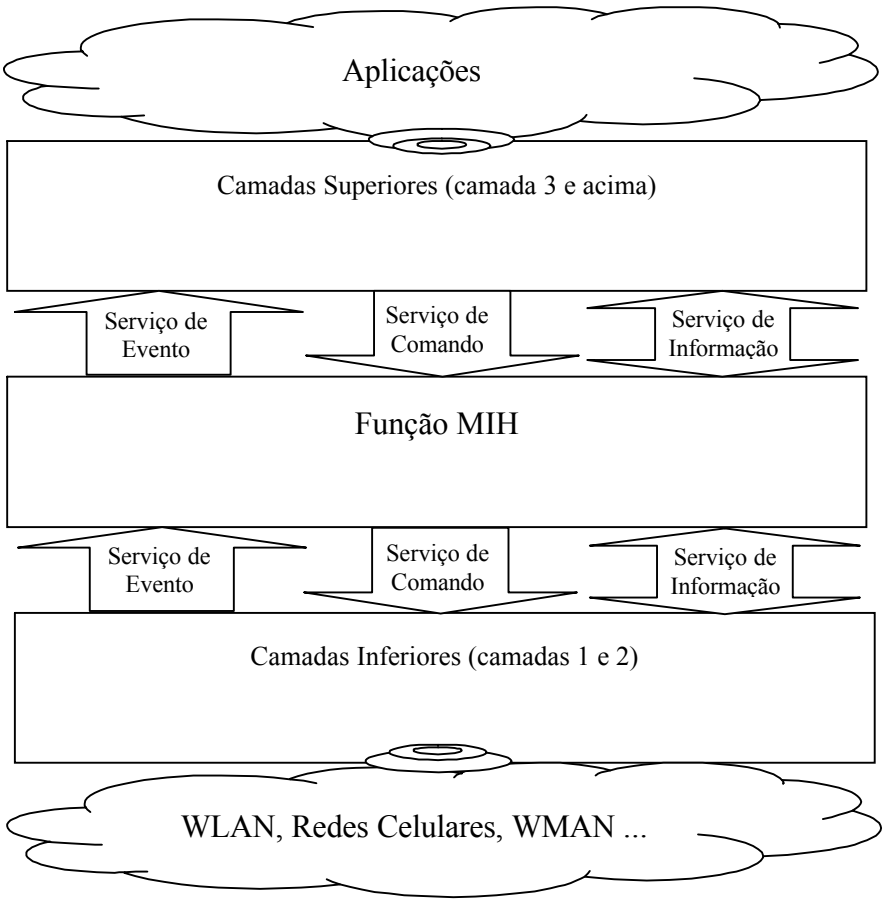


Figura 4.13 – arquitetura do padrão IEEE 802.21 - MIH

Apesar de viabilizar uma série de ferramentas a serem utilizadas para a consecução de *handovers* verticais, a especificação base do padrão [IEEE 802.21] não trata de aspectos de segurança ligados, por exemplo, ao problema da autenticação dos equipamentos móveis. Essa lacuna, entretanto, tem sido alvo de um novo grupo de trabalho (802.21-a *Task Group*) vinculado ao padrão e que foi criado especificamente para tratar dos problemas de segurança.

Esse grupo considera, de uma forma geral, os seguintes passos como necessários para viabilizar segurança no acesso e utilização das redes envolvidas nos *handovers* verticais:

1. Autenticação para acesso à rede – passo fundamental que permite a identificação, por parte de um servidor de autenticação, do usuário (nó móvel) a fim de que sejam verificados seus privilégios para que seja viabilizado um acesso adequado;
2. Criação de uma associação segura – compartilhamento e derivação de parâmetros que permitirão a troca segura de informações entre as partes envolvidas na comunicação aérea (nó móvel e ponto de acesso) do processo;

3. Controle de acesso e cifração de dados – o controle de acesso permite que somente usuários autenticados acessem aos serviços adequados, já a cifração viabiliza a proteção das informações que trafegarão entre as partes envolvidas.

Alguns outros detalhes sobre algumas propostas que têm sido feitas por esse grupo (e por outros acadêmicos que estão preocupados com os aspectos de segurança vinculados ao padrão IEEE 802.21) poderão ser observadas na seção 4.8 deste trabalho.

4.8. Protocolos de segurança aplicados no *Handover* Inter-tecnologias

Nesta seção serão tratados alguns protocolos de autenticação envolvidos nas operações de *handover* inter-tecnologias. Dessa forma, serão apresentados os protocolos MPA (*Media Independent Pre-Authentication* – Pré-autenticação Independente do Meio), que será priorizado tendo em vista sua importância no contexto tratado, o HOKEY (*Handover Keying* – "Chaveamento" de *Handover*) e o PANA (*Protocol for Carrying Authentication for Network Access* – Protocolo para Transporte de Autenticação para Acesso a Rede). Cabe salientar que a apresentação desses protocolos tem o objetivo de ilustrar os diversos esforços que têm sido feitos no contexto da integração de padrões existentes para a resolução do problema tratado nesse curso.

4.8.1. MPA

O MPA é um *framework* que encontra-se atualmente em estudo por um grupo de trabalho do IETF (*Internet Enginnering Task Force* – Força Tarefa de Engenharia da Internet). O último rascunho do padrão publicado pelo IETF (*Internet Enginnering Task Force* – Força Tarefa de Engenharia da Internet) data de 16 de abril de 2010 [Dutta ET AL, 2010]. Ele é considerado um mecanismo de otimização de *handover* pois procura minimizar o tempo de latência gerada pelos diversos processos de autenticação que podem estar envolvidos na passagem de uma rede para outra. O framework focaliza não apenas *handover* inter-tecnologias, mas também, e principalmente, *handover* inter-domínios. Ele propõe-se a trabalhar sobre qualquer camada de enlace e com qualquer protocolo de gerenciamento de mobilidade, incluindo Mobile IP [RFC3344], Mobile IPv6 [RFC3775], MOBIKE [RFC4555], HIP [RFC5201]. Possui operações de pré-autenticação e pré-configuração que permitem que muitas das operações inerentes a um *handover* ocorram antes da passagem do equipamento móvel para a outra rede.

Foram consideradas, dentre outras, as seguintes idéias na elaboração da proposta do *framework*:

- Otimização de mobilidade unificada que funcione com qualquer protocolo de gerenciamento de mobilidade;
- Possibilidade de trabalhar em domínios administrativos diferentes de forma segura, na qual exista um relacionamento de confiança entre o terminal móvel e cada domínio administrativo, sendo que não necessariamente esses domínios devam ter associações de segurança firmadas previamente entre eles;

- Suporte a terminais de diversos tipos, dos que têm múltiplas interfaces e que suportam conexões simultâneas com mais de uma rede aos que possuem apenas uma interface.

Essas idéias deram origem a requisitos que são atendidos por meio de quatro procedimentos básicos do MPA [Dutta ET AL, 2008]:

1. Pré-autenticação, na qual há o estabelecimento de uma SA (*Security Association* - Associação de Segurança) entre o equipamento móvel e a CTN (*Candidate Target Network* – Rede Candidata Alvo) para viabilizar a segurança do protocolo de sinalização que será executado em seguida;
2. Pré-configuração, onde é executado um protocolo de configuração que permite a obtenção, por parte do equipamento móvel, de um endereço IP e de outros parâmetros obtidos a partir da CTN;
3. Execução de um protocolo de tunelamento que estabelece o que é conhecido como PHT (*Proactive Handover Tunnel* – Túnel Pró-ativo de *Handover*) entre o equipamento móvel e um roteador de acesso à CTN (salienta-se que os endereços internos do túnel que permitem o tráfego dos pacotes contêm o endereço IP obtido na fase anterior);
4. Exclusão do PHT, o que é feito imediatamente antes do equipamento móvel conectar-se diretamente à CTN, e re-atribuição do endereço interno do túnel excluído à interface física, o que é feito imediatamente após a conexão à nova rede.

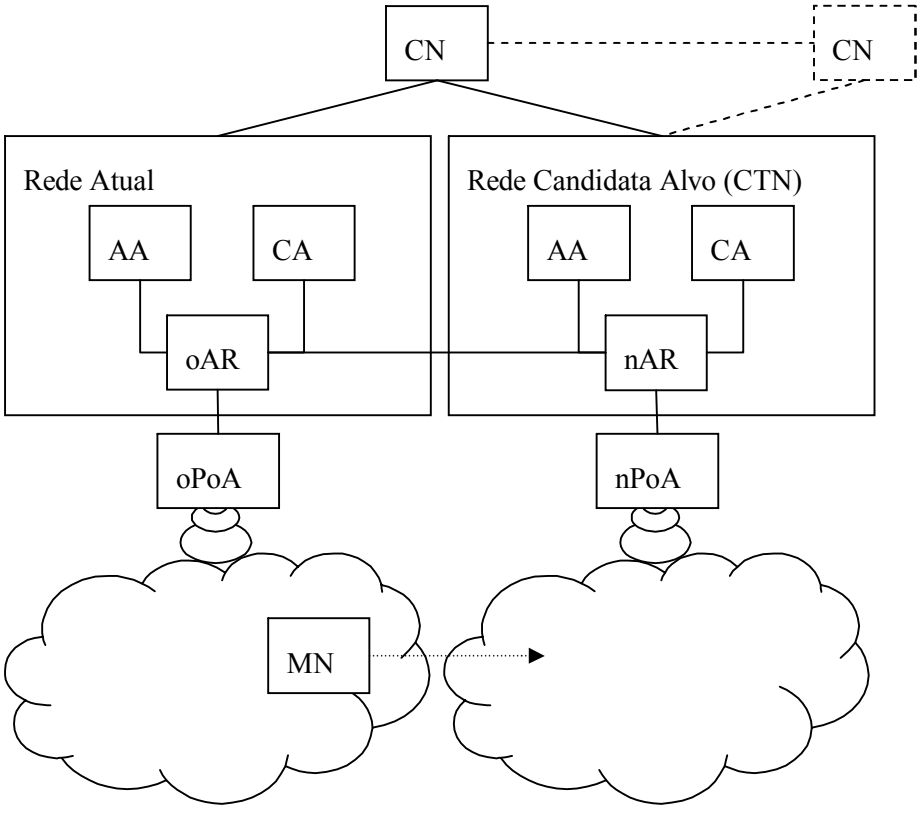


Figura 4.14 – Elementos funcionais do MPA (adaptado de [Dutta ET AL, 2010])

Os elementos funcionais que fazem parte do MPA (ou estão vinculados a este) podem ser entendidos por meio da Figura 4.14:

- CN (Core Network – *Núcleo da Rede*) – não faz parte do *framework* mas está ligado diretamente a este (o MPA prevê a interligação entre núcleos de redes diferentes, o que pode viabilizar, por exemplo, que a CTN esteja vinculada ao núcleo de outra rede).
- AA (*Authentication Agent* – Agente de Autenticação) – responsável pela pré-autenticação do MN por meio da execução de um protocolo de autenticação que permite a geração de uma Associação de Segurança MPA (MPA-SA). Note que esse protocolo de autenticação deve prever a possibilidade de contato do AA com servidores de AAA (do núcleo da própria rede ou de outra vinculada) com o objetivo de se obter material a ser utilizado na criação de chaves que viabilizarão a criação da MPA-SA considerada. Salienta-se que o EAP e seus métodos podem ser considerados protocolos adequados à pré-autenticação tratada no âmbito do MPA.
- CA (*Configuration Agent* – Agente de Configuração) – é o agente responsável pela execução segura do protocolo de pré-configuração que viabilizará a entrega de um endereço IP (e outros parâmetros de configuração) para o MN. Essas informações de configuração devem ser protegidas a partir de material derivado da MPA-SA.
- AR (*Access Router* – Roteador de Acesso) – é o roteador responsável pelo restante do processo de configuração do MN baseado na execução de um protocolo de gerenciamento de túnel para estabelecer o PHT (tratado anteriormente). Dessa forma, todos os pacotes IP transmitidos por meio do PHT deverão ser protegidos por meio de chaves criadas a partir da MPA-SA.
- PoA (*Point of Attachment* – Ponto de Conexão) – elemento de se comunica diretamente com o MN e o AR, podendo estar embutido fisicamente neste último.
- MN (*Mobile Node* – Nó Móvel) – entidade que se encontra em movimento e gera a necessidade de realização do *handover*.

O fluxo básico de mensagens do protocolo MPA pode ser entendido por meio da Figura 4.15:

1. Inicialmente é feita a descoberta das CTN (as comunicações utilizam o endereço de transporte antigo – oCoA – *old Carry-of-Address*). Após isso, tendo sido percebida a queda de sinal (abaixo do limiar 1 configurado) vinculado à conexão com a rede atual, é iniciada a fase de pré-autenticação na qual é criada a MPA-SA que viabiliza a distribuição de chaves criptográficas entre as partes interessadas. Nesse passo são distribuídas as chaves MN-CA *Key* (para a comunicação segura entre o MN e o CA), MN-AR *Key* (análoga à anterior) e

PSK (*Preshared Key* – Chave pré-compartilhada, que viabilizará a troca segura de mensagens entre o nPoA e o MN).

2. Tendo sido constatada uma alta probabilidade de migração para uma dada CTN, executa-se a pré-configuração que tem como principais objetivo a obtenção de um novo endereço de transporte (nCoA – *new CoA*) e o estabelecimento do PHT.
3. Um vez que seja tomada a decisão de se realizar o *handover* para a CTN escolhida, é iniciado o *handover* pró-ativo seguro com a execução da alteração de atribuição (*binding*) por meio do protocolo de gerenciamento de mobilidade e com a utilização do PHT criado anteriormente (já será utilizado o nCoA em substituição ao oCoA).
4. Concluída a alteração de atribuição e estando pronto para realizar a troca de rede, o MN pode executar o protocolo de gerenciamento de túnel para excluir o PHT em uso.
5. Após a exclusão do PHT é iniciado, no nAR (AR da CTN escolhida), o armazenamento temporário dos pacotes em trânsito e, uma vez conectado ao nPoA (*new* PoA), o MN envia um sinal para o nAR a fim de que os pacotes armazenados temporariamente sejam encaminhados para o nPoA.
6. A troca de rede será procedida em seguida e dependerá da política de *handover* utilizada (na figura, a análise está centralizada na razão sinal-ruído – SNR). A partir daí deverá ocorrer um *handover* na camada inferior (camada 2) que poderá embutir um a criação de uma nova SA. Nesse caso, os parâmetros obtidos do processo de pré-autenticação do MPA poderão ser utilizados para minimizar a execução de protocolos completos de autenticação (EAP, por exemplo) resumindo-os a um processo de aperto de mãos de quatro fases (*four way handshake*).
7. Tendo sido concluída a troca de rede, passa-se à fase pós-troca na qual o MN atribui o nCoA à sua interface física que está ligada ao nPoA, o que permite que o nAR pare de armazenar os pacotes.
8. Está concluído o *handover* e os dado podem trafegar normalmente na nova infra-estrutura.

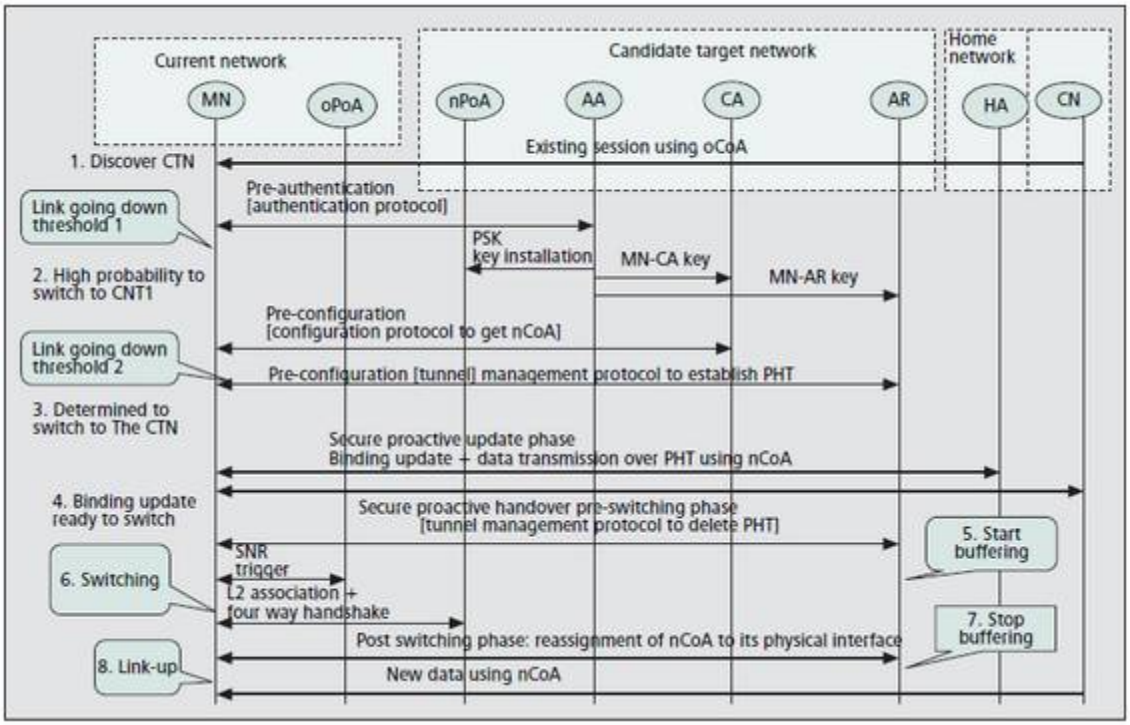


Figura 4.15 – Fluxo de mensagens do MPA (retirado de [Dutta ET AL, 2010])

Fazendo uma ligação com o assunto explorado na seção 4.7.4 (MIH), cabe salientar que o protocolo MPA visto nesta seção pode ter seu papel complementado pelo MIH pois, enquanto o um (MPA) dedica-se mormente à consecução de um *handover* inter-tecnologias pró-ativo seguro, os serviços do outro (MIH) podem fornecer informações valiosas para auxiliar das fases de inicialização e preparação de um *handover*. A integração entre os dois protocolos tem sido alvo de estudos promissores particularmente realizados pelo mesmo grupo de trabalho criado para analisar soluções de segurança para o padrão MIH [Tauil ET AL, 2010].

4.8.2. HOKEY

O HOKEY é uma arquitetura que tem como principal objetivo minimizar o atraso causado pelos processos de autenticação baseados em EAP existentes em *handovers*. Para que isso seja possível, são utilizadas duas abordagens, a autenticação prévia (*early authentication*), a qual permite que a autenticação seja realizada antes do próprio processo de *handover*, e a re-autenticação rápida (*fast reauthentication*), que pode ser aplicada a quaisquer métodos baseados em EAP e na qual é reutilizado material criptográfico gerado na autenticação inicial [Hoper ET AL, 2010].

Em relação à re-autenticação rápida (assunto já tratado na seção 4.5), cabe salientar que em um processo de re-autenticação completa baseada em EAP, ocorrerá a troca de várias mensagens entre o suplicante e o servidor EAP envolvido (fato este que já ocorreu durante a autenticação). O atraso gerado por essa nova troca de mensagens provavelmente interferirá nas transações de dados das quais um equipamento móvel

esteja participando, degradando o serviço e, na pior das hipóteses, forçando a sua interrupção. Esse problema é exatamente o que a re-autenticação rápida busca minimizar.

O problema da autenticação prévia vinculada ao EAP é explorado em detalhes na [RFC 5836] e consiste basicamente na execução de um processo parcial de autenticação (basicamente o estabelecimento de material para criação de chaves a serem utilizadas em uma autenticação EAP) entre um nó móvel (MN) e um ponto de conexão (PoA) potencialmente interessante para a consecução de uma futura conexão, o que viabiliza uma diminuição relevante no processo de autenticação EAP propriamente dito que poderá ocorrer no futuro. Tal qual no caso da re-autenticação rápida, a autenticação prévia também permite uma diminuição no tempo gasto para os processos de autenticação necessários a um *handover*.

A Figura 4.16 ilustra de forma geral processos de autenticação e re-autenticação (na rede de origem – *Home Network* - e em uma rede visitada – *Local Network*) viabilizados por meio do HOKEY. A setas existentes na figura têm o seguinte significado:

1. Execução do método EAP completo entre um MN e a rede de origem da qual faz parte um servidor de AAA (*Home AAA Server*);
2. Distribuição das chaves do servidor AAA para o primeiro AP (*Access Point* - ponto de acesso);
3. Processo de aperto-de-mãos em quatro fases (*four-way handshake*) executado entre o MN e o primeiro AP utilizando o material de chaves recebido em 2;
4. O MN deseja conectar-se ao segundo AP que se encontra em outro domínio administrativo (*handover* INTER-domínios), para tal executa o método ERP (Re-autenticação EAP vista na seção 4.4.2) apoiado pelo servidor de AAA da rede visitada (*Local AAA Server*);
5. O servidor de AAA da rede visitada faz requisições ao servidor de AAA da rede de origem e recebe uma DSRK (*Domain-Specific Root Key* – Chave Raiz de Domínio Específico) a ser utilizada na re-autenticação;
6. O servidor de AAA da rede visitada gera chaves de sessão a partir da DSRK recebida e as entrega para o segundo AP;
7. Processo de *four-way handshake* executado entre o MN e o segundo AP utilizando o material de chaves recebido em 6;
8. O MN deseja conectar-se ao terceiro AP que também se encontra no outro domínio administrativo (*handover* INTRA-domínios), para tal executa o método ERP apoiado pelo servidor de AAA da rede visitada;
9. O servidor de AAA da rede visitada gera novas chaves de sessão a partir da DSRK que já se encontra com ele e as entrega para o terceiro AP;
10. Processo de *four-way handshake* executado entre o MN e o terceiro AP utilizando o material de chaves recebido em 9.

Cabe salientar que, no exemplo explorado, as redes de origem e visitada poderão, além de fazer parte de domínios diferentes, estar baseadas em tecnologias de acesso distintas, o que caracterizará também o *handover* INTER-tecnologias que interessa a esse trabalho

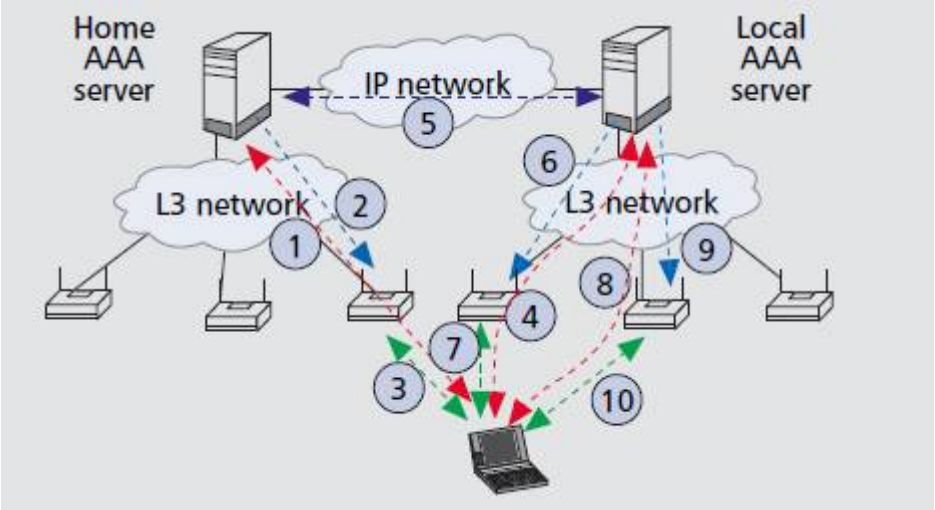


Figura 4.16 – Autenticação e Re-autenticação com HOKEY (retirado de [Clancy, 2008])

4.8.3. PANA

O PANA é descrito pela [RFC 5191] e, diferente de mecanismos constantes de sub-sessões anteriores, é um protocolo executado na camada de rede (baseado em UDP) e viabiliza, nas camadas superiores, a utilização de métodos de autenticação para controle de acesso à rede. De forma mais específica, pode-se dizer que o PANA representa uma camada inferior (modelo conceitual em camadas do EAP – item 4.4.1) sobre a qual funciona o *framework* EAP. Em outras palavras, o PANA "carrega" o EAP (que por sua vez pode "carregar" diversos métodos EAP).

A partir do momento em que viabiliza o emprego do *framework* EAP sobre a camada IP, o PANA permite a execução de quaisquer métodos EAP sobre quaisquer tecnologias de enlace. Dessa forma o PANA é uma ferramenta valiosa no auxilio à execução de *handovers* entre tecnologias de acesso distintas (WLAN e 3G, por exemplo).

A troca de mensagens PANA consiste de uma série de requisições e respostas, tal qual os outros protocolos vistos neste curso, que podem conter vários AVPs (o conceito de AVP já foi explorado na seção **Erro! Fonte de referência não encontrada.** deste trabalho) transportando várias cargas (*payload*). As principais cargas transportadas por esses AVPs são os pacotes EAP.

Por ser baseado em UDP, o PANA precisa implementar seu próprio mecanismo de retransmissão a fim de que haja confiabilidade no processo de troca de mensagens do protocolo.

Uma sessão PANA deve ser executada entre um cliente (PaC – *PANA Client* – Cliente PANA) e um servidor (PAA – *PANA Authentication Agent* – Agente de Autenticação PANA) para que seja possível o fornecimento de autenticação e autorização (AA) aos serviços de controle de acesso a redes. Essa sessão consiste das seguintes fases:

1. Fase de autenticação e autorização - etapa na qual é iniciada uma sessão PANA e onde é executado o método EAP entre o PaC e o PAA (qualquer um dos dois pode iniciar a sessão). Ao término desta fase, o PAA fornece o resultado do processo (AA) para o PaC.
2. Fase de acesso – tendo havido sucesso na fase anterior (AA), o PaC passa a ter acesso à rede almejada e pode realizar seus envios e recebimentos de pacotes de dados normalmente.
3. Fase de re-autenticação – ainda durante a fase anterior (por esse motivo esta fase é considerada uma sub-fase da anterior), o PaC ou o PAA (qualquer um dos dois) podem iniciar a re-autenticação para que seja possível alterar o tempo de vida (*lifetime*) da sessão PANA antes que este expire.
4. Fase de finalização – pode ser, tal qual as anteriores, disparada por qualquer uma das duas entidades envolvidas (PaC ou PAA) a qualquer momento (decisões alheias ao PANA). A entidade interessada no término da sessão remete um sinal (mensagem de desconexão) para a sua parceira e o processo é finalizado.

A Figura 4.17 retrata um cenário típico e utilização do PANA. Neste cenário, inicialmente um MN (PaC) realiza procedimentos de autenticação (utilizando o PANA) a uma rede localizada no domínio A (fase 1 explicada anteriormente). Tendo havido êxito no processo, o MN realiza o acesso à rede e inicia suas transações. Esse MN, desejando movimentar-se para outro domínio (o C no exemplo) mas ainda conectado ao domínio A (acessando a rede original), dispara um processo de re-autenticação (fase 3) direcionado para o domínio C. O contato a ser feito pelo MN com o PAA da rede alvo (localizada no domínio C), aqui representado pelo servidor de AAA do domínio C, deve ser feito de forma direta (PANA não permite, por exemplo, que o PAA atual sirva de *proxy* para o contato com o próximo PAA), entretanto isso não se mostra algo problemático tendo em vista o fato do PANA estar baseado no UDP. Tendo havido sucesso na re-autenticação, o MN pode migrar (*handover*) sem problemas de uma rede para a outra.

Como é possível perceber, o PANA preocupa-se tão somente com os processos de autenticação e re-autenticação, deixando a atividade de *handover* em si para outros protocolos necessariamente suportados pelo MN e demais participantes do processo (salienta-se que o PANA é um dos protocolos propostos para utilização vinculada ao padrão MIH pelo grupo de trabalho que se preocupa com os aspectos de segurança do padrão IEEE 802.21).

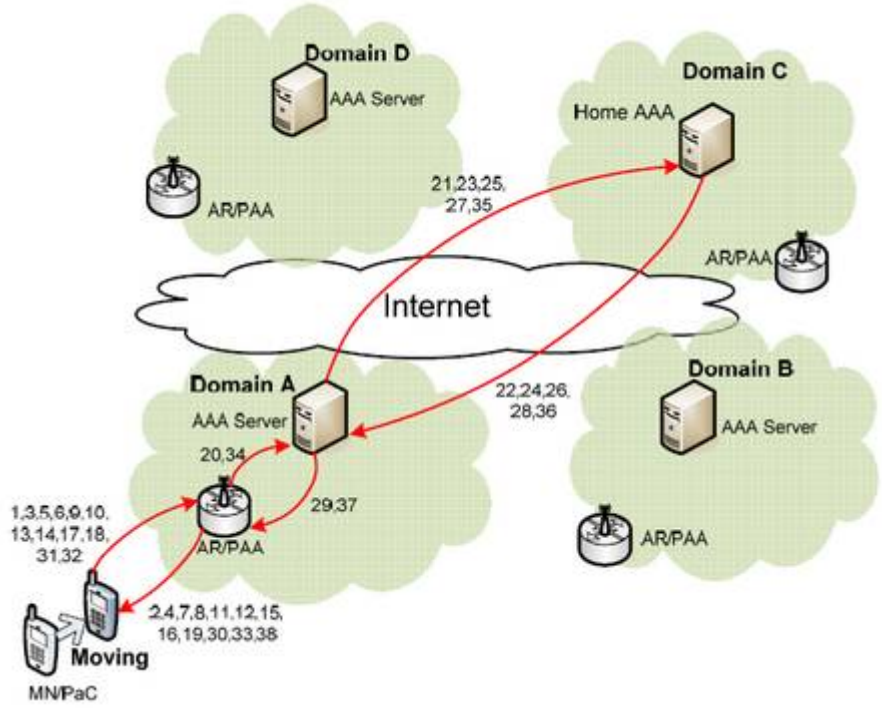


Figura 4.17 – Cenário típico de aplicação do PANA (retirado de [Chamuczynski, 2008])

4.9. Conclusões

O curso encerrado nessa seção teve os objetivos de apresentar aspectos teóricos e práticos ligados à segurança necessária ao *handover* entre redes celulares e WLANs, além de focalizar formas de minimizar a latência gerada pelos processos que viabilizam essa segurança.

Os autores procuraram explorar uma gama bastante grande de assuntos, em muitos casos com pequeno nível de detalhamento, vinculando padrões clássicos a soluções inovadoras e extremamente atuais, com o objetivo de criar no leitor uma idéia ampla e irrestrita das várias áreas ligadas ao assunto-núcleo explorado.

Ao concluir a leitura deste texto, certamente o leitor não poderá se considerar conhecedor profundo de um determinado assunto tratado no curso, entretanto ele poderá ter a certeza de que passou a ter um conhecimento razoável sobre um conjunto bastante grande de assuntos que estão interligados ao tema central. Além disso, ele também perceberá que a palavra chave a ser utilizada na busca da resolução dos problemas abordados é INTEGRAÇÃO. Integrar de forma sinérgica diversas soluções específicas para gerar o resultado esperado.

Especificamente sobre os problemas que foram explorados, pôde-se perceber que, não só no contexto da segurança, mas também em vários outros aspectos (garantia de QoS, modelos de mobilidade, análise de desempenho, etc.), a questão da interconexão entre redes heterogêneas é algo bastante atual e que certamente ainda demandará muito esforço dos diversos elementos envolvidos (comunidade acadêmica, entidades e órgãos

de padronização, fabricantes de equipamentos e provedores de serviços, dentre outros) para que possa ser considerado um problema bem resolvido.

Verificou-se também que, caso as partes integrantes de um processo de comunicação envolvendo usuários, operadoras de redes celulares e provedores de acesso a WLANs não mantenham um relacionamento de confiança entre si, a existência de uma passagem segura entre rede celular e WLAN para ser utilizada por usuários será algo praticamente inviável e, caso venha a ser possível, poderá ser inócua.

Também pode ser extraída do texto a idéia de que a integração entre WLANs e redes celulares, aproveitando aquilo que de melhor elas podem fornecer para o bom provimento de um serviço, é algo muito útil e tem-se mostrado uma tendência de mercado explorada por muitas entidades interessadas.

Em relação ao padrão MIH, foi possível verificar que ele não tem a capacidade de resolver sozinho o problema central tratado neste curso, entretanto pode ser considerado uma ferramenta preciosa que, caso integrada a outros protocolos com atribuições específicas (MPA, por exemplo), gerará resultados importantíssimos no rumo das soluções procuradas.

Sobre os processos de autenticação, pôde-se perceber que, apesar de sua importância incontestável, eles ainda são grandes "vilões" quando o assunto tratado é a diminuição de tempo de um *handover*. Apesar disso, também foi possível observar que muitos esforços têm sido feitos para a proposição de melhorias e adaptações a serem aplicadas nesses métodos de autenticação. Algumas melhorias utilizam idéias de reuso de material criptográfico criado em momentos passados, outras propõem a realização de processos em momentos anteriores ao *handover*, outras buscam aproveitar ambos os benefícios. De qualquer forma, independente do maior ou menor êxito, fica patente o entendimento e a busca de soluções por parte dos inúmeros pesquisadores do assunto.

Também foi possível constatar que os problemas de segurança realmente interferem na qualidade dos serviços prestados por meio de redes sem fio. Sob qualquer prisma, analisando o enfoque da latência gerada pelos processos que se propõem a resolver esses problemas, ou observando as próprias conseqüências danosas advindas da não aplicação dos mecanismos de segurança, fica claro que a não existência, bem como o excesso de segurança, certamente gerarão conseqüências indesejáveis para os serviços fornecidos pelas redes sem fio.

Nesse contexto, percebe-se que ainda há muito trabalho a ser feito e que o grande desafio não é criar uma solução extremamente segura, nem uma tremendamente rápida, mas sim uma que viabilize a segurança necessária e suficiente sem ser "agressiva", ou seja, que não prejudique o próprio "bem" que está sendo protegido: a informação oportuna.

Referências

Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed. (2004). "Extensible Authentication Protocol (EAP)". RFC 3748, June 2004.

Aboba, B., Calhoun, P.(2003). "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)". RFC 3579, September 2003.

Aboba, B., Simon, D., Eronen, P.(2008). "Extensible Authentication Protocol (EAP) Key Management Framework". RFC 5247, August 2008.

Arkko, J., Haverinen, H.(2006). "Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA)". RFC 4187, January 2006.

Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.(2003). "Diameter Base Protocol", RFC 3588, September 2003.

Chamuczynski, P., Alfandi, O., Werner, C., Brosene, H., Hogrefe, D.(2008). "Performance Study of PANA Pre-authentication for Interdomain *Handover*". In: Fourth International Conference on Networking and Services, ICNS 2008, March 2008.

Clancy, T.(2008). "Secure *Handover* in Enterprise WLANs: CAPWAP, HOKEY, and IEEE 802.11r". In: IEEE Wireless Communications, vol. 15, nr. 5, October 2008.

Clancy, T., Nakhjiri, M., Narayanan, V., Dondeti, L.(2008). "*Handover* Key Management and Re-Authentication Problem Statement". RFC 5169, March 2008.

Congdon, P., Aboba, B., Smith, A., Zorn, G., Roese, J.(2003). "IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines". RFC 3580, September 2003.

Dierks, T., Rescorla, E.(2008). "The Transport Layer Security (TLS) Protocol Version 1.2". RFC 5246, August 2008.

Dutta, A., Fajardo, V., Ohba, Y., Taniuchi, K., Schulzrinne, H.(2010). "A Framework of Media-Independent Pre-Authentication (MPA) for Inter-domain *Handover* Optimization". draft-irtf-mobopts-mpa-framework-07, April 2010, trabalho em andamento.

Dutta, A., Famolari, D., Das, S., Ohba, Y., Fajardo, V., Taniuchi, K., Lopez, R., Schulzrinne, H.(2008). "Media-Independent Pre-Authentication Supporting Secure Interdomain *Handover* Optimization". In: IEEE Wireless Communications, vol. 15, nr. 2, April 2008.

Eronen, P.(2006). "IKEv2 Mobility and Multihoming Protocol (MOBIKE)". RFC 4555, June 2006.

ETSI TS 133 234 (2010). "Universal Mobile Telecommunications System (UMTS); LTE; 3G security; Wireless Local Area Network (WLAN) interworking security". V. 9.2.0, Release 9, July 2010

Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., Yegin, A.(2008). “Protocol for Carrying Authentication for Network Access (PANA)”. RFC 5191, May 2008.

Funk, P., Blake-Wilson, S.(2008). "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)". RFC 5281, August 2008.

Haverinen, H., Ed., and J. Salowey, Ed.(2006). “Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)”. RFC 4186, January 2006.

Hoeper, K., Decugis, S., Zorn, G., Wu, Q., Taylor, T.(2010). “*Handover* Keying (HOKEY) Architecture Design”. draft-hoeper-hokey-arch-design-03, July 2010, trabalho em andamento.

Hoeper, K., Nakhjiri, M., Ohba, Y.(2010). “Distribution of EAP-Based Keys for *Handover* and Re-Authentication”. RFC 5749, March 2010.

IEEE Standard 802.11 (2007). “Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”.

IEEE Standard 802.1X (2004). “IEEE Standard for Local and metropolitan area networks Port-Based Network Access Control”.

IEEE Standard 802.21 (2008). “IEEE Standard for Local and metropolitan area networks - Part 21: Media Independent *Handover* Services”.

Johnson, D., Perkins, C., Arkko, J.(2004). “Mobility Support in IPv6”. RFC 3775, June 2004.

Josang, A., Ismail, R., Boyd, C.(2007). “A survey of trust and reputation systems for online service provision”. In: Decision Support Systems, vol. 43, nr. 2. Elsevier Science Publishers B. V., March 2007.

Kaufman, C.(2005). “Internet Key Exchange (IKEv2) Protocol”. RFC 4306, December 2005.

Koien, G. M., Haslestad, T.(2003). “Security Aspects of 3G-WLAN Interworking”. In: IEEE Communications Magazine, vol. 41, nr. 11, November de 2003.

Liu, J., Jiang, S., Lin, H.(2006). “Introduction to Diameter – Get the next generation AAA protocol”. Available in http://www.ibm.com/developerworks/wireless/library/wi-diameter, April 2006, Access: July 2010.

Machan, P., Serwin, S., and Wozniak, J.(2008). “Performance of mobility support mechanisms in a heterogeneous UMTS and IEEE 802.11 network offered under the IEEE 802.21 standard”. In: 1st International Conference on Information Technology, pages 1–4. IEEE. 2008

Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.(2008). “Host Identity Protocol”, RFC 5201, April 2008.

Munasinghe, K. and Jamalipour, A.(2007). “A 3GPP-IMS based approach for converging next generation mobile data networks”. In: International Conference on Communications, pages 5264–5269. IEEE. 2007.

Munasinghe, K. and Jamalipour, A.(2008). “Interworking of WLAN-UMTS networks: an IMS-based platform for session mobility”. In: IEEE Communications Magazine, Vol. 46, nr. 9, IEEE, 2008.

Narayanan, V., Dondeti, L.(2008). "EAP Extensions for EAP Re-authentication Protocol (ERP)". RFC 5296, August 2008.

Ohba, Y., Wu, Q., Zorn, G.(2010). “Extensible Authentication Protocol (EAP) Early Authentication Problem Statement”. RFC 5836, April 2010.

Perkins, C.(2002). “IP Mobility Support for IPv4”. RFC 3344, August 2002.

Postel, J.(1980). “User Datagram Protocol”. RFC 768, August 1980.

Postel, J.(1981). “Transmission Control Protocol”. RFC 793, September 1981.

Rigney, C., Willens, S., Rubens, A., Simpson, A.(2000). “Remote Authentication Dial In User Service (RADIUS)”. RFC 2865, June 2000.

Romkey, J.(1988). “A Nonstandard for Transmission of IP Datagrams Over Serial Lines: SLIP”. RFC 1055, June 1988

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.(2002). “SIP: Session Initiation Protocol”. RFC 3261, June 2002.

Salowey, J., Dondeti, L., Narayanan, V., Nakhjiri, M.(2008). “Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK)”. RFC 5295, August 2008.

Schneier, B. (1996). "Applied Cryptography: Protocols, Algorithms, and Source Code in C". Second Edition. John Wiley & Sons.

Silva, A., Endler, M. Colcher, S.(2008). “Otimização do *Handover* na Camada de Rede (L3) utilizando o Media Independent *Handover* (MIH)”. Tese não publicada. Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, Rio de Janeiro. 2008.

Simon, D., Aboba, B., Hurst, R.(2008). “The EAP-TLS Authentication Protocol”. RFC 5216, March 2008.

Simpson, W.(1994). “The Point-to-Point Protocol (PPP)”. RFC 1661, July 1994

Song, W., Jiang, H., and Zhuang, W.(2007). “Performance analysis of the wlan-first scheme in cellular/wlan interworking”. In: IEEE Transactions on Wireless Communications, Vol. 6, nr. 5, IEEE, 2007.

Stanley, D., Walker, J., Aboba, B.(2005). “Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs”. RFC 4017, March 2005.

Tauil, M., Dutta, A., Cheng, Y., Das, S., Baker, D., Yajnik, M., Famolari, D., Ohba, Y., Taniuchi, K., Fajardo, V., Schulzrinne, H.(2010). “Integration of IEEE 802.21

services and pre-authentication framework". In: International Journal of Communication Networks and Distributed Systems, vol. 5, nr.1/2, February 2010.

Yang, P., Deng, H.(2007). "Seamless integration of 3G and 802.11 wireless network". In: 5th ACM international workshop on Mobility management and wireless access, pages 60–65. ACM. 2007.

Yusof, A. L., Ismail, M., and Misran, N.(2007). "Architecture and mobility management protocols for next-generation wireless systems (NGWS)". In: IEEE International Conference on Telecommunications and Malaysia International Conference on Communications, pages 747–752. IEEE, 2007

Zorn, G.(1999). "Microsoft Vendor-specific RADIUS Attributes". RFC 2548, March 1999.

Zorn, G., Cobb, S.(1998). "Microsoft PPP CHAP Extensions". RFC 2433, October 1998.

Capítulo 5

Introdução à segurança de aplicações para a TV digital interativa brasileira

Alexandre Melo Braga[1], Gilmara Santos Restani[1]

[1]Fundação CPqD – Centro de Pesquisa e Desenvolvimento em Telecomunicações. Rod. Campinas-Mogi-Mirim (SP-340) km 118,5 – 13086-902 – Campinas – SP – Brazil.

ambraga@cpqd.com.br, grestani@cpqd.com.br

Abstract

This chapter contains an overview of the security aspects of the Brazilian interactive digital TV. The text is structured around technical security assessments of set top boxes and shows preliminary results obtained by the contributing authors on security assessments of interactive applications for these devices. The text deals with the aspects of secure programming, secure execution environment, and security of receivers, when connected to an IP network. Further, several vulnerabilities found in the software technologies associates to the TVD receivers are shown. The vulnerabilities identified and documented in this text have parallel in known vulnerabilities found in other programming languages and traditional vulnerabilities catalogs.

Resumo

Este capítulo descreve resultados preliminares obtidos pelo CPqD na avaliação de segurança de aplicações interativas para a TV digital brasileira. Foram avaliados os aspectos de programação segura, ambiente de execução segura dos aplicativos e segurança de receptores conectados a uma rede IP. Foi iniciado um trabalho de identificação e documentação de vulnerabilidades de programação insegura, de modo que vulnerabilidades em Lua encontrem paralelo em vulnerabilidades já conhecidas em outras linguagens de programação.

5.1. Introdução

No Brasil, os receptores de televisão digital interativa (TVDi) tendem a se tornar instrumentos de inclusão digital para grandes parcelas da população. Nos próximos anos, diversos serviços serão oferecidos via televisão digital. Muitos dos quais somente serão possíveis com o atendimento a requisitos de segurança fortes. Para atender a demanda prometida, os receptores de televisão digital, enquanto equipamentos de computação e comunicação possuirão capacidade computacional equivalente a dos computadores pessoais mais simples; ou ainda, a dos telefones celulares mais sofisticados.

O desafio em relação à plataforma de software de TVDi brasileira está em oferecer pelo menos o mesmo grau de confiança de que gozam as plataformas mantidas pelos gigantes da indústria de software. Por outro lado, as implementações das linguagens de programação da plataforma de TVDi ainda não sofreram o mesmo grau de escrutínio da comunidade científica de segurança que as implementações de linguagens utilizadas pelos grandes produtores mundiais de software. Além disso, por ainda não serem tão universais quanto os sistemas de mercado, elas também não sofreram o mesmo grau de exposição às ameaças comuns aos sistemas de Internet, de computação doméstica e corporativa.

Este trabalho foi motivado pela ausência, até onde vai o conhecimento dos autores, tanto de documentação técnica especializada, quanto de ferramentas de mercado, seja para programação segura, seja para proteção de aplicações interativas, na TVDi brasileira, em particular para as tecnologias da plataforma Ginga. A lista bibliográfica reflete a ausência de textos técnicos e científicos nesta área. Vale ressaltar que não foram encontradas, até o momento de submissão deste minicurso, referências bibliográficas sobre desenvolvimento seguro e nem sobre programação segura em Ginga e TVDi.

O texto a seguir está organizado da seguinte forma. A seção 5.2 descreve de forma resumida a plataforma da televisão digital brasileira, descreve as aplicações interativas e as camadas do *middleware* Ginga embarcado em um receptor de tv digital. A seção 5.3 aborda a segurança do receptor de TVDi de acordo com a segurança de sistemas embarcados, a segurança nos dispositivos portáteis, e trata os receptores de TVDi como plataforma de computação confiável para transações comerciais. E de forma resumida mostra a iniciativa de normatização dos mecanismos de segurança de aplicativos no sistema brasileiro de televisão digital. Na seção 5.4 é apresentado um catalogo resumido das vulnerabilidades de programação insegura identificadas nas linguagens de programação utilizadas pelo Ginga, em particular, o NCLua. A seção 5.5 mostra, com ferramenta construída para tal, mas em ambiente simulado, como realizar avaliações de segurança em receptores de TVDi com Ginga. A seção 5.6 traz uma abordagem sistêmica para a segurança da informação e dos sistemas embarcados nos receptores de TVDi onde serão tratados aspectos de desenvolvimento seguro de aplicações, em particular aspectos de projeto seguro de sistemas, programação segura, robustecimento de sistemas operacionais, segurança em redes e implantação e operação seguras. Todos estes aspectos de segurança serão contextualizados na utilização do receptor de TVDi como dispositivo de computação e comunicação. E por fim a seção 5.7 apresenta as considerações finais e trabalhos futuros.

5.2. Visão Geral da TV Digital interativa brasileira e suas aplicações

Esta seção contém uma visão geral bastante resumida da plataforma de televisão digital brasileira. A maior parte do conteúdo será voltada para a descrição da arquitetura de software de apoio às aplicações interativas, as camadas de software tipicamente embarcadas em um receptor de TVDi (*set-top box*), o *midlleware* ginga, e a estrutura das aplicações interativas, Ginga-J, Ginga-NCL, Lua e a ponte NCLua.

Um pouco da história da experiência do CPqD - Centro de Pesquisa e Desenvolvimento com a TV Brasileira. Desde da década de 90, o CPqD vem assessorando o Governo Federal e os organismos competentes (ANATEL e Ministério das Comunicações) de forma geral, na definição do modelo de TV digital brasileiro. A vasta experiência e elaborações de documentos no cenário da TV digital fazem com que seus colaboradores se preocupem cada vez mais com a questão de adequação de padrões e regulamentações nos componentes deste cenário. Desta maneira, uma breve descrição do cenário de TV digital se faz necessária.

O sistema atual de televisão digital na grande maioria do país ainda é híbrido, ou seja, analógico-digital. Mas, este cenário já começou a ser alterado e algumas operadoras de TV aberta já estão transmitindo o sinal digital. Para a recepção da sinalização digital é utilizado um receptor, aparelho responsável pela conversão destes sinais. Em algumas cidades, a produção, a edição e o armazenamento de conteúdo são quase inteiramente digitais.

Desta forma, o usuário final é beneficiado com: uma qualidade superior de áudio e vídeo; o transporte digital dos sinais, os quais permitem uma recepção mais limpa e menos suscetível às interferências, inclusive em receptores móveis e portáteis. E ainda, a diversidade de programação é maximizada com a utilização de padrões do grupo MPEG- *Moving Picture Experts Group*. Soma-se a todos estes benefícios a característica potencialmente mais interessante, a distribuição de diversos serviços interativos agregados à programação, tais como: eventos esportivos com câmeras, placar em tempo real, legendas e áudio em vários idiomas, informativos de previsão de tempo e indicativos do mercado financeiro entre outras possibilidades. A facilidade de um canal de retorno entre o usuário e a emissora permite outros serviços mais completos como transações bancárias, por exemplo.

A tecnologia de TV digital cria um novo ambiente, onde a experiência de assistir televisão é transformada, ou seja, é possível uma interatividade do usuário final, o qual pode ter um papel ativo perante o remetente dos dados via canal de retorno. Este novo ambiente permite um cenário com diversas possibilidades, oportunidades e desafios e entender este processo e a tecnologia envolvida na disponibilização desses serviços interativos é fundamental para o entendimento do foco deste trabalho.

A arquitetura genérica comum aos sistemas de TV digital basicamente é uma plataforma multimídia capaz de transmitir sinais de áudio e vídeo de alta qualidade, e também dados, utilizando o sinal de radiodifusão em freqüências de VHF – *Very High Frequency*/UHF – *Ultra High Frequency*. A capacidade de transmissão de dados, podendo ou não estar vinculados à programação, permite o desenvolvimento de novos serviços e aplicações.

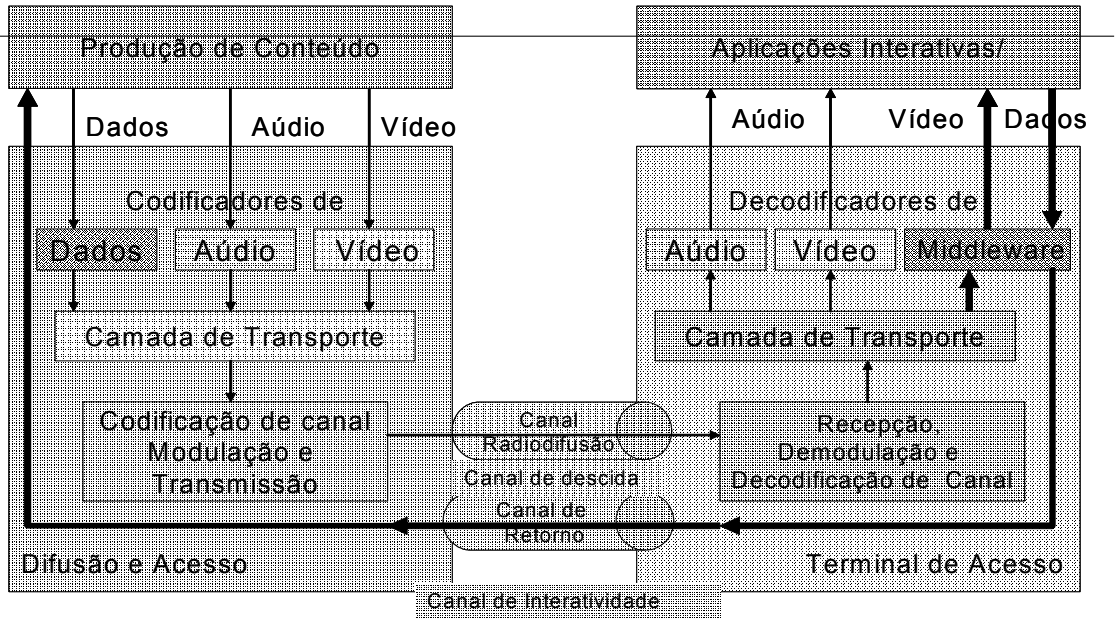


Figura 1 - Diagrama de Fluxo de Informação.

A Figura 1 mostra de forma sintetizada a arquitetura sistêmica do modelo de TV digital Aberta através da representação do fluxo de informação. A parte de Difusão e Acesso, do lado esquerdo da figura, é composta pelos módulos de codificação e empacotamento das informações a serem transmitidas para os receptores digitais. O lado direito da figura é à parte do Terminal de Acesso, ou seja, o receptor de TVDi , foco deste trabalho, e é composta por módulos necessários para efetuar o processamento reverso da parte de Difusão e Acesso, reconstituindo as informações originais de áudio, vídeo e dados. O sinal é recebido por meio de antenas receptoras, e passa pelo processo de demodulação de canal, de onde resulta o sinal de transporte que será enviado à etapa de demultiplexação, no módulo da Camada de Transporte. Nesta camada os sinais são codificados em áudio e vídeo e são submetidos aos respectivos decodificadores, e os dados são submetidos ao *middleware*. Os primeiros reconstituem os sinais originais. No caso do *middleware*, além do trato das instruções, funciona também como uma plataforma de execução de software e será descrito mais à diante. E como resultados finais têm-se as Aplicações Interativas, utilizadas pelo usuário final.

Além disto, o sistema de TV digital Aberta possui um Canal de Interatividade, composto de um Canal de Descida e um Canal de Retorno, o qual possibilita a interação do usuário final com a Produção de Conteúdo, permitindo-lhe receber ou enviar solicitações e informações. Neste trabalho, o foco é o terminal de acesso e a seguir especificamente serão mais detalhados o Canal de Interatividade e o *Middleware*.

O Canal de Interatividade é o subsistema de TV digital responsável pela infraestrutura para a comunicação das Aplicações Interativas, do Terminal de Acesso com os servidores de aplicação do Provedor de Conteúdo. Este canal é conceituado como o meio que o usuário pode interagir encaminhando ou recebendo dados/programações das emissoras de TV. Este canal é formado por dois canais de comunicação: Canal de Descida – o qual estabelece a comunicação via radiodifusão das emissoras para o usuário final; e o Canal de Retorno – compreendido por qualquer tecnologia de redes de acesso que estabeleça a comunicação IP (*Internet Protocol*) no sentido do usuário para

as emissoras. No lado do receptor é possível perceber as diferentes camadas que compõe esta arquitetura conforme mostra a Figura 2.

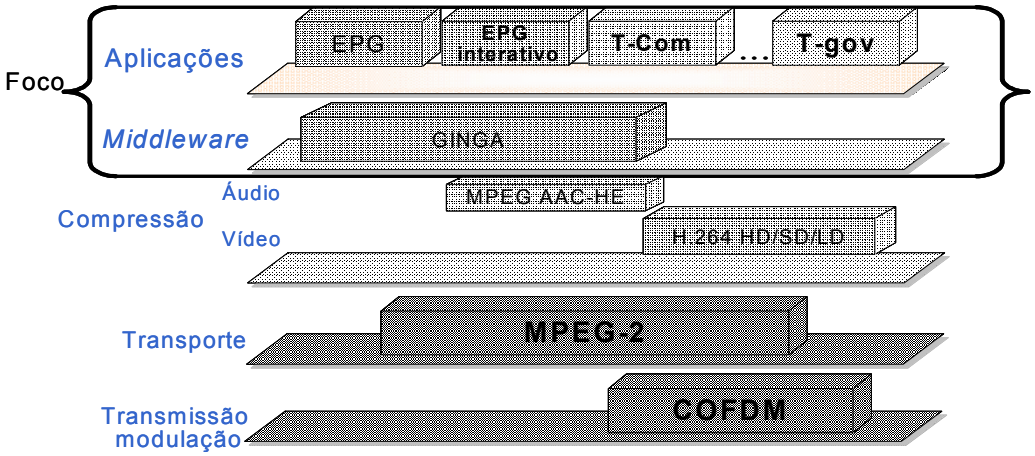


Figura 2- Canal de Interatividade

Na Figura 3 a seguir é possível verificar que o receptor de TVDi recebe o canal de radiodifusão de diversas emissoras/programas, que contém dados de interatividade do Canal de Descida no feixe de transporte (TS). Para a interface entre o receptor de TV digital e as Redes de Comunicações é necessário um modem adequado à solução de rede alocada pelo usuário final.

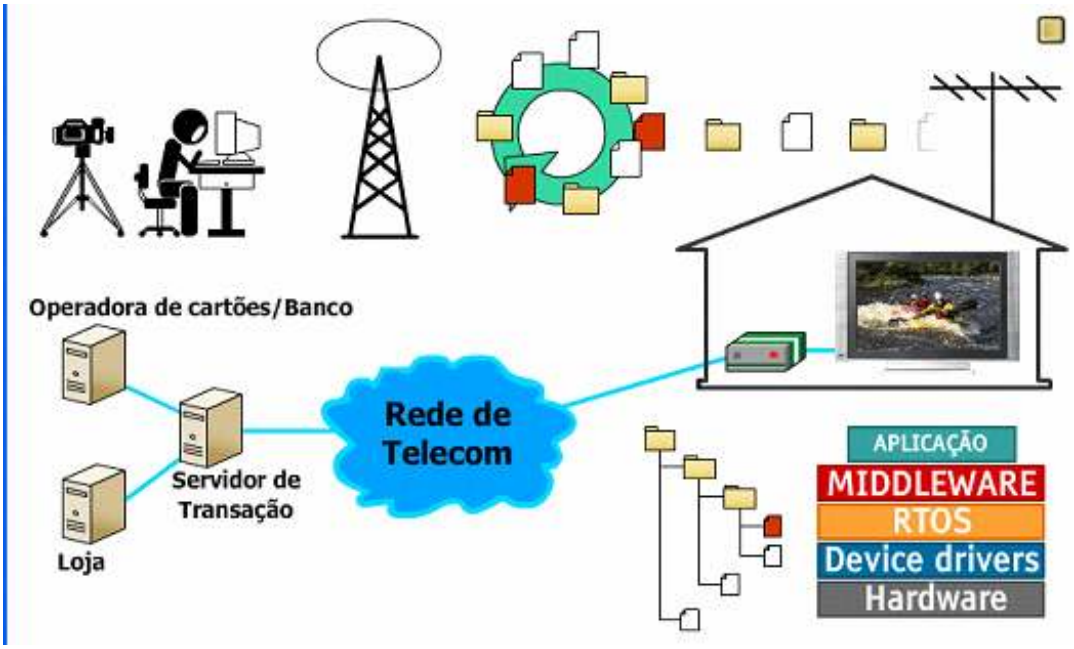


Figura 3 - TV Digital e seus componentes.

Na Figura 3 também é possível verificar a presença de aplicações no desenho menor representando o carrossel. O desenho demonstra o meio de distribuição do conjunto de dados no sistema de radiodifusão denominado *Object Carousel*. Os dados são

transmitidos como um conjunto de objetos de arquivos e diretórios (File e *Directory objects*). O *File Object* é o próprio conteúdo do arquivo que se deseja transmitir. Ele é composto de referências (via ponteiros) a outros arquivos ou diretórios e a representação dos dados ocorre de forma hierárquica. Cada objeto presente no *Object Carousel* é transmitido em uma única messagem chamada BIOP (*Broadcast Inter ORB Protocol*). Basicamente os princípios do DSM-CC (*Digital Storage Media – Command & Control*) são: *Data Carousel*: este é utilizado na transmissão de blocos de dados, não havendo nenhuma indicação do significado do dado transmitido, ficando a interpretação por conta do receptor. E o princípio do *Object Carousel* é construído no topo do carrossel sendo similar a um sistema de arquivos.

A seguir será detalhado o *Middleware* SBTVD presente na solução de TV digital aberta.

Segundo Soares o *middleware* SBTVD é uma camada de software posicionada entre os códigos das aplicações e a infraestrutura de execução (plataforma de hardware e sistema operacional) como ilustrado na Figura 4. O *middleware* no padrão brasileiro consiste de máquinas de execução das linguagens oferecidas, e bibliotecas de funções, que permitem o desenvolvimento rápido e fácil de aplicações.

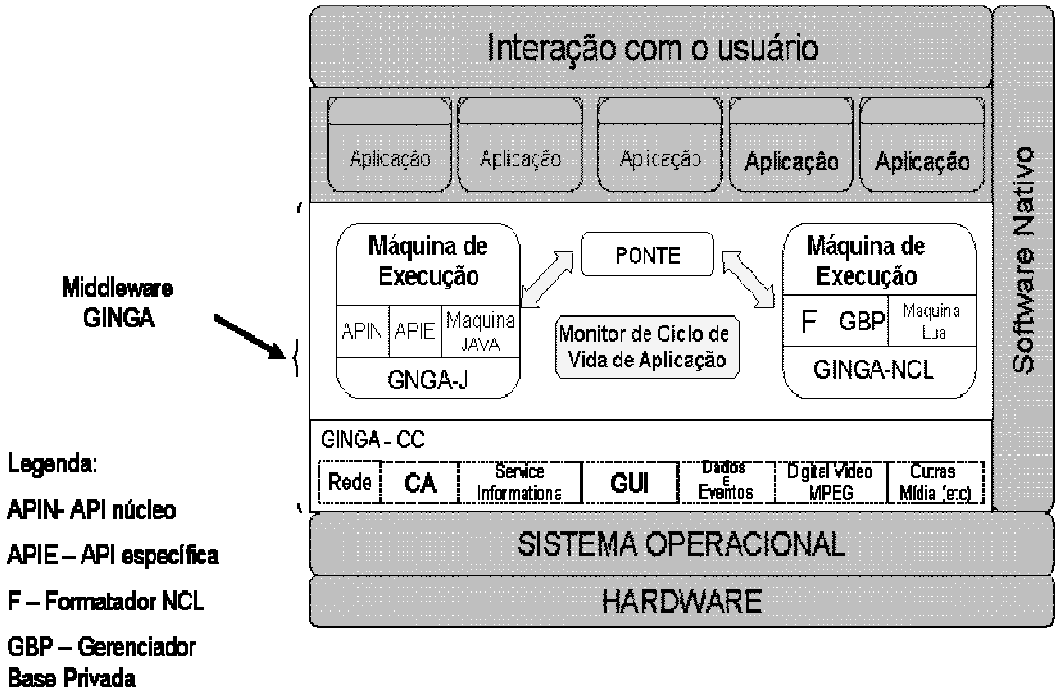


Figura 4 - Arquitetura *Middleware* SBTVD – GINGA.

O universo das aplicações pode ser dividido em três módulos principais: Ginga-CC, Ginga-NCL e Ginga-J, sendo que os dois últimos compõem a camada de serviços específicos e estão separados pelo tipo de aplicações que são responsáveis. O GINGA-NCL é responsável pelas Aplicações Declarativas e o GINGA –J pelas Aplicações Interativas.

A máquina de apresentação declarativa é baseada na linguagem NCL –*Nested Context Language*, uma linguagem declarativa para autoria de documentos multimídia e trata de interpretação semântica do modelo NCM – *Nested Context Model*. Este último é um modelo conceitual e descreve as estruturas de dados, os eventos e o relacionamento entre estes dados, e também definem as regras de estruturação e operações sobre os dados para a manipulação e atualização das estruturas. O Ginga-J é responsável pela maquina da apresentação imperativa e utiliza a linguagem Java. O Ginga-J está dividido em três módulos: a máquina virtual Java; o núcleo e suas APIs(verde); e o módulo responsável pelo suporte às APIs específicas (amarela e vermelha) do Ginga-J.

O *middleware* brasileiro deve apresentar as duas máquinas de aplicações declarativas e imperativas nos receptores fixos e móveis, sendo somente exigida a maquina declarativa em receptores portáteis.

5.3. Cenário de ameaças à TVDi e às aplicações interativas

Esta seção aborda a segurança do receptor de TVDi de acordo com três linhas de atuação. A primeira é a segurança de sistemas embarcados. A segunda é a segurança de dispositivos portáteis, mas com grande poder de computação e comunicação (similares aos *smartphones*). A terceira trata os receptores de TVDi como plataforma de computação confiável para transações comerciais. Além disso, será tratada a iniciativa de normatização dos mecanismos de segurança de aplicativos no sistema brasileiro de televisão digital. As ameaças serão avaliadas no contexto de aplicações com requisitos de segurança fortes, como por exemplo, o oferecimento de serviços bancários pela TVDi (*t-banking*) e de comércio eletrônico pela TVDi (*t-commerce*) et al.

Esta seção contém a análise das questões de segurança relacionadas ao receptor de TVDi e a linguagem Lua. A Figura 5 é uma extrapolação de figura análoga de *Ravi*, et al (2004) e ilustra as necessidades de segurança de cada participante da cadeia de valor da TVDi, desde a produção de hardware, passando pela geração de conteúdo televisivo, até o oferecimento de serviços interativos, vinculado à programação, para o usuário final. Na Figura 5, no lado esquerdo (borda quadrada) estão as ameaças mais comuns associadas a cada nível da pilha de valor. No lado esquerdo, o requisito de segurança correspondente à ameaça em questão.

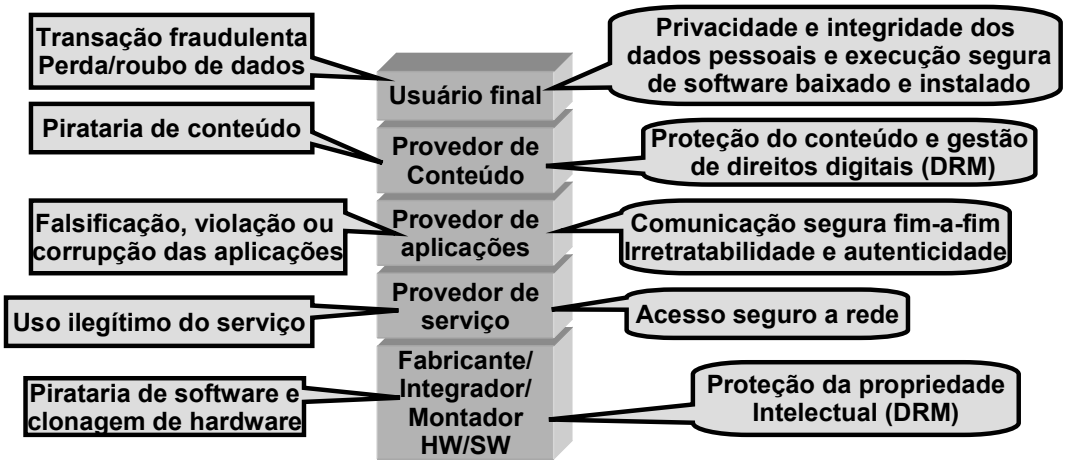


Figura 5 - Necessidades de segurança em cada nível da pilha (cadeia) de valor agregado do sistema de TVDi.

O trabalho apresentado neste texto é principalmente voltado para proteção das aplicações (camada 3 da pilha de valor) e da privacidade e integridade dos dados de usuários finais (camada 5).

5.3.1. Semelhanças de segurança de *smartphones* e de receptores de TVDi

Quando confrontadas por problemas semelhantes, as indústrias de receptores de TVDi e de *smartphones* responderam com soluções semelhantes. Esta seção traça paralelos entre as técnicas de ataque aos *smartphones* e aos receptores TVDi. Assim como também faz analogias entre as estratégias de seguranças adotadas por estas indústrias.

5.3.1.1 Evolução das técnicas de ataque aos *smartphones* e aos receptores TVDi

A tendência mundial de evolução das técnicas de ataque aos *smartphones* e às redes subjacentes de telecomunicações pode ser comparada ao estado das técnicas de ataque sobre PCs conectados a Internet e, mais recentemente, nos aparelhos de telefones celulares inteligentes (*smartphones*), e os receptores de televisão digital interativa. Conforme ilustrado pela Figura 6.

No caso dos PCs, foi na década de 80 que a evolução das técnicas de ataque, em particular, os softwares maliciosos, começou a afetar o público em geral. Naquela época, os espécimes de softwares maliciosos típicos eram os vírus de computador, os quais contaminavam o setor de boot dos discos rígidos e disquetes. Atualmente existem redes inteiras comprometidas e controladas pelo crime organizado para realização dos mais diversos tipos de ataques, incluindo os ataques maciços coordenados pelas *Botnets*. Este texto não detalha a história dos vírus de computador, por se tratar de informação pública e amplamente disponível.

De acordo com Hypponen (2007), o primeiro vírus de *smartphone* surgiu em 2004, desde então há relatos de centenas de softwares maliciosos para estes aparelhos, conforme Bickford, et al (2010). Os ataques mais sofisticados às redes via *smartphones*, com o controle remoto de diversos aparelhos para a realização de ataques coordenados, já são considerados, em textos acadêmicos, uma possibilidade técnica e seriam possíveis em teoria, conforme Traynor, et al (2009). Segundo *Oberheide and Jahanian* (2010),

Oberheide, et al (2008) e Cai, *Machiraju and Chen* (2009), os *smartphones* serão uma fonte de vazamento de informações privadas em redes públicas e ambientes de computação em nuvem, serão ainda vetores de ataques maciços às redes de computadores e de telecomunicações, assim como representarão a próxima fronteira de proliferação dos softwares maliciosos.

De forma análoga ao que tem ocorrido com os PCs e ao que ocorre com os *smartphones*, espera-se ainda nesta década um aumento significativo da quantidade de incidentes de segurança envolvendo os receptores de TVDi.

Ainda, os estudos, experimentos e trabalhos correlatos realizados pelos autores deste texto levam a crer que a semelhança entre os aspectos de segurança de *smartphones* e de receptores de TVDi vão além do paralelismo das técnicas de ataque, abrangendo também, com grande similaridade, as técnicas de segurança e proteção. Conforme descrito na próxima seção.

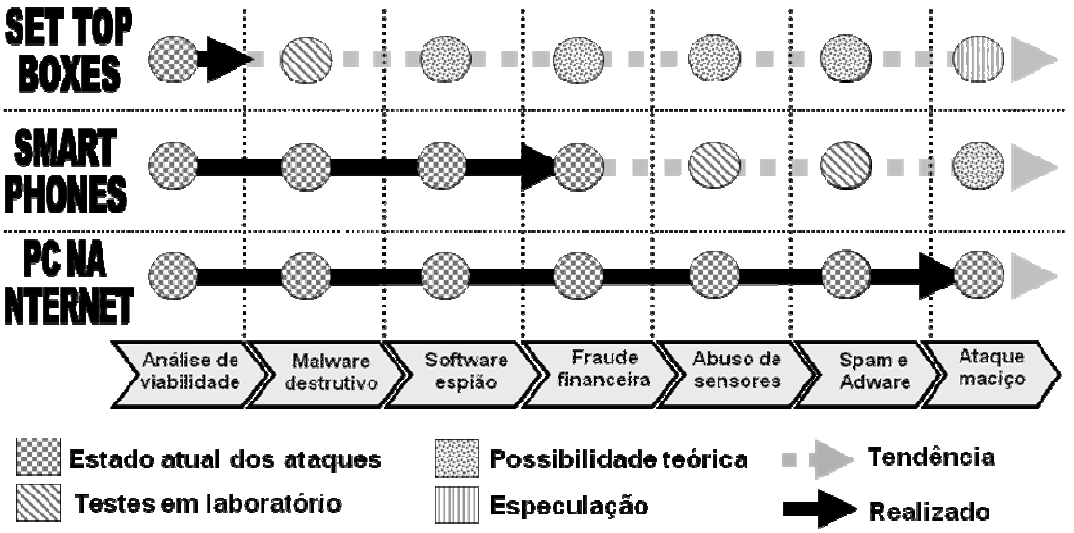


Figura 6 - Paralelo de evolução das técnicas de segurança em três tipos de tecnologias, computadores pessoais conectados a Internet, smartphones com grande poder computacional e receptores de TV digital interativa.

5.3.1.2 Analogias na implementação de mecanismos de segurança

A aderência a normas e padrões é uma parte importante da garantia de integridade de qualquer peça de tecnologia. O receptor de TVDi, em relação aos aspectos de segurança, está sujeito a norma ABNT NBR 15605. Esta norma especifica os mecanismos do sistema de segurança para o sistema brasileiro de televisão digital terrestre. A primeira parte desta norma, ABNT NBR 15605-1 (2008), trata de controle de conteúdo; isto é, DRM (*Digital Rights Management*).

Na segunda parte desta norma, ABNT NBR 15605-2 são definidos os mecanismos de autenticação dos receptores, dos dispositivos externos e de usuários, além das questões de segurança e autenticação de aplicativos interativos, assim como do canal de interatividade.

Para *Oberheide* e *Jahanian* (2010), as plataformas de software modernas para *smartphones* implementam uma arquitetura de segurança baseada em três pilares. Primeiro, a entrega segura de aplicações, a qual está relacionada à habilidade de uma plataforma móvel em verificar a integridade e a autenticidade de origem de uma aplicação a ser instalada no dispositivo móvel. Segundo níveis de confiança, que determinam graus de segurança e privilégios e são implementados por mecanismos de controle de acesso. Terceiro, o isolamento de aplicações e do Sistema Operacional, que se refere à habilidade de uma plataforma móvel em isolar ou conter uma aplicação em particular, como uma estratégia de prevenção contra o comprometimento de outras aplicações ou o próprio S.O.

O modelo de segurança adotado pela norma ABNT NBR 15605-2 não é fundamentalmente diferente daquele adotado pelos fabricantes de dispositivos móveis, os três pilares citados anteriormente. Não por acaso, a norma brasileira de segurança de aplicações interativas para TV digital oferece uma arquitetura de segurança semelhante em diversos aspectos à arquitetura de segurança das plataformas móveis modernas.

De modo análogo ao oferecido pelas plataformas de aplicativos para dispositivos móveis, um receptor interativo, com *middleware* Ginga, aderente a norma brasileira, poderá possuir no mínimo as seguintes características de segurança: autenticação de usuário, segurança no canal de interatividade com SSL/TLS e suporte à autenticação de aplicativos com o mecanismo de controle de acesso correspondente.

Em relação à segurança na distribuição das aplicações. A norma ABNT NBR 15605-2 estabelece que as aplicações disponíveis pelo carrossel sejam autenticadas. A autenticação de aplicações será feita com assinaturas digitais, onde a entidade responsável pela assinatura da aplicação possuiria um certificado de identidade ICP-Brasil e o próprio processo de assinatura seria semelhante à assinatura de código móvel da plataforma Java.

O mecanismo de controle de acesso defendido pela norma tem semelhanças com o modelo mandatório. A autonomia de uma aplicação interativa, assinada ou não, sempre será limitada às restrições de acesso do receptor de TV digital. Uma aplicação não assinada ou não autenticada nunca poderá solicitar a modificação das restrições impostas pelo receptor para ampliar sua autonomia. Já uma aplicação assinada e autenticada poderia solicitar permissões extras, mas ainda assim estaria limitada a no máximo o conteúdo de um arquivo de requisição de permissões (certificado de atributos),emitido e assinado por entidade reconhecida.

A implementação do mecanismo de isolamento de aplicações seria semelhante ao *sand-box* utilizado em outras tecnologias de código móvel. Esta implementação pode ser bastante flexível. Por exemplo, o *sand-box* poderia até dispensar o uso de autenticação para aplicativos considerados inofensivos, atribuindo a estes uma autonomia mínima, de acordo com a política de segurança defendida pela norma ABNT NBR 15605-2.

E quanto ao sistema embarcado de hardware restrito, o receptor de TVDi apresenta uma série de desafios para segurança da informações. A Figura 7 ilustra, em um diagrama de blocos, as camadas de software do receptor de TVDI e suas necessidades de segurança de aplicações e de dados manipulados por elas, nos ambientes de multi-aplicações e de multi-serviços: (1) Isolamento de aplicações, proporcionado pelo *middleware*; (2)

comunicação segura de informações em trânsito; (3) armazenamento seguro de dados e programas; (4) biblioteca criptográfica plenamente funcional; (5) placa aceleradora criptográfica, independente do hardware restrito dedicado às funcionalidades principais do receptor.

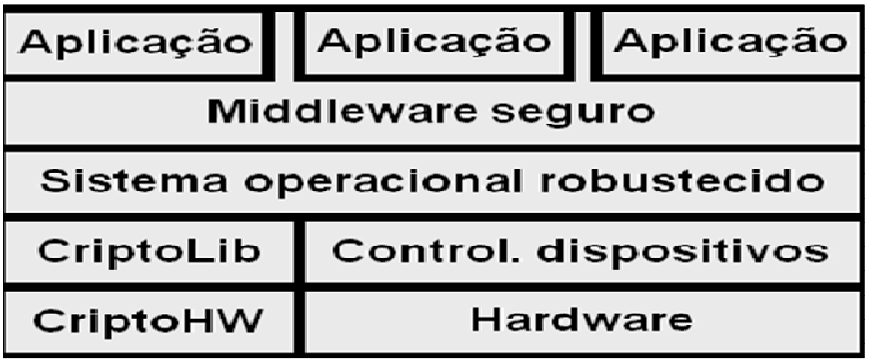


Figura 7 - Camadas de software e hardware para Set-Top Box multi-serviços seguro.

5.4. Vulnerabilidades de programação insegura

Esta seção apresenta o embrião de um trabalho contínuo de catalogação de vulnerabilidades de programação insegura identificadas nas linguagens de programação utilizadas pelo Ginga, em particular, o NCLua. Cada vulnerabilidade catalogada será documentada com uma descrição geral, programas exemplo de exploração e de simulação da vulnerabilidade e estratégias de mitigação.

Foram investigadas bibliotecas, APIs e rotinas perigosas. E foi elaborado um catálogo de vulnerabilidades de programação inseguras identificadas e documentadas em Lua, refletindo um trabalho em progresso.

Foram estudadas diversas fontes de informação sobre Lua. As seguintes podem ser citadas como mais relevantes: Lerusalimschy (2003), Lerusalimschy, Figueiredo e Celes (2006), Barbosa e Soares (2008), Lerusalimschy (2009), NCLua Tutorial, Comunidade Lua, a norma ABNT NBR 15605, partes 1 e 2, e os pacotes de software LuaCrypto e LuaSec.

Os programas exemplo e vulnerabilidades encontradas foram testados na distribuição Lua for Windows, no simulador Ginga-NCL Virtual STB e em receptores TVDi de mercado com Ginga-NCL e Lua embarcados. Cada vulnerabilidade identificada e documentada é acompanhada de uma estratégia de mitigação. Recomendações mais amplas sobre segurança de sistemas são apresentadas adiante no texto.

5.4.1. Rotinas, bibliotecas e APIs perigosas

A norma ABNT NBR 15606-2 (2007), na seção 10 - Objetos procedurais Lua em apresentações NCL, Anexo B, estabelece o uso de pacotes Ginga-J em NCLua e define as rotinas perigosas a seguir como opcionais. O *package loadlib*. Todas as funções do pacote IO. No pacote OS, as rotinas *clock, execute, exit, getenv, remove,*

rename, tmpname e *setlocale*. Todas as funções do pacote debug. Elas podem não estar disponíveis em NCLua, pois são dependentes do sistema operacional.

As investigações realizadas durante este trabalho identificaram diversas APIs opcionais disponíveis de forma perigosa. Esta lista apresentada na Tabela 1 não é exaustiva. De fato, trata-se de um trabalho em progresso.

Tabela 1: funções com potencial para exposição de vulnerabilidades.

Categoria	Rotina	Ameaça
Pacote OS	os.remove() e os.rename()	Estas rotinas proporcionam acesso e modificação do sistema de arquivos.
	os.exit() os.getenv()	Controle e acesso a informações de programas. Por exemplo, terminação da execução de um programa ou obtenção de ambiente.
	os.date()	Defeito documentado. Na versão 5.1, falha e termina a execução do programa, em algumas plataformas.
	os.execute()	Esta rotina pode ser usada na injeção de comandos arbitrários, repassados ao sistema operacional.
Carga /execução de código	loadfile() dofile()	Estas rotinas podem ser usadas na injeção de código malicioso via arquivos.
	loadstring	Esta rotina pode ser usada na injeção de códigos maliciosos via strings.
Pacote IO	io.write() io.flush() io.read()	Por se tratar de API para acesso a arquivos, é comumente associada a ataques de negação de serviço. Pode ser usada de modo negligente e consumir todo o disco/armazenamento. Também é associada a condições de competição (race conditions). Ainda, a rotina de leitura de caracteres pelo teclado pode ser usada sem verificação dos dados de entrada.

No receptor de TVD, o pacote IO não seria usado para entrada de dados pelo console (teclado + display), pois naquele ambiente vale a integração Lua e NCL (NCLua). Mas ainda assim seria usado para manipulação de arquivos.

5.4.2. Vulnerabilidades de programação Lua

As vulnerabilidades identificadas nesta seção são descritas de acordo com o seguinte formato: uma descrição geral, como explorar a vulnerabilidade, programas exemplo em Lua e estratégias de mitigação da vulnerabilidade. Além disso, é feita uma classificação da vulnerabilidade Lua, com base na vulnerabilidade análoga em dois catálogos conhecidos: SANS/CWE Top 25 (2010) e OWASP Top 10 (2010), quando aplicável.

5.4.2.1. Injeção de comandos

Esta vulnerabilidade pode se manifestar quando o software constrói totalmente ou parcialmente um comando do sistema operacional usando dados de entrada repassados diretamente a outro componente, o qual executa o comando. A rotina de entrada de dados não saneia ou saneia incorretamente caracteres especiais, capazes de modificar o significado dos dados quando enviados a outro componente.

Nesta implementação da vulnerabilidade, dois pontos devem ser observados. Primeiro, a aplicação aceita uma entrada de dados, cujo uso esperado é desviado para injetar o nome de um programa a executar. A aplicação simplesmente redireciona o comando inteiro para o sistema operacional. Segundo é suposto que o programador não teve a intenção de expor a execução de comandos a qualquer pessoa não confiável, mas o programador provavelmente não identificou as formas alternativas que os atacantes mal intencionados manipulam a aplicação.

A vulnerabilidade de injeção de comandos em Lua tem semelhanças com as seguintes vulnerabilidades classificadas:

- Vulnerabilidade A1, *Injection*, de OWASP Top 10 (2010), e
- Erro de programação número 9 (CWE-78), *Failure to Preserve OS Command Structure – OS Command Injection*, de SANS/CWE Top 25 (2010).

5.4.2.1.1. Exploração da vulnerabilidade

Se um atacante pode executar códigos arbitrários no receptor de TVDi, então isto pode significar que o equipamento está seriamente comprometido. O atacante pode ser capaz de assumir o controle sobre o programa em execução ou potencialmente romper os limites do processo em execução para abrir um novo shell no computador.

A vulnerabilidade é exposta quando um programa usa a rotina loadstring() para executar um trecho de código Lua construído a partir de material digitado pelo usuário. O código exemplo, mostrado no Programa 1, é o de uma calculadora simples que lê uma expressão aritmética e a resolve, apresentando o resultado. O código usa a facilidade Lua de resolver expressões aritméticas e executar seqüências de caracteres como comandos da linguagem.

```
01   i = 0;
02   f = "i= "..io.read();
03   g = assert(loadstring(f));
04   g();
05   print(i);
```

Programa 1: Calculadora de expressões aritméticas simples.

A calculadora pode ser abusada do seguinte modo. Quando o programa solicita a entrada de dados, digita-se a seqüência maliciosa de caracteres "`1; << código injetado>>;`". Neste trecho de código, o "1;" fecha a string da expressão aritmética.

O sinal de ponto e vírgula ao final do código injetado encerra o comando e evita concatenações defeituosas. A seguir, três explorações da calculadora.

Exploração 1. Injetando um código simples. Basta digitar a seqüência de caracteres a seguir quando o programa estiver esperando a expressão aritmética.

```
"1 ; print('oi');"
```

O resultado é a impressão de "oi" antes da impressão do resultado da expressão aritmética, "1".

Exploração 2. Injetando comando de sistema operacional. Basta digitar a seqüência de caracteres a seguir quando o programa estiver esperando a expressão aritmética.

```
"1 ; os.execute('dir');"
```

O resultado, em Windows, é a impressão da listagem dos arquivos, contidos na pasta corrente, antes da impressão do resultado da expressão aritmética, "1".

Exploração 3. Injeção de código Lua. Basta digitar a seqüência de caracteres a seguir quando o programa estiver esperando a expressão aritmética

```
"1; dofile('do.lua');
```

O resultado é a execução do arquivo do.lua, mostrado no Programa 2, onde é impressa a string "Código injetado", antes da impressão do resultado da expressão aritmética.

```
01   Print("Código injetado")
```

Programa 2: Código injetado a partir de arquivo.

5.4.2.1.2. Estratégia de mitigação

A estratégia de mitigação mais simples é evitar o uso das funções consideradas perigosas: loadstring(), dofile(), os.execute(). Se isto não for possível, o saneamento de dados de entrada, para evitar a injeção de comandos, oferece uma solução parcial. Algumas recomendações preventivas genéricas são as seguintes:

- Executar a aplicação com o mínimo de privilégios necessários;
- Evitar passar a entrada do usuário diretamente para os comandos que avaliam código (em Lua, loadstring, dofile, os.execute). Em vez disso, usar a entrada do usuário para selecionar uma ação a partir de conjunto de pré-definido de opções de comandos.
- Aplicar os controles que limitem a ação de código em linguagens consideradas inseguras. No caso deste exemplo, conter ou evitar os scripts do sistema operacional. O sand-box de código também se aplica a esta situação.

5.4.2.2. Condição de competição

Uma *Race Condition* (traduzida aqui como Condição de Competição) é a situação na quais programas em execução concorrem pelo uso simultâneo de um recurso de computação. A Condição de Competição pode ocorrer quando uma seqüência de operações, dependente do tempo, é realizada simultaneamente em várias *threads* (linha de execução) ou ambiente de multiprocessamento. Neste contexto, as instruções de uma

thread podem ser suspensas para dar vez à outra thread. Enquanto uma thread está suspensa, o ambiente do qual a thread depende pode ser alterado de forma inesperada. A causa desta vulnerabilidade é a suposição implícita e incorreta de que a seqüência de operações em questão é atômica. Uma solução possível é a de isolar o código mais relevante em um ponto crítico que pode ser tratado como logicamente atômico. As condições de competição que têm implicações de segurança são aquelas relacionadas com acessos ao sistema de arquivo. Tipicamente, a verificação do direito de acesso antes do acesso propriamente. Do ingles *Time-of-Check Time-Of-Use*, ou ToCToU.

A vulnerabilidade de condição de competição em Lua tem semelhanças com as seguintes vulnerabilidades classificadas:

- Não se aplica ao OWASP Top 10 (2010), e
- Erro de programação número 25 (CWE-362), *Race Condition*, de SANS/CWE Top 25 (2010).

5.4.2.2.1. Exploração da Vulnerabilidade

Corrotinas de Lua, similares a *threads* em outras linguagens de programação, podem ser construídas de modo a competirem por recursos, causando uma situação de *deadlock*. Um exemplo simples do conceito não precisa usar *threads*, ou corrotinas.

Esta seção mostra um exemplo de como fazer uma condição de competição em Lua. O exemplo explora ToCToU sobre arquivos. Desdobramento interessante do conceito bastante simples de condição de competição em outras linguagens de programação (C/C++/Java). O cenário é o seguinte: enquanto o programa executa, o testador muda a permissão do arquivo acessado pelo programa.

Nesta exploração, duas corrotinas diferentes de um mesmo programa estão competindo pelo arquivo. Cada corrotina abre e fecha o arquivo. Mas neste caso, quando a corrotina toctou2 é executada no meio da corrotina toctou1, a corrotina toctou1 perde a referência ao arquivo.

```
01   function toctou1 ()
02   -- Time of Check (ToC)
03      f,m = assert(io.open(".\\teste.txt","w"));
04      print(f,m);
05      print(io.type(f))
06      coroutine.yield()
07
08   -- Time of Use (ToU)
09      m = assert(f:write("\n Corrotina 1 "..os.time()))
10      print(m);
11      f:close();
12   end;
13
14   function toctou2 ()
15   -- Time of Check (ToC)
16      f,m = assert(io.open(".\\teste.txt","w"));
17      print(f,m);
18      print(io.type(f))
19
20   -- Time of Use (ToU)
21      m = assert(f:write("\n Corrotina 2 "..os.time()))
22      print(m);
23      f:close();
24   end;
25
26   co1 = coroutine.create(toctou1)
27   co2 = coroutine.create(toctou2)
28   coroutine.resume(co1)
29   coroutine.resume(co2)
30   coroutine.resume(co1)
```

Programa 3: toctou1 abre o arquivo e sede a vez para toctou2 abrir e gravar dados no mesmo arquivo, antes de toctou1.

5.4.2.2.2. Estratégia de mitigação

A mitigação dos dois primeiros casos de exploração da condição de competição está fora do escopo deste texto, pois depende de fatores externos aos programas Lua envolvidos.

O terceiro caso de exploração pode ser mitigado se a referência ao arquivo for obtida e mantida por código fora das corrotinas. No Programa 4, as corrotinas não estão competindo pelo arquivo. O acesso é mantido pelo programa principal. Primeiro a função toctou2 escreve e depois a função toctou1 escreve.

```
01   f,m = assert(io.open(".\\teste.txt","w"));
02   print(f,m);
03
04   function toctou1 ()
05   -- Time of Check (ToC)
06      print(io.type(f))
07      coroutine.yield()
08   -- Time of Use (ToU)
09      m = assert(f:write("\n Corrotina 1 "..os.time()))
10      print(m);
```

```
11   end;
12
13   function toctou2 ()
14   -- Time of Check (ToC)
15      print(io.type(f))
16   -- Time of Use (ToU)
17      m = assert(f:write("\n Corrotina 2 "..os.time()))
18      print(m);
19   end;
20
21   co1 = coroutine.create(toctou1)
22   co2 = coroutine.create(toctou2)
23   coroutine.resume(co1)
24   coroutine.resume(co2)
25   coroutine.resume(co1)
27   f:close();
```

Programa 4: Competição eliminada pela gerência externa do recurso compartilhado.

5.4.2.3. Integridade de arquivos e código malicioso

Em Lua, tal e qual descrito no capítulo 12 de Lerusalimschy (2003), arquivos de dados podem ser lidos como programas, se formatados para tal, de modo que as estruturas de dados são carregadas diretamente e com facilidade. Este é um mecanismo extremamente simples de persistência e de recuperação de dados persistidos.

Em contrapartida, a facilidade de implementação sacrifica a segurança. Dois aspectos devem ser observados como causas raiz das vulnerabilidades. Primeira, o ambiente de execução Lua não verifica a integridade e nem a autenticidade de arquivos carregados em tempo de execução (com as funções dofile() e loadfile()). Segundo, visto que não haverá diferença entre dados e programas, códigos maliciosos podem ser embutidos nos arquivos de dados. Estes códigos serão executados quando a estrutura de dados for interpretada.

A vulnerabilidade de Integridade de arquivos e código malicioso em Lua tem semelhanças com as seguintes vulnerabilidades classificadas:

- Vulnerabilidade A1, Injection, de OWASP Top 10 (2010), e
- Erro de programação número 20 (CWE-494), Download of Code Without Integrity Check, de SANS/CWE Top 25 (2010).

5.4.2.3.1. Exploração da vulnerabilidade

Usando a estratégia de persistência de dados ilustrada no Programa 5 e Programa 6, para explorar a vulnerabilidade, basta acrescentar o código que se deseja executar ao arquivo de dados. Isto preserva o comportamento aparente esperado do programa e acrescenta o comportamento malicioso.

No Programa 5, o comando os.execute("Dir ..") foi acrescentado ao final do arquivo de dados. O código malicioso é carregado pelo Programa 6 e executado e antes da saída esperada do programa original.

Outra possibilidade é a substituição completa do arquivo de dados por outro arquivo Lua contendo apenas código malicioso. Neste caso, o comportamento aparente do programa original não seria preservado.

```
01   Registro { nome = "Alexandre", }
02   Registro { nome = "Alex",}
03   -- este eh comando injetado que corrompe o arquivo de dados
04   os.execute("dir ..");
```

Programa 5: Arquivo de dados codificado como programa Lua.

```
01   regs = {}
02   function Registro (r)
03      if r.nome then regs[r.nome] = true end
04   end
05   dofile("Registro.lua")
06   for nome in pairs (regs) do print(nome) end
```

Programa 6: Programa Lua para carregar e imprimir dados de um arquivo.

O Programa 7 mostra uma estratégia simples para geração do arquivo de dados. O arquivo de saída contém os registros mostrados no Programa 5.

```
01   function serialize (o)
02         if type(o) == "string" then
03         io.write('Registro { nome = "', o, '" } \n') end
04   end
05
06   io.output(".\\Registro.lua")
07
08   serialize("Alexandre");
09   serialize("Alex");
10
11   io.close();
```

Programa 7: Estratégia simples para persistência de dados.

5.4.2.3.2. Estratégia de mitigação

A estratégia de mitigação é a utilização da técnica de assinatura de código, baseada em criptografia de assinaturas digitais, capaz de garantir não somente a integridade da aplicação e seus arquivos constituintes, mas também a autenticação de origem, ou autoria da aplicação. Esta técnica tem sido usada com sucesso em diversas implementações de código móvel, tais como o Java e o .NET, assim como no Android.

Além disso, algumas medidas paliativas, descritas a seguir, podem ser adotadas. Programas ou dados sensíveis que não são persistidos não têm os problemas detalhados nesta seção. Persistir dados efetivamente cria um acesso público para os dados.

Sobre a persistência de tabelas em arquivos Lua. Depois que uma tabela foi persistida, qualquer controle de segurança da linguagem Lua não poderá mais ser executado, os atacantes poderão acessar campos particulares em uma tabela analisando o dado diretamente. Portanto, é uma boa prática não armazenar dados confidenciais em uma tabela gravada em arquivos.

Sobre a recuperação ou carga de tabelas armazenadas em arquivos. Recuperação cria uma nova instância de uma tabela sem invocar a função construtora convencional, caso exista. Duas medidas de segurança simples evitarão vulnerabilidades. Primeira, realizar, na recuperação, as verificações de validação de entrada do mesmo modo que realizado na entrada dos dados ou em função construtora. Segundo, atribuir valores padrão consistentes com aqueles atribuídos em uma função construtora, para os dados não preservados quando da persistência da tabela.

5.4.2.4. Script cruzado armazenado

Esta vulnerabilidade é semelhante ao tradicional *Cross Site Scripting* (XSS), uma das vulnerabilidades mais perigosas das aplicações web. A vulnerabilidade se manifesta quando o software não valida, filtra e nem decodifica a entrada de dados do usuário, e nem a codifica antes de colocá-la na saída de dados apresentados para outros usuários.

Esta manifestação da vulnerabilidade segue a linha de atuação do XSS armazenado. O aplicativo armazena dados em arquivo (mas poderia ter sido usada outro armazenamento). O atacante mal intencionado manipula a aplicação para que um script executável seja armazenado junto com os dados. Mais tarde, o dado perigoso (código malicioso) é posteriormente lido de volta para a aplicação e incluídos no conteúdo dinâmico, vitimando outro usuário.

A vulnerabilidade de script cruzado armazenado em Lua tem semelhanças com as seguintes vulnerabilidades classificadas:

- Vulnerabilidade A2, *Cross-site scripting*, de OWASP Top 10 (2010), e
- Erro de programação número 1 (CWE-79), *Failure to Preserve Web Page Structure – Cross-site Scripting*, de SANS/CWE Top 25 (2010).

5.4.2.4.1. Exploração da Vulnerabilidade

O exemplo clássico de vulnerabilidade XSS é a injeção de um script simples que quando refletido é executado. Neste exemplo, o script cruzado armazenado é exemplificado com a exploração da codificação de strings em Lua.

Exploração 1. O Programa 8 lê dados de entrada e, com uma estratégia simples de serialização, os salva no arquivo Programa 10. Os colchetes duplos usados como delimitadores de string pelo Programa 8 permite a inclusão de caracteres especiais.

O Programa 9 recupera o dado armazenado, o qual pode conter o script injetado, e o apresenta ao usuário. Os seguintes códigos script podem ser injetados durante a leitura de dados pelo Programa 8 e serão armazenados e refletidos na resposta ao usuário.

A seqüência de caracteres a seguir fecha uma string em aberto, injeta o script malicioso e reconstitui a atribuição de variável. O Programa 10 mostra o resultado.

```
]] ; print("Olá") ; aaaaa = [[
```

A seqüência a seguir fecha uma string em aberto, injeta o script malicioso e transforma em comentário tudo o que estiver após o trecho injetado.

```
]] ; print("olá") --
```

```
01   function serialize (o)
02      if type(o) == "string" then
03         io.write("variavel = [[", o, "]]")
04      end
05   end
06   io.output(".\\serial.lua")
07   o = io.read();
08   serialize("isto é um texto de teste "..o);
09   io.close();
```

Programa 8: Usando strings delimitadas por colchetes duplos.

```
01   dofile(".\\serial.lua");
02   print("\n")
03   print(variável)
```

Programa 9: Reconstituição da variável a partir de string armazenada.

```
01   variavel = [[isto é um texto de teste ]] ; print("Olá") ;
02   aaaaa = [[]]
```

Programa 10: variável armazenada, violada por código malicioso e colchetes duplos.

Exploração 2. O Programa 11 ilustra uma estratégia de serialização diferente da anterior. A string persistida será delimitada por aspas simples. Em termos práticos, o resultado é idêntico ao obtido pelo uso de aspas duplas. O método de recuperação é o mesmo da exploração anterior, o Programa 9. O Programa 12 mostra o arquivo gerado com a violação, o código injetado. A seqüência de caracteres a seguir fecha uma string em aberto, injeta o script malicioso e transforma em comentário tudo o que estiver após o trecho injetado.

```
' ; print("olá") --
```

```
01   function serialize (o)
02      if type(o) == "string" then
03         io.write("variavel = '", o, "'")
04      end
05   end
```

Programa 11: Usando strings delimitadas por aspas simples.

```
01   Variável = 'isto é um texto de teste ' ; print("ola") ; --'
```

Programa 12: variável armazenada, violada por código malicioso em aspas simples.

5.4.2.4.2. Estratégia de mitigação

O ataque XSS tradicional sobre HTML possui duas estratégias de mitigação, uma voltada para os dados de entrada e outra voltada para a saída. A validação, ou o saneamento, de dados de entrada evita a entrada de código malicioso. A formatação (ou

codificação) dos dados de saída evita a saída de informação mal formatada ou corruptível. Em Lua, o uso da função string.format com o parâmetro "%q" dificulta a construção de strings defeituosas, pois a string estará em aspas duplas e todo caractere especial estará formatado com "\". Conforme ilustrado no Programa 13. O Programa 14 ilustra o arquivo de dados armazenados quando uma aspas extra é inserida pelo usuário. A string não é corrompida, graças à estratégia de codificação de saída.

```
01   function serialize (o)
02      if type(o) == "string" then
03         io.write(string.format("variavel = %q", o))
04      end
05   end
```

Programa 13: Codificação de strings de saída evita a execução de código injetado.

```
01   variavel = "isto é um texto de teste \""
```

Programa 14: Aspas extra precedida por barra sem corromper a string.

5.4.2.5. Referência insegura a tabelas

Diversas linguagens de programação procedimentais que possuem vetores e acesso a dados por referência sofrem de uma dificuldade bastante peculiar: o acesso múltiplo, via diversas referências, a um mesmo dado. Nesta condição, o dado referenciado pode ser modificado por qualquer um dos detentores de referências, sem que os outros detentores tomem conhecimento, em tempo hábil, da modificação.

Em Lua, esta vulnerabilidade se manifesta no acesso a tabelas mutáveis. Isto é, tabelas cujas referências são compartilhadas. Tabelas mutáveis podem ser alteradas depois e até durante a execução de uma função, causando comportamento incorreto. Se uma função não é especificada para operar direta e cautelosamente sobre tabelas mutáveis compartilhadas, é recomendável criar uma cópia do dado e executar a lógica do programa apenas na cópia.

De fato, se a tabela é compartilhada por vários, os programas que usam a tabela podem sofrer da vulnerabilidade Condição de Competição, causada por diferenças de conteúdo e inconsistências de *Time-of-Check Time-of-Use* (ToCToU). Por exemplo, ao ser acessada e alterada por corrotinas distintas, a tabela compartilhada conteria um valor mutável durante o momento de verificação do valor, mas um valor diferente durante o uso da tabela.

A vulnerabilidade de acesso por referência a elementos de tabelas em Lua tem semelhanças com as seguintes vulnerabilidades classificadas:

- Vulnerabilidade A4, *Insecure Direct Object References*, de OWASP Top 10 (2010), e
- Erro de programação número 25 (CWE-362), *Race Condition*, de SANS/CWE Top 25 (2010).

5.4.2.5.1. Exploração da vulnerabilidade

Esta seção descreve uma exploração da vulnerabilidade de referência insegura a tabelas com acesso múltiplo a dados mutáveis. Todos os programas desta seção fazem uso do Programa 15, para a impressão de tabelas.

```
01    Function printtable (o)
02         if type(o) == "table" then
03            io.write("{")
04            for k,v in pairs(o) do
05              io.write("  "..k .." = ")
06              printtable(v)
07              io.write(",")
08            end
09            print("}\n")
10         end
11         if type(o) == "string" then
12         io.write(string.format("%q", o))
13         end
14         if o == nil then io.write("nil \n \n") end
15   end
```

Programa 15: rotina de apoio para impressão de tabelas.

A exploração da vulnerabilidade é construída sobre o conceito de *proxies* de tabelas, com *metatables* e *metamethods* de Lua. O Programa 16 contém o código de um proxy de acesso, um monitor de referências, que serve de intermediário ou mediador em todo acesso ao conteúdo de uma dada tabela original.

O monitor de referências, MR, possui duas funções principais. Primeira, a função MR.monitor(t), responsável pela construção da *metatable*, o proxy propriamente dito. Segunda, a função MR.interno() que retorna a tabela original.

O aspecto de implementação vulnerável deste monitor é o fato dele manter uma referência interna, não uma cópia ou instância única, da tabela mediada, a variável MR._t, e externá-la, espalhando descontroladamente as referências.

```
01    MR = { _t = nil}
02
03   function MR.monitor(t)
04               MR._t = t
05         local proxy = {}
06         local    mt = {
07         __index    = function (t,k) return MR._t[k] end,
08         __newindex = function (t,k,v) MR._t[k]="MR: ".. v end
09         }
10         setmetatable(proxy, mt)
11         return proxy
12   end
13
14   function MR.interno() return MR._t end
```

Programa 16: Monitor de referências a tabelas.

O Programa 17 mostra a utilização do monitor de referências. A tabela t é criada e preenchida. O proxy é criado. A tabela original é modificada. Em seguida a referência à tabela original é anulada. Uma alteração é feita via proxy. O estado interno do proxy é recuperado na variável t original. A saída do programa é mostrada no quadro Output 1.

```
01   T = {}
02   for i=0, 4 do t[i] = ""..i end
03
04   dofile('printtable.lua')
05   dofile('monitorreferencia.lua')
06
07   proxy = MR.monitor(t) -- cria proxy
08
09   printtable(t) -- print tabela original
10
11   t[1] = "A"    -- modifica tabela original
12   printtable(t) -- print tabela modificada
13
14   t = nil       -- perde referência original
15   printtable(t) -- print tabela original
16
17   proxy[2] = "A" -- modifica via proxy
18
19   t = MR.interno()
20   printable(t)
```

Programa 17: Manipulação de referências.

```
01   {  1 = "1",  2 = "2",  3 = "3",  4 = "4",  0 = "0",}
02   {  1 = "A",  2 = "2",  3 = "3",  4 = "4",  0 = "0",}
03   nil
04   {  1 = "A",  2 = "MR: A",  3 = "3",  4 = "4",  0 = "0",}
```

Output 1: Saída do programa de manipulação de referências.

5.4.2.5.2. Estratégia de mitigação

A estratégia geral de tratamento desta vulnerabilidade é a eliminação das referências extras e indesejadas às tabelas, pelo uso de cópias, valor a valor, da tabela original.

Uma variação da vulnerabilidade é a exposição do estado interno. Se uma função retorna uma referência a uma tabela interna mutável, em seguida, o código do cliente da função pode modificar o estado interno da tabela. Portanto, é uma boa prática de programação segura fazer cópias de tabelas mutáveis e retornar as cópias, a menos que a intenção seja de fato compartilhar o estado interno. O Programa 18 contém uma função de cópia, valor a valor, da tabela para o monitor de referência, MR.clone(). Uma função que retorne o estado interno, sem espalhar referência, usa a rotina de clonagem.

```
01   function MR.clone()
02        local c = {}
03        for k,v in pairs(MR._t) do c[k] = v end
04        return c
```

```
05   end
06
07   function MR.internoSeguro() return MR.clone() end
```

Programa 18: Clonagem da referência interna.

Outra variação da vulnerabilidade é a guarda de referências a tabelas mantidas externamente. Funções que recebem tabelas devem clonar as tabelas e armazenar a cópia para trabalho. Em vez de trabalhar sobre a referência original. Isso impede que as mudanças inesperadas na tabela original, realizadas sobre alguma referência perdida, afetem o funcionamento do programa. Expor tabelas internas diretamente permite alteração de algum código ou dado, sobre a referência original, a qual deveria ser mantida sob o controle interno. É mais seguro retornar uma cópia da tabela interna.

O monitor de cópias, mostrado no Programa 19, é uma solução mais abrangente para a questão das referências extras indesejadas. O monitor de cópias não é mais um proxy para a tabela original, mas sim um intermediário no acesso a uma cópia da tabela original. O Programa 20 usa o monitor de cópias e exercita seu funcionamento.

```
01    MC = { _t = {} }
02
03   function MC.copia(t)
04        for k,v in pairs(t) do MC._t[k] = v end
05   end
06
07   function MC.monitor(t)
08        MC.copia(t)
09
10        local proxy = {}
11        local mt = {
12        __index = function (t,k) return MC._t[k] end,
13
14        __newindex = function (t,k,v) MC._t[k] ="MC: "..v end
15        }
16        setmetatable(proxy, mt)
17        return proxy
18   end
19
20   function MC.interno() return MC._t end
21
```

Programa 19: Monitor de cópias

```
01   t = {}
02   for i=0, 4 do t[i] = ""..i end
03
04   dofile('printtable.lua')
05   dofile('monitorcopia.lua')
06
07   proxy = MC.monitor(t) -- cria proxy
08
09   printtable(t) -- print tabela original
10
```

```
11   t[1] = "A" -- modifica tabela original
12   printtable(t) -- print tabela modificada
13
14   t = nil -- perde referência original
15   printtable(t) -- print tabela original
16
17   proxy[2] = "A" -- modifica via proxy
18
19   t = MC.interno()
20
21   printtable(t)
```

Programa 20: Manipulação de cópias

O funcionamento do Programa 20 é o seguinte. A tabela t é criada e preenchida. O proxy é criado. A tabela original é modificada. Em seguida a referência à tabela original é anulada. Uma alteração é feita via proxy. O estado interno do proxy é recuperado na variável t original. A saída do programa é mostrada no quadro Output 2, onde se vê que os conteúdos de cada tabela são distintos das demais.

```
01   {  1 = "1",  2 = "2",  3 = "3",  4 = "4",  0 = "0",}
02   {  1 = "A",  2 = "2",  3 = "3",  4 = "4",  0 = "0",}
03   nil
04   {  1 = "1",  2 = "MC: A",  3 = "3",  4 = "4",  0 = "0",}
```

Output 2: Saída do programa de manipulação de cópias.

O Programa 21 acrescenta ao monitor de cópias a facilidade de expor cópias de seu estado interno, de modo análogo ao monitor de referências.

```
01   function MC.clone()
02        local c = {}
03        for k,v in pairs(MC._t) do c[k] = v end
04        return c
05   end
06
07   function MC.internoSeguro() return MC.clone() end
```

Programa 21: Exposição de clone do estado interno em monitor de cópia.

O Programa 22 e sua saída em Output 3 ilustram mais uma vez as diferenças entre cópias e referências. No monitor de cópias, as referências às tabelas são impressas para mostrar que a tabela original, o estado interno e a cópia externada são de fato tabelas diferentes. No monitor, as referências são impressas para mostrar que a tabela original e o estado interno do proxy são de fato a mesma tabela e a cópia externada é uma tabela diferente.

```
01   t = {}
02   for i=0, 4 do t[i] = ""..i end
03
04   print("\n clonagem da cópia \n")
05   dofile('printtable.lua')
06   dofile('monitorcopia.lua')
07
```

```
08   print("Origem",t)
09   MC.monitor(t)
10   print("Interno",MC.interno())
11   print("IntSeg",MC.internoSeguro())
12
13   print("\n clonagem da referencia \n")
14   dofile('monitorreferencia.lua')
15
16   print("Origem",t)
17   MR.monitor(t)
18   print("Interno",MR.interno())
19   print("IntSeg",MR.internoSeguro())
```

Programa 22: Programa de análise de clones.

```
01   Clonagem da cópia
02
03   Origemtable: 0x6b7ac0
04   Internotable: 0x6a3280
05   IntSegtable: 0x6a71f8
06
07   clonagem da referencia
08
09   Origemtable: 0x6b7ac0
10   Internotable: 0x6b7ac0
11   IntSegtable: 0x6a79f8
```

Output 3: Saída do programa de análise de clones.

5.4.2.6. Injeção de SQL

As aplicações interativas instaladas no receptor, quando bem construídas, provavelmente seguirão o modelo de arquitetura em três camadas, no qual apenas a camada de apresentação (escrita em Lua) estaria no receptor e acessaria serviços web, via rede IP, para consulta a bases de dados remotas. Porém, sempre há a possibilidade de que um SGBD (Sistema de Gerenciamento de Base de Dados) simples seja portado para um receptor e seja localmente acessado por programas Lua, via a biblioteca LuaSQL.

LuaSQL (www.keplerproject.org/luasql), uma interface ODBC para acesso de programas Lua à vários SGBDs, seria uma escolha possível para viabilizar o acesso de programas Lua a bases de dados.

Porém, LuaSQL não possui sentenças pré-definidas ou pré-compiladas, uma característica comum a pacotes semelhantes disponíveis em outras plataformas de software, tais como Java e dotNET.

A vulnerabilidade de Injeção de SQL em Lua tem semelhanças com as seguintes vulnerabilidades classificadas:

- Vulnerabilidade A1, *Injection*, de OWASP Top 10 (2010), e
- Erro de programação número 2 (CWE-89), *Failure to Preserve SQL Query Structure – SQL Injection*, de SANS/CWE Top 25 (2010).

5.4.2.6.1. Exploração da vulnerabilidade

A ausência de sentenças SQL pré-definidas tornaria os programas Lua, com LuaSQL, mais vulneráveis aos ataques de injeção de SQL. Isto se deve ao fato da API LuaSQL só aceitar sentenças SQL como strings. O Programa 23 ilustra o uso da API LuaSQL para execução de sentenças SQL. A primeira sentença executada é uma string simples. A segunda sentença executada é formatada (string.format) e parametrizada.

```
01   cur = assert (con:execute"SELECT name, email from people")
02   -- ou
03   res = assert (con:execute(string.format([[
04       INSERT INTO people
05       VALUES ('%s', '%s')]], name, email)
06     ))
```

Programa 23: Trecho de programa ilustrando o uso do pacote LuaSQL.

A construção e a formatação de strings em Lua podem ser manipuladas para a injeção de comandos SQL. Conforme exemplificado a seguir pelo Programa 24, que constrói três sentenças SQL para *login* com busca em uma tabela de usuários. Este exemplo é uma variação do caso mais simples de injeção SQL encontrado na literatura. A primeira sentença (SQL1) é construída por concatenação simples de strings. A segunda sentença (SQL2) usa *string.format* e parametriza *login* e senha como strings simples (%s). A terceira sentença (SQL3) também usa *string.format*, mas parametriza *login* e senha como strings cotadas e com tratamento dos caracteres de escape (%q). As sentenças com SQL injetado são mostradas no Output 4.

```
01   login  = '" or 1=1 --'
02   passwd = '1234'
03
04   sql1 = 'SELECT * FROM USERS WHERE LOGIN = \"'..
05   login ..'\" AND PASSWD = \"' .. passwd ..'\"'
06
07   sql2 = string.format(
08   [[SELECT * FROM USERS WHERE LOGIN = "%s" AND PASSWD =
09   "%s"]],
10   login, passwd)
11
12   login  = '\" or 1=1 --'
13
14   sql3 = string.format(
15   [[SELECT * FROM USERS WHERE LOGIN = %q AND PASSWD = %q]],
16   login, passwd)
17
18   print("SQL1 = " .. sql1)
19   print("SQL2 = " .. sql2)
     print("SQL3 = " .. sql3)
```

Programa 24: A construção de três sentenças SQL de forma maliciosa.

No Output 4, a sentença SQL1, gerada por concatenação simples de strings, foi corrompida com a injeção SQL mais simples, o trecho `" or 1=1 --`. A sentença SQL2 foi corrompida do mesmo modo, apesar de utilizar strings formatadas. A

sentença SQL3 pode não ser considerada pelo SGBD alvo uma sentença SQL válida, devido ao "\". Se for válida, então foi corrompida, apesar de utilizar strings formatadas e cotadas.

```
01   SQL1 = SELECT * FROM USERS WHERE LOGIN = "" or 1=1 --" AND
02   PASSWD = "1234"
03
04   SQL2 = SELECT * FROM USERS WHERE LOGIN = "" or 1=1 --" AND
05   PASSWD = "1234"
06
07   SQL3 = SELECT * FROM USERS WHERE LOGIN = "\" or 1=1 --" AND
08   PASSWD = "1234"
```

Output 4: Sentenças SQL mal construídas.

5.4.2.6.2. Estratégia de mitigação

As consultas pré-definidas ou pré-compiladas são uma das linhas de defesas mais comuns contra a injeção de SQL. Uma defesa alternativa e parcial seria o uso de procedimentos armazenados (*stored procedures*), se disponível, no próprio SGBD. Porém é necessário tomar cuidado com procedimentos armazenados, que são parametrizados, mas que ainda podem ser vulneráveis à injeção.

O uso de string.format e %q evita certas construções maliciosas, mas não deve ser usado como única linha de defesa. Ainda é necessário sanear os dados de entrada.

A melhor opção de proteção defensiva é usar uma API segura para acesso ao SGBD que evite o uso direto do interpretador SQL e forneça uma interface parametrizada. A API LuaSQL avaliada neste texto não é segura contra injeção SQL. Logo, existe uma oportunidade de melhoria deste pacote.

5.4.2.7. Mau uso de criptografia

De acordo com Anderson (1993), a maioria das falhas de segurança, devidas ao mau uso de criptografia, são de fato erros de configuração, de implementação e de funções administrativas; como por exemplo, a gestão inadequada de chaves criptográficas, a utilização de configurações vulneráveis de criptografia forte ou o uso de criptografia reconhecidamente fraca.

Assim como ocorre com as outras vulnerabilidades de programação insegura, as vulnerabilidades associadas ao mau uso de criptografia não são detectadas por ferramentas de varredura externas de vulnerabilidades. Ferramentas de análise estática de código fonte podem detectar o uso de APIs criptográficas conhecidas (pelos fabricantes de ferramentas), mas não podem detectar se a API criptográfica está sendo usada adequadamente e de acordo com regras de negócio.

No caso de Lua, a análise manual de programas ainda é a melhor maneira de verificar se uma aplicação cifra adequadamente os dados sensíveis e tem um bom mecanismo e gerenciamento de chaves. Pois as ferramentas de análise estática de programas ainda não tratam Lua. Foram analisadas manualmente as APIs e a documentação (incluindo programas exemplo) de 3 bibliotecas criptográficas em Lua publicamente disponíveis:

- LuaCripto é uma fachada Lua para as funções criptográficas do OpenSSL. Isto pode significar que o LuaCripto não seria mais seguro que o OpenSSL subjacente. Porém ele pode ser mais inseguro, ao fazer uso de partes inseguras e eliminar partes seguras.

LuaSec é um encapsulamento das rotinas OpenSSL para comunicação segura com SSL/TLS. Assim como no caso anterior, isto pode significar que o LuaSec não seria mais seguro que o OpenSSL subjacente. Porém ele pode ser mais inseguro, ao fazer uso de partes inseguras e eliminar partes seguras.

- LuaMD5 é uma biblioteca criptográfica simples, com implementações próprias de algoritmos criptográficos conhecidos.

Este é um trabalho em progresso. O resultado preliminar da análise das APIs das bibliotecas criptográficas é mostrado na próxima seção.

A vulnerabilidade de mau uso de criptografia em Lua tem semelhanças com as seguintes vulnerabilidades classificadas:

- Vulnerabilidade A7, *Insecure Cryptographic Storage*, de OWASP Top 10 (2010).
- Erro de programação número 24 (CWE-327), *Use of a Broken or Risky Cryptographic Algorithm*, de SANS/CWE Top 25 (2010).
- Erro de programação número 10 (CWE-311), *Missing Encryption of Sensitive Data*, de SANS/CWE Top 25 (2010).

5.4.2.7.1. Exploração da vulnerabilidade

De fato, não há nesta seção a exploração da vulnerabilidade, mas o nome da seção é mantido por consistência com as outras vulnerabilidades documentadas. O que há nesta seção é a listagem de características das APIs criptográficas que levariam a vulnerabilidade exploráveis, quando utilizadas em alguma aplicação.

A análise preliminar apresentada neste texto sugere que não há atualmente em Lua uma biblioteca criptográfica completa, plenamente funcional, disponível publicamente. O fato de tanto LuaCripto, quanto de LuaMD5, somente oferecerem acesso a algoritmos de hash ou algoritmos de cifração fracos é um grande limitante da aplicabilidade destas bibliotecas. Além disto, não foi encontrado um receptor de TVDi que oferecesse outra fonte de serviços criptográficos, diferente de LuaCripto e LuaSec.

LuaMD5 usa algoritmos quebrados ou inseguros. Esta biblioteca criptográfica não oferece algoritmos fortes, apenas algoritmos fracos, como o MD5, o DES e até XOR. LuaMD5 favorece o armazenamento inseguro de chaves criptográficas. Diferente de outras APIs criptográficas (JCA, CAPI, OpenSSL), LuaMD5 não oferece mecanismos para armazenamento seguro de chaves, tais como uma KEK (chave de cifração de chaves) ou um repositório seguro para chaves.

LuaCripto usa algoritmos quebrados ou inseguros. Esta biblioteca praticamente só oferece algoritmos considerados inseguros (md5, sha1). LuaCripto favorece o armazenamento inseguro de chaves criptográficas. Diferente de outras APIs criptográficas (JCA, CAPI, OpenSSL) , LuaCripto não oferece mecanismos para

armazenamento seguro de chaves, tais como uma KEK (chave de cifração de chaves) ou um repositório seguro para chaves.

LuaSec é uma fachada bastante flexível para estabelecimento de conexões SSL/TLS com uma instalação do OpenSSL subjacente. Porém LuaSec não oferece acesso às funções criptográficas do OpenSSL. Isto não é uma vulnerabilidade, dado que o objetivo da LuaSec é outro.

5.4.2.7.2. Estratégia de mitigação

Conforme as argumentações de Kocher et al (2004) e Ravi et al (2004), as restrições de hardware impõem limitações severas ao desempenho de algoritmos criptográficos. Os receptores de TVDi disponíveis atualmente são extremamente limitados em memória de trabalho, armazenamento e processamento. Este pode ser um dos motivos da grande dificuldade no oferecimento de serviços de segurança plenos.

A implementação do algoritmo DES, oferecida em LuaMD5, não deve mais ser usada profissionalmente. Implementações do AES ou do 3DES seriam preferíveis se estivessem disponíveis em LuaMD5.

Comparativamente a outras plataformas de software e linguagens de programação (tais como Java e .NET), existe uma lacuna a ser preenchida em Lua. LuaCripto e LuaMD5 poderiam ser estendidas para oferecer mais funções criptográficas e um repositório seguro de chaves. Esta extensão viabilizaria novas aplicações escritas em Lua, com requisitos fortes de sigilo e de autenticação.

Os exemplos de utilização das bibliotecas criptográficas contêm chaves *hardcoded*. Isto pode passar a idéia de que a gestão de chaves é uma tarefa simples. Quando de fato é bastante complicada e uma das principais causas de vulnerabilidades no uso de criptografia. Algumas questões genéricas a se considerar na proteção e na gestão de chaves criptográficas são as seguintes:

- Geração de chaves criptográficas robustas.
- Distribuição segura de chaves criptográficas.
- Armazenamento seguro de chaves criptográficas.
- Alteração periódica das chaves criptográficas.
- Inutilização ou substituição de chaves criptográficas comprometidas, antigas ou suspeitas.
- Prevenção contra a substituição não autorizada de chaves criptográficas.

5.5. Vulnerabilidades de arquitetura, de rede e de sistema operacional

Esta seção mostra, com ferramenta construída para tal, mas em ambiente simulado, como realizar avaliações de segurança em receptores de TVDi com Ginga. A ferramenta em questão dá apoio a um método de testes que utiliza o controle remoto de TV/receptor como dispositivo de entrada de dados e realiza testes de intrusão por injeção de comandos. Além disso, serão apresentados resultados gerais de avaliação de segurança obtidos em laboratório.

5.5.1. Deficiências arquiteturais e de projeto da plataforma Lua

Segundo Hoare (1981), há duas maneiras de construir um software: A primeira é fazê-lo tão simples que obviamente não há deficiências, a outra é fazê-lo tão complicado que não existem deficiências óbvias. O projeto da linguagem de programação Lua buscou a simplicidade extrema. Porém, o atual uso desta linguagem, em sistemas com requisitos de segurança complexos, levanta diversas questões de segurança sistêmica. A fim de realizar requisitos de sistemas mais complexos, muitas vezes são utilizadas construções artificiais que, de fato, aumentam a complexidade a ser tratada pelo programador, aumentando também a suscetibilidade a vulnerabilidades de programação insegura.

As questões de segurança tratadas nesta seção estão mais relacionadas ao projeto seguro, pois as vulnerabilidades não dependem da sintaxe da linguagem ou de bibliotecas predefinidas, mas sim do modo como as construções são projetadas e utilizadas.

5.5.1.1. A técnica de sand-boxing

Na Comunidade Lua (Lua-users.org/wiki/SandBoxes) é descrita uma técnica para a execução de código não confiável em um ambiente de execução Lua restrito. Isto é, um mecanismo de contenção de código. Esta técnica não considera que pode haver níveis de confiança de código. Além disso, não são consideradas mecanismos de promoção de confiança, tais como a assinatura de programas (code signing) e nem proteções contra ataques de negação de serviço por esgotamento de CPU ou de memória.

O método de sand-boxing proposto na referência supracitada é essencialmente programático e de fato muito flexível, pois quase tudo que não está implementado pode ser construído. Porém, com o sacrifício da legibilidade do código e a conseqüente maior suscetibilidade a erros. A principal vantagem deste método é que o sand-box não oferece aos programas acesso ao sistema de arquivos e variáveis globais fora do próprio sand-box. A principal desvantagem é que esta técnica de sand-boxing, por si só, não impõe limites ao consumo de memória e nem de CPU pelos programas maliciosos.

5.5.1.2. Metatables e Metamethods

Em Lerusalimschy (2003), este conceito possui vários usos interessantes para a segurança da informação. A saber, proxies, rastreamento de acessos e tabelas somente de leitura. Este conceito poderia ainda ser ampliado para oferecer mecanismos de segurança mais amigáveis ao programador, tais como o acesso transparente (autorizado) a uma tabela cifrada. Neste exemplo, uma tabela proxy guardaria as chaves criptográficas. Uma característica curiosa é que Lua oferece funções que burlam a suposta proteção oferecida pelas metatables. As rotinas rawget, rawset, e rawequal.

5.5.1.3. Orientação a Objetos

Apesar de um espaço significativo de Lerusalimschy (2003) se dedicado para a orientação a objetos em Lua, a linguagem não oferece qualquer mecanismo explícito (isto é, uma estrutura que faça parte da sintaxe da linguagem) para mecanismos presentes em muitas das linguagens orientadas a objeto na implementação do conceito de *information riding*, a saber: pacotes, controle de acesso (visibilidade variável) a atributos e a métodos.

Estas características de Orientação a Objetos ausentes na linguagem Lua podem ser obtidas a partir de construções lógicas e disciplina de programação. Isto é perigoso, pois tudo o que depende da disciplina do programador é geralmente negligenciado quando o prazo, orçamento ou qualidade precisam ser comprometidos. Sendo uma fonte comum de vulnerabilidades.

As questões de privacidade de dados, tanto em pacotes quanto em objetos são artificiais à linguagem e ilustram bem a suscetibilidade a erros. Por exemplo, uma vez que todas as variáveis são globais por default, uma variável local deve ser explicitamente declarada. Visto que a ausência da palavra reservada "local" não representa um erro de sintaxe, ela pode ser facilmente esquecida pelo programador, sem que o lapso seja detectado de forma automática no receptor da TV digital

Segundo Piccolo 2005, o receptor de TV digital terrestre é também denominado *set-top-box*, e é conhecido também no mercado como URD (Unidade Receptora e Decodificadora). Este equipamento é o responsável pela recepção do sinal digital recebido para que este sinal possa ser visualizado no aparelho de TV analógico convencional. Basicamente ele trata os sinais digitais recebidos por radiodifusão terrestre, cabo, ou satélite e os converte para o formato analógico. Possui como funcionalidades básicas segundo Nicholls: demultiplexar o sinal digital recebido; decodificar informações de áudio e vídeo; processar os dados recebidos e, se for o caso, sincronizá-los com a programação; enviar dados via canal de retorno; construir a imagem a ser exibida no aparelho de TV e a conversão desta para o sinal analógico.

A fonte do sinal pode ser recebida pelo receptor via interfaces distintas que o mesmo apresenta: porta ethernet (serviços *triple play*), cabo coaxial, antena VHF/UHF, ou ainda via modem 3G pela porta USB que o mesmo possui. O canal de retorno permite ao usuário enviar dados relativos a um programa ou emissora ou provedor de serviço via controle remoto. Para que isto seja possível, pode ser utilizado como canal de retorno, telefone fixo (serviços XDSL), modem 3G ou ainda PLC (*Power Line Communication*)

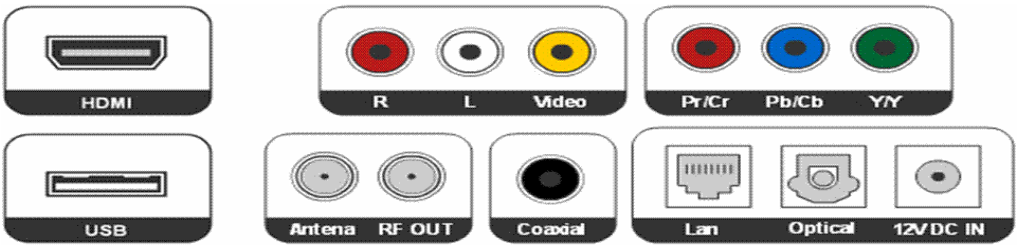


Figura 8 – Interfaces Presentes no Receptor

5. 5.2. Metodologia

A metodologia utilizada pelos autores deste minicurso está embasada nas boas práticas de diversas metodologias dentre as quais: o PMBOK – Project Managment Body of Knowledge, ISSAF – *Information System Security Assessment Framework*, e a

OSSTMM - *Open Source Security Testing Methodology Manual* (Livro Professional *Penetation Testing*), e o *OWASP Testing Guide* v3 somado a contribuições especificas e as adaptações que o cenário de coleta às informações do receptor de TV digital propiciou ao desenvolvimento deste trabalho.

O método foi estruturado em três módulos, a saber: a Injeção de comandos, a Execução dos comandos injetados, e a obtenção do resultado na Coleta dos comandos injetados. Levando em consideração as camadas no sentido do topo para baixo (top/down) do receptor digital: aplicação, *middleware*, software e hardware, como demonstrado na

Figura 9. Neste, sentido foi utilizado o ciclo PDCA (Plan-Do-Check-Act / Planejar, Fazer, Verificar e Agir) Deming, W. Edwards (1986). para analisar as vulnerabilidades encontradas. Foram utilizadas ainda as duas fases do *ISSAF*: Planejamento e Preparação – fase correspondente ao planejamento e elaboração dos testes, e a Avaliação – fase correspondente aos passos para a execução do teste de penetração em camadas. E por fim, o uso da metodologia OSSTMM, utilizada para auditoria de segurança, harmonizando os padrões e normas já existentes.

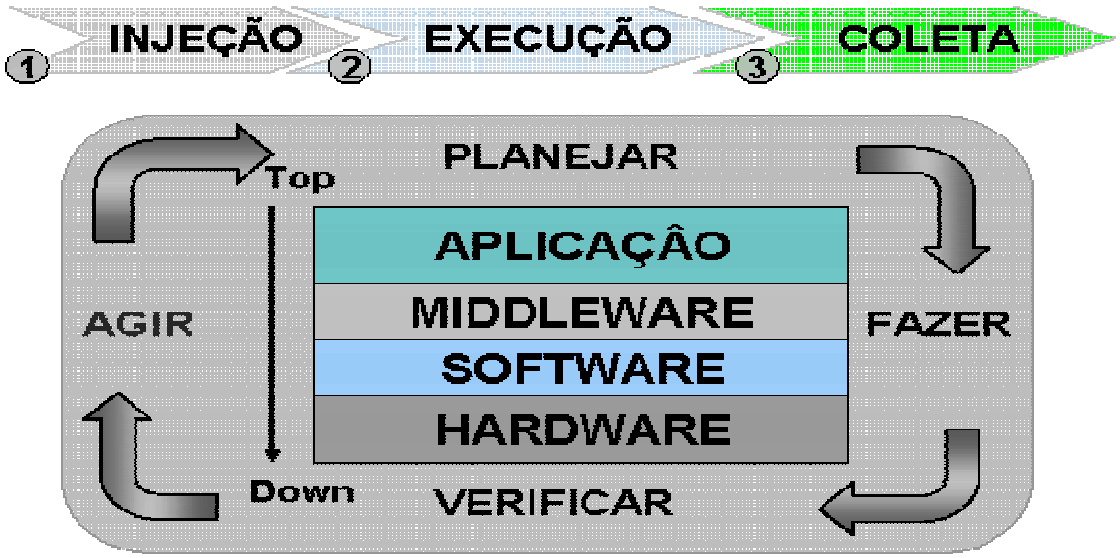


***Figura* 9 - Método de Analise de Vulnerabilidades no Receptor de TV digital interativaObtendo Informações do Receptor de TV digital.**

Segundo Wilhelm, a obtenção de informações é o primeiro passo para conduzir um teste de penetração. Esta ação pode ser segregada em dois tipos diferentes – passivo e ativo. Obter informação de modo passivo permite ter mais informações sobre a rede e os sistemas que estão conectados diretamente ao dispositivo, no caso deste trabalho, o receptor de TV digital. Mais a frente será demonstrado os resultados encontrados utilizando ferramentas de varreduras de rede e a ferramenta desenvolvida pelo CPqD , o AIC – Aplicativo de Injeção de Comandos, para injeção de comandos no receptor. O

segundo tipo para se obter informação é o ativo, ou seja, é possível a conexão a outros receptores alvos. Neste trabalho não foi utilizado este ultimo tipo.

5.5.3. Obtendo Informação Passiva

Foram realizados dois tipos de testes nos receptores. No primeiro foram realizadas varreduras simples e não destrutivas com as seguintes ferramentas comuns em redes de computadores: Nessus (versão livre/gratuita), Nmap/ZenMap (livre/gratuito) e WireShark (livre/gratuito). Para este teste buscou-se respostas para a seguinte questão: Há nos receptores TVDi de mercado vulnerabilidades comuns aos computadores em rede?

A primeira varredura foi executada na interface de rede existente (porta ethernet) no receptor. A preocupação era a obtenção da identificação dos receptores na rede (determinação de endereços IP). A segunda foi à realização de varredura simples com perfil básico das ferramentas, e a terceira foi o monitoramento da comunicação realizada por aplicação interativa a partir do receptor via rede.

Os resultados foram os seguintes. Foi possível capturar dados HTTP em texto claro enviados e recebidos por aplicação interativa instalada no receptor localizado no mesmo barramento da máquina de varredura. Neste caso a aplicação não possuía controles de segurança implementados, isto é, comunicação segura com HTTPS. E por fim, foram identificadas vulnerabilidades possíveis de exploração em ataques. Vulnerabilidades comuns aos equipamentos em rede: TCP *timestamps* (*uptime* pode ser computado); Ethernet; portas abertas sunrpc (111/tcp e 111/udp). Os serviços do protocolo RPC pode possibilitam identificação e conexão remota aos serviços ativos e o RPC *portmapper* permite a identificação de outros serviços RPC no próprio receptor; vulnerabilidade associada ao ICMP. *Timestamp Request Remote Date Disclosure* – receptor responde a requisições de tempo. Permite ao atacante determinar o tempo exato configurado no receptor. Pode ser usada na exploração de protocolos de autenticação.

A SANS1 fornece uma lista de vulnerabilidades para a porta TCP 111. Após esta analise foi possível concluir que a vulnerabilidade desta porta 111 se encontrada aberta só poderá ser explorada por um canal de retorno (acesso via Internet), não sendo tratado nesta analise preliminar.

Este primeiro teste possibilitou aos autores a descoberta do sistema operacional utilizado por um dos modelos na análise. A partir desta revelação foi possível investigar maiores detalhes quanto as vulnerabilidades neste sistema operacional embarcado, bem como as vulnerabilidades de hardware, do *middleware* Ginga NCLua, e da aplicação presentes no receptor. Para a exploração destas vulnerabilidades foi identificada a necessidade de desenvolvimento de uma Aplicação que possibilitasse a investigação interna no receptor digital, a qual será detalhada a seguir.

[1] http://www.sans.org/security-resources/idfaq/blocking.php

5.5.4. A Interface restrita no Receptor

Um dos aspectos mais diferenciados do desenvolvimento de aplicações para os receptores de televisão digital brasileiros é a interface restrita para a interação homem-maquina. De modo geral, o único dispositivo de entrada de comandos pelo usuário é o controle remoto da TV, o qual é geralmente mais limitado em opções de comandos que um telefone celular comum com teclado numérico e poucas teclas extras.

Esta dificuldade (independe da resolução da imagem ou do tamanho da tela da TV) relativa de interação homem-máquina afeta não somente a usabilidade dos aplicativos, mas também a exploração, a partir da interface com usuário, das vulnerabilidades presentes nas camadas de software subjacentes. Trata-se neste texto do uso do receptor como plataforma de computação, daí a dificuldade de interação e entrada de dados.

Dado o padrão atual de utilização destes receptores e a hipótese de evolução das técnicas de ataque aos receptores ser análoga às dos *smartphones* e PCs, surgem às seguintes questões:

1. Seria possível explorar as vulnerabilidades conhecidas em aplicações web ou móveis (p.ex. OWASP top 10) nas aplicações para TVDi?
2. Ainda, como seria possível testar a segurança de aplicações (realizar testes de intrusão) destas aplicações, dadas as limitações de IHC?
3. Outra questão é a execução de scripts de comandos a partir de ferramenta construída para injeção de comandos no receptor. Os scripts de teste de segurança foram exercitados no simulador Ginga-NCL Virtual STB e em receptores TVDi de mercado com Ginga-NCL e Lua embarcados.

Durante a analise dos receptores digitais verificou-se uma facilidade presente em uma caixa comercial onde o usuário habilitado no sistema operacional era o root, e esta descoberta do usuário facilitou a investigação mais detalhada das vulnerabilidades presentes em alguns receptores de TVDi atualmente disponíveis no mercado.

A analise possibilitou ainda, o desenvolvimento de uma investigação automatizada e a alteração dos dados existentes no equipamento em questão, por exemplo, a substituição de contas e senhas do arquivo *passwd*, a remoção de arquivos e as alterações de dados relevantes ao sistema.

Para a execução do teste de penetração no receptor houve a preocupação em obter informações de hardware, do sistema operacional, do *middleware,* da aplicação, e de usuário. Para conseguir uma interface junto ao receptor, foi necessário o desenvolvimento de uma ferramenta denominada AIC – Aplicativo de Injeção de Comandos. Esta ferramenta é um conjunto de interfaces (Figura 10) somadas a alguns scripts automatizadas os quais permitem a Injeção de Comandos, e a Coleta de resultados após a execução pelo receptor. A ferramenta é carregada via USB.

Figura 10 - Aplicativo de Injeção de Comandos

Na camada de Hardware do receptor, por meio de comandos injetados via AIC foi possível identificar a capacidade do equipamento em termos de memória, processamento. A questão da limitação do hardware, quando os comandos de investigação exigiam maior carga de processamento por parte do receptor demandou maior tempo na investigação.

Um dos equipamentos (caixa comercial) investigados apresentava um hardware cuja capacidade de memória é de 1 kbps, e com um processador interno.

```
1. Filesystem           1k-blocks      Used Available Use% Mounted on
2. /dev/mtdblock2           20992     20992         0 100% /
3. /dev/mtdblock3           21504     11660      9844  54% /root
4. /dev/sda1              1952192   1763872    188320  90% /mnt
```

Coleta 1 - Informações de Memória do Receptor

```
1. machine: dados omitidos por medidas de sigilo
2. processor: 0
3. cpu family: sh4
4. cpu type: IDx (nome do equipamento omitido)
5. cpu flags: fpu
6. cache type: split (harvard)
7. icache size: 16KiB (2-way)
```

```
8.  dcache size: 32KiB (2-way)
9.  pll0_clk    : 530.00MHz
10. pll1_clk    : 400.00MHz
11. sh4_clk     : 265.00MHz
12. sh4_ic_clk  : 132.50MHz
13. module_clk  : 66.25MHz
14. slim_clk    : 265.00MHz
15. comms_clk   : 100.00MHz
16. tmu0_clk    : 16.56MHz
```

Coleta 2 - Informações da CPU do Receptor

Foi possível verificar que em alguns receptores o SO embarcado encontra-se com vulnerabilidades e que não há preocupação com a segurança, apresentando algumas vulnerabilidades como excesso de contas onde os usuários não estão marcados como *no login*, et al.

```
1.  root::0:0:root:/root:/bin/sh
2.  bin:*:1:1:bin:/bin:
3.  daemon:*:2:2:daemon:/usr/sbin:
4.  sys:*:3:3:sys:/dev:
5.  adm:*:4:4:adm:/var/adm:
6.  lp:*:5:7:lp:/var/spool/lpd:
7.  sync:*:6:8:sync:/bin:/bin/sync
8.  shutdown:*:7:9:shutdown:/sbin:/sbin/shutdown
9.  halt:*:8:10:halt:/sbin:/sbin/halt
10. mail:*:9:11:mail:/var/spool/mail:
11. news:*:10:12:news:/var/spool/news:
12. uucp:*:11:13:uucp:/var/spool/uucp:
13. operator:*:12:0:operator:/root:
14. games:*:13:100:games:/usr/games:
15. ftp:*:15:14:ftp:/var/ftp:
16. man:*:16:100:man:/var/cache/man:
17. nobody:*:65534:65534:nobody:/home:/bin/sh
18. httpd:*:75:75:httpd:/home/httpd:/bin/false
19. sshd:*:74:74:sshd:/var/empty/sshd:/bin/false
```

Coleta 3 - Arquivo passwd

5.6. Recomendações de segurança

Esta seção traz uma abordagem sistêmica para a segurança da informação e dos sistemas embarcados nos receptores de TVDi. Serão tratados aspectos de desenvolvimento seguro de aplicações, em particular aspectos de projeto seguro de sistemas, programação segura, robustecimento de sistemas operacionais, segurança em redes e implantação e operação seguras. Todos estes aspectos de segurança serão contextualizados na utilização do receptor de TVDi como dispositivo de computação e comunicação.

5.6.1. Limites da segurança de sistemas

Com uma freqüência bastante grande, os programadores são induzidos a falhas de julgamento quando o assunto é segurança da informação. Alguns exemplos são os seguintes: o uso de criptografia seria suficiente para garantir o sigilo das informações; ou ainda, revisões de código e testes de penetração seriam capazes de identificar todas as vulnerabilidades em programas; entre outras. O programa absolutamente seguro é um mito, simplesmente pelo fato de que é impossível determinar se um programa está livre de defeitos que levem a vulnerabilidades e falhas de segurança.

Harrison, Ruzzo, and Ullman (1976) desmistifica a noção de que existe segurança absoluta (por exemplo, "site 100% seguro") em sistemas de computação. Uma das provas é que não há algoritmo genérico capaz de decidir sobre a segurança de uma configuração qualquer de um sistema de proteção qualquer. A força desta afirmação deve ser entendida. Não há esperança de encontrar um algoritmo que possa atestar a segurança de uma configuração arbitrária de um sistema de proteção qualquer, ou de todas as configurações de um determinado sistema. Tal problema é indecidível e remonta ao problema da parada, estabelecido por A. Turing nos anos 30 do século passado.

Outro ponto relevante é o levantado por Thompson (1984). Programas são gerados e executados por outros programas, em uma pilha de execução que vai até o hardware. Thompson (1984) argumenta que vulnerabilidades podem ser exploradas em qualquer nível da pilha de software. Por isso, seria necessário verificar a correção e a integridade de toda a pilha de execução para garantir a confiança no funcionamento do programa do topo da pilha. Thompson (1984) sugere que em casos críticos só é possível confiar em programas de produção doméstica própria. Qualquer outra fonte de programas deve ser considerada suspeita e não merece confiança. Em uma situação extrema, um programador só deveria confiar nos programas escritos por ele mesmo, deste o sistema operacional, até o compilador e as bibliotecas de apoio.

Estes resultados não devem inibir os esforços em verificação de sistemas e segurança de sistemas. Pois, mesmo sem uma solução genérica ampla, há aplicações práticas em casos restritos capazes de atender a maior parte das situações do dia-a-dia. A questão levantada neste texto é como aplicar as boas práticas de segurança da informação ao desenvolvimento seguro de sistemas e a programação segura em Lua. Como estas boas práticas se manifestam em Lua e em NCL?

5.6.2. Desenvolvimento seguro de software

De acordo com Viega and McGraw (2001), Howard and LeBlanc (2002) e Braga (2007), a tendência atual para a segurança de sistemas de software é a prevenção de

vulnerabilidades pela consideração dos aspectos de segurança o mais cedo possível no desenvolvimento dos sistemas. Esta tendência se opõe à cultura reativa de remediação após incidentes. A prevenção de vulnerabilidades nos primeiros estágios do desenvolvimento dos sistemas diminui o custo da segurança, pois costuma ser mais barato evitar a inclusão de vulnerabilidades, que achá-las em testes e corrigi-las. Ainda, a correção de vulnerabilidades encontradas durante testes tem custo menor que a remediação de sistemas em produção ou de produtos finalizados.

A segurança é um aspecto da qualidade dos sistemas. A inclusão da segurança antecipadamente no desenvolvimento dos sistemas aumenta a confiança em um produto final com qualidade. Não é possível ter certeza absoluta da ausência de defeitos, porém a maior confiança no software resultante seria justificada pela utilização de ferramentas apropriadas juntamente aos processos de garantia de qualidade e controle de qualidade do produto final. A Figura 11 ilustra um processo de desenvolvimento seguro de sistemas que a implementação de mecanismos de segurança (segurança funcional) e a consideração de propriedades arquitetônicas na segurança (segurança não funcional), assim como atividades de garantia de qualidade. A Figura 12 ilustra os entregáveis de software, relativos a segurança, distribuídos pelo desenvolvimento de sistemas e acompanhados dos entregáveis tradicionais.

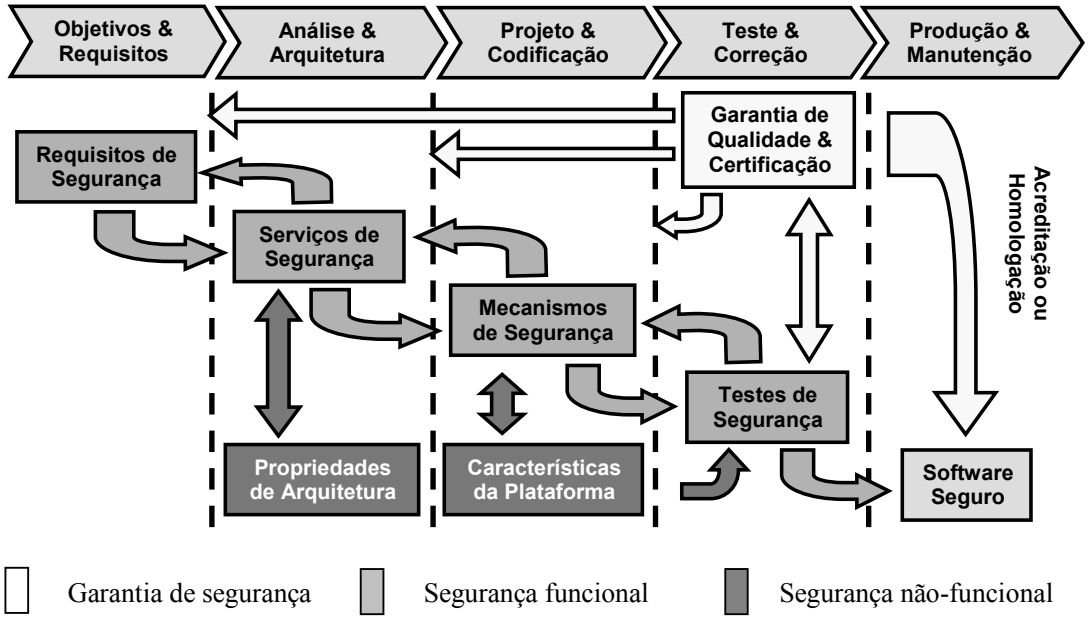


Figura 11: Processo de desenvolvimento seguro de software.

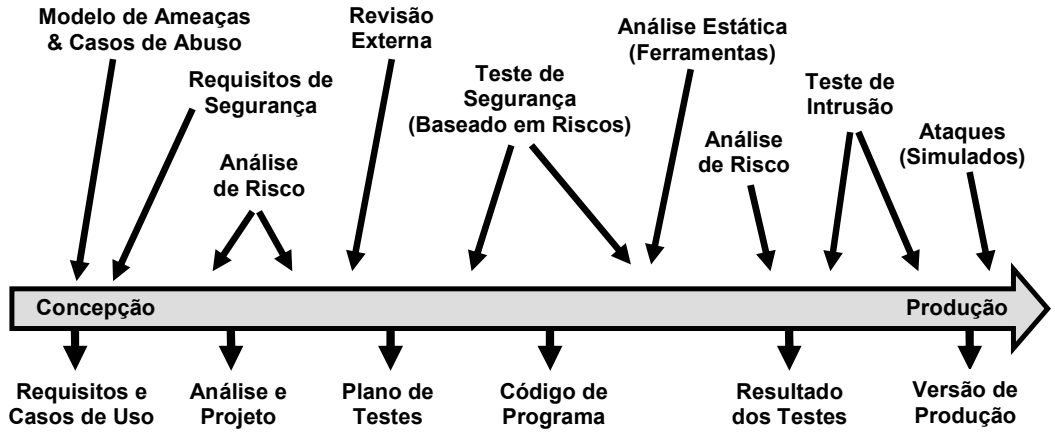


Figura 12: Entregáveis do desenvolvimento seguro de software.

5.6.3. Regras de ouro do software seguro

Em um trabalho pioneiro, Saltzer and Schroeder (1975), foi estabelecido um conjunto bastante famoso de princípios para o desenvolvimento de softwares seguros. Este conjunto é apresentado a seguir.

(1) Economia de Mecanismos: O projeto de segurança deve ser o mais simples e menor possível. Em sendo simples, conterá menos vulnerabilidades. Em sendo pequeno, poderá ser verificado.

(2) Falha Segura: Diante de uma falha, o sistema deve sempre adotar o estado mais seguro para o recurso de valor. Por exemplo, se as decisões de acesso são baseadas em permissões e não em proibições; então tal configuração torna possível ao sistema negar o acesso sempre que não for possível verificar a existência de uma permissão explícita.

(3) Mediação Completa: Todo acesso a todo recurso computacional deve ser verificado. Deve sempre existir um caminho confiável entre o usuário e o recurso computacional, de tal forma que seja sempre possível verificar as permissões de acesso. Atalhos para administradores (backdoors) quebram este princípio.

(4) Projeto Aberto: A segurança não deve ser baseada em segredos de construção e tecnologias proprietárias. Muitas plataformas de software modernas (bastante usadas em sistemas corporativos) possuem programas facilmente descompilados. A segurança por obscuridade, considerada uma prática de eficácia duvidosa, está bastante comprometida pelo avanço das técnicas de engenharia reversa.

(5) Separação de Privilégios ou de Responsabilidades: A criação de usuários todo-poderosos deve ser evitada. Um único usuário (por mais importante que seja) nunca dever ser capaz de realizar sozinho todos os passos de uma tarefa importante, pois tal usuário poderia agir em benefício próprio, sem que outros saibam. Para inibir a fraude, a responsabilidade sobre uma tarefa complexa deve ser dividida entre dois ou mais usuários. Nesta situação, a fraude só seria possível pela conspiração das partes envolvidas.

(6) Privilégios Mínimos: O usuário deve possuir apenas as permissões necessárias para a realização das tarefas necessárias a sua função da organização. Este princípio

complementa o anterior. O usuário com permissões insuficientes é ineficiente, pois não consegue realizar suas funções. Por outro lado, o usuário com permissões em excesso expõe os recursos da organização a riscos desnecessários.

(7) Mínimo Denominador Comum: O compartilhamento de informação de segurança entre muitos usuários deve ser evitado. O compartilhamento de senhas, de chaves criptográficas, de segredos e de outros parâmetros de segurança, entre vários usuários deve ser evitado. Este compartilhamento dificulta a autenticação inequívoca e a responsabilização do usuário.

5.6.4. Codificação (ou programação) segura

Normas de codificação são ferramentas de desenvolvimento seguro, na parte de codificação segura. Deve ser observado que qualquer norma de codificação deverá ser complementada por templates, cujo uso correto poderia ser validado por ferramentas automáticas de análise estática de código.

No que se refere à aplicação de orientação a objetos em Lua, as técnicas de segurança como as sugeridas em (Programming best practices for Java) não podem ser aplicadas diretamente, mas podem servir de inspiração para as boas práticas OO específicas em Lua.

Em relação a boas práticas genéricas para programação segura, diversas práticas podem ser aplicadas. De fato, o catálogo de vulnerabilidades de programação em Lua, descrito em outra parte deste texto, sinaliza diversas estratégias de mitigação das vulnerabilidades. Estas estratégias são aplicações das técnicas e boas práticas de programação segura em Lua.

Não há, até onde vai o conhecimento dos autores deste texto, documentação ou ferramentas de programação segura para Lua. Foram investigadas diversas fontes de informação e observou-se uma ausência completa de informação formal sobre boas práticas de segurança de software em Lua. Dentre as fontes investigadas, podem ser citadas as seguintes como mais relevantes: Lerusalimschy (2003), Lerusalimschy, Figueiredo e Celes (2006), Barbosa e Soares (2008), NCLua Tutorial, Comunidade Lua.

Dois catálogos mais tradicionais de vulnerabilidades de software, SANS/CWE Top 25 (2010) e OWASP Top 10 (2010), não fazem qualquer referência as vulnerabilidades de Lua e nem de NCL.

A norma ABNT NBR 15605-2 define diversos mecanismos de segurança, mas não trata do uso programático seguro destes mecanismos. O mesmo pode ser dito sobre a documentação dos pacotes de software LuaCrypto e LuaSec, ambos responsáveis pelo uso de criptografia em NCLua.

Por outro lado, há em Lua a oportunidade para ferramentas de análise estática. Em duas modalidades, a análise léxica e a análise sintática. Segundo Evans and Larochelle (2002), a ferramenta de análise léxica faz, a grosso modo, uma busca por palavras chave da linguagem e, em princípio, não precisaria entender a sintaxe da linguagem de programação. Um exemplo simples deste tipo de ferramemta é o OWASP CodeCrawler. Ainda de acordo com Evans and Larochelle (2002), a ferramenta de análise sintática faz a varredura de código fonte do programa Lua e, por isto, precisa entender a sintaxe da

linguagem e identificar programas válidos de acordo com ela. Exemplos simples são a ferramenta PMD e o *bytecode verifier* da plataforma Java.

5.6.5. Separação de responsabilidades, compartimentação

Dois conceitos geralmente presentes em políticas de segurança voltadas para controle de acesso são a compartimentação e a separação de responsabilidades.

A compartimentação estabelece a separação de um conjunto ou grupo de entidades em partes isoladas, geralmente sem comunicação entre si. Cada parte agrupa elementos semelhantes e relacionados, de modo que a compartimentação também serve para classificação. Em controle de acesso, a compartimentação não apenas evita a interferência das ações de uma entidade sobre as de outra, mas também impede o conhecimento de uma entidade sobre os assuntos de outras, garantindo o sigilo das informações e a privacidade das pessoas. A técnica de sand-boxing, utilizada atualmente de modo informal em Lua, pode ser um mecanismo poderoso de compartimentação e isolamento de aplicações e do sistema operacional, tal e qual ocorre nos *smartphones* com a plataforma Android da Google, segundo Enck, Ongtang and McDaniel (2009) e Shabtai et al (2010).

A separação de responsabilidades estabelece a divisão de uma transação (tarefa maior) em tarefas menores realizadas por entidades distintas, a fim de evitar que uma única entidade sozinha seja capaz de realizar a transação completa. Uma vantagem direta é a inibição de fraudes individuais, pois cada entidade verifica o trabalho realizado nas etapas anteriores. Com a separação de responsabilidades, a fraude só é possível pela conspiração ou conluio entre as entidades. Quando as aplicações estão associadas a um único usuário root. O usuário de serviço ganha privilégios de administrador e os princípios de separação de responsabilidades e compartimentação não são obedecidos.

5.6.7. Monitor de Referências

Um conceito bastante usado na implementação de mecanismos de controle de acesso é o de monitor de referências, explicado pela primeira vez em Anderson (1972). O monitor é uma entidade mediadora que autoriza ou proíbe o uso de um recurso toda vez que um acesso é requisitado por uma entidade do sistema. Ainda de acordo com Anderson (1972), uma implementação ideal de um monitor de referências deve possuir as seguintes características: (1) Deve ser inviolável (*tamper-proof*) ou, no mínimo, oferecer meios de detecção de violação (*tamper-evident*). (2) Deve mediar todos os acessos a qualquer recurso, isto é, deve ser invocado para todo e qualquer acesso a um recurso. (3) Deve estar confinado em um perímetro definido e facilmente identificável (*security kernel*). (4) Deve ser suficientemente pequeno para que possa ser analisado e verificado quanto a correção de funcionamento.

Uma implementação real do monitor de referências, em geral, negligencia uma ou outra característica. Em particular, implementações em software não costumam ser invioláveis. Por exemplo, as implementações em software do monitor de referências são geralmente encapsuladas em bibliotecas de software e usam códigos de integridade ou de autenticação de programas como mecanismo de detecção de violação.

5.7. Considerações Finais

Em relação à segurança de aplicações, foram identificadas e documentadas diversas vulnerabilidades de programação insegura em Lua. Tais como injeção de comandos, condição de competição, corrupção de arquivos e código malicioso, script cruzado armazenado, referencia insegura a tabela, injeção de SQL e mau uso de criptografia.

Em relação à avaliação de segurança de receptores, foram realizadas varreduras de vulnerabilidades via interface de rede e testes caixa branca de injeção de comandos. Foram encontradas vulnerabilidades graves na configuração dos sistemas operacionais e de proteção de aplicações e seu ambiente de execução.

No futuro próximo, os receptores de TVDi, assim como está acontecendo com os *smartphones* e já aconteceu com os PCs, se tornarão uma fonte importante de vazamento de informações privadas em redes públicas e ambientes de computação em nuvem. Além disso, eles estarão na fronteira de proliferação dos softwares maliciosos e, em conjunto aos computadores pessoais, os *smartphones* e as redes corporativas serão vetores de ataques maciços às redes de computadores e de telecomunicações. As três ameaças que oferecem maior risco seria as seguintes: a contaminação de receptores por softwares maliciosos, o uso destes receptores contaminados como vetores de ataques maciços de negação de serviço e as violações de sigilo e de privacidade decorrentes da descentralização dos ambientes de computação. A primeira ameaça causa uma variedade de problemas, desde a revelação de informação privada até o desvio de recursos do aparelho para outras finalidades. A segunda ameaça afeta diretamente a disponibilidade de serviços na rede subjacente, devido à interconexão das redes a Internet. A terceira ameaça revela a exposição de informações privadas em ambientes de computação em nuvem.

Muito pode ser aprendido com as estratégias de segurança adotadas pelos fabricantes de dispositivos móveis. A despeito de não sofrerem da vulnerabilidade de consumo de bateria, os receptores de TVDi, no que diz respeito a segurança de aplicativos, possuem praticamente o mesmo modelo de ameaças dos *smartphones*. Não é surpresa que a norma brasileira para segurança de aplicativos Ginga seja, em diversos aspectos, semelhante ao modelo de segurança das plataformas modernas de *smartphones*.

5.7.1. Trabalhos futuros

Este texto é o início de um trabalho longo, de esforço contínuo, para dar um tratamento mais amplo à segurança da plataforma de televisão digital interativa brasileira. Há diversas oportunidades para trabalhos futuros.

Em primeiro lugar, a investigação contínua das vulnerabilidades de programação, não somente em Ginga-NCL e Lua, mais também em Ginga-J. Levando a expansão do catálogo de vulnerabilidades iniciado neste texto. Ainda, a elaboração de boas práticas de programação e desenvolvimento de software seguro com a plataforma de TVDi.

Em seguida, avaliação de segurança em implementações de mercado da plataforma Ginga de TVDi, incluindo receptores, simuladores e outros pacotes de software, podendo levar a determinação de normas de segurança mais específicas.

Além disso, outra área a ser explorada é a definição e especificação de requisitos de segurança para serviços ou aplicações de *T-commerce, T-gov e T-banking*, que oferecem no mínimo o mesmo grau de segurança dos serviços equivalentes na Internet.

Finalmente, a construção de ferramentas e de componentes de segurança específicos para a plataforma de TVDi brasileira. Abrangendo tanto ferramentas de apoio ao desenvolvimento seguro, como também a implementação de controles que atendam a determinados requisitos de segurança de aplicações.

5.7.2. Agradecimentos

Os autores gostariam de agradecer a Andre Dorte Dos Santos, a Jéssica Caroline da Silva, a Tadeu Aparecido L. Secco Gessoli, e a todos os profissionais envolvidos no Projeto SMTVI ,pela grande dedicação e ajuda inestimável durante a realização deste trabalho e sem as quais os resultados obtidos não teriam sido possíveis.

Referências

ABNT NBR 15606-2 (2007). Associação Brasileira de Normas Técnicas. Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital. Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações. ABNT 2007. ISBN 978-85-07-00583-4.

ABNT NBR 15605-1 (2008). Associação Brasileira de Normas Técnicas. Televisão digital terrestre — Tópicos de Segurança. Parte 1: Controle de cópias. ABNT 2008. ISBN 978-85-07-01041-8

ABNT NBR 15605-2 (em preparação). Associação Brasileira de Normas Técnicas. Televisão digital terrestre — Tópicos de Segurança. Parte 2: Mecanismos de segurança para aplicativos interativos.

Anderson, R. (1993). Why cryptosystems fail. In Proceedings of the 1st ACM Conference on Computer and Communications Security (Fairfax, Virginia, United States, November 03 - 05, 1993). CCS '93. ACM, New York, NY, 215-227.

Braga, A. M. (2007). Visão geral das boas práticas para construção de softwares seguros. Revista Técnica IPEP, São Paulo, SP, V. 7, N. 2, p. 65-78, jul./dez. 2007. ISSN 1807-8125.

Bragra A. M.; Restani, G. S**.** (2010). Hacking Ginga: uma avaliação de segurança da plataforma de aplicações interativas da TV digital brasileira Anais do X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (No prelo 2010).

CWE/SANS Top 25 (2010). CWE/SANS Top 25 Most Dangerous Programming Errors. Version 2.0, 2010. Disponível on-line nas URLs www.sans.org/top25-programming-errors e cwe.mitre.org/top25

Deming, W. Edwards (1986). Out of the Crisis. MIT Center for Advanced Engineering Study. ISBN 0-911379-01-0.

Ginga-NCL Virtual STB. Máquina virtual Linux para VMWare e simulador de receptor com Ginga-NCL. Disponível on-line na URL www.gingancl.org.br

Howard, M and LeBlanc, D (2002). Writing Secure Code, Second Edition. December 04, 2002. ISBN 9780735617223

Kocher, P., Lee, R., McGraw, G., and Raghunathan, A. 2004. Security as a new dimension in embedded system design. In Proceedings of the 41st Annual Design Automation Conference (San Diego, CA, USA, June 07 - 11, 2004). DAC '04. ACM, New York, NY, 753-760.

Lerusalimschy, R. (2003). Programming in Lua, 1st. Ed. 2003. ISBN 85-903798-1-7. Disponível on-line na URL www.lua.org/pil.

Lerusalimschy, R., Figueiredo L. H. e Celes, W. (2006). Lua 5.1 Reference Manual. 2006. ISBN 85-903798-3-3. Disponível on-line na URL www.lua.org/manual/5.1/pt.

Lua MD5. Cryptographic Library for Lua. MD5 and DES56 basic cryptographic facilities for Lua. http://www.keplerproject.org/md5

LuaCrypto. LuaCrypto - A Lua Frontend to OpenSSL. Disponível on-line na URL luacrypto.luaforge.net/index.html

LuaSec. LuaSec - TLS/SSL Support for Lua. Disponível on-line na URL www.inf.puc-rio.br/~brunoos/luasec/index.html.

Nicholls, R. (2000) SMS – Today´s Interactive Television. Australia, http://www.broadcastpapers.com.

Oberheide, J. and Jahanian, F. (2010) When mobile is harder than fixed (and vice versa): demystifying security challenges in mobile environments. In Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications (Annapolis, Maryland, February 22 – 23, 2010). HotMobile '10. ACM, New York, NY, 43-48. 2010.

OWASP Top 10 (2010). The Ten Most Critical Web Application Security Risks. 2010. Disponível on-line na URL www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Ravi, S., Raghunathan, A., Kocher, P., and Hattangady, S. (2004). Security in embedded systems: Design challenges. ACM Trans. Embed. Comput. Syst. 3, 3 (Aug. 2004), 461-491.

Viega, J. and McGraw, G. (2001). Building Secure Software: How to Avoid Security Problems the Right Way. Addison-Wesley Professional. October 4, 2001. ISBN 978-0201721522.

Wilhelm, Thomas. (2010) Professional Pentration Testing, Creating and Operating a Format Hacking Lab – SYNGRESS 2010. ISBN: 978-1-59749-425-0.

Capítulo

6

Estratégias de Contingência para Serviços de Tecnologia da Informação e Comunicação

Leonardo L. Fagundes[1], Fernando Karl[1], Luis Baptista[2] e Rafael Santos da Rosa[3]

Universidade do Vale do Rio dos Sinos – UNISINOS

[1]{llemes, fkarl}@unisinos.br; [2]luis_baptista@sicredi.com.br; [3]rsrosa@reno.unisinos.br

Abstract

The business continuity management is a process that identifies threats and their possible impacts. This process provides an appropriate structure for the organization to respond effectively in case of incidents. The purpose of this chapter is to present the theoretical and practical aspects of business continuity management with focus to the preparation disaster recovery plans.

Resumo

A gestão da continuidade de negócio é um processo de gestão que identifica ameaças e os seus possíveis impactos. Este processo fornece uma estrutura adequada para que a organização responda efetivamente em casos de incidentes. O objetivo desse curso é apresentar os aspectos teóricos e práticos da gestão da continuidade de negócio com foco para a elaboração dos planos de recuperação de desastres.

6.1. Introdução

Segundo as normas [ABNT 2008a] e [ABNT 2008b], a Gestão de Continuidade de Negócio (GCN) é um processo abrangente de gestão que identifica ameaças potenciais para uma organização e os possíveis impactos nas operações de negócio, caso estas ameaças se concretizem. Portanto, a mesma atua de forma proativa na organização, a fim de melhorar a resiliência da organização contra ruptura ou interrupção de sua capacidade de fornecer seus produtos ou serviços.

De acordo com a [IBM Global Services 2010], a continuidade dos negócios é fundamental para o sucesso das empresas e devido à grande interdependência tecnológica atual dos processos de negócio, praticamente todos os aspectos da operação estão suscetíveis a falhas.

Conforme [Continuity Central 2010], para 32% das organizações, apenas quatro horas de tempo de inatividade podem ser fatais. Pode ser tomado, por exemplo, o incidente envolvendo a Nokia e a Ericsson. A interrupção do processo de produção da Ericsson, causada por falha de um fornecedor principal do seu processo produtivo, quase a tirou do mercado mundial de celulares. A Phillips, àquela época, era a principal fornecedora de microchips tanto para a Ericsson quanto para a Nokia, e quando ocorreu um incêndio na sua planta mexicana, houve avarias em praticamente todo o seu estoque.

Neste ponto, foi perceptível a diferença na maturidade das estratégias das duas empresas, quanto à continuidade dos seus negócios. Enquanto a Ericsson aceitou as estimativas otimistas da Phillips, que em poucos meses a produção estaria restabelecida, a Nokia foi à busca de outros fornecedores de microchips, inclusive alterando a tecnologia adotada em seus telefones, assim tornando-os compatíveis com os novos chips. Quando a Ericsson percebeu que não poderia mais ficar esperando o retorno do seu principal fornecedor, a Nokia já havia garantido a capacidade produtiva dos principais outros fabricantes mundiais. A falta de estratégia adequada para este desastre, aliada a falta de rapidez ao atuar sobre o mesmo, causou a Ericsson um grande prejuízo financeiro, como também uma grande perda de mercado, então assumido pela sua concorrente, a Nokia [Husdal 2008].

Em outros casos de desastres como aquele ocorrido no World Trade Center se observa situações ainda mais criticas, por exemplo, a empresa Cantor Fitzgerald perdeu com a queda de uma das torres 700 funcionários, talento e conhecimento referente aos seus processos, já com a queda da segunda torre essa mesma empresa perdeu todas as suas cópias de segurança e informações armazenadas, somente restando a esta empresa, a falência e a extinção.

Toda a organização está, com maior ou menor probabilidade, suscetível a interrupções das suas atividades críticas ocasionadas por ameaças tais como: falhas tecnológicas, enchentes, interrupções nos serviços públicos e atos de terrorismo. O objetivo desse capítulo é apresentar a Gestão da Continuidade como um aspecto de fundamental relevância para que uma organização possa responder de maneira eficiente aos cenários de incidentes e manter a continuidade dos serviços TI considerados críticos.

O capítulo em questão está estruturado conforme a descrição a seguir:

Seção 6.2: apresenta os diversos estágios do ciclo de vida da Gestão da Continuidade de Negócios, conforme as normas brasileiras [ABNT, 2008a] e [ABNT, 2008b];

Seção 6.3: descreve algumas das boas práticas internacionais empregadas para o desenvolvimento de estratégias de contingência em tecnologia da informação;

Seção 6.4: relata um estudo de caso, cujo objetivo é propiciar a reflexão e a aplicação dos conceitos e práticas apresentadas anteriormente no

desenvolvimento de estratégias de contingência para os serviços de tecnologia da informação e comunicação considerando uma companhia aérea fictícia;

Seção 6.5: encerra o capítulo a partir de um resgate dos objetivos propostos e da síntese dos principais aspectos apresentados ao longo do minicurso, com destaque para as questões relacionadas aos desafios da implementação e manutenção dos planos de recuperação de desastres.

6.2. Ciclo de Vida da Gestão de Continuidade de Negócios

Segundo Código de Prática para a Gestão de Continuidade de Negócios, NBR 15999 – Parte 1, o ciclo de vida da Gestão de Continuidade de Negócios segue a estrutura ilustrada Figura 6.1 [ABNT 2008a]:

Figura 6.1. Ciclo de Vida da Gestão da Continuidade de Negócios.

Este ciclo representa as etapas da Gestão de Continuidade de Negócios, em que se inicia pela Gestão do Programa de GCN, onde são designadas as responsabilidades e como será executada a gestão continua do programa. Contempla também a definição e requisitos de documentação que farão parte do programa.

Em entendendo a organização, é realizada a Análise de Impacto nos negócios (AIN/BIA) a identificação das atividades críticas, a determinação dos requisitos de continuidade, a análise de risco e definição de que ações serão tomadas quanto aos riscos identificados.

Já na etapa de determinando a estratégia de continuidade de negócios, são escolhidas as estratégias para as pessoas, instalações, tecnologia, informação e suprimentos que fazem parte do processo de negócio alvo da GCN.

Na etapa de desenvolvimento e implementação de uma resposta de GCN, é estruturada a Resposta a Incidentes e são criados os planos de Gerenciamento de Incidentes e o Plano de Continuidade de Negócios.

Em Testando, mantendo e analisando criticamente os preparativos de GCN, ocorre a validação dos testes e análises dos preparativos de GCN, Programa de Testes, a manutenção do programa, a análise crítica (auditoria interna), a Auditoria externa e o processo de auto-avaliação.

Na última etapa do ciclo de vida da GCN, Incluindo a GCN na cultura da organização, é onde são tratados os requisitos de treinamento e conscientização no que tange a GCN.

Antes de iniciar a descrição de cada um dos estágios do ciclo de vida é fundamental o entendimento de algumas definições adotadas pela ABNT NBR 15999-1 [ABNT 2008a]. A Figura 6.2 representa os principais intervalos de tempo considerados na gestão da continuidade de negócios.

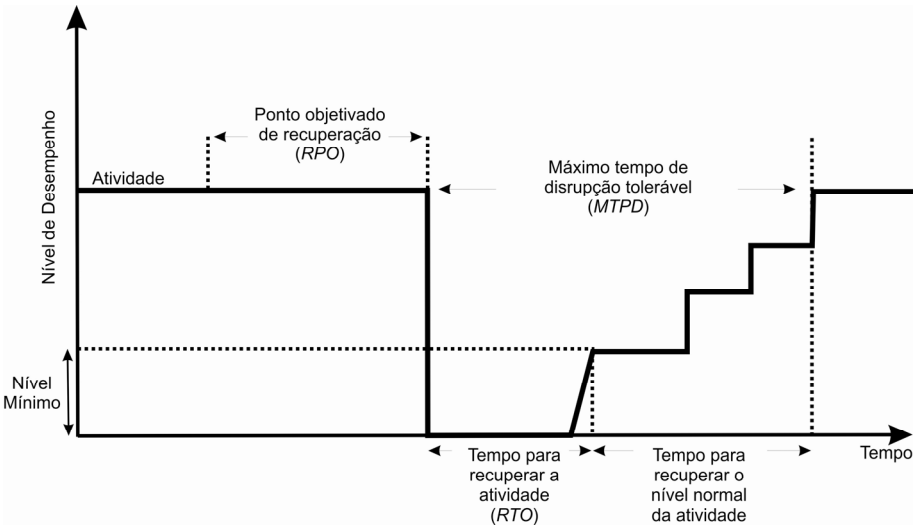


Figura 6.1. Representação dos tempos considerados em GCN.

- **Período máximo de interrupção tolerável (MTPD – maximum tolerable period of downtime)**

  Duração a partir da qual a viabilidade de uma organização será ameaçada de forma inevitável, caso a entrega de produtos e serviços não possa ser reiniciada.

- **Tempo objetivado de recuperação (RTO – recovery time objective)**

  Tempo alvo para: (a) retomada da entrega de produtos ou serviços após um incidente; ou (b) recuperação do desempenho de uma atividade após um incidente; ou ainda (c) recuperação de um sistema ou aplicação de TI após um incidente.

- **Ponto objetivado de recuperação (RPO – recovery point objective)**

  Posição (no tempo) na qual deverão estar disponíveis os dados das Aplicações recuperadas após a ocorrência de um desastre. O RPO está diretamente relacionado ao processo e freqüência de geração de cópias de segurança

6.2.1. Gestão do Programa de GCN

A gestão do programa de GCN é a estrutura principal de um processo de GCN. Na gestão do programa que é estabelecida a abordagem que a organização terá em relação à continuidade de negócios.

Importante observar a importância da participação da alta direção na introdução da GCN na cultura organizacional. Na gestão do programa, objetivando atender a Política de Gestão de Continuidade de Negócios, são desenvolvidas as seguintes etapas:

- **A atribuição de responsabilidades;**

  Nesta etapa, a direção da empresa necessita designar uma pessoa com a senioridade e autoridade necessárias para ser responsável pela política de GCN e a sua implementação.

  Também é necessário que se aponte um ou mais indivíduos para implementar e manter o programa de GCN. Assim formando a equipe de GCN da organização. Importante observar que na criação desta equipe, podem-se nomear representantes de outras áreas e/ou níveis de negócio para apoiar a implementação da GCN.

- **A implementação da continuidade de negócios na organização;**

  Na fase de implementação, é importante que contemplem o planejamento, o desenvolvimento e implementação do programa.

  Durante a implementação, deve-se prever a comunicação do programa às partes interessadas, organizar e fornecer treinamento apropriado a equipe, e realizar testes da capacidade de Continuidade de negócios da Organização.

  Para apoiar a etapa de implementação, a organização pode utilizar uma metodologia de gerenciamento de projetos reconhecida para garantir uma implementação efetiva.

- **A gestão contínua da continuidade de negócios.**

  Na etapa da gestão contínua, convém seja assegurada a incorporação da GCN na cultura da organização. Como também se prevê o mantenimento dos componentes da Continuidade de negócios, como também a análise crítica e atualização dos planos e soluções de continuidade de negócios.

  Importante observar algumas atividades que impreterivelmente fará parte da gestão contínua da GCN:
  - A definição de escopo, papéis e responsabilidades
  - A nomeação de uma pessoa ou equipe responsável pela GCN
  - Manter o programa de GCN atualizado.
  - A promoção da GCN por toda a organização
  - A administração do programa de testes.
  - Manter atualizadas as avaliações de risco e de impacto nos negócios
  - Manter atualizada a documentação do programa de GCN.
  - Monitorar o desempenho da capacidade de continuidade de negócios.
  - Gerência sobre os custos, frente à capacidade de continuidade da organização.

- Estabelecer e monitorar o gerenciamento de mudanças e o regime de sucessão.

- **A documentação da continuidade de negócios.**

  Convém para a manutenção da Gestão da Continuidade de Negócios, não se limitando a esta lista, a criação e atualização contínua da seguinte documentação:
  - A política de GCN
  - A análise de impacto nos negócios
  - A avaliação de riscos e ameaças
  - As estratégias de GCN
  - Programa de Conscientização e Treinamento
  - Plano de gerenciamento de incidentes
  - Planos de continuidade e recuperação de negócios
  - Agenda de testes
  - Contratos e acordos de níveis de serviço

6.2.2. Entendendo a organização

Para a continuidade de negócios, o entendimento da organização é provido por:

- Identificar os objetivos da organização, a obrigação das partes interessadas, deveres legais e o ambiente no qual a organização opera.
- Identificar as atividades, ativos e recursos, internos e externos, que suportam a entrega desses produtos e serviços. É importante a identificação das atividades críticas para a organização, como também a sua categorização quanto a prioridades de recuperação. Também é importante a determinação dos requisitos de continuidade que cada atividade necessitará.
- Avaliar o impacto e as conseqüências sobre o tempo de falhas sobre estas atividades, ativos e recursos. Nesta etapa, é importante definir o tempo máximo de interrupção tolerável de cada atividade, o nível mínimo no qual a atividade deve ser desempenhada após o seu reinício e o tempo máximo até a retomada dos níveis normais de operação.
- Identificar e avaliar as ameaças que possam interromper os produtos e serviços fundamentais e os ativos, atividades e recursos que os suportam.

Com o resultado da análise de impactos e da análise de riscos, cabe a organização decidir quais escolhas ela adotará para cada cenário identificado. Estas estratégias visam: ou reduzir a chance de uma interrupção, ou diminuir o tempo de uma interrupção, ou limitem o impacto de uma interrupção em produtos ou serviços na organização. Dentre as escolhas a serem adotadas, cabem:

- **Continuidade de negócios:** neste caso, serão adotadas ações que visem garantir a continuidade da atividade em caso de indisponibilidade, atendendo aos tempos levantados na análise de impacto;

- **Aceitação:** um risco identificado, pode de toda forma, ser tido como aceitável pela organização. Então, por uma decisão da direção, o mesmo pode ser tido como aceito;
- **Transferência:** ocorre, para alguns casos, que a melhor estratégia é transferi-los, sendo por meio de seguros ou acordos contratuais;
- **Mudar, suspender ou terminar:** em dadas circunstancias, devido a um risco identificado, e o beneficio adquirido com a atividade, a Direção pode decidir por terminar a atividade.

É relevante observar a importância da aprovação da direção da relação de atividades relacionadas, seus riscos, e as estratégias adotadas. Visando garantir que o trabalho realizado reflete verdadeiramente a realidade da organização.

6.2.3. Determinando a estratégia de continuidade de negócios

A seleção da estratégia de contingência mais adequada para cada situação é uma questão complexa e que exige uma análise detalhada que considere os requisitos técnicos e de negócio [Wiboonrat 2008] e [Cegiela 2006]. Convém que a abordagem da organização para determinar suas estratégias de GCN:

- Implemente medidas apropriadas, de forma a reduzir a probabilidade de ocorrência de incidentes e/ou reduzir os potenciais efeitos destes incidentes;
- Mantenha um registro das medidas de resiliência e mitigação;
- Forneça continuidade para as atividades críticas durante e após um incidente; e
- Mantenha um registro das atividades classificadas como não críticas.

Quanto à definição das opções de estratégias, convém que sejam considerados uma série de fatores, dentre os quais: (1) o período máximo de interrupção tolerável da atividade crítica, (2) os custos de implementação de uma ou mais estratégias e (3) as consequências da falta de ação. Convém que sejam elaboradas estratégias de contingência para todos os recursos da organização, o que inclui além dos aspectos tecnológicos, as pessoas, os suprimentos, as instalações, as informações e as demais partes interessadas.

Para organizações que buscam definir, implementar ou validar suas estratégias de gerenciamento de incidentes e gestão de continuidade de negócios é um fator critico de sucesso a interação com as autoridades responsáveis por responder às emergências. Estas autoridades serão fundamentais para a declaração oficial de que ocorreu uma emergência civil, além de fornecer:

- Ajuda pré ou pós-incidente;
- Procedimentos de aviso e informação; e
- Acordos de recuperação comunitária após uma emergência civil.

6.2.4. Desenvolvimento e implementação de uma resposta de GCN

Este elemento do ciclo de vida de GCN é relacionado ao desenvolvimento e implementação dos planos apropriados e dos preparativos realizados, de forma a garantir a continuidade das atividades críticas e o gerenciamento dos incidentes. Durante esta fase, convém que a organização:

- Identifique suas atividades críticas
- Avalie as ameaças a estas atividades críticas
- Escolha estratégias apropriadas que diminuam a probabilidade e os impactos dos incidentes; e
- Escolha estratégias apropriadas que permitam a continuidade ou recuperação de suas atividades críticas.

Quanto à estrutura de resposta a incidentes, convém que a organização defina uma estratégia de resposta a incidentes,com uma determinada estrutura que permita, quando da ocorrência de um incidente:

- Confirmar a natureza e extensão do incidente;
- Tomar controle da situação;
- Controlar o incidente;
- Comunicar-se com as partes interessadas

Quanto ao conteúdo dos planos, convém que todos eles, sejam de gerenciamento de incidentes, continuidade de negócios ou recuperação de negócios, sejam concisos e acessíveis àqueles que possuam responsabilidades definidas nesses planos. Quanto à estruturação dos planos, convém que contenham:

- Objetivo e escopo;
- Papéis e responsabilidades definidas
- Procedimentos de ativação dos planos;
- Detalhes de contato;

Convém que a organização nomeie o principal responsável por cada plano, e identifique e documente os responsáveis pela análise crítica, correção e atualização dos planos em intervalos regulares. Quanto aos tipos de planos, esses podem ser:

- **Plano de gerenciamento de Incidentes (PGI)**

O propósito de um PGI é permitir que a organização gerencie a fase inicial (crítica) de um incidente. Convém que o conteúdo do PGI contenha:

  - Lista de tarefas e ações
  - Contatos de emergência

- Atividade das pessoas
- Comunicação à mídia
- Gestão de partes interessadas
- Localização para o gerenciamento de incidentes
- Anexos relevantes (Plantas, mapas, planos de acesso ao local, etc.)

- **Plano de Continuidade de Negócios (PCN)**

O propósito de um PCN é permitir que a organização recupere ou mantenha suas atividades em caso de uma interrupção das operações normais de negócio. Os PCNs são ativados para dar suporte às atividades críticas necessárias para cumprir os objetivos da organização. Eles podem ser executados integral ou parcialmente e em qualquer etapa da resposta a um incidente. Convém que o conteúdo do PCN contenha:

- Plano de ação / Lista de tarefas
- Recursos necessários
- Responsáveis
- Formulários e anexos

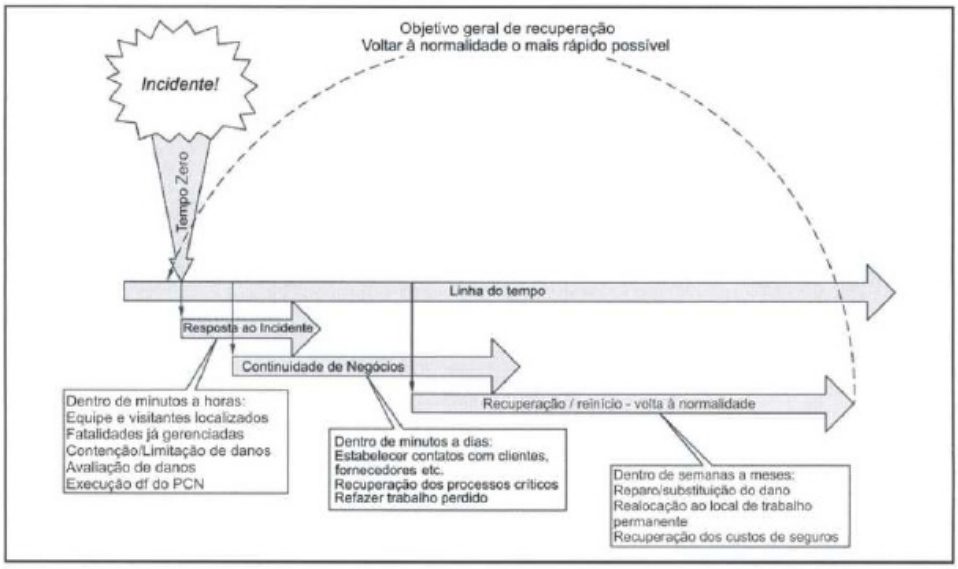


Figura 6.2. Linha do tempo do incidente, e a relação entre a ativação dos planos de gerenciamento de incidentes, continuidade de negócios e recuperação de negócios.

A Figura 6.3 ilustra a sequência em que as ações (planos) são executadas após a ocorrência de um incidente que compromete a continuidade de uma operação e/ou serviço crítico. É importante salientar que devem existir regras bem definidas para a a avaliação do incidente e a ativação dos planos.

6.2.5. Testando, mantendo e analisando criticamente os preparativos de GCN

Os preparativos de continuidade de negócios e de gerenciamento de incidentes da organização não podem ser considerados confiáveis até serem testados e apenas se estiverem atualizados. Portanto, convém que os preparativos sejam verificados por meio de testes, auditoria e processos de auto-avaliação, de forma a garantir que estejam adequados. Para garantir a obtenção destes objetivos, convém que:

- Seja instituído um programa de testes, partindo dos testes de mesa, até testes completos da solução de continuidade de negócios;
- Seja instituído um programa de manutenção do GCN, visando garantir que, quaisquer mudanças, internas ou externas, que causem um impacto à organização, sejam analisadas criticamente quanto a GCN;
- A alta direção, nos intervalos que considerar apropriados, analise criticamente a capacidade de GCN da organização, de forma a garantir sua aplicabilidade, adequação e funcionalidade;
- A organização providencie uma auditoria independente para avaliar a sua competência de GCN e a sua capacidade de identificar falhas reais e potenciais;
- Um processo de auto-avaliação seja instituído objetivando garantir que a organização tenha competência e capacidade de GCN sólidas, eficazes e adequadas.

A Tabela 6.1 representa os métodos de testes aplicáveis a fim de avaliar e identificar oportunidades de melhorias das estratégias de contingência.

Tabela 6.1. Tipos e métodos de teste de estratégias de GCN.

Complexidade	Teste	Processo	Variações	Freqüência recomendada [a]
Simples	Testes-de-mesa	Análise crítica/correção	Atualização/Validação	Ao menos anualmente
		Questionar conteúdo do PCN	Auditoria/Verificação	Anualmente
	"Walk-through" (repassar os passos) do plano	Questionar o conteúdo do PCN	Incluir interação e validar papéis dos participantes	Anualmente
Médio	Simulação	Usar situação "artificial" para validar se os PCN possuem as informações necessárias e suficientes, de forma a permitir uma recuperação com sucesso	Incorporar planos associados	Anualmente ou duas vezes ao ano
	Testar atividades criticas	Execução em ambiente controlado que não prejudique o andamento normal dos negócios	Executar algumas operações a partir de um local alternativo por um tempo determinado	Anualmente ou menos
Complexo	Testar todo o PCN, incluindo o gerenciamento de incidentes	Teste que envolve todo o prédio/campus/zona de exclusão		Anualmente

[a] Convém que a freqüência dos testes dependa das necessidades da organização, do ambiente no qual ela opera e das necessidades das partes interessadas. Porém, convém que o programa de testes seja flexível, levando em conta a freqüência de ocorrência de mudanças na organização e o resultado dos testes anteriores. Os métodos de teste acima podem ser empregados para cada componente de um plano ou para um ou mais planos.

6.2.6. Incluindo a GCN na cultura da organização

Para obter sucesso, a continuidade de negócios precisa se tornar parte da gestão da organização, independente de seu tamanho ou setor. O desenvolvimento, promoção e incorporação da cultura de GCN na organização garantem que a GCN se tornará parte dos valores básicos e da gestão da organização.

Convém que a organização possua um processo para identificar e implementar os requisitos de treinamento de GCN e para avaliar a eficácia desta implementação.

6.2.6.1 Conscientização

Convém que a organização crie, aumente e mantenha uma consciência por meio da educação permanente em GCN e de um programa de informações para toda a equipe. Este programa deve incluir:

- Um processo de consulta junto a toda equipe sobre a implementação do programa de GCN;
- Discussão de GCN nos informativos, apresentações, programas ou relatórios diários da organização;
- Inclusão da GCN nas paginas pertinentes da web ou da intranet;
- Aprendizado por meio de incidentes internos e externos;

- GCN como um tópico nas reuniões de equipe;
- Testes de planos de continuidade em locais alternativos, por exemplo, um local de recuperação; e
- Visita a esses locais alternativos.

A organização deve estender seu programa de conscientização de GCN para seus fornecedores e outras partes interessadas.

6.2.6.2 Treinamento

Convém que a organização treine a equipe de GCN para tarefas como:

- Gestão do programa de GCN
- Execução de uma análise de impacto nos negócios
- Desenvolvimento e implementação de PCN
- Execução de um programa de testes de PCN
- Avaliação de riscos e ameaças
- Comunicação com a mídia

Além da equipe de GCN o pessoal não relacionado diretamente a GCN, mas que tenha algum papel definido no processo de GCN também deve ser treinado, pois isso pode representar o sucesso ou fracasso no momento da execução dos planos [Wei 2009].

6.3. Boas Práticas

Essa seção apresenta um conjunto de boas práticas relacionadas com a gestão da continuidade de negócios e de maneira mais especifica com as estratégias de contingência de TI, também denominados de planos de recuperação de desastres.

6.3.1. Disaster Recovery International Institute (DRII)

As práticas profissionais recomendadas pelo DRII – Disaster Recovery International Institute, ilustradas da Figura 6.4, para atuação em Gestão de Continuidade de negócios são distribuídas em dez aspectos [DRII 2010]:

Figura 6.4. As 10 Práticas Profissionais do Disaster Recovery Internacional Institute (DRII)

Inicio e Gestão do Programa

São definidos os requisitos de continuidade, obtidos apoio da alta direção quanto ao programa e definição de papéis e responsabilidades.

Avaliação de riscos e controles

Nesta etapa, são identificados os riscos levantados junto às pessoas, instalações e tecnologias do escopo, identificação de perdas potenciais e definição de controles a serem aplicados.

Análise de Impacto nos Negócios (AIN / BIA)

Identificação dos impactos resultantes de interrupções de negócio, e técnicas que podem ser usadas para quantificar e qualificar esses impactos. Definição também de tempos críticos, prioridades de recuperação e interdependências.

Estratégias de continuidade de negócios

Apoiado pelos resultados da AIN/BIA e da análise de riscos e controles, recomendar estratégias de continuidade de negócios.

Preparação e Resposta a emergência

Preparar um estado de prontidão para a organização para responder a uma emergência de forma coordenada e eficaz.

Planos de continuidade de negócios

Projetar, desenvolver e implementar Planos de Continuidade de Negócios.

Programas de sensibilização e formação

Preparar um programa para criar a consciência referente à GCN.

Exercício, auditoria e Manutenção dos Planos de Continuidade de Negócios

Estabelece o plano de exercícios e testes dos PCN's, e estabelece também os procedimentos de auditoria do programa e planos de continuidade de negócios.

Comunicação de Crises

Desenvolve os planos de ação para comunicação com as partes interessadas para garantir a clareza das informações na comunicação das crises.

Coordenação com Agências Externas

Estabelecer procedimentos e políticas para a coordenação e continuidade das atividades de restauração com agências externas.

6.3.2. Business Continuity Institute (BCI)

A Figura 6.5 ilustra o ciclo de vida da gestão de continuidade de negócios segundo o *Business Continuity Institute* [BCI 2010].

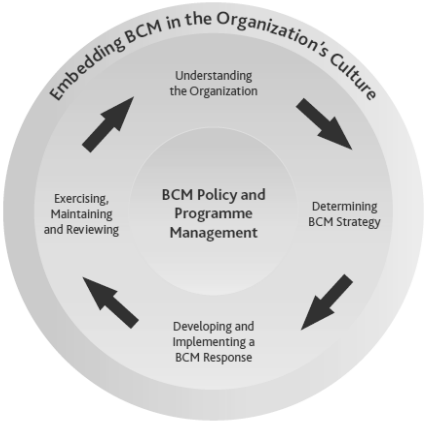


Figura 6.5. Ciclo de Vida da Gestão de Continuidade de Negócios

Gestão da Política e do Programa

A política de GCN é o documento chave que define o escopo e a governança do programa de GCN, e reflete os motivos pelos quais a GCN está sendo implementada. Ela fornece o contexto em que os recursos solicitados serão implementados, e identifica os princípios aos quais a organização aspira e contra os quais seu desempenho pode ser auditado.

Incorporando a GCN na Cultura da Organização

A criação bem sucedida da cultura de GCN da organização depende da sua integração com o planejamento estratégico da organização, bem como o seu alinhamento com as prioridades de negócios.

Entendendo a Organização

Prática profissional dentro do Ciclo de Vida da GCN que analisa a organização em termos de quais seus objetivos, como estrutura funcional e os obstáculos do ambiente em que opera. As informações coletadas tornam possível determinar a melhor forma de preparar uma organização para ser capaz de gerenciar as suas interrupções.

Determinando a Estratégia de Continuidade de Negócios

Prática profissional dentro do ciclo de vida do BCM que determina quais as estratégias que vão ao encontro da política de GCN e exigências organizacionais e seleciona respostas tácticas dentre as opções disponíveis.

Desenvolvimento e Implementando uma Resposta de GCN

Essa é a prática profissional que implementa estratégias de acordo com o processo de desenvolvimento de um conjunto de planos de continuidade de negócios.

Exercitando, mantendo e revisando a GCN

"Exercitando, mantendo e revisando a GCN" é a prática profissional no âmbito do Ciclo de Vida da GCN, que visa assegurar que a melhoria contínua é alcançada através das ações em curso. As atividades realizadas nesta seção serão apoiadas pela política de BCM.

6.3.3. Gerenciamento dos Serviços de TI

A norma denominada ISO 20000 [ABNT 2008c] descreve, entre outros aspectos, a importância do gerenciamento da continuidade e da disponibilidade dos serviços de TI que oferecem suporte ao negócio. Por ser focada em tecnologia, essa norma agrega conceitos práticos refere a TI que devem ser considerados ao elaborar os planos de recuperação de desastres, tais como:

- Avaliação do acordo de nível de serviço quando da definição dos Planos de continuidade;
- A importância de uma série de testes dos planos, após grandes mudanças no ambiente de TI;
- A análise de impacto que qualquer mudança no ambiente de TI pode acarretar na disponibilidade dos serviços prestados;

A Figura 6.6 ilustra a estrutura do processo de gerenciamento dos serviços de TI. A norma em questão destaca que eventos inesperados que tenham impactado na disponibilidade dos serviços devem ser investigados, e ações adequadas devem ser tomadas. Com isto, busca-se a excelência operacional de serviços, mantendo-os disponíveis aos clientes com a qualidade requerida.

A norma ISO 20000 ressalta que os planos de continuidade de serviço, lista de contatos e a base de dados de gerenciamento devem estar disponíveis quando da ocorrência de uma indisponibilidade para que os planos de ação possam ser colocados em execução.

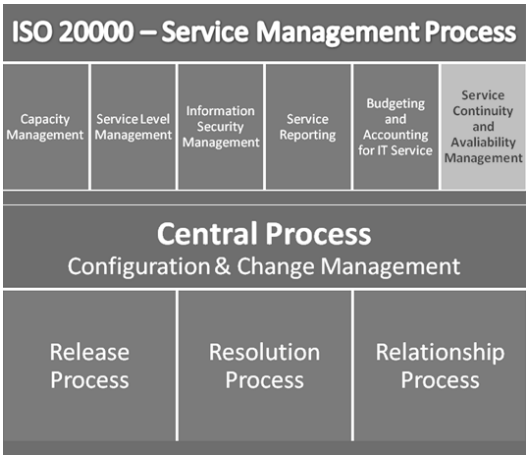


Figura 6.6. Processos e serviços representados pela ISO 20000.

6.3.4. Código de Prática para a Gestão da Segurança da Informação

A norma denominada ABNT NBR ISO/IEC 27002, cujo objetivo é estabelecer diretrizes e princípios para iniciar, implementar, manter e melhorar a gestão da segurança da informação em uma organização define o seguinte objetivo de controle no que tange a gestão da continuidade do negócio: não permitir a interrupção das atividades do negócio e proteger os processos críticos contra defeitos de falhas ou desastres significativos, e assegurar a sua retomada em tempo hábil, se for o caso [ABNT 2005].

Para tal, são definidos controles para (1) incluir a segurança da informação no processo de gestão da continuidade de negócio, (2) identificar eventos de risco que possam causar interrupções aos processos de negócio, (3) desenvolver e implementar planos de continuidade relativos à segurança da informação, (4) garantir a consistência dos planos e (5) para testar e analisar criticamente os planos de continuidade do negócio.

6.3.5. COBIT

O *Control Objectives for Information and related Technology* (COBIT) é um conjunto de boas práticas para o gerenciamento da tecnologia da informação criado pela *Information Systems Audit and Control Association* (ISACA) e pelo IT Governance Institute (ITGI) em 1996 [IT Governance Institute 2008a].

O COBIT está organizado em quatro domínios, sendo que conforme a Figura 6.7 um desses domínios possui objetivos de controle voltados para assegurar a continuidade dos serviços.

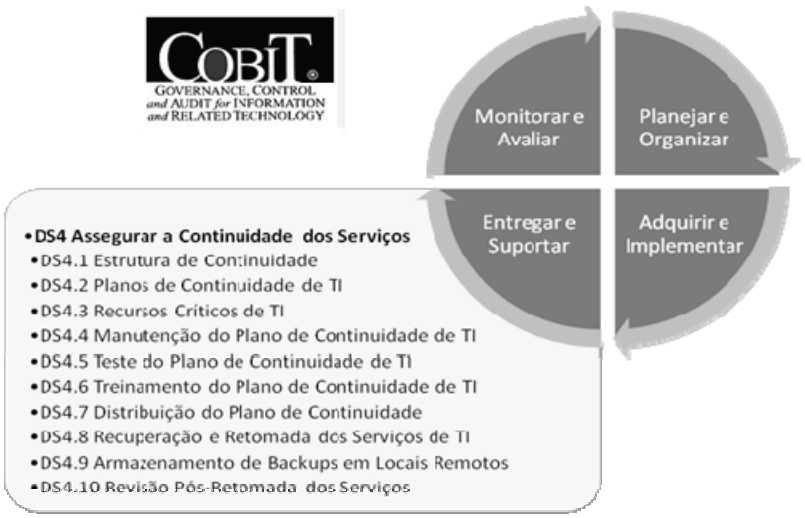


Figura 1.7. COBIT – Controles e Objetivos em Tecnologia da Informação

A seguir a descrição de cada um dos objetivos de controle que possuem relação com a continuidade dos serviços de TI.

Estrutura de Continuidade

Desenvolver um modelo para continuidade de TI a fim de apoiar o gerenciamento da continuidade do negócio de toda a empresa através de um processo consistente orientado a estrutura organizacional quanto ao gerenciamento da continuidade, contemplando papéis, tarefas e responsabilidades dos provedores de serviço internos e externos, seus gerenciamentos, clientes e as regras e estruturas para documentar, testar e executar planos de recuperação de desastres e continuidade de TI

Planos de Continuidade de TI

Desenvolver planos de continuidade de TI com base na estrutura e projetados para reduzir o impacto de uma grande interrupção de funções e processos de negócio fundamentais.

Recursos Críticos de TI

Dar atenção especial aos itens mais críticos no plano de continuidade de TI para assegurar a capacidade de restabelecimento e definir prioridades em situações de recuperação. Prevenir o desvio de atenção para os itens de recuperação menos críticos e assegurar resposta e recuperação em alinhamento com as necessidades de negócio de maior importância; ao mesmo tempo, assegurar que os custos sejam mantidos em um nível aceitável e em conformidade com os requisitos contratuais e regulamentares.

Manutenção do Plano de Continuidade de TI

Encorajar o gerenciamento de TI a definir e executar procedimentos de controle de mudança para assegurar que o plano de continuidade de TI seja mantido atualizado e reflita sempre os requisitos de negócios atuais.

Teste do Plano de Continuidade de TI

Testar o plano de continuidade de TI regularmente para assegurar que os sistemas de TI possam ser efetivamente recuperados, que desvios sejam tratados e que o plano se mantenha relevante. Para tanto, são necessários preparação cuidadosa, documentação, registro dos resultados dos testes e implementação de planos de ação de acordo com os resultados.

Treinamento do Plano de Continuidade de TI

Assegurar que todas as partes envolvidas recebam treinamento regular sobre os procedimentos, papéis e respectivas responsabilidades no caso de um incidente ou desastre. Verificar e intensificar o treinamento de acordo com os resultados dos teste de continuidade.

Distribuição do Plano de Continuidade

Definir e gerenciar uma estratégia de distribuição para assegurar que os planos sejam seguramente distribuídos e que estejam apropriadamente disponíveis às partes interessadas e autorizados quando e onde necessário. Toda atenção deve ser dispensada para tornar o plano acessível em todos os cenários de desastre.

Recuperação e Retomada dos Serviços de TI

Planejar as ações a serem executadas nos momentos de recuperação e retomada dos serviços de TI. Isto pode incluir ativação de backup sites, iniciação de processamento alternativo, comunicação para as partes interessadas e os clientes, procedimentos de retorno à produção etc. Assegurar que o negócio entenda o tempo de recuperação de TI e os investimentos tecnológicos necessários para sustentar as necessidades de recuperação e retorno à produção.

Armazenamento de Backups em Locais Remotos

Armazenar remotamente todas as mídias de cópias de segurança críticas, documentação e outros recursos de TI necessários para a recuperação da TI e os planos de continuidade de negócio. O conteúdo armazenado nas cópias de segurança precisa ser determinado em colaboração entre os proprietários dos processos de negócio e o pessoal de TI. Assegurar a compatibilidade de hardware e software para restaurar os dados arquivados e testar e atualizar periodicamente os dados arquivados

Revisão Pós-Retomada dos Serviços

Após a retomada bem-sucedida da função de TI depois de um desastre, determinar se o gerenciamento de TI tem procedimentos para avaliar a adequação do plano atual e realizar sua atualização, se necessário.

Importante observar que o COBIT, além das recomendações alinhadas com as demais boas práticas, apresenta recursos de governança até então não adotadas pelas demais. Dentre elas, podemos citar a proposta de matriz de responsabilidade (Figura 6.8) incluída na versão 4.1 do COBIT.

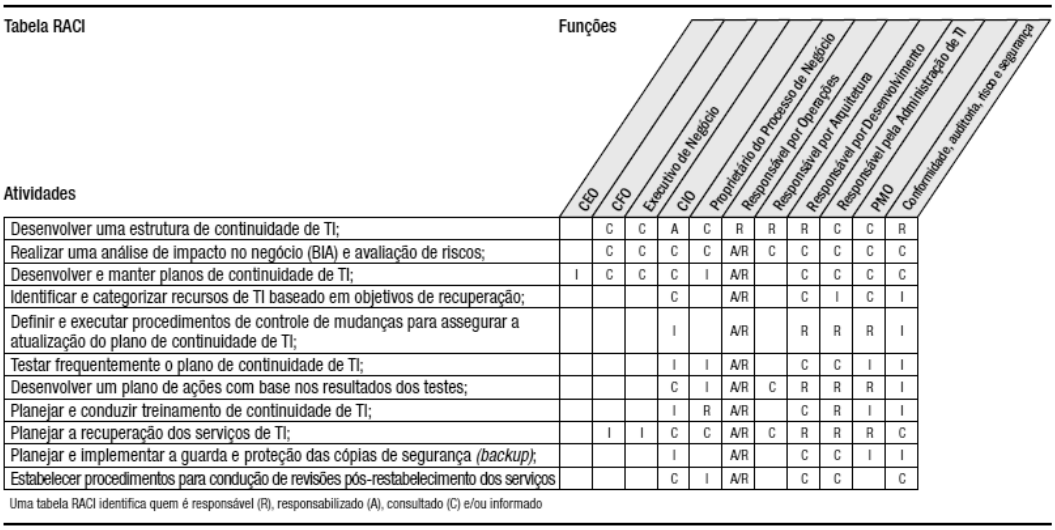


Tabela RACI — Atividades	CEO	CFO	Executivo de Negócio	CIO	Proprietário do Processo de Negócio	Responsável por Operações	Responsável por Arquiteturа	Responsável por Desenvolvimento	Responsável pela Administração de TI	PMO	Conformidade, auditoria, risco e segurança
Desenvolver uma estrutura de continuidade de TI;		C	C	A	C	R	R	R	C	C	R
Realizar uma análise de impacto no negócio (BIA) e avaliação de riscos;		C	C	C	C	A/R	C	C	C	C	C
Desenvolver e manter planos de continuidade de TI;	I	C	C	C	I	A/R		C	C	C	C
Identificar e categorizar recursos de TI baseado em objetivos de recuperação;				C		A/R		C	I	C	I
Definir e executar procedimentos de controle de mudanças para assegurar a atualização do plano de continuidade de TI;				I		A/R		R	R	R	I
Testar frequentemente o plano de continuidade de TI;				I	I	A/R		C	C	I	I
Desenvolver um plano de ações com base nos resultados dos testes;				C	I	A/R	C	R	R	R	I
Planejar e conduzir treinamento de continuidade de TI;				I	R	A/R		C	R	I	I
Planejar a recuperação dos serviços de TI;		I	I	C	C	A/R	C	R	R	R	C
Planejar e implementar a guarda e proteção das cópias de segurança *(backup)*;				I		A/R		C	C	I	I
Estabelecer procedimentos para condução de revisões pós-restabelecimento dos serviços				C	I	A/R		C	C		C

Uma tabela RACI identifica quem é responsável (R), responsabilizado (A), consultado (C) e/ou informado

Figura 6.8. Matriz de Responsabilidade (RACI)

Outra abordagem trazida pelo COBIT é apresentação de um modelo de maturidade, em que o processo de continuidade de serviço de TI pode ser comparado e avaliado, conforme os níveis de maturidade descritos na Tabela 6.2.

Tabela 6.2. Modelo de Maturidade para Continuidade de serviços conforme o COBIT.

Nível	Descrição
Inexistente	Não há entendimento dos riscos, vulnerabilidades e ameaças às operações de TI ou do impacto da perda dos serviços de TI nos negócios. Não é considerado que a continuidade dos serviços deve ter atenção da Direção.
Inicial /*Ad hoc*	As responsabilidades pela continuidade dos serviços são informais e a autoridade para exercer essas responsabilidades é limitada. O gerenciamento está se tornando consciente dos riscos relacionados e da necessidade da continuidade dos serviços. O foco da Direção quanto à continuidade dos serviços está relacionado aos recursos de infra-estrutura e não aos serviços de TI. Os usuários implementam paliativos em resposta a interrupções nos serviços. A resposta da TI para a maioria das interrupções é reativa e despreparada. Paralisações dos sistemas são agendadas para atender às necessidades da TI, porém não consideram os requisitos do negócio.
Repetível, porém Intuitivo	A responsabilidade de assegurar a continuidade do serviço é estabelecida. As abordagens para assegurar a continuidade do serviço são fragmentadas. Relatórios de disponibilidade de sistema são esporádicos, podem ser incompletos e não levam em consideração o impacto nos negócios. Não existe um plano de continuidade de TI documentado, embora haja comprometimento da continuidade da disponibilidade de serviços e seus maiores princípios sejam conhecidos.

	Existe um inventário de sistemas e componentes críticos, mas ele pode não ser confiável. Práticas de serviços contínuos estão surgindo, contudo o sucesso depende das pessoas.
Processo Definido	A responsabilidade solidária pelo gerenciamento da continuidade dos serviços está clara. A responsabilidade pelo planejamento e pelos testes da continuidade dos serviços é claramente definida e atribuída. O plano de continuidade de TI é documentado e baseia-se na importância do sistema e no impacto nos negócios. Há relatos periódicos dos testes de continuidade de serviços. As pessoas tomam a iniciativa de seguir padrões e recebem treinamento para lidar com a maioria dos incidentes ou desastres. A Direção comunica consistentemente a necessidade do plano de assegurar a continuidade de serviço. Componentes de alta disponibilidade e redundância de sistema estão sendo aplicados. É mantido um inventário sobre os componentes e sistemas críticos.
Gerenciado e Mensurável	As responsabilidades e os padrões para a continuidade dos serviços são impostos. A responsabilidade por manter o plano de continuidade de serviço é atribuída. As atividades de manutenção são baseadas nos testes de continuidade de serviço, em boas práticas internas, e na mudança do ambiente de negócio e de TI. Dados estruturados sobre a continuidade dos serviços estão sendo coletados, analisados, relatados e gerando ações. É dado treinamento obrigatório e formal sobre os processos de continuidade de serviço. Boas práticas de disponibilidade de sistemas estão sendo consistentemente implementadas. As práticas de disponibilidade e planejamento de continuidade de serviços influenciam um ao outro. Os incidentes de descontinuidade são classificados e os procedimentos de encaminhamento de cada incidente é bem conhecido por todos os envolvidos. Objetivos e métricas de continuidade dos serviços foram desenvolvidos e acordados, mas podem ser inconsistentemente medidos.
Otimizado	Processos integrados de continuidade de serviços consideram a comparação com o mercado *(benchmarking)* e as melhores práticas externas. O plano de continuidade de TI é integrado ao plano de continuidade de negócio e é rotineiramente mantido. A necessidade de assegurar a continuidade de serviços é garantida pelos fornecedores e principais prestadores de serviço. Ocorrem testes formais do plano de continuidade de TI, e seus resultados são a base da atualização do plano. Coleta e análise dos dados são utilizados para melhoria contínua do processo. O planejamento de continuidade de serviço e as práticas de disponibilidade

	estão completamente alinhados. A Direção assegura que um desastre ou incidente importante não ocorrerá devido a um único ponto de falha. Práticas de encaminhamento são entendidas e rigorosamente impostas. Os objetivos e métricas sobre o alcance da continuidade de serviços são mensurados de forma sistemática. A Direção ajusta o planejamento à continuidade do serviço em resposta às medições

6.3.6. Alinhamento entre Boas Práticas

As organizações adotam diferentes modelos, padrões e normas para orientar o seu processo de gestão de segurança e de tecnologia da informação. As boas práticas discutidas na seção anterior representam o que convém que seja implementado (COBIT e ISO 27002) e os processos que oferecem suporte e orientação para a definição de como pode se dar a aplicação dos objetivos de controles.

Tabela 6.3. Consolidação das boas práticas na Gestão de Continuidade de Negócios.

CobiT 4.1	ITIL V3	ISO/IEC 27002:2005
DS4.1 IT continuity framework	SD 4.5 IT service continuity management	6.1.6 Contact with authorities
	SD 4.5.5.1 Stage 1—Initiation	6.1.7 Contact with special interest groups
	CSI 5.6.3 IT Service continuity management	14.1.1 Including information security in the business continuity management process
		14.1.2 Business continuity and risk assessment
		14.1.4 Business continuity planning framework
DS4.2 IT continuity plans	SD 4.5.5.2 Stage 2—Requirements and strategy	6.1.6 Contact with authorities
	SD 4.5.5.3 Stage 3—Implementation	6.1.7 Contact with special interest groups
	SD App K The typical contents of a recovery plan	14.1.3 Developing and implementing continuity plans including information security
DS4.3 Critical IT resources	SD 4.4.5.2 The proactive activities of availability management	14.1.1 Including information security in the business continuity management process
	SD 4.5.5.4 Stage 4—Ongoing operation	14.1.2 Business continuity and risk assessment
DS4.4 Maintenance of the IT continuity plan	SD 4.5.5.4 Stage 4—Ongoing operation	14.1.5 Testing, maintaining and reassessing business continuity plans
DS4.5 Testing of the IT continuity plan	SD 4.5.5.3 Stage 3—Implementation	14.1.5 Testing, maintaining and reassessing business continuity plans
	SD 4.5.5.4 Stage 4—Ongoing operation	
DS4.6 IT continuity plan training	SD 4.5.5.3 Stage 3—Implementation	14.1.5 Testing, maintaining and reassessing business continuity plans
	SD 4.5.5.4 Stage 4—Ongoing operation	

DS4.7 Distribution of the IT continuity plan	SD 4.5.5.3 Stage 3— Implementation	14.1.5 Testing, maintaining and reassessing business continuity plans
	SD 4.5.5.4 Stage 4—Ongoing operation	
DS4.8 IT services recovery and resumption	SD 4.4.5.2 The proactive activities of availability management	14.1.1 Including information security in the business continuity management process
	SD 4.5.5.4 Stage 4—Ongoing operation	14.1.3 Maintain or restore operations and ensure availability of information
DS4.9 Offsite backup storage	SD 4.5.5.2 Stage 2— Requirements and strategy	10.5.1 Information backup
	SO 5.2.3 Backup and restore	
DS4.10 Post-resumption review	SD 4.5.5.3 Stage 3— Implementation	14.1.5 Testing, maintaining and reassessing business continuity plans
	SD 4.5.5.4 Stage 4— Ongoing operation	

A Tabela 6.3 representa o mapeamento entre os objetivos de controle do COBIT, os processos do ITIL e os controles previstos no Código de Prática para Gestão da Segurança da Informação [IT Governance Institute 2008b]. Através do alinhamento entre as boas práticas é possível realizar tanto a análise de aderência das práticas implementadas em cada organização, como aprofundar o entendimentos dos controles e processos referentes à continuidade dos serviços de TI.

6.4. Estudo de Caso

O estudo de caso proposto demonstra a aplicação dos conceitos abordados nas seções anteriores na produção de planos para companhia aérea Voe Sempre. As fases a serem descritas são: (a) entendendo a organização, (b) determinando a estratégia de Continuidade de Negócios, (c) desenvolvendo e implementando uma resposta de GCN e (d) testando, Mantendo e Analisando Criticamente os preparativos de GCN.

A organização fictícia deste estudo situa-se no segmento aéreo tendo como diferencial das demais sua utilização de tecnologia da informação voltada à redução de custos e alta confiabilidade.

6.4.1. Entendendo a organização

A organização conta com dois datacenters de Tier 2[1] conforme ANSI/TIA-942, o primeiro de construção própria e o segundo locado através de modalidade *colocation* com uma empresa especializada. Alguns sistemas já possuem contingências, porém não existe uma análise formal dos processos críticos para a organização, sendo assim também não sabemos se as contingências existentes são suficientes e se os sistemas que não tem contingência deveriam ter.

Então, o primeiro passo a ser executado é uma análise de impacto no negócio através de um formulário que terá como produto final os processos críticos da

[1] Datacenter com componentes redundantes.

organização, os RTOs e RPOs exigidos para os sistemas críticos e uma visão ampla sobre MTPD e impactos financeiros.

6.4.1.1. Análise de Impacto no Negócio

Conforme definido anteriormente a Análise de Impacto no Negócio é o processo que envolve a análise das funções de negócio e os efeitos que uma interrupção possa causar nelas. Neste ponto é essencial a participação de todas as áreas de negócios da empresa, pois os dados e informações aqui coletadas trarão a luz todos os requisitos de organização para montagem de estratégias de recuperação de desastres em tecnologia da informação.

Ao fim desta etapa, a área de Tecnologia da Informação poderá entender quais as premissas deverá utilizar para montagem da infraestrutura, contratação de serviços, definição de acordos de níveis de serviço, impacto financeiro por processo de negócio e aplicação e, de priorizações entre sistemas para recuperação após desastres.

O envolvimento principal da área de Tecnologia da Informação neste momento é fornecer uma lista de sistemas utilizados pelo negócio que seja de fácil entendimento e compreensão pelos usuários de negócio. Neste sentido é necessário compreender que as áreas de negócio podem enxergar um conjunto de aplicações como um sistema único, requerendo que a área de Tecnologia da Informação trate este conjunto como uma entidade inseparável para o planejamento de planos de recuperação de desastres em tecnologia da informação.

Para o entendimento da organização, deve-se organizar uma análise de impacto do negócio, atividade na qual são realizadas entrevistas com as pessoas chaves de cada processo de negócio visando identificar a criticidade de cada processo de negócio e sua relação com sistemas da informação.

Um trabalho prévio deve ser realizado na análise de impacto do negócio buscando preparar dados iniciais requeridos durante todo trabalho:

- Identificação de todos os processos de negócio;
- Identificação de todos os sistemas da informação da companhia;
- Estabelecer critérios para cada nível de criticidade (alto, médio e baixo);
- Identificar as pessoas que serão entrevistadas.

Após a identificação de todos os processos de negócio, sistemas, o estabelecimento de critérios e identificação das pessoas que serão entrevistadas, então é organizado uma agenda de entrevistas. No formulário proposto para a análise de impacto no negócio, foram inseridas apenas as informações relevantes para a criação de estratégias e planos de recuperação de desastres de TI, porém cabe registrar que outras informações sobre o negócio que auxiliem na construção de estratégias e planos de continuidade de operacional, planos de resposta a emergências e planos de gerenciamento de crises podem ser incluídas.

Para este estudo de caso se utilizou o formulário da Tabela 6.4. Vale destacar que a variável de criticidade está vinculada diretamente com o RTO e o MTPD, que são

dependentes dos impactos legais, financeiros e de imagem em caso de indisponibilidade do processo de negócio em questão.

Tabela 6.4. Formulário de Análise de Impacto do Negócio.

PROCESSO DE NEGÓCIO:			
RTO:			
RPO:			
PERÍODO CRÍTICO:			
MTPD:			
CRITICIDADE:	□ Alto	□ Médio	□ Baixo
SISTEMAS DE INFORMAÇÃO:			

Para o desenvolvimento da análise de impacto dos negócios, aconselha-se a utilizar um roteiro de entrevista informal, de forma a poder entender o processo de negócio e o entrevistado entender o significado daquela atividade. Abaixo segue o roteiro utilizado para a companhia aérea em estudo:

(1) O processo de negócio em questão é mais crítico em qual período do mês? Por quê?
(2) Quais sistemas você utiliza para executar as atividades deste processo de negócio?
(3) Se o sistema não está disponível existe alguma atividade alternativa que você realiza?
(4) Quanto tempo é necessário para executar esta atividade alternativa sem o sistema estar disponível?
(5) Os dados no sistema precisam estar sempre atualizados e disponíveis para consulta? Se os mesmos estivessem alguns dias atrasados causariam algum problema operacional?
(6) Qual o período máximo de atraso dos dados aceitável para a execução das atividades críticas desse processo de negócio?
(7) Na área de Tecnologia da Informação temos um segundo Datacenter, o mesmo provê 50% da capacidade para este processo de negócio. Logo, se você tiver que trabalhar usando um sistema com a metade da velocidade do atual, quanto tempo você conseguiria trabalhar assim?

(8) Em caso de indisponibilidade desse processo de negócio existe algum impacto legal, tais como: multas, advertências ou outro tipo de sanção pelo órgão regulador?
(9) Com o processo de negócio indisponível o impacto recairia sobre os clientes?
(10) Qual o percentual de receita direta que esse processo de negócio gera para a organização?

Com base nas respostas para as questões supracitadas se obtêm as informações para o modelo representado na Tabela 6.5.

Tabela 6.5. Modelo de Informações Gerais de um Plano de Continuidade.

PROCESSO DE NEGÓCIO:	**[Nome do Processo de Negócio]**
RTO:	**[Resposta3 e Resposta 4]**
RPO:	**[Resposta 5 e Resposta 6]**
PERÍODO CRÍTICO:	**[Resposta 1]**
MTPD:	**[Resposta 7]**
CRITICIDADE:	**[Resposta 8, Resposta 9, Resposta 10 e uma análise sobre o RTO e MTPD]**
SISTEMAS DE INFORMAÇÃO:	**[Resposta 2]**

Para a organização deste estudo de caso, a Tabela 6.6 apresenta os resultados obtidos através da análise de impacto de negócio:

Tabela 6.6. Resultado de Análise de Impacto do Negócio.

PROCESSO DE NEGÓCIO:	Vendas		
RTO:	1h		
RPO:	0h		
PERÍODO CRÍTICO:	24h durante os 7 dias da semana		
MTPD:	3h		
CRITICIDADE:	■ Alto	□ Médio	□ Baixo
SISTEMAS DE INFORMAÇÃO:	ERP – Módulo de Vendas ERP – Módulo de Relatórios		

		Telecomunicações (WAN) Navegação Web	
PROCESSO DE NEGÓCIO:		Check-In	
RTO:		30min	
RPO:		0h	
PERÍODO CRÍTICO:		24h durante os 7 dias da semana	
MTPD:		1h	
CRITICIDADE:	■ Alto	□ Médio	□ Baixo
SISTEMAS DE INFORMAÇÃO:		ERP – Módulo de Check-In ERP – Módulo de Vendas ERP – Módulo de Relatórios Telecomunicações (WAN) Navegação Web	
PROCESSO DE NEGÓCIO:		Back Office (Administrativo, Pessoal, Contabilidade, etc..)	
RTO:		72 h	
RPO:		Encerramento do Mês	
PERÍODO CRÍTICO:		Todo 5º. útil de cada mês	
MTPD:		3 meses	
CRITICIDADE:	□ Alto	■ Médio	□ Baixo
SISTEMAS DE INFORMAÇÃO:		ERP – Módulo Administrativo Sistema de Contabilidade e Tributos Sistema de Talentos Telecomunicações (LAN) Navegação Web E-mail	
PROCESSO DE NEGÓCIO:		Manutenção e Compras	
RTO:		720h	
RPO:		Encerramento do Mês	
PERÍODO CRÍTICO:		Dias 15 e 30 de cada mês.	
MTPD:		6 meses	
CRITICIDADE:	□ Alto	□ Médio	■ Baixo

SISTEMAS DE INFORMAÇÃO:		Sistema de Manutenção ERP – Módulo de Compras Telecomunicações (LAN) E-mail	
PROCESSO DE NEGÓCIO:		Call Center	
RTO:		1h	
RPO:		Última Transação Efetuada	
PERÍODO CRÍTICO:		8h às 20h de segunda a sábado	
MTPD:		1 dia	
CRITICIDADE:	■ Alto	□ Médio	□ Baixo
SISTEMAS DE INFORMAÇÃO:		Sistema de Atendimento Telefonia Telecomunicações (LAN) E-mail	

Um resumo do resultado da análise de impacto de negócio pode ser observado na Tabela 6.7:

Tabela 6.7. Resumo do Resultado da Análise de Impacto de Negócio.

Processo de Negócio	RTO	RPO	MTPD	Criticidade
Vendas	1h	0h	3h	Alta
Call Center	1h	0h	24h	Alta
Check-in	3h	24h	48h	Média
Back Office	72h	744h	2232h	Média
Manutenção e Compras	720h	744h	4464h	Baixa

6.4.2 Determinando a estratégia de Continuidade de Negócios

Agora a organização já conhece seus processos de negócio críticos, ou seja, com alta criticidade e de baixo RTO. Nessa etapa o mais importante é reunir a equipe de infraestrutura de TI e pensar nas estratégias para recuperação de desastres de TI.

Cada estratégia pensada deve ser precificada, além de observar a viabilidade técnica e esforço de implementação. Uma lista para avaliar o custo benefício deve ser entregue a alta administração para a escolha da estratégia a ser implementada.

6.4.2.1 Tipos de Estratégias

As estratégias de recuperação de desastres serão definidas dentro de três categorias:

- ***Hot site*:** os aplicativos podem ser balanceados e trabalhar com servidores ativos nos dois datacenters, ou seja, em caso de indisponibilidade do datacenter principal o usuário do sistema não percebe a queda;
- ***Warm site*:** os aplicativos trabalham com um dos dois datacenter estiver em modo de espera e são necessárias algumas configurações;
- ***Cold site*:** para restaurar os aplicativos é necessário reinstalar todo o sistema, pois no datacenter secundário existe apenas a infraestrutura de comunicação.

A área de Tecnologia da Informação deverá analisar os resultados obtidos e comparar com a situação atual dos seguintes aspectos:

- **Infraestrutura Tecnológica:**
  - Capacidade de o serviço funcionar em dois sites simultaneamente;
  - Tempo para ativação da contingência para serviços que não forem ativos nos dois *datacenters*;
  - Capacidade de atender o RPO com a estrutura de backups e replicações;
  - Custo para atender os RTOs e RPOs solicitados;
- **Contratos com fornecedores**
  - Validar a possibilidade de fazer atualizações dos sistemas de forma parcial, ou seja, por datacenter;
  - Realizar contratos para priorização de entrega de insumos em momentos de crise;
- **Acordos de Nível de Serviço**
  - Verificar o tempo de atendimento em caso de indisponibilidades;
  - Verificar a capacidade de processamento necessária no ambiente de contingência;

Nesse estudo de caso, os resultados demonstraram que havia aderência parcial dos aspectos analisados, requerendo que acordos de níveis de serviço tivessem sido revistos para adequação, devido a escalabilidade da execução dos processos de negócios relacionados a vendas e check-in. Os serviços obedecerão às estratégias de recuperação de desastres em TI conforme a Tabela 6.8

Tabela 6.8. Estratégias de Recuperação.

Sistema	Estratégia de Recuperação	Processos de Negócio	Custo de Implantação

ERP – Módulo de Vendas	*Hot Site*	Vendas, Check-In	Já existe a estrutura.
ERP – Módulo de Relatórios	*Warm Site*	Vendas, Check-In	Já existe a estrutura.
Telecomunicações (LAN / WAN)	*Hot Site*	Vendas, Check-In, Back Office, Manutenção e Compras, Call Center	Já existe a estrutura.
Navegação Web	*Hot Site*	Vendas, *Check-In*, *Back Office*, Manutenção e Compras	Já existe a estrutura.
ERP – Módulo de Check-In	*Hot Site*	Check-In	Já existe a estrutura.
ERP – Módulo Administrativo	*Cold Site*	*Back Office*	Já existe a estrutura.
Sistema de Contabilidade e Tributos	*Cold Site*	*Back Office*	*R$ 100.000,00*
Sistema de Talentos	*Cold Site*	*Back Office*	*R$ 150.000,00*
E-mail	*Hot Site*	*Back Office*, Manutenção e Compras, *Call Center*	*R$ 350.000,00*
Sistema de Manutenção	*Cold Site*	Manutenção e Compras	R$ 20.000,00
ERP – Módulo de Compras	*Cold Site*	Manutenção e Compras	Já existe a estrutura.
Sistema de Atendimento	*Warm Site*	*Call Center*	*R$ 1.000.000,00*
Telefonia	*Warm Site*	*Call Center*	Já existe a estrutura.

Conforme dito anteriormente o RTO e o MTPD guiam o impacto financeiro, de imagem ou legal gerado através de uma parada dos serviços de TI que suportem processos de negócios críticos, mas cabe salientar que a análise custo/beneficio é fator determinante na escolha de uma estratégia de recuperação de desastres em TI. As estratégias *hot site* e *warm site*, têm uma estrutura duplicada, ou seja, custos duplicados com a infraestrutura de TI.

Além da conotação de estrutura, há uma implicação nas tecnologias a serem utilizadas para implementar esta estratégia. Não há possibilidade de implementar uma tecnologia de alta redundância com uma tecnologia de *software* e *hardware* que não suporte tal configuração. Um exemplo que pode ser estabelecido foi do sistema de e-mail da Voe Sempre, que utilizava uma plataforma baseada no *software sendmail* em um ambiente Unix que não suportava uma configuração de cluster. Logo, foi necessário um investimento em uma nova plataforma de e-mail corporativo baseado em Postfix com dois servidores em cluster ativo/ativo localizados um em cada datacenter.

Deve-se salientar que não necessariamente o uso de uma estratégia de Hot Site acarreta na necessidade de suporte da aplicação à alta disponibilidade, pois se pode

considerar um período de ativação pequeno ainda como Hot Site. Nestes casos, sempre se deve analisar qual a necessidade de negócio e o custo necessário para implementação da alta disponibilidade requerida.

6.4.3 Desenvolvendo e Implementando uma Resposta de GCN

A organização decidiu apenas criar os planos da estrutura existente e irá avaliar para o próximo ciclo do programa de continuidade de negócios o investimento para a infraestrutura de telefonia.

6.4.3.1 Plano de Recuperação de Desastres em TI

Na etapa de desenvolvimento dos planos é o momento em que é necessário o maior esforço por parte das áreas de Tecnologia da Informação e demais áreas de negócio, pois é aqui que os planos são desenvolvidos baseados nas necessidades definidas na etapa anterior e no ambiente corporativo.

Um plano de recuperação de desastres em TI pode ser documentado utilizando como base o modelo 5W2H[2]. A Tabela 6.9 mostra o modelo utilizado para documentação de cada passo dos procedimentos:

Tabela 6.9. Modelo para Documentação de Procedimentos.

ORDEM	
O QUE:	*Ação a ser realizada*
QUEM:	*Cargo do responsável pela ação*
QUANDO:	*Momento de execução*
COMO:	*Passo a passo das ações para ativação*
DURAÇÃO:	*Tempo de duração da ação*

Em virtude das constantes mudanças ao qual o ambiente de TI está sujeito, é aconselhável documentar os procedimentos até a um nível tático, não incluindo informações de procedimentos como instalação de sistemas operacionais, bancos de dados, etc. Estes procedimentos inclusive devem estar documentados separadamente em outros locais como manuais e guias de sistemas da informação. Devendo a configuração dos ambientes estar armazenada em cópias de segurança que serão restauradas no caso da ocorrência de incidentes. A Tabela 6.10 descreve os procedimentos definidos para o processo de Vendas.

[2] Modelo de plano de ação que define: responsabilidades, o que deve ser feito, quando, como, onde porque e os custos e prazos.

Tabela 6.10. Registro de Procedimentos por Processo.

PLANO	Vendas
RTO:	1h
RPO:	0h
CENÁRIO:	*Indisponibilidade do Data Centre Alpha*
RESPONSÁVEL:	*Leonardo Silva - +55 55 555-5678*
SUBSTITUTO:	*Rafael Alves - +55 55 555-8765*
ORDEM	1
O QUE:	Comunicar a indisponibilidade do Datacenter Alpha para o Gerente de TI
QUEM:	Equipe de Monitoramento de TI da Voe Sempre
QUANDO:	Após a detecção da indisponibilidade do Data Center
COMO:	Através de uma ligação telefônica, utilizando a árvore de chamadas pré-estabelecida.
DURAÇÃO:	10 minutos
ORDEM	2
O QUE:	Reunir o time de gestão de crises
QUEM:	Gerente de TI
QUANDO:	Após receber a comunicação da indisponibilidade do Data Center Alpha
COMO:	Através de uma conferência via telefone
DURAÇÃO:	10 minutos
ORDEM	3
O QUE:	Decidir Ativar o Data Center Beta
QUEM:	Time de Gestão de Crises
QUANDO:	Durante a reunião via conferência
COMO:	Através da análise das possibilidades
DURAÇÃO:	10min
ORDEM	3

O QUE:	Ativar equipe de TI
QUEM:	Equipe de Monitoramento de TI
QUANDO:	Após comunicar o gerente de TI
COMO:	Através dos telefones celulares, contidos na árvore de chamadas
DURAÇÃO:	15 minutos
ORDEM	4
O QUE:	Comunicar equipe para ativação do Datacenter Beta
QUEM:	Gerente de TI
QUANDO:	Após decisão de ativar o Datacenter Beta
COMO:	Através de uma ligação para equipe de Monitoramento de TI
DURAÇÃO:	15 minutos
ORDEM	5
O QUE:	Ativar o Datacenter Beta
QUEM:	Equipe de TI
QUANDO:	Após comunicação da equipe de monitoramento
COMO:	Utilizando os procedimentos de ativação
DURAÇÃO:	2h
ORDEM	6
O QUE:	Reunir equipe de Gestão de Crises
QUEM:	Equipe de Gestão de Crises
QUANDO:	Após reunião via telefone do time de Gestão de Crises
COMO:	Reunindo-se em um ponto de encontro a ser definido na reunião
DURAÇÃO:	30 minutos
ORDEM	7
O QUE:	Preparar comunicados internos e externos
QUEM:	Equipe de Gestão de Crises
QUANDO:	Após reunir-se

COMO:	Em conjunto
DURAÇÃO:	1h
ORDEM	8
O QUE:	Identificar a extensão dos danos e passos necessários para recuperar o Datacenter Alpha
QUEM:	Equipe de Gestão de Crises
QUANDO:	Após reunião do time de Gestão de Crises
COMO:	Através da análise de dados obtidos de diversas áreas, incluindo: - Monitoramento de TI - Segurança Corporativa - Engenharia
DURAÇÃO:	45 minutos

Os serviços relacionados ao ERP de vendas e *check-in* adotam a estratégia denominada *hot site* e mesmo com a queda do datacenter Alpha continuaram em operação. A equipe de infraestrutura foi ativada para recuperar os serviços que adotam as estratégias *warm site* e *cold site* da infraestrutura existente no datacenter Beta.

6.4.4 Testando, Mantendo e Analisando Criticamente os preparativos de GCN

A organização passou por um incidente crítico que foi o Black-out do seu datacenter Alpha, esse evento serviu como um teste para seus planos de recuperação de desastres em TI.

Os processos de manutenção e de analise crítica também são ativados no caso do uso dos planos por motivos de incidente, porém sem os testes periódicos nenhum outro processo dessa etapa é possível.

6.4.4.1 Testes Periódicos

Após implementar um plano de recuperação de desastres em TI, deve-se testar a solução a fim de verificar se a mesma está adequadamente implantada ou requer melhorias a fim de atingir as necessidades do negócio. Para isto, a organização definiu:

- Planos de Sistemas Críticos - devem ser testados anualmente via simulação real com uma revisão a cada seis meses via teste de mesa
- Testes de Árvore de Chamadas – devem ser realizados sem aviso prévio.

Para os testes de mesa é essencial a definição dos papéis e responsabilidades, conforme a Tabela 6.11.

Tabela 6.11. Papéis e Responsabilidades.

Pessoa/Cargo	**Papel no Teste**
Gestor de Continuidade de Negócio	Coordenador
Gestor da área de Tecnologia da Informação	Participante / Patrocinador
Dono do plano de continuidade de negócios	Responsável
Usuários da área de negócio	Participantes
Observadores	Observadores

Os observadores são pessoas elencadas pelo Gestor de Continuidade de Negócios para estarem atentos durante o andamento dos testes, já que cabe ao gestor coordenar todas as atividades e desenvolver novas situações durante a leitura dos planos.

Na definição do escopo do teste é essencial avaliar os seguintes aspectos:

- Escopo e a maturidade dos participantes e do plano
- Tempo e orçamento requerido
- Definir objetivos e indicadores a serem medidos durante o teste
- O cenário e duração do teste de maneira a atingir seus objetivos
- Entendimento de todos envolvidos dos planos
- Encaminhar a todos participantes os objetivos e limitações do teste

A Tabela 6.12 apresenta um registro de teste de mesa executado na organização Voe Sempre.

Tabela 6.12. Registro de Teste

Informação Inicial	
Data:	23/08/2008
Hora de Início:	22 h
Localização:	Unidade São Sebastião
Patrocinador:	João Humberto Gonzaga
Responsável:	Luis Garcia
Coordenador:	Manoel Elias
Processos de Negócio:	Vendas
Sistemas Envolvidos:	ERP – Módulo de Vendas
Cenário:	Cenário 3 – Falha de comunicação com a base de dados
RTO:	1h
RPO:	0h
Pressupostos:	- Os demais sistemas devem continuar ativos - O teste ocorrerá com participação via telefone para contato com técnicos - A sincronia deverá indisponibilizar somente a base de dados

	central para área de vendas

Neste caso, o cenário a ser testado apresenta uma falha de sincronia entre bases de dados e, como se pode ver nos pressupostos, o teste ocorrerá somente com a área de Vendas, principal usuária deste sistema. Ou seja, não é necessário envolver todas as áreas que acessam determinado sistema se o mesmo será testado, mas sim as áreas que dependem essencialmente deste sistema ou processo de negócio.

O teste de mesa é executado percorrendo os procedimentos vigentes do plano de continuidade em conjunto com as pessoas que constam como participantes no mesmo. Cabe ao coordenador da atividade, o Gestor de Continuidade de Negócios, conduzir o andamento passo-a-passo, cabendo então aos observadores da atividade identificar: se o passo é passível de ser executado, se não são necessários passos adicionais ou se existem oportunidades para racionalizar o plano. Um relatório final de um teste de mesa aplicado a uma aplicação deve obter como resultado:

- Tempo requerido para restaurar a aplicação
- Problemas encontrados
- Aderência ao plano documentado
- Lições aprendidas
- Plano de ação para resolução dos problemas

A Tabela 6.13 reúne os resultados obtidos após a realização do teste descrito anteriormente.

Tabela 6.13. Resultado de um Teste de Mesa.

Resultados	
Resultado Geral:	Completado com falhas.
Aderência ao plano:	Durante a execução do passo 3, houve a necessidade de comunicação de mais uma pessoa
Tempo:	01:15:20
Problemas:	Demonstrou-se que o plano não prevê situações como horário de almoço e de saída, onde pode ocorrer que as pessoas estejam em locais sem acesso a telefone, Internet ou que requeiram mais de 30 minutos de deslocamento.
Lições Aprendidas:	O tempo requerido para o passo 7, que deveria ser de 15 minutos, tomou cerca de 20 minutos devido à indisponibilidade do contato principal. Logo, indicou-se um terceiro substituto que deve constar no plano.

No caso acima, pode-se notar que há uma falha no tempo estimado do plano e que o mesmo pode falhar no caso de uma situação real ocorrer em horários específicos. Então será necessário rever todo o plano para adequação do mesmo aos requisitos do negócio.

6.5. Conclusão

A proteção dos ativos e a continuidade do negócio são alguns dos principais objetivos da segurança da informação. Para garantir a continuidade das operações mesmo mediante cenários de desastres é fundamental que as organizações, independente do segmento e/ou porte, coloquem em prática um programa de gestão da continuidade de negócio.

Esse programa, conforme discutido na seção 6.2, é composto por planos que têm como objetivo: gerenciar incidentes, garantir o estado de contingência e a recuperação da organização. Nesse capítulo foram apresentados conceitos, práticas e um estudo de caso com foco na elaboração das estratégicas de contingência para os recursos de tecnologia da informação, normalmente denominado de plano de recuperação de desastres.

A elaboração de planos de recuperação de desastres precisa ser cuidadosamente planejada, definida e testada para assegurar que: (a) as estratégias de contingência escolhidas estão de acordo com o nível de serviço vigente e (b) que as pessoas estejam devidamente capacitadas e cientes sobre como proceder mediante eventos que comprometam a continuidade dos serviços de TI.

O processo de construção desses planos compreende um conjunto coordenado de atividades interdependentes que inicia com o entendimento das necessidades da organização. Nessa etapa é necessário obter informações que permitam definir a priorização dos produtos e serviços da organização e a urgência das atividades que são necessárias para fornecê-los. Isso estabelece os requisitos que irão definir a seleção das estratégias de GCN apropriadas.

A definição da estratégia de continuidade permite que uma série de estratégias seja avaliada a fim de que uma resposta apropriada seja escolhida de modo que a organização possa continuar fornecendo esses produtos e serviços em um nível de operações aceitável pelo tempo necessário. Essas escolhas levarão em consideração muitas variáveis, entre elas a resiliência e as opções de contramedidas já presentes na organização.

Em seguida, é chegado o momento de desenvolver e aplicar uma resposta de gestão de continuidade de negócios para cada produto e/ou serviço critico. No caso especifico dos ativos de TI, conforme apresentado na seção 6.3, existe um conjunto de boas práticas amplamente divulgadas e que servem de apoio no momento do desenvolvimento das estratégias de contingência. Essa resposta deve ser regularmente testada, revista e auditada para que a organização esteja ciente a que ponto as suas estratégias e planos estão completos, atualizados.

Ao longo do desenvolvimento das atividades dos autores como profissionais da área de gestão de continuidade de negócio e das pesquisas realizadas para elaboração deste capítulo foram identificados os seguintes aspectos que podem vir a fomentar questões de pesquisa: (a) a grande parte das ferramentas de apoio e suporte ao processo de GCN realizam apenas a gestão da documentação, (b) inexistência de sistema especialista que possa contribuir para definição / estimativa de grandezas como RPO, RTO e MTPD e (c) da mesma forma não foram identificados muitos trabalhos científicos focados no desenvolvimento de ambientes para simulação das estratégias de

contingência como, por exemplo, no trabalho proposto em [Bartolini, Stefanelli and Tortonesi 2009] e [Tjoa and Jakoubi 2008].

Referências

[ABNT 2005] ABNT (2005). Tecnologia da Informação – Técnicas de Segurança - Código de Prática para a Gestão da Segurança da Informação. ABNT NBR ISO/IEC 27002:2005.

[ABNT 2008a] ABNT (2008). Gestão de Continuidade de Negócios Parte 1: Código de Prática. ABNT NBR 15991-1:2008.

[ABNT 2008b] ABNT (2008). Gestão de Continuidade de Negócios Parte 2: Requisitos. ABNT NBR 15991-2:2008.

[ABNT 2008c] ABNT (2008). Gerenciamento de Serviços de TI Parte 1: Especificação. ABNT NBR ISO/IEC 2000-1:2008.

[Bartolini, Stefanelli and Tortonesi 2009] Bartolini, C., Stefanelli, C., Tortonesi, M. (2009). Business-impact analysis and simulation of critical incidents in IT service management. University of Ferrara, Ferrara, Italy.

[BCI 2010] BCI (2010). The Business Continuity Institute Good Practice Guidelines. Disponível em: http://www.thebcicertificate.org/bci_gpgdownload.html. Acessado em 21 de mar. de 2010.

[Cegiela 2006] Cegiela, R. (2006). Selecting Technology for Disaster Recovery. Warsaw University of Technology, Institute of Control and Computation Engineering, Warsaw, Poland.

[Continuity Central 2006] Continuity Central (2006). Business Continuity Unwrapped, Disponível em: http://www.continuitycentral.com/feature0358.htm (em inglês), acessado em 21 de mar. de 2010.

[DRII 2010] DRII (2010). Disaster Recovery International Institute: Professional Practices. Disponível em: https://www.drii.org/docs/profprac_details.pdf. Acessado em 21 de mar. de 2010.

[Husdal 2008]. Husdal (2008). Ericsson versus Nokia – the now classic case of supply chain disruption. Disponível em: http://www.husdal.com/2008/10/18/ericsson-versus-nokia-the-now-classic-case-of-supply-chain-disruption/print/. Acessado em 21 de mar. de 2010.

[IBM Global Services 2007]. IBM Global Services (2007). Continuidade de negócios e resiliência" Disponível em: http://www.ibm.com/br/services/bcr/. Acessado em 21 de mar. de 2010.

[IT Governance Institute 2008a]. IT Governance Institute (2008). IT Governance Institute (2007). COBIT - Control Objectives for Information and related Technology. Disponível em http://www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx. Acessado em 20/06/2010.

[IT Governance Institute 2008b]. IT Governance Institute (2008). Aligning CobiT® 4.1, ITIL® V3 and ISO/IEC 27002 for Business Benefit. Disponível em http://www.isaca.org/knowledge-

center/Research/ResearchDeliverables/Pages/Aligning-COBIT-4-1-ITIL-V3-and-ISO-IEC-27002-for-BusinessBenefit.aspx. Acessado em 20/08/2010.

[Tjoa and Jakoubi 2008] Tjoa, S., Jakoubi, S. (2008). Enhancing Business Impact Analysis and Risk Assessment applying a Risk-Aware Business Process Modeling and Simulation Methodology. The Third International Conference on Availability, Reliability and Security, EUA.

[Wei 2009] Wei, N.Z.W. (2009). The strategic skills of business continuity managers: Putting business continuity management into corporate long-term planning. Journal of Business Continuity & Emergency Planning Vol. 4 No. 1, pp. 62–68. United Kingdom.

[Wiboonrat 2008] Wiboonrat, M. (2008). An Empirical IT Contingency Planning Model for Disaster Recovery Strategy Selection. Graduate School of Information Technology, Assumption University. Bangkok, Thailand.