Capítulo

# 1

# Análise de Malware: Investigação de Códigos Maliciosos Através de uma Abordagem Prática

Laerte Peotta de Melo[1], Dino Macedo Amaral[1], Flavio Sakakibara[2], André Resende de Almeida[2], Rafael Timóteo de Sousa Jr.[1], Anderson Nascimento[1].

[1]Departamento de Engenharia Elétrica – Universidade de Brasília (UnB).
[2]Departamento de Ciência da Computação – Universidade de Brasília (UnB).
Campus Universitário Darcy Ribeiro – Asa Norte – 70910-900 – Brasília, DF – Brasil.

***Resumo***

*Este curso destina-se a divulgar o conhecimento necessário para que profissionais interessados em análise de malware desenvolvam habilidades esperadas em um grupo de resposta a incidentes e forense computacional. A abordagem utilizada é teórico-prática, de modo que o aluno terá acesso aos principais conceitos e discussões sobre as novas tendências de desenvolvimento de códigos maliciosos bem como suas contramedidas. Modelos de condução de incidentes e análise de malware serão apresentados, juntamente com ferramentas utilizadas no processo, apontando quais informações são primordiais e as preocupações necessárias para se conter um incidente. Após a introdução o aluno será desafiado a executar uma análise real, discutida e apresentada em dois cenários distintos.*

## 1.1. Introdução

Nos últimos anos o número de códigos maliciosos vem crescendo, acompanhando o aumento do número de usuários que acessa a Internet [Finjan Research Center 2009]. As defesas frente a essa ameaça têm se mostrado ineficientes. Empresas de segurança que desenvolvem ferramentas de detecção utilizam a forma clássica para encontrar e neutralizar os códigos maliciosos, ou seja, a forma baseada em assinaturas [Dube et al. 2010]. Assim, torna-se notório para o entendimento dos processos de segurança e inovação da tecnologia que é necessário conhecer um artefato para que este seja considerado malicioso. Da mesma forma, é preciso que um vetor de infecção seja encontrado e analisado para que uma assinatura seja produzida.

Devido à grande quantidade desses artefatos, algumas organizações e empresas de segurança utilizam métodos de análise automatizada ou semi-automatizada. Conhecer o funcionamento de um código malicioso é a base para se produzir ferramentas de detecção e proteção eficientes, pois permite que se conheça o contexto que o *malware* pretende atingir, entendendo o público alvo da ameaça, as informações coletadas, o uso e o destino dos dados. Entretanto, o tempo para encontrar as defesas não tem se mostrado condizente com o cenário atual dos ataques, que é extremamente desfavorável para o usuário final. Com essa motivação, os pesquisadores têm se mostrado dispostos a enfrentar o problema e confiantes em propor soluções que tornem os processos, tanto de análise quanto de detecção, eficientes e confiáveis.

Vale notar que, em casos que envolvem fraudes financeiras e roubo de identidades, conhecer a atuação do *malware* é preponderante para erradicar o incidente. A simples descoberta de códigos maliciosos pode sugerir uma ação preventiva em relação, por exemplo, a impedir que um usuário tenha acesso a um sistema, pois não é possível garantir que esse usuário seja corretamente reconhecido, pois o comprometimento de suas informações de identificação fazem o risco de fraude tornar-se alto. Um grande problema enfrentado nessas situações é determinar qual o tipo de comprometimento, ou seja, encontrar qual a capacidade que o código malicioso tem em capturar informações. Esse fator é determinante do tempo de resposta que a organização e as empresas de segurança têm para produzir uma assinatura de reconhecimento do malware.

### 1.1.1. Justificativas

Equipes de resposta a incidentes de segurança (CSIRT)[Brownlee and Guttman 1998] têm enfrentado grandes desafios para manter seus sistemas seguros e controlar possíveis comprometimentos de sistemas que estão sob sua responsabilidade. Entre os desafios a serem enfrentados estão os *malwares*, que são códigos capazes de comprometer totalmente uma infraestrutura de tratamento da informação e que estão tornando cada vez mais complexas as defesas. Além da questão das inovações nas formas de ataques, o número crescente desses códigos maliciosos [Finjan Research Center 2009] representa uma preocupação, não somente para as empresas, mas também para entidades governamentais, que devem se preocupar com ameaças que podem causar danos sistemas críticos de um país[Chen and Abu-Nimeh 2011] e afetar diretamente a segurança física do cidadão desse país.

### 1.1.2. Objetivos

O curso tem como objetivo introduzir o assunto tanto à comunidade acadêmica quanto a profissionais interessados no tema, de modo a subsidiar o entendimento de atuação de *malwares*. O foco é colocado em *trojans* bancários, procurando fazer o aluno compreender todo o processo de detecção, análise e tratamento desse tipo de artefato, bem como demonstrar a obtenção de evidências de que o artefato teve sucesso na coleta de informações da máquina comprometida.

A proposta é que o aluno desenvolva habilidades de identificar uma ameaça, analisar e avaliar o grau de sofisticação. Para isso o curso é aplicado utilizando exposições teóricas e práticas em laboratório. O perfil esperado para os alunos é o de profissionais

que atuem na área de segurança da informação, centro de resposta e tratamento de incidentes de segurança, forense computacional, assim como de discentes/docentes de cursos de engenharia e ciência da computação, com noções básicas de sistemas operacionais Windows e Linux, e redes TCP/IP.

### 1.1.3. Organização do minicurso

Este minicurso está organizado da seguinte maneira: no capítulo 1.1 se apresenta o tema sobre o ponto de vista de sua importância e justificativas para uma formação com objetivo final esperado. A questão do tratamento de incidentes, os termos mais importantes, juntamente com um estudo de caso de análise forense para *phishing*, sãos os temas apresentados no capítulo 1.2. O capítulo 1.3 detalha metodologias aplicadas em forense computacional, suas estratégias de atuação e preservação de evidências. Técnicas e metodologias para análise e condução de incidentes que envolvem *malwares* são detalhadas no capítulo 1.4. A seguir, tratam-se os modelos de análise de metadados no capítulo 1.5, a identificação de códigos ofuscados no capítulo 1.6 e as funções de *hash* e *fuzzy hash* no capítulo 1.7. Finalmente, no capítulo 1.8 são apresentados estudos de casos para análise de *malwares* e, no capítulo 1.9, um modelo de relatório de análise que serve para documentar e gerir o processo de investigação de *malwares*.

## 1.2. Identificação e Tratamento de Incidentes de Segurança

Nesta seção apresentamos as definições de tipos de incidentes e ataques, incluindo os artefatos de vírus, *worm*, *trojan horse* e os principais ataques, como engenharia social, *spam*, *honeynet*, *phishing* [Steding-Jessen 2008], *botnets*, negação de serviço [Binsalleeh et al. 2009] e exploração de vulnerabilidades [Abbas et al. 2006]. São também abordados os principais fatores da resposta a incidentes, os grupos de resposta a incidentes de segurança (CSIRT) [Brownlee and Guttman 1998], suas etapas de trabalho e as metodologias de análise e condução de incidentes, compostas por atividades de coleta, extração, tratamento e resposta[Kent et al. 2006].

Um incidente de segurança é um evento indesejado ou inesperado, que pode ser único ou em série e que possui características de comprometimento do bom funcionamento de uma organização, resultante de uma ameaça à segurança da informação[ABNT 2005]. Especificamente, um incidente pode ser relacionado a eventos como invasões, acessos não autorizados, negação de serviço, contaminação de sistemas através de *malwares*, furto de informações ou qualquer atividade considerada ilegal ou ilegítima. Nesses casos, é necessário que os eventos sejam investigados por pessoas qualificadas a fim de resolverem o problema causado.

Programa malicioso, código malicioso, softwares malicioso, ou simplesmente *malware*, são expressões que se referem a um programa que é inserido em um sistema, normalmente de forma encoberta, com a intenção de comprometer a confidencialidade, integridade ou disponibilidade dos dados da vítima, aplicativos ou o próprio sistema operacional. A produção de *malwares* se tornou a ameaça mais significativa para a maioria dos sistemas, causando danos generalizados e perturbação, ocorrências que necessitam de esforços de recuperação extensa dentro da maioria das organizações.

As técnicas utilizadas pelos *malwares* são associados à violação de privacidade de

um usuário, valendo observar que o denominado *spyware* tornou-se uma grande preocupação para as organizações. Embora a violação da privacidade através desses programas não seja novidade, tornou-se muito mais difundida recentemente [Mell and karen Kent 2005]. A monitoração das atividades pessoais e a realização de fraudes financeiras são ocorrências cada vez mais comuns, assim como a ocorrência de sistemas invadidos. As organizações também enfrentam outras ameaças que são frequentemente associadas aos *malwares*. Uma das formas que tem se tornado comum é o *phishing*. Outras formas de artefatos, como vírus, *backdoors*, cavalos de tróia, *keyloggers*, *rootkits*, também se denominam pelo termo *malware* e serão abordadas a seguir neste capítulo.

### 1.2.1. Spam e Phishing

O termo *spam* é utilizado para referenciar o recebimento de uma mensagem não solicitada, que geralmente tem o caráter de fazer propaganda de algum produto ou assunto não desejado. O termo em inglês utilizado para esse tipo de mensagem é *UCE (Unsolicited Commercial Email)*. A palavra *spammer* é o termo utilizado para definir quem envia esse tipo de mensagem, em cuja elaboração geralmente são utilizados softwares especiais automatizados para coletar bases de emails através de listas de discussão, caixas postais de grupos, ou exploração, através de listas adquiridas, por empresas especializadas nesse tipo de *marketing*. Existem diversas soluções no mercado que buscam conter o recebimento de *spams*, ferramentas estas que basicamente utilizam filtros contendo *blacklists* de *proxys* e *relays* abertos, que são utilizados para o envio das mensagens.

Servidores de *proxies* são utilizados por organizações de maneira a tornar suas estruturas de comunicação mais seguras, pois permitem prover acesso de redes internas a redes externas, como a *Internet*, sendo capazes de impor controles de acesso a conteúdos definidos em uma política de segurança e gerar informações de auditoria. Os servidores *relay* são utilizados conjuntamente com o protocolo SMTP *(Simple Mail Transfer Protocol)* de transmissão e recebimento de mensagens, protocolo este que é considerado inseguro, pois, caso um atacante tenha acesso ao meio de comunicação e capture o tráfego gerado entre cliente e servidor, as credenciais de cada um não são protegidas por criptografia. Apesar de existir serviços SMTP que utilizam de criptografia, a utilização é baixa. Por tais razões, servidores SMTP que estejam com o serviço de *relay* ativado podem permitir que os serviços sejam utilizados por *spammers* e que as mensagens sejam enviadas. Diante de tal situação, recomenda-se que os servidores sejam configurados utilizando técnicas de *hardening*, de modo a permitir que somente pessoas autenticadas e autorizadas possam enviar mensagens, restringindo o uso por terceiros. A Figura 1.1 confirma que é crescente o envio de mensagens não solicitadas.

O termo *phishing*, *phishing scam* ou *phishing/scam* ("fishing"significa pescaria), refere-se ao método pelo qual *iscas* (e-mails) são usadas para *pescar* informações de usuários da Internet, constituindo-se um ataque que tem como objetivo efetuar algum ilícito através do envio de mensagem não solicitada. Essa técnica geralmente é associada à engenharia social, atividade em que o *spammer* explora algum tema de modo a induzir o usuário final a executar alguma atividade ou ação que não faria normalmente. Essas mensagens são projetadas para roubar informações como: dados pessoais, credenciais de clientes de instituições financeiras de usuários. Para compreender a análise da mensagem, deve-se entender o comportamento esperado de um servidor SMTP padrão:

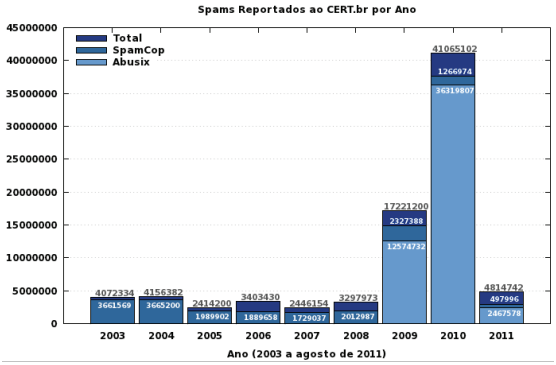


**Figura 1.1. Valores acumulados: 2003 a agosto de 2011 - Fonte: CGI**

1. Usuário escreve uma mensagem utilizando um software ou agente web (MUA - Mail User Agent Outlook, Eudora, Webmail);
2. Usuário envia a mensagem através do software utilizado para escrever a mensagem (MUA);
3. O software (MUA) conecta-se ao servidor SMTP do usuário (MTA - Mail Transfer Agente Sendmail), para isso usa o protocolo SMTP-AUTH, que envia o *username* e *password* do usuário;
4. O servidor SMTP recebe e armazena a mensagem na fila para envio;
5. O servidor SMTP localiza o domínio de destino da mensagem utilizando o protocolo DNS;
6. O servidor SMTP conecta-se conecta ao servidor SMTP do destino da mensagem;
7. Os servidores SMTP trocam identificação;
8. O servidor SMTP remoto verifica os cabeçalhos da mensagem;
9. O servidor SMTP remoto autoriza ou rejeita a mensagem, de acordo com regras de verificação de reverso e políticas anti-spam;
10. O servidor SMTP remoto recebe e entrega ao servidor local (LMTA - Local Mail Transfer Agent Procmail);
11. O servidor local (LMTA) processa a mensagem e entrega na Caixa Postal do usuário destino;
12. O usuário, utilizando o software para acesso (MUA) e suas credenciais *username* e *password* se autentica ao seu servidor de email MRA (Mail Reader Agent POP3, IMAP4) POP *(Post Office Protoco)* ou Imap *(Internet Message Access Protocol)* e

13. O servidor de email (MRA) entrega a nova mensagem para o software de acesso do cliente (MUA);

14. O Usuário pode ler a mensagem.

A Figura 1.2 apresenta um exemplo desse tipo de mensagem. Todo o processo de envio e recebimento é passível de auditoria, permitindo que se compreenda a taxonomia do evento que um *spammer* utiliza.

Em certas ocasiões, o *spammer* não se preocupa em ocultar sua origem, tornando o rastreamento rápido. Entretanto, na maioria dos casos a origem é ocultada ou alterada de maneira a indicar outro local de envio da mensagem.

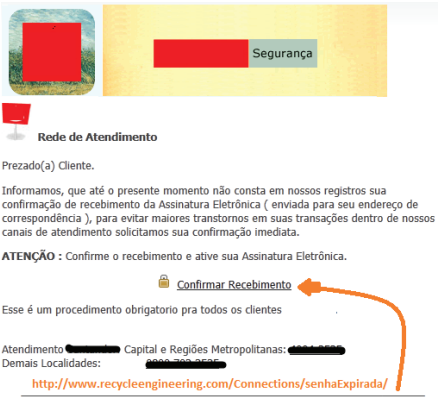


**Figura 1.2. Exemplo de phishing**

Analisando o exemplo da Figura 1.2 é possível coletar uma grande quantidade de informações que podem ser utilizadas para se determinar pontos de comprometimento. A informação coletada apontando o mouse para o link de "Confirmar Recebimento"apresenta o link para download de artefato malicioso, que neste caso indica ser um cavalo de tróia para roubo de credenciais de clientes bancários. Portanto qualquer e-mail enviado possui informações que não são apresentadas pelos programas ou interfaces de usuário. Essas informações podem ser obtidas analisando diretamente o código fonte da mensagem. O código da mensagem, referente a Figura 1.2, é apresentada pela Listagem 1.1.

**Listagem 1.1. Listagem de cabeçalho e-mail phishing**

```
1 Delivered-To: peotta@gmail.com
2 Received: by 10.142.166.12 with SMTP id o12cs148327wfe;
3         Tue, 20 Sep 2011 00:13:22 -0700 (PDT)
4 Received: by 10.236.179.97 with SMTP id g61mr2104588yhm
    .119.1316502409640;
5         Tue, 20 Sep 2011 00:06:49 -0700 (PDT)
6 Received-SPF: neutral (google.com: 200.175.8.217 is neither permitted
    nor denied by best guess record for domain of unknown) client-ip
    =200.175.8.217;
```

```
7  Received: by 10.200.27.12 with POP3 id a12mf92257gwa.33;
8          Tue, 20 Sep 2011 00:06:49 -0700 (PDT)
9  X-Gmail-Fetch-Info: lpi@pop.com.br 1 mail.pop.com.br 110 lpi
10 Received: (qmail 8396 invoked from network); 20 Sep 2011 06:52:38 -0000
11 Received: from unknown ([200.175.8.217])
12           by pop-37.pop.com.br (qmail-ldap-1.03) with QMQP; 20 Sep 2011
                 06:52:38 -0000
13 Delivered-To: CLUSTERHOST pop-8.pop.com.br lpi@pop.com.br
14 Received: (qmail 25792 invoked from network); 20 Sep 2011 06:52:37
       -0000
15 Received: from vvps-152833.dailyvps.co.uk ([91.223.16.239])
16           (envelope-sender <root@vvps-152833.dailyvps.co.uk>)
17           by pop-8.pop.com.br (qmail-ldap-1.03) with SMTP
18           for <lpi@pop.com.br>; 20 Sep 2011 06:52:37 -0000
19 X-C3Mail-ID: 1316501546274674
20 Received-SPF: none (pop-8.pop.com.br: domain at vvps-152833.dailyvps.co
       .uk does not designate permitted sender hosts)
21 Received: from vvps-152833.dailyvps.co.uk (localhost.localdomain
       [127.0.0.1])
22    by vvps-152833.dailyvps.co.uk (8.13.8/8.13.8) with ESMTP id p8K6qPYf
          023569
23    for <lpi@pop.com.br>; Tue, 20 Sep 2011 07:52:25 +0100
24 Received: (from root@localhost)
25    by vvps-152833.dailyvps.co.uk (8.13.8/8.13.8/Submit) id p8K6qPgZ
          023568
26    for lpi@pop.com.br; Tue, 20 Sep 2011 07:52:25 +0100
27 Date: Tue, 20 Sep 2011 07:52:25 +0100
28 From: Grupo Santander S/A <santander@santander.infoemail.com>
29 To: lpi@pop.com.br
30 Subject: Regularize sua Assinatura ôEletrnica!
31 Message-ID: <1316501545.@santander.infoemail.com>
32 Content-Type: text/html
33 X-Remote-IP: 91.223.16.239
34 X-Spam-Status: No, tests=filter, spamicity=0.500001
```

A investigação desse tipo de ocorrência é feita analisando o código fonte da mensagem sempre de baixo para cima, neste caso a linha 34 é uma informação de que esta mensagem especificamente foi testada por um filtro de spam e não foi considerada uma mensagem não solicitada. A linha 33 é importante para definir a origem do spam, o servidor SMTP insere essa informação para facilitar a identificação de quem envia, o campo pode variar de acordo com a versão, podendo ter nomenclatura como *X-Originating-IP* ou *X-IP*.

Para identificar a origem do *spammer* utilizaremos o campo *X-Remote-IP* que aponta para o IP 91.223.16.239. Utilizando o protocolo *whois*, é possível encontrar informações como: Contato Administrativo (Admin Contact), Contato Técnico (Technical Contact) e Contato de Cobrança (Registrant Contact). A saída resumida está disponível através da Listagem 1.2.

**Listagem 1.2. Resultado para a consulta whois**

```
1 IP Information for 91.223.16.239
2 IP Location:   United Kingdom United Kingdom Daily Internet Ltd
3 ASN:  AS31727
```

```
4  Resolve Host:   vvps-152833.dailyvps.co.uk
5  IP Address:     91.223.16.239
6  inetnum:        91.223.16.0 - 91.223.16.255
7  netname:        DAILY-UK-VPS
8  descr:          Daily Internet LTD
9  country:        GB
10 org-name:       Daily Internet Ltd
11 org-type:       OTHER
12 address:        67-69 George Street
13 address:        LONDON
14 address:        W1U 8LT
15 phone:          +44 115 9737260
16 fax-no:         +44 115 9725140
17 person:         Andrew Gilbert
18 address:        Node4 Ltd
19 address:        Millennium Way
20 address:        Pride Park
21 address:        Derby
22 address:        DE24 8HZ
23 phone:          +44 845 123 2222
```

A listagem 1.1, os campos apresentados pela linha 28 e 29, respectivamente informam origem e destino da mensagem. A linha 27 apresenta informação sobre a data, importante recurso para se conduzir identificação de quebra de sigilo judicial, pois a data é necessária para associar uma conexão a um usuário naquele exato momento. Como o servidor está localizado no Reino Unido (linha 2 da Listagem 1.2 é importante considerar o horário mundial, que neste caso é determinado pelo Greenwich Mean Time (GMT). A linha 11 revela que a mensagem foi entregue para o cliente do servidor *pop.com.br*, entretanto este ainda não é o destino final da mensagem, continuando a análise podemos perceber através da linha 9 que o servidor *Gmail* coletou a mensagem utilizando uma conexão através do protocolo Post Office Protocol (POP3) utilizando a porta 110. O destino final então é determinado pelo campo *Delivered-to*, apresentado na linha 1.

De acordo com a análise da mensagem recebida e apresentada pela Figura 1.2, foi possível analisar informações de headers da mensagem, conforme Listagem 1.1, identificando sua origem, conforme Listagem 1.2.

O método de roubo pode ser associado a um formulário web, onde as informações são solicitadas à vitima, outro método é conhecido como cavalo de tróia, em que a vítima instala o artefato de maneira desavisada. Os cavalos de tróia são tratados na seção 1.2.4. Alguns exemplos de situações que envolvem esse tipo de ataque:

- Formulários recebidos por e-emails que solicitam informações confidenciais;
- Links recebidos que direcionam para a instalação de cavalos de tróia;
- Alteração através de malwares que direcionam o tráfego DNS para um *proxy* controlado pelo atacante;
- Páginas falsificadas de empresas de comércio eletrônico, sites de leilões, instituições financeiras, órgãos do governo.

Praticamente os ataques que, utilizam técnicas de engenharia social, tem como vetor de comprometimento o *phishing*. *Spammers* precisam convencer o usuário a executar ações que tem como objetivo comprometer seu dispositivo, portanto esse tipo de ataque é fortemente dependente da colaboração da vítima.

Ganhar a confiança do usuário tem sido um desafio para os atacantes, que buscam camuflar suas investidas tornando-se parecidos com sites de serviços mais comumentes utilizados, como bancos, redes sociais, servidores de e-mail, comércio eletrônico. Como exemplo podemos observar a Figura 1.3, onde o atacante tenta reproduzir, de maneira fiel, o *site* de uma instituição financeira, entretanto, nesse tipo de ataque o ponto a ser explorado é a distração da vítima, que não percebe as modificações, muitas vezes grosseiras e que solicita de maneira direta o que o atacante precisa para se passar por um cliente legítimo da instituição.

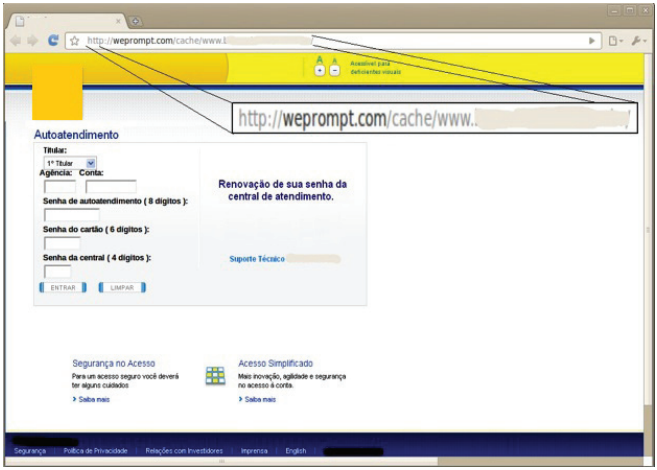


**Figura 1.3. Exemplo de página falsa**

Os programas maliciosos podem ser divididos primordialmente através de sua dependência ou não do hospedeiro. Os tipos dependentes são por exemplo: vírus, bombas lógicas e *backdoors*, os não-dependentes *worms* e zumbis. A replicação é outra maneira de se classificar, pois existem os programas que possuem a capacidade de se replicar, como vírus e *worms*, e os que não se replicam, como bombas lógicas, *backdoors*, zumbis, *rootkits* e *keylogger*. Entretanto é bastante comum os *malwares* que possuem capacidades híbridas, sendo capazes de se replicar e ao mesmo tempo possuírem outras características inerente às necessidades. Para que a classificação de cada tipo seja feita, é necessário que os termos utilizados sejam de conhecimento do analista que irá conduzir a investigação, portanto apresentaremos uma breve descrição de cada tipo mais comum de malware.

### 1.2.2. *Honeynets* e *Honeypots*

*Honeypots* são recursos e serviços computacionais em rede criados e monitorados com o objetivo de serem atacados e comprometidos, visando identificar novos ataques e ferramentas utilizadas. *Honeynet* são conjuntos de sensores monitorados de maneira a identificar tráfego suspeito e malicioso.

Um sistema de *honeypots* pode ser dividido em duas categorias[Steding-Jessen 2008]:

**Alta Interatividade:** Todos os sistemas e serviços disponibilizados são serviços reais sem inserções de vulnerabilidades propositadas. Nesse ambiente todo o tráfego é monitorado e controlado, de maneira que um comprometimento não seja utilizado para causar danos a outras redes.

**Baixa Interatividade:** Os sistemas são reais, entretanto os serviços disponibilizados são emulados através de ferramentas criadas exclusivamente para reproduzir o comportamento esperado desses serviços. Oferecem um grau menor de risco de serem utilizados como base a ataques a outras redes.

O objetivo dos *honeypots*, independentemente de sua categoria, é coletar informações de ataques, identificando origens e destinos, bem como serviços mais procurados. A monitoração é feita geralmente utilizando-se sistemas de detecção de intrusos (IDS) e armazenamento e correlacionamento de *logs*.

### 1.2.3. Vírus

Os vírus de computador foram introduzidos na década de 1980, com funções simples que ocasionalmente geravam inconvenientes ou apenas mostravam informações ao usuário. Atualmente esse tipo de código traz um risco significativo com potencial destrutivo e que demandam grande esforço das organizações para manterem seus sistemas a salvo. A designação de vírus é tratada de acordo com sua função. Para que um código malicioso seja considerado um vírus ele deve ter a capacidade de auto replicação, ou seja, fazer uma cópia de si mesmo e distribuir essa cópia para outros arquivos e programas do sistema infectado.

Cada vírus pode utilizar um mecanismo de infecção diferente, por exemplo: inserir seu código em programas executáveis sobrescrevendo parte do código de maneira a permitir que toda vez que o programa infectado seja executado o código viral também seja executado. Portanto o principal objetivo de um vírus é replicar-se e contaminar o maior número possível de programas, de maneira a comprometer outros sistemas. Podemos dividir os vírus em duas grandes categorias [Kent et al. 2006]: vírus compilados e vírus interpretados.

**A) Vírus compilados**

Esse tipo de código malicioso é composto de um programa fonte que é compilado com alguma linguagem e direcionado para um determinado sistema operacional, tipicamente esse tipo de vírus pode dividido em três categorias:

**Infector de programas:** Infectam programas executáveis, onde o programa infectado se

propaga para infectar outros programas no sistema, bem como outros sistemas que usam um programa infectado compartilhado.

**Setor de Boot:** Infectam o setor Master Boot Record (MBR) do disco rígido ou mídia removível inicializável. Esse tipo de vírus foi muito utilizado nas décadas 1980 e 1990, atualmente está em desuso.

**Multipartite:** Utiliza métodos de infecção múltipla, normalmente infectando arquivos e setores de boot. Assim, os vírus multipartite combinam as características dos vírus que infectam arquivos e setor de inicialização.

**B) Vírus interpretados**

Esse tipo de vírus é composto de código-fonte que pode ser executado apenas por uma determinada aplicação ou serviço, ao contrário dos vírus compilado, que pode ser executada por um sistema operacional. Tornaram-se mais comuns, pois considerados mais fáceis de escrever e modificar do que outros tipos de vírus. Não é necessário desenvolver capacidades técnicas elevadas, pois um atacante pode adquirir um vírus interpretado e modificar seu código fonte. Muitas vezes há dezenas de variantes de um único vírus interpretado, apenas com alterações triviais a partir do original. Os dois principais tipos de vírus interpretados são: vírus de macro e vírus de script.

**Vírus de macro:** Utilizam técnicas de propagação baseadas em anexo de documentos que executam macros, como planilhas eletrônicas e processadores de texto. Estes vírus tendem a se espalhar rapidamente, porque os usuários freqüentemente compartilham documentos com recursos de macro habilitados. Além disso, quando uma infecção ocorre, o vírus infecta o modelo que o programa usa para criar e abrir arquivos. Uma vez que um modelo é infectado, cada documento que é criado ou aberto com esse modelo também será infectado.

**Vírus de script:** Nesse tipo de vírus a principal diferença, em relação ao de macro, é a linguagem utilizada. O código é escrito para um serviço e sistema operacional específico, como por exemplo VBScript que executa em um servidor com sistema operacional Windows.

Técnicas de ofuscação podem ser utilizadas no código de vírus, particularmente para dificultar sua detecção. As técnicas mais comuns são:

**Criptografia:** O código possui funções para cifrar e decifrar o executável do vírus, gerando porções de códigos aleatórios para cada infecção.

**Polimorfismo:** Um modelo mais robusto que usa criptografia, com o código gerando um executável com tamanho e aparência diferentes para cada infecção, podem também habilitar funcionalidades diferentes na ação.

**Metamorfismo:** Assim como os polimórficos mudam o comportamento, podendo gerar códigos novos a cada iteração através de técnicas de códigos desnecessários, sendo recompilado e criando um novo executável.

### 1.2.4. Cavalos de Tróia

De acordo com a mitologia Grega, um cavalo de Tróia é um "presente"que esconde uma ação não esperada, são programas que não se replicam e que tem aparência inofensiva, escondendo sua proposta maliciosa. Esse tipo de malware pode tomar o lugar de um programa com intenções legítimas, executar todas as funções esperadas e então, conjuntamente, executar as ações para a qual foi programado, dentre essas ações pode-se citar:

- Coletar informações da máquina contaminada, como credenciais, fotos, arquivos;
- Esconder sua presença e ações de maneira a impedir ou dificultar sua detecção;
- Desabilitar softwares de segurança como *firewall*, anti-virus, *anti-malwares*.

Atualmente os cavalos de tróia utilizam diversas técnicas para que sua presença passe desapercebida ou que sua detecção seja difícil. Esse tipo de malware é o mais comum quando se trata do cenário da Internet brasileira, pois é bastante utilizado por fraudadores que aproveitam sua ação para coletar todo o tipo de informação necessária para fraudes financeiras em lojas de comércio eletrônico, bancos, financeiras, etc. No capítulo 1.6 trataremos das técnicas de ofuscação que são utilizadas pelos atacantes com o objetivo de dificultar sua identificação e análise.

### 1.2.5. Worms

Um programa que é capaz de se propagar automaticamente através de redes, enviando cópias de si mesmo de computador para computador, de sistema para sistema, essa é a definição de um worm. Diferentemente do vírus, o *worm* não tem a capacidade de infectar outros programas inserindo seu código em outros programas ou arquivos, também não depende de execução para se propagar em outros sistemas, pois é através da exploração de vulnerabilidades que esse tipo de malware contamina outros sistemas, aproveitando de falhas de configuração ou softwares vulneráveis.

A grande ameaça do *worm* reside em deteriorar a performance de um sistema, sobrecarregando serviços e gerando tráfego, que em alguns casos, pode provocar negação de serviço (*DoS - Deny of Service*). Por ser capaz de comprometer sistemas sem a necessidade de um usuário instalar ou executar algum programa, esse tipo de ataque tem se popularizado entre os criminosos, pois permite que em um curto espaço de tempo um grande número de sistemas seja comprometido, diferente dos vírus, que dependem do hospedeiro e usuário. Outra vantagem é que vulnerabilidades e configurações mal feitas são bastante comuns e diariamente surgem novas fraquezas em sistemas. Para efeito de classificação, existem dois tipos principais de worms, serviços em rede (*network service worms*) e envio em massa (*mass mailing worms*).

Os *worms* de serviços (*network service worms*) utilizam o processo de exploração de uma vulnerabilidade em um serviço de rede, geralmente associado a um determinado sistema operacional ou aplicativo. Depois que um sistema esteja infectado o código malicioso inicia a verificação de outros sistemas vulneráveis pela rede. Sua ação aumenta a medida que outras máquinas são comprometidas, o que justifica a queda de performance do sistema de comunicação. Como não há necessidade de interação humana, rapidamente uma rede vulnerável pode ser totalmente mapeada e comprometida.

### 1.2.6. Backdoors

O termo *backdoor* é geralmente associado a um programa malicioso que fica aguardando uma determinada ação para que permita uma conexão remota ou acesso não autorizado, e que não seja detectado pelos dispositivos de segurança. A instalação de um backdoor é inerente à invasão de um sistema, onde o atacante habilita um "serviço"que o permita ter acesso quando necessário ao sistema comprometido, geralmente possuem capacidades especiais:

**Zombies:** O termo zumbi é decorrente da utilização de máquinas comprometidas através de *bots*, possuindo a capacidade de propagação como um worm. Essas máquinas são controladas por atacantes que compõem um sistema maior. Botnets são redes compostas por milhares de computadores zumbis e que são utilizadas para ataques coordenados, como DDoS (*Distribution Deny of Service*), envio de *spams* e *phishing*.

**Remote Administration Tools - RAT:** Permite a um atacante ter acesso a um sistema quando desejar e em alguns casos o sistema controlado coleta e envia informações como imagens de tela, teclas digitadas, arquivos acessados sendo possível ativar remotamente dispositivos como impressoras, webcams, microfones e autofalantes. Exemplos de RAT's: Back Orifice, Netbus e SubSeven.

Um invasor, que tenha controle sobre uma botnet, poderá utilizá-la com o objetivo de aumentar o poder em seus ataques, inclusive a outras botnets, de modo a incorporar as máquinas zumbis de outras redes. Vários ataques tem sido reportados utilizando esse tipo de rede, como por exemplo: Envio de *phishing* ou *spam*, ataques de negação de serviço, fraudes bancárias entre outros.

### 1.2.7. Keyloggers

*Keylogger* ou *keystroke logger*, São programas utilizados para monitorar e armazenar atividades do teclado de uma determinada máquina. Qualquer tipo de informação inserida, como conteúdo de documentos, credenciais de bancos, número de cartão de crédito, senhas são capturadas e armazenadas localmente ou enviadas para servidores remotos utilizando protocolos como FTP *(File Transfer Protocol)*, HTTP/HTTPS *(HyperText Transfer Protocol Secure)* ou anexo de e-emails. Existem algumas variações desse tipo de malware, que monitora cliques do mouse (mouseloggers) ou efetuam cópias de imagens de tela (screenloggers). Apesar de ser classificado como um *malware*, esse tipo de software pode ter sua utilização legítima, quando uma organização, através de uma política de segurança que preveja seu uso, habilite a funcionalidade e monitore as atividades de uso de sistemas que compõem sua rede interna.

### 1.2.8. Rootkits

O termo *rootkit* provém do uso em sistemas Unix/Like, onde um atacante, após obter sucesso em comprometer um sistema, instala um conjunto de ferramentas, ou *toolkit* que tem como finalidade esconder a presença maliciosa dos artefatos. Sua utilização também ocorre em sistemas baseado em outras plataformas como o Windows. Apesar do termo

*root* ser proveniente de sistemas Unix, que neste caso seria mais indicado ter a nomenclatura modificada para *adminkits*, entretanto não é isso que acontece.

Os *rootkits* permitem que um atacante possa controlar remotamente um sistema comprometido, apagar todas os rastros de comprometimento, e esconder sua presença do administrador, de modo que alguns programas são modificados ou simplesmente substituídos por programas modificados que compõem o pacote de ferramentas de um *rootkit*. É tarefa de um *rootkit* capturar outras informações que possam ser utilizados pelo atacante, como senhas, credenciais, outros sistemas que possam ser comprometidos.

### 1.2.9. Kits de ferramentas

Um atacante possui um conjunto de ferramentas que o auxiliam a identificar sistemas que possam ser comprometidos, de modo a ter o controle de uma rede, total ou parcialmente. Dentre essas ferramentas podemos listar:

**Sniffers:** Utilizados para monitorar uma rede capturando todos os pacotes do tráfego gerado, tanto wireless quanto redes cabeadas. Alguns tipos de *sniffers* tem a capacidade de interagir com uma rede enviando pacotes, são conhecidos como *sniffers* ativos. Outros são projetados exclusivamente para capturar dados sensíveis como senhas e credenciais, utilizando protocolos mais comuns, como FTP, HTTP, POP3, IMAP *(Internet Message Access Protocol)*, entre outros.

**Port Scanners:** É um programa utilizado para varrer uma rede buscando serviços que esteja disponíveis através de portas TCP *(Transmission Control Protocol)* ou UDP *(User Datagram Protocol)*, potencialmente localizam um alvo e coletam informações como serviços e versões, permitindo que o atacante identifique serviços vulneráveis.

**Scanners de vulnerabilidades:** Esse tipo de programa pode ter sua utilização licita, onde uma organização o utiliza para varrer seus sistemas em busca de vulnerabilidades conhecidas, permitindo que seja identificada e corrigida, entretanto um atacante pode utilizar de maneira a encontrar e explorar as vulnerabilidades antes que sejam descobertas pela organização.

**Password Crackers:** Sistemas que utilizam-se de senhas para proteger sua aplicações são suscetível a ataques de força bruta, onde são testadas um conjunto de senhas aleatórias, ou ataques por dicionário, onde uma lista de palavras são testadas para encontrar qual é utilizada para proteger um segredo.

Um atacante sofisticado pode desenvolver suas próprias ferramentas e ter seu kit exclusivo, já um atacante que possua menos conhecimento, como *script kids*, utilizam-se de ferramentas prontas, de modo a agir com a intenção de invadir ou comprometer um sistema de maneira indiscriminada.

## 1.3. Método aplicado em forense computacional

A forense computacional é uma parte do processo de resposta a incidentes, geralmente o profissional que atua na área está associado a um centro de resposta a incidentes de segurança (CSIRT)[Mieres 2009]. O objetivo é, através de técnicas de investigação, confirmar

ou descartar uma ocorrência de incidente, permitindo que a organização estabeleça controles eficientes para recuperação e tratamento de evidências, protegendo a privacidade e as políticas da organização.

Toda a atividade é documentada de modo a minimizar interrupções das operações de um sistema, do mesmo modo a preocupação em se manter a integridade das evidências deve ser seguida, evitando-se que as informações sejam contaminadas em sua condução. Como resultado de uma análise, um relatório é produzido de maneira clara a explorar e responder as seguintes indagações: **onde, como, onde e porquê** um incidente ocorreu, de modo a relacionar o incidente a uma entidade, ou seja, **quem** gerou o incidente.

A condução de uma análise é algo complexo, que envolve diversas variáveis e que exigem uma abordagem clara e sem dúvidas quanto aos quesitos solicitados, portanto é importante ter claro os termos envolvidos[Prosise et al. 2003]:

- Preparação: É necessário que exista planos pré incidentes, pois quando o incidente ocorrer existirá um processo a ser seguido.
- Detecção: É necessário ter maneiras de se detectar a ocorrência de um incidente, seja ele através de monitoração ou análise dos eventos nos dispositivos de segurança.
- Resposta: Ao se identificar um incidente é necessário obter respostas sobre os fatos, em uma resposta inicial é necessário que as evidências sejam coletadas, não descartando entrevistas com as pessoas envolvidas.
- Estratégia: É necessário que exista estratégias de ação, que permita definir qual a melhor resposta utilizar mediante as evidências coletadas.
- Duplicação: Alguns casos a duplicação (Cópia pericial) pode ser descartada, entretanto para as análises quem envolvam acesso a imagens originais recomenda-se que a duplicação seja executada.
- Investigação: Deve-se o fato de esclarecer os acontecimentos, buscando responder os quesitos relacionados à análise, principalmente o autor do incidente e suas razões.
- Relatório: Toda análise deve ser concluída com a apresentação de um documento que descreva todos os fatos relevantes à investigação.

O método utilizado por um atacante representa a taxonomia do próprio ataque, sendo possível listar da seguinte maneira:

1. Identificação do alvo;
2. Coleta de informações;
3. Identificação de vulnerabilidades;
4. Comprometimento do sistema;

5. Controle do sistema;
6. Instalação de ferramentas;
7. Remoção de rastros;
8. Manutenção do sistema comprometido.

Quando um atacante identifica seu alvo ele coleta o maior número possível de informações que possam auxiliá-lo na identificação de vulnerabilidades. O comprometimento de um sistema é baseado nas vulnerabilidades encontradas, sendo que o controle é feito através da exploração dessas vulnerabilidades, é nesse ponto que o atacante controla e busca aumentar o nível de privilégio, que tem como objetivo ser o administrador do sistema. A instalação de ferramentas é o passo posterior ao comprometimento, nesta etapa o atacante instala o conjunto de ferramentas que geralmente é composto de um *rootkit* que irá auxiliá-lo a manter o sistema funcional e com suas atividades protegidas do administrador legítimo, onde todos os rastros serão removidos e o ultimo passo é a correção da vulnerabilidade explorada, o atacante deve proteger o sistema de modo que outro atacante não encontre e explore a mesma vulnerabilidade.

As fases que compõem a análise forense são apresentadas pela Figura 1.4.

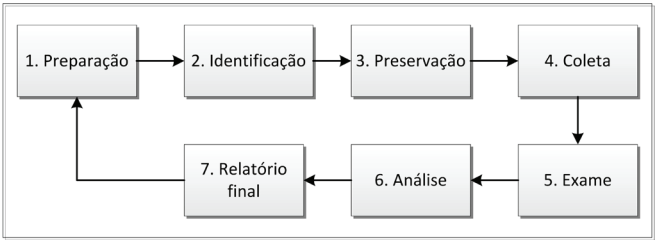


**Figura 1.4. Metodologia Forense**

A metodologia proposta não é algo fechado, mas que deve ser seguida, sendo a preparação algo inerente ao próprio processo. Um analista forense deve considerar a detecção como primeira passo no tratamento de um incidente, mesmo que essa função não esteja associada a atuação ela deve ser considerada no processo de análise, pois cabe ao analista forense obter a informação de como o incidente foi detectado. A preservação do local deve fazer parte da política de segurança da organização, pois o fato de haver interação com o sistema comprometido pode prejudicar a coleta da evidencias. O exame deve ser feito com objetivos diretos do caso, ou seja, o analista deve possuir os quesitos que deverão ser respondidos através do exame das evidências, e só então emitir o relatório final da análise, é nessa fase que o analista expõe o caso, baseado totalmente em evidências coletadas durante o processo.

O analista forense irá se deparar com dois tipos de cenários: Análise Post mortem e Máquinas vivas.

**Post mortem:** Este cenário contempla a situação sem atividade no disco rígido onde evidências voláteis estão perdidas e não há atividade do invasor, sendo que não existe a necessidade de contenção do ataque e é provável que algumas evidências foram modificadas ou perdidas.

**Máquinas vivas:** Neste cenário o equipamento encontra-se em funcionamento e há atividades tanto do sistema quanto de rede, há chances do invasor estar presente e ativo, sendo necessário procedimentos de contenção do ataque e preocupação quanto a modificações e contaminação das evidências.

Para os cenários em que o equipamento encontra-se inativo a fase de coleta é simplificada, entretanto a fase de exame e análise pode ser dificultada, pois uma séria de informações podem ter sido perdidas. Nos cenários em que o equipamento encontra-se ativo a preocupação recai sobre quais processos coletar, portanto a volatilidade é uma importante variável a ser utilizada, na seção 1.3.1 será discutido o tema. Após a definição de ordem de coleta tem-se a decisão a tomar: Desligar o equipamento de maneira elegante (*shutdown*) ou simplesmente cortar a energia (*power-off*).

Para executar o shutdown deve-se considerar o ganho em se manter a integridade do sistema em funcionamento, isso é bastante relevante quando se trata de um banco de dados em funcionamento, entretanto existe a contrapartida, que é alterações em arquivos e dependendo do controle efetuado o atacante pode executar ações na máquina que detectem o desligamento e iniciam outras ações como criptografia de disco ou destruição de informações através de remoção segura de arquivos, conhecido como técnicas de *wipe*[1], que além de apagar um arquivo, sobrescreve a área ocupado com dados aleatórios, dificultando assim a análise.

O corte de energia (*power-off*) não modifica os arquivos, entretanto impacta diretamente no sistema de arquivos do sistema operacional e seus serviços, podendo danificar a estrutura da informação e tornar o sistema inoperante em uma eventual decisão de retornar os sistemas.

Nos dois casos apresentados o analista forense ainda deve se preocupar com sistemas criptografados, atualmente bastante comuns, neste caso a falta de observância deste fator pode inviabilizar totalmente a análise, pois sistemas criptografados enquanto estão ativos ficam disponíveis, assim que desligados tornam-se verdadeiras caixas-pretas. Como exemplo pode ser citado o projeto Truecrypt[2] de criptografia *on-the-fly* de disco, *flashcard* (pendrive) e volumes, sendo totalmente transparente para o usuário, possuindo outras características como esteganografia associada a proteção de volumes criptográficos, conhecido como *Deniability*, protege o usuário caso seja de alguma forma obrigado a informar sua senha de acesso, pois não há como o atacante saber se existe uma ou duas senhas de proteção.

### 1.3.1. Volatilidade de informações: Coleta e Exame

Apresentar métodos de preservação de dados voláteis, bem como a ordem de volatilidade para a extração e análise dessas informações [Brezinski and Killalea 2002].

[1] `http://wipefreespace.sourceforge.net/`
[2] `http://www.truecrypt.org/`

Incidentes de segurança geram algum tipo de violação a política da organização, portanto devem ser tratados de maneira a minimizar as perdas e maximizar a resposta com o objetivo de fornecer aos administradores do sistema a melhor solução. As provas decorrente do incidentes são eventos muitas vezes voláteis e deve existir orientações sobre a coleta e o arquivamento de provas relevantes. Uma boa prática é que os analistas conheçam e saibam como coletar as evidências consideradas voláteis, portanto protegendo a informação quando a coleta e armazenamento relacionados ao incidente em questão.

Quando um sistema é identificado como comprometido, existem diversas maneiras para acelerar o re-instalação do Sistema Operacional e facilitar a reversão para um estado conhecido, deixando dessa maneira as coisas mais fáceis para os administradores. Entretanto a obtenção de provas são mais difíceis se comparada às facilidades em se recuperar um sistema, pois ainda não existe um modelo que contemple todas as necessidades de uma investigação, e isso pode ser considerado um agravante quando os atacantes conseguem ser mais rápidos que o analista forense.

De acordo com a RFC-3227 [Brezinski and Killalea 2002] que trata exclusivamente da ordem de volatilidade na coleta de evidências, o importante é capturar as informações que não estejam disponíveis caso o equipamento analisado fosse desligado, a Tabela 1.1 apresenta a ordem de acordo com a prioridade de coleta de cada evidência.

**Tabela 1.1. Ordem de prioridade para coleta de evidências**

| Ordem | Tipo | Descrição |
|---|---|---|
| 1 | Registros | Registros do SO |
| 2 | Tabelas de roteamento | cache arp, tabela de processos, estatísticas do SO (kernel) |
| 3 | Memória | Dump memória RAM (Random Access Memory) |
| 4 | Arquivos temporários do SO | Arquivos criados e mantidos temporariamente pelo SO |
| 5 | Disco rígido | Cópia *"bit-a-bit"* |
| 6 | Logs de monitoração | Registros de dispositivos de rede como proxy, servidores de acesso |
| 7 | Configurações físicas | Informações sobre o local e topologia de acesso |
| 8 | Dispositivos de armazenamento | cartões de memória, disquetes, pendrives, cameras |

## 1.4. Metodologia de Análise de Malware

Apresentar metodologia de condução de incidente [Mell and karen Kent 2005] que envolve artefatos maliciosos:

- Preparação: Equipe de resposta habilitada a lidar com esse tipo de incidente, com políticas internas que contemplem a atividade;
- Detecção e análise: Detectar, analisar e validar rapidamente um incidente envolvendo malwares;

- Contenção: Interromper e evitar a propagação do malware e prevenir maiores danos aos sistemas;
- Erradicação: Remover o malware dos sistemas infectados;
- Recuperação: Retornar a funcionalidade e os dados dos sistemas infectados, implantando medidas de contenção temporária;
- Pós incidente: Reduzir custos no tratamento dos incidentes utilizando a base de informações adquiridas e sugerindo contramedidas.

A Figura 1.5 apresenta o fluxo de tratamento de um incidente que envolve códigos maliciosos. A detecção é prioritária na condução da análise do incidente, onde a organização deve possuir um modelo que permita a identificação de um evento de segurança de maneira pró ativa. Os passos da metodologia abstrai processos técnicos, mostrando as etapas de forma a permitir a adequação onde esta será utilizada.

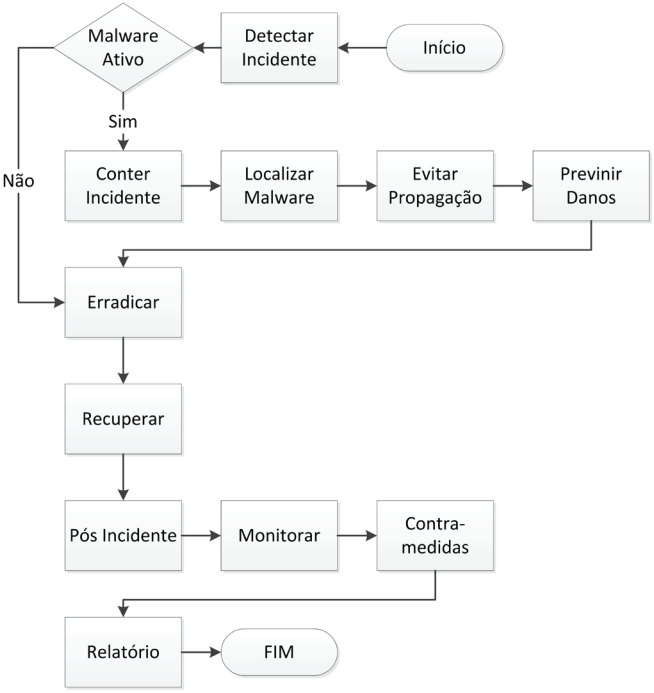


**Figura 1.5. Metodologia Análise de Malware**

A fase de preparação envolve ações que devem ser seguidas quanto a identificação de um incidente que envolve *malwares*, nesta fase é importante que a organização possua

uma política de segurança que preveja esse tipo de incidente e as atividades de tratamento e mitigação, permitindo a redução do risco de segurança e consequentemente uma menor atividade que envolva a exploração desse tipo de vulnerabilidade. A rapidez em se detectar as atividades de um *malware* tornam a resposta mais eficientes, pois possibilita a erradicação e recuperação com um escopo menor do que seria em um sistema totalmente comprometido, permitindo assim, diminuir o número de sistemas comprometidos.

Organizações que possuem um grupo de resposta a incidentes de segurança, convém que sejam adotados métodos para se detectar infecção de *malwares*, nesses métodos é importante que os profissionais que serão responsáveis por conduzir a análise e tratamento do incidente possuam uma sólida formação em identificação e classificação que envolve a segurança dos ativos da empresa.

A forma clássica para se detectar um *malware* é a utilização de softwares como antivírus, sistemas de detecção de intrusão baseados em análise de redes ou análise de hosts monitorados (Intrusion Detection System - IDS) e (Host Intrusion Detection System - HIDS), bem como softwares que detectam e removem *spywares* ou qualquer outro tipo de ferramenta que possa ser utilizada pela organização. Convém que um profissional tenha competências listadas no capítulo 1.3 para em casos em que um malware possua características mais complexas e que requeiram uma análise mais detalhada.

A metodologia adotada deve prever treinamentos a usuários do sistema da organização, de modo a que todos tenham conhecimento dos riscos, fazendo-se as recomendações de segurança que devem constar na política de segurança da organização.

## 1.5. Análise de Metatados

Informações disponíveis no código compilado: Bibliotecas utilizadas, módulos estáticos e dinâmicos, compiladores, sistemas operacionais 32 ou 64 bits, versões e outras informações importantes.

O termo *metadado* refere-se a informações sobre o próprio dado, possui diversos contextos como informações de programas compilados que estão armazenadas internamente ao código compilado e apenas é revelado quando solicitado ou extraído pelo analista forense. Cada tipo de arquivo possui informações relacionadas ao conteúdo, como tipo de compilador utilizado, bibliotecas compiladas junto ao código, identificadores únicos, tipos de sistemas operacionais compatíveis, versões 32 ou 64 bits de execução e outras variáveis.

Extrair as informações que estão ocultas é apenas uma questão de saber o que está procurando. Existem diversas ferramentas que se propõem em resolver essa tarefa, entretanto em alguns casos pode ser que o analista não tenha certeza de qual informação está buscando, esses casos geralmente são associados quando não existe ainda um incidente relatado, portanto o especialista está em busca de evidências que podem sugerir um incidente. Diversos formatos de arquivos permitem que informações sejam "escondidas"ou alteradas, particularmente um arquivo possui um formato de codificação em que este deverá ser armazenado, cabendo ao analista saber diferenciar esses tipos de arquivos.

Para cada sistema operacional existe um método que é utilizado para determinar o formato de um arquivo específico, além dos metadados é possível utilizar informa-

ções encontradas no cabeçalho (file header) desses arquivos. Nesta abordagem é possível identificar o tipo do arquivo analisado independente de sua extensão, permitindo que seja utilizada a melhor estratégia para se recuperar os metadados de acordo com o formato identificado.

Na Listagem 1.3 apresenta-se a extração da informação do cabeçalho do arquivo, neste caso temos dois arquivos hexdump.exe e winhex.exe de tamanhos diferentes e mesma extensão. Para este exemplo utilizou-se a ferramenta *hexdump* [3].

**Listagem 1.3. Dump de arquivos EXE**

```
1  hexdump.exe   71.168 bytes    13/06/2005   12:35:32
2
3   -0 -1 -2 -3   -4 -5 -6 -7   -8 -9 -A -B   -C -D -E -F
4
5   4D 5A 90 00   03 00 00 00   04 00 00 00   FF FF 00 00
6   B8 00 00 00   00 00 00 00   40 00 00 00   00 00 00 00
7   00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
8   00 00 00 00   00 00 00 00   00 00 00 00   80 00 00 00
9   0E 1F BA 0E   00 B4 09 CD   21 B8 01 4C   CD 21 54 68
10
11 winhex.exe   2.034.688 bytes    18/05/2011   16:00:00
12
13  -0 -1 -2 -3   -4 -5 -6 -7   -8 -9 -A -B   -C -D -E -F
14
15  4D 5A 50 00   02 00 00 00   04 00 0F 00   FF FF 00 00
16  B8 00 00 00   00 00 00 00   40 00 1A 00   00 00 00 00
17  00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
18  00 00 00 00   00 00 00 00   00 00 00 00   00 01 00 00
19  BA 10 00 0E   1F B4 09 CD   21 B8 01 4C   CD 21 90 90
```

Comparando os dois arquivos é possível determinar sua semelhança completa da linha 5 e 15 onde cada byte, representado pela informação em hexadecimal, permanece a mesma para os tipos de arquivos EXE nos sistemas operacionais windows, pois não necessariamente o que defini um executável desse tipo seja sua extensão. É importante considerar que um atacante pode alterar a extensão de um arquivo para outra menos crítica na tentativa de enganar sua vitima ou mesmo os sistemas de anti virus que analisam arquivos de acordo com sua extensão, muito utilizado em sistemas web mail de empresas.

Os dois primeiros bytes (4*D*5*A*) representam um arquivo executável *Windows/-DOS*, são utilizados para identificar as seguintes extensões: COM, DLL, DRV, EXE, PIF, QTS, QTX e SYS.

Esta técnica de identificação é conhecida como *magic numbers*, que é uma maneira de incorporar metadados muitas vezes associada com o Unix e seus derivados, inserindo um "número mágico"dentro do próprio arquivo. Originalmente, esta técnica foi usado para um conjunto específico de 2 bytes identificadores, inseridos sempre no começo de um arquivo, entretanto qualquer sequência pode ser considerada como um número válido, possibilitando que um formato de arquivo seja identificado de maneira exclusiva. Portanto possibilitar a detecção dessas variáveis é uma maneira simples e eficaz de distinguir entre diversos tipos de formatos de arquivos, facilitando ao sistema operacional, a

[3] http://www.richpasco.org/

obtenção de informações importantes. A Tabela 1.2 apresenta algumas assinaturas mais comuns, a lista completa e atualizada está disponível[4].

**Tabela 1.2. Lista parcial de assinaturas de arquivos (*magic numbers*)**

| Assinatura | Tipo | Descrição |
|---|---|---|
| 50 4B 03 04 | ZIP | Compactador PKZIP |
| 37 7A BC AF 27 1C | 7Z | Compactador 7-Zip |
| 50 4B 03 04 14 00 06 00 | DOCX, PPTX, XLSX | Microsoft Office |
| 50 4B 03 04 14 00 08 00 08 00 | JAR | Arquivo Java |
| 7B 5C 72 74 66 31 | RTF | Texto (Rich text format) |
| 49 44 33 | MP3 | Arquivo de audio (MPEG-1 Audio Layer 3) |
| 7F 45 4C 46 | ELF | Arquivo executável (Linux/Unix) |
| 89 50 4E 47 0D 0A 1A 0A | PNG | Imagem (Portable Network Graphics) |
| 00 00 00 18 66 74 79 70 33 67 70 35 | MP4 | Arquivo video (MPEG-4 video) |
| 25 50 44 46 | PDF, FDF | Adobe (Portable Document Format) |

Existem diversas ferramentas que permitem identificar os tipos de arquivos baseado em assinaturas, é conveniente que o analista conheça como é feito a identificação e só então utilize a aplicação que melhor atenda suas necessidades. Neste curso recomendamos a utilização de ferramentas livres com códigos abertos, como *file*, que é um comando padrão dos sistemas Linux. Existem diversos projetos de portar as ferramentas de linha de comando do Linux para Windows, como é o caso do projeto *gnu utilities for windows*[5].

Por ser um padrão utilizado para definir os tipos de arquivos, a identificação simplesmente baseada em uma sequência binários pode ser aproveitadas pelos adversários para explorar vulnerabilidades, inserindo características de outros tipos de arquivos em arquivos aparentemente inofensivo, como é o caso de arquivos de formato de documentos como *PDF*, que podem ser alterados de maneira a explorar falhas em softwares. Nesse entendimento é necessário conhecer técnicas que não sejam somente as clássicas.

## 1.6. Códigos Ofuscados

Métodos utilizados pelos desenvolvedores para dificultar a análise e detecção pelos antivírus. Será apresentado métodos de ofuscação baseado em packers (compactadores de executável), criptografia, *blinders*, *joiners* e *wrappers*.

[4] http://www.garykessler.net/library/file_sigs.html
[5] http://unxutils.sourceforge.net/

### 1.6.1. *Packers*

*Packers* ou *compressors* são técnicas utilizadas para ofuscação, *packing*, de código. Basicamente atuam comprimindo ou cifrando executáveis originais, buscando dificultar a identificação pelos antivirus e análise estática dos códigos por parte do analista. Porém, enquanto compactados, os programas não desempenham as suas funcionalidades primárias. Assim, uma rotina de descompressão, normalmente localizada ao final do arquivo, é necessária para descompressão antes de serem carregados à memória. A Figura 1.6 apresenta as etapas de codificação e decodificação de um arquivos executável.

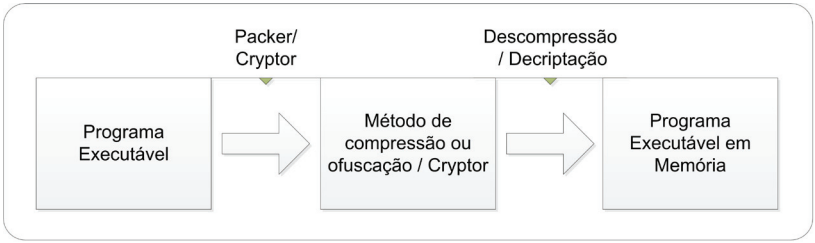


**Figura 1.6. Criação e execução de arquivos compactados ou encriptados**

Essa descompressão ocorre, normalmente, por meio de rotinas específicas para determinados *packers*, Tabela 1.3, salvos raros casos em que os programas responsáveis por comprimir também possuem funções nativas para descompressão. Nesse ponto, o analista deve tomar cuidado, uma vez que o programa por ele utilizado para tentar descomprimir determinado *malware* pode ser desde um programa falso, assim sendo ele nunca alcançará êxito na descompressão, até *malwares* em si, cujo o objetivo é comprometer as máquinas de possíveis analistas.

**Tabela 1.3. *Packers* e seus respectivos *unpackers***

| **Packers** | **Unpackers** |
|---|---|
| ASPack | ASPackDie |
| MEW | UnMew |
| FSG | UnFSG |
| PECompact | UnPECompact |

Em alguns casos, o código de algum malware pode ser ofuscado utilizando packers diferentes, gerando polimorfismo [Martins et al. 2010], pois para cada código uma assinatura diferente é necessária para sua presença seja detectada.

Existem algumas técnicas para se detectar a presença de packers em códigos executáveis, as mais conhecidas são as que analisam a estrutura de arquivos que utilizam a estrutura *portable executable*, que foi projetado para suportar versões dos sistemas operacionais da Microsoft.

O projeto PEiD[6], que atualmente encontra-se suspenso, tem a capacidade de detectar mais de 600 tipos diferentes de ofuscadores packers, cryptors e compiladores em arquivos executáveis portáveis. Para detecções de arquivos modificados e desconhecidos é utilizado técnicas heurísticas, que aumentam a possibilidade de falsos positivos.

TrID[7] é um utilitário desenvolvido para identificar os tipos de arquivo a partir de suas assinaturas binárias, atualmente identifica cerca de 4410 arquivos diferentes. Embora haja utilitários similares a ferramenta não tem regras fixas, em vez disso, pode reconhecer novos formatos de arquivos de forma automática. Para isso possui uma base de assinaturas e utiliza a probabilidade para calcular a proximidade de um determinado arquivo a um tipo específico.

Para análises estáticas o analista deve ter a preocupação em identificar se existe ofuscador ou não, entretanto, nos casos em que a análise seja executado utilizando o método dinâmico é possível realizar uma cópia *dump* do processo em memória, uma vez que quando é carregado, todas as suas funcionalidades estão ativas. Os processos de análise serão discutidos no capítulo 1.8.

A identificação do tipo de ofuscador utilizado auxiliará a análise em busca de strings e padrões entre os tipos de *malwares*, a Tabela 1.3 apresenta alguns tipos de ferramentas para esse tipo de atividade, entretanto, mesmo que não seja possível executar o processo inverso do ofuscador, o analista pode seguir com o trabalho, neste caso em um ambiente controlado o código malicioso será executado e as atividades monitoradas armazenadas, a análise dinâmica será detalhada no Cenário 1.8.3.

### 1.6.2. *Cryptors*

Criptografar um executável é o objetivo de um *Cryptor*, também conhecido como *encryptors* ou *protectors*. Esta técnica possuiu características similares aos *packers*, discutido na sessão 1.6.1, porém alcançado de uma maneira diferente. Ao invés de compactar o arquivo executável, um *Cryptor* simplesmente utiliza de algoritmos criptográficos para ofuscar um arquivo, resultando em dificultar a análise estática e sua identificação por softwares antivírus ou IDSs (*Intrusion Detection System*). Esse tipo de técnica torna difícil a análise através de engenharia reversa. Assim como os packers, criptografar um executável torna necessário acoplar ao código uma função inversa de decifração, que funciona em tempo de execução ao programa original, onde todo o processo é feito em memória.

Uma lista com os packers, unpackers e cryptors mais comuns pode ser encontrada no Anexo 1.11 Tabela 1.7.

### 1.6.3. *Binders, Joiners e Wrappers*

Binders são utilizados para ofuscar códigos com propriedades similares aos packers e cryptors, com o objetivo de dificultar sua identificação e análise. Joiners são utilizados para combinar dois arquivos em um, mantendo a propriedade de execução simultânea, isso é ideal quando um atacante envia um programa licito e junto insere um malware (ex. Keylogger) que pode ser instalado pela vitima sem qualquer tipo de interação, esse

[6] http://www.peid.info/
[7] http://mark0.net/

método é conhecido como instalação silenciosa.

## 1.7. Funções de *Hashing*

Neste capítulo será apresentado os modelos de funções de hash utilizado na comparação de novos malwares, e sistema de indexação por similaridade de arquivos, que utiliza um processo conhecido como *context triggered piecewise hashes* (CTPH) ou *fuzzy hashing* [Kornblum 2006], auxiliando assim a análise por similaridade.

As funções de *hash* são algoritmos desenvolvidos para verificação de integridade de um arquivo, de modo que dois arquivos diferentes tenha saídas diferentes. O algoritmo é conhecido como uma função unidirecional, onde a entrada variável gera uma saída fixa de tamanho suportado pelo algoritmo, ou seja, com base no resultado de um cálculo não é possível obter o texto de entrada.

Para efeito da pericia forense e análise de malware, as funções de hash são indispensáveis, pois é uma maneira rápida e barata para se comparar arquivos. Não é intenção deste minicurso apresentar o algoritmo em si, mas seu resultado como um todo, pois sem ele não seria possível garantir que a informação está integra.

As funções SHA *(Secure Hash Algorithm)*[Jones 2001] e MD5 *(message-digest algorithm)*[Rivest 1992] são funções de hash que atendem os requisitos de segurança provendo uma baixa probabilidade de colisões, que são resultados de dois arquivos diferentes com a mesma saída. Para que exista uma maior conformidade de segurança é recomendado que o analista utilize mais de uma função simultaneamente.

A ferramenta hashdeep[8] possui a capacidade de computar, comparar, ou verificar vários arquivos, como resultado tem-se o exemplo apresentado na Tabela 1.4.

**Tabela 1.4. Funções *hashing***

| Função | Saída | Tamanho |
|---|---|---|
| MD5 | 79970AF07E4FDD8AAA6AFC3EFFF1B5D5 | 128 bits |
| SHA1 | 3A7FE56DAE27BBF7EDCB4C7C839D5EDBA46A05A8 | 160 bits |
| SHA256 | 9949EB0D844B2AB6A1623C1EA69CBE1CEDC19BABBFAE6BC24865DE5B4E18A681 | 256 bits |

As funções de *hashing* facilitam a busca por arquivos, pois se algum artefato é encontrado e seu hash calculado, é possível efetuar uma busca por um mesmo resultado independente de nome ou extensão.

Em casos em que um arquivo seja alterado o resultado será diferente para cada alteração, entretanto é uma técnica conhecida pelos atacantes, que podem efetuar pequenas alterações com o objetivo de que não sejam reconhecidos. Portanto é necessário ir além dessa técnica.

*Context Triggered Piecewise Hash*(CTPH)[Kornblum 2006] ou simplesmente *fuzzy hashing* foi desenvolvido para identificar essas pequenas alterações. O técnica apresenta o conceito de comparar arquivos pela semelhança do contexto, independente do tipo de função de hash utilizado.

Para os casos de análise de malware, é possível determinar o grau de semelhança

[8] `http://md5deep.sourceforge.net/`

analisando *dumps* de memória dos processos gerados pelos artefatos maliciosos. Para efeitos práticos a ferramenta *ssdeep* [9] implementa as funções esperadas de *fuzzy hashing*.

**Listagem 1.4. Utilização ferramenta ssdeep**

```
1 ssdeep.exe -b teste.c >ssdeep.hash
2 ssdeep,1.1--blocksize:hash:hash,filename
3 48:a/mssIVGFPE3Z/iPYmYDFjf7KkItzWWsxCAw7fwTwpCS0sbhxdL:omsVVeMQAZDIAPw7
    oTwp50slD,"teste.c"
4 ssdeep.exe -bm ssdeep.hash teste1.c
5 teste1.c matches ssdeep.hash:teste.c (93)
```

A Listagem 1.4 apresenta a ferramenta *ssdeep*. Na linha 1 é criado a base de informações armazenando o resultado no arquivo *ssdeep.hash*, o conteúdo deste arquivo é apresentado pelas linhas 2 e 3, uma alteração é feita propositadamente no arquivo analisado *teste.c* e salvo em *teste1.c*. A linha 4 apresenta o comando para comparação e finaliza na linha 5 mostrando que o arquivo *teste1.c* é 93% similar ao arquivo *teste.c*.

## 1.8. Preparando uma imagem para análise

Ao cursista será entregue uma imagem composta por sistema operacional funcional, preparado em ambiente virtualizado[10], com todas as ferramentas necessárias para iniciar os trabalhos de análise. Neste capítulo serão apresentados dois estudos de casos reais.

### 1.8.1. Estudo de caso: Analisando códigos maliciosos

A etapa prática busca aplicar os conhecimentos iniciais adquiridos no minicurso em análises de casos reais. Neste intuito serão apresentados dois cenários distintos e mais comuns.

**Cenário 1.** *Análise forense de máquina comprometida por malware.*

Neste cenário, a máquina a ser analisada, encontra-se comprometida, onde o programa malicioso está totalmente funcional e em memória. Através de técnicas forenses [Aquilina et al. 2008], o analista deve ser capaz de identificar quais procedimentos a seguir na captura e análise, tendo como premissa inicial, encontrar o vetor de comprometimento e atividades que possam ser consideras suspeitas.

**Cenário 2.** *Análise de código malicioso capturado.*

Neste cenário o analista terá acesso a um código malicioso disponibilizado para os trabalhos. Utilizando técnicas de análise de um *phishing* discutido no capítulo 1.2.1 através da análise da Listagem 1.1, preparando o analista para responder a um incidente de maneira eficiente, analisando o código e as interações com a máquina alvo, no intuito de investigar quais informações são coletadas pelo *malware* e qual o destino destas informações.

[9] http://ssdeep.sourceforge.net/
[10] http://www.virtualbox.org/

### 1.8.2. Cenário 1: Análise forense de máquina comprometida por malware

O objetivo do analista nesse cenário será investigar se a máquina em questão está infectada por um *malware* ou não.

**Identificando e encontrando a ameaça**

Como o analista não sabe exatamente a que tipo de código malicioso a máquina em questão foi exposta, cabe a ele buscar por indícios que corroborem a identificação tanto do tipo da ameaça em questão quanto do local onde o código está armazenado.

Alguns fatores que influenciam a análise, e que devem ser seguidas na localização e identificação dos códigos maliciosos são[Aquilina et al. 2008]:

- Processo de origem desconhecida;
- Processos que não são familiares ao analista;
- Processos familiares, porém localizados em locais não usuais do sistema;
- Processos que parecem ser familiares, porém com pequenos erros de grafia;
- Processos que já foram identificados como suspeitos, por originar comportamentos anômalos, em outras etapas de análise, como na captura do tráfego de rede.

É bastante comum os atacantes tentarem esconder a presença do artefato de sua vitima, algumas técnicas geralmente associam processos conhecidos do sistema operacional com outros nomes similares ou em locais diferente do normal. Existem tentativas de se identificar quais processos através do nome em execução, como é o caso do projeto *Process Library*[11].

Uma característica dos malwares para a plataforma de sistema operacional *Windows*, é que a maioria necessita ser capaz de se manter em execução quando o sistema é desligado ou reiniciado, dessa maneira a máquina estará sempre sob o controle do atacante. Portanto é muito comum, que ao infectar um equipamento, o malware deve inserir chaves no registro do sistema, permitindo que em qualquer situação este seja executado. O analista deve conhecer registros mais utilizados e ferramentas que o auxiliem a obter essa informação.

A tática utilizada por diversos *malwares* é armazenar seu código em diretórios, chaves de registro ou arquivos específicos no qual o Windows busca por aplicativos que são inicializados a cada sessão, os chamados *autoruns*. Essa é uma maneira eficaz de um processo se manter em execução sem que o usuário tenha conhecimento. Algumas ferramentas que auxiliam na detecção de programas *autorun* são [Aquilina et al. 2008]:

**Autoruns** [12]: Ferramenta com interface gráfica (GUI) ou de linha de comando (CUI) que mostra quais programas estão configurados para serem executados durante a inicialização do sistema ou login;

[11] http://www.processlibrary.com/
[12] http://www.sysinternals.com/

**WhatInStartup** [13]: Utilitário exibe a lista de todos os aplicativos que são carregados automaticamente.

O analista deve ser capaz de diferenciar quais programas representam, de fato uma ameaça, uma vez que *autoruns* são extremamente comuns. É necessário uma busca por indícios que apontem para uma atividade suspeita de qualquer processo em execução no momento da análise. Uma recomendação é que a busca inicie pela análise dos processos executados a partir de diretórios diferentes dos esperados para cada processo, como por exemplo, Internet Explorer sendo executado a partir da pasta pessoal do usuário, ou que o nome do processo seja diferente do esperado.

Segundo a documentação da Microsoft, existem quatro chaves *Run* no registro que faz com que determinados programas sejam executados automaticamente, neste caso a análise se baseia nos registros do sistema operacional Windows XP.

**Tabela 1.5. Chaves de registro - *autoruns***

| Registro | Descrição |
|---|---|
| `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run` | Os comandos explicitados nessa chave são invocados a cada *login* do usuário. |
| `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run` | Os comandos explicitados nessa chave são invocados a cada *login* do usuário. |
| `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce` | Após a execução dos comandos presentes nessas chaves, elas são apagadas. |
| `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce` | Após a execução dos comandos presentes nessas chaves, elas são apagadas. |

Cada chave da Tabela 1.5 contém um conjunto de valores e cada valor é uma linha de comando. Dessa forma, caso o *malware* altere o valor original de uma chave e adicione sua localização através da variável *path*, ele será executado sem que haja necessidade de interação com o usuário, que neste caso passa a ser sua vitima.

Chaves *RunOnce* possuem algumas características que podem ser utilizadas por *malwares*:

- As chaves desse tipo são executadas de maneira sequencial, fazendo com que o Windows Explorer aguarde o término das mesmas, antes de seguir com o *logon* habitual, isso permite que o *malware* execute rotinas antes mesmo de o usuário realizar o processo de *logon*. Dessa maneira, após a autenticação do usuário, o ambiente pode já estar comprometido.
- Chaves *Run* e *RunOnce* são ignoradas em modo de segurança, porém, caso um asterisco $*$ seja acrescentado ao início do valor, chaves *RunOnce* podem ser executadas

[13] `http://www.nirsoft.net/`

mesmo em tal modo: dessa maneira o *malware* tem mais uma garantia de que será executado.

A utilização do *Autoruns*, Figura 1.7, permite ainda uma outra maneira de se identificar processos suspeitos. A Ferramenta disponibiliza a função *jump*, que direciona o analista ao diretório onde se encontra o arquivo executável referente ao processo. Por meio desse diretório, o arquivo suspeito pode ser capturado e enviado para testes em serviços de antivirus online, como o virustotal[14], que possui uma base atualizada de mais de 40 tipos diferentes de antivirus, sendo possível efetuar pesquisas da data que o artefato foi enviado e caso tenha sido considerado malicioso, apresenta a quantidade de antivirus que o detectam.

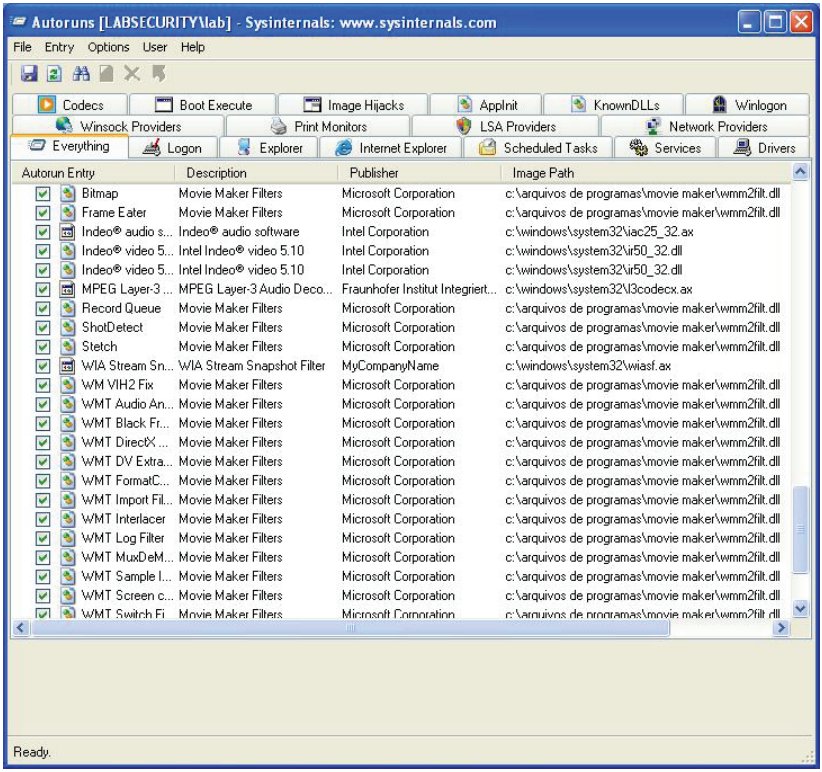


**Figura 1.7. Execução do programa *Autoruns***

É possível que o atacante tente enganar sua vitima fazendo pequenas alterações no nome do processo, mas isso dificilmente enganará um expert que deve conhecer os

[14] http://www.virustotal.com

processos mais comuns do sistema operacional em que trabalha. A ferramenta também permite que seja listada as bibliotecas (DLL's) que cada processo utiliza, localizando diferenças e identificando ataques conhecidos como *dll injection* que é uma técnica usada para executar código no espaço de endereço de memória de outro processo, forçando a carregar uma biblioteca dinâmica. Uma lista das bibliotecas do sistema pode ser obtida através da ferramenta *ListDLLs*[15], permitindo que se identifique arquivos e suas respectivas bibliotecas associadas que estejam fora do padrão esperado.

Para que o analista tenha uma capacidade maior de identificação de processos, existe a ferramenta *HijackThis*, que permite ao analista rapidamente coletar as chaves de registros que aproveitam a execução automática de programas. Essa ferramenta analisa o sistema e gera como saída um arquivo de *log* de todas as chaves de registro que podem ser exploradas pelos malwares a fim de se manterem em execução, a Listagem 1.5 apresenta o resultado de uma coleta.

**Listagem 1.5. Log gerado pelo programa HijackThis**

```
1  Logfile of Trend Micro HijackThis v2.0.4
2  Scan saved at 16:17:42, on 23/9/2011
3  Platform: Windows XP SP2 (WinNT 5.01.2600)
4  MSIE: Internet Explorer v6.00 SP2 (6.00.2900.2180)
5  Boot mode: Normal
6  Running processes:
7  C:\WINDOWS\System32\smss.exe
8  C:\WINDOWS\system32\winlogon.exe
9  C:\WINDOWS\system32\services.exe
10 C:\WINDOWS\system32\lsass.exe
11 C:\WINDOWS\system32\VBoxService.exe
12 C:\WINDOWS\system32\svchost.exe
13 C:\WINDOWS\System32\svchost.exe
14 C:\WINDOWS\system32\spoolsv.exe
15 C:\WINDOWS\Explorer.EXE
16 C:\WINDOWS\system32\VBoxTray.exe
17 C:\Documents and Settings\lab\Desktop\Nova pasta\voegol.com.exe
18 C:\Arquivos de programas\Internet Explorer\iexplore.exe
19 C:\Documents and Settings\lab\Desktop\HijackThis.exe
20 R1 - HKCU\Software\Microsoft\Internet Explorer\Main,Search Page = &http
       ://home.microsoft.com/intl/br/access/allinone.asp
21 O2 - BHO: AcroIEHelperStub - {18DF081C-E8AD-4283-A596-FA578C2EBDC3} - C
       :\Arquivos de programas\Arquivos comuns\Adobe\Acrobat\ActiveX\
       AcroIEHelperShim.dll
22 O4 - HKLM\..\Run: [] C:\WINDOWS\system\sistem.exe
23 O4 - HKLM\..\Run: [Adobe Reader Speed Launcher] "C:\Arquivos de
       programas\Adobe\Reader 9.0\Reader\Reader_sl.exe"
24 O4 - HKCU\..\Run: [Google Update] "C:\Documents and Settings\lab\
       çõConfiguraes locais\Dados de aplicativos\Google\Update\GoogleUpdate
       .exe" /c
25 O4 - HKUS\S-1-5-19\..\Run: [CTFMON.EXE] C:\WINDOWS\system32\CTFMON.EXE
       (User 'LOCAL SERVICE')
26 O4 - HKUS\S-1-5-20\..\Run: [CTFMON.EXE] C:\WINDOWS\system32\CTFMON.EXE
       (User 'NETWORK SERVICE')
27 O4 - HKUS\S-1-5-18\..\Run: [CTFMON.EXE] C:\WINDOWS\system32\CTFMON.EXE
```

[15] www.sysinternals.com

```
    (User 'SYSTEM')
28  O4 - HKUS\.DEFAULT\..\Run: [CTFMON.EXE] C:\WINDOWS\system32\CTFMON.EXE
    (User 'Default user')
29  O9 - Extra button: Messenger - {FB5F1910-F110-11d2-BB9E-00C04F795683} -
     C:\Arquivos de programas\Messenger\msmsgs.exe
30  O9 - Extra 'Tools' menuitem: Windows Messenger - {FB5F1910-F110-11d2-BB
    9E-00C04F795683} - C:\Arquivos de programas\Messenger\msmsgs.exe
31  O14 - IERESET.INF: SEARCH_PAGE_URL=&http://home.microsoft.com/intl/br/
    access/allinone.asp
32  O16 - DPF: {E2883E8F-472F-4FB0-9522-AC9BF37916A7} - http://platformdl.
    adobe.com/NOS/getPlusPlus/1.6/gp.cab
33  O22 - SharedTaskScheduler: éPr-carregador Browseui - {438755C2-A8BA-11D
    1-B96B-00A0C90312E1} - C:\WINDOWS\system32\browseui.dll
34  O22 - SharedTaskScheduler: Daemon de cache de categorias de componente
    - {8C7461EF-2B13-11d2-BE35-3078302C2030} - C:\WINDOWS\system32\
    browseui.dll
35  O23 - Service: Remote Packet Capture Protocol v.0 (experimental) (
    rpcapd) - CACE Technologies, Inc. - C:\Arquivos de programas\
    WinPcap\rpcapd.exe
36  O23 - Service: VirtualBox Guest Additions Service (VBoxService) - Sun
    Microsystems, Inc. - C:\WINDOWS\system32\VBoxService.exe
37  --
38  End of file - 2727 bytes
```

As linhas 7 a 19 apresentam informações dos processos em memória, permitindo que se identifique qualquer atividade fora do padrão esperado, por isso é importante que o analista conheça os principais processos do sistema analisado. Entretanto essa informação ainda não é definitiva, pois um código malicioso pode facilmente substituir um processo válido por outro processo que irá desempenhar uma função dupla. Pode-se perceber que a linha 17 apresenta um processo suspeito, pois é um arquivo executável em uma pasta não convencional. Seguindo a análise, na linha 22 outro arquivo suspeito *sistem.exe*, grafias erradas de arquivos de comandos lícitos é uma tática comum que os atacante utilizam para nomear os malwares, pois um usuário muitas vezes desatento não irá perceber que o comando está errado, simplesmente passa desapercebido.

Para que as coisas sejam facilitadas é possível submeter o arquivo de *log* para análise remota, através do site do desenvolvedor `http://www.hijackthis.de` o *log* obtido é analisado de acordo com o padrão esperado de um arquivo desse tipo, a Figura 1.8 apresenta o resultado. O analista ao examinar o resultado deve estar atento para os processos desconhecidos *unknown process* que geralmente é marcado com um caracter "?"e para os arquivos sujos *nasty*, geralmente marcado em vermelho com o caracter "*X*".

Com base em um *pool* de ameaças e programas já conhecidos, os programas que estavam em execução na máquina alvo são identificados e classificados de acordo com sua periculosidade. Processos marcados como desconhecidos ou perigosos já são um bom ponto de partida para a identificação do *malware*.

**Captura do código malicioso**: Uma vez que o código malicioso tenha sido identificado, não é indicado que o analista realize qualquer tipo de interação com o mesmo, uma vez que se tem pouco ou nenhum controle sobre o estado da máquina e sobre a execução do programa.

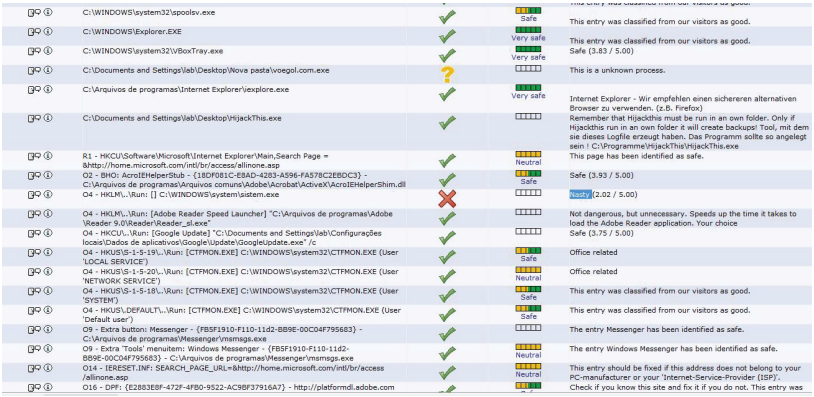


**Figura 1.8. Análise do log gerado pelo HijackThis**

O analista pode optar por extraí-lo da máquina infectada e transportá-lo a uma máquina específica na qual será possível executar uma análise mais profunda e controlada. Isso ocorre porque, em um ambiente mais controlado, o analista poderá utilizar, por exemplo, *snapshots* do sistema, comparando o estado anterior e posterior da execução do *malware*. Uma vez que o *malware* seja extraído, pode-se seguir os passos descritos no Cenário 2.

### 1.8.3. Cenário 2: Análise de código malicioso capturado

O objetivo desse cenário é preparar o analista para a investigação de um código malicioso.

Como um *malware* é um código potencialmente perigoso, devem ser tomadas medidas de segurança para que a análise desse código não comprometa ainda mais a máquina alvo e, tão pouco, infecte outros *hosts*. Por esse motivo, recomenda-se que, sempre que possível, a análise de um código malicioso seja executada em um ambiente isolado, como uma máquina virtual, ou em *Online Sandboxes* como é o caso do projeto *Anubis* [16] e outros projetos como *threatexpert* [17] e *joesecurity* [18].

Neste cenário a análise será dividida em duas etapas distintas:

- **Análise estática**: Análise do código sem execução.
- **Análise dinâmica**: Análise do código através de sua execução e monitoração das interações.

[16] http://anubis.iseclab.org/
[17] http://www.threatexpert.com/
[18] http://www.joesecurity.org/

### 1.8.4. Análise estática

Análise estática é geralmente mais eficiente na detecção de malware, porque usa informações de alto nível. Nessa etapa da análise, tentaremos extrair o maior número de informações úteis - IP's, *paths* de arquivos, *logins* e senhas, entre outros - do código em si do objeto.

Poderão ser encontradas duas situações distintas:

- Quando o código não encontra-se ofuscado, logo seu código encontra-se em linguagem natural e as informações podem ser coletadas de maneira simples, procurando-as diretamente no código.

- Quando o código foi ofuscado, ou seja, foi utilizada uma técnica de ofuscamento e o código foi "compactado". Nesse caso pouca ou nenhuma informação pode ser obtida sem que o código seja descompactado.

Independentemente de o código ter sido ofuscado ou não, deve-se ser capaz de garantir a consistência do arquivo analisado, bem como sua unicidade. Isso é importante para determinar se o *malware* alterou seu próprio código durante a execução, ou pode ser necessário provar que o código não foi alterado durante a análise.

Para isso, utilizaremos *hashing functions*. Uma função *hash* é uma função unidirecional, que gera uma string única (salvo alguns casos de colisões), para cada entrada, e de tamanho fixo. Aqui utilizaremos o MD5 (*Message Digest 5*). O programa responsável por realizar essa conversão será o *md5sum*. Esse programa recebe como entrada determinado arquivo e gera, como saída, uma string de 128 bits em hexadecimal, como mostrado na Listagem 1.6. Na primeira 1, executa-se o comando *md5sum* e redirecionamos o seu resultado a um arquivo texto e a linha 2 apresenta o conteúdo do arquivo texto *voegol.txt*

**Listagem 1.6. Utilização do programa md5sum**

```
1 C:\Trojans> md5sum voegol.com.exe > voegol.txt
2 02e44077c7b43040683ce2434f81d563 *voegol.com.exe
```

Uma vez que se tem um identificador para o *malware*, é verificado se existe o mesmo *hash* na base de dados do analista, ou se existe similaridade com outro malware analisado anteriormente, para o caso de similaridade o capítulo 1.7 apresenta a discussão do assunto. Caso não exista padrões encontrados a análise do código é iniciada.

O primeiro passo é obter uma lista de *strings* do código. A ferramenta *strings* da *sysinternals* recebe como entrada um arquivo e produz uma lista. Caso poucas ou nenhuma string tenham sido encontradas, é provável que o malware esteja utilizando alguma técnica para ofuscamento de código, o que pode ser identificado utilizando detectores como os apresentados no capítulo 1.6.1.

A utilização do programa *strings* é exemplificado na Listagem 1.7. Aplicado o comando sobre o *malware* presente no arquivo *voegol.com.exe* onde a saída é direcionada para o arquivo *strings.txt*. Feito isso, é possível executar buscas diretamente no arquivo

*strings.txt*, por meio de uma combinação dos comandos *grep* e *cat*[19], por palavras chaves relacionadas ao caso investigado, se for um *trojan* bancário, uma busca por nomes dos bancos é recomendado. Para fins didáticos a busca se dará pelas palavras *ftp* e *http*.

**Listagem 1.7. Utilização do programa strings sobre o malware**

```
1 C:\Trojans> strings strings.com.exe > voegol.txt
2 C:\Trojans> cat voegol.txt | grep -i ftp
3 C:\Trojans> cat voegol.txt | grep -i http
```

### 1.8.5. Análise dinâmica

Análise dinâmica é uma abordagem baseada em comportamento e requer observação em tempo de execução de um artefato malicioso, utilizando um ambiente protegido e preparado para coletar informações das atividades do malware. Nessa etapa da análise, serão observadas as interações que o *malware* tem com o computador alvo. Serão analisados registros, processos, sistema de arquivos e tráfego de rede.

Com relação à execução de um código malicioso, podemos realizá-la de três maneiras distintas:

1. Execução manual simples;
2. Execução monitorada por meio de um *Instalation Monitor*;
3. Execução monitorada por meio de um *API Monitor*.

Para fins didáticos será conduzida uma análise através de execução manual simples e a qual pode ser divida em 4 etapas:

1. Preparação do ambiente;
2. Análise dos processos em execução;
3. Análise do tráfego de rede;
4. Análise do sistema de arquivos e registros.

**1. Preparação do Ambiente**: Analise de registro é uma técnica indispensável, pois permite que o analista encontre rapidamente informações sobre quais registros foram alterados durante a execução do código malicioso. Nesta etapa será utilizada a ferramenta *RegShot*[20], que possui todas as características esperadas com o objetivo de capturar dois *snapshots* do sistema, um antes e outro depois da execução do *malware*. O *RegShot* permite realizar a comparação entre os dois *snapshots* capturados. As diferenças são registradas em um arquivo texto, facilitando o tratamento e coleta da informação.

É recomendado que todos os programas necessários durante a análise já estejam em execução antes da captura do primeiro *snapshot*. Caso contrário, esses programas

[19] http://unxutils.sourceforge.net/
[20] http://sourceforge.net/projects/regshot/

podem gerar alterações nos registros que eventualmente podem ser confundidas com as próprias interações do *malware* analisado.

**2. Análise dos processos em execução**: Uma etapa importante é a identificação do modo de atuação do *malware* através da análise dos processos que serão criados durante a execução do código malicioso. A ferramenta que recomendada nesta etapa será o *ProcExp*.

O *ProcExp* é uma ferramenta poderosa, cuja função básica é verificar quais processos estão sendo executados, facilitando a identificação dos processos pais e filhos, informando, também, a localização (*path*) do arquivo que originou o processo, bem como a quantidade de memória utilizada e identificador PID (*Process Identification*). Além disso, pode-se verificar quais são as *DLLs* (*Dynamic Link Library*) que estão sendo utilizadas pelo processo, bem como *handles* abertos sendo possível realizar *dump* da memória do processo e analisar as strings encontradas, pois a ferramenta possui sua própria versão do comando *strings*.

Em algumas situações, o analista pode ter modelos semi automatizados, portanto ferramentas gráficas podem não trazer a facilidade em capturar as informações dos processos que já se conhece, dessa maneira recomenda-se a utilização da ferramenta *procdump* da *sysinternals*. A Tabela 1.6 apresenta alguns exemplos de utilização da ferramenta.

**Tabela 1.6. Principais comandos - *Procdump***

| Comando | Descrição |
|---|---|
| procdump -h voe-gol.exe malware.dmp | dump do processo chamado *voe-gol.exe* quando uma de suas janelas não responde por mais de 5 segundos |
| procdump iexplore | dump do processo chamado *iexplore* e salva para arquivo iexplore.bmp (default) |
| procdump -c 20 -s 5 -n 3 -o teste `c:\dump\teste` | Grava 3 Dumps do processo chamado *teste* quando exceder uso da CPU em 20% por cinco segundos |

Caso o analista deseje analisar o processo por meio de outro programa, o *ProcExp* permite a criação de um *dump* da região de memória na qual o processo é executado. Basta clicar com o botão direito do mouse sobre o processo desejado, escolher a opção *"create dump"* e escolher a versão que deseja realizar, Figura 1.9.

Recomenda-se atenção quanto à localização (*path*) do arquivo que originou o processo em memória, uma vez que não é necessariamente o mesmo arquivo que originou a infecção na máquina. Isso ocorre, por exemplo, quando o *malware* utiliza-se de ferramentas para instalar o artefato remotamente, conhecida como *downloader*, direcionando o sistema para o código malicioso que efetivamente interagirá com a máquina alvo. Essa tática é utilizada para diminuir o tamanho dos arquivos utilizados para a disseminação e restringe a detecção por parte de alguns antivirus, pois um *downloader* não é propriamente um artefato malicioso, mas sim um forte indicio de evidência.

Caso o arquivo que esteja sendo executado não seja o que executamos no início da análise, isso é um forte indício de que houve um conteúdo baixado da internet para a má-

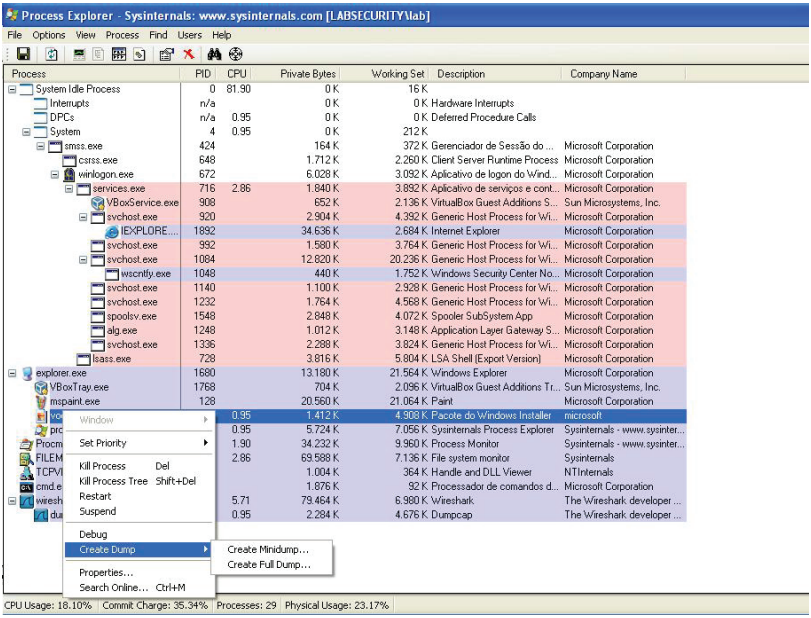


**Figura 1.9. Dump de processo em memória com ProcExp**

quina infectada, que pode ser confirmado através da análise de rede capturando os pacotes transmitidos e recebidos. Recomenda-se muita a atenção quanto ao nome verdadeiro do processo, que não é necessariamente o mesmo do arquivo executado. Esse nome, bem como o PID, serão importantes como filtros para as demais aplicações que utilizaremos.

O conjunto de ferramentas que será utilizada deve estar preparada junto com o analisador de processos *ProcExp* que em seguida, permitirá verificar quais processos são criados quando o *malware* for executado. Uma vez localizado o processo originado da execução do código malicioso, a análise propriamente dita é iniciada.

**3. Análise do tráfego de rede**: Ao se analisar o tráfego de rede originado pelas interações do *malware* procuramos por informações que auxiliem a identificação de aspectos chaves como:

- *Host* de destino dos dados capturados na máquina infectada;
- Tipos de dados capturados pelo *malware*;
- Possível *download* de atualizações ou de outros artefatos.

Nessa etapa são apresentadas técnicas para se capturar e analisar o tipo de tráfego gerado após a execução do *malware*, sendo necessário um analisador de rede. Um

dos mais populares é *Wireshark*[21], que atua como um sniffer de rede e permite coletar e analisar todo o tráfego gerado pelo artefato, de maneira a criar filtros rapidamente, identificando os protocolos e comunicações geradas, é multiplataforma e permite analisar o tráfego em tempo real ou *off-line*, possuindo suporte a decifração de diversos protocolos como *IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP* e *WPA/WPA2*. A Figura 1.10 apresenta a imagem dessa captura.

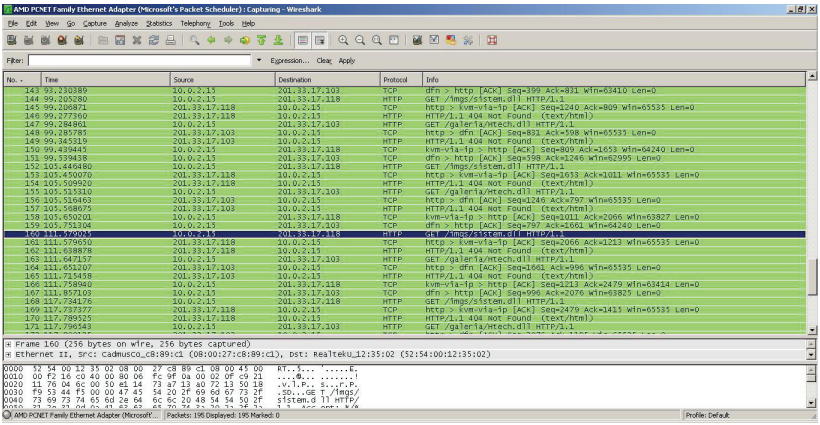


**Figura 1.10. Análise de tráfego de rede com Wireshark**

Tendo por base a captura realizada por esse programa, o analista deve procurar por pacotes que indiquem atividades do tipo:

- *http-post*: é utilizado para enviar dados a um servidor sem que haja necessidade de autenticação;
- *http-get*: é utilizado para *downloads* desde bibliotecas que serão utilizadas pelo *malware* até outros códigos maliciosos;
- *ftp*: usado tanto para receber quanto para enviar arquivos a um servidor.

Para facilitar a localização dos pacotes relevantes, é de fundamental importância a utilização dos filtros de captura. Neles podem ser especificados desde os IPs que se deseja capturar bem como os protocolos a serem exibidos ou uma combinação de ambos.

Exemplos de filtros:

- *ip.addr* == 192.168.0.102: filtra pacotes pelo endereço de IP informado;
- *icmp*: filtra os pacotes pelo protocolo desejado;

[21] `http://www.wireshark.org/`

- $ip.addr == 192.168.0.102||ftp||dns||http$: combinação de filtros pode ser realizada por meio dos operadores lógicos AND (&&) OR (||) ou comparadores igual (==) e diferente (!=).

Esse tipo de tráfego é importante pois permite determinar para onde as informações estão sendo enviadas, de maneira a facilitar a identificação pelo endereço IP de origem, caso os pacotes estejam sendo recebidos pela máquina infectada, ou de destino, caso eles estejam sendo enviados, de qualquer modo deve-se considerar que todo o tráfego é decorrente da interação das atividades do código malicioso.

É importante considerar que os dados coletados na análise estática, através das ferramentas como *strings*, bem como os obtidos pelos *dumps*, oriundos do *ProcExp*, Figura 1.9. Ao informar IPs, *URLs* (*Uniform Resource Locators*) e *PATHs*, eles podem auxiliar a nortear a busca por pacotes relevantes na captura.

Como a comunicação do malware aproveita a conexão de sua vitima, pode analisar os processos de rede que estejam ativos, neste caso recomenda-se a ferramenta *TCPView* ou *netstat*, que lista as conexões *TCP* (*Transmission Control Protocol*), *UDP* (*User Datagram Protocol*) e as portas estabelecidas com a máquina hospedeira. Saber quais conexões estão abertas ou mesmo aguardando pode ser útil a fim de se identificar quais *hosts* estão interagindo com a máquina infectada. A Listagem 1.8 mostra o resultado da execução *TCPView*.

**Listagem 1.8. Resultado do programa *TCPView***

```
1  lsass.exe    736   UDP   labsecurity isakmp    *  *
2  lsass.exe    736   UDP   labsecurity 4500  *  *
3  svchost.exe 984   TCP   labsecurity epmap labsecurity 0  LISTENING
4  svchost.exe 1076  UDP   labsecurity ntp   *  *
5  svchost.exe 1204  UDP   labsecurity 1900  *  *
6  svchost.exe 1076  UDP   labsecurity ntp   *  *
7  svchost.exe 1136  UDP   labsecurity 1026  *  *
8  svchost.exe 1204  UDP   labsecurity 1900  *  *
9  System    4       TCP   labsecurity microsoft-ds   labsecurity 0
       LISTENING
10 System    4       TCP   labsecurity netbios-ssn labsecurity 0  LISTENING
11 System    4       UDP   labsecurity netbios-ns  *  *
12 System    4       UDP   labsecurity netbios-dgm *  *
13 System    4       UDP   labsecurity microsoft-ds   *  *
```

A primeira coluna diz respeito ao processo que está ativo seguido da porta de comunicação e o protocolo utilizado, as seguintes mostram os endereços resolvidos tanto de origem como destino conjuntamente com a porta de comunicação. Os serviços que se encontram em *LISTENING*, estão aguardando uma solicitação para que a conexão seja fechada, devem ser acompanhados com cuidado, pois alguns *malwares* instalam processos que são interpretados como serviços do sistema operacional.

Após a identificação do comprometimento, é de responsabilidade da equipe de resposta a incidentes (CSIRT), contatar o servidor remoto para onde as informações estão sendo enviadas, de modo a conter o incidente e neutralizar outras fontes de infecção que não serão tratadas, pois a partir do momento em que o incidente é neutralizado e sua

fonte de informação e conexão é suspensa, todo o sistema deixa de estar ativo, portanto o incidente passa a ser considerado erradicado.

**4. Análise do sistema de arquivos e registros**: Nessa etapa é dada uma atenção especial a análise das interações do *malware* com o sistema de arquivos e registros da máquina infectada. Para isso, a ferramenta utilizada será o *ProcMon*, Figura 1.11. A utilização dessa ferramenta será complementar a analise obtida através do *RegShot*. O *ProcMon* permite visualizar, em tempo real, quais arquivos e registros estão sendo acessados ou modificados e qual operação está sendo realizada sobre eles. O resultado obtido é ordenado por um número sequencial e cada entrada possui, também, um *timestamp*. Para facilitar a sua utilização, a ferramenta provê o recurso de filtragem pelo nome do processo ou seu PID. Dessa forma, basta adicionar ao filtro o nome do processo sendo analisado, assim todas as alterações de arquivo ou registro que serão capturadas dirão respeito a tal processo.

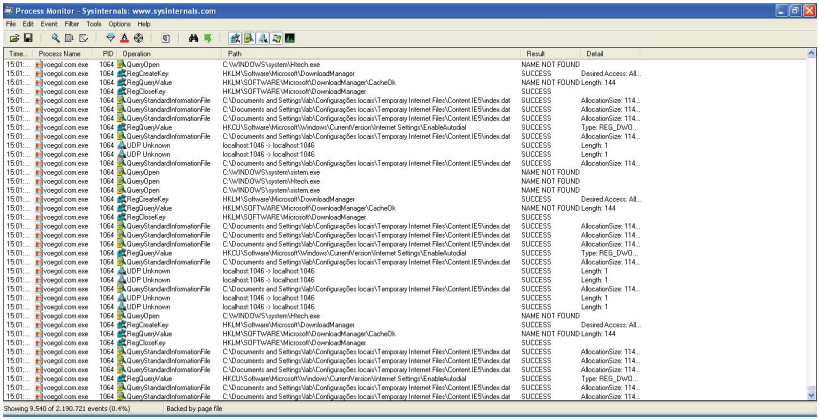


**Figura 1.11. Análise sistema de arquivos e registros com ProcMon**

Como apresentado na seção 1.8.4, calcular o *hash* do código malicioso a ser analisado é de extrema importância, entre outros fatores, para verificar se o *malware* já foi analisado ou se alterou seu próprio código durante a execução. Uma técnica utilizada para realizar tal verificação é a do *rehashing*. Essa técnica consiste em, cada vez que o artefato analisado for alterado, como quando o arquivo que originou o processo em memória não é o mesmo que executamos no início da análise, calcular o *hash* do novo artefato e comparar com o primeiro [Aquilina et al. 2008].

Alterar seu próprio código fonte é uma técnica que pode ser útil burlar os mecanismos de detecção dos antivírus.

## 1.9. Modelo de relatório de análise

Nesta etapa será apresentado um modelo de relatório de análise, que servirá para documentar e registrar todos os detalhes na condução da investigação e propostas de mitigação

de impacto, visando a melhoria dos processos de segurança.

**Relatório 1.9. Exemplo de relatório de uma análise de malware**

```
[1] Data do Report:
13/08/2009 11:48:23

[2] Tipo de Ataque:
Pop-up

[3] Incidente:
2009-08-12_BO3__[http_www.groupautounion.si_images_images_Voegol.php]

[4] URL:
http://www.groupautounion.si/images/images/Voegol.php

[5] Analisado por:
John Doe

[6] Hash (MD5):
02e44077c7b43040683ce2434f81d563 voegol.com.exe

[7] Arquivo criado/acessado:
c:\Documents and Settings\kurumin\Configuracoes locais\Historico\
    History.IE5\MSHist012009081320090814
c:\Documents and Settings\kurumin\Configuracoes locais\Historico\
    History.IE5\MSHist012009081320090814\index.dat
c:\Documents and Settings\kurumin\Configuracoes locais\Temp\~DFD33A.tmp
c:\WINDOWS\control.ctr
c:\WINDOWS\system\Htech.exe
c:\WINDOWS\system\sistem.exe

[8] Gerenciador de Tarefas:
c:\windows\system32\userinit.exe
c:\windows\explorer.exe
c:\windows\system32\vboxservice.exe
c:\windows\system\sistem.exe

[9] Registry:

   C:\WINDOWS\system32\userinit.exe

   Aplicativo de logon Userinit

   Microsoft Corporation

   c:\windows\system32\userinit.exe
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
   Explorer.exe
   Windows Explorer
   Microsoft Corporation
   c:\windows\explorer.exe
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
   VBoxService
   VirtualBox Additions Service
   innotek GmbH
```

```
51      c:\windows\system32\vboxservice.exe
52      msne
53      c:\windows\system\sistem.exe
54
55  [10] Bancos Alvo:
56  BRADESCO
57
58  [11] Acao via Rede:
59  FTP
60
61  [12] Dados de rede (Destino):
62  ciroinfect@201.7.184.42
63  FTP= 201.7.184.42:21 LOGIN= ciroinfect SENHA= flor123.
64  http://200.106.147.115/recadastramento/stats.php
65  http://200.106.147.115/cartao2/finaliza.php
66
67  [13] Dados de rede (Origem):
68  Nao identificado.
69
70  [14] Observacoes:
71
72  Protecoes atacadas:
73  BRADESCO
74
75  URLs acessadas pelo trojan:
76  kaos.local - - [13/Aug/2009:11:39:43 -0300] "GET http://voegol.com.br/
        HTTP/1.1" - - "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
        5.1; SV1)"
77  kaos.local - - [13/Aug/2009:11:39:43 -0300] "GET http://
        oficinadacostura.com.br/imgs/sistem.dll HTTP/1.1" - - "-" "Mozilla
        /4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
78  kaos.local - - [13/Aug/2009:11:39:44 -0300] "GET http://www.voegol.com.
        br/Paginas/home.aspx HTTP/1.1" - - "-" "Mozilla/4.0 (compatible;
        MSIE 6.0; Windows NT 5.1; SV1)"
```

## 1.10. Considerações Finais

Empresas têm demonstrado grande preocupação frente aos desafios em manter seus sistemas seguros. Entender a taxonomia de um ataque, sistematizando os processos de responsabilidade de um grupo de resposta a incidentes e proteger os ativos da organização, é uma necessidade crescente e que exige preparo constante do profissional responsável pela segurança da informação.

Atividades que envolvem códigos maliciosos podem ser exploradas para criar verdadeiros exércitos de soldados do crime. Para enfrentar esse desafio, é preciso entender não só as ferramentas utilizadas para criar e distribuir esses códigos, mas também é preciso interpretar o pensamento do próprio atacante, buscando estar à frente dele para que seja possível a detecção e neutralização desses eventos.

Quando um incidente de segurança é reportado, a atuação da equipe de resposta deve ser precisa. A tentativa de tratar cada código malicioso isoladamente é uma estratégia fadada ao insucesso. Por outro lado, é preciso ter a capacidade de distinguir e identificar cada artefato malicioso, e só então colocar em prática a estratégia de segurança,

integrando contra-medidas e medidas de proteção relativas ao conjunto dos *malwares* que se apresentam.

Neste curso, foram apresentados os conceitos e os métodos para que o tratamento de incidentes que envolvem *malwares* seja realizado com fortes bases técnicas e que instrumentalize o profissional com as ferramentas que permitam o desenvolvimento do trabalho de análise. Entender o comportamento de um *malware* é o primeiro passo para identificar seu autor e então neutralizar suas ações. Isoladamente, o ato de detectar o ataque constitui-se como um processo de caráter apenas reativo. Entretanto, é necessário ter a capacidade de agir preventivamente, os seja, de criar e manter sistemas seguros que contemplem as proteções contra os ataques já conhecidos, o que requer a contínua análise da estrutura e funcionamento dos *malwares*.

## Referências

[Abbas et al. 2006] Abbas, A., Saddik, A., and Miri, A. (2006). A Comprehensive Approach to Designing Internet Security Taxonomy. *2006 Canadian Conference on Electrical and Computer Engineering*, pages 1316–1319.

[ABNT 2005] ABNT (2005). ISO IEC 27001 Tecnologia da informacao Tecnicas de seguranca Sistemas de gestao de segurança da informacao Requisitos.

[Aquilina et al. 2008] Aquilina, J. M., Casey, E., and Malin, C. H. (2008). *Malware Forensics: Investigating and Analyzing Malicious Code*. Syngress Publishing, 1 edition.

[Binsalleeh et al. 2009] Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., and A (2009). On the Analysis of the Zeus Botnet Crimeware Toolkit. In *Proceedings of the Eighth Annual Conference on Privacy, Security and Trust (PST'2010)*, Ottawa, ON, Canada. IEEE Press.

[Brezinski and Killalea 2002] Brezinski, D. and Killalea, T. (2002). Guidelines for Evidence Collection and Archiving. RFC 3227, Internet Engineering Task Force - IETF.

[Brownlee and Guttman 1998] Brownlee, N. and Guttman, E. (1998). Expectations for Computer Security Incident Response. RFC 2350.

[Chen and Abu-Nimeh 2011] Chen, T. M. and Abu-Nimeh, S. (2011). Lessons from stuxnet. *IEEE Computer Society*, 44:91–93.

[Dube et al. 2010] Dube, T., Raines, R., Peterson, G., Bauer, K., Grimaila, M., and Rogers, S. (2010). Malware Type Recognition and Cyber Situational Awareness. *2010 IEEE Second International Conference on Social Computing*, pages 938–943.

[Finjan Research Center 2009] Finjan Research Center (2009). Cybercrime Intelligence: Cybercriminals use Trojans & Money Nules to Rob Online Banking Accounts. Number 3 in 1, pages 1–9. Finjan Malicious Code Research Center, Finjan Malicious Code Research Center.

[Jones 2001] Jones, P. (2001). US secure hash algorithm 1 (SHA1) RFC 3174.

[Kent et al. 2006] Kent, K., Chevalier, S., Grance, T., and Dang, H. (2006). *Guide to integrating forensic techniques into incident response*. National Institute of Standards and Technology, 800-86 edition.

[Kornblum 2006] Kornblum, J. D. (2006). Identifying almost identical files using context triggered piecewise hashing. In *Proceedings of the Digital Forensic Workshop*, pages 91–97.

[Martins et al. 2010] Martins, V., Grégio, A., Afonso, V., and Fernandes, D. (2010). xFile: Uma Ferramenta Modular para Identificação de Packers em Executáveis do Microsoft Windows. In SBC, editor, *SBSeg 2010*, pages 31–40, Fortaleza - CE.

[Mell and karen Kent 2005] Mell, P. and karen Kent, N. J. (2005). *Guide to Malware Incident Prevention and Handling Recommendations of the National Institute of Standards and Technology*, volume 800-83. Department of Homeland Security, Gaithersburg, 800-83 edition.

[Mieres 2009] Mieres, J. (2009). Analysis of an attack of malware web-based. Technical report, Malware Intelligence.

[Prosise et al. 2003] Prosise, C., Mandia, K., and Pepe, M. (2003). *Incident Response & Computer Forensics, 2nd Ed.* McGraw-Hill, Inc., New York, NY, USA, 2 edition.

[Rivest 1992] Rivest, R. (1992). The MD5 message-digest algorithm (RFC1321).

[Steding-Jessen 2008] Steding-Jessen, K. (2008). *Uso de Honeypots para o estudo de Spam e Phishing (Doutorado)*. Tese, INPE - Instituto Nacional de Pesquisas Espaciais.

## 1.11. Anexo I: Lista de Packers, Unpackers, Cryptors

**Tabela 1.7. Ferramentas: Packers, Unpackers, Cryptors[Aquilina et al. 2008]**

| Tipos | URL's |
|---|---|
| Armadillo | www.siliconrealms.com/armadillo_engine.shtml |
| ASPack | www.aspack.com |
| BeRoEXEPacker | bero.0ok.de/blog/projects/beroexepacker/ |
| CExe | www.scottlu.com/Content/CExe.html |
| Exe32pack | www.steelbytes.com |
| EXECryptor | www.strongbit.com/execryptor.asp |
| eXPressor | www.expressor-software.com/ |
| FSG | www.exetools.com/protectors.htm |
| Krypton | programmerstools.org/taxonomy/term/17?from=20 |
| MEW | www.exetools.com/protectors.htm |
| Molebox | www.molebox.com/ |
| Morphine | www.exetools.com/protectors.htm |
| NeoLite | www.exetools.com/protectors.htm |
| Obsidium | www.obsidium.de/show.php?download |
| PEBundle | www.bitsum.com/pebundle.asp |
| PECompact | www.bitsum.com/ |
| PE Crypt 32 | www.opensc.ws/asm/1071-pecrypt.html |
| PELock | http//pelock.com/page.php?p=pelock#download |
| PEPack | www.dirfile.com/freeware/pepack.htm |
| PESpin | pespin.w.interia.pl/ |
| Petite | www.exetools.com/protectors.htm |
| PKLite32 | pklite32.qarchive.org/ |
| PolyCryptPE | www.cnet.com.au/downloads/0,239030384,10420366s,00.htm |
| RLPack | rlpack.jezgra.net |
| SFX | www.exetools.com/protectors.htm |
| Shrinker32 | www.exetools.com/protectors.htm |
| Themida | www.oreans.com/downloads.php |
| UPX | upx.sourceforge.net/ |
| yoda | yodap.cjb.net/ |