# Análise Forense de Intrusões em Sistemas Computacionais: Técnicas, Procedimentos e Ferramentas

**Marcelo Abdalla dos Reis**
Instituto de Computação
Universidade Estadual de Campinas
13083-970 Campinas - SP
http://www.ic.unicamp.br/ra000504
*marcelo.reis@ic.unicamp.br*

**Paulo Lício de Geus**
Instituto de Computação
Universidade Estadual de Campinas
13083-970 Campinas - SP
http://www.ic.unicamp.br/paulo
*paulo@ic.unicamp.br*

RESUMO

*A tecnologia dos computadores está envolvida em um número crescente de atividades ilícitas, o que requer um maior entendimento de como se obter e utilizar evidências eletrônicas armazenadas em computadores. Este trabalho apresenta uma discussão detalhada sobre a investigação forense de intrusões em sistemas computacionais, tendo como objetivo principal fornecer uma descrição completa sobre onde, como e o quê procurar em um sistema invadido.*

ABSTRACT

*The computer tecnology is involved in a growing number of illegal activities. This situation requires a major understanding on how to obtain and use digital evidence stored in computers. This work presents a detailed discution about forensic investigation of intrusions in computer systems, providing a complete description about where, how and what to look for in a compromised system.*

## 1 Introdução

As últimas décadas foram marcadas pela integração dos computadores no modo de vida das pessoas. Infraestruturas básicas da sociedade, como redes financeiras, sistemas de comunicação, estações de energia e sistemas de saúde, dependem todas de sistemas computacionais para seu funcionamento eficiente e confiável. Além disso, é crescente o número de indivíduos que utilizam computadores pessoais por conveniência, educação e entretenimento. A conectividade oferecida pela Internet também introduziu uma série de novas facilidades no dia a dia das pessoas, como o correio eletrônico e a *World Wide Web*, permitindo o acesso anônimo a quase todo tipo de informação ou sistema.

Não é de se espantar o fato de que essa revolução computacional tenha atingido também o mundo do crime. A tecnologia dos computadores está envolvida em um número crescente de

atividades ilícitas. Além de serem utilizados como ferramentas para a consumação de alguns tipos de crimes (como, por exemplo, invasão de sistemas, disseminação de pornografia infantil e fraude), os computadores podem conter evidências[1] relacionadas com qualquer tipo de atividade ilícita, incluindo homicídio e estupro.

O aumento dramático em crimes relacionados com computadores requer um maior entendimento de como se obter e utilizar evidências eletrônicas armazenadas em computadores. Tal entendimento pode ser alcançado através dos conceitos e metodologias da forense computacional.

A forense computacional compreende a aquisição, preservação, identificação, extração, restauração, análise e documentação de evidências computacionais, quer sejam componentes físicos ou dados que foram processados eletronicamente e armazenados em mídias computacionais [14, 17].

O propósito do exame forense é a procura e extração de evidências relacionadas com o caso investigado, que permitam a formulação de conclusões acerca da infração [9]. Existem duas abordagens no que diz respeito ao objetivo final da análise forense. Na primeira, a análise forense busca obter informação de valor probante (coerente com as regras e leis das evidências e admissível em uma corte de justiça) a ser utilizada em um processo criminal. Na segunda abordagem, o exame é realizado dentro de uma corporação com o objetivo de determinar a causa de um incidente e assegurar que o mesmo não ocorra novamente, sem que haja preocupação com formalidades legais.

Mesmo que não haja intenção de se instituir um processo criminal, toda investigação deve considerar como prática padrão a utilização de metodologias e protocolos que garantam sua possível aceitação em uma corte de justiça [14]. Tratar todo caso com a formalidade de um processo criminal ajuda a desenvolver bons hábitos de investigação.

Nesse sentido, existem alguns aspectos chave que constituem as etapas do processo de análise forense de um sistema computacional [5]:

- coleta de informações (ou *information gathering*);
- reconhecimento das evidências;
- coleta, restauração, documentação e preservação das evidências encontradas;
- correlação das evidências;
- reconstrução dos eventos;

Toda informação relevante deve ser coletada para análise e, conforme as evidências digitais são encontradas, elas devem ser extraídas, restauradas quando necessário (evidências danificadas ou cifradas, por exemplo), documentadas e devidamente preservadas. Em seguida, as evidências encontradas podem ser correlacionadas, permitindo a reconstrução dos eventos relacionados ao ato ilícito. Muitas vezes a análise das evidências (correlação e reconstrução) resulta na descoberta de novas informações, formando um ciclo no processo de análise forense

[1] Devido ao carater substancialmente técnico deste trabalho, a linguagem adotada foi destituída de características jurídicas. Nesse sentido, o termo "evidência" pode ser interpretado, sob o enfoque criminalístico, tanto como vestígio quanto indício, condicionado ao contexto em que o termo é utilizado.

[5]. Devido a sua importância, o *framework* geral do processo de análise forense é discutido detalhadamente na seção 5.

A forense computacional é uma área de pesquisa relativamente recente, entretanto, é crescente a necessidade de desenvolvimento nesse campo, uma vez que a utilização de computadores em atividades criminosas tem se tornado uma prática comum. Os computadores atingiram crimes tradicionais, como extorsão, roubo e tráfico de drogas, e também originaram uma nova classe de crimes, conhecidos como "crimes da Internet", como ataques de negação de serviço, invasão de sistemas e disseminação de vírus de computador.

O escopo deste trabalho restringe-se à investigação de uma categoria de crime onde o computador é o alvo, comumente denominada de intrusão de sistemas. Com o advento da Internet, ataques remotos a sistemas computacionais tornaram-se mais comuns, tirando vantagem da crescente complexidade e vulnerabilidade dos serviços de rede [14]. O aumento na sofisticação e frequência com que esses ataques têm ocorrido representa um desafio crescente para os encarregados de investigar e responder a esses incidentes.

Este trabalho apresenta uma discussão detalhada sobre a investigação forense de intrusões em sistemas computacionais. Embora o enfoque principal seja dado à análise de invasões em ambientes Linux, muitos dos conceitos e técnicas apresentados neste trabalho podem ser aplicados à investigação forense de outros tipos de crimes e extendidos a outras plataformas, como o Windows e outros sistemas da família UNIX (Solaris e BSD, por exemplo). O objetivo principal deste trabalho é fornecer uma descrição detalhada sobre onde, como e o quê procurar em um sistema computacional invadido. Para tal, são apresentadas diversas técnicas, ferramentas e procedimentos, com exemplos práticos que facilitam a compreensão.

A organização deste trabalho é apresentada como segue. A seção 2 discute brevemente os objetivos e métodos de operação dos invasores. Na seção 3 são apresentadas as principais fontes de informação de um sistema computacional, evidenciando, em cada uma, os métodos de extração das informações e as evidências digitais mais comumente encontradas. Algumas questões sobre o correlacionamento de evidências são abordadas na seção 4 e o *framework* do processo de investigação forense é discutido na seção 5. A seção 6 aborda a questão do conjunto de ferramentas do investigador e, por fim, algumas conclusões são apresentadas na seção 7.

# 2 *Modus operandi*

Novas formas de invadir e interferir com computadores são desenvolvidas a cada dia. Com o mínimo de conhecimento de redes de computadores, praticamente qualquer um pode obter gratuitamente na Internet e utilizar ferramentas para invadir um sistema computacional e provocar todo tipo de estrago. O número de ataques externos a uma organização tem crescido consideravelmente, equiparando-se à quantidade de ataques cometidos por indivíduos de dentro da própria organização [5]. A intrusão de sistemas tem sido considerada um risco à segurança nacional em muitos países, de modo que a compreensão acerca dos objetivos e métodos empregados nesses incidentes tem se tornado alvo de muitos estudos [5, 14].

*Modus operandi* é um termo em latin que significa "método de operação". Entender a motivação e o comportamento de um intruso é um ponto chave para orientar a investigação, pois essa compreensão fornece pistas sobre onde e o quê procurar durante a análise forense [5]. Quanto maior a consciência acerca dos objetivos e *modus operandi* de um atacante, maior o

preparo do investigador para analisar e responder a um incidente [14].

A invasão de sistemas computacionais ocorre com finalidades diversas, podendo ser destacadas as seguintes:

- obtenção de informações (roubo de segredos, números de cartões de crédito, senhas e outros dados relevantes ao intruso);
- promover algum estrago ("pichação" de *sites*, destruição de informações e paralisação do sistema, por exemplo);
- utilização dos recursos do sistema (repositório de dados, disseminação de ataques distribuídos, provimento de serviços, por exemplo);

Dependendo da finalidade e da habilidade, o *modus operandi* de um invasor pode sofrer algumas variações. Entretanto, os passos tomados pelo atacante para comprometer um sistema computacional podem ser generalizados como segue [5, 14]:

- identificação do alvo;
- busca de vulnerabilidades no alvo (*probing*);
- comprometimento inicial;
- aumento de privilégio;
- tornar-se "invisível" (*stealth*);
- reconhecimento do sistema (*reconnaissance*);
- instalação de *back doors*;
- limpeza dos rastros;
- retorno por uma *back door*, inventário e comprometimento de máquinas vizinhas;

A primeira atitude do atacante é a escolha de um alvo em potencial. Após a localização, o atacante começa a reunir informações sobre o sistema alvo a fim de identificar vulnerabilidades no sistema operacional ou serviços de rede disponíveis. Se o invasor ainda não possui uma combinação de usuário e senha válida para o sistema alvo, ele utiliza métodos como *sniffing* e adivinhação de senhas, engenharia social ou *scanning* para encontrar um ponto de entrada.

Uma vez encontrado um ponto de entrada (conta de usuário ou vulnerabilidade em um serviço, por exemplo) o invasor realiza o comprometimento inicial do sistema. Essa primeira intrusão geralmente provoca muito "barulho", especialmente se o sistema alvo estiver devidamente guarnecido, e costuma ocorrer quando ninguém está presente para "ouvir" [14]. Tentativas de adivinhar senhas criam um número incomum de registros de *logon* falhos, comprometimento de aplicativos através de *buffer overflow* geralmente ficam registrados nos arquivos de log ou geram *core files*, e mensagens de advertência são produzidas em decorrência das várias tentativas de se invadir o sistema [14].

Depois que o atacante ganha acesso ao sistema, ele busca por privilégios irrestritos (conta de administrador ou *root*) – assumindo que o comprometimento inicial já não lhe forneceu acesso à conta de *root*. O invasor transfere programas maliciosos (conhecidos por *exploits*) para o sistema e tenta explorar vulnerabilidades que possam fornecer o acesso de *root*. Com acesso ilimitado, o atacante procura remover traços de sua presença, tornando-se "invisível", através da instalação de *rootkits* e *trojan horses*.

Quando o invasor obtém acesso de *root* e garante sua "invisibilidade", ele executa uma verdadeira varredura no sistema [14]. O atacante procura saber o quanto sua presença perturba o sistema invadido e, por conseguinte, pode ser descoberta (analisando a configuração de log). Em seguida, ele investiga as medidas de segurança implementadas no sistema invadido – em alguns casos, o atacante até corrige vulnerabilidades existentes para impedir que outro invasor faça uso do sistema.

Após compreender as configurações do sistema, o atacante instala *back doors* para facilitar seu retorno e apaga os rastros deixados por sua presença no sistema. Utilizando uma *back door*, o invasor retorna de maneira mais discreta que o comprometimento inicial e faz um inventário acerca das informações existentes na máquina invadida e dos potenciais alvos da vizinhança.

A habilidade do invasor em executar o *modus operandi* descrito anteriormente pode ser fundamental para o processo de análise forense, pois a quantidade de evidências deixadas depende diretamente do nível de conhecimento do atacante [14]. Para ilustrar essa relação, é possível classificar a habilidade do invasor em quatro classes, de acordo com [14]: *Clueless*, *Script Kiddie*, *Guru* e *Wizard*. A tabela 1 apresenta a relação entre a habilidade do invasor e a quantidade de evidências deixadas.

| **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 *back doors* para um acesso futuro |
| *Wizard* | Possui um g rande conhecimento do funcionamento interno de um sistema. Capaz de manipular *hardware* e *software* | Praticamente não deixa evidências úteis. Pode comprometer totalmente o sistema |

Tabela 1: Relação entre a habilidade do invasor e a quantidade de evidências deixadas.

# 3 Evidências digitais

Um dos princípios fundamentais da forense é o Princípio da Troca de Locard [5]. De acordo com esse princípio, qualquer um, ou qualquer coisa, que entra em um local de crime leva consigo algo do local e deixa alguma coisa para trás quando parte [5]. No mundo virtual dos computadores, o Princípio da Troca de Locard ainda é válido (ou pelo menos parte dele): onde quer que o intruso vá ele deixa rastros [27]. Tais rastros podem ser extremamente difíceis ou praticamente impossíveis de serem identificados e seguidos, mas eles existem [27]. Nesses casos o processo de análise forense pode tornar-se extremamente complexo e demorado, necessitando do desenvolvimento de novas tecnologias para a procura de evidências.

O termo evidência digital refere-se a toda e qualquer informação digital capaz de determinar que uma intrusão ocorreu ou que provê alguma ligação entre a intrusão e as vítimas ou entre a intrusão e o atacante[2].

A evidência digital não deixa de ser um tipo de evidência física, embora seja menos tangível [5]. Ela é composta de campos magnéticos e pulsos eletrônicos que podem ser coletados e analisados através de técnicas e ferramentas apropriadas. Entretanto, a evidência digital possui algumas características próprias:

- ela pode ser duplicada com exatidão, permitindo a preservação da evidência original durante a análise;
- com os métodos apropriados é relativamente fácil determinar se uma evidência digital foi modificada;
- por outro lado, a evidência digital é extremamente volátil, podendo ser facilmente alterada durante o processo de análise;

A busca de evidências em um sistema computacional constitui-se de uma varredura minuciosa nas informações que nele residam, sejam dados em arquivos ou em memória, "deletados" ou não, cifrados ou possivelmente danificados.

Dan Farmer e Wietse Venema introduziram um conceito denominado de ordem de volatilidade [10]. Tal conceito determina que o tempo de vida de uma evidência digital varia de acordo como o local onde ela está armazenada. As principais fontes de informação de um sistema computacional são apresentadas, na ordem descendente de volatilidade, como segue [3, 10, 14]:

- dispositivos de armazenagem da CPU (registradores e *caches*);
- memória de periféricos (memória de vídeo, por exemplo);
- memória principal do sistema;
- tráfego de rede (pacotes em trânsito na rede);
- estado do sistema operacional (como, por exemplo, estado das conexões de rede e dos processos em execução, usuários logados e configurações do sistema);

[2]Essa definição foi adaptada para o contexto de intrusão de sistemas a partir da definição apresentada em [5].

- dispositivos de armazenagem secundária

Quanto maior a volatilidade de uma informação, mais difícil se torna sua extração e menos tempo há para capturá-la. O simples ato de observar informações altamente voláteis pode alterá-las, de modo que é pouco provável que alguém possa utilizar o conteúdo dos registradores da CPU, por exemplo [14]. Entretanto, informações voláteis como o conteúdo da memória principal do sistema, o tráfego de rede e o estado do sistema operacional podem ser capturadas com relativa facilidade e podem conter pistas valiosas a respeito de intrusões em andamento.

O detalhamento de cada fonte de informação, bem como das possíveis evidências encontradas em cada uma e das técnicas utilizadas para extração das informações, é apresentado com segue[3].

## 3.1 Dispositivos de armazenagem da CPU

As informações contidas nos registradores da CPU são de mínima utilidade e sua captura é impraticável [14]. As *caches* podem conter informações que ainda não foram atualizadas na mémoria principal do sistema, entretanto sua captura também pode ser impraticável [14].

## 3.2 Memória de periféricos

Muitos dispositivos como *modems*, *pagers*, aparelhos de fax e impressoras, contêm memórias que podem ser acessadas e salvas [13]. Nelas podem estar armazenadas informações que não mais residem no sistema analisado, como documentos e mensagens de texto ou números de fax e telefone.

A memória de vídeo também pode prover informação útil no caso do invasor estar utilizando um console ou terminal gráfico, de modo que a tela corrente pode ser capturada e reproduzida [14]. Além do uso de fotografias, o comando `xwd`, do sistema X Windows, pode ser usado para capturar uma janela particular ou toda a tela. O comando `xwd` necessita de acesso de *root* e deve ser executado a partir de outro terminal virtual ou remotamente para não alterar a tela que se deseja capturar:

```
# xwd -display localhost:0 -root > screen.xwd
```

A opção `-display localhost:0` serve para identificar a máquina e o número do terminal gráfico de onde se deseja capturar a tela (no formato *nome ou endereço IP*:*número do teminal*). E a opção `-root` especifica que toda a tela deve ser capturada. O comando `xwd` gera sua saída em um formato especial que pode ser salvo em um arquivo. Esse arquivo pode ser visualizado através do utilitário `xwud`:

```
# xwud -in screen.xwd
```

[3]As técnicas apresentadas nesta seção apenas ilustram como determinadas informações podem ser obtidas na máquina analisada, não havendo preocupação com o destino da saída dos comandos apresentados. Maiores detalhes sobre o processo de coleta de informações são discutidos na seção 5. As ferramentas utilizadas nas diversas explanações e apresentadas nos exemplos são compatíveis com a plataforma Linux (plataforma adotada para esta pesquisa científica). Algumas ferramentas destinadas a outras plataformas, como DOS e Windows, são apresentadas na seção 6. Com relação aos exemplos, o caracter "\" é utilizado para indicar que uma determinada linha continua na linha seguinte.

A opção `-in` serve para especificar o arquivo contendo a saída do comando `xwd`. Vários utilitários, como `fbm`, `pbmplus` e `ImageMagick`, podem ser utilizados para converter o formato XWD para outros mais comuns, como TIFF e GIF [14].

## 3.3 Memória principal do sistema

A memória principal contém todo tipo de informação volátil, como, por exemplo, informações dos processos que estão em execução, dados que estão sendo manipulados e muitas vezes ainda não foram salvos no disco e informações do sistema operacional [24, 26]. Tais informações podem ser facilmente capturadas por meio de *dumps* da memória, pela geração de *core files* ou pela interface provida pelo diretório `/proc` [14].

Ao fazer a captura das infomações da memória (processo chamado de *dump* da memória), uma porção da mesma será alterada [15]. Quando o utilitário usado para fazer o *dump* é executado, o sistema operacional aloca uma área da memória para o processo em execução. Portanto não é possível verificar se as informações capturadas são exatamente iguais às originais [14].

O *dump* da memória pode ser feito através do comando `dd`:

```
# dd bs=1024 < /dev/mem > mem.dump
# dd bs=1024 < /dev/kmem > kmem.dump
```

É importante lembrar que tudo no sistema operacional UNIX é tratado como arquivo, de modo que a memória principal do computador e a memória virtual do *kernel* são acessíveis através dos arquivos de dispositivo (*deviceo*

processo tiver permissão de escrita no diretório onde se encontra o arquivo executável relativo ao processo, se o processo não redefiniu a ação a ser tomada ao receber o sinal e se o tamanho do *core file* não exceder o limite máximo imposto para o usuário dono do processo [12].

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 [12, 16]. Ataques de *buffer overflow* geralmente causam a geração de *core files* e alguns *exploits* usados por atacantes geram propositadamente *core dumps* de programas que manipulam senhas, de modo que tais senhas podem ser resgatadas do arquivo gerado [14, 27].

O comando `file` pode ser usado para determinar o programa relacionado ao *core file*, bem como o sinal que causou sua geração [14]:

```
# file core
core: ELF 32-bit LSB core file of 'teste' (signal 11), Intel 80386
```

No exemplo anterior, o *core dump* originou-se do programa `teste` que teve sua execução interrompida pelo sinal 11 (SIGSEGV), indicando uma possível falha de segmentação. Pode-se usar o comando `strings` para identificar os arquivos que o programa estava referenciando, ou ainda fazer uma análise mais profunda com a ajuda de programas de depuração, como `adb` ou `gdb` [12, 14]:

```
# strings -a core | more
# gdb -c core
```

Os *core files* podem revelar algumas evidências como, por exemplo:

- o programa origem do *core file* é suspeito (podendo ser um programa desconhecido; um comando conhecido, mas que não deveria ter sido terminado; um programa com nome que faz alusão a um código hostil, como *sniffer* ou *cracker*; ou ainda um programa que manipula algum tipo de senha);
- o sinal recebido pelo processo é suspeito (o sinal pode indicar um *bug* no programa);
- o programa origem do *core file* faz referência a arquivos suspeitos (arquivos com nomes suspeitos ou em diretórios suspeitos, ou arquivos que o programa não deveria estar acessando, por exemplo);

Além dos *core files*, existe outra fonte de informação bastante semelhante, denominada de *crash dump*, que contém uma imagem da memória do sistema no momento em que uma falha inesperada acontece (denominada de *crash*) [16, 27]. Quando um sistema UNIX falha, isto é, ocorre um *crash* do sistema, ele pode criar um arquivo denominado *crash dump*, para ajudar os especialistas a determinar a causa da falha [27]. Esse arquivo, gerado pela função `panic()`[4], contém informações sobre o programa que causou a falha, além de outros dados que estavam na memória, como senhas por exemplo [27]. Alguns atacantes desenvolvem programas que causam um *crash* do sistema e examinam o *crash dump* resultante à procura de senhas.

[4] A função `panic()` não pode ser invocada por um aplicativo [27].

O *crash dump* é uma espécie de "caixa preta" do sistema – todo tipo de informação que estava na memória no momento da falha será salva nele. Assim como no caso dos *core files*, os comandos `strings` e `adb` podem ser usados para acessar as informações contidas no *crash dump* [27]. Dentre essas informações podem estar mensagens de log que não puderam ser gravadas nos respectivos arquivos em decorrência do *crash* do sistema. Quando uma mensagem de log é gerada, ela é colocada no *buffer* de mensagens da memória antes de ser gravada no disco, de modo que a falha do sistema impede que a mensagem seja salva em arquivo [27]. Além disso, algumas versões do sistema UNIX permitem a execução de comandos de "status" (como, por exemplo, `ps`, `netstat`, `nfsstat` e `arp`) sobre o *crash dump*[5], de modo que é possível resgatar, por exemplo, informações sobre os processos que estavam executando no momento da falha do sistema e as conexões de rede que foram estabelecidas [27]:

```
# ps -[opções] imagem_do_kernel crash_dump
# netstat -[opções] imagem_do_kernel crash_dump
# nfsstat -[opções] imagem_do_kernel crash_dump
# arp -[opções] imagem_do_kernel crash_dump
```

Outra forma de acessar a memória é através do pseudo-arquivo `/proc/kcore`, que representa a memória física do sistema no formato de um *core file*, podendo ser examinado com o auxílio dos comandos `strings` e `gdb`. Maiores detalhes sobre a interface provida pelo diretório `/proc` são apresentados adiante e informações adicionais sobre análise de *crash dump* podem ser encontradas em [8].

## 3.4 Tráfego de rede

A captura do tráfego de rede pode ser comparada à gravação em vídeo de um crime. A partir dos datagramas capturados, é possível recontruir a comunicação entre o atacante e a máquina alvo, de modo que uma sequência de eventos pode ser estabelecida e comparada com as outras evidências encontradas na máquina invadida [6].

Existem vários programas que podem ser usados para capturar o tráfego de rede, comumente denominados de *sniffers*. Além de capturar os datagramas que trafegam na rede (não importando o endereço destino do datagrama), os *sniffers* podem decodificá-los e exibi-los em um formato mais legível, ou ainda executar operações mais complexas como recontrução de sessão e recuperação de arquivos transferidos pela rede [6].

Talvez o exemplo mais comum desses programas seja o `tcpdump`[6]. O `tcpdump` pode ser usado para capturar todo tipo de tráfego de rede, decodificar e exibir os datagramas à medida que eles são coletados (no caso de uma análise em tempo real) ou armazenar os datagramas em um arquivo binário, permitindo uma análise posterior (através do próprio `tcpdump` ou de outros aplicativos). A segunda abordagem permite fazer uma cópia exata das informações que trafegam na rede, sendo a mais indicada no caso de uma análise em tempo real não ser necessária [6].

[5] Geralmente, para executar comandos de "status" sobre o *crash dump* é necessário o arquivo de imagem do *kernel* do sistema [27].

[6] Maiores informações sobre o `tcpdump` podem ser encontradas na URL http://www.tcpdump.org (disponível em agosto de 2001).

```
# tcpdump -l -n -e -x -vv -s 1500
# tcpdump -w net.dump
# tcpdump -n -e -x -vv -s 1500 -r net.dump
```

No primeiro exemplo, o `tcpdump` é usado para capturar e exibir os datagramas à medida que eles são coletados. A opção `-l` permite a visualização imediata da saída à medida que ela é produzida, a opção `-n` impede a conversão de endereços em nomes, a opção `-e` imprime o cabeçalho da camada de enlace, a opção `-x` imprime os datagramas no formato hexadecimal, a opção `-vv` imprime informações extras na saída do `tcpdump` e a opção `-s` determina o número de bytes de dados que deve ser capturado de cada datagrama (o padrão é 68 bytes). No segundo exemplo, o `tcpdump` é usado para armazenar os datagramas em um arquivo binário (`net.dump`), através da opção `-w`. Tal arquivo pode ser processado posteriormente, como no terceiro exemplo, através da opção `-r`.

O `tcpdump` possui, ainda, um conjunto de filtros que permitem selecionar os datagramas que devem ser capturados:

```
# tcpdump -l -n -e -x -vv -s 1500 host 192.168.1.3
# tcpdump -l -n -e -x -vv -s 1500 dst port 22
```

No primeiro exemplo, somente serão capturados os datagramas provenientes do endereço IP 192.168.1.3 ou que tenham o mesmo como destino. Já no segundo exemplo, será capturado todo tráfego destinado à porta 22. Maiores detalhes sobre as opções e regras de filtragem do `tcpdump` podem ser encontrados na *man page* do mesmo.

Além dos *sniffers*, alguns sistemas de detecção de intrusão podem ser usados para capturar e analisar o tráfego de rede, ou ainda inspecionar o tráfego e armazenar apenas os dados que pareçam suspeitos [6].

O `snort`[7] é um programa que pode ser usado tanto como *sniffer* quanto como sistema de detecção de intrusão. Ele é capaz de inspecionar a área de dados do datagrama, decodificar as diversas camadas do mesmo e comparar o conteúdo com uma lista de regras. Desse modo, o `snort` pode ser configurado com regras para detectar certos tipos de datagramas, inclusive aqueles relacionados com atividades hostis, como varredura de portas, *buffer overflows* e ataques a servidores web [6]. Além disso, ele pode remontar pacotes fragmentados antes de compará-los com assinaturas de ataques conhecidos.

```
# snort -v -d -e
# snort -d -l log_dir -c snort.conf
# snort -l log_dir -b
# snort -v -d -e -r net.dump
# snort -d -l log_dir -c snort.conf -r net.dump
```

No primeiro exemplo, o `snort` é invocado no modo *sniffer*, através da opção `-v`. As opções `-d` e `-e` instruem o `snort` a exibir, respectivamente, a área de dados do datagrama e o cabeçalho da camada de enlace. No segundo exemplo, o `snort` é executado como sistema de detecção de

[7] Maiores informações sobre o `snort` podem ser encontradas na URL http://www.snort.org (disponível em agosto de 2001).

intrusão, através da opção `-c` que indica o arquivo de configuração do programa. No terceiro exemplo, o `snort` armazena os datagramas em um arquivo binário (no formato do `tcpdump`) no diretório `log_dir`, através das opções `-l` e `-b`. Tal arquivo pode ser processado posteriormente, através da opção `-r`, tanto no modo *sniffer* (quarto exemplo) quanto no modo de detecção de intrusão (quinto exemplo). Maiores detalhes sobre as opções e regras de detecção do `snort` podem ser encontrados na *man page* do mesmo.

Algumas considerações devem ser feitas quando se deseja coletar o tráfego de rede:

- onde deve ser colocado o *sniffer* ?
- todo tráfego deve ser coletado ou apenas um conjunto específico ?
- os datagramas devem ser decodificados e analisados à medida que são coletados ou devem ser armazenados para análise posterior ?

Em uma primeira instância, o *sniffer* deve ser colocado em um ponto da rede que tenha acesso ao tráfego relacionado à máquina invadida. Entretanto, o atacante pode ter comprometido diversas máquinas da rede e a investigação busca determinar a extensão do comprometimento. Nesse caso, o *sniffer* deve ser colocado em uma posição onde possa monitorar todo tráfego da rede, considerando sua topologia e as configurações dos elementos comutadores, como *hubs* e *switches*.

Geralmente, quanto maior a quantidade de dados coletados e maior o número de operações realizadas sobre os datagramas capturados (decodificação, exibição, remontagem ou checagem de assinaturas de ataques, por exemplo), maior será a carga sobre o sistema de coleta, aumentando a chance de que alguns datagramas sejam descartados [6]. Uma filtragem específica dos datagramas que devem ser coletados e a armazenagem dos mesmos, sem decodificação, para futura análise podem reduzir a perda de informações. Em alguns casos, a análise do tráfego de rede necessita ser feita em tempo real, exigindo a decodificação e exibição dos datagramas à medida que eles são capturados. Entretanto, a prática mais recomendada para a coleta de informações é a captura de datagramas em seu estado original no formato binário [6].

A análise dos datagramas capturados geralmente é feita com o auxílio de ferramentas que reconstroem e exibem as informações em um formato mais adequado ao investigador. Algumas ferramentas como, por exemplo, o `ethereal`[8] e o `review` [6], permitem a reprodução da sessão capturada, além da visualização dos eventos de uma maneira mais fácil que a oferecida pelo `tcpdump`. Outra funcionalidade presente em algumas ferramentas (no `review` por exemplo) é a reconstrução de arquivos que foram transferidos durante a sessão capturada, permitindo a recuperação de todo tipo de informação transferida pelo atacante (*exploits*, imagens ou dados roubados, por exemplo) [6]. Os sistemas de detecção de intrusão também são ferramentas úteis na análise do tráfego de rede, pois permitem automatizar o processo de reconhecimento de evidências da intrusão.

Entre as evidências mais comuns que podem ser encontradas na análise do tráfego de rede estão:

- endereço IP inválido ou suspeito (endereços reservados ou conhecidos de outros ataques, por exemplo);

[8] Maiores informações sobre o `ethereal` podem ser encontradas na URL http://www.ethereal.com (disponível em agosto de 2001).

- portas suspeitas (como, por exemplo, programa servidor utilizando uma porta desconhecida acima de 1024);
- porta destino que não deveria estar aceitando datagramas (no caso de um servidor que deveria estar desabilitado, como `telnet` e `rlogin`, por exemplo);
- tráfego não condizente com o padrão do protocolo[9] (*flags*, opções, fragmentação e tamanho dos datagramas inválidos);
- tamanho suspeito do datagrama (não condizente com o normal para o serviço);
- tráfego TCP sem estabelecimento de conexão, sem *flags* ou com números de sequência inválidos (estáticos, aleatórios ou sobrepostos);
- tráfego intenso de datagramas incomuns à rede ou que deveriam estar desabilitados (ICMP *echo request* e *reply*, por exemplo);
- datagrama ICMP *echo reply* sem correspondente *echo request* anterior;
- datagramas ICMP *echo request* e *reply* com número de sequência estático ou não incremental;
- área de dados dos datagramas contendo comandos UNIX, prompts de shell, saídas de comandos e códigos de NOP's;
- *flood* de datagramas para um determinado serviço ou máquina (datagramas chegando com intervalo de tempo muito pequeno)[10];
- requisições HTTP suspeitas (mesma URL em várias requisições ou URL's contendo referências a comandos UNIX ou a *scripts* suspeitos);
- tráfego `telnet` e `ftp` em portas incomuns;

O volume do tráfego de rede pode ser muito grande, de modo que a análise pode concentrar-se nos tipos de tráfego mais comumente utilizados pelos intrusos: sessões de `telnet` e `ftp` e tráfego ICMP, HTTP, SMTP, POP, IRQ e RPC, por exemplo.

## 3.5 Estado do sistema operacional

Informações relacionadas com o estado do sistema operacional podem fornecer pistas importantes quanto à existência, tipo e origem de um ataque em andamento [14]. Tais informações representam uma image do sistema operacional em um determinado instante (processos em execução, conexões de rede estabelecidas, usuários logados, tabelas e *caches* mantidas pelo sistema) e geralmente são perdidas quando o sistema é desligado. Algumas dessas informações são detalhadas como segue.

[9] Informações sobre os protocolos de rede podem ser encontradas nas respectivas RFC's ou em [7, 28, 29].
[10] Detalhes sobre ataques de negação de serviço podem ser encontrados em [12].

### 3.5.1 Estado da rede

O estado da rede provê informações valiosas acerca das conexões de rede em andamento e dos processos aguardando uma conexão [14]. A partir dessas informações é possível determinar se o atacante instalou e ativou uma *back door* no sistema ou se existe alguma conexão não autorizada (ou suspeita) em andamento. O comando `netstat` pode ser usado para capturar informações sobre o estado das conexões e portas abertas no sistema:

```
# netstat -anpe
```

A opção `-a` permite a visualização de servidores aguardando uma conexão, a conversão de endereços IP em nomes é desativada com a opção `-n` (uma consulta DNS reversa pode alertar um atacante experiente [14]), a opção `-p` mostra o número de identificação do processo (PID) e o nome do programa associado a cada conexão, e informações extra, como o usuário dono do processo, são habilitadas através da opção `-e`.

Alguns dos vestígios mais comumente encontrados na saída do comando `netstat` são:

- endereços IP suspeitos (endereços conhecidos de outros ataques ou endereços incomuns a determinados serviços, por exemplo);
- serviços suspeitos aguardando conexão ou com conexões estabelecidas (portas incomuns e nomes suspeitos podem ser indicadores);
- serviços que deveriam estar desabilitados;
- ausência de servidores que deveriam estar em execução;
- *raw sockets*[11] suspeitos (*raw sockets* ICMP, por exemplo);

### 3.5.2 Estado dos processos

A coleta de informações acerca dos processos em execução no sistema analisado é de grande importância, pois tais informações podem revelar evidências de atividades não autorizadas. Cada processo é executado em um ambiente com privilégios específicos que determinam quais recursos do sistema, programas e arquivos de dados podem ser acessados, e de que modo [24]. Um invasor pode desvirtuar a execução de um programa ou serviço, causando sua falência, ou fazendo com que ele opere de maneira inesperada ao administrador ou usuário (acessando informações não autorizadas ou consumindo recursos excessivos, por exemplo). Além disso, o atacante pode executar processos maliciosos como *rootkits*, *back doors* e *trojan horses*.

Existem várias formas de acessar as informações relacionadas aos processos em execução. Uma série de utilitários permitem coletar uma lista de todos os processos do sistema e visualizar detalhes sobre cada um (como, por exemplo, data e hora de início do processo, comando executado, arquivos abertos e consumo de recursos).

O comando `uptime` pode ser usado para determinar a atividade atual e recente do sistema, fornecendo informações sobre a quantidade de tempo em que o sistema encontra-se em execução, além de dados sobre as médias de carga de processamento para os últimos 1, 5 e 15 minutos.

[11] Um *raw socket* provê acesso direto a um protocolo de comunicação de baixo nível, permitindo tirar proveito de algumas características do protocolo não acessíveis pela interface normal [16].

```
# uptime
9:58am  up  1:55,  3 users,  load average: 0.14, 0.03, 0.01
```

No exemplo anterior, o comando `uptime` foi executado às 9:58 da manhã, informando que o sistema estava em execução há uma hora e cinquenta e cinco minutos e que havia 3 usuários logados, além das médias de carga de processamento referentes aos últimos 1, 5 e 15 minutos (nessa ordem).

Uma lista de todos os processos em execução, com detalhes sobre o contexto e estado de cada um, pode ser obtida através do comando `ps`:

```
# ps -auxeww
```

O comando `ps` possui uma grande quantidade de opções que variam bastante entre as diferentes plataformas UNIX. O exemplo anterior provê uma lista de todos os processos (inclusive aqueles sem terminal de controle), contendo informações detalhadas de cada um. Dentre essas informações estão, por exemplo, o nome do programa relativo ao processo, o número de identificação do processo, seu usuário dono e o tempo de início da execução. Maiores detalhes sobre a saída do comando `ps` e suas opções podem ser encontrados na *man page* do mesmo ou em [12].

Uma lista com atualização em tempo real dos processos que mais consomem CPU pode ser obtida através do comando `top`. O comando `lsof` provê uma lista de todos os arquivos abertos no momento de sua execução e os processos relacionados com cada um (a opção `-Dr` pode ser necessária em algumas plataformas UNIX, para evitar que o comando `lsof` escreva no disco da máquina analisada [15]). Uma listagem de todas as chamadas de sistema e chamadas de rotinas de bibliotecas feitas por um processo em execução pode ser obtida, respectivamente, através dos comandos `strace` (ou versões mais modernas como `truss` e `ktrace`) e `ltrace`.

Outra forma de acessar as informações relacionadas aos processos em execução é através da interface provida pelo diretório `/proc`. O diretório `/proc` é um pseudo sistema de arquivos usado como uma interface para as estuturas de dados do *kernel* do sistema operacional, permitindo uma visão do ambiente de cada processo em execução. Cada processo na memória possui um sub-diretório em `/proc` correspondente, cujo nome é o número de identificação do processo (PID). Dentro desse sub-diretório existem informações bastante úteis para o processo de análise forense. O exemplo seguinte ilustra o conteúdo do sub-diretório correspondente ao processo `/root/teste`:

```
# /root/teste -d
```

Com o comando `ps` é possível obter o PID do processo `/root/teste`:

```
# ps -aux | grep /root/teste
USER  PID  %CPU  %MEM  VSZ   RSS  TTY    STAT  START  TIME  COMMAND
root  944  98.8  0.4   1300  292  pts/1  R     09:29  0:29  /root/teste -d
```

O PID do processo `/root/teste` é 944. O conteúdo do diretório `/proc/944` é listado como segue:

```
# ls -la /proc/944
total 0
dr-xr-xr-x    3 root   root   0 May 14 09:39 .
dr-xr-xr-x   55 root   root   0 May 14 04:28 ..
-r--r--r--    1 root   root   0 May 14 09:39 cmdline
lrwxrwxrwx    1 root   root   0 May 14 09:39 cwd -> /
-r--------    1 root   root   0 May 14 09:39 environ
lrwxrwxrwx    1 root   root   0 May 14 09:39 exe -> /root/teste
dr-x------    2 root   root   0 May 14 09:39 fd
-r--r--r--    1 root   root   0 May 14 09:39 maps
-rw-------    1 root   root   0 May 14 09:39 mem
lrwxrwxrwx    1 root   root   0 May 14 09:39 root -> /
-r--r--r--    1 root   root   0 May 14 09:39 stat
-r--r--r--    1 root   root   0 May 14 09:39 statm
-r--r--r--    1 root   root   0 May 14 09:39 status
```

O arquivo `cmdline` contém a linha de comando completa usada para executar o programa:

```
# cat /proc/944/cmdline
/root/teste-d
```

O conteúdo do arquivo `cmdline` é normalmente utilizado pelo comando `ps` para exibir o nome do programa relacionado ao processo [15]. Um atacante pode facilmente alterar a linha de comando através do código do programa[12], com o intuito de "camuflar" o processo na saída do comando `ps` [15].

O arquivo `exe` é um *link* simbólico para o binário que foi executado. Um atacante pode iniciar a execução de um programa e "deletar" o binário correspondente, para esconder seus rastros no sistema. Entretanto, o atacante não está verdadeiramente "deletando" o programa. O sistema UNIX mantém para cada arquivo um contador, chamado de *link count*, que representa o número de processos usando o arquivo. Quando o *link count* iguala-se a zero, nenhum processo está usando o arquivo e o mesmo será "deletado". Quando um atacante executa e "deleta" seu programa, este será removido da cadeia de diretórios (não será mais visível através do comando `ls`), o *link count* do programa é decrementado em uma unidade e o tempo de "deleção" recebe o horário corrente. Contudo, o *link count* não se igualará a zero até que o processo do atacante termine. Os arquivos com *link count* igual a zero são denominados de *unlinked files* e estão marcados para "deleção" quando o sistema for desligado [15].

É de grande importância a recuperação desses arquivos marcados para "deleção", pois após o desligamento do sistema, a recuperação dos arquivos deletados torna-se um processo mais complexo e tedioso [15]. O exemplo seguinte mostra o procedimento usado por muitos atacantes para limpar seus rastros:

```
# /root/teste -d &
[1] 1094
```

[12]Em um programa escrito em código C, basta copiar o nome desejado na primeira posição do vetor `argv`, passado como parâmetro para a função principal `main` (através da função `strcpy(argv[0], ''nome'')`, por exemplo).

```
# rm -f /root/teste
# ls -la /proc/1094
total 0
dr-xr-xr-x    3 root   root   0 May 14 11:43 .
dr-xr-xr-x   61 root   root   0 May 14 04:28 ..
-r--r--r--    1 root   root   0 May 14 11:43 cmdline
lrwxrwxrwx    1 root   root   0 May 14 11:43 cwd -> /
-r--------    1 root   root   0 May 14 11:43 environ
lrwxrwxrwx    1 root   root   0 May 14 11:43 exe -> /root/teste (deleted)
dr-x------    2 root   root   0 May 14 11:43 fd
-r--r--r--    1 root   root   0 May 14 11:43 maps
-rw-------    1 root   root   0 May 14 11:43 mem
lrwxrwxrwx    1 root   root   0 May 14 11:43 root -> /
-r--r--r--    1 root   root   0 May 14 11:43 stat
-r--r--r--    1 root   root   0 May 14 11:43 statm
-r--r--r--    1 root   root   0 May 14 11:43 status
```

O programa `/root/teste` é invocado e "deletado" em seguida. O *link* `/proc/1094/exe` aponta para o binário executado, apresentando a indicação `(deleted)`, pois o programa será marcado para "deleção" quando o processo (PID 1094) terminar. O programa pode ser facilmente recuperado, enquanto o processo estiver em execução, através do comando `cp`:

```
# cp /proc/1094/exe /root/teste
```

Outra fonte de informação útil em `/proc` é o sub-diretório `fd`, que possui um *link* simbólico para cada arquivo[13] que o processo tem aberto. Cada *link* recebe um inteiro positivo como nome, referente ao *file descriptor* do arquivo aberto:

```
# ls -la /proc/547/fd
total 0
dr-x------   2 root   root    0 May 15 10:16 .
dr-xr-xr-x   3 root   root    0 May 15 10:16 ..
lrwx------   1 root   root   64 May 15 10:16 0 -> socket:[732]
l-wx------   1 root   root   64 May 15 10:16 1 -> /var/log/messages
l-wx------   1 root   root   64 May 15 10:16 2 -> /var/log/secure
l-wx------   1 root   root   64 May 15 10:16 3 -> /var/log/maillog
l-wx------   1 root   root   64 May 15 10:16 4 -> /var/log/cron
l-wx------   1 root   root   64 May 15 10:16 5 -> /var/log/spooler
l-wx------   1 root   root   64 May 15 10:16 6 -> /var/log/boot.log
```

O exemplo anterior mostra os *file descriptors* dos arquivos abertos pelo processo `syslogd` (PID 547). Os *links* `0`, `1` e `2` referem-se, respectivamente, à entrada padrão, saída padrão e saída de erro padrão.

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

[13] Por arquivo entenda-se arquivos comuns, *devices*, *sockets* e *pipes* [12, 16].

- existência de processos com nome suspeito (comandos não reconhecidos ou comandos maliciosos, por exemplo);
- ausência de processos que deveriam estar executando (`syslogd` por exemplo);
- acesso a um arquivo suspeito ou não permitido;
- processo abrindo *sockets* suspeitos ou escutando em portas suspeitas;
- processo com tempo de início suspeito ou executado por usuário incomum;
- processo com consumo de recursos incomum;
- existência de várias execuções de programas que deveriam estar executando apenas uma vez (como o `inetd` por exemplo);
- existência de servidores executando fora do *super daemon* `inetd` (como `telnet`, `finger` ou `echo`);
- existência de redirecionadores de porta, como o `fpipe` ou `datapipe`;
- discrepâncias entre as informações dos comandos `ps` e do diretório `/proc`;
- existência de *unlinked files* em `/proc`;
- inconcistências em `/proc`, como `exe` e `cmdline` indicando comandos diferentes;
- *file descriptors* suspeitos no sub-diretório `fd` em `/proc` (saída padrão do `syslogd` apontando para `/dev/null`, por exemplo);

### 3.5.3 Usuários logados

Determinar quais usuários estão logados no sistema é um processo bastante simples [15]. O comando `w` lista, entre outras informações, os usuários logados, o endereço do sistema a partir do qual eles efetuaram o *login* (caso seja remoto), o horário do *login* e o comando que eles estão executando no sistema:

```
# w
  3:47pm  up  1:46,  5 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM   LOGIN@   IDLE   JCPU    PCPU   WHAT
root     tty1     -      3:23pm   23:39  0.21s   0.15s  -bash
root     pts/0    -      3:11pm   35:46  0.04s   0.04s  /bin/cat
root     pts/1    -      3:12pm   0.00s  11.56s  0.03s  w
root     pts/2    -      3:24pm   22:45  0.09s   0.09s  /bin/bash
```

A partir desses dados é possível identificar possíveis evidências como, por exemplo:

- nomes de usuários suspeitos (usuários que não deveriam ter acesso de *login* ou usuários desconhecidos, por exemplo);

- *logins* de origens suspeitas e/ou em horário suspeito;
- *login* remoto de usuário não permitido (*login* remoto de *root*, por exemplo);

### 3.5.4 Estado das interfaces de rede

O estado das interfaces de rede pode revelar a existência de um possível *sniffer* no sistema analisado, bem como uma tentativa de isolar a máquina através da mudança do endereço IP das interfaces. O comando `ifconfig` pode ser usado para obter informações sobre as interfaces de rede:

```
# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:30:21:08:8C:CE
          inet addr:10.1.1.1  Bcast:10.1.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0xde00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:68 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

No exemplo anterior, o comando `ifconfig` é executado com a opção `-a` para fornecer informações sobre todas as interfaces de rede do sistema. A indicação `PROMISC` na terceira linha da interface `eth0` indica que esta encontra-se em modo promíscuo, caracterizando a existência de um *sniffer* [15].

### 3.5.5 Tabelas e *caches*

Embora raro, é possível que um atacante altere as tabelas de rotas ou ainda desvirtue a resolução de endereços MAC [14]. O comando `route` pode ser usado para determinar a tabela de rotas corrente no sistema analisado:

```
# route -een
Kernel IP routing table
Destination Gateway Genmask       Flags Metric Ref Use Iface MSS Window irtt
10.1.1.0    0.0.0.0 255.255.255.0 U     0      0     0 eth0  40  0      0
127.0.0.0   0.0.0.0 255.0.0.0     U     0      0     0 lo    40  0      0
```

No exemplo anterior, o comando `route` é executado com a opção `-ee` que permite visualizar todos os parâmetros da tabela de rotas, além da opção `-n` que impede a resolução de nomes.

Maiores detalhes sobre a saída do comando `route` e suas opções podem ser encontrados na *man page* do mesmo.

O comando `route` pode ser usado ainda para visualizar o conteúdo da *cache* de roteamento do *kernel* do sistema, através da opção `-C`:

```
# route -nC
Kernel IP routing cache
Source          Destination     Gateway         Flags Metric Ref Use Iface
143.106.23.1    200.154.152.241 200.154.152.241 l     0      0     6 lo
143.106.23.1    200.154.152.241 200.154.152.241 l     0      0     0 lo
200.176.2.10    200.154.152.241 200.154.152.241 l     0      0     0 lo
200.154.152.241 143.106.23.1    200.175.132.2         0      0     0 ppp0
200.154.152.241 200.176.2.10    200.175.132.2         0      0     0 ppp0
```

Para analisar o conteúdo da tabela e *cache* de rotas é necessário algum conhecimento sobre a topologia da rede onde encontra-se a máquina invadida, para entender se existe algum tipo de inconscistência, como por exemplo:

- rotas alteradas ou incomuns;
- rotas que passam por redes distantes e/ou desconhecidas;
- rotas estáticas anormais;

Ocasionalmente, os atacantes falsificam endereços IP e MAC para transpor controles de segurança, como listas de controle de acesso, regras de filtragem ou configurações de *switches* [15]. Nesse sentido, a *cache* de resolução de endereços MAC pode ser últil no processo de análise forense para possivelmente determinar se algum tipo de falsificação foi utilizada. O comando `arp` pode ser usado para visualizar o conteúdo da *cache* ARP do sistema analisado:

```
# arp -nv
Address     HWtype  HWaddress           Flags Mask   Iface
10.1.1.44   ether   00:E0:4C:39:01:FB   C            eth0
10.1.1.45   ether   00:E0:4C:39:01:F3   C            eth0
10.1.1.40   ether   00:50:BF:7D:53:FF   C            eth0
10.1.1.1    ether   00:00:21:27:40:C0   C            eth0
Entries: 4      Skipped: 0      Found: 4
```

A opção `-n` impede a resolução de nomes e a opção `-v` fornece informações extra. Maiores detalhes sobre a saída do comando `arp` e suas opções podem ser encontrados na *man page* do mesmo.

Um atacante pode ainda corromper o conteúdo da *cache* de DNS como parte de seu ataque [15]. É possível extrair o conteúdo da *cache* do servidor DNS para procurar entradas indevidas, através do comando `rndc` do BIND 9:

```
# rndc dumpdb
```

Outras informações voláteis, como, por exemplo, as regras de filtragem em execução no *firewall*, os sistemas de arquivo montados e informações sobre serviços RPC, podem ser bastante úteis para reconstruir o estado do sistema quando do início da investigação, permitindo identificar vestígios que podem estar relacionados com a intrusão (regras de filtragem inválidas ou ausentes, sistemas de arquivos montados indevidamente e serviços RPC ativados ou desativados impropriamente, por exemplo).

### 3.5.6 Módulos do *kernel*

Através dos chamados *loadable kernel modules* (LKM) o comportamento do sistema operacional pode ser alterado sem reiniciar o sistema. Módulos maliciosos instalados pelos atacantes podem comprometer totalmente o funcionamento do sistema operacional, interceptando comandos do sistema, como `netstat`, `ifconfig`, `ps` e `ls`, e produzindo resultados falsos [15]. *Rootkits* que instalam LKMs maliciosos representam um desafio aos investigadores, pois quando o atacante tem controle do sistema no nível do *kernel* não é prudente confiar nas informações providas pelos programas executados no sistema analisado, mesmo que estes sejam confiáveis. Entretanto, a própria identificação desses módulos maliciosos representa uma evidência de intrusão. A investigação dos LKMs do sistema analisado pode ser feita através da comparação das informações fornecidas pelos comandos `lsmod` e `kstat`[14] e pela interface `/proc/modules`:

```
# lsmod
Module                  Size  Used by
serial_cs               5040   0  (unused)
pcnet_cs               10052   1
8390                    5860   0  [pcnet_cs]
ds                      6088   2  [serial_cs pcnet_cs]
i82365                 21276   2
pcmcia_core            43424   0  [serial_cs pcnet_cs ds i82365]
bsd_comp                3620   0

# cat /proc/modules
serial_cs               5040   0 (unused)
pcnet_cs               10052   1
8390                    5860   0 [pcnet_cs]
ds                      6088   2 [serial_cs pcnet_cs]
i82365                 21276   2
pcmcia_core            43424   0  [serial_cs pcnet_cs ds i82365]
bsd_comp                3620   0 (unused)

# ./kstat -M
Module              Address
knull               0xc283a000
oMBRa               0xc2838000
```

[14] A ferramenta `kstat` pode ser encontrada na URL http://www.s0ftpj.org/en/site.html (disponível em junho de 2002), juntamente com um HOWTO para detecção de LKMs.

```
serial_cs           0xc2835000
pcnet_cs            0xc282f000
8390                0xc282c000
ds                  0xc2827000
i82365              0xc281c000
pcmcia_core         0xc2810000
bsd_comp            0xc280e000

Do you want to scan memory for LKMs ? [n]
```

A ferramenta `kstat` permite a visualização de informações extraídas diretamente de estruturas do *kernel*, através da interface provida pelo arquivo `/dev/kmem`. Sua utilização é especialmente útil quando não se pode confiar no sistema analisado, embora seja possível modificar o conteúdo da memória do *kernel*.

O exemplo anterior ilustra uma situação onde um módulo malicioso foi instalado no sistema, modificando a chamada de sistema `sys_query_module`, para "camuflá-lo" na saída do comando `lsmod`, e a chamada de sistema `sys_read`, para impedir que o módulo seja visível em `/proc/modules`. O comando `kstat` é executado com a opção `-M`, que permite listar os LKMs instalados. A opção `-M` do comando `kstat` permite ainda varrer uma porção da memória do *kernel* em busca de LKMs invisíveis (*stealth*) e, então, tentar torná-los removíveis novamente. No exemplo anterior,é possível notar uma inconcistência na saída do comando `kstat`, onde aparece o módulo `oMBRa`, possivelmente suspeito (o módulo `knull` é instalado pelo próprio comando `kstat`). Outra indicação comum da existência de módulos maliciosos é a existência de erros ou ausência de dados na saída do comando `lsmod`.

Através da ferramenta `kstat` é possível determinar se alguma chamada de sistema foi possivelmente alterada. Os atacantes geralmente modificam os ponteiros da tabela de chamada de sistemas para que estes apontem para as chamadas de sistema maliciosas.

```
# kstat -s 0
SysCall               Address
sys_exit              0xc011608c
sys_fork              0xc0107668
sys_read              0xc5801230 WARNING! should be at 0xc011356c
sys_query_module      0xd5809060 WARNING! should be at 0xc01169e0

# kstat -s 1
Restoring system calls addresses...
```

Na primeira execução do comando `kstat`, no exemplo anterior, é utilizada a opção `-s` com o argumento `0`, permitindo checar todos os endereços das chamadas de sistema do *kernel*. Já na segunda execução, com a opção `-s` e o argumento `1`, o comando `kstat` pode restaurar os endereços originais das chamadas de sistema. Maiores detalhes sobre a ferramenta `kstat` podem ser obtidos na *man page* da mesma.

Pior do que descobrir a existência de um módulo malicioso no sistema analisado, é executar a coleta de informações sobre o estado do sistema operacional sem descobrir o LKM do atacante. Por esse motivo, a investigação dos módulos do *kernel* é de grande importância para o processo de análise forense.

## 3.6 Dispositivos de armazenagem secundária

O disco representa a maior fonte de informação para o exame forense [22]. Além de ser uma memória não volátil, sua capacidade de armazenamento pode atingir centenas de gigabytes de informações. Do ponto de vista da forense computacional, o disco é muito mais que um conjunto de partições e sistemas de arquivos. De fato, cada porção do disco, por menor que seja, onde se possa ler e/ou escrever um conjunto de bits, representa uma possível fonte de informação para a análise forense.

É possível ler e escrever dados no dispositivo de armazenagem sem a abstração de arquivos, o que é chamado de *raw I/O* [24], seja através dos *device files* do UNIX ou diretamente pelo controlador do disco [22]. Determinadas áreas do disco não são acessíveis através do sistema de arquivos e podem conter informações que o atacante mantém escondidas ou, ainda, vestígios que o mesmo tentou apagar. Nesse sentido, pode-se dividir as informações armazenadas no disco em duas grandes classes: aquelas acessíveis pelo sistema de arquivos (os arquivos propriamente ditos e seus atributos) e as informações armazenadas em áreas não acessíveis através do sistema de arquivos (*file slacks* e espaços não alocados, por exemplo).

As diversas fontes de informação que constituem o disco são apresentadas, de acordo com a classificação do parágrafo anterior, como segue.

### 3.6.1 Áreas não acessíveis através do sistema de arquivos

Cada setor do disco pode ser unicamente identificado por seu endereço CHS: C para o número do cilindro (começando em 0), H para o número da cabeça (começando em 0) e S para o número do setor (começando em 1) [22]. Normalmente o cilindro mais externo e a cabeça mais ao topo recebem o índice 0. No entanto, o primeiro cilindro físico do disco é reservado para o uso do controlador do disco, recebendo o índice -1 [22]. O cilindro -1 é acessível somente ao controlador, através de comandos internos específicos, e contém dados sobre o mapeamento de setores defeituosos e sobre a geometria física do disco (usados para auto-configuração pela BIOS, por exemplo) [22]. Um setor pode ser identificado também por seu endereço absoluto, que se inicia no zero e é incrementado até o final do disco [15].

Todo disco possui um *master boot record* (MBR) localizado na mesma posição: cilindro 0, cabeça 0 e setor 1. O MBR contém o programa de análise de partições (*partition analysis program*), executado durante a inicialização do sistema, e a tabela de partições [22]. Quando o computador é ligado, um código armazenado na BIOS é executado. Após finalizar algumas tarefas iniciais, o código da BIOS carrega os 512 bytes do MBR na memória, começando no endereço de memória `0000:7C00`. Assim que o MBR é carregado na memória, a BIOS desvia o processador (faz um *branch*) para o primeiro byte do MBR na memória (*offset* `00h`), que será executado como código de um programa [22]. No caso do disco, os primeiros bytes do MBR causarão a execução do programa de análise de partições, cujo código encontra-se também no setor do MBR. Esse programa investiga a tabela de partição, contida nos últimos bytes do setor do MBR, e determina qual é a partição ativa do disco (partição que contém o sistema operacional que deve ser inicializado). Determinada a partição ativa, o programa de análise de partições carrega o primeiro setor da partição (setor de *boot*) no mesmo endereço de memória `0000:7C00`, desviando sua execução para o código contido no setor carregado (o código contido no setor de *boot* da partição é o chamado *bootstrap loader*) [22]. O *bootstrap loader* inicializa, então, o sistema operacional contido na partição.

A tabela de partições sempre começa no *offset* `1beh` do MBR e pode conter até quatro entradas de 16 bytes [22]. Cada uma dessas entradas pode referenciar uma partição primária, ou seja, uma partição que pode conter um *bootstrap loader* no primeiro setor, juntamente com um sistema operacional [22]. Para superar essa limitação de ter apenas quatro partições no disco, uma das entradas da tabela de partições do MBR pode referenciar uma partição extendida. A primeira partição extendida, cuja entrada encontra-se na tabela de partições do MBR, é apenas um "recipiente" para uma ou mais partições lógicas (também chamadas de partições secundárias). O primeiro setor de uma partição extendida contém uma nova tabela de partições e o restante desse setor é usualmente preenchido com zeros. Essa tabela de partições extendida também começa no *offset* `1beh` do setor e, embora tenha quatro entradas de 16 bytes, é permitida a utilização de apenas duas entradas. Apenas a primeira entrada da tabela de partições extendida pode conter uma partição lógica e essa partição pode abrigar um sistema de arquivos e, excepcionalmente, um sistema operacional (capaz de ser inicializado a partir de uma partição lógica, como o Windows NT, OS/2 e Linux, por exemplo) [22]. A segunda entrada pode apenas especificar uma nova partição extendida, que por sua vez contém uma nova tabela de partições extendida [22]. A figura 1 ilustra a organização das partições extendidas e lógicas.

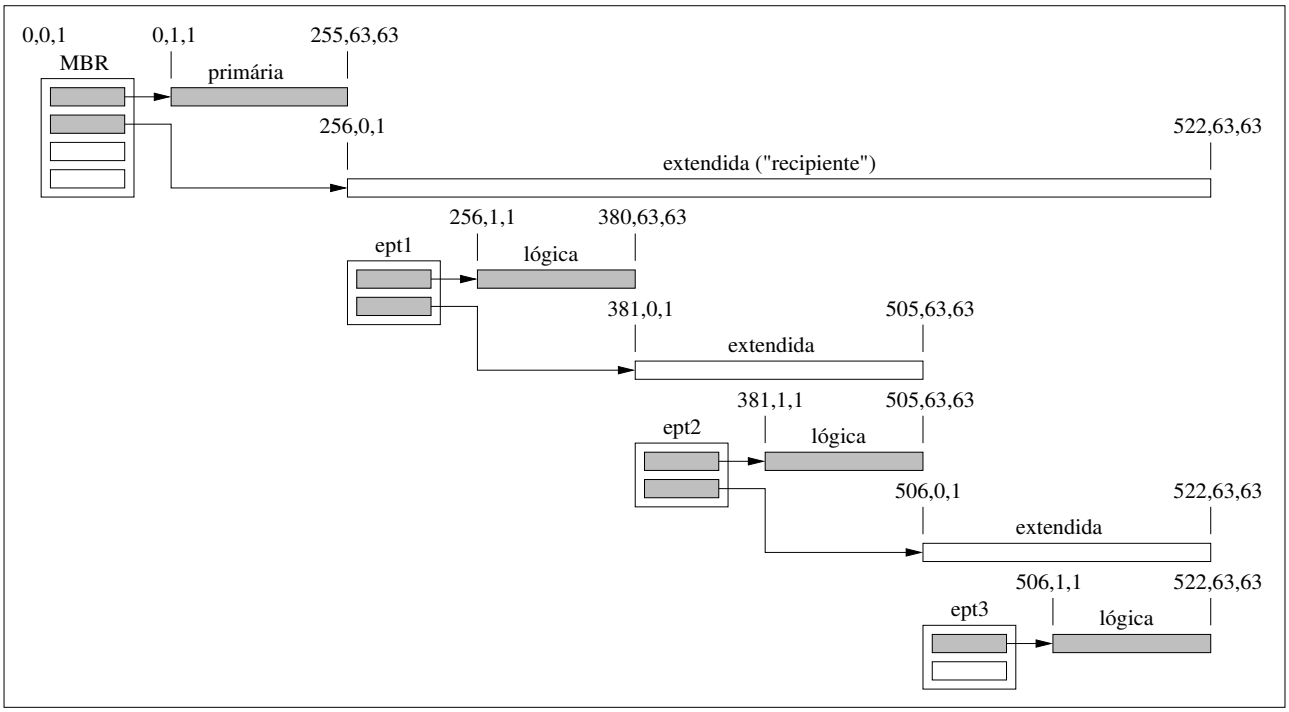


Figura 1: Partições extendidas e lógicas.

Toda essa explanação apresentada anteriormente, no nível de detalhes usado, é necessária para identificar as diversas oportunidades que existem para esconder informações em um disco, como listadas a seguir [22]:

- com controladores antigos, setores e trilhas perfeitos podem ser marcados como defeituosos, através de comandos internos do controlador, de modo que as informações contidas neles tornam-se inacessíveis;

- algumas partições podem ser deliberadamente marcadas como "escondidas", usando por exemplo o programa `PartitionMagic`[15];
- as partições acessíveis podem não ocupar todo o disco (pode haver espaço entre partições ou no final do disco). A tabela de partições pode ter sido modificada, usando algum editor de disco, de modo que uma ou mais partições válidas não são mais registradas;
- a BIOS pode não suportar a geometria do disco de modo que não é permitido acesso a toda porção física do disco;
- as partições lógicas podem não ocupar totalmente a primeira partição extendida (partição "recipiente");
- como indicado na figura 1, é uma prática normal as tabelas de partições (MBR ou extendidas) começarem na cabeça 0, setor 1 de um cilindor, e o primeiro setor de uma partição (setor de *boot*) começar na cabeça 1, setor 1 de um cilindro. A consequência dessa prática é a existência de setores não utilizados entre o setor da tabela de partições e o primeiro setor da partição. Claramente esses setores podem ser utilizados para esconder informações, sem qualquer risco de serem detectadas pelo uso normal do sistema de arquivos;
- apenas o MBR normalmente contém um programa de análise de partições. Uma tabela de partições extendida (ept1, ept2 e ept3 na figura 1) apenas contém uma ou duas entradas de partições, começando no *offset* `1deh` do setor que a contém, de modo que a maior parte do setor não é utilizada. Os bytes do *offset* `00h` ao `1ddh` do setor que contém uma tabela de partições extendida podem ser usados para esconder informações;
- o sistema de arquivos pode não ocupar totalmente a partição devido ao tamanho de bloco (*cluster*) utilizado. Os últimos setores desperdiçados devem ser checados;
- podem existir partições que não contêm um sistema de arquivos. Isso pode ser proposital, com o intuito de desviar a atenção da partição e esconder informações;
- um sistema de arquivos pode estar sendo utilizado apenas para desviar a atenção, de modo que os espaços não alocados aos arquivos existentes estão sendo usados sem a organização do sistema de arquivos;
- espaços alocados a arquivos, mas não totalmente utilizados (os chamados *file slacks*[16]), também podem ser usados para esconder informações;
- o programa de análise de partições do MBR e o *bootstrap loader* do setor de *boot* das partições podem ser deliberadamente alterados, permitindo a execução de código malicioso durante a inicialização do sistema;

[15] Informações sobre o programa `PartitionMagic` podem ser encontradas na URL http://www.powerquest.com (disponível em julho de 2002).

[16] Os *file slacks* [24] ocorrem pelo fato do sistema de arquivos alocar blocos de tamanho fixo. 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, desperdiçando os últimos 612 bytes alocados.

- áreas de *swap* podem conter diversos dados temporários como, por exemplo, dados do *kernel* e de processos ou *buffers* de arquivos temporários e da impressora. Entre essas informações podem estar dados que nunca foram salvos no disco ou, ainda, dados deliberadamente escondidos;

Além de informações deliberadamente escondidas, os espaços não utilizados por arquivos no dispositivo de armazenagem (espaços não alocados dentro de um sistema de arquivos; espaços alocados a arquivos, mas não totalmente utilizados; e áreas do dispositivo de armazenagem que não constituem uma partição de disco ou que não contêm um sistema de arquivos) podem conter informações "deletadas"[17]. Quando um arquivo é "deletado", 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 "deletados" e parcialmente sobrescritos [5, 14].

Os espaços não utilizados por arquivos no dispositivo de armazenagem podem ser examinados de várias formas. Antes de iniciar a análise do disco, é interessante reunir o máximo de informações sobre sua configuração. Uma ferrameta bastante útil é o programa `fdisk`[18], que pode ser utilizado para visualizar informações sobre as partições do disco:

```
# fdisk -lu

Disk /dev/hda: 255 heads, 63 sectors, 1240 cylinders
Units = sectors of 1 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *        63   9221309   4610623+   c  Win95 FAT32 (LBA)
/dev/hda2       9221310  18040994   4409842+  83  Linux
/dev/hda3      18040995  18458684    208845   82  Linux swap
/dev/hda4      18458685  19920599    730957+   5  Extended
/dev/hda5      18458748  19486844    514048+  83  Linux
/dev/hda6      19486908  19695689    104391   83  Linux
/dev/hda7      19695753  19920599    112423+  83  Linux
```

No exemplo anterior o comando `fdisk` é executado com a opção `-l` que permite visualizar informações sobre as partições sem alterar nenhum dado no disco, além da opção `-u` que fornece as informações em termos de setores (ao invés de cilindros). O comando `fdisk` com a opção `-u` utiliza o endereçamento absoluto de setores (começando em zero e incrementando até o final do disco), facilitando a identificação da localização exata das partições. Maiores detalhes sobre o comando `fdisk` podem ser obtidos na *man page* do mesmo.

[17] No caso dos sistemas de arquivos EXT2FS e FFS, os *file slacks* podem conter informações deliberadamente escondidas, entretando, os dados pertencentes a arquivos "deletados" são sobrescritos. Devido à organização de blocos e fragmentos, adotada por esses sistemas de arquivos, quando um novo bloco é alocado, o restante do último fragmento utilizado é preenchido com zeros, apagando qualquer informação existente. Os fragmentos não utilizados no bloco alocado são disponibilizados como espaço não alocado.

[18] Além do `fdisk` existe uma versão mais correta e poderosa, chamada `sfdisk`, segundo a própria *manpage* do comando `fdisk`.

Além de obter informações sobre a configuração do disco, é importante efetuar a imagem bit a bit do dispositivo. Esse tipo de imagem é diferente de uma cópia normal de arquivos, pois todo conteúdo do disco será copiado, incluindo os espaços não utilizados mencionados anteriormente. A imagem do disco pode ser efetuada através do comando `dd` em ambientes UNIX. Maiores detalhes sobre o processo de imagem serão abordados na seção 5.

Utilizando uma imagem do disco, que pode estar armazenada em um arquivo ou reconstruída em outro disco, é possível extrair os dados contidos nas áreas não acessíveis através do sistema de arquivos. A extração das informações pode ser feita com o comando `dd`:

```
# sfdisk -luS hda.dd

Disk hda.dd: 1240 cylinders, 255 heads, 63 sectors/track
Units = sectors of 512 bytes, counting from 0

   Device Boot  Start       End  #sectors  Id  System
hda.dd1   *        63   9221309   9221247   c  Win95 FAT32 (LBA)
hda.dd2       9221310  18040994   8819685  83  Linux
hda.dd3      18040995  18458684    417690  82  Linux swap
hda.dd4      18458685  19920599   1461915   5  Extended
hda.dd5      18458748  19486844   1028097  83  Linux
hda.dd6      19486908  19695689    208782  83  Linux
hda.dd7      19695753  19920599    224847  83  Linux

# dd if=hda.dd of=unusedSectors.dd bs=512 skip=1 count=62
62+0 records in
62+0 records out
```

É possível visualizar informações sobre as partições de uma imagem de disco armazenada em um arquivo (`hda.dd`), como mostrado no exemplo anterior. Nesse mesmo exemplo, o comando `dd` é usado para extrair o espaço não utilizado entre o setor do MBR (setor 0) e o setor de *boot* da primeira partição (setor 63). A informação extraída é armazenada no arquivo binário `unusedSectors.dd` e as opções `bs`, `skip` e `count` são utilizadas para instruir o comando `dd` a copiar 62 setores de 512 bytes a partir do setor de número 1. Maiores detalhes sobre os comandos `sfdisk` e `dd` podem ser encontrados nas respectivas *man pages*.

Uma maneira mais eficiente de extrair as informações contidas nas áreas não acessíveis através do sistema de arquivos é através de ferramentas específicas para análise forense. O utilitário `unrm`, que compõe o conjunto de ferramentas chamado *The Coroner's Toolkit* (TCT)[19], é capaz de criar um único arquivo contendo toda informação presente nos espaços não alocados de um sistema de arquivos:

```
# unrm hda2.dd > unallocated.unrm
```

No exemplo anterior, o comando `unrm` é usado para copiar, no arquivo binário `unallocated.unrm`, toda informação contida nos espaços não alocados do sistema de arquivos

[19] O TCT será apresentado em maiores detalhes na seção 6.

da imagem `hda2.dd` (imagem da partição Linux `/dev/hda2`). É importante mencionar que o arquivo destino deve estar em outro sistema de arquivos que o utilizado como fonte do comando `unrm` e o seu tamanho pode compreender vários gigabytes de dados. Além disso, o comando `unrm` acessa apenas as informações dos espaços não alocados, não extraindo, por exemplo, os dados contidos nos *file slacks* (que são áreas alocadas).

A análise das informações contidas nas áreas não acessíveis através do sistema de arquivos é, na maioria das vezes, um processo tedioso e demorado, uma vez que esses dados geralmente constituem um fluxo de bits sem estrutura alguma aparente. Um editor hexadecimal pode ser usado para visualizar e examinar essas informações. O comando `xxd` é bastante útil, permitindo a conversão de um fluxo de bits qualquer para o formato hexadecimal:

```
# dd if=/dev/hda bs=512 count=1 skip=0 | xxd
1+0 records in
1+0 records out
0000000: faeb 7c6c 6261 4c49 4c4f 0100 1504 5a00  ..|lbaLILO....Z.
0000010: 0000 0000 b789 9f3c a742 b0ca 00a8 42b0  .......<.B....B.
0000020: ca00 a642 b0ca 0001 445a aa42 b0ca 00ab  ...B....DZ.B....
0000030: 42b0 ca00 6f2b b0c9 0070 2bb0 c900 712b  B...o+...p+...q+
...
00001b0: 595f eb02 b440 0000 0000 0000 0000 8001  Y_...@..........
00001c0: 0100 0cfe bf3d 3f00 0000 7fb4 8c00 0000  .....=?.........
00001d0: 813e 83fe ffff beb4 8c00 e593 8600 00fe  .>..............
00001e0: ffff 82fe ffff a348 1301 9a5f 0600 00fe  .......H..._....
00001f0: ffff 05fe ffff 3da8 1901 9b4e 1600 55aa  ......=....N..U.
```

O exemplo anterior mostra o setor do MBR do disco `/dev/hda`. É possível ver parte do código do `lilo`, no início do setor, e a tabela de partições, extendendo-se do *offset* `1beh` ao `1fdh`.

Outra abordagem de análise é a busca de palavras-chave, utilizando, por exemplo, os comandos `strings` e `grep`:

```
# unrm /dev/hda5 | strings -a | grep Jul
Jul 19 14:38:05 host pppd[866]: Exit.
Jul 19 13:52:22 host kde[690]: session opened for user root by (uid=0)
```

No exemplo anterior, é utilizada uma combinação dos comandos `unrm`, `strings` e `grep` para buscar entradas de log deletadas, referentes ao período de julho, na partição que abriga o diretório `/var/log` (repositório dos principais arquivos de log do Linux).

Além da análise dos dados com editores hexadecimal e da busca de palavras-chave, existe uma terceira abordagem: tentar organizar as informações de maneira coerente e estruturada, de modo a formar arquivos de dados (arquivos binários, textos, figuras, entre outros tipos de arquivos). O conjunto de ferramentas TCT contém um programa, chamado `lazarus`, que recebe um fluxo de bits como entrada e sistematicamente analisa cada bloco de dados, tentando reconstruir arquivos com dados coerentes. O `lazarus` pode ser usado com qualquer tipo de fluxo de bits, como, por exemplo, *dumps* da memória, áreas de *swap* e *core files*.

Uma prática importante durante a análise de espaços livres no dispositivo de armazenagem é a procura de sistemas de arquivos escondidos, por exemplo, pela remoção proposital da partição na tabela de partições. É necessário procurar anormalidades nas tabelas de partições (tanto do MBR quanto nas tabelas de partições extendidas) e partições do tipo "escondida" (com os códigos 11h, 14h ou 16h, por exemplo). Além disso, deve-se checar os setores de *boot* das partições e o setor do MBR, para certificar-se de que o programa de análise de partições e o *bootstrap loader* não possuem instruções suspeitas (pode-se fazer uma comparação com versões confiáveis desses programas).

### 3.6.2 Sistema de arquivos

A fonte de informação onde o processo de análise forense geralmente se concentra mais é o sistema de arquivos. Como um banco de dados, o sistema de arquivos é a parte do sistema operacional responsável por organizar as informações do disco na forma de arquivos [24]. Cada arquivo é representado em um sistema UNIX por uma estrutura de dados chamada *inode*, que contém a localização do arquivo no disco e informações sobre propriedade, permissões de acesso e marcas de tempo (*timestamps*). A tabela 2 resume o conteúdo de um *inode*.

| Elemento | Descrição |
|---|---|
| IDs do proprietário e do grupo | Números de identificação do proprietário do arquivo e do grupo proprietário. Os valores são indexados aos nomes contidos em `/etc/passwd` e `/etc/group` |
| Tipo | Tipo do *inode* : regular (arquivo comum), diretório, dispositivo de caracter (*character device*), dispositivo de bloco ( *block device*) ou *named pipe* |
| Permissões de acesso | Permissões para escrita, leitura e execução. As permissões são dadas para três diferentes abstrações: o proprietário, o grupo e todos que têm acesso ao sistema |
| Tempos:<br>*mtime*<br>*atime*<br>*ctime* | Tempos de: última modificação no arquivo, último acesso e última mudança nas propriedades do arquivo |
| Número de *links* | Número de entradas de diretório apontando para o *inode* |
| Ponteiros para blocos de dados | Localização dos blocos de dados no disco |
| Tamanho | Tamanho do arquivo |

Tabela 2: Conteúdo de um *inode*.

Um *inode* não contém o nome do arquivo, de modo que essa informação encontra-se na entrada de diretório que aponta para o *inode*. Quando uma entrada de diretório aponta para um *inode* tem-se o chamado *link*. Dessa forma, um *inode* pode ser referenciado por várias

entradas de diretório diferentes. Logo, durante a análise forense, é importante lembrar que um mesmo arquivo pode aparecer um múltiplos diretórios simultaneamente e que esse arquivo pode ter nomes diferentes em cada diretório [14].

O conteúdo de um *inode* pode ser visualizado através do comando `stat`:

```
# stat /var/log/messages
  File: "/var/log/messages"
  Size: 2984638    Blocks: 5858      Regular File
Access: (0600/-rw-------)         Uid: ( 0/ root )  Gid: ( 0/ root )
Device: 305        Inode: 12259      Links: 1
Access: Fri Jul 19 16:24:38 2002
Modify: Tue Jul 23 14:12:25 2002
Change: Tue Jul 23 14:12:25 2002
```

No exemplo anterior, é visualizado o *inode* de número 12259, referente ao arquivo `/var/log/messages`. É possível identificar as permissões[20] de acesso ao arquivo (`Access`), os *userid* e *groupid* do proprietário do arquivo (`Uid` e `Gid`, respectivamente) e os chamados *mactimes* (`Modify`, `Access` e `Change`).

É importante mencionar que os aplicativos como `stat` e `ls`, por exemplo, utilizam os valores numéricos *userid* e *groupid* para indexar os respectivos nomes associados em `/etc/passwd` e `/etc/group` da máquina onde o sistema de arquivos está sendo examinado. Desse modo, os nomes dos usuários e dos grupos associados aos *userid* e *groupid* pelos aplicativos podem estar inconcistentes com os nomes presentes no sistema invadido, se o sistema de arquivos for analisado em outra máquina (o que deve ser a prática comum). Por esse motivo, é recomendado utilizar os valores numéricos *userid* e *groupid* durante a análise e fazer manualmente a associação com os respectivos nomes, utilizando os arquivos `/etc/passwd` e `/etc/group` do sistema analisado [14].

As marcas de tempo dos arquivos (*mactimes*) representam recursos importantes para a reconstrução dos eventos associados a uma intrusão [6]. O sistema UNIX possui pelo menos três marcas de tempo para cada arquivo (o sistema de arquivos EXT2FS do Linux possui quatro): última modificação de conteúdo (*mtime*), último acesso (*atime*) e última mudança nas propriedades do arquivo (*ctime*). O Linux possui ainda uma marca com o tempo de "deleção" (*dtime*). Esses tempos são armazenados no formato correspondente ao número de segundos passados desde a "época do UNIX" (01 de janeiro de 1970) e indicam o tempo no horário de *Greenwich* (GMT). A tradução dos tempos feita pelos utilitários como o `ls` varia de acordo com a zona de tempo (*timezone*) indicada pela variável de ambiente TZ, portanto, a análise forense deve computar os tempos no horário de *Greenwich* ou adequar a variável de ambiente TZ, da máquina de análise, ao valor encontrado no sistema analisado [14].

O conjunto de ferramentas TCT possui um utilitário chamado `mactime` que permite criar uma linha de tempo, cronologicamente ordenada, de arquivos e seus respectivos *mactimes*:

```
# mactime -d /root 07/24/2002
Jul 24 02 08:49:14   4096 mac drwxr-xr-x root root  /root/backups
```

[20] Uma explanação detalhada sobre permissões de acesso e propriedade de arquivos pode ser encontrada em [12].

```
Jul 24 02 08:49:48  19269 m.c -rw------- root root  /root/.bash_history
Jul 24 02 08:55:10     16 .a. -rw------- root root  /root/.esd_auth
Jul 24 02 13:14:06      7 .a. -rw-r--r-- root root  /root/.wmrc
Jul 24 02 13:14:10      7 m.c -rw-r--r-- root root  /root/.wmrc
                      101 m.c -rw------- root root  /root/.Xauthority
Jul 24 02 13:14:11    266 .a. -rw-r--r-- root root  /root/.bash_profile
                     1126 .a. -rw-r--r-- root root  /root/.Xresources
```

No exemplo anterior, o utilitário `mactime` foi executado sobre o diretório `/root`, através da opção `-d`, requisitando uma linha de tempo a partir da data 24 de julho de 2002 (no formato americano mês/dia/ano). É possível observar, por exemplo, que o arquivo `/root/.wmrc` foi acessado pela última vez em 24 de julho de 2002 às 13:14:06 (quarta linha da saída do comando) e teve seu conteúdo modificado em 24 de julho de 2002 às 13:14:10 (quinta linha da saída do comando).

Infelizmente os *mactimes* são facilmente modificados e extremamente efêmeros – a próxima vez que alguém acessar ou modificar um arquivo de qualquer forma, os valores anteriores dos *mactimes* atualizados serão perdidos. Entretanto, se houver razoável certeza de que os *mactimes* não foram deliberadamente distorcidos, eles podem fornecer pistas valiosas sobre os arquivos e diretórios acessados e/ou modificados pelo invasor, os programas compilados e executados na máquina invadida e outros eventos que envolvam o sistema de arquivos.

Existem outras formas de acessar as informações do sistema de arquivos, além das descritas anteriormente. O programa `stat` pode ser executado com a opção `-f`, para obter informações sobre o sistema de arquivos onde reside o arquivo passado como parâmentro:

```
# stat -f .
  File: "."
    ID: 0        0      Namelen: 255  Type: EXT2
Blocks: Total: 1085138  Free: 668780  Available: 613657  Size: 4096
Inodes: Total: 551616   Free: 445510
```

Com a opção `-f`, o comando `stat` permite visualizar, por exemplo, o tipo do sistema de arquivos (EXT2FS, no caso do exemplo), o número de blocos total, livres e disponíveis para alocação aos arquivos (o número de blocos livres não corresponde ao número de blocos disponíveis para arquivos, pois alguns blocos são alocados aos *inodes*) e o número de *inodes* total e livres (o número de arquivos e diretórios é determinado pelo número de *inodes* existentes no sistema de arquivos, alocados no momento da formatação).

O conjunto de ferramentas TCT possui ainda um utilitário, chamado `ils`, que permite visualizar informações sobre *inodes* não alocados. Quando um arquivo é "deletado", o *inode* associado a ele é disponibilizado para reutilização por um novo arquivo, entretanto, as informações armazenadas no *inode* não são apagadas até que o mesmo seja reutilizado. Desse modo, o utilitário `ils` permite recuperar informações sobre arquivos "deletados" como, por exemplo, os *userid* e *groupid* do proprietário e os *mactimes*. Outro utilitário do TCT, chamado `icat`, permite visualizar o conteúdo de um arquivo ou diretório a partir do número do *inode* correspondente, podendo ser utilizado inclusive com *inodes* não alocados (permitindo recuperar arquivos "deletados", caso o sistema não apague os ponteiros para os blocos do arquivo). O

exemplo a seguir ilustra como tais ferramentas podem ser utilizadas em conjunto para obter informações sobre arquivos "deletados" e até mesmo recuperá-los:

```
# echo ‘‘Hello World’’ > teste
# touch inode_info
# ls -i teste
 441149 teste
# rm -f teste
# sync
# ils /dev/hda2 441149 | ils2mac > inode_info
# mactime -b inode_info 7/25/2002
Jul 25 02 10:31:00  6 ma. -rw-r--r-- root root  <hda2-dead-441149>
Jul 25 02 10:32:00  6 ..c -rw-r--r-- root root  <hda2-dead-441149>
# icat /dev/hda2 441149
Hello World
```

O arquivo `teste` é criado contendo a frase "`Hello World`". Em seguida, o arquivo `inode_info` é criado para armazenar as informações sobre o *inode* do arquivo `teste`. O comando `ls` é executado com a opção `-i` para visualizar a número do *inode* do arquivo `teste` (*inode* 441149). Então, o arquivo `teste` é "deletado" (o comando `sync` é usado para forçar a atualização do sistema de arquivos) e a ferramenta `ils` é utilizada para recuperar as informações contidas no *inode* 441149 (disponibilizado pela "deleção" do arquivo `teste`), armazenando-as no arquivo `inode_info`. Juntamente com o comando `ils`, é executado o *script* `ils2mac` (também pertencente ao TCT) para converter a saída do `ils` para o formato processado pela ferramenta `mactime`. Em seguida, o comando `mactime` é executado sobre os dados contidos no arquivo `inode_info`, permitindo a visualização das informações relacionadas ao arquivo "deletado" (arquivo `teste`). Por fim, a ferramenta `icat` é utilizada para recuperar o conteúdo do *inode* 441149, referente ao arquivo "deletado".

Outro conjunto de ferramentas de forense, denominado TCTUTILs, provê funcionalidades adicionais ao TCT, apresentando utilitários que permitem obter e recuperar informações do sistema de arquivos. A ferramenta `istat` permite visualizar todas as informações de um determinado *inode*, inclusive o endereço dos blocos de dados do arquivo:

```
# istat /dev/hda2 2
inode: 2
Allocated
uid / gid: 0 / 0
mode: rwxr-xr-x
size: 4096
num of links: 18
Modified:       07.25.2002 08:17:50     (AMT+0)
Accessed:       07.25.2002 09:26:33     (AMT+0)
Changed:        07.25.2002 08:17:50     (AMT+0)
Deleted:        12.31.1969 20:00:00     (AMT+0)
Direct Blocks:
  511
```

No exemplo anterior, o conteúdo do *inode* de número 2 (associado ao diretório raiz do sistema de arquivos em `/dev/hda2`) é visualizado. O utilitário `bcat`, também pertencente ao TCTUTILs, pode ser usado para visualizar o conteúdo do bloco 511, alocado ao diretório `/` (associado ao *inode* 2, como visto no exemplo anterior):

```
# bcat -h /dev/hda2 511
0       02000000 0c000102 2e000000 02000000     .... .... .... ....
16      0c000202 2e2e0000 0b000000 14000a02     .... .... .... ....
32      6c6f7374 2b666f75 6e640000 c17e0000     lost +fou nd.. .~..
48      0c000302 76617200 81fd0000 28000402     .... var. .... (...
64      70726f63 1d000000 1c000801 706f7765     proc .... .... powe
80      726f6666 6c2d7374 61676532 2e696d67     roff l-st age2 .img
96      417c0100 0c000302 746d7000 01fb0100     A|.. .... tmp. ....
112     0c000302 64657600 c1790200 0c000302     .... dev. .y.. ....
128     65746300 41770300 0c000302 75737200     etc. Aw.. .... usr.
144     d43f0000 0c000302 62696e00 0fb70300     .?.. .... bin. ....
160     0c000402 626f6f74 13b70300 0c000402     .... boot .... ....
176     686f6d65 14b70300 0c000302 6c696200     home .... .... lib.
192     6aec0700 0c000302 6d6e7400 6dec0700     j... .... mnt. m...
208     0c000302 6f707400 6eec0700 0c000402     .... opt. n... ....
224     726f6f74 6fec0700 0c000402 7362696e     root o... .... sbin
240     36f20500 100f0402 6d697363 0c000000     6... .... misc ....
256     040f1401 7268696e 7374616c 6c2d7374     .... rhin stal l-st
272     61676532 2e696d67 00000000 00000000     age2 .img .... ....
```

A listagem do exemplo anterior continua até completar o tamanho do bloco, no caso 4096 bytes. É possível observar as entradas de diretório contidas no bloco, cada uma indicando o número do *inode* referente ao arquivo ou sub-diretório, o tamanho em bytes da entrada, o nome do arquivo ou sub-diretório e o tamanho em bytes do nome. Por exemplo, a entrada que vai do *offset* 96 ao *offset* 107 da listagem apresenta as informações detalhadas na tabela 3 (de acordo com a estrutura `ext2_dir_entry_2` do sistema de arquivos EXT2FS usado no Linux).

Outra ferramenta integrante do TCTUTILs é o programa `fls`, que permite listar as entradas de diretório contidas nos blocos alocados a um diretório qualquer. É possível inclusive visualizar entradas referentes a arquivos "deletados", pois estas não são apagadas até que outra entrada de diretório ocupe o espaço utilizado [4]:

```
# ls -di /tmp
 97345 /tmp
# echo ‘‘Hello World’’ > /tmp/teste
# ls -i /tmp/teste
 100496 /tmp/teste
# rm -f /tmp/teste
# sync
# fls -d /dev/hda2 97345
r * 100496:     teste
r * 100497:     xfig-export001001.err
```

| Campo | Valor haxadecimal | Significado |
|---|---|---|
| número do *inode* (32 bits no formato *little endian*) | `417c0100` | *inode* número 97345 |
| tamanho da entrada (16 bits no formato *little endian*) | `0c00` | entrada ocupa 12 bytes no total |
| tamanho do nome contido na entrada (8 bits) | `03` | nome ocupa 3 bytes (sem contar o caracter nulo) |
| tipo da entrada (8 bits) | `02` | entrada do tipo *diretório* |
| nome do arquivo ou subdiretório (tamanho variável de no máximo 255 caracteres, terminado com um caracter nulo) | `746d7000` | nome do diretório é `tmp` |

Tabela 3: Conteúdo de uma entrada de diretório.

No exemplo anterior, o arquivo `/tmp/teste` é criado e "deletado" em seguida (o comando `sync` força a atualização do sistema de arquivos). Então, a ferramenta `fls` é executada com a opção `-d`, para listar as entradas de diretório, referentes aos arquivos "deletados", contidas nos blocos apontados pelo *inode* 97345 (associado ao diretório `/tmp`). É possível observar a entrada relacionada ao arquivo "deletado" `/tmp/teste`, mostrando o número do *inode* ocupado pelo arquivo (100496). De posse dessas informações, é possível verificar se o *inode* associado ao arquivo "deletado" encontra-se disponível e, então, utilizar as técnicas descritas anteriormente para tentar recuperar maiores informações sobre o arquivo e até mesmo seu conteúdo.

Os utilitários apresentados, pertencentes tanto ao TCT quanto ao TCTUTILs, facilitam bastante o processo de recuperação de informações sobre arquivos "deletados". A reconstrução de arquivos "deletados", pelo processo de identificação de informações coerentes nos setores disponíveis do disco, sem a utilização de qualquer estrutura de dados que indique alguma organização desses setores, é uma tarefa bastante complexa e na maioria das vezes ineficiente. Por outro lado, a utilizando das pistas deixadas nas estruturas de dados do sistema de arquivos torna o processo de recuperação de informações sobre arquivos "deletados" mais simples e imediato.

A quantidade e qualidade das pistas deixadas nas estruturas de dados do sistema de arquivos irá determinar o número de informações que podem ser recuperadas sobre um arquivo "deletado". Por exemplo, seguindo a representação ilustrada na figura 2, quando o arquivo `/file.txt` é "deletado", o bloco de dados de número 1000, o *inode* 10 e a respectiva entrada de diretório no bloco 511 são disponibilizados para reutilização. Utilizando as técnicas descritas anteriormente e dependendo da utilização do sistema de arquivos, é possível recuperar a entrada de diretório referente ao arquivo "deletado" `/file.txt` (através da ferramenta `fls`), caso o espaço disponibilizado no bloco 511 não tenha sido reutilizado por uma nova entrada. Recuperada a entrada de diretório, é possível identificar o nome do arquivo (no caso `/file.txt`) e o número do *inode*

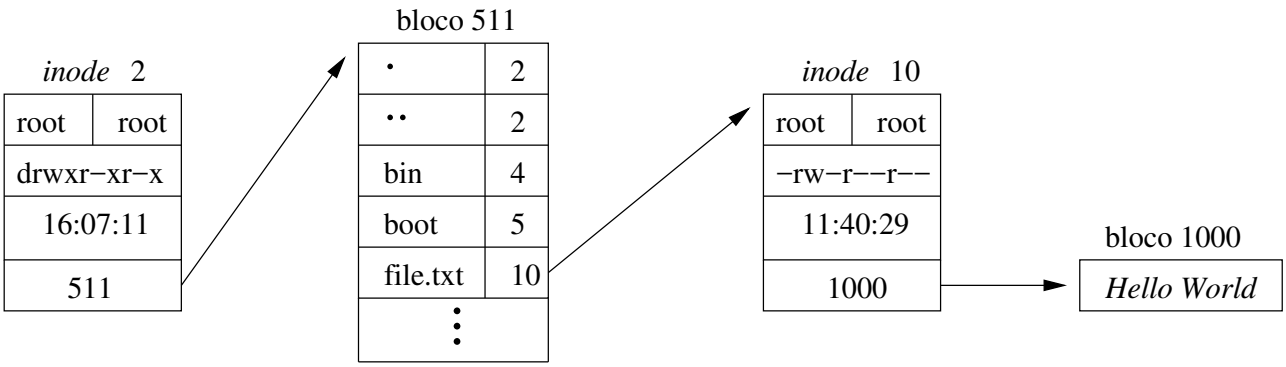


Figura 2: Representação da estrutura interna de um sistema de arquivos.

que ele utilizava. Caso o *inode* identificado não esteja alocado a um novo arquivo ou diretório (facilmente determinado através da ferramenta `istat`), é possível recuperar várias informações sobre o último "ocupante" do *inode* (por sorte, o arquivo `/file.txt`), como, por exemplo, os *mactimes* e os blocos de dados utilizados. Seguindo a trilha dos blocos de dados, é possível recuperar o conteúdo, ou parte dele, do último "ocupante" dos blocos (por sorte, novamente, o arquivo `/file.txt`), caso estes não estejam alocados.

As técnicas apresentadas até o momento visam a extração e recuperação de informações contidas nas estruturas internas do sistema de arquivos. É importante observar que, em todos os comandos apresentados nos exemplos, foi fornecido como parâmetro o nome de algum arquivo ou diretório, ou o *device file* relativo à partição que contém o sistema de arquivos analisado (`/dev/hda2`, no caso dos exemplos). Durante uma análise forense real, o sistema de arquivos analisado pode apresentar-se, idealmente, de três formas:

- espelhado em um disco do sistema de análise;
- contido no arquivo de imagem do disco analisado;
- contido no arquivo de imagem da partição analisada;

Como será visto na seção 5, a imagem pode conter o disco todo ou cada sistema de arquivos (partição) separadamente, podendo ser armazenada em um arquivo ou espelhada em outro disco.

Na primeira forma, o sistema de arquivos analisado encontra-se em uma das partições do disco espelhado na máquina de análise, podendo ser acessado diretamente através de um *device file*, como nos exemplos apresentados anteriormente. Na terceira forma, o arquivo contendo a imagem da partição que abriga o sistema de arquivos pode ser fornecido como parâmetro aos comandos, em substituição ao *device file*. Já na segunda forma, é preciso identificar no arquivo de imagem do disco, onde começa o sistema de arquivos em questão. Isso pode ser feito através de um *device file* especial, chamado *loopback device*, e do comando `losetup`:

```
# sfdisk -luS hda.dd

Disk hda.dd: 1240 cylinders, 255 heads, 63 sectors/track
```

```
Units = sectors of 512 bytes, counting from 0

   Device Boot  Start       End  #sectors  Id  System
hda.dd1   *        63   9221309   9221247   c  Win95 FAT32 (LBA)
hda.dd2       9221310  18040994   8819685  83  Linux
hda.dd3      18040995  18458684    417690  82  Linux swap
hda.dd4      18458685  19920599   1461915   5  Extended
hda.dd5      18458748  19486844   1028097  83  Linux
hda.dd6      19486908  19695689    208782  83  Linux
hda.dd7      19695753  19920599    224847  83  Linux

# losetup -o 4721310720 /dev/loop0 hda.dd
```

No exemplo anterior, o *loopback device* `/dev/loop0` é associado ao sistema de arquivos presente na partição `hda.dd2` (referente à partição `/dev/hda2` do disco original). Isso é feito indicando-se o *offset*, em bytes, a partir do início do arquivo `hda.dd` (arquivo contendo a imagem de todo o disco), onde se encontra o sistema de arquivos desejado (no caso do exemplo, o *offset* é 4721310720 bytes, referente ao setor de início da partição, 9221310, vezes 512 bytes, que é o tamanho de cada setor). Dessa forma, o *loopback device* `/dev/loop0` pode ser fornecido como parâmetro aos comandos apresentados para extração de informações do sistema de arquivos, em substituição ao *device file* `/dev/hda2`. Maiores detalhes sobre o comando `losetup` podem ser obtidos na *man page* do mesmo.

Para se ter acesso aos arquivos e diretórios do sistema de arquivos analisado é preciso antes montá-lo no sistema de análise. Se a partição contendo o sistema de arquivos analisado foi espelhada em um disco do sistema de análise, então a montagem do sistema de arquivos é feita normalmente, utilizando o *device file* correspondente à partição:

```
# mount -t ext2 -o ro,noexec,nodev /dev/hda2 /mnt/mountpoint
```

É importante observar que o sistema de arquivos analisado deve ser montado apenas para leitura (com a opção `-o ro`), impedindo qualquer tipo de alteração, e que a execução de binários e interpretação de *device files* no sistema de arquivos montado devem ser desabilitadas (com as opções `-o noexec,nodev`). É conveniente criar um diretório específico, referente ao caso investigado, como ponto de montagem do sistema de arquivos analisado. Maiores informações sobre o comando `mount` e suas opções podem serm encontradas na *man page* do comando.

Nos casos em que o sistema de arquivos analisado está contido em um arquivo de imagem, a montagem é feita através de um *loopback device* (com a opção `-o loop`):

```
# mount -t ext2 -o ro,noexec,nodev,loop=/dev/loop0 hda2.dd \
  /mnt/mountpoint
# mount -t ext2 -o ro,noexec,nodev,loop=/dev/loop1,offset=4721310720 \
  hda.dd /mnt/mountpoint
```

No exemplo anterior, o arquivo `hda2.dd` contém a imagem da partição referente ao sistema de arquivos analisado, que é montado através do *loopback device* `/dev/loop0`. Já o arquivo `hda.dd` contém a imagem de todo o disco, precisando ser indicado o *offset* (em bytes) onde se

encontra o sistema de arquivos analisado (de maneira indêntica ao apresentado para o comando `losetup`). Se o comando `losetup` foi utilizado anteriormente para associar um *loopback device* ao sistema de arquivos analisado, contido em um arquivo de imagem, o mesmo *loopback device* pode ser utilizado diretamente no comando `mount` para montar o sistema de arquivos:

```
# losetup -o 4721310720 /dev/loop0 hda.dd
# mount -t ext2 -o ro,noexec,nodev /dev/loop0 /mnt/mountpoint
```

Devidamente preparado e montado, o sistema de arquivos pode ser analisado em busca de evidências da intrusão. Um resumo das principais fontes de informações contidas no sistema de arquivos é apresentado na tabela 4.

| Fonte de Informação | Descrição |
|---|---|
| arquivos e diretórios de configuração | O sistema UNIX mantém certos arquivos de configuração que são comumente acessados e/ou alterados pelos atacantes. Em especial estão os *scripts* de inicialização, arquivos de informações sobre contas e senhas, configuração dos serviços de rede, tarefas agendadas, relações de confiança e do sistema de log |
| filas | Incluem informações sobre processos em execução e atividades incompletas. Podem conter mensagens de correio eletrônico e documentos enviados para impressão que não existem em qualquer outro lugar do sistema |
| *caches* | Arquivos de *cache* são usados para melhorar a performance do sistema ou de algum aplicativo. Podem conter informações valiosas como, por exemplo, um histórico de *sites Web* supostamente visitados pelo atacante ( *sites* contendo informações e ferramentas para intrusões) |
| arquivos de *swap* | Quando a demanda do sistema excede a capacidade da memória, algumas informações são retiradas da memória e armazenadas temporariamente nos arquivos de *swap*. Tais arquivos podem conter fragmentos de dados ou até mesmo um arquivo completo que nunca foi salvo no disco |
| diretórios temporários | Diretórios temporários (`/tmp` e `/usr/tmp` ) servem como diretórios de "rascunho" para todo o sistema. Como eles são limpos periodicamente (em teoria), acabam se tornando locais apropriados para armazenar dados que não serão usados no futuro. Por essa razão, muitos atacantes usam esses diretórios como diretórios de serviço, não se importando em apagar os rastros deixados. Além disso, é possível encontrar nesses diretórios arquivos temporários de aplicativos (editores de texto, por exemplo), cujos originais foram cifrados ou nunca foram salvos no disco |

Tabela 4: Resumo das fontes de informações contidas no sistema de arquivos.

| Fonte de Informação | Descrição |
|---|---|
| `/dev` | Com exceção de alguns poucos arquivos especiais, o diretório `/dev` deve conter apenas os *device files*. Muitos atacantes exploram a complexidade do conteúdo desse diretório para esconder suas informações |
| arquivos e diretórios escondidos ou não usuais | Arquivos e diretórios "invisíveis" ou com nomes pouco usuais são frequentemente usados pelos atacantes para esconder informações |
| executáveis e bibliotecas | Os atacantes comumente instalam *trojan hourses, rootkits* e *exploits* no sistema invadido. Esses programas geralmente modificam algum executável (binário ou *script* ) do sistema ou alguma biblioteca carregada dinamicamente. Além disso, muitos atacantes deixam para trás seus códigos maliciosos (executáveis ou códigos−fonte) que podem ser analisados em busca de detalhes sobre a intrusão |
| arquivos de log | Os arquivos de log representam um papel crucial na análise do sistema de arquivos, pois permitem a reconstituição de fatos que ocorreram no sistema. Tais arquivos pode registrar, entre outras informações, as atividades dos usuários, dos processos e do sistema, as conexões e atividades da rede, e informações específicas dos aplicativos e serviços |
| arquivos e diretórios de usuários | Arquivos de texto, mensagens de correio eletrônico, registros de conversas em salas de bate papo, arquivos de imagens, entre outros tipos de dados, podem conter informações úteis relacionadas ao atacante |

Tabela 4: Resumo das fontes de informações contidas no sistema de arquivos (continuação).

Uma ferramenta importante para o processo de análise das informações do sistema de arquivos é o *hash* criptográfico[21]. O propósito do *hash* de um arquivo[22] é estabelecer uma assinatura do mesmo em seu estado confiável, sendo utilizada para checagem de integridade. O processo de criação de bases de assinaturas criptográficas e checagem de integridade pode ser totalmente automatizado com a utilização de ferramentas como o `Tripwire`[23]. Entretanto, o processo pode ser "manual", através do comando `md5sum`:

```
# md5sum -b /etc/passwd
e437472d827159584ddb2f7e0f038d88 */etc/passwd
```

Existe uma "quase certeza" de que muitos arquivos e diretórios irão conter evidências relacionadas com a intrusão. Entretanto, o sucesso da análise forense em identificar todos os arquivos e diretórios relevantes é bem menos certo [15]. Felizmente, existem certos tipos de arquivos e diretórios (resumidos na tabela 4) que comumente contêm ou representam evidências relacionadas com a intrusão, como, por exemplo, arquivos SUID e SGID, arquivos e diretórios

[21] Maiores informações sobre criptografia podem ser encontradas em [23].

[22] Na verdade, o *hash* criptográfico pode ser usado com qualquer fluxo de bits, não necessariamente um arquivo.

[23] Informações sobre o programa `Tripwire` podem ser encontradas na URL http://www.tripwire.com (disponível em julho de 2002).

escondidos ou com nomes incomuns, arquivos de configuração e diretórios temporários. Alguns desses arquivos e diretórios são discutidos como segue.

### Arquivos de configuração

Os arquivos de configuração representam excelentes locais para se encontrar evidências de uma intrusão [15]. Um atacante experiente pode facilmente modificar a configuração do sistema e aplicativos, com o intuito de abrir *back doors*, esconder sua presença ou apenas causar estragos. Entre os alvos mais frequentes estão os *scripts* de inicialização, o controle de acesso à máquina (como, por exemplo, os arquivos de configuração `/etc/hosts.allow` e `/etc/hosts.deny`, a configuração do *firewall* e do sistema de autenticação PAM), as relações de confiança, as configurações das contas de usuários e grupos, os serviços de rede, as tarefas agendadas e os sistemas de log e monitoramento.

No sistema Linux, alguns serviços e o próprio sistema operacional são especialmente inicializados e terminados com base nas configurações dos *scripts* localizados no diretório `/etc/rc.d`. Entretanto, um atacante pode facilmente adicionar comandos maliciosos a esses *scripts*, permitindo, por exemplo, a execução de *trojan hourses* durante a inicialização do sistema. Outros *scripts* de inicialização encontram-se nos diretórios dos usuários, como, por exemplo, `.bash_profile`, `.bashrc` e `.cshrc`, que são consultados automaticamente quando o usuário acessa o sistema ou quando determinados programas são executados. É necessário checar cada *script* de inicialização em busca de comandos maliciosos e verificar se os programas executados a partir deles são legítimos. Como exemplos de comandos maliciosos encontrados nos *scripts* de inicialização, podem ser citados os seguintes:

- modificar as permissões de acesso de arquivos sensíveis (arquivos de log, por exemplo), para torná-los inacessíveis (por exemplo, `chmod 0 /var/log/*`);
- colher informações sobre o sistema e armazenar em algum lugar acessível (por exemplo, `tcpdump >> /tmp/arquivo`);
- enviar informações sensíveis por correio eletrônico;
- adicionar usuários e senhas em `/etc/passwd` e `/etc/shadow`;
- remover arquivos sensíveis;

Relações de confiança entre sistemas UNIX podem ser convenientes para os administrados da rede, mas introduzem falhas de segurança que são comumente utilizadas pelos atacantes [12]. Se algum dos utilitários de acesso remoto, conhecidos por *Berkeley R utilities* (notórios por sua insegurança), `rsh`, `rlogin` e `rcp` estiverem habilitados, os atacantes podem tentar aumentar o conjunto de máquinas que podem utilizá-los [14].

As relações de confiança são geralmente configuradas através de arquivos como `.rhosts`, presente nos diretórios dos usuários[24], e `/etc/hosts.equiv`. Tais relações podem ser estabelecidas também através do programa `ssh` (por meio dos arquivos `.shosts` e

[24]Os arquivos de configuração localizados nos diretórios dos usuários podem ser encontrados através do comando `find /home -type f -name ‘‘.?hosts’’ -print0 | xargs -0 ls -l`. Deve-se prestar atenção especial àqueles encontrados no diretório de usuários com privilégios administrativos.

`/etc/ssh/shosts.equiv`), por compartilhamentos através de NFS e pelo serviço NIS [15]. Além disso, os *firewalls* e outros mecanismos de controle de acesso, como o TCP Wrapper e o PAM, implementam outra forma de relação de confiança, uma vez que geralmente são configurados para permitir que determinados endereços origem tenham acesso à máquina protegida.

Os atacantes podem modificar ou "deletar" os arquivos que configuram o controle de acesso e as relações de confiança para permitir seu retorno facilitado à máquina invadida. Desse modo, todas as possíveis relações de confiança e permissões de acesso devem ser investigadas para determinar se elas estão relacionadas com o incidente. Entre as principais evidências encontradas, com relação ao controle de acesso e relações de confiança, podem ser citadas:

- permissão de acesso para máquinas desconhecidas (especialmente para máquinas fora da organização);
- permissão de acesso para usuários desconhecidos ou suspeitos;
- permissão de acesso a serviços desconhecidos ou suspeitos (inseguros ou desativados);
- habilitação de relações de confiança quando deveriam estar desativadas;
- presença do caracter '+' (sinal positivo) nos arquivos de configuração das relações de confiança, permitindo a conexão remota proveniente de qualquer máquina;
- compartilhamentos via NFS suspeitos em `/etc/exports` (inseguros ou desconhecidos);
- servidor NIS desconhecido ou diferente do normal em `/etc/yp.conf`;

Os arquivos de configuração de contas e senhas de usuários e grupos (`/etc/passwd`, `/etc/shadow` e `/etc/group`) frequentemente mostram sinais de comprometimento em sistemas invadidos [14]. Tais sinais podem vir, por exemplo, na forma de novas contas ou escalonamento de privilégios de contas existentes, com o objetivo usual de criar *back doors* para futuros acessos. As informações sobre as contas e senhas de usuários e grupos devem ser investigadas em busca de inconscistências ou características suspeitas, como, por exemplo:

- contas criadas sem o conhecimento do administrador do sistema;
- contas sem senha;
- contas com diretório de usuário suspeito (por exemplo, `/tmp`);
- contas com *shell* de *login* suspeito (como, por exemplo, outro tipo de programa ou algo diferente de `/bin/false`, quando for o caso);
- entradas nos arquivos de configuração com número impróprio de campos ou com características diferentes das demais;
- contas com *userid* e/ou *groupid* suspeitos (principalmente *userids* e *groupids* 0 e 1);
- contas que deveriam estar desabilitadas ou indisponíveis para acesso remoto como, por exemplo, `daemon`, `sync` e `shutdown`;

- usuários suspeitos em grupos de alto privilégio;

Alguns utilitários como `pwck` e `grpck` podem ser utilizados para checar as entradas contidas em `/etc/passwd`, `/etc/shadow` e `/etc/group`. Essas ferramentas apenas verificam se as entradas estão no formato apropriado e se apresentam dados válidos em cada campo, não fazendo nenhuma avaliação quanto a características suspeitas ou inseguras.

Uma instalação padrão do sistema Linux oferece um grande conjunto de serviços de rede, incluindo o NFS, `telnet`, `finger`, `rlogin` e muitos outros. Qualquer um desses serviços de rede pode potencialmente permitir algum tipo de acesso remoto a usuários indesejáveis [15]. Alguns do pontos de acesso mais comumente explorados pelos atacantes incluem, por exemplo, servidores X, FTP, telnet, TFTP, DNS, sendmail, finger, SNMP, IMAP, POP, HTTP, HTTPS e RPC. Durante a investigação dos arquivos de configuração dos serviços de rede, todos devem ser examinados como potenciais pontos de acesso ao sistema, considerando que alguns serviços podem apresentar vulnerabilidades ou ter sido substituídos por *trojan hourses*. Nesse sentido, é preciso verificar se os pontos de acesso apresentam-se configurados de maneira segura e possuem os *patches* e versões mais recentes dos programas. Como exemplos das principais evidências encontradas em relação à configuração dos serviços de rede, podem ser citadas as seguintes:

- habilitação de serviços que estavam previamente desativados (como, por exemplo, `telnet`, `finger` e `rlogin`);
- adição de entradas suspeitas em `/etc/xinetd.conf`, permitindo acesso a portas e serviços incomuns ou desativados;
- adição de portas e serviços suspeitos (geralmente portas altas) ou modificação das entradas legítimas em `/etc/services`;
- alterações inesperadas no serviço de resolução de nomes;
- configurações inseguras no serviço FTP [12] (como, por exemplo, permissão para usuários anônimos executarem comandos "perigosos" como `delete` e `chmod`);
- diretório de FTP com permissões indevidas;

O serviço de agendamento de tarefas é comumente disponibilizado através dos programas `crond`, `anacron` e `at`, apresentando diversos arquivos e diretórios de configuração, como, por exemplo, `/var/spool/cron`, `/var/spool/at`, `/var/spool/anacron`, `/etc/crontab`, `/etc/cron.d` e `/etc/anacrontab`. Assim como os administradores do sistema utilizam tarefas agendadas para automatizar suas atividades, os atacantes também fazem uso dessa facilidade. Como as tarefas agendadas nos arquivos de configuração são executadas com os privilégios do proprietário do arquivo, esses arquivos de configuração costumam ser utilizados para executar programas maliciosos [15]. É preciso localizar todos os diretórios e arquivos contendo tarefas agendadas e examinar cuidadosamente cada uma, bem como os executáveis invocados por elas. Entre as principais evidências encontradas com relação a tarefas agendadas, podem ser citadas as seguintes:

- tarefas desconhecidas ao administrador do sistema;

- referência a executáveis desconhecidos ou incomuns (por exemplo, localizados fora dos diretórios normais dos binários do sistema);
- referência a executável com permissões de escrita para todos os usuários;
- arquivos de configuração de tarefas agendadas com permissões de escrita para todos os usuários;

Os sistemas de segurança implementados na máquina invadida também podem ser alvos dos atacantes. Os sistemas de log e de detecção de intrusão podem ser alterados com o intuito de esconder os rastros do invasor e permitir que ele retorne sem ser descoberto. Ao analisar a máquina invadida, é importante verificar a configuração do esquema de log no arquivo `/etc/syslog.conf`, bem como determinar se as opções de log dos diversos serviços e aplicativos estão devidamente configuradas. No caso da máquina analisada hospedar algum tipo de sistema de detecção de intrusão, é necessário verificar a configuração do mesmo, pois o atacante pode ter desabilitado o monitoramento de alguma parte do sistema ou rede.

Em geral, a configuração de algumas partes do sistema tende a permanecer constante após um determinado momento. Desse modo, os arquivos de configuração podem ser verificados, quanto a modificações inesperadas, através da comparação com cópias de segurança e registros de alterações mantidos pelo administrador do sistema. Nesse sentido, algumas evidências podem ser facilmente encontradas, como, por exemplo:

- modificação das permissões de acesso aos arquivos de configuração, permitindo que sejam facilmente alterados no futuro;
- tempos de modificação dos arquivos de configuração diferem da última modificação legítima;
- assinaturas criptográficas dos arquivos diferem das cópias de segurança, indicando uma possível alteração indevida;
- remoção de alguns arquivos de configuração (relacionados com controle de acesso, por exemplo);

O comando `diff` permite comparar o conteúdo de dois arquivos, podendo ser usado para identificar as diferenças entre os arquivos de configuração e suas cópias de segurança. Outra alternativa é o comando `rpm`, que pode ser usado para checar a consistência dos binários instalados e de alguns arquivos de configuração do sistema[25]:

```
# rpm -Va
S.5....T c /etc/printcap
S.5....T c /etc/man.config
.M....G.   /dev/tty1
S.5....T   /usr/share/umb-scheme/slibcat
.......T c /etc/pam.d/system-auth
S.5....T c /etc/xinetd.d/telnet
```

[25]Existe uma discussão na SANS sobre a utilização do RPM como ferramenta de recuperação. Maiores detalhes podem ser encontrados na URL http://www.sans.org/y2k/RPM.html (disponível em julho de 2002).

Todo objeto que tem alterada alguma das seguintes características é listado pelo comando, juntamente com a letra associada à mudança (maiores detalhes sobre o comando `rpm` podem ser encontrados na *man page* do mesmo):

- 5: alteração na assinatura MD5;
- S: alteração no tamanho do arquivo;
- L: alteração no *link* simbólico;
- T: alteração no tempo de modificação;
- D: alteração no *device*;
- U: alteração no usuário proprietário;
- G: alteração no grupo proprietário;
- M: alteração nas permissões;

No sentido de identificar algum comprometimento na configuração do sistema, é possível utilizar ferramentas de análise de vulnerabilidades [2]. Talvez o exemplo mais conhecido de programa de análise de vulnerabilidades seja o COPS [11], que varre o sistema em busca de falhas de segurança conhecidas e geralmente utilizadas pelos atacantes.

### Diretórios temporários

Os diretórios temporários são comumente utilizados pelos atacantes como diretórios de serviço, podendo conter diversos sinais relacionados com a intrusão, como, por exemplo:

- arquivos de rascunho criados por aplicativos durante o tempo em que se suspeita que o invasor estava no sistema;
- arquivos executáveis (tais arquivos não são normalmente encontrados em diretórios temporários);
- arquivos e sub-diretórios escondidos;
- arquivos intermediários de processos de compilação e instalação de programas;
- código-fonte e *scripts* de instalação de programas (possivelmente *rootkits*, *exploits* ou *trojan hourses*);
- arquivos do tipo `tar` ou compactados (muitos atacantes fazem o *download* de suas ferramentas, que se encontram em arquivos desses tipos);

### Diretório de *device files*

Assim como os diretórios temporários, o diretório `/dev` apresenta uma certa predileção por parte dos invasores, que exploram a complexidade de seu conteúdo para esconder suas informações [14]. Qualquer arquivo regular ou sub-diretório presentes em `/dev` devem ser examinados cuidadosamente:

```
# find /dev -type f -print0 | xargs -0 ls -l
-rwxr-xr-x    1 root     root        13526 Jul 31 18:02 /dev/.fd001
-rwxr-xr-x    1 root     root        15256 Mar 24  2001 /dev/MAKEDEV

# file /dev/.fd001
/dev/.fd001: ELF 32-bit LSB executable, Intel 80386, version 1
```

O *script* `MAKEDEV` sempre é encontrado em `/dev`, embora seja necessário verificar se o mesmo é legítimo. O arquivo "invisível" `.fd001`, mostrado no exemplo anterior, não deveria estar no diretório `/dev`. O comando `file` mostra que o referido arquivo trata-se de um binário, necessitando de uma investigação mais detalhada.

### Arquivos e diretórios escondidos ou não usuais

Os atacantes geralmente utilizam arquivos e diretórios "invisíveis" (iniciados com ponto) ou com nomes pouco usuais, para esconder suas informações. Diferentes combinações de pontos, espaços em branco e caracteres de controle podem ser usadas nos nomes dos arquivos e diretórios dos atacantes. Alguns exemplos são mostrados no exemplo a seguir:

```
# ls -la
total 28
drwxr-xr-x    2 root     root         4096 Aug  3 09:00
drwxr-xr-x    7 root     root         4096 Aug  3 09:00 .
drwxr-xr-x    3 root     root         4096 Aug  3 08:59 ..
drwxr-xr-x    2 root     root         4096 Aug  3 09:00 ..
drwxr-xr-x    2 root     root         4096 Aug  3 09:00 ...
drwxr-xr-x    2 root     root         4096 Aug  3 09:00 .?
drwxr-xr-x    2 root     root         4096 Aug  3 09:00 ?

# ls -la | cat -A
total 28$
drwxr-xr-x    2 root     root         4096 Aug  3 09:00    $
drwxr-xr-x    7 root     root         4096 Aug  3 09:00 .$
drwxr-xr-x    3 root     root         4096 Aug  3 08:59 ..$
drwxr-xr-x    2 root     root         4096 Aug  3 09:00 .. $
drwxr-xr-x    2 root     root         4096 Aug  3 09:00 ...$
drwxr-xr-x    2 root     root         4096 Aug  3 09:00 .^T$
drwxr-xr-x    2 root     root         4096 Aug  3 09:00 ^T$
```

O comando `cat` com a opção `-A` pode ser usado para processar a listagem do diretório, permitindo a visualização de caracteres especiais[26] e a identificação do final de linha (através do caracter "$").

### Executáveis e bibliotecas

Quando um sistema é comprometido, o atacante provavelmente deixa algum tipo de código

[26] Os caracteres de controle são identificados pelo acento circunflexo.

malicioso para trás, seja na forma de executáveis (binários ou *scripts*) ou bibliotecas compartilhadas [14]. O atacante frequentemente instala no sistema alguns programas maliciosos ou bibliotecas comprometidas, como, por exemplo, *back doors* para permitir seu retorno, *rootkits* e *trojan hourses* para esconder seus rastros, *exploits* e *scanners* para descobrir e explorar vulnerabilidades, *sniffers* e *password crackers* para coletar informações. Durante a investigação, é necessário localizar todos os executáveis e bibliotecas do sistema, isolar aqueles mais suspeitos e examinar detalhadamente cada um, para determinar se são legítimos ou se executam algum tipo de tarefa não autorizada.

Os programas SUID e SGID *root* (ou qualquer outra conta com privilégios administrativos) são bastante explorados pelos atacantes, tanto como *back doors* quanto para o aumento de privilégios [15]. Através do comando `find`, é possível encontrar todos os programas SUID e SGID do sistema:

```
# find / -perm -4000 -print0 | xargs -0 ls -l
# find / -perm -2000 -print0 | xargs -0 ls -l
```

Qualquer programa SUID ou SGID *root* localizado fora dos diretórios de executáveis normais do sistema (`/bin`, `/sbin`, `/usr/bin` e `/usr/sbin`, por exemplo) são suspeitos [14]. No entanto, aqueles encontrados nos diretórios padrão também devem ser examinados para verificar se são legítimos. A assinatura criptográfica dos programas SUID ou SGID *root* suspeitos pode ser comparada com a assinatura de programas comumente utilizados pelos atacantes, como o `/bin/ksh`.

Além dos programas SUID e SGID, é necessário localizar todos os executáveis presentes nos diretórios dos usuários e nos diretórios mais comumente utilizados pelos atacantes (`/tmp`, `/usr/tmp`, `/var/tmp` e `/dev`, por exemplo). É importante observar que os atacantes podem "camuflar" seus executáveis, desabilitando a permissão de execução dos arquivos. Desse modo, uma combinação dos comandos `find`, `file` e `grep` pode ser usada para localizar os executáveis[27]:

```
# find /home /tmp /usr/tmp /var/tmp /root /dev -type f -print0 | xargs \
  -0 file -zL | grep -E 'executable|script|perl'
```

Obviamente é necessário mais do que simplesmente listar os executáveis do sistema. Uma das primeiras coisas a se fazer é identificar a presença de *trojan hourses* instalados, checando a integridade dos executáveis mais comumente visados pelos atacantes, como, por exemplo, `ps`, `netstat`, `ls`, `login`, `in.telnetd` e `ssh` (em geral todos os executáveis localizados em `/bin`, `/sbin`, `/usr/bin` e `/usr/sbin` devem ser verificados) [14]. Essa checagem pode ser feita através dos *mactimes* dos arquivos (*ctime* ou *mtime* próximos do período da invasão) ou de assinaturas criptográficas, utilizando-se ferramentas automatizadas como o `Tripwire` ou manualmente com o comando `md5sum` (comparando-se com cópias confiáveis). Outra forma de checar a consistência dos executáveis instalados é através do comando `rpm`, como apresentado anteriormente. Além de verificar os executáveis encontrados, é importante também notar a ausência ou adição de novos programas no sistema.

Os executáveis e bibliotecas identificados como suspeitos devem ser analisados para determinar se são hostis e quais funcionalidades são implementadas. Algumas vezes, as próprias

[27] A utilização do comando `file` sobre todos os arquivos é uma abordagem lenta, porém confiável.

circunstâncias (nome e diretório do arquivo) podem fornecer pistas úteis. Muitos atacantes nem mesmo se preocupam em modificar os nomes de suas ferramentas, de modo que se o executável suspeito tiver nomes do tipo *sniffer* ou *passwordcrack*, existem boas chances do mesmo implemetar a funcionalidade sugerida por seu nome (uma busca na Internet pelo nome do arquivo pode revelar pistas valiosas) [14]. Entretanto, existem outras formas de analisar os executáveis e bibliotecas suspeitos, conforme descrito a seguir.

Tentar ler o arquivo deve ser a primeira abordagem. Se o código suspeito é um *script*, é possível analisar os comandos e determinar sua funcionalidade. Entretanto, se o código é um binário, pouco se pode fazer nesse sentido, a menos que o código-fonte esteja disponível. Geralmente os atacantes compilam suas ferramentas na máquina invadida, de modo que o código-fonte[28] pode ainda estar presente no sistema (geralmente nas proximidades do executável ou nos diretórios mais comumente utilizados pelos atacantes, como `/tmp`) ou pode ter sido "deletado" e ainda estar disponível para recuperação (segundo as técnicas descritas anteriormente).

Outra abordagem é analisar as cadeias de caracteres (*strings*) contidas nos binários, através do comando `strings`:

```
# strings -a arquivo
```

A opção `-a` é importante para garantir que todo o arquivo seja varrido pelo comando `strings`. A execução do comando `strings` sobre o binário suspeito pode revelar informações sobre os arquivos que ele acessa, suas mensagens de erro e ajuda, nomes de funções e bibliotecas utilizadas, entre outros dados que podem fornecer pistas sobre a funcionalidade do executável. Em alguns casos é possível encontrar palavras comumente utilizadas pelos atacantes, na forma de senhas (a palavra "sartori" é usada como senha em alguns *rootkits*), gírias (a palavra "warez", por exemplo) ou nomes de ferramentas hostis (uma busca na Internet[29] pode revelar muitas dessas palavras). Desse modo, é possível utilizar o comando `strings` em conjunto com o utilitário `grep` para procurar palavras-chave, como as mencionadas anteriormente ou ainda nomes de arquivos específicos e mensagens produzidas no terminal.

Além do comando `strings`, a ferramenta `nm` pode ser usada para listar a tabela de símbolos do programa suspeito. Executando o comando `nm` com as opções `-Du`, é possível visualizar as bibliotecas utilizadas pelo binário suspeito, o que pode fornecer pistas, como, por exemplo, se o código desconhecido pode abrir um *socket* ou utiliza algum tipo de cifragem. Técnicas de engenharia reversa também podem ser aplicadas, embora não seja uma tarefa trivial, para "desmontar" e recontruir o código-fonte do binário suspeito [14].

A última abordagem é executar o código suspeito em um ambiente devidamente preparado e controlado, de modo que seja possível rastrear a execução do programa e monitorar as alterações causadas no sistema, limitando ao máximo a possibilidade de estragos fora do ambiente preparado. Algumas etapas são necessárias para a execução do código suspeito, como listadas a seguir:

- preparar o ambiente de execução com o mesmo sistema operacional e plataforma da máquina invadida. Não utilizar o sistema de análise, muito menos a própria máquina

[28] Existe um linha de pesquisa baseada na utilização do código suspeito para a identificação do seu autor. Maiores informações sobre essa abordagem podem ser encontradas em [25].

[29] Em *sites* como o http://www.rootshell.com (disponível em julho de 2002), por exemplo.

invadida, para conduzir o teste do código suspeito. É possível utilizar o VMWare[30] com a opção *nonpersistent writes* habilitada, para impedir que o programa suspeito escreva no disco [15]. Criar um segmento de rede fechado para o ambiente de execução, instalando um *sniffer* em um sistema separado, para monitorar o tráfego de rede nesse segmento;

- produzir uma imagem (*snapshot*) da condição inicial do ambiente de execução, com assinaturas criptográficas e *mactimes* dos principais arquivos do sistema;
- executar o código suspeito e interceptar as chamadas de sistema e de biblioteca feitas pelo processo (através das ferramentas `strace` e `ltrace`) [15]. Além de executar o programa, é possível depurá-lo através de ferramentas como `adb` ou `gdb`, permitindo observar passo a passo cada ação executada pelo código suspeito;
- analisar as chamadas de sistema executadas, principalmente as funções `open`, `read`, `write`, `unlink`, `lstat`, `socket` e `close` [15], bem como as bibliotecas acessadas. Observar o estado das conexões de rede e das portas abertas no ambiente de execução (através do comando `netstat`), além dos processos em execução (por meio do comando `ps`) e os arquivos abertos por eles (através do utilitário `lsof`). Determinar as alterações ocorridas no ambiente de execução, utilizando a imagem produzida anteriormente, e avaliar os dados coletados pelo *sniffer* (se for o caso);

### Arquivos de log

Os arquivos de log potencialmente representam a fonte de informação mais valiosa sobre as atividades do sistema [14]. Tais arquivos podem registrar, entre outras informações, as atividades dos usuários, dos processos e do sistema, as conexões e atividades da rede, e informações específicas dos aplicativos e serviços. A maioria dos arquivos de log está localizada em um diretório comum, usualmente `/var/log` [15]. Os principais arquivos de log que podem ser encontrados durante a análise forense são listados na tabela 5.

Existem algumas observações importantes, acerca dos arquivos de log, que devem ser consideradas durante a análise do sistema invadido:

- os arquivos de log binários geralmente contêm mais informações do que é mostrado na saída dos programas que os manipulam [15];
- nem todos os programas registram uma entrada nos logs `utmp` e `wtmp`, de modo que podem existir mais usuários usando o sistema;
- o comando `su` não altera os logs `utmp` e `wtmp`, de modo que o usuário acessado pelo comando `su` não é listado pelos programas que utilizam esses logs (o atacante pode "logar" no sistema como um usuário sem privilégios e insuspeito, e executar o comando `su` para acessar a conta de *root*);

[30] Informações sobre o VMWare podem ser encontradas na URL http://www.vmware.com (disponível em agosto de 2002).

| arquivo de log | descrição e características |
|---|---|
| `utmp` | Registra os usuários atualmente "logados" no sistema. Apresenta−se no formato binário e pode ser acessado pelos comandos `w`, `who`, `users` e `finger`. Encontra−se no diretório `/var/run` |
| `wtmp` | Registra todos os *logins, logouts, reboots, shutdowns* e mudanças no tempo do sistema. Apresenta−se no formato binário e pode ser acessado pelos comandos `last` e `ca` |
| `btmp` | Registra as tentativas mal sucedidas de *login*. Apresenta−se no formato binário e pode ser acessado pelo comando `lastb` |
| `lastlog` | Registra o momento e origem do *login* mais recente de cada usuário. Apresenta−se no formato binário e pode ser acessado pelo comando `lastlog` |
| `boot.log` e `dmesg` | Registram as mensagens relativas ao processo de inicialização do sistema. Apresentam−se no formato texto |
| `messages` ou `syslog` | Registra vários eventos e informações do sistema e aplicativos. Representa o principal arquivo de log do sistema e geralmente contém informações encontradas também em outros arquivos de log, como, por exemplo, tentativas mal sucedidas de *login* |
| `secure` | Registra mensagens privadas de programas relativos a autorização de usuários. Apresenta−se no formato texto |
| `sulog` | Registra o uso do comando `su` |
| `access_log` | Registra os acessos ao servidor HTTP. Apresenta−se no formato texto |
| `xferlog` | Registra os acessos ao servidor FTP. Apresenta−se no formato texto |
| `cron` | Registra a execução de tarefas agendadas. Apresenta−se no formato texto |
| `maillog` | Registra mensagens relativas ao serviço de correio eletrônico. Apresenta−se no formato texto |
| `aculog` | Registra o uso de conexões *dial−out* |
| `pacct` ou `acct` | Registram os processos executados por cada usuário. Apresentam−se no formato binário e podem ser acessados pelos comandos `lastcomm`, `acctcom` e `sa` |
| *history files* (`.history`, `.sh_history`, `.bash_history`) | Registram os comandos recentemente executados por cada usuário. Apresentam−se no formato texto e encontram−se nos diretório de cada usuário |
| logs de *firewalls* e sistemas de detecção de intrusão | Registram conexões permitidas e/ou rejeitadas e eventos caracterizados como possíveis intrusões |

Tabela 5: Principais arquivos de log encontrados durante a análise forense.

- os arquivos de log podem ter nomes e funcionalidades diferentes do apresentado na tabela 5, ou podem estar localizados em outras partes do sistema, em outras máquinas ou em cópias de segurança. É importante consultar o administrador do sistema e o arquivo de configuração do serviço *syslog*[31] (`/etc/syslog.conf`), para entender a organização do esquema de log do sistema invadido;

- a capacidade de log de alguns serviços ou aplicativos pode estar desabilitada, ou pode estar configurada para um nível de detalhes insuficiente;

- os arquivos de log podem sofrer alterações deliberadas, principalmente aqueles armazenados em formato texto. Se o atacante conseguir acesso de *root* no sistema, ele pode facilmente modificar os arquivos de log, removendo, alterando ou acrescentando entradas nos arquivos [15];

- a falta de autenticação no uso do *syslog* permite que qualquer usuário crie mensagens de log [12]. Essa característica abre a possibilidade de inserção de mensagens falsas nos arquivos de log mantidos pelo *syslog*;

- o *syslog* utiliza um mecanismo não confiável para o envio de mensagens de log para uma máquina remota (protocolo UDP), de modo que não há como determinar se a mensagem de log foi realmente recebida pela máquina remota [6];

- a falta de sincronização dos relógios dos sistemas que utilizam o *syslog* pode modificar totalmente a ordem dos eventos registrados. O *syslog* marca as entradas de log recebidas com a data e a hora da máquina onde a entrada será armazenada, ao invés dos tempos da máquina que produziu a mensagem de log. Além disso, mesmo que as mensagens de log sejam armazenadas localmente, o *syslog* pode introduzir algum atraso entre o tempo em que a mensagem é gerada e o momento em que ela é gravada no respectivo arquivo de log [6];

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

- remoção do arquivo ou redirecionamento para `/dev/null`;

- indicações de modificações deliberadas nos arquivos de log, como, por exemplo, a primeira entrada de log com data posterior à data esperada para início do serviço de log, períodos de tempo sem qualquer registro, ausência de alguma mensagem de log específica (de algum serviço que certamente deveria estar gerando mensagens de log ou uma mensagem de *logout* sem o registro de *login* correspondente, por exemplo), registros com alguma informação inconscistente ou ausente, ou ainda alguma desordenação nos tempos cronológicos das entradas (as entradas devem ter sempre um incremento maior ou igual a zero nas marcas de tempo);

[31] O *syslog* é uma facilidade que utiliza um processo de log centralizado chamado `syslogd`. Os programas individuais que desejam registrar mensagens de log enviam suas mensagens para o processo `syslogd`. Tais mensagens podem, então, ser armazenadas em vários arquivos locais ou em sistemas remotos, dependendo da configuração do serviço *syslog*. Maiores informações sobre o *syslog* podem ser encontradas em [12] ou na *man page* do programa `syslogd`.

- atividades em horários ou datas não usuais (datas e horários sem expediente, por exemplo);
- atividades não usuais (podendo ser apenas visíveis ao administrador do sistema);
- *logons* de origens suspeitas ou não usuais (provenientes de fora da organização, de outros países ou de origens conhecidamente mal intencionadas);
- *logons* na conta de *root* provenientes de origens remotas (alguns sistemas são configurados para recusar *logons* na conta de *root* que não sejam provenientes do console).
- tentativas mal sucedidas de *login* (principalmente se forem várias);
- uso não autorizado ou mal sucedido do comando `su` (principalmente tentando acessar a conta de *root*);
- tentativas de acessar o arquivo `/etc/passwd` (muitos *exploits* remotos tentam obter uma cópia do arquivo de senhas);
- erros provenientes de serviços de rede (a maioria dos serviços de rede produzem mensagens de erro quando são comprometidos pelos atacantes [14]);
- *reboots* e *shutdowns* suspeitos (datas e horários imprevistos ou não usuais);
- alterações indevidas na data e hora do sistema;
- entradas de log contendo caracteres não usuais, embaralhados ou não legíveis, fora do padrão normal, podem indicar tentativas de ataques de *buffer overflow*. Além disso, entradas contendo NOPs e comandos (algum *shell*, por exemplo) também podem estar relacionadas com esse tipo de ataque;
- mensagens de serviços que deveriam estar desabilitados (conexões com o serviço `telnet`, por exemplo);
- conexões de rede provenientes de origens desconhecidas ou suspeitas;
- conexões no intervalo de tempo em que se suspeita ter acontecido a intrusão;
- transmissão de arquivos suspeitos via FTP;
- requisições DNS referentes a endereço não pertencente ao domínio do servidor;
- mensagens de pacotes rejeitados pelo *firewall*;
- mensagens de advertência produzidas pelo sistema de detecção de intrusão;

Algumas circunstâncias suspeitas podem ser encontradas exclusivamente nos registros de comandos executados pelos usuários (*history files* e `pacct`), como, por exemplo:

- obtenção de informações sobre o sistema (execução de comandos como `ps`, `ifconfig`, `netstat`, `what` e visualização de arquivos de configuração);

- remoção ou adição não autorizada de usuários;
- *download* de arquivos suspeitos ou provenientes de origens suspeitas;
- criação ou acesso a arquivos e/ou diretórios com nomes pouco comuns;
- compilação e/ou instalação de programas;
- execução de programas não autorizados ou desconhecidos;
- remoção de arquivos ou diretórios sensíveis (arquivos de log ou de configuração, por exemplo);

A análise dos arquivos de log representa um verdadeiro desafio à medida que eles crescem substancialmente. Existem ferramentas automatizadas que permitem varrer um arquivo de log em busca de padrões específicos definidos pelo usuário. Como exemplo desse tipo de ferramenta, pode ser citado o `swatch`[32], que busca por padrões definidos em seu arquivo de configuração, no formato de expressões regulares da linguagem Perl. Maiores informações sobre a ferramenta `swatch` podem ser encontradas em [12]. Os arquivos de log binários, listados na tabela 5, podem ser facilmente analisados com a ajuda dos respectivos utilitários apresentados na referida tabela. Maiores detalhes sobre tais utilitários podem ser encontrados nas respectivas *man pages*.

# 4 Correlação de evidências

A recontrução dos eventos e formulação de conclusões acerca de um incidente muitas vezes requer mais do que a simples identificação de evidências isoladas nas diversas fontes de informações do sistema comprometido. Na maioria dos casos é preciso correlacionar as informações extraídas do sistema invadido, quer seja para corroborar uma suspeita ou identificar novas pistas, levando a um maior entendimento sobre o incidente.

O relacionamento entre duas ou mais informações pode ser estabelecido segundo vários parâmetros como, por exemplo, registros de tempo, relações de causa e efeito, e números de identificação (PID, UID, GID, endereços IP e portas de serviços de rede, entre outros). Por esse motivo, é importante que as evidências encontradas no sistema comprometido sejam descritas minuciosamente, contendo os dados necessários para um possível correlacionamento.

A construção de uma linha de tempo e de gráficos relacionais pode ajudar o investigador a visualizar com maior clareza a ordem dos eventos e suas relações, permitindo a identificação de padrões e a descoberta de novas evidências. Muitas vezes os processos de busca e correlacionamento de evidências se misturam, de modo que a descoberta de uma evidência gera informações necessárias para a busca de outras.

[32] A ferramenta `swatch` pode ser encontrada na URL ftp://coast.cs.purdue.edu/pub/tools/swatch (disponível em outubro de 2002).

# 5 *Framework* do processo de investigação forense

O processo de investigação forense, seja para fins judiciais ou corporativos, deve garantir a autenticidade e integridade das evidências coletadas e dos resultados produzidos [5]. Em outras palavras, a investigação forense deve assegurar que as informações obtidas existem nas evidências analisadas e não foram alteradas ou contaminadas pelo processo de investigação. Isso pode ser particularmente difícil em se tratando de evidências digitais, devido ao seu alto grau de volatilidade. O simples ato de ligar ou desligar o computador pode alterar ou destruir evidências. Por esse motivo, é importante que a investigação seja conduzida de maneira metódica, bem organizada e em sintonia com a tecnologia envolvida [5].

Desse modo, para suportar os resultados da investigação forense, são necessários procedimentos e protocolos que assegurem que as evidências são coletadas, preservadas e analisadas de maneira consistente, minuciosa e livre de contaminações [6]. Tais características são essenciais para evitar erros durante o processo de investigação, para assegurar que as melhores técnicas disponíveis são utilizadas e para aumentar a probabilidade de dois investigadores chegarem às mesmas conclusões ao examinarem as mesmas evidências [6]. Além disso, os procedimentos e protocolos de análise forense devem ser detalhados, documentados, revisados e aceitos pela comunidade científica relevante [17]. Tais procedimentos são comumente definidos pelo termo *Standard Operating Procedures* (SOP), como sendo um guia prático, documentado, de controle de qualidade, que deve ser aplicado toda vez que um sistema é analisado [6, 18].

Com o objetivo de garantir que as evidências digitais sejam coletadas, preservadas e examinadas de maneira a resguardar sua autenticidade e integridade, os procedimentos e protocolos usados no processo de investigação forense devem ser coerentes com um conjunto de princípios legais e técnicos, que representam conceitos e práticas importantes entre os profissionais da área [17, 18]. Alguns princípios e boas práticas são sugeridos pela Associação de Oficiais Chefes de Polícia do Reino Unido (ACPO) [1] e pelo Grupo de Trabalho Científico em Evidências Digitais (SWGDE)[33] [18]. Alguns desses princípios, entre outros, são apresentados como segue:

- as ações tomadas durante a investigação forense não devem alterar as evidências;
- qualquer ação que tenha o potencial de alterar, danificar ou destruir qualquer aspecto da evidência original deve ser conduzida por uma pessoa qualificada (por exemplo, quando a evidência original precisa ser acessada para coleta de informações voláteis ou para a criação de imagens);
- o investigador não deve confiar cegamente no sistema invadido, nem nos programas e bibliotecas dinâmicas nele encontrados;
- cópias das evidências originais devem ser produzidas e, sempre que possível, a investigação deve ser conduzida sobre as cópias. Tais cópias devem ser idênticas às evidências originais, contendo toda a informação em seu estado original;
- todas as evidências digitais coletadas e as cópias produzidas devem ser autenticadas por meio de assinaturas criptográficas, permitindo a verificação de sua integridade;

[33] Maiores informações sobre o SWGDE podem ser encontradas na URL http://www.for-swg.org/swgdein.htm (disponível em agosto de 2001).

- toda evidência coletada deve ser identificada, contendo o número do caso investigado, uma breve descrição da evidência e a data e horário da coleta;
- toda evidência coletada deve ser preservada em local de acesso controlado e livre de alterações;
- todas as informações relativas à investigação (atividades relacionadas à aquisição, armazenamento, análise ou transferência das evidências, anotações e observações do investigador e os resultados produzidos) devem ser documentadas de maneira permanente e devem estar disponíveis para revisão. A documentação das ações executadas e dos resultados obtidos deve ser feita em relatórios minuciosos, contendo detalhes que incluem as versões das ferramentas utilizadas, os métodos empregados para coleta e análise das evidências e explicações que fundamentam a utilização desses métodos. Desse modo, outro investigador deve ser capaz de examinar as informações documentadas e chegar às mesmas conclusões;
- a cadeia de custódia das evidências coletadas deve ser mantida, documentando a jornada completa de cada evidência durante a investigação. Devem ser relatadas, entre outras informações, o nome da pessoa que coletou a evidência, como, onde e quando foi feita a coleta, o nome do investigador que está de posse da evidência, data e horário de retirada e devolução da evidência e as atividades executadas pelo investigador;
- as ferramentas usadas na investigação (*hardware* e *software*) devem ser amplamente aceitas na área e testadas para garantir sua operação correta e confiável. Além disso, elas devem ser conhecidas em cada detalhe para evitar implicações inesperadas na evidência analisada;
- os procedimentos devem ser aceitos pela comunidade científica relevante ou suportados por demonstrações da precisão e confiabilidade das técnicas aplicadas;
- os procedimentos devem ser revistos periodicamente para garantir sua contínua adaptabilidade e eficácia em relação às evoluções tecnológicas;
- o investigador deve ser responsável pelos resultados da investigação e pelas evidências enquanto estiverem em sua posse;
- a pessoa responsável pela investigação deve assegurar o cumprimento dos procedimentos e protocolos estabelecidos;

O *framework* do processo de investigação forense, devidamente elaborado, deve fornecer um guia passo a passo para se conduzir a investigação do sistema invadido [27]. Entretanto, existem múltiplas variantes que tornam uma investigação particularmente diferente das outras, como, por exemplo, a tecnologia envolvida, as configurações do sistema, as condições em que o sistema é encontrado e restrições impostas à investigação. Desse modo, o *framework* da análise forense deve ser definido em linhas gerais, permitindo que o investigador possa utilizá-lo, em todo ou em parte, como um roteiro na grande maioria das investigações. Nesse sentido, o *framework* geral do processo de investigação forense pode ser definido como segue:

- considerações e inteligência preliminares;
- planejamento;
- estabilização do sistema e decisões iniciais;
- coleta, autenticação, documentação e preservação de material para análise;
- análise;

Como as chances de sucesso aumentam conforme o processo de investigação é melhor definido [27], alguns procedimentos específicos utilizados no *framework* geral podem ser detalhados, bem como o uso de determinadas ferramentas e técnicas. Existem pesquisas no campo da forense computacional que visam o desenvolvimento de procedimentos e protocolos de análise forense em sistemas computacionais, segundo uma abordagem hierárquica (proposta em [17] e extendida pelos autores deste trabalho em [19, 20]), onde no topo da hierarquia estão os princípios e boas práticas que o processo de investigação deve obedecer, seguidos do *framework* geral da análise forense, e na base da hierarquia encontram-se os procedimentos detalhados sobre o uso de ferramentas e técnicas específicas. As publicações [19, 20, 21] são frutos de uma linha de pesquisa dos autores deste trabalho, paralela ao estudo aqui abordado. Nesses artigos, são discutidos o desenvolvimento e padronização de procedimentos e protocolos de análise forense, segundo a abordagem hierárquica resumida anteriormente.

As diversas fases do *framework* geral do processo de investigação forense são discutidas na sequência.

## 5.1 Considerações e inteligência preliminares

Antes de abordar o sistema computacional comprometido (ou suspeito de comprometimento), é preciso considerar uma série de questões e fazer um trabalho inicial de inteligência [5]. Algumas dessas considerações e tarefas de inteligência preliminares são apresentadas como segue:

- entrar em contato com o administrador do sistema, que comumente é a pessoa mais indicada a responder questões sobre o sistema e, geralmente, é o primeiro a perceber e relatar o incidente;
- entender o problema. A compreensão acerca dos motivos e suspeitas que originaram a investigação pode ser vital para o andamento do processo;
- compreender as condições iniciais em que o sistema será entregue para a investigação. Se o sistema já foi desligado ou encontra-se em operação, se foram tomadas medidas de contenção e resposta a incidentes e se foi coletado algum tipo de informação, são questões comumente abordadas neste momento;
- conhecer as políticas de segurança e privacidade aplicadas. É importante identificar as responsabilidades sobre cada parte do sistema e as relações de propriedade sobre as informações nele contidas. Desse modo, o investigador sabe a quem recorrer em busca de esclarecimentos ou permissões de acesso. Além disso, é importante observar questões relacionadas ao controle de acesso, monitoramento e registros de log;

- determinar se alguma lei ou direito individual pode ser violado. Algumas informações podem estar protegidas por leis rigorosas de privacidade, havendo necessidade de cuidados extremos em relação ao acesso ou disponibilização acidental desse tipo de informação. Em alguns casos, principalmente em investigações criminais, um mandado de busca e apreensão far-se-á necessário para a realização da investigação [5]. Nesse caso, a investigação deve observar rigorosamente o disposto no mandado;

- conhecer as premissas e restrições impostas à investigação, como, por exemplo, impossibilidade de desligamento do sistema ou remoção de qualquer componente físico e cuidados com qualquer interrupção no fornecimento dos serviços;

- levantar informações sobre o alvo da investigação como, por exemplo, o tipo de equipamento e tecnologia envolvidos, tipo e versão do sistema operacional, conexões e topologia da rede, serviços e configurações implementados e dados sobre os usuários do sistema;

## 5.2 Planejamento

Com base nas informações adquiridas na etapa anterior, a investigação pode ser planejada com o maior nível de detalhes possível. Entre as principais atividades da etapa de planejamento estão as seguintes:

- definir a melhor abordagem para a investigação, identificando as principais atividades que precisarão ser executadas (considerando as condições iniciais em que o sistema será entregue para a investigação);

- preparar o conjunto de ferramentas e o sistema de análise com a mais adequada configuração de *hardware* e *software*. Pode ser necessário gerar uma mídia removível (CDROM ou disquete, por exemplo) contendo o conjunto de ferramentas, permitindo sua utilização no sistema comprometido. É importante que as ferramentas e o sistema de análise sejam confiáveis, que tenham sua integridade checada e seu funcionamento testado. No caso de utilitários que fazem uso de bibliotecas dinânicas, estas devem fazer parte do conjunto de ferramentas ou devem ser compiladas juntamente com os programas (desse modo, apenas o sistema operacional da máquina suspeita é utilizado pelas ferramentas de análise, no caso da coleta de informações voláteis);

Na maioria das ocasiões, uma operação urgente e imediata é necessária, com o objetivo de reunir o maior número de informações sobre o incidente e diminuir os estragos causados pelo atacante [22]. Desse modo, um planejamento detalhado e considerações preliminares sobre o incidente, embora importantes para o processo de análise forense, podem não ser possíveis [22]. Isso reforça a necessidade de adoção de políticas de resposta a incidentes, bem como do treinamento e manutenção de times de especialistas responsáveis pelas primeiras medidas quando da ocorrência de uma invasão [15].

## 5.3 Estabilização do sistema e decisões iniciais

Esta etapa só se aplica se o sistema comprometido ainda estiver em funcionamento. Com a máquina ainda em operação, o investigador pode deparar-se com situações do tipo:

- o atacante ainda se encontra no sistema;
- processos em execução ou conexões abertas foram deixadas pelo atacante, representando evidências importantes para a investigação;
- o atacante preparou armadilhas (*booby traps*) que visam a destruição das evidências deixadas ou a danificação do sistema;

Desse modo, em um primeiro contato com o sistema suspeito, o investigador deve estabilizar a condição inicial da máquina para a etapa seguinte de coleta de informações, com o objetivo de preservar o máximo de evidências e proteger os sistemas e dados fora do escopo da investigação. Nesse momento, algumas decisões importantes devem ser tomadas, por exemplo, com relação ao método de coleta de informações mais adequado, à necessidade de coleta de dados voláteis (conteúdo da memória e informações sobre o estado do sistema operacional, por exemplo), ao método de desligamento do sistema (caso seja necessário e possível), à necessidade de coleta de tráfego de rede e possibilidade de rastreamento do atacante (*backtracking*) e à necessidade de remoção do sistema para um ambiente controlado de análise. A necessidade e método de execução dessas e outras atividades são melhor definidos após uma análise inicial do sistema em funcionamento e, por esse motivo, são discutidos nesta etapa do *framework* da análise forense. É importante, também, descrever o sistema suspeito em sua condição inicial e tomar nota de todas as atividades executadas.

Decidir entre manter a máquina em funcionamento, desligá-la imediatamente através da interrupção do fornecimento de energia, ou proceder o desligamento administrativo normal do sistema é uma das questões mais amplamente discutidas no campo da forense computacional [14]. Muitos investigadores alegam que o desligamento imediato pelo cabo de energia é a melhor opção para congelar o sistema em seu estado corrente, considerando aceitável que algumas evidências sejam perdidas em face à preservação da integridade daquelas presentes nos dispositivos de armazenagem secundária [6, 14, 27].

O principal argumento a favor dessa abordagem é que qualquer erro cometido em um sistema em funcionamento não pode ser remediado, de modo a desfazer todas as suas consequências, que podem incluir, por exemplo, a alteração ou destruição de informações fundamentais à investigação [14]. Além disso, a manutenção do sistema em funcionamento e o seu desligamento administrativo normal podem expor o investigador a situações potencialmente catastróficas, como, por exemplo:

- muitos atacantes instalam armadilhas que destroem seus rastros ou até mesmo o sistema todo, quando o mesmo é desligado. No ambiente UNIX existem diversos arquivos envolvidos no processo de desligamento que podem ser facilmente alterados [27];
- o próprio processo de desligamento pode alterar ou remover arquivos como parte do procedimento normal;
- um atacante ainda no sistema pode desconfiar da investigação e iniciar um limpeza das evidências deixadas (podendo até danificar o sistema);

Por outro lado, o desligamento imediato pelo cabo de energia, antes da coleta das informações voláteis, pode resultar em uma grande perda de evidências (principalmente aquelas

associadas a um ataque em andamento), que talvez possam ser encontradas apenas enquanto o sistema estava em operação [6, 14]. Além disso, um desligamento abrupto do sistema pode corromper dados importantes ou danificar algum dispositivo físico [27]. Em muitos casos, ainda, o desligamento do sistema pode não ser permitido pela gerência da organização (evidentemente essa situação não se aplica no cumprimento de um mandado judicial de busca e apreensão).

É importante ressaltar que qualquer atividade executada sobre um sistema suspeito não deve utilizar os programas nele encontrados (com excessão do sistema operacional, com o risco deste também estar comprometido), pois o atacante pode ter antecipado uma possível investigação e alterado alguns dos executáveis do sistema [14].

Conforme o apresentado, cada abordagem relacionada ao desligamento do sistema apresenta vantagens e desvantagens, que devem ser ponderadas pelo investigador de acordo com situações particulares da investigação em questão. O importante é conhecer as implicações de cada abordagem e manter-se flexivel quanto à utilização de cada uma.

Igualmente dependente de situações particulares da investigação em questão, é a decisão sobre o que fazer quando se descobre que o intruso ainda está presente no sistema. Basicamente existem três opções a considerar [27]: o investigador pode mantê-lo no sistema para coletar informações e iniciar um rastreamento (maiores detalhes sobre *backtracking* podem ser encontrados em [15, 27]), a conexão do atacante pode ser interrompida (por *software* ou pela desconexão do cabo de rede) ou o investigador pode tentar repreender o intruso a deixar o sistema e não mais retornar.

Esta etapa do *framework* do processo de investigação envolve decisões importantes que afetarão na escolha do método a ser utilizado para a coleta e análise das informações do sistema suspeito, de acordo com o nível de esforço empregado para proteger as evidências e evitar a execução de código hostil. Essa relação é resumida na tabela 6.

## 5.4 Coleta, autenticação, documentação e preservação de material para análise

Existem dois grandes perigos ao se coletar material para análise: perda e alteração. Se os cuidados necessários não forem tomados, dados importantes podem ser sobrescritos e perdidos totalmente, ou apenas parte deles, alterando seu significado ou apagando informações críticas [27]. Uma coleta conduzida de maneira inadequada pode comprometer totalmente a investigação forense, em particular aquela com fins judiciais [27]. Desse modo, alguns aspectos fundamentais devem ser considerados durante a etapa de coleta de material para análise:

- tratar cada informação como se ela fosse ser usada para fins judiciais;
- coletar o máximo de material possível, observando sua ordem de volatilidade. Uma vez terminada a etapa de coleta, geralmente não há retorno, de modo que algumas das informações inicialmente consideradas desnecessárias podem ter desaparecido [14] (na dúvida é melhor coletar). O termo "material" é usado no sentido amplo de "tudo que pode ser coletado", seja fisicamente tangível ou não, como por exemplo, informações digitais, *hardware*, documentação, configuração das conexões externas da máquina (cabos de rede, impressoras e outros dispositivos externos), anotações e *post-its*;
- autenticar, identificar, catalogar e preservar cada item coletado;

| método | vantagens | desvantagens |
|---|---|---|
| Usar uma estação forense dedicada para examinar o disco suspeito (protegido contra escrita) ou uma imagem do mesmo | Não requer preocupação quanto a validade do *software* ou *hardware* da máquina suspeita | Pode depender do desligamento do sistema suspeito. Pode resultar na perda de informações voláteis |
| Inicializar a máquina comprometida usando um *kernel* e ferramentas verificados e protegidos contra escrita, contidos em uma mídia removível | Conveniente e rápido. Nesse caso, os discos da máquina suspeita devem ser montados somente para leitura | Assume que o *hardware* da máquina suspeita não foi comprometido (o que é raro). Resulta na interrupção dos serviços disponibilizados pelo sistema suspeito e pode causar a perda de informações voláteis |
| Reconstruir o sistema suspeito a partir de sua imagem e, então, examiná−lo | Recria completamente o ambiente operacional do sistema suspeito, sem o risco de alterar as informações originais | Requer disponibilidade de *hardware* idêntico ao do sistema suspeito. Pode resultar na perda de informações voláteis |
| Examinar o sistema suspeito através de mídias removíveis contendo ferramentas verificadas | Conveniente e rápido. Permite acessar informações voláteis | Se o *kernel* estiver comprometido, os resultados podem ser inconcistentes |
| Verificar o *software* contido no sistema suspeito e, então, utilizá−lo para conduzir o exame | Requer uma preparação mínima. Permite acessar informações voláteis e pode ser realizado remotamente | Pode tomar muito tempo e o programa usado para verificação de integridade pode estar comprometido. A falta de proteção contra escrita nos discos do sistema suspeito pode resultar na alteração ou destruição de informações |
| Examinar o sistema suspeito usando o *software* nele contido, sem qualquer verificação | Requer nenhuma preparação. Permite acessar informações voláteis e pode ser realizado remotamente | Método menos confiável e representa exatamente aquilo que os atacantes esperam que seja feito. Na maioria das vezes é uma completa perda de tempo |

Tabela 6: Relação entre o método de coleta e análise e o nível de esforço empregado para proteger as evidências e evitar a execução de código hostil.

- produzir cópias exatas e autenticadas das informações digitais coletadas;

A documentação dos itens coletados é uma das atividades de grande importância realizadas durante a etapa de coleta de material para análise. Cada item coletado deve ser unicamente identificado e minuciosamente descrito em seu estado original. Além disso, essa descrição deve identificar a localização original do item, data e hora da coleta e a identificação da pessoa responsável. Não menos importante é a atividade de transporte e armazenamento dos itens coletados, que deve zelar pela integridade dos mesmos, tomando especial cuidado, por exemplo, com campos magnéticos, quedas e acessos não autorizados.

A autenticidade e integridade das informações digitais coletadas pode ser estabelecida através de assinaturas criptográficas, como o MD5 e SHA [12, 23]. É possível determinar que uma informação digital coletada é autêntica, através da simples comparação do seu *hash* criptográfico (produzido, por exemplo, pelo comando `md5sum`) com a assinatura criptográfica da informação original (produzida de maneira idêntica à assinatura da cópia) [14]. Entretanto, muitas vezes essa comparação não é possível, pois a informação original não existe mais

ou foi alterada (principalmente no caso de informações voláteis ou quando o sistema suspeito não pôde ser desligado). Por outro lado, o *hash* criptográfico das informações digitais coletadas, produzido no momento da coleta (tal procedimento é exemplificado adiante), permite provar, a qualquer momento, que os dados usados durante a análise são idênticos às informações inicialmente coletadas [14].

A coleta de informações digitais de um sistema suspeito pode ser tecnicamente desafiadora, mas o procedimento básico pode ser resumido como segue: coletar as informações voláteis, desligar a máquina suspeita, instalar o disco da máquina suspeita no sistema de análise e produzir sua imagem [5]. Como visto anteriormente, nem todas essas atividades podem ser executadas, de modo que a coleta de informações pode apresentar variações de acordo com situações particulares da investigação em questão.

Durante a coleta de informações voláteis, o sistema suspeito precisa ser acessado ainda em funcionamento. Desse modo, é importante observar duas questões fundamentais: não utilizar os programas contidos no sistema suspeito e não escrever no disco da máquina investigada. A primeira questão pode ser abordada através da utilização de um conjunto de ferramentas confiáveis, compiladas estaticamente e testadas, contidas em algum tipo de mídia removível (protegida contra escrita) que é montada na máquina suspeita. Essa abordagem considera que o sistema operacional da máquina suspeita não foi comprometido pelo atacante, o que pode ser investigado, através das técnicas apresentadas na seção 3.5, antes de se proceder à coleta das informações. A segunda questão pode ser abordada pelo redirecionamento da saída dos comandos executados, para algum tipo de mídia removível ou para a rede. A transmissão das informações pela rede pode ser feita através do comando `nc` (*netcat*[34]), o que é executado em dois passos. No primeiro passo, uma máquina (a estação forense) deve ser preparada para receber as informações, executando o comando `nc` com as opções `-l` e `-p`:

```
# nc -l -p 2222 | tee ps.out | md5sum -b > ps.out.md5
```

Nesse caso, a estação forense é configurada para receber as informações pela porta 2222 e armazená-las no arquivo `ps.out`. É utilizada uma combinação dos comandos `tee` e `md5sum`, que permite armazenar as informações recebidas e gerar sua assinatura criptográfica no mesmo instante. O comando `tee` simplesmente recebe um fluxo de dados pela entrada padrão e o armazena em um arquivo, replicando de maneira exata o fluxo de dados recebido para a saída padrão. Maiores detalhes sobre os comandos envolvidos no exemplo anterior pedem ser encontrados nas respectivas *man pages*.

No segundo passo, o comando desejado é executado no sistema suspeito e sua saída é redirecionada para o programa `nc`, informando o endereço IP da estação forense e o número da porta configurada para receber os dados (no exemplo a seguir, os programas confiáveis são acessados a partir de um CDROM):

```
# /mnt/cdrom/tookit/ps -aux | /mnt/cdrom/tookit/nc 10.1.1.1 2222
```

É importante observar que a assinatura criptográfica gerada na estação forense (pela combinação dos comandos `tee` e `md5sum`), ao receber as informações provenientes do comando

[34] O programa *netcat* e maiores detalhes sobre o mesmo podem ser encontrados na URL http://www.netcat.org (disponível em agosto de 2002).

executado na máquina analisada, permite apenas provar, futuramente, a integridade dos dados armazenados na máquina de análise (no arquivo `ps.out`, no caso do exemplo). A autenticidade desses dados só pode ser garantida se tal assinatura criptográfica for comparada com o *hash* criptográfico dos dados produzidos pelo comando executado no sistema suspeito. A geração do *hash* criptográfico dos dados enviados para a estação forense deve ser feita no momento em que as informações são coletadas pelo comando executado, devido à alta volatilidade dessas informações (uma nova execução do comando provavelmente irá gerar informações diferentes, resultando em um *hash* criptográfico completamente distinto). Isso pode ser feito através do mesmo princípio utilizado pelo comando `tee`, com uma pequena alteração – o fluxo de dados recebido pela entrada padrão não pode ser armazenado em um arquivo (não se pode escrever no disco da máquina investigada), ao invés disso, o fluxo de dados deve ser direcionado para outro comando (no caso o comando responsável por gerar o *hash* criptográfico) e replicado exatamente para a saída padrão. Os autores deste trabalho implementaram, na linguagem Perl, um *script* simples, chamado `tee_command.pl`[35], que modifica o comando `tee` da maneira desejada, podendo ser usado na máquina suspeita da seguinte forma:

```
# /mnt/cdrom/tookit/ps -aux | /mnt/cdrom/tookit/perl \
  /mnt/cdrom/tookit/tee_command.pl ‘‘/mnt/cdrom/tookit/nc 10.1.1.1 2222’’ \
  | /mnt/cdrom/tookit/md5sum -b
```

Essa abordagem permite ainda provar que a trasmissão dos dados pela rede não introduziu qualquer tipo de alteração nas informações coletadas. Para a geração das assinaturas criptográficas, é importante que o fluxo de dados seja sempre o mesmo, desde a coleta das informações (pela execução do respectivo comando no sistema suspeito) até seu armazenamento na estação forense, como ilustrado pela figura 3.

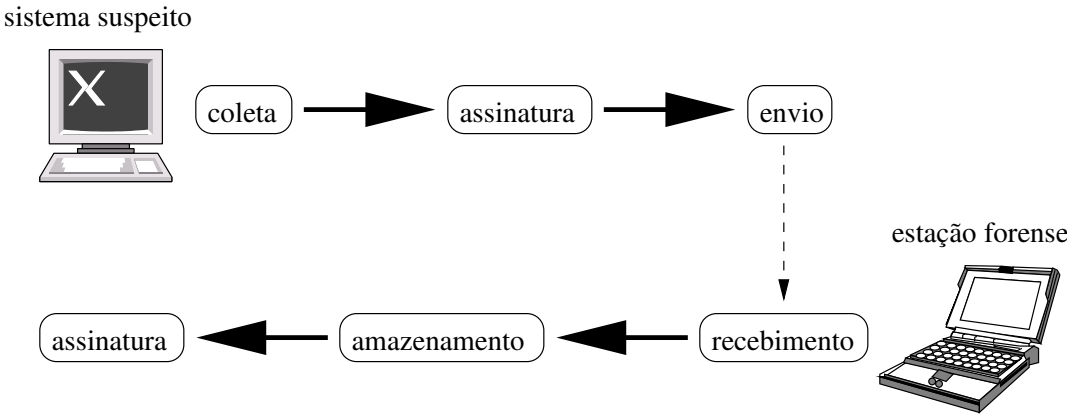


Figura 3: Fluxo das informações voláteis desde a coleta no sistema suspeito até seu armazenamento na estação forense.

Outro procedimento importante da etapa de coleta de material para análise é a produção de imagens dos discos da máquina comprometida. O princípio por trás do procedimento de imagem

[35]O *script* `tee_command.pl` pode ser obtido na URL http://www.ic.unicamp.br/ra000504 (disponível em outubro de 2002).

é a obtenção de toda informação contida no disco, seja ela pertencente a um sistema de arquivos ou não, de tal forma que a imagem possa ser examinada como se o disco original estivesse sendo analisado [22]. Para produzir uma image, geralmente se utiliza alguma ferramenta que lê cada bit do disco suspeito, do início ao fim, e cria um arquivo de imagem contendo todo o fluxo de bits lido, sem qualquer alteração na ordem ou conteúdo [22]. O arquivo de imagem, dependendo da situação, pode ser dividido em arquivos menores, contendo separadamente cada partição do disco suspeito, ou ainda pode ser usado para espelhar o disco suspeito em um outro disco de capacidade similar ou maior [22]. Essa segunda abordagem, chamada de espelhamento, permite reconstituir exatamente[36] o disco investigado, de modo que os dados contidos em um determinado setor do disco suspeito aparecem no mesmo setor do disco espelhado [22].

É importante frisar que as imagens devem conter cada bit do disco de origem e, portanto, programas normais de cópia e *backup* (como `cp` e `tar`, por exemplo) não devem ser usados, pois eles copiam apenas os dados reconhecidos pelo sistema de arquivos [6]. Existem diversos utilitários que podem ser usados para a produção de imagens, inclusive equipamentos especiais dedicados à duplicação de discos (como, por exemplo, o Image MaSSter Solo Forensic Unit[37]). O comando `dd` pode ser usado com facilidade e em diversas situações diferentes. O procedimento mais comumente utilizado é a instalação do disco suspeito na estação forense, onde a imagem é produzida e armazenada em um arquivo:

```
# dd if=/dev/hdc of=suspect_img.dd
19920600+0 records in
19920600+0 records out
```

No exemplo anterior, o disco suspeito é acessado na estação forense através do *device file* `/dev/hdc` e a imagem é armazenada no arquivo `suspect_img.dd`. Uma prática recomendada é a utilização de nomes, para os arquivos de imagens, que identifiquem precisamente o disco copiado. Após a geração da imagem é necessário verificar se os dados foram copiados com exatidão, o que pode ser feito através de assinaturas criptográficas:

```
# dd if=/dev/hdc | md5sum -b
19920600+0 records in
19920600+0 records out
54d3ca2fe9b2e61b6d4b35580b42b6ba *-
# dd if=suspect_img.dd | md5sum -b
54d3ca2fe9b2e61b6d4b35580b42b6ba *-
```

No exemplo anterior, as assinaturas criptográficas (geradas pelo comando `md5sum`) do disco suspeito e de sua imagem são idênticas. Devido ao grande volume de dados contido nos discos, a geração do *hash* criptográfico do disco origem pode ser feita em conjunto com a produção de sua imagem, através do comando `tee`:

[36] Devido ao mecanismo de tratamento de áreas defeituosas do disco e à transparência com que isso é feito, os dados espelhados podem não estar necessariamente na mesma localização física do disco original. Por esse motivo, o disco espelhado pode não ser considerado verdadeiramente uma cópia exata do disco original. Entretanto, essas mudanças na disposição física dos dados na superfície dos discos são invisíveis durante o acesso às informações, e portanto não têm quaisquer implicações no processo de imagem e análise dos discos [22].

[37] Maiores detalhes sobre o Image MaSSter podem ser encontrados na URL http://www.ics-iq.com (disponível em agosto de 2002).

```
# dd if=/dev/hdc | tee suspect_img.dd | md5sum -b
19920600+0 records in
19920600+0 records out
54d3ca2fe9b2e61b6d4b35580b42b6ba *-
# md5sum -b suspect_img.dd
54d3ca2fe9b2e61b6d4b35580b42b6ba *-
```

O procedimento do exemplo anterior adiciona um atraso no geração da imagem do disco suspeito, entretanto, reduz consideravelmente o tempo gasto para checar as assinaturas criptográficas, pois o disco suspeito é lido apenas uma vez.

Caso o investigador deseje produzir um arquivo de imagem para cada partição do disco suspeito, é possível utilizar as opções `skip` e `count` do comando `dd` para indicar o início e fim de cada partição:

```
# fdisk -lu /dev/hdc

Disk /dev/hdc: 255 heads, 63 sectors, 1240 cylinders
Units = sectors of 1 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1   *        63   9221309   4610623+   c  Win95 FAT32 (LBA)
/dev/hdc2       9221310  18040994   4409842+  83  Linux
/dev/hdc3      18040995  18458684    208845   82  Linux swap


# dd if=/dev/hdc of=suspect_imgINI.dd bs=512 count=63
63+0 records in
63+0 records out
# dd if=/dev/hdc of=suspect_img1.dd bs=512 skip=63 count=9221247
9221247+0 records in
9221247+0 records out
# dd if=/dev/hdc of=suspect_img2.dd bs=512 skip=9221310 count=8819685
8819685+0 records in
8819685+0 records out
# dd if=/dev/hdc of=suspect_img3.dd bs=512 skip=18040995 count=417690
417690+0 records in
417690+0 records out
# dd if=/dev/hdc of=suspect_imgFIM.dd bs=512 skip=18458685
1461915+0 records in
1461915+0 records out
```

No exemplo anterior, o disco suspeito possui três partições, que não ocupam todo o disco. A imagem de cada partição é armazenada respectivamente nos arquivos `suspect_img1.dd`, `suspect_img2.dd` e `suspect_img3.dd`. Além disso, os espaços não utilizados também são coletados, como os espaços anterior à primeira partição e posterior à terceira, armazenados respectivamente em `suspect_imgINI.dd` e `suspect_imgFIM.dd`.

Para o procedimento de espelhamento do disco suspeito, o comando `dd` pode ser usado para "esterilizar" o disco destino, eliminando qualquer informação que possa ser confundida com os dados do disco suspeito:

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

No exemplo anterior, o disco destino é acessado pelo *device file* `/dev/hdb`. A limpeza do disco destino pode ser verificada através da combinação dos comandos `xxd` e `grep`:

```
dd if=/dev/hdb | xxd | grep -v "0000 0000 0000 0000 0000 0000 0000 0000"
```

Caso a combinação de comandos anterior produza alguma saída (com excessão da saída normal do comando `dd`), o disco destino não foi devidamente "esterilizado". Depois de preparado o disco destino, o espelhamento pode ser executado de duas maneiras:

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

No primeiro caso do exemplo anterior, o espelhamento é feito diretamente do disco suspeito (`/dev/hdc`) para o disco destino (`/dev/hdb`). É preciso extrema atenção nesse caso, pois uma inversão nas opções do comando `dd` pode ser desastrosa (por esse motivo esse procedimento não é recomendado). No segundo caso, o arquivo de imagem, produzido em um momento anterior, é utilizado para gerar o disco espelhado.

Caso o sistema suspeito não possa ser desligado, as imagens dos discos podem ser feitas a partir da máquina comprometida e enviadas para a estação forense pela rede, de maneira análoga à coleta de informações voláteis descrita anteriormente. É importante observar que nesse caso, as assinaturas criptográficas dos discos suspeitos podem apresentar valores diferentes a cada vez que forem geradas, de modo que não podem ser comparadas com as assinaturas das respectivas imagens (a menos que a assinatura do disco suspeito seja gerada no momento da produção se sua imagem, por exemplo, através do *script* `tee_command.pl` descrito anteriormente).

## 5.5 Análise

A etapa de análise representa o objetivo principal do processo de investigação forense. É o momento em que todo o material coletado é minuciosamente examinado em busca de evidências, permitindo formular conclusões acerca do incidente que originou a investigação. Durante a análise, é importante investigar todas as fontes de informação do sistema suspeito, buscando identificar características anormais e indevidas, possivelmente provocadas pelo atacante. Conhecer o *modus operandi* dos atacantes e manter um contato próximo com o administrador do sistema comprometido são requisitos essenciais para a eficácia e eficiência do processo de análise, pois aumentam a capacidade do investigador de reconhecer possíveis evidências.

A documentação das tarefas executadas e evidências encontradas, bem como a atualização da cadeia de custódia dos itens analisados, devem ser atividades rotineiras durante a etapa de análise. Outra atividade importante é a correlação das evidências encontradas, permitindo, entre outras conclusões, determinar se houve realmente um incidente de segurança; reconstruir

as atividades do atacante; identificar causas, suspeitos e consequências da invasão. Com base nos resultados obtidos pela investigação, um relatório final pode ser elaborado (o laudo pericial, no caso de uma perícia criminal), servindo de base para amparar uma ação judicial ou para auxiliar na reconstituição do sistema comprometido e revisão das soluções de segurança da corporação.

# 6 Conjunto de ferramentas

Além de experiência e sólidos conhecimentos, o investigador depende totalmente do conjunto de ferramentas que ele utiliza para coletar, documentar, preservar e processar as informações provenientes do sistema investigado. No mundo real da investigação forense, o investigador deve estar preparado para lidar com as mais diversas arquiteturas e sistemas operacionais, logo, seu conjunto de ferramentas deve ser o mais amplo possível.

O conjunto de ferramentas de investigação deve conter uma estação forense, que é a máquina onde idealmente a análise das informações coletadas é realizada. Essa estação representa um ambiente controlado onde o investigador dispõe de todos os dispositivos de *hardware* e utilitários de *software* necessários para coletar, receber, preservar e analisar as informações digitais provenientes do sistema suspeito. O ideal é que seja uma máquina portátil, com alta capacidade de processamento e armazenagem de dados, com interface de rede e que permita a conexão dos mais variados tipos de periféricos e dispositivos (como, por exemplo, unidades de CDROM, *zip drives* e discos rígidos). Existem computadores especialmente desenvolvidos para a prática forense, com configurações que permitem acessar informações através de diferentes tipos de tecnologias. Um exemplo é a unidade portátil fornecida pela Forensic-Computers.com[38], ilustrada na figura 4.

Quanto ao sistema operacional da estação forense, cada investigador possui uma preferência, orientada principalmente pela familiaridade, facilidade de uso, disponibilidade de utilitários e capacidade de acesso a informações provenientes de outras plataformas. O ambiente Linux oferece suporte a uma grande variedade de sistemas de arquivos (como, por exemplo, FAT, NTFS e UFS), de modo que a possibilidade de utilização de um único conjunto de utilitários para análise de diferentes plataformas, segundo técnicas consistentes, torna a escolha do ambiente Linux quase irresistível [6]. No entanto, algumas situações requerem a utilização da mesma plataforma encontrada no sistema suspeito. É importante ter sempre disponível os programas de instalação dos principais sistemas operacionais, em suas mais variadas versões, permitindo recriar as mesmas condições de acesso às informações do sistema suspeito. Pode-se ainda instalar múltiplos sistemas operacionais na estação forense, possibilitando a escolha durante a inicialização do sistema ou através de programas como o VMWare.

Muitas vezes a coleta de informações, e até mesmo a análise, precisa ser feita diretamente no sistema comprometido. Os atacantes frequentemente alteram os programas contidos na máquina invadida e, portanto, o investigador necessita de um conjunto de utilitários confiáveis para conduzir a investigação. Uma solução é a utilização de um CDROM ISO 9660, que virtualmente qualquer plataforma pode acessar, contendo todos os utilitários (binários e *scripts*), destinados a diferentes plataformas, que o investigador possa precisar. Além disso, é preciso

[38]Maiores informações podem ser encontradas na URL http://www.forensic-computers.com (disponível em agosto de 2002).


Figura 4: Unidade portátil de investigação forense da Forensic-Computers.com.

que essas ferramentas sejam compiladas estaticamente, ou que o CDROM contenha também cópias seguras das bibliotecas dinâmicas, para evitar problemas com bibliotecas comprometidas. Existem iniciativas voluntárias no sentido de se criar um conjunto de utilitários[39], compilados estaticamente para difirentes plataformas, apropriado para a prática forense. Como sugestão de ferramentas que podem compor esse CDROM, podem ser citados os seguintes utilitários UNIX:

```
ls     strings  rpm      ps      ltrace    route   what      telnet
cp     grep     gzip     kill    uptime    rndc    lastcomm  ssh
cd     find     fdisk    gcore   netstat   lsmod   lastlog   ftp
cat    file     dd       gdb     nfsstat   kstat   perl      TCT
more   diff     xxd      top     arp       md5sum  bash      TCTUTILs
less   stat     losetup  lsof    tcpdump   pwck    tee
vi     tar      mount    strace  ifconfig  grpck   nc
```

Outra ferramenta muitas vezes necessária e que deve compor o arsenal do investigador é uma distribuição de algum sistema operacional que caiba em uma mídia removível (em alguns disquetes ou em um CDROM, por exemplo). É importante que tal distribuição não execute qualquer operação contra o dispositivo de armazenagem (criação de áreas de *swap*, por exemplo), tenha suporte à comunicação por rede e permita a criação de imagens dos discos [15]. Existem algumas distribuições compactas do sistema Linux que podem ser usadas como, por exemplo,

[39]Tal conjunto de ferramentas pode ser encontrado na URL http://www.incident-response.org (disponível em agosto de 2002).

o *Pocket-Linux*, *Tomsrtbt*, *Trinux* e *Linuxcare Bootable Business Card*[40].

No conjunto de ferramentas do investigador não pode faltar, obviamente, itens auxiliares para desmontagem das máquinas; identificação, transporte e armazenagem de materiais coletados; e documentação das atividades. Como exemplos, podem ser citados os seguintes itens: chaves para diversos tipos de parafusos, alicates, extensões e filtros de linha, cabos e conectores de rede, etiquetas e formulários, plástico bolha, embalagem contra eletricidade estática, máquina fotográfica, mídias removíveis e discos rígidos.

Independente do tipo de ferramenta à disposição do investigador, é preciso estar familiarizado com seu funcionamento e verificar se elas executam exatamente aquilo que se espera. O investigador deve ser capaz de interpretar com confiança a saída dos programas e deve estar ciente de todas as implicações que eles podem causar no sistema ou informação analisada. Praticar e testar as ferramentas é tão importante quanto adquiri-las.

Ao longo deste trabalho foram apresentados diversos utilitários, conhecidos dos usuários e administradores de sistemas Linux, que originalmente não foram desenvolvidos para o propósito único da análise forense. Um aspecto importante da prática forense é o uso de ferramentas não necessariamente destinadas a essa finalidade [14]. Entretanto, as necessidades singulares da investigação forense resultaram na criação de ferramentas especialmente desenvolvidas para tal finalidade.

No restante desta seção, algumas dessas feramentas de forense são apresentadas. Algumas delas, como o TCT e o TCTUTILs, já foram mencionadas anteriormente. Apesar do escopo deste trabalho restringir-se ao ambiente Linux, algumas ferramentas destinadas a outras plataformas também são apresentadas, devido a sua ampla utilização e relevência na área da forense computacional. É importante mencionar, ainda, que existem algumas ferramentas disponíveis apenas para mantenedores da lei e agências governamentais e, portanto, não são discutidas nesta dissertação.

## 6.1 *The Coroner's Toolkit* (TCT)

O TCT[41] é um conjunto de ferramentas de código aberto e gratuito, desenvolvido por Dan Farmer e Wietse Venema, que permite investigar um sistema UNIX comprometido. É importante mencionar que o TCT não foi inicialmene desenvolvido para apresentar evidências úteis em um processo criminal, e sim para ajudar a determinar o que aconteceu em um sistema invadido [14]. O TCT apresenta atualmente quatro partes principais:

- `grave-robber`;
- `mactime`;
- utilitários (`icat`, `ils`, `pcat`, `md5`, `timeout`);
- `unrm` e `lazarus`;

[40] Encontrados respectivamente nas URLs http://pocket-linux.coven.vmh.net, http://www.toms.net/rb/, http://trinux.sourceforge.net, http://www.linuxcare.com/bootable_cd/ (todas disponíveis em agosto de 2002).

[41] O TCT pode ser encontrado na URL http://www.porcupine.org/forensics/tct.html (disponível em agosto de 2002).

Os principais componentes do TCT são detalhados na sequência. No entanto, a maioria das ferramentas possui uma série de opções de execução, que não são discutidas neste trabalho. Além disso, alguns componentes do TCT possuem restrições quanto ao sistema de arquivos e plataforma suportados, que também não são abordadas. Maiores detalhes sobre as ferramentas do TCT, suas opções e compatibilidades podem ser encontradas nas respectivas *man pages*.

### 6.1.1 Grave-robber

A ferramenta `grave-robber` pode ser considerada a parte central de todo o TCT. O `grave-robber` é basicamente uma ferramenta de coleta de dados que executa uma série de comandos em uma tentativa de reunir informações básicas sobre o sistema e salvar alguns arquivos necessários para a análise. É importante mencionar que o `grave-robber` apenas coleta informações, não fazendo qualquer esforço no sentido de interpretá-las.

A ferramenta captura os dados de acordo com a ordem de volatilidade (conceito introduzido por Farmer e Venema), e como padrão de execução ela coleta informações efêmeras (processos e estado da rede), descobre o que puder sobre a configuração de *hardware* do sistema (especialmente sobre os discos e partições) e busca por arquivos críticos (como, por exemplo, arquivos de log e de configuração).

Antes de iniciar a coleta dos dados, o `grave-robber` examina alguns dos utilitários e arquivos mais importantes do sistema, permitindo que o investigador possa utilizá-los com segurança. Usualmente são verificados os arquivos contidos no diretório `/etc` e os utilitários comumente localizados em diretórios como `/bin` e `/usr/bin`. Essa atividade é controlada pelo arquivo de configuração `look@first` do TCT.

Embora seja um processo bastante demorado, recomenda-se executar o `grave-robber` sobre o sistema todo (passando o diretório raiz do sistema como argumento). Como saída, o `grave-robber` produz os seguintes arquivos e diretórios (toda saída produzida contém uma marca de tempo e é gerada a assinatura MD5 do arquivo produzido):

- `body`: banco de dados para o programa `mactime`, contendo os atributos dos arquivos analisados (incluindo os *mactimes*);
- `body.S`: contém os atributos de todos os arquivos SUID encontrados;
- `command_out`: sub-diretório contendo a saída dos diversos comandos executados pelo `grave-robber` como, por exemplo, o `ps` e `netstat`;
- `removed_but_running`: sub-diretório contendo arquivos que foram "deletados", mas ainda estão em execução;
- `icat`: sub-diretório contendo arquivos em execução que ainda estão no disco;
- `proc`: sub-diretório contendo arquivos copiados do diretório `/proc`;
- `conf_vault`: sub-diretório contendo os arquivos e diretórios críticos salvos pelo `grave-robber`;
- `user_vault`: sub-diretório contendo os arquivos e diretórios de usuário salvos pelo `grave-robber` (*history files* por exemplo);

- `trust`: sub-diretório contendo relações de confiança do sistema (como `.rhosts`, `.forward` e tarefas agendadas, por exemplo);
- `coroner.log`: log de execução do `grave-robber`, contendo a data e horário de execução de todos os programas invocados;
- `error.log`: log de erro do `grave-robber`;
- `MD5_all`: assinatura MD5 de tudo que é produzido como saída do programa `grave-robber`;

### 6.1.2 Mactime

A ferramenta `mactime` pode ser usada para processar o arquivo `body`, produzido pelo `grave-robber`, ou pode ser executada sobre um determinado diretório. Essa ferramenta produz uma listagem de todos os arquivos e sub-diretórios que tiveram algum de seus *mactimes* alterados a partir de uma determinada data (passada como parâmetro). Tal listagem é ordenada segundo uma linha de tempo e cada entrada corresponde à alteração de um ou mais *mactimes* (identificados pelas letras m, a, c), como, por exemplo:

```
(data      horário  tamanho  MAC  permissões  UID   GID   arquivo)
Apr 05 99  04:05:00  64092   m..  -r-xr-xr-x  root  root  /bin/ps
Apr 10 99  04:05:00  45724   m..  -rwxr-xr-x  root  root  /bin/ls
Apr 12 99  01:04:39  45724   .a.  -rwxr-xr-x  root  root  /bin/ls
Apr 12 99  14:10:15  14716   m.c  -rwxr-xr-x  root  root  /bin/cat
```

### 6.1.3 Utilitários

O TCT possui uma série de utilitários que são executados pela ferramenta `grave-robber`. Entretanto, esses programas auxiliares podem ser bastante úteis em outras situações, como já apresentado na seção 3. Alguns dos principais utilitários presentes no TCT são descritos como segue.

A ferramenta `icat` permite visualizar o conteúto de um arquivo ou diretório a partir do número de seu *inode*. Pode ser usada para recuperar o conteúdo de *inodes* não alocados. O utilitário `ils` lista as informações sobre os *inodes* de um sistema de arquivos. Em sua execução padrão o `ils` lista apenas informações sobre *inodes* de arquivos "deletados". O *script* `ils2mac` pode ser usado para converter a saída do comando `ils` para o formato do arquivo `body`, permitindo sua utilização com a ferramenta `mactime`. Outro utilitário, chamado `pcat`, permite visualizar a memória de um processo. O comando `md5` computa a assinatura MD5 de um fluxo de bits qualquer (um arquivo, por exemplo). Por fim, a ferramenta `timeout` permite a execução de um comando qualquer com uma restrição de tempo. Caso o comando desejado não termine dentro do limite de tempo estipulado, o comando é terminado.

### 6.1.4 Unrm e lazarus

A ferramenta `unrm` é uma variante do programa `dd`, sendo capaz de copiar os blocos de dados de um sistema de arquivos. Em sua execução padrão, o `unrm` extrai apenas os blocos de dados

não alocados do sistema de arquivos. Essa ferramenta é utilizada na tentativa de recuperar informações "deletadas". É importante redirecionar a saída do programa `unrm` para outro sistema de arquivos diferente do analisado. Caso contrário, o fluxo de bits gerado irá ocupar os blocos não alocados que deveriam ser extraídos, possivelmente sobrescrevendo as informações procuradas.

O resultado da execução do programa `unrm` é apenas um enorme fluxo de bits sem qualquer sentido aparente. Para tentar recontruir arquivos "deletados" ou outros tipos de dados, a partir de um fluxo de bits qualquer, o TCT possui uma ferramenta chamada `lazarus`. Esse programa toma os dados produzidos pela ferramenta `unrm` (ou qualquer outra fonte como, por exemplo, a memória e áreas de *swap*) e tenta criar alguma estrutura a partir de dados não estruturados.

Basicamente o processo de análise realizado pelo programa `lazarus` realiza os seguintes passos:

1. Leitura de um bloco de dados da entrada (tipicamente são lidos blocos de 1024 bytes);

2. Determinação do formato dos dados contidos no bloco lido – texto ou binário. Isso é feito pela checagem dos dados contidos nos 10% iniciais do bloco. Se eles são todos caracteres que podem ser impressos, o conteúdo é classificado como texto, caso contrário o programa assume que se trata de um bloco com dados binários;

3. Se o bloco contém texto, o programa testa os dados contra um conjunto de expressões regulares para tentar determinar do que se trata;

4. Se o bloco contém dados binários, o comando `file` é executado sobre o bloco. Se não há sucesso, os primeiros bytes do bloco são examinados para determinar se os dados parecem estar no formato ELF (representando um binário executável);

5. Se o bloco (contendo dados binários ou texto) é reconhecido nos passos anteriores como sendo de um determinado tipo, ele é marcado como tal. Se esse bloco é de um tipo diferente do bloco processado anteriormente, então ele é salvo como um novo arquivo. Caso contrário, ele é concatenado ao bloco anterior. Se o bloco não é reconhecido nos passos anteriores, mas é posterior a um bloco reconhecido, o programa assume que o bloco em questão é uma continuação dos dados contidos no bloco reconhecido e, portanto, o concatena ao bloco anterior (considerando o Princípio da Localidade);

A saída produzida pelo programa `lazarus` corresponde aos blocos de dados e um mapa dos blocos reconhecidos e seus respectivos tipos. É possível gerar uma saída no formato HTML, contendo um mapa com *links* que permitem navegar pelos dados interpretados, como ilustrado nas figuras 5 e 6.

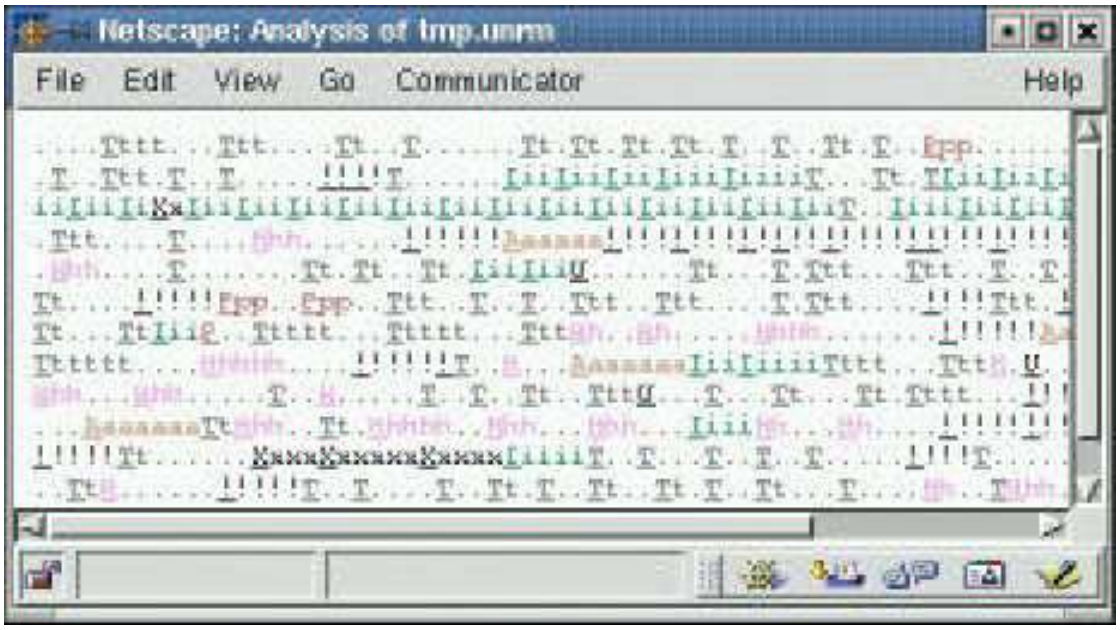


Figura 5: Mapa dos blocos reconhecidos pelo programa `lazarus`.

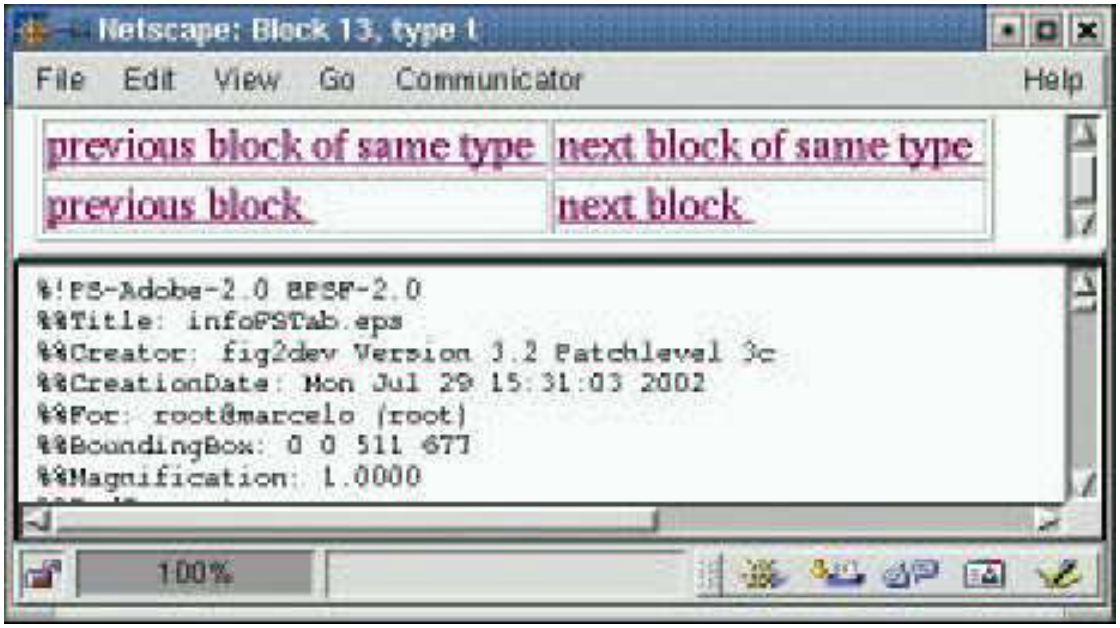


Figura 6: Navegação pelos blocos interpretados pelo programa `lazarus`.

## 6.2 TCTUTILs

O TCTUTILs[42] é um conjunto de ferramentas de código aberto e gratuito, desenvolvido por Brian Carrier, que utiliza funções e estruturas do TCT para prover funcionalidades extras. Muitas dessas funcionalidades já foram apresentadas anteriormente, entretanto, um resumo dos componentes do TCTUTILs é apresentado como segue:

- `bcat`: permite visualizar o conteúdo de um determinado bloco de dados do sistema de arquivos;
- `blockcalc`: 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 (produzida pelo programam `unrm` do TCT);
- `fls`: lista as entradas de diretório contidas nos blocos de dados de um determinado diretório. Permite a visualização de entradas referentes a arquivos "deletados";
- `find_file`: encontra o arquivo correspondente a um determinado *inode*, mesmo que o *inode* não esteja alocado (permitindo recuperar o nome de arquivos "deletados");
- `find_inode`: encontra o *inode* que tem alocado um determinado bloco de dados do sistema de arquivos;
- `istat`: permite visualizar informações sobre um determinado *inode*;

Maiores detalhes sobre as ferramentas do TCTUTILs podem sem encontrados nas respectivas *man pages* ou em [4]. O TCTUTILs, em sua versão 1.01, suporta apenas os sistemas de arquivos FFS e EXT2FS, sendo testado apenas nas plataformas Linux, OpenBSD e Solaris. Atualmente, o TCTUTILs não se encontra mais em desenvolvimento e suas funcionalidades foram todas transferidas para o conjunto de ferramentas chamado *The @stake Sleuth Kit* (TASK), apresentado na sequência.

## 6.3 *The @stake Sleuth Kit* (TASK)

O TASK[43] é um conjunto de ferramentas de código aberto e gratuito, desenvolvido por Brian Carrier, para análise de sistemas de arquivos UNIX (BSDi FFS, FreeBSD FFS, OpenBSD FFS, Solaris FFS, Linux EXT2FS e Linux EXT3FS) e Microsoft (FAT, FAT12, FAT16, FAT32 e NTFS). O TASK integra as ferramentas de análise de sistemas de arquivos do TCT com os utilitários do TCTUTILs, além de adicionar novas funcionalidades. Entre suas principais características, segundo o autor, estão a independência de plataforma e o suporte aos sistemas de arquivos NTFS e FAT.

As ferramentas do TASK são organizadas em uma abordagem de camadas e o nome de cada programa em uma mesma camada inicia-se com a mesma letra, facilitando a identificação

[42] O TCTUTILs pode ser encontrado na URL http://www.cerias.purdue.edu/homes/carrier/forensics/ (disponível em agosto de 2002).

[43] O TASK pode ser encontrado na URL http://www.atstake.com/research/tools/task (disponível em agosto de 2002).

de sua função. Essas camadas incluem o sistema de arquivos como um todo, os blocos de dados (conteúdo dos arquivos), as estruturas de dados do sistema de arquivos (*inodes*, por exemplo) e a interface de interação humana (nomes dos arquivos). Uma breve descrição dos componentes do TASK é apresentada como segue (algumas ferramentas são pertencentes ao TCT ou TCTUTILs, embora apresentem nomes diferentes):

- `dcalc`: corresponde ao `blockcalc` do TCTUTILs;
- `dcat`: corresponde ao `bcat` do TCTUTILs;
- `dls`: corresponde ao `unrm` do TCT;
- `dstat`: permite visualizar informações sobre um determinado bloco de dados (incluindo o número do grupo e se o bloco encontra-se alocado);
- `ffind`: corresponde ao `find_file` do TCTUTILs;
- `fls`: corresponde ao `fls` do TCTUTILs;
- `fsstat`: permite visualizar informações detalhadas sobre um sistema de arquivos (incluindo o nome do volume, tempo de última montagem e detalhes sobre cada grupo);
- `icat`: corresponde ao `icat` do TCT;
- `ifind`: corresponde ao `find_inode` do TCTUTILs;
- `ils`: corresponde ao `ils` do TCT;
- `istat`: corresponde ao `istat` do TCTUTILs;
- `mactime`: corresponde ao `mactime` do TCT;
- `md5`: corresponde ao `md5` do TCT;
- `sha1`: computa a assinatura criptográfica de um fluxo de bits qualquer, usando o algoritmo SHA-1 [12, 23];

Pequenas alterações, além do suporte a outros sistemas de arquivos, estão presentes em algumas das ferramentas que possuem correspondentes no TCT ou TCTUTILs (apenas o programa `mactime` apresenta modificações substanciais). Maiores detalhes sobre os programas do TASK podem ser encontrados nas respectivas *man pages*.

## 6.4 *Autopsy Forensic Browser* (AFB)

O AFB[44] é uma ferramenta de código aberto e gratuito, desenvolvida por Brian Carrier, que provê uma interface gráfica (ilustrada na figura 7) para as ferramentas do TASK, permitindo a análise dos arquivos, diretórios, blocos de dados e *inodes* (alocados ou "deletados") presentes

[44] O AFB pode ser encontrado na URL http://www.atstake.com/research/tools/autopsy (disponível em agosto de 2002).

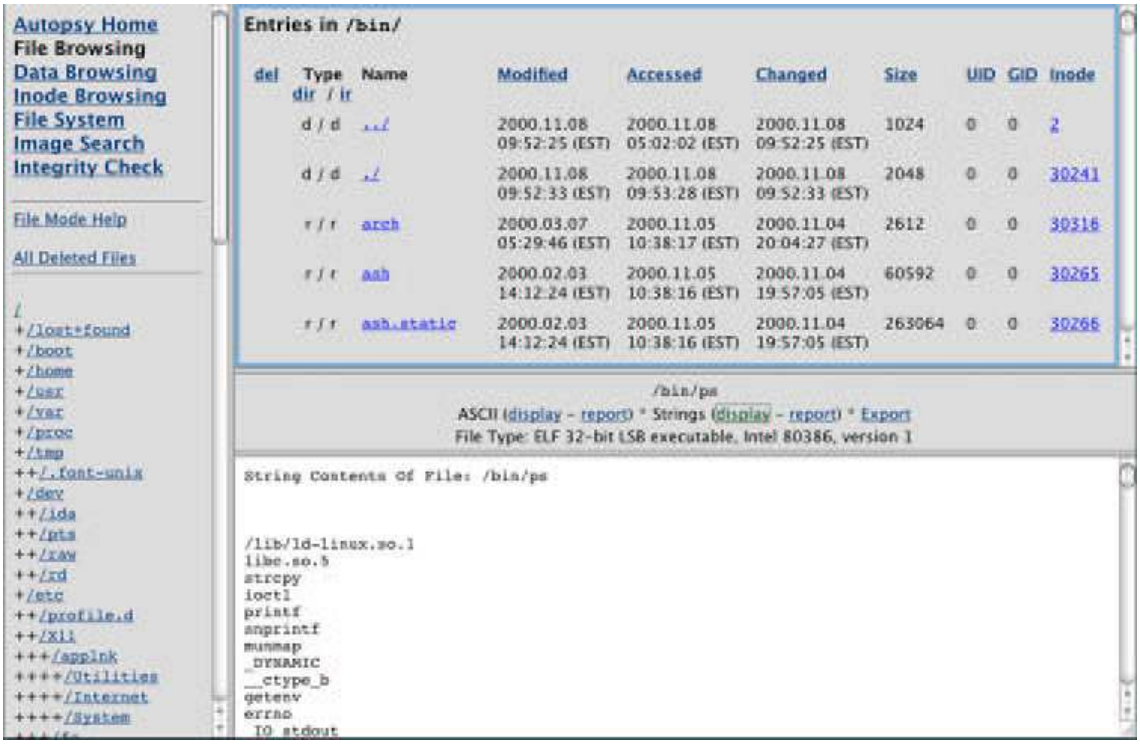


Figura 7: Exemplo da interface provida pelo *Autopsy Forensic Browser*.

na imagem de um sistema de arquivos (ou no arquivo gerado pelo `dls`). Através do AFB, as imagens dos sistemas de arquivos da máquina invadida podem ser examinadas no nível de abstração de arquivos, *inodes* ou blocos de dados. É possível, também, buscar por palavras-chave e expressões regulares nas imagens, bem como criar uma linha de tempo contendo os *mactimes* dos arquivos e diretórios.

A interface provida pelo AFB é baseada em HTML, segundo um modelo cliente-servidor. O AFB corresponde à parte servidora e qualquer navegador HTML (com suporte a *frames* e *forms*) pode ser usado como cliente. Essa abordagem permite que o AFB seja executado diretamente no sistema comprometido (caso não seja possível extrair imagens dos sistemas de arquivos), por meio de uma mídia removível, fornecendo acesso remoto e protegido contra escritas ao investigador, que utiliza um navegador HTML no sistema de análise.

O autor dos conjuntos de ferramentas TCTUTILs e TASK recomenda que eles sejam utilizados através do AFB[45]. Maiores detalhes sobre o AFB podem ser encontrados na *man page* do mesmo.

[45]No caso do TCTUTILs, o AFB deve ser usado em sua versão 1.01.

## 6.5 ForensiX

A ferramenta ForensiX[46] é um sistema que integra uma coleção de programas de coleta e análise através de uma interface gráfica (ilustrada na figura 8). Desenvolvido por Fred Cohen, o ForensiX é baseado no ambiente Linux, explorando muitas de suas vantagens como sistema de análise forense, e foi desenvolvido com o intuito de facilitar, de maneira eficiente e apropriada, a documentação, imagem e exame de informações digitais.

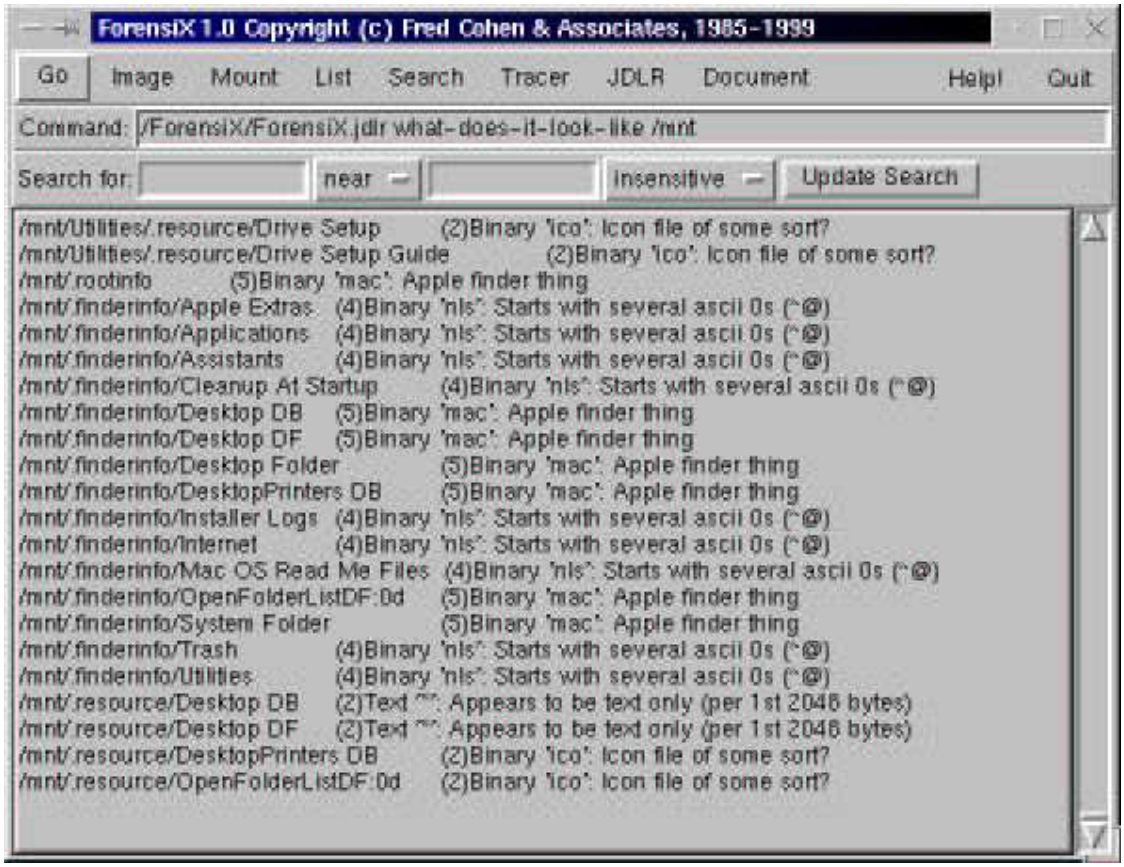


Figura 8: Exemplo da interface provida pelo ForensiX.

Algumas das principais características do ForensiX são listadas como segue:

- capacidade de produzir imagens a partir de diferentes tipos de fontes de dados, incluindo tráfego IP, disquetes, discos rígidos (IDE e SCSI), CDROMs, cartões PCMCIA e portas paralela e serial;
- pode armazenar uma imagem em arquivos, discos, fitas e CD-Ws;
- capacidade de montar (com proteção a escritas) imagens e mídias contendo diferentes tipos de sistemas de arquivos, incluindo os dos ambientes Windows, WindowsNT, DOS, Unix, MacIntosh e PalmOS;

[46] Maiores informações sobre o ForensiX podem ser encontradas na URL http://www.all.net/ForensiX/index.html (disponível em agosto de 2002).

- provê documentação automática das ações e cadeia de custódia das informações;
- permite a reprodução do processo de análise forense;
- provê checagem de integridade das informações;
- permite a busca de palavras-chave e assinaturas digitais conhecidas;
- capacidade de identificar o tipo de um arquivo através de seu conteúdo, permitindo encontrar tentativas de esconder informações;
- permite a análise de arquivos "deletados", blocos não alocados, áreas de *swap*, blocos defeituosos e *file slacks*;
- possibilita a busca e visualização de arquivos de imagens gráficas;
- permite checar um sistema Unix em busca de vulnerabilidades conhecidas;
- capacidade de produzir um *snapshot* de um sistema de arquivos e compará-lo com outros;
- permite a customização e programação de capacidades de análise e busca;

## 6.6 *New Technologies Incorporated* (NTI)

A empresa NTI[47] estabeleceu-se como uma das maiores vendedoras de *software* para forense. A NTI oferece treinamentos e pacotes de ferramentas para os mais variados propósitos, incluindo resposta a incidentes, proteção de evidências, limpeza de discos e produção de imagens. Seus programas são implementados na forma de ferramentas de linha de comando, baseadas no ambiente DOS. Por esse motivo são rápidas e pequenas o suficiente para caber em um disquete, de modo que o sistema suspeito pode ser inicializado com um disco de *boot* do DOS e analisado através da mídia contendo as ferramentas de forense. Algumas das principais ferramentas desenvolvidas pela NTI são descritas como segue:

- `CRCMD5`: valida o conteúdo de um ou mais arquivos;
- `DiskScrub`: realiza a esterilização de um disco, eleminando todos os dados;
- `DiskSig`: checa a integridade de uma imagem;
- `FileList`: cria uma lista dos arquivos de um sistema, ordenados pelo tempo de última utilização;
- `Filter_we`: filtro inteligente que utiliza lógica fuzzy;
- `GetFree`: coleta os blocos não alocados de um sistema de arquivos;
- `GetSlack`: coleta os *file slacks*;

[47] Maiores informações podem ser encontradas na URL http://www.forensics-intl.com (disponível em agosto de 2002).

- `GetTime`: captura a data e hora do sistema analisado;
- `Net Threat Analyzer`: usado para identificar abusos em contas pela Internet;
- `M-Sweep`: utilitário de limpeza de segurança;
- `NTI-Doc`: programa de documentação da análise;
- `PTable`: analisa e documenta as partições de um disco;
- `Seized`: usado para travar e proteger computadores apreendidos;
- `ShowFL`: usado para analisar uma listagem de arquivos;
- `TextSearch Plus`: busca palavras-chave e arquivos gráficos;
- `SafeBack`: utilitário para produção de imagens de discos;

## 6.7 EnCase

O EnCase[48] é um sistema integrado de análise forense baseado no ambiente Windows, desenvolvido pela *Guidance Software* (provedora de outras ferramentas e treinamento no campo da forense). Assim como as ferramentas da NTI, o EnCase é amplamente utilizado por mantenedores da lei e profissionais de segurança de computadores em todo o mundo [14].

O processo utilizado pelo EnCase começa com a criação das imagens dos discos (disquetes, Zips, Jaz, CDROMs e discos rígidos) relacionados ao caso investigado. Depois da criação das imagens, chamadas de EnCase *evidence files*, pode-se adicioná-las a um único caso (*case file*) e conduzir a análise em todas elas simultaneamente.

O ambiente Windows não é considerado apropriado, por muitos pioneiros da área, para a prática forense, uma vez que ele rotineiramente altera os dados e escreve no disco rígido sempre que é acessado [6]. Entretanto, o EnCase não opera na mídia original ou discos espelhados. Ao invés disso, o EnCase monta os *evidence files* como discos virtuais protegidos contra escritas. Então, o EnCase (não o sistema operacional) reconstroi o sistema de arquivos contido em cada *evidence file*, permitindo ao investigador visualizar, ordenar e analisar os dados, de maneira não invasiva, através de uma interface gráfica (ilustrada na figura 9).

Várias funções e ferramentas de análise são integradas pelo EnCase, podendo ser citadas as seguintes:

- visualização do caso através de uma interface do tipo Windows Explorer;
- suporte a múltiplos sistema de arquivos, incluindo FAT (12, 16 e 32), NTFS, Macintosh, CDROM, EXT2FS, UFS, PDAs e RAID;
- visualização de todos os arquivos de um caso, incluindo os *file slacks* (no formato texto ou hexadecimal);

[48] Maiores informações podem ser encontradas em [6] ou na URL http://www.encase.com (disponível em agosto de 2002).

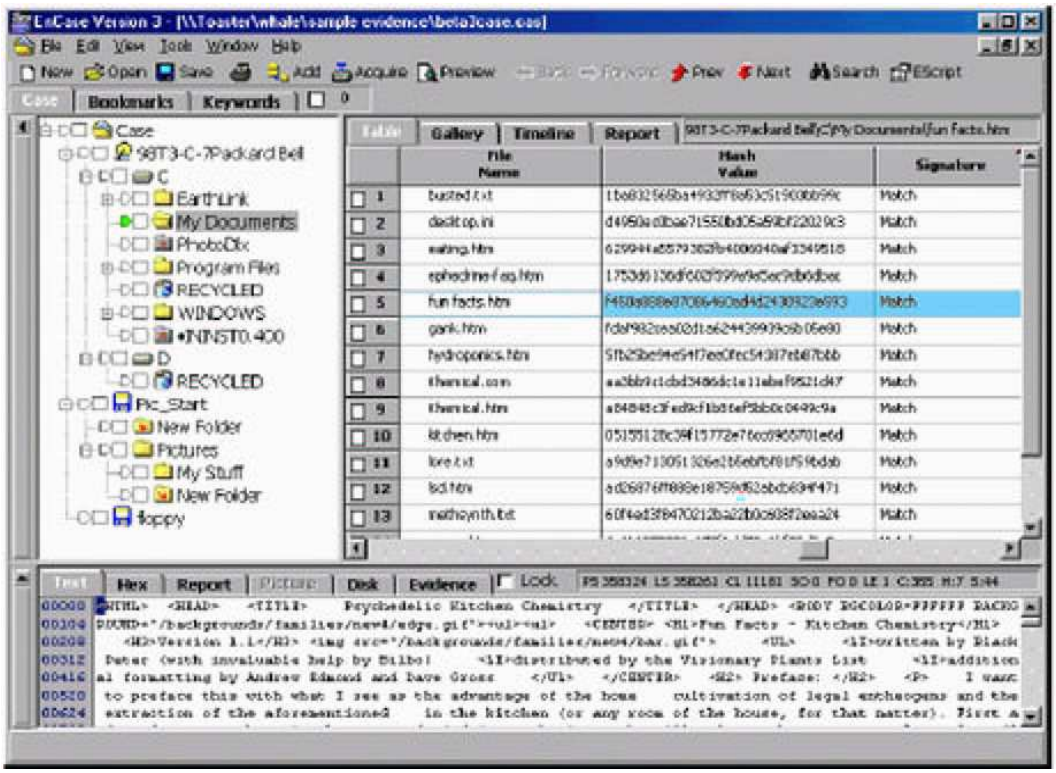

Figura 9: Exemplo da interface provida pelo EnCase.

- visualização e análise remota dos dados contidos na máquina investigada, através da interface paralela ou da rede;
- busca e visualização de imagens gráficas (mesmo "deletadas");
- mecanismo de busca de palavras-chave e expressões;
- relatório do caso, contendo a descrição dos dados e sua cadeia de custódia;
- criação e checagem de assinaturas criptográficas de arquivos;
- customização de filtros e funções através de uma linguagem de *scripts*;
- visualização de arquivos compostos, incluindo o registro do Windows, anexos de mensagens de correio eletrônico e arquivos compactados;
- visualização da linha de tempo de utilização dos arquivos;
- identificação do tipo de um arquivo;
- visualização de arquivos de *swap*, *file slacks*, filas e arquivos localizados em *Recycle Bin*;
- mapa gráfico de alocação do disco;

# 7 Conclusão

Infelizmente, aqueles que cometem crimes não estão alheios à revolução computacional que tem ocorrido nas últimas décadas. Um número crescente de criminosos fazem 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 em 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 algumas investigações criminais.

A criminalística, por ser uma disciplina que se utiliza do conhecimento de outras ciências [9], necessita manter-se atualizada em relação aos desenvolvimentos técnico-científicos. A forense computacional, por ser um ramo da criminalística bastante recente e relacionado a uma das áreas científicas que mais evolui atualmente, requer atenção especial. Além disso, a crescente utilização de computadores em atividades criminosas fundamenta tal necessidade de desenvolvimento, no sentido de se ampliar o uso de evidências digitais no amparo à justiça.

O estudo apresentado neste trabalho representa um esforço no sentido de suprir essa necessidade de desenvolvimento científico na área da forense computacional. 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. De maneira análoga, o *framework* apresentado para o processo de investigação forense constitui um ponto de partida para o desenvolvimento de bons procedimentos de análise, podendo ser aplicado a qualquer tipo de investigação envolvendo sistema computacionais. Além disso, a apresentação de diversas ferramentas (com exemplos práticos) permite orientar o investigador na escolha e manipulação de seus utilitários de análise.

# Referências

[1] Good practices guide for computer based evidence. Association of Chief Police Officers of England, Wales and Northern Ireland, Junho de 1999. ACPO Crime Committee.

[2] Rebecca G. Bace. *Intrusion Detection*. Macmillan Technical Publishing, Indianapolis, IN, 2000.

[3] Adriano M. Cansian. Crime na internet: Conhecendo e observando o inimigo. Notas de aula e slides de micro-curso durante o *SSI'2000*: Simpósio Segurança em Informática, São José dos Campos, SP, Outubro de 2000.

[4] Brian Carrier. Performing an autopsy examination on FFS and EXT2FS partition images: An introduction to TCTUTILs and the Autopsy Forensic Browser. Disponível *online* em agosto de 2001 na URL http://www.cerias.purdue.edu/homes/carrier/forensics.html.

[5] Eoghan Casey. *Digital Evidence and Computer Crime*. Academic Press, San Diego, California, 2000.

[6] Eoghan Casey, editor. *Handbook of Computer Crime Investigation*. Academic Press, San Diego, California, 2002.

[7] Douglas E. Comer. *Internetworking with TCP/IP*, volume 1. Prentice Hall, Upper Saddle River, New Jersey, 3ª edição, 1995.

[8] Chris Drake e Kimberly Brown. *Panic! Unix System Crash Dump Analysis*. Prentice Hall, Upper Saddle River, New Jersey, 1995.

[9] Alberi Espindula. A função pericial do estado. Disponível *online* em agosto de 2001 na URL http://www.espindula.com.br/artigo1.htm.

[10] Dan Farmer e Wietse Venema. Computer forensics analysis class handouts. Disponível *online* em agosto de 2001 na URL http://www.fish.com/forensics/class.html, Agosto de 1999.

[11] Daniel Farmer e Eugene H. Spafford. The COPS security checker system. Em *Proceedings of the Summer USENIX Conference*, pp. 165–170, Anaheim, CA, Junho de 1990.

[12] Simson Garfinkel e Gene Spafford. *Practical UNIX and Internet Security*. O'Reilly & Associates, Sebastopol, California, 2ª edição, 1996.

[13] Chet Hosmer, John Feldman, e Joe Giordano. Advancing crime scene computer forensic techniques. WetStone Technologies Inc, 2000. Disponível *online* em agosto de 2001 na URL http://wetstonetech.com/crime.htm.

[14] Warren G. Kruse II e Jay G. Heiser. *Computer Forensics: Incident Response Essentials*. Addison-Wesley, Reading, Massachusetts, 2002.

[15] Kevin Mandia e Chris Prosise. *Incident Response: Investigating Computer Crime*. McGraw-Hill, Berkeley, California, 2001.

[16] Marshall K. McKusick, Keith Bostic, Michael J. Karels, e John S. Quarterman. *The Design and Implementation of the 4.4 BSD Operating System*. Addison-Wesley, Reading, Massachusetts, 1996.

[17] Michael G. Noblett, Mark M. Pollitt, e Lawrence A. Presley. Recovering and examining computer forensic evidence. *Forensic Science Communications*, 2(4), Outubro de 2000. U.S. Department of Justice, FBI.

[18] Scientific Working Group on Digital Evidence (SWGDE) e International Organization on Digital Evidence (IOCE). Digital evidence: Standards and principles. *Forensic Science Communications*, 2(2), Abril de 2000. U.S. Department of Justice, FBI.

[19] Marcelo A. Reis e Paulo L. Geus. Forense computacional: Procedimentos e padrões. Em *Anais do SSI2001: 3º Simpósio Segurança em Informática*, pp. 73–81, Instituto Tecnológico de Aeronáutica-ITA, São José dos Campos, SP, Outubro de 2001.

[20] Marcelo A. Reis e Paulo L. Geus. Standardization of computer forensic procedures and protocols. Em *Proceedings of the* $14^{th}$ *Annual Computer Security Incident Handling Conference*, Waikoloa, HI, USA, Junho de 2002. FIRST.Org, Inc.

[21] Marcelo A. Reis, Flávio S. Oliveira, Célio C. Guimarães, e Paulo L. Geus. Forense computacional: Aspectos legais e padronização. Em *Anais do Wseg2001: Workshop em Segurança de Sistemas Computacionais*, pp. 80–85, Florianópolis, SC, Março de 2001. Workshop realizado durante o SCTF2001: IX Simpósio de Computação Tolerante a Falhas.

[22] Tony Sammes e Brian Jenkinson. *Forensic Computing: A Practitioner's Guide*. Springer, London, UK, 2000.

[23] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, New York, 1996.

[24] A. Silberschatz e P. Galvin. *Operating System Concepts*. John Wiley & Sons, New York, 5ª edição, 1998.

[25] Eugene H. Spafford e Stephen A. Weeber. Software forensics: Can we track code to its authors? Em *Proceedings of the* $15^{th}$ *National Computer Security Conference*, pp. 641–650, Outubro de 1992.

[26] W. Stallings. *Computer Organization and Architecture: Designing for Performance*. Prentice Hall, Upper Saddle River, New Jersey, 5ª edição, 2000.

[27] Peter Stephenson. *Investigating Computer-Related Crime*. CRC Press, Boca Raton, Florida, 2000.

[28] W. Richard Stevens. *TCP/IP Illustrated*, volume 1. Addison-Wesley, Reading, Massachusetts, 2ª edição, 1994.

[29] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, Upper Saddle River, New Jersey, 3ª edição, 1996.