LAUDELINO AZEREDO DOS SANTOS

COMPUTAÇÃO FORENSE EM SISTEMAS GNU/LINUX

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação *Lato Sensu* em Administração em Redes Linux para obtenção do título de Especialização.

Oritentador
Prof. Msc. Sandro Melo

LAVRAS
MINAS GERAIS – BRASIL
2008

LAUDELINO AZEREDO DOS SANTOS

COMPUTAÇÃO FORENSE EM SISTEMAS GNU/LINUX

Monografia apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras, como parte das exigências do curso de Pós-Graduação *Lato Sensu* em Administração em Redes Linux para obtenção do título de Especialização.

Aprovada em ____ de _____ de _____ .

Prof. _____

Prof. _____

Prof. _____
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2008

Dedico esta monografia à minha querida esposa Jociane, por me compreender nos momentos de ausência e pelo incentivo e apoio.

Agradecimentos

Primeiramente a Deus pela oportunidade de cumprir mais uma etapa da minha vida.

À minha esposa e grande amor, Jociane, pelo apoio e ajuda nos momentos em que mais precisei.

Ao meus pais e irmãos, pelo apoio e pela compreensão durante minha ausência e distância.

Ao meu professor e orientador, Sandro Melo, pela compreensão e paciência com que guiou o desenvolvimento deste trabalho.

Sumário

Lista de figuras

Lista de tabelas

Resumo

Este trabalho apresenta conceitos referentes ao processo de análise computacional forense em sistemas GNU/LINUX, tendo como objetivo demonstrar técnicas e procedimentos que auxiliem na coleta e análise dos dados, levando-se em consideração, principalmente, o tempo de vida e a ordem de volatilidade dos mesmos. Também é abordada a documentação das provas por intermédio de uma cadeia de custódia.

Palavras-chave: forense; volatilidade; *in vivo*; *post mortem*.

Capítulo 1
Introdução

Na última década, a Internet passou a ser um dos principais meios de comunicação para se trocar informações, adquirir produtos e fazer uso de serviços que até pouco tempo atrás eram incomuns e muito pouco utilizados pelas pessoas. O crescimento da utilização da Internet motivou o aparecimento de novos serviços em vários segmentos do mercado, como, por exemplo, no setor bancário. Esta nova era serviu como meio para o surgimento de crimes virtuais procurando explorar esses novos tipos de serviço.

Segundo (OLIVEIRA, 2002), apesar de hoje as organizações estarem preocupadas em utilizar mecanismos para aumentar a segurança dos sistemas computacionais que utilizam a Rede de Computadores, não há garantias de que elas não poderão ser vítimas de um incidente de segurança, mesmo que sigam todos os tipos de recomendações e implantem as mais modernas tecnologias.

Como nenhum tipo de tecnologia garante 100% de segurança, é importante que se encontrem meios para que, no caso de uma ocorrência de invasão, a organização possa agir da melhor maneira possível para evitar o agravamento da situação, colocando em prática, por exemplo, o programa de resposta a incidentes. Segundo (SÊMOLA, 2003), este programa é composto por uma equipe multiespecializada, capaz de atuar de forma integrada e com velocidade para reduzir o tempo de exposição da empresa, minimizando os impactos da invasão. Essa equipe, em conjunto com as metodologias de análise forense, podem fazer surtir efeitos positivos para a empresa no que se refere à resolução de problemas causados pelas invasões e ações que visem desvendar as causas e os responsáveis pelos mesmos. Atualmente tais ferramentas e

metodologias são muito pouco comuns na maioria das empresas.

A forense computacional é um campo de pesquisa relativamente novo no mundo e está desenvolvendo-se principalmente pela necessidade das instituições legais atuarem no combate aos crimes eletrônicos. No Brasil, conta-se ainda com poucos pesquisadores na área e existem poucas normas estabelecidas, o que gera um grande número de possibilidades de pesquisa. Neste trabalho, desenvolveu-se um estudo de ferramentas e técnicas que podem contribuir para uma boa resposta a incidentes nas organizações.

1.1 Objetivos

O principal objetivo deste trabalho é demonstrar técnicas e procedimentos que auxiliem no processo de análise forense de sistemas computacionais baseados no Sistema Operacional Linux; além de contribuir para o enriquecimento de material técnico sobre esta área, até então pouco explorada didaticamente.

1.2 Motivação

A preparação para a resposta a um incidente de segurança passa pela elaboração de procedimentos e pela tomada de medidas que possam futuramente auxiliar na realização de uma investigação forense com eficácia. Infelizmente existe uma escassez de metodologias e estudos abordando este tipo de

necessidade em relação às plataformas Linux, apesar do seu uso ter se difundido nos últimos anos.

Embora exista muita documentação que pode ser encontrada na Internet, ela se encontra em muitos casos desestruturada e confusa; e a maioria dos livros desta área estão em língua estrangeira, o que dificulta a leitura dos mesmos.

A vontade de preencher estas lacunas de falta de material em português e de forma organizada serviram como motivação para o desenvolver deste trabalho.

1.3 Estado da arte

Atualmente existem algumas ferramentas que auxiliam na análise forense e que são suportadas pela plataforma GNU/LINUX, dentre elas estão os aplicativos do TCT (The Coroner's Toolkit), que é um conjunto de aplicativos escritos por Dan Farmer e Wietse Venema, que permitem investigar se um sistema UNIX está comprometido através de coleta de informações de processos e informações alteradas e excluídas.

Seguindo a mesma idéia dos aplicativos do TCT (The Coroner's Toolkit), existem também os aplicativos: TCTUTILs, The @stake Sleuth Kit (TASK), Autopsy Forensic Browser (AFB) e o ForensiX.

Outra ferramenta que está sendo bastante utilizada na análise forense é o aplicativo Chaosreader, desenvolvido por Brendan Gregg; este analisa as informações de rede capturadas por *sniffers* e gera um relatório em HTML com várias informações que possibilitam ao perito remontar cronologicamente e com

detalhes o que foi trafegado na rede.

Outras ferramentas que também são muito utilizadas são *Live CDs*, com distribuição customizada para auxiliar na análise forense, estes contêm vários programas utilitários, sendo que muitos destes aplicativos já estão presentes na maioria das distribuições Linux.

1.4 Estrutura dos tópicos

No Capítulo 2, serão apresentados conceitos sobre a computação forense, de forma a alinhar os conhecimentos entre o leitor e autor.

No Capítulo 3, serão descritos procedimentos para a execução de uma perícia do tipo *forense in vivo*, ou seja, coletar evidências sem precisar desligar o equipamento a ser periciado, levando em consideração a ordem de volatilidade dos dispositivos de entrada e saída.

No Capítulo 4, serão descritos procedimentos para a execução de uma perícia do tipo *forense post mortem*, que consiste na análise do sistema após o desligamento do mesmo.

No Capítulo 5, serão descritos procedimentos para uma análise mais detalhada das informações de rede, abordados de forma mais sucinta em um dos tópicos do Capítulo 3.

No Capítulo 6, serão feitas as considerações finais sobre todos os conceitos citados no decorrer deste trabalho.

Capítulo 2

Fundamentação teórica sobre forense computacional

Segundo (NOBLETT , 2000), a Forense Computacional foi criada com o objetivo de suprir as necessidades das instituições legais no que se refere à manipulação das novas formas de evidências eletrônicas. Ela é a ciência que estuda a aquisição, preservação, recuperação e análise de dados que estão em formato eletrônico e armazenados em algum tipo de mídia computacional.

Conforme citado por (NOBLET, 2000), a forense computacional pode produzir informações diretas, que por sua vez, podem ser decisivas em um dado caso.

É importante salientar que a veracidade dessas informações também são passíveis de validação como em qualquer outro tipo de análise forense, pois dependendo das circunstâncias, as informações coletadas podem ter sido manipuladas de forma a induzir o perito a tirar conclusões erradas sobre o caso.

Segundo (FARMER, 2000), para resolver um mistério computacional nem sempre é fácil; deve-se analisar o sistema como um detetive que examina a cena de um crime, e não como um usuário comum. Muitas das habilidades necessárias para se procurar um erro em um código fonte são também necessárias para uma análise forense, tais como: raciocínio lógico, entendimento das relações de causa e efeito em sistemas computacionais e talvez a mais importante, ter uma mente aberta.

De acordo com (FARMER, 2000), uma perícia em um computador suspeito envolve uma série de conhecimentos técnicos e a utilização de ferramentas adequadas para a análise, que é justificado pela necessidade

indiscutível de não alterar o sistema que está sendo analisado. Tais alterações, se efetuadas, podem ser traduzidas como mudanças nos tempos de acesso aos arquivos, prejudicando assim uma das mais poderosas formas de se reconstituir o que aconteceu na máquina em um passado próximo. Geralmente as ferramentas convencionais não têm a preocupação de manter a integridade dos tempos de acesso.

Para (FREITAS, 2006) alguns procedimentos devem ser seguidos pelo perito para garantir que a evidência não seja comprometida, substituída ou perdida, pois, se estes não forem seguidos, num tribunal, os juízes poderão considerar que essas evidências são inválidas e os advogados de defesa poderão contestar sua legitimidade, prejudicando assim o caso.

> 'Em muitos casos, as únicas evidências disponíveis são as existentes em formato digital. Isto poderia significar que a capacidade de punição a um invasor pode estar diretamente relacionada com a competência do perito em identificar, preservar, analisar e apresentar as evidências.' (FREITAS, 2006, p. 2)

Sendo assim, é notória a importância do perito e a responsabilidade a ele confiada. Tal postura, se seguida, produzirá um trabalho revestido de incontestabilidade diante do tribunal.

Segundo (FREITAS, 2006), a perícia forense possui quatro procedimentos básicos: identificação, preservação, análise e apresentação. As tarefas envolvidas em uma investigação se enquadram em um desses grupos, como podem ser realizadas através da maioria ou de todos eles.

2.1 Identificando as evidências

Dentre as várias tarefas envolvidas no caso, é necessário estabelecer quais são as informações mais relevantes, como datas, horários, nomes de pessoas, empresas, endereços eletrônicos, nome do responsável ou proprietário pelo computador e etc.

Para (FREITAS, 2006), diferentes crimes resultam em diferentes tipos de evidência, e, por este motivo, cada caso deve ser tratado de forma específica. Por exemplo, em um caso de acesso não autorizado, o perito deverá procurar por arquivos log, conexões e compartilhamentos suspeitos; já em casos de pornografia, buscará por imagens armazenadas no computador, histórico dos sites visitados recentemente, arquivos temporários e etc.

A velocidade do perito em identificar as evidências vai depender do seu conhecimento sobre o tipo de crime que foi cometido e dos programas e Sistemas Operacionais envolvidos no caso. Conforme (FREITAS, 2006), para encontrar possíveis evidências deve-se:

- Procurar por dispositivos de armazenamentos (*hardware*): *laptops*, *HDs*, disquetes, *CDs*, *DVDs*, *drives Zip/Jaz*, *memory keys*, *pendrives*, câmeras digitais, *MP3 player*, fitas *DAT*, *Pocket PC*, celulares, dispositivos de backup ou qualquer equipamento que possa armazenar evidências;
- Procurar por informações relacionadas ao caso como: anotações, nomes de pessoas, datas, nomes de empresas e instituições, números de telefones, documentos impressos etc.;
- Distinguir entre evidências relevantes e irrelevantes em uma análise.

2.2 Cadeia de custódia

Segundo o princípio de Locard, através do contato entre dois itens, irá haver uma permuta. Este princípio é aplicável nas cenas do crime, no qual o interveniente da cena do crime entra em contato com a própria cena onde o crime foi executado, trazendo algo para este. Cada contato deixa o seu rastro.

Sendo assim, todo contato entre quaisquer pessoas ou objetos com a cena do crime pode produzir vestígios, que por mínimos que sejam, poderão servir como base para elucidação dos fatos de um crime.

Conforme (FREITAS, 2006), a regra número um em uma investigação é não destruir ou alterar as provas, ou seja, as evidências precisam ser preservadas de tal forma que não haja qualquer dúvida sobre a sua veracidade.

A inobservância de certos procedimentos no momento da coleta de dados e no transcorrer da análise forense pode significar a diferença entre o sucesso e o fracasso de uma perícia, pois, além de comprometer a veracidade de uma prova, o que a colocaria em dúvida perante o tribunal, poderia também comprometer a sua integridade.

De acordo com (FREITAS, 2006), devido ao uso do contraditório, num tribunal, a defesa pode questionar a legitimidade dos resultados de uma investigação, alegando que as evidências foram alteradas ou substituídas por outras.

Devido a essa possibilidade, é de total importância que o perito faça uso da cadeia de custódia, que, segundo (FREITAS,2006), é utilizada para provar onde as evidências estavam em um determinado momento e quem era o responsável por elas durante o curso da perícia. Tal documentação é de suma

importância para garantir que as evidências não foram comprometidas e para mapear individualmente os responsáveis num dado momento.

No Anexo segue um exemplo de um documento para dar início a uma cadeia de custódia.

Conforme (FREITAS, 2006), para que as evidências não sejam comprometidas, substituídas ou perdidas durante o transporte ou manuseio no laboratório, é recomendável que sejam seguidos os seguintes passos:

- Criar imagens do sistema investigado, também conhecido como duplicação pericial, para que as evidências digitais possam ser analisadas posteriormente;
- Se o caso necessitar de uma análise ao vivo, salvar as evidências em um dispositivo externo e bloqueá-los contra regravação;
- Todas as evidências físicas deverão ser lacradas em sacos e etiquetadas;
- A etiqueta deverá conter um número para a identificação das evidências, o número do caso, a data e o horário em que a evidência foi coletada, e o nome da pessoa que está manuseando-a;
- Etiquetar todos os cabos e componentes do computador, para que depois possam ser montados corretamente quando chegar ao laboratório;
- Os *HDs* deverão ser armazenados em sacos antiestáticos, para evitar danos e corrompimento dos dados;
- Durante o transporte das provas, tomar cuidado com líquidos, umidade, impacto, sujeira, calor excessivo, eletricidade e estática;
- Quando já tiverem sido transportadas, as evidências deverão ser armazenadas e trancadas para evitar a adulteração até o momento em que poderão ser examinadas e analisadas;
- Todas as mudanças feitas durante esta fase deverão ser documentadas e

justificadas no documento referente à cadeia de custódia.

2.3 Analisando as evidências

O propósito desta fase é tentar identificar quem fez, quando fez, que dano causou e como foi realizado o crime. Mas, para isso, o perito deverá saber o que procurar, onde procurar e como procurar. Para (FREITAS, 2006), após analisar as evidências o perito poderá responder às seguintes questões:

- Qual a versão do Sistema Operacional que estava sendo investigado?
- Quem estava conectado ao sistema no momento do crime?
- Quais arquivos foram usados pelo suspeito?
- Quais portas estavam abertas no Sistema Operacional?
- Quem logou ou tentou logar no computador recentemente?
- Quem eram os usuários e a quais grupos pertenciam?
- Quais arquivos foram excluídos?

Esta fase será a pesquisa propriamente dita, em que praticamente todos os filtros de informação já foram executados. A partir deste ponto o perito poderá focalizar-se nos itens realmente relevantes ao caso em questão. É nesta fase que os itens farão sentido e os dados e fatos passarão a ser confrontados.

2.4 Apresentando a análise

Para (FREITAS, 2006) o laudo pericial é um relatório técnico sobre a investigação, no qual são apontados os fatos, procedimentos, análises e o resultado. O perito faz o laudo, e, a partir das evidências, a decisão é da Justiça.

- O laudo deve ser claro, conciso, estruturado e sem ambigüidade, de tal forma que não deixe dúvida alguma sobre a sua veracidade;
- Deverão ser informados os métodos empregados na perícia, incluindo os procedimentos de identificação, preservação e análise, e o *software* e *hardware* utilizados;
- O laudo pericial deve conter apenas afirmações e conclusões que possam ser provadas e demonstradas técnica e cientificamente.

Capítulo 3

Realizando procedimentos de *forense in vivo*

Antes de iniciar uma análise, o perito deverá avaliar qual procedimento utilizar para a coleta dos dados. A escolha deverá levar em conta o cenário e as circunstâncias em que se encontram o sistema a ser analisado. Alguns questionamentos devem ser feitos, como por exemplo: O que deverá ser coletado? O que coletar primeiro? Vale a pena coletar tudo? Será que não serão colocadas em risco todas as outras evidências só porque se pretende coletar algumas evidências em tempo de execução?

A *forense in vivo* consiste num conjunto de procedimentos que são utilizados para a coleta de dados sem que o sistema seja desligado. Nesta é priorizada a coleta dos dados mais voláteis para os menos voláteis, pois em sua maioria, os dados mais voláteis serão perdidos caso o sistema venha a ser deligado; diferente da *post mortem forense*, que consiste numa coleta de dados após o desligamento do sistema; este será abordado no próximo Capítulo com mais detalhes.

Quando a *forense in vivo* é utilizada, é importante atentar para a ordem de volatilidade ou OOV (*order of volatility*), que segundo (FARMER, 2006), pode ajudar a preservar os detalhes mais importantes que uma simples coleta de dados pode destruir.

Na Tabela 3.1 pode ser mostrado o ciclo de vida dos dados.

Tabela 3.1: O ciclo de vida esperado dos dados. (FARMER,2006)

Tipos de dados	Tempo de vida
Reistradores, memória periférica, caches etc.	Nanossegundos
Memória principal	Dez nanossegundos
Estado da rede	Milissegundos
Processos em execução	Segundos
Disco	Minutos
Disquetes, mídia de backup etc.	Anos
CD-ROMs, impressões etc.	Dezenas de anos

Para ilustrar um pouco melhor como é esta relação volatilidade versus tempo de vida dos dados, e dada a sua importância para este trabalho e para qualquer perícia forense computacional; é apresentado na Figura 3.1 um gráfico com tais informações.

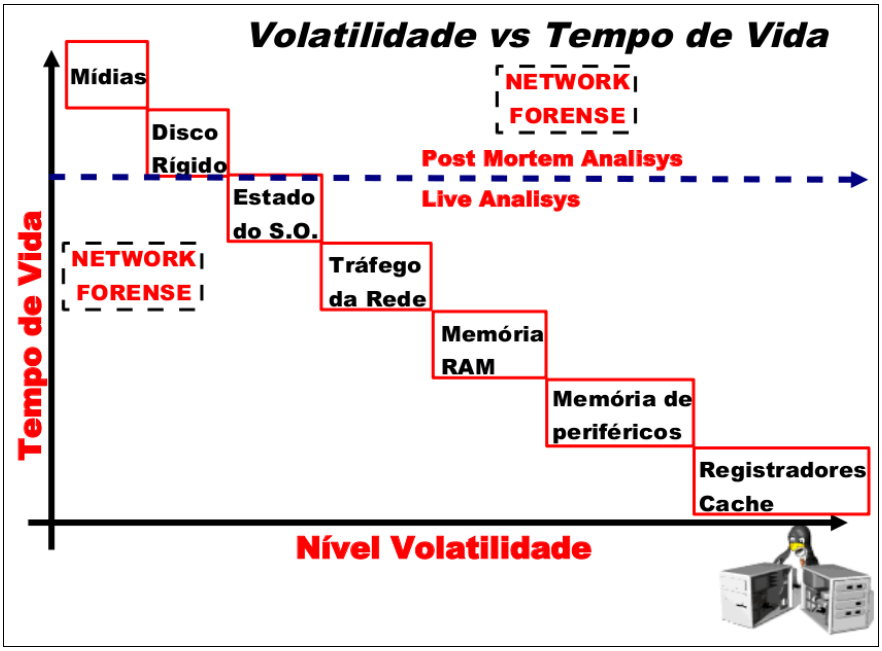


Figura 3.1: Volatilidade versus Tempo de vida. (MELO, 2007)

Se o que interessa for o conteúdo de um disco rígido e dados que já foram alterados há muito tempo, como por exemplo em disquetes e CD-ROMs, quase não existe razão para a coleta de dados da memória e muito menos focar na técnica de *forense in vivo*. Neste caso, poderia se fazer uso da técnica *post mortem forense*, a ser discutida no Capítulo 4.

De acordo com a ordem de volatilidade citada na Figura 3.1 e nos itens que cabem à *forense in vivo*, seguem nos subtópicos seguintes mais informações.

3.1 Capturando informações de registradores e *cache*

Segundo (KRUSE, 2001), o simples fato de observar informações nos registradores do processador, já pode alterá-las. Por isso tornam-se de mínima utilidade e sua captura é impraticável, assim como os dados em *cache*.

3.2 Capturando informações da memória de periféricos

Alguns periféricos, como por exemplo impressora e vídeo, possuem as suas próprias memórias. Nelas podem existir informações que talvez não existam mais no sistema suspeito, ou seja, cabe ao perito tentar capturá-las.

No caso da memória do vídeo por exemplo, numa situação em que o suspeito utilizou o terminal gráfico, este dado é de suma importância, pois o que aparece na tela poderá ser capturado utilizando o aplicativo Xwd, conforme mostrado na Figura 3.2.

```
[root@forense ~]# xwd -display 127.0.0.1:0.0 -root > mem_video.xwd
```

Figura 3.2: Copiando a memória da placa de vídeo.

Este arquivo de saída contém a tela capturada e poderá ser visualizado utilizando o aplicativo zwud seguido do parâmetro *in* e o nome do arquivo. Outro aplicativo que lê este arquivo é o GIMP.

3.3 Capturando informações da memória principal

Conforme já visto na tabela de ciclo de vida dos dados, a memória principal, também conhecida como memória RAM, contém diversas informações voláteis do sistema operacional, como por exemplo, dados sobre os processos que estão em execução, dados que estão sendo manipulados ou que ainda não foram gravados em disco.

Para capturar estes dados, pode ser utilizado o aplicativo savecore, que segundo (FARMER, 2006), faz uma cópia de parte ou de toda a memória principal do computador para um sistema de arquivos convencional. Esta cópia é feita após o sistema ser configurado.

Além do aplicativo savecore, a memória também pode ser acessada (caso tenha privilégios suficientes) via arquivo de dispositivo, já que sistemas UNIX os tratam como tal.

Segundo (NEMETH, 2004), arquivos de dispositivo apresentam uma interface de comunicação padrão que se parece com um arquivo comum; arquivo este que repassa a solicitação de entrada e saída de dados para o *driver*

apropriado. Para copiar os dados do arquivo de dispositivo, é ideal que se faça uma cópia *bit*-a-*bit* para garantir a integridade, conforme pode ser mostrado na Figura 3.3.

```
[root@forense ~]# dd if=/dev/mem of=/copia.memoria
dd: lendo `/dev/mem': Endereço inválido
1031832+0 registos dentro
1031832+0 registos fora
528297984 bytes (528 MB) copiados, 18,3172 s, 28,8 MB/s
```

Figura 3.3: Copiando o conteúdo da memória principal.

A análise dessas cópias da memória, criadas pelo savecore ou `dd` por exemplo, podem ser auxiliadas pelas aplicações strings e grep; aquela retorna o conteúdo texto do arquivo e esta filtra o resultado exibindo somente as palavras requeridas pelo executor, assim como na Figura 3.4.

```
[root@forense ~]# strings /copia.memoria | grep arquiv_suspeito.txt
[00marquiv_suspeito.txt
arquiv_suspeito.txt
arquiv_suspeito.txt
arquiv_suspeito.txt
-writer arquiv_suspeito.txt
[00marquiv_suspeito.txt
[root@forense ~]# swriter arquiv_suspeito.txt
arquiv_suspeito.txt
swriter arquiv_suspeito.txt
ter arquiv_suspeito.txt
swriter arquiv_suspeito.txt
swriter arquiv_suspeito.txt
mv arq_suspeito.txt arquiv_suspeito.txt
arquiv_suspeito.txt
arquiv_suspeito.txt
arquiv_suspeito.txt
swriter arquiv_suspeito.txt
arquiv_suspeito.txt
swriter arquiv_suspeito.txt
arquiv_suspeito.txt
arquiv_suspeito.txt
swriter arquiv_suspeito.txt
```

Figura 3.4: Analisando conteúdo da memória principal

A saída demonstrada na Figura 3.4 mostra passo-a-passo as últimas ações com o arquivo suspeito.

3.4 Capturando informações do estado da rede

O estado de rede provê informações importantes sobre as conexões de rede em andamento e dos processos que estão aguardando uma conexão. A partir destas informações, é possível verificar se o invasor instalou algum programa suspeito como por exemplo um *backdoor*.

Primeiramente é interessante visualizar quais as interfaces de rede presentes e o *IP* de cada uma delas, seguido da tabela de roteamento da máquina. Essas informações podem ser capturadas com os aplicativos `ifconfig` e `route`, respectivamente, conforme as figuras 3.5 e 3.6.

```
[root@forense ~]# ifconfig
eth0      Link encap:Ethernet  Endereço de HW 00:14:22:D8:2F:55
          inet end.: 189.14.196.81  Bcast:189.14.196.255  Masc:255.255.255.0
          endereço inet6: fe80::214:22ff:fed8:2f55/64 Escopo:Link
          UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1
          RX packets:438 errors:0 dropped:0 overruns:0 frame:0
          TX packets:134 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:1000
          RX bytes:49236 (48.0 KiB)  TX bytes:14378 (14.0 KiB)
          IRQ:18
```

Figura 3.5: Visualizando o ip da máquina com o `ifconfig`.

```
[root@forense ~]# route
Tabela de Roteamento IP do Kernel
Destino         Roteador        MáscaraGen.     Opções Métrica Ref   Uso Iface
189.14.196.0    *               255.255.255.0   U     0      0        0 eth0
link-local      *               255.255.0.0     U     0      0        0 eth0
default         189.14.196.1    0.0.0.0         UG    0      0        0 eth0
```

Figura 3.6: Visualizando a tabela de roteamento com o `route`.

Para visualizar o estado das conexões de rede, pode ser utilizado o aplicativo `netstat`, que, segundo (RIBEIRO, 2004), exibe as conexões de rede, tabela de rotas, estatísticas das interfaces e várias outras informações. Este aplicativo seguido dos parâmetros *-atunp* permite ao perito visualizar todas as conexões de rede TCP e UDP, o número de identificação dos processos e o *IP's* de destino e local envolvidos na conexão.

```
[root@forense ~]# netstat -atunp
Conexões Internet Ativas (servidores e estabelecidas)
Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program na
me
tcp        0      0 0.0.0.0:25152               0.0.0.0:*                   OUÇA       2911/skype

tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   OUÇA       1760/rpcbind

tcp        0      0 0.0.0.0:44273               0.0.0.0:*                   OUÇA       1779/rpc.statd

tcp        0      0 0.0.0.0:22                  0.0.0.0:*                   OUÇA       2118/sshd

tcp        0      0 127.0.0.1:631               0.0.0.0:*                   OUÇA       2207/cupsd

tcp        0      0 127.0.0.1:25                0.0.0.0:*                   OUÇA       2142/sendmail:
acce
tcp        0      0 189.14.196.81:46616         189.60.242.237:51548        ESTABELECIDA2911/skype

tcp        0      0 :::22                       :::*                        OUÇA       2118/sshd
```

Figura 3.7: Visualizando os estados das conexões de rede com o `netstat`

Conforme a figura 3.7, na saída do aplicativo `netstat` é possível detectar algumas informações que podem indicar, por exemplo, a ausência de serviços que deveriam estar habilitados, endereços de *IP* suspeitos e conexões suspeitas.

Capturar as informações que trafegam na rede também é muito importante para uma boa perícia, mas será abordado no Capítulo 5.

3.5 Capturando informações do estado do sistema operacional

Ainda com o sistema ligado, alguns dados do sistema operacional devem ser coletados, como por exemplo, o estado dos processos e os usuários logados no sistema, não deixando de se levar em consideração a ordem de volatilidade dos dados. É recomendável que não sejam utilizadas ferramentas do sistema suspeito, pois podem ter sido alteradas pelo invasor.

A análise dos processos em execução no sistema é de grande importância, pois mostra a situação real do momento. O comando `ps` junto com o parâmetro *aux*, por exemplo, é uma alternativa para listar todos os processos em execução no sistema, conforme a Figura 3.8.

```
[root@forense ~]# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1948   496 ?        Ss   14:25   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S<   14:25   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   14:25   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S<   14:25   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   14:25   0:00 [watchdog/0]
root         6  0.0  0.0      0     0 ?        S<   14:25   0:00 [events/0]
root         7  0.0  0.0      0     0 ?        S<   14:25   0:00 [khelper]
root        60  0.0  0.0      0     0 ?        S<   14:25   0:00 [kblockd/0]
root        62  0.0  0.0      0     0 ?        S<   14:25   0:00 [kacpid]
root        63  0.0  0.0      0     0 ?        S<   14:25   0:00 [kacpi_notify]
root       141  0.0  0.0      0     0 ?        S<   14:25   0:00 [cqueue]
root       143  0.0  0.0      0     0 ?        S<   14:25   0:00 [ksuspend_usbd]
root       148  0.0  0.0      0     0 ?        S<   14:25   0:00 [khubd]
root       151  0.0  0.0      0     0 ?        S<   14:25   0:00 [kseriod]
root       189  0.0  0.0      0     0 ?        S<   14:25   0:02 [kswapd0]
root       232  0.0  0.0      0     0 ?        S<   14:25   0:00 [aio/0]
root       367  0.0  0.0      0     0 ?        S<   14:25   0:00 [pccardd]
root       382  0.0  0.0      0     0 ?        S<   14:25   0:00 [kpsmoused]
root       429  0.0  0.0      0     0 ?        S<   14:25   0:01 [ata/0]
root       430  0.0  0.0      0     0 ?        S<   14:25   0:00 [ata_aux]
root       437  0.0  0.0      0     0 ?        S<   14:26   0:00 [scsi_eh_0]
root       438  0.0  0.0      0     0 ?        S<   14:26   0:01 [scsi_eh_1]
root       446  0.0  0.0      0     0 ?        S<   14:26   0:00 [kjournald]
root       500  0.0  0.0   2812   468 ?        S<s  14:26   0:00 /sbin/udevd -d
root       788  0.0  0.0      0     0 ?        S<   14:26   0:00 [ipw2200/0]
```

Figura 3.8: Listando os processos em execução com o `ps`.

A saída gerada pela execução do comando `ps` lista os processos em execução de todos os usuários que estão logados no sistema, com o número de identificação de cada processo, dono do processo, consumo de memória e processador, horário do início do processo e vários outras informações.

Outro aplicativo que possui finalidade parecida com a do aplicativo `ps` é o aplicativo `top`, pois este além de apresentar a lista de todos os processos em execução, mostra-os em tempo real, conforme pode ser ilustrado na Figura 3.9.

```
top - 18:52:09 up  4:26,  3 users,  load average: 0.04, 0.18, 0.30
Tasks: 137 total,   1 running, 136 sleeping,   0 stopped,   0 zombie
Cpu(s):  6.6%us,  1.0%sy,  0.0%ni, 92.1%id,  0.0%wa,  0.3%hi,  0.0%si,  0.0%st
Mem:    506824k total,   495396k used,    11428k free,     5388k buffers
Swap:   522072k total,   248892k used,   273180k free,   135072k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
13264 root      20   0  374m  34m 7444 S  0.7  6.9   2:53.72 Xorg
13689 junior    20   0 92528  22m 7836 S  0.7  4.5   0:31.10 skype
 1559 root      15  -5     0    0    0 S  0.3  0.0   0:04.37 kondemand/0
15223 junior    20   0 82588  18m  11m S  0.3  3.8   0:07.45 gnome-terminal
17559 root      20   0  2364 1036  800 R  0.3  0.2   0:00.02 top
    1 root      20   0  1948  496  440 S  0.0  0.1   0:01.52 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S  0.0  0.0   0:00.30 ksoftirqd/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:00.16 events/0
    7 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
   60 root      15  -5     0    0    0 S  0.0  0.0   0:00.31 kblockd/0
   62 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   63 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  141 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue
  143 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 ksuspend_usbd
  148 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 khubd
  151 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  189 root      15  -5     0    0    0 S  0.0  0.0   0:02.89 kswapd0
```

Figura 3.9: Listando os processos em tempo real com o `top`.

O aplicativo `lsof` lista todos os arquivos abertos por processos em execução e também é de grande importância. Caso seja necessário fazer referência a um processo, basta utilizar o parâmetro *-p* seguido do número do processo, conforme é exemplificado na Figura 3.10.

```
[root@forense ~]# lsof -p 13264
COMMAND   PID USER   FD   TYPE     DEVICE     SIZE       NODE NAME
Xorg    13264 root  cwd    DIR        8,2     4096     625651 /var/gdm
Xorg    13264 root  rtd    DIR        8,2     4096          2 /
Xorg    13264 root  txt    REG        8,2  1747908     409391 /usr/bin/Xorg
Xorg    13264 root  mem    REG        8,2   165600     491412 /usr/lib/xorg/modules/exten
Xorg    13264 root  mem    REG        8,2    16024     489802 /usr/lib/xorg/modules/exten
Xorg    13264 root  mem    REG        8,2     3328     592184 /usr/lib/xorg/modules/fonts
Xorg    13264 root  mem    REG        8,2    35616     491410 /usr/lib/xorg/modules/exten
Xorg    13264 root  mem    REG        8,2    10468     491411 /usr/lib/xorg/modules/exten
Xorg    13264 root  mem    REG        8,2     4624     539343 /usr/lib/xorg/modules/drive
Xorg    13264 root  mem    REG        8,2    25152     450163 /usr/lib/xorg/modules/input
Xorg    13264 root  mem    REG        8,2    45660     449414 /usr/lib/xorg/modules/input
Xorg    13264 root  mem    REG        8,2    21004     449489 /usr/lib/xorg/modules/libvb
Xorg    13264 root  mem    REG        8,2    25552     449606 /usr/lib/xorg/modules/libvg
Xorg    13264 root  mem    REG        8,2    58500     449391 /usr/lib/xorg/modules/libex
Xorg    13264 root  mem    REG        8,2    15668     498772 /lib/libcap.so.2.06
Xorg    13264 root  mem    REG        8,2   261720     498773 /lib/libdbus-1.so.3.4.0
Xorg    13264 root  mem    REG        8,2   154324     449486 /usr/lib/xorg/modules/libin
Xorg    13264 root  mem    REG        8,2   154772     449485 /usr/lib/xorg/modules/libfb
Xorg    13264 root  mem    REG        8,2    17648     449413 /usr/lib/xorg/modules/input
Xorg    13264 root  mem    REG        8,2    53944     498771 /lib/libgcc_s-4.3.0-2008042
Xorg    13264 root  mem    REG        8,2   425680     491413 /usr/lib/xorg/modules/exten
```

Figura 3.10: Listando arquivos aberto por processos com o `lsof`.

Seguindo a idéia de que nesta primeira fase da perícia é essencial coletar todas as informações voláteis possíveis, também é de suma importância coletar dados como, por exemplo, o tempo de atividade dos usuários na máquina, os usuários que estão conectados ao sistema e várias outras informações a serem descritas a seguir.

Segundo (NEMETH, 2004), o aplicativo `uptime` mostra o tempo em que a máquina está ligada, seguido da média de carga de CPU da máquina nos últimos 1, 5 e 15 minutos, dados que poderão ser confrontados com outros para atestar a veracidade dos mesmos conforme a Figura 3.11.

```
[root@forense ~]# uptime
 19:09:52 up  4:43,  3 users,  load average: 0.77, 0.56, 0.53
```

Figura 3.11: Verificando o tempo de atividade do sistema com o `uptime`.

Com o aplicativo `last`, consegue-se ver os últimos usuários que entraram no sistema e o tempo de duração de sua sessão de *login*, informações estas muito importantes. E com o aplicativo `w`, é possível descobrir quais os usuários que estão conectados ao sistema naquele momento e o que estão executando, para ser mais específico, qual aplicativo estão utilizando. Esses dois aplicativos podem servir para verificar a entrada de usuários não autorizados e o tempo de permanência deles no sistema conforme figuras 3.12 e 3.13.

```
[root@forense ~]# last
junior   pts/2        :0.0             Mon Jul 28 18:32   still logged in
junior   pts/1        :0.0             Mon Jul 28 18:04   still logged in
junior   tty7         :0               Mon Jul 28 17:29   still logged in
junior   pts/1        :0.0             Mon Jul 28 17:28 - 17:29  (00:01)
junior   tty7         :0               Mon Jul 28 17:28 - 17:29  (00:01)
junior   pts/1        :0.0             Mon Jul 28 17:24 - 17:27  (00:02)
junior   pts/4        :0.0             Mon Jul 28 17:10 - 17:24  (00:14)
junior   pts/2        :0.0             Mon Jul 28 17:10 - 17:22  (00:12)
junior   pts/3        :0.0             Mon Jul 28 15:04 - 17:21  (02:16)
junior   pts/2        :0.0             Mon Jul 28 15:00 - 16:56  (01:55)
junior   pts/1        :0.0             Mon Jul 28 14:58 - 17:24  (02:25)
junior   tty7         :0               Mon Jul 28 14:27 - 17:27  (03:00)
reboot   system boot  2.6.25.10-86.fc9 Mon Jul 28 14:26          (04:48)
junior   pts/1        :0.0             Mon Jul 28 12:39 - 13:59  (01:19)
junior   tty7         :0               Mon Jul 28 12:36 - down   (01:22)
reboot   system boot  2.6.25.10-86.fc9 Mon Jul 28 12:35          (01:23)
junior   pts/1        :0.0             Mon Jul 28 07:30 - 10:13  (02:42)
junior   tty7         :0               Mon Jul 28 05:11 - down   (05:02)
```

Figura 3.12: Verificando o históricos de *logins* com o `last`.

```
[root@forense ~]# w
 19:18:06 up  4:52,  3 users,  load average: 0,54, 0,42, 0,45
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
junior   tty7     :0               17:29    0.00s  4:37   0.09s gnome-session
junior   pts/1    :0.0             18:04    0.00s  0.04s 10.02s gnome-terminal
junior   pts/2    :0.0             18:32    3:19   0.07s 10.02s gnome-terminal
```

Figura 3.13: Verificando os usuários logados com o `w`.

Ainda fazendo referência à captura de dados voláteis, existe também o aplicativo `free`, que apresenta a quantidade de memória física em uso e disponível do sistema. O resultado desse aplicativo poderá ser comparado com a

saída dos aplicativos `ps` e `top` por exemplo.

Segundo (RIBEIRO, 2004), o aplicativo `mount` mostra quais os dispositivos de sistemas de arquivos estão configurados e em funcionamento no sistema; já o aplicativo `lsmod` exibe os módulos do *kernel* que estão sendo utilizados.

Outro aplicativo útil é o `dmesg`, que segundo (FARMER, 2006), serve para exibir informações da memória do *kernel*, que contêm dados do *hardware* detectado no momento da inicialização, e algumas mensagens de *log* de sistema gerados até o presente momento, conforme pode ser ilustrado na Figura 3.14.

```
[root@forense ~]# dmesg
Initializing cgroup subsys cpuset
Linux version 2.6.25.10-86.fc9.i686 (mockbuild@) (gcc version 4.3
on Jul 7 20:46:03 EDT 2008
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f000 (usable)
 BIOS-e820: 000000000009f000 - 00000000000a0000 (reserved)
 BIOS-e820: 0000000000100000 - 000000001f7d3800 (usable)
 BIOS-e820: 000000001f7d3800 - 0000000020000000 (reserved)
 BIOS-e820: 00000000e0000000 - 00000000f0007000 (reserved)
 BIOS-e820: 00000000f0008000 - 00000000f000c000 (reserved)
 BIOS-e820: 00000000fec00000 - 00000000fec10000 (reserved)
 BIOS-e820: 00000000fed20000 - 00000000fee10000 (reserved)
 BIOS-e820: 00000000ffb00000 - 0000000100000000 (reserved)
0MB HIGHMEM available.
503MB LOWMEM available.
Scan SMP from c0000000 for 1024 bytes.
Scan SMP from c009fc00 for 1024 bytes.
Scan SMP from c00f0000 for 65536 bytes.
Scan SMP from c009f000 for 1024 bytes.
Using x86 segment limits to approximate NX protection
Entering add_active_range(0, 0, 128979) 0 entries of 256 used
```

Figura 3.14: Visualizando informações da memória do *kernel* com o `dmesg`.

Como pode ser observado, devido à grande quantidade de aplicativos que provêem dados similares, é de essencial importância que eles sejam coletados e comparados, para atestar a sua veracidade. Muitas dessas

informações estão disponíveis no diretório `/proc`, que é um sistema de arquivos falso, utilizado como uma interface para as estruturas de dados do *kernel* do sistema operacional, permitindo ao mesmo uma visão de cada processo em execução, assim como o estado da memória e várias outras informações.

Essas informações podem ser confrontadas com as saídas das aplicações citadas no tópicos anteriores, como, por exemplo, os dados da memória disponível, que podem ser vistos no arquivo `/proc/meminfo`.

3.6 Duplicação forense

Após a coleta de todas as evidências e gravação do resultado destas em disco, o próximo passo é gerar um *hash* de verificação de todos os arquivos, que nada mais é do que a soma dos *bits* destes, gerada à partir de um algoritmo e apresentada geralmente em hexadecimal, conforme é feito pelo comando `md5sum` nas Figuras 3.15 e 3.16. Esse *hash* sofrerá mudança caso o tamanho do arquivo mude.

```
[root@forense ~]# md5sum mem_video.xwd > mem_video.xwd.md5
```

Figura 3.15: Utilizando `md5sum` para gerar *hash* de verificação.

```
[root@forense ~]# cat mem_video.md5
9f978892a6f2ce81e764786c52eb29fb *-
```

Figura 3.16: Visualizando o resultado do *hash* de verificação.

Após criar esses arquivos, o perito tem que se preocupar em copiar tais

dados para um dispositivo externo, seja *pen drive*, *zip drive*, *HD* externo ou pela rede. Basta conectar o dispositivo, configurá-lo no sistema operacional e copiar os dados. Após copiado, também é importante que seja gerado um novo *hash* de verificação com o comando `md5sum` destes arquivos, para uma posterior comparação, caso seja necessário.

Em uma situação em que o perito se depara com um sistema suspeito que não pode ser desligado, muito menos instalado algum dispositivo de armazenamento externo, como por exemplo *pen drive*, *HD* externo ou algo do tipo, os dados deverão ser enviados pela rede. Existe um aplicativo chamado Netcat que pode ser muito útil para este fim.

Segundo (GIACOBBI, 2006), o aplicativo Netcat permite ler e escrever dados através de uma conexão de rede, tornando possível, por exemplo, a criação de um ambiente cliente-servidor para transferir dados de um sistema suspeito, dados estes que podem variar de pequenas informações do estado do sistema operacional até a cópia completa dos dados de um sistema.

Lembrando que todas as informações já citadas nos tópicos anteriores podem ser enviadas via rede com o Netcat, para isso basta incluir algumas informações na linha de comando, conforme demonstrado na Figura 3.17.

Supondo que o servidor ou a máquina onde será feita a análise posterior dos dados tenha o nome `serv_fore`, segue abaixo um exemplo de configuração da mesma para receber os dados da máquina suspeita.

```
[root@serv_fore ]# nc -l 2500 | tee mem_video.xwd | md5sum -b > mem_video.md5
```

Figura 3.17: Utilizando nc para receber arquivo.

O comando `nc`, da ferramenta Netcat, com os parâmetros *-l 2500* indica

que a estação ficará em modo de escuta, à espera de pacotes na porta 2500. O comando `tee` recebe os dados da entrada padrão, armazena em um arquivo e gera como saída uma cópia idêntica com o nome especificado, além de também gerar com o aplicativo `md5sum` um *hash* criptográfico em um arquivo externo com final *.md5*. Esse servirá para comparar com a saída do md5sum da máquina suspeita.

Dada a continuidade ao processo de envio do arquivo *mem_video.xwd* pela rede, a máquina suspeita deverá executar o comando descrito na Figura 3.18.

```
[root@forense ~] xwd -display 127.0.0.1 -root |
tee mem_video.xwd | nc serv_fore 2500
```

Figura 3.18: Utilizando nc para enviar arquivo.

A saída do comando da Figura 3.18 é direcionada pelo aplicativo `netcat` para a porta 2500 da máquina *serv_fore*.

Para finalizar esta etapa de coleta de dados, é necessário fazer uma cópia de todo o conteúdo contido no disco rígido para uma análise posterior. Para tanto, deve-se utilizar o aplicativo `dd`, que irá copiar os dados *bit*-a-*bit*.

```
[root@forense ~]# dd if=/dev/sda of=sda_img.dd
```

Figura 3.19: Utilizando `dd` para copiar os dados do disco rígido.

Vale ressaltar que também é de grande importância gerar o *hash* de verificação do arquivo de saída do comando citado na Figura 3.19.

Capítulo 4

Realizando procedimentos de *post mortem forense*

Conforme a ordem de volatilidade, este procedimento é feito após o desligamento do computador, ou seja, neste momento só existe a memória não volátil para a análise, que inclui o disco rígido, *CD-ROMs* e disquetes, por exemplo. Esta etapa da análise pode ser considerada a mais custosa quanto ao tempo, dada a quantidade imensa de informações a serem analisados.

Esta coleta de dados deverá ser preferencialmente feita em uma máquina diferente daquela de onde foram coletadas as evidências, isto para que o disco rígido original seja preservado. O perito deverá fazer uma cópia do disco rígido da máquina suspeita, conforme comentado no Capítulo anterior.

4.1 Análise dos sistemas de arquivos

Com o ambiente para análise montado, o próximo passo é acessar os arquivos coletados da máquina suspeita e em modo de somente leitura, para impedir a execução de binários e interpretação de arquivos de dispositivo, assegurando assim, a integridade dos dados. Na Figura 4.1 segue um exemplo desta montagem.

```
[root@serv_fore ~]# mount -o ro,noexec,nodev,loop sda_img.dd /mnt/analise
```

Figura 4.1: Montando imagem de disco para análise.

Com a imagem do computador suspeito já configurada, o perito passará

a ter acesso a todo o sistema a ser periciado. Como existem diversos arquivos e diretórios, o perito deverá ter conhecimentos das principais fontes de informação que podem fornecer alguma informação sobre o sistema.

De acordo com (REIS, 2003), segue na Tabela 4.1 uma breve descrição das principais fontes de informação em sistemas GNU/LINUX.

Tabela 4.1: Principais fontes de informações

Fonte de Informação	Descrição
Arquivos de configuração	O sistema Linux possui certos arquivos de configuração comumente acessados ou alterados pelos invasores. São geralmente *scripts* de inicialização, arquivos de contas e senhas e outras configurações do sistema.
Diretórios temporários	Os diretórios temporários `/tmp` e `/usr/tmp` servem como diretórios de "rascunho" para todo o sistema. Como eles são limpos periodicamente, acabam se tornando locais apropriados para armazenar dados que não serão usados no futuro e excluídos automaticamente.
Diretório de arquivo de dispositivos	Com exceção de alguns arquivos especiais, o diretório `/dev` deve conter apenas os arquivos de dispositivo.
Arquivos e diretórios escondidos ou não usuais	Arquivos e diretórios ocultos ou com nomes incomuns são freqüentemente usadas por atacantes para escondê-los.
Executáveis e bibliotecas	Arquivos executáveis e bibliotecas são alterados pelos atacantes para esconder sua presença.
Arquivos de *log*	Os arquivos de *log* são muito importantes, pois podem registrar as atividades dos usuários, dos processos, do sistema das atividades de rede e informações específicas de aplicativos e serviços.
Arquivos e diretórios de usuários	Arquivos de texto, mensagens de correio eletrônico, imagens, entre outros tipos de dados que podem conter informações úteis relacionadas ao atacante.

Em posse da lista de arquivos, diretórios e dispositivos a serem pesquisados, é de suma importância que o perito saiba como fazer a coleta dos dados, ou seja, usar as informações a seu favor para que elas possam ajudá-lo a elucidar o caso.

Segundo (FARMER, 2007), às vezes saber quando algo aconteceu é mais valioso do que saber o que aconteceu, ou seja, é de suma importância obter dados em relação ao tempo. Para isso existem os *MACtimes*, que são uma maneira abreviada de se referir aos três atributos de tempo dos arquivos: *mtime*, *atime* e *ctime*.

Ainda segundo (FARMER, 2007), o atributo *atime* refere-se à última data/hora em que o arquivo ou diretório foi acessado. O atributo *mtime*, ao contrário, muda quando o conteúdo de um arquivo é modificado. O atributo *ctime* monitora quando o conteúdo ou as meta-informações sobre o arquivo mudaram: o proprietário, o grupo , as permissões de arquivo e assim por diante. O atributo *ctime* também pode ser utilizado como uma aproximação de quando um arquivo foi executado.

O aplicativo `stat` mostra os *MACtimes* de um arquivo; basta passar como parâmetro o nome do arquivo.

4.2 Recuperação de evidências em áreas não alocadas

Segundo (FARMER, 2007), quando um arquivo é excluído, a entrada de diretório com o nome do arquivo e o número de *inode* é marcada como não utilizada, entretanto os dados contidos no arquivo não são apagados do disco.

Tais dados permanecem no disco indefinidamente, até que sejam sobrescritos por outros arquivos.

Arquivos de *log* e binários utilizados pelo invasor podem ser removidos pelo mesmo com o intuito de esconder seus rastros, por isso a importância do perito analisar as informações contidas nas áreas não alocadas.

Segundo (REIS, 2003), o utilitário unrm, que compõe o conjunto de ferramentas chamado The Coroner's Toolkit (TCT), é capaz de extrair toda informação presente nos espaços não alocados de um sistema de arquivos.

```
[root@serv_fore ~]# unrm sda_img.dd > naoalocados.unrm
```

Figura 4.2: Extraindo informações excluídas de imagem

Conforme a Figura 4.2, o aplicativo unrm é usado para copiar, no arquivo binário `naoalocados.unrm`, toda informação contida nos espaços não alocados do sistema de arquivos da imagem `sda_img.dd` (imagem do disco suspeito, citado no Capítulo 3). Lembrando que o arquivo destino deve estar em outro sistema de arquivos que o da imagem, pois o seu tamanho pode compreender vários *gigabytes* de dados.

Um editor hexadecimal pode ser utilizado para visualizar e examinar as informações do arquivo `naoalocados.unrm`, mas seria um trabalho muito tedioso. Para diminuir a complexidade da análise, pode ser utilizado, por exemplo, os comandos `strings` e `grep`, que poderão ser utilizados para buscar informações por palavras-chave, conforme Figura 4.3.

```
[root@serv_fore ~]# cat naoalocados.unrm |
strings –a | grep May
```

Figura 4.3: Analisando o arquivo de saída do unrm.

No exemplo da Figura 4.3, é utilizada uma combinação de comandos para buscar entradas de *log* excluídas, referentes ao mês de maio.

Existe também o aplicativo lazarus, que faz parte do pacote de aplicativos do `TCT`; este analisa os dados coletados pelo unrm e tenta classificá-los de acordo com o seu conteúdo, conforme exemplo da Figura 4.4.

```
[root@serv_fore ~]# lazarus –h naoalocados.unrm
```

Figura 4.4: Visualizando os dados coletados pelo unrm com o lazarus.

4.3 Identificação e análise dinâmica de artefatos

Esta etapa consiste em identificar todos os arquivos criados ou manipulados no sistema durante o incidente. Após checar as principais fontes de informação, citados no Capítulo 4.1, é importante identificar se, por exemplo, o aplicativo `poweroff`, utilitário para reinicializar a máquina, possui no sistema analisado realmente esta finalidade, e garantir que o mesmo não foi substituído ou alterado por outro aplicativo com outra finalidade. O invasor poderia por exemplo ter copiado um *script* no lugar deste utilitário, e com uma finalidade maliciosa.

Segundo (BAILEY,2000), para fazer este tipo de checagem em distribuições Linux onde os aplicativos foram instalados com pacotes RPM por exemplo, pode ser utilizado o aplicativo `rpm`, que, com o parâmetro *-Va*,

verifica todos os pacotes instalados e a consistência de seus binários e arquivos de configuração, fazendo um comparativo entre o estado atual e o inicial, desde quando foi instalado pela primeira vez no sistema. Caso tenha alguma diferença, esta será mostrada.

```
[root@forense ~]# rpm -Va
S.5....T  c /var/log/mail/statistics
SM5....T  c /var/lib/nfs/state
S.5....T  c /etc/ggz.modules
S.5....T  c /etc/ppp/chap-secrets
S.5....T  c /etc/ppp/pap-secrets
.M....G.    /var/log/gdm
SM5...GT    /sbin/poweroff
```

Figura 4.5: Verificando mudanças desde o estado inicial dos RPMS

Conforme Figura 4.5, cada letra demonstrada na coluna da esquerda equivale a um item que foi alterado conforme lista da Tabela 4.2:

Tabela 4.2: Checando mudanças dos arquivos com o rpm

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

Determinados arquivos terão diferenças de tamanho, assinatura do *MD5* e até do tempo de modificação, principalmente os arquivos de configuração que precisaram ser editados pelo usuário do sistema. Vale ressaltar que determinadas modificações em determinados arquivos, como por exemplo um binário que é utilizado para desligar o computador, o `poweroff`, seria um motivo para analisá-lo melhor, pois este não deveria ser alterado, só se foi atualizado por uma nova versão.

Para os arquivos que tiveram grandes modificações, é importante atentar para a data da modificação para fazer analogia à data do incidente. Checar com o aplicativo `file` o tipo do arquivo e checar com o aplicativo `stat` os *MACtimes* destes, conforme Figuras 4.6 e 4.7.

```
[root@serv_fore ~]# file /sbin/poweroff
/sbin/poweroff: ASCII text
```

Figura 4.6: Utilizando `file` para visualizar o tipo do arquivo.

```
[root@serv_fore ~]# stat /sbin/poweroff
  File: `/sbin/poweroff'
  Size: 19              Blocks: 8          IO Block: 4096   arquivo comum
Device: 802h/2050d      Inode: 57102       Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2008-08-09 08:17:00.000000000 -0300
Modify: 2008-08-09 08:17:00.000000000 -0300
Change: 2008-08-09 08:17:00.000000000 -0300
```

Figura 4.7: Utilizando `stat` para visualizar *MACtimes*.

Também para verificar quais arquivos foram acessados ou modificados recentemente, pode-se utilizar o aplicativo `find`, que com os parâmetros *atime*, *mtime*, *amin* e *cmin*, podem verificar os arquivos que foram acessados ou modificados nos últimos dias e acessados ou modificados nos últimos minutos

respectivamente. Conforme a Figura 4.8, no comando deve ser indicado o tempo com os parâmetros + ou –, e as unidades dias ou minutos a partir da data atual e horário atual.

```
[root@forense /]# find / -ctime -1
/
/tmp
/tmp/keyring-pizY41
/tmp/virtual-junior.XaYgHN
/tmp/vbox.1
/tmp/virtual-junior.2ZyKkb
/tmp/.vbox-junior-ipc
/tmp/vbox.6
/tmp/virtual-junior.GV66X5
/tmp/pulse-gdm
```

Figura 4.8: Utilizando aplicativo ctime

4.4 Analisando e correlacionando informações da forense in vivo com a post mortem

Após a coleta de todos os dados considerados importantes para a perícia, tanto na *forense in vivo*, quanto na *post mortem forense*, a confrontação das informações se torna um item indispensável para que conclusões sobre o caso comecem a surgir.

Segundo (REIS, 2003), o relacionamento entre as informações pode ser estabelecido através de vários parâmetros, como, por exemplo, registros de tempo, relações de causa e efeito, e números de identificação como por exemplo: PID, USERID, GROUPID, endereços IP e portas de serviços de rede.

É nesta etapa que pode-se ver a importância de se fazer uma descrição

minuciosa das evidências encontradas no sistema comprometido no momento da coleta, pois assim estas conterão dados que tornem possível um relacionamento entre as informações.

Por exemplo, caso o perito, por intermédio da cópia da memória *RAM* e dos processos em execução, consiga descobrir vestígios de arquivos suspeitos instalados na máquina, pode-se confrontar com o relatório do aplicativo `find`, citado no tópico anterior e ter certeza de que determinados dados foram alterados, além de também confirmar quando o usuário acessou o sistema e de onde fez o acesso; dados estes que poderiam ser colhidos pelos aplicativos `lastlog` e `netstat`, ambos já citados neste trabalho.

A construção de uma linha de tempo pode ajudar o perito a visualizar com maior clareza a ordem dos eventos e suas relações, permitindo a identificação de padrões e a descoberta de novas evidências.

Capítulo 5

Capturando informações de rede

Por intermédio do tráfego de rede é possível analisar toda a comunicação entre a máquina atacante e a que foi invadida, tornando possível estabelecer uma seqüência de eventos que poderá ser comparada com as outras evidências encontradas ao longo da perícia.

Segundo (REIS, 2003), existem vários programas que podem ser utilizados para a captura de tráfego de rede, estes são conhecidos como *sniffers*. Além de capturar os pacotes que trafegam na rede, os *sniffers* podem decodificá-los e exibi-los num formato mais inteligível para o perito.

Um dos mais conhecidos *sniffers* é o aplicativo Tcpdump, que pode ser utilizado para capturar e interpretar todo tipo de tráfego de rede. Esse aplicativo decodifica e exibe os pacotes à medida que eles são coletados ou armazena-os em um arquivo binário para uma análise posterior, que poderá ser feita utilizando o próprio Tcpdump ou outra aplicação.

Nas Figuras à seguir são apresentados alguns exemplos de utilização do aplicativo Tcpdump.

```
[root@forense /]# tcpdump -l -n -e -x -vv -s 1518
tcpdump: listening on eth0, link-type EN10MB (Ethernet),
capture size 1518 bytes
20:26:55.915470 00:30:b8:cd:bd:00 > Broadcast, ethertype
ARP (0x0806), length 60: arp who-has 200.165.49.205 tell
200.165.49.1
      0x0000:  0001 0800 0604 0001 0030 b8cd bd00 c8a5
      0x0010:  3101 0000 0000 0000 c8a5 31cd 2078 4000
      0x0020:  6806 f8c3 51a6 25cb bd0e c2fc cf14
```

Figura 5.1: Capturando dados de rede com Tcpdump

```
[root@forense chaos]# tcpdump -w analise.dump
```

Figura 5.2: Capturando e gravando dados de rede com o Tcpdump

```
[root@forense /]# tcpdump -l -n -e -x -vv -s 1518 -r
analise.dump
reading from file analise/dump, link-type EN10MB
(Ethernet)
21:13:50.735242 00:30:b8:cd:bd:00 > Broadcast, ethertype
ARP (0x0806), length 60 : arp who-has 201.59.94.241 tell
201.59.94.1
      0x0000:  0001 0800 0604 0001 0030 b8cd bd00 c93b
      0x0010:  5e01 0000 0000 0000 c93b 5ef1 63ec 3b71
      0x0020:  495b c745 1bca 371b 541f eda1 21f0
```

Figura 5.3: Lendo dados de rede com o Tcpdump

Na Figura 5.1, o Tcpdump é utilizado para capturar e exibir os pacotes em tempo real, no qual a opção *-l* permite a visualização dos dados em tempo de execução, a opção *-n* impede a tradução dos endereços de rede em nomes, a opção *-e* imprime o cabeçalho da camada enlace, a opção *-x* imprime o conteúdo

dos pacotes no formato hexadecimal, a opção *-vv* permite a visualização de informações extras, e a opção `-s` especifica o número de *bytes* de dados que devem ser capturados de cada pacote.

Na Figura 5.2, o Tcpdump é usado para armazenar o tráfego de rede em um arquivo binário através da opção -w. Esse arquivo poderá ser processado posteriormente através da opção -r, conforme a Figura 5.3.

O Tcpdump ainda possui um conjunto de parâmetros que permitem selecionar os pacotes que devem ser capturados, como, por exemplo, por endereço de rede, porta, protocolo e várias outras opções.

Além dos *sniffers*, os sistemas de detecção de intrusão podem ser utilizados para capturar e analisar o tráfego de rede. Existe, por exemplo, o aplicativo Snort, que, segundo (AMARAL, 2004), é capaz de decodificar e inspecionar a área de dados dos pacotes.

Tal aplicação, ao inspecionar a área de dados dos pacotes, faz uma comparação desta com as assinaturas (lista de regras) cadastradas para detectar certos tipos de pacotes que podem ser uma ameaça ao sistema, como por exemplo pacotes relacionados com varredura de portas. Quando detectado semelhante tipo de anormalidade, o aplicativo gera um alerta.

Além da análise em tempo real, o Snort pode fazer a análise após captura dos dados, inclusive os gerados pelo aplicativo Tcpdump, abordado anteriormente. Segue na Figura 5.4 um exemplo.

```
[root@servidor01 ~]# snort -vde -c /etc/snort/snort.conf
-r analise.dump -l analise.log
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file /etc/snort/snort.conf
PortVar 'HTTP_PORTS' defined :  [ 80 ]
```

Figura 5.4: Analisando saída do tcpdump com o Snort

Conforme o exemplo, o parâmetro *-d* e *-e* servem para mostrar o datagrama e o cabeçalho da camada enlace; a opção *-l* para gravar o *log*, a opção *-c* indica o arquivo de configuração do programa, de onde sairão as assinaturas para a análise dos pacotes; e a opção *-r* serve para processar o arquivo de saída a ser analisado.

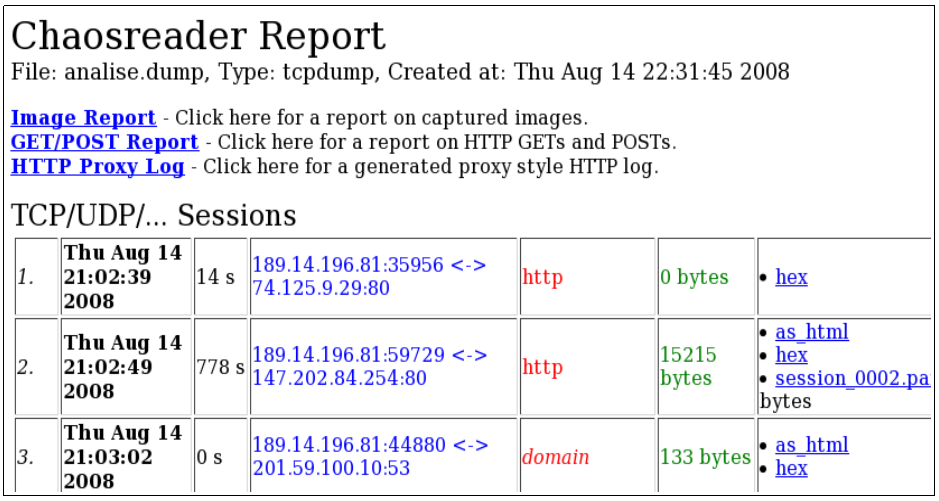
Segundo (CASEY, 2002), geralmente é aconselhável não fazer uma análise de rede em tempo real devido à grande quantidade de dados a serem analisados, por isso recomenda-se que os dados sejam gravados num arquivo externo para uma posterior análise.

Uma outra opção para a análise do arquivo de saída do Tcpdump é o aplicativo Chaosreader, que gera um relatório em HTML com a descrição de cada ação dos pacotes capturados. Esse relatório retrata de forma muito competente o que aconteceu entre os *hosts* envolvidos. Segundo (PARKER, 2005)Para gerar o relatório basta utilizar o comando da Figura 5.5.

```
[root@forense analise]# ./chaosreader0.94 -e analise.dump
-D analise/
```

Figura 5.5: Analisando e gerando relatório com o Chaosreader

Conforme pode ser apresentado na Figura 5.6, o relatório organiza todas as informações em seções de dados; estes são ordenados pelo tempo e mostram informações como o protocolo utilizado, o tamanho do pacote, o pacote em hexadecimal, e informações mais detalhadas sobre a seção. Ainda no relatório, é possível entrar em sub-itens, nos quais é possível visualizar as imagens que foram capturadas e os sites que foram acessados.



Chaosreader Report

File: analise.dump, Type: tcpdump, Created at: Thu Aug 14 22:31:45 2008

Image Report - Click here for a report on captured images.
GET/POST Report - Click here for a report on HTTP GETs and POSTs.
HTTP Proxy Log - Click here for a generated proxy style HTTP log.

TCP/UDP/... Sessions

1.	**Thu Aug 14 21:02:39 2008**	14 s	189.14.196.81:35956 <-> 74.125.9.29:80	http	0 bytes	• hex
2.	**Thu Aug 14 21:02:49 2008**	778 s	189.14.196.81:59729 <-> 147.202.84.254:80	http	15215 bytes	• as_html • hex • session_0002.pa bytes
3.	**Thu Aug 14 21:03:02 2008**	0 s	189.14.196.81:44880 <-> 201.59.100.10:53	*domain*	133 bytes	• as_html • hex

Figura 5.6: Relatório gerado com o Chaosreader

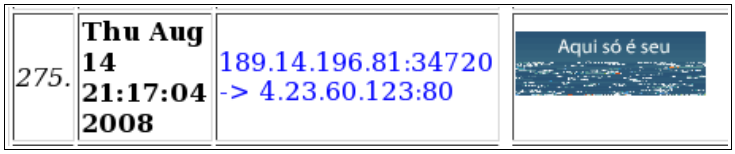


Figura 5.7: Relatório com imagens do Chaosreader

Conforme a Figura 5.7, o aplicativo conseguiu remontar grande parte de uma figura que tinha sido visualizada numa página de *Internet* no momento da captura.

O aplicativo Chaosreader ainda gera, juntamente com os arquivos do relatório, arquivos `.replay`, que são vídeos em forma de texto que simulam as ações do invasor, como por exemplo uma tentativa de acesso via `telnet`.

Conforme demonstrado, a coleta e a posterior análise de pacotes da rede são de suma importância para uma análise forense, dada a quantidade de informações que podem ser capturadas e a possibilidade de reconstrução de todo um cenário já vivido pelo usuário ou invasor do sistema.

Para compreender melhor as ações de um invasor, muitos administradores, segundo (HOEPERS, 2007), implementam os chamados *honeypots* de alta interatividade; que são computadores com sistemas operacionais, aplicações e serviços reais vulneráveis a ataques que têm como objetivo servirem de alvo fácil para os invasores. Esse tipo de estratégia propicia ao administrador a possibilidade de conhecer melhor as técnicas utilizadas pelo invasor e as ferramentas utilizadas pelo mesmo.

Todos os dados coletados servem como base de estudo para o comportamento e identificação das ferramentas utilizadas pelo invasores. Conforme já citado neste Capítulo, os aplicativos Tcpdump, Snort e Chaosreader podem ser utilizados para esta captura e análise.

Capítulo 6

Conclusão

De acordo com o exposto, conclui-se que a análise computacional forense, quando feita de forma correta, ou seja, respeitando a ordem de volatilidade das informações e os padrões para documentação com o uso da cadeia de custódia, pode ser muito útil para elucidar os crimes virtuais.

Com uma análise forense bem documentada, cria-se uma oportunidade de disseminar a informação e deixar todos os envolvidos cientes das causas e efeitos das falhas de segurança que culminaram no incidente; sendo assim de grande valia para a organização, por deixá-la imune a tais erros no futuro.

Observa-se ainda pequena, mas crescente a quantidade de aplicativos que podem ser utilizados no processo de coleta e análise de dados, muitos desses já presentes na maioria das distribuições Linux, e os que não estão, são de fácil acesso e configuração. É também notório a qualidade das informações geradas por muitas destas aplicações, que em muitos os casos geram relatórios em HTML.

Observa-se também a importância do papel desempenhado pelo perito, que, por sua vez, em muitos casos tem que usar o seu poder de decisão durante a perícia para optar entre a coleta ou não de alguns dados, pensando na preservação de outros. Caso muito comum ao manusear as informações mais voláteis.

6.1 Proposta de trabalhos futuros

Criar procedimentos baseados no que é mais comum ou mais seguro para automatizar o processo de coleta e análise das informações através de um *software* ou *script*; levando-se em consideração a ordem de volatilidade dos dados e as possíveis nuanças do ambiente de análise, como, por exemplo, o risco de perda dos dados e outros fatores a serem mapeados durante o trabalho. É desejável que haja uma interface de comunicação entre o perito e o sistema, com o intuito de parametrizar algumas regras a serem levadas em consideração durante a análise e também para definir, por exemplo, a máquina para onde os dados serão enviados.

Esses dados deverão sofrer uma análise prévia, facilitando assim o segundo nível da análise, que incluiria a do perito.

Capítulo 7

Referências bibliográficas

AMARAL, Dino; LARI, Paulo. *Snort, MySQL, Apache e ACID: guia prático*. Rio de Janeiro: Editora Brasport, 2004.

BAILEY, Edward. *Maximum RPM*. 2000. Disponível em: <http://www.rpm.org/max-rpm/s1-rpm-verify-output.html>. Acessado em 09/08/2008

CASEY, Eoghan. *Handbook of Computer Crime Investigation*. Academic Press. San Diego, California. 2002.

FARMER, Dan; VENEMA, Wietse. *Forensic Computer Analysis: An Introduction.* Setembro de 2000. Disponível em: <http://www.ddj.com/184404242>. Acesso em: 11/11/2007;

FARMER, Dan. *Perícia forense computacional*. São Paulo-SP. Editora Pearson Prentice Hall, 2007.

FARMER, Dan. *What are MACTimes?*. Outubro de 2000. Disponível em: <http://www.ddj.com/windows/184404275;jsessionid=RQTQ0DTVYXG3OQSNDLPSKHSCJUNN2JVN?pgno=1>. Acessado em: 11/11/2007;

FREITAS, Andrey Rodrigues de; *PERÍCIA FORENSE aplicada a informática: ambiente Microsoft.* p. 55. Rio de Janeiro: Brasport, 2006.

GIACOBBI, Giovani. *The GNU Netcat project*. Novembro de 2006; Disponível em: <http://netcat.sourceforge.net/>. Acessado em: 30/07/2008.

HOEPERS, Cristiane; STEDING, Klaus; JESSEN; CHAVES, *Marcelo. Honeypots e Honeynets: Definições e Aplicações*. Agosto de 2007.
Disponível em:
<http://www.cert.br/docs/whitepapers/honeypots-honeynets/#ref-06>. Acessado em: 20/06/2008.

KRUSE, Warren; HEISER, Jay. *Computer Forensics: Incident Response Essentials*. Editora Pearson. 2001.

MELO, Sandro. Notas de Aulas: *Conceitos Iniciais de Resposta a Incidente de Segurança e Computação Forense*. 2007.

MELO, Sandro. Notas de Aulas: *Exemplo de Analise de caso baseado em Network Forensic com Software Livre*. 2007.

NEMETH, Evi: SNYDER, Garth e HEIN, Trent R. *Manual Completo do Linux*. São Paulo: Makron Books, 2004

NOBLETT, Michael G.; POLLITT, Mark M.; PRESLEY, *Lawrence A.; Recovering and Examining Computer Forensic Evidence; Forense Science Communications.*outubro 2000, Vol. 2 N. 4; Federal Bureau of Investigation;

OLIVEIRA, Flávio de Souza. *Resposta a incidentes e análise forense para redes baseadas em Windows 2000*.2002 . p. 19. Universidade Estadual de Campinas, Campinas-SP.

PARKER, Don. *Studying Network Activity Using the Chaosreader Tool*. Dezembro de 2005. Disponível em : <http://www.windowsecurity.com/articles/Studying-Network-Activity-Using-Chaosreader-Tool.html#>. Acessado em 14/08/2008

REIS, Marcelo Abdalla dos. *Forense computacional e sua aplicação em segurança imunológica. Universidade.* 2003. Dissertação de Mestrado. Universidade de Campinas. Campinas, SP.

RIBEIRO, Uirá. *Certificação LINUX*. Rio de Janeiro-RJ: Axcel Books. 2004

SÊMOLA, Marcos. *Gestões da segurança da informação: visão executiva da segurança da informação: aplicada ao Security Officer*. Rio de Janeiro. Editora Campus, 2003.

Anexo

Exemplo de Cadeia de Custódia

Data/Local			
Caso Nº.			
Perito			
Descrição da Evidência			
CADEIA DE CUSTÓDIA			
De Local	Data	Razão	Para Local
De Local	Data	Razão	Para Local
De Local	Data	Razão	Para Local
De Local	Data	Razão	Para Local
De Local	Data	Razão	Para Local

Fonte: (FREITAS, 2006).