Investigação Forense Digital de Rootkits em Sistemas Unix comprometidos.

Bruno Siqueira da Silva

URI Santo Ângelo – Universidade Regional Integrada do Alto Uruguai e das Missões
Departamento de Engenharias e Ciência da Computação – 8º Semestre

brunosiqueiradasilva@gmail.com

Abstract. *This article presents a discussion of the forensic investigation of computer intrusions in Unix systems and seeks to provide grants to better understand the types of attack systems involved in these invasions caused by Rootkits.*

Resumo. *Este artigo apresenta uma discussão sobre a investigação forense de intrusões em sistemas computacionais Unix, tendo como objetivo fornecer subsídios para o melhor entendimento dos tipos de ataque envolvidos em invasões desses sistemas provocadas por Rootkits.*

1. Introdução

Quando alguém do departamento de tecnologia (TI) da empresa diz que supostamente um servidor pode ter sido vítima de um ataque (ou foi invadido), sendo que esse servidor começou a apresentar um comportamento estranho, como a faltava espaço em disco, ciclos de CPU e conexões de rede que estranhamente não apareciam no comando `netstat` ou `top`, surgem rapidamente várias preocupações: "*será que os dados da empresa foram divulgados? Nosso sistema foi copiado ou alterado? Será que nossas informações sigilosas foram roubadas? entre outras*", e essa preocupação aumenta quando os responsáveis não conseguem dizer como isso aconteceu, e mesmo após inúmeras análises no sistema não forem encontrados vestígios do problema, provavelmente, o sistema pode ter sido vítima de um ataque de Rootkit.

Esconder do administrador de sistema o fato do servidor ter sido invadido, e que o atacante tem acesso *root,* é uma função dos rootkits. De acordo com (Shadowserver, 2006) e (Murilo, 2001) rootkits auxiliam os atacantes a esconder, dos olhos do proprietário do sistema, a invasão do mesmo, de modo que possam permanecer por mais tempo no computador, utilizando-o, assim, para benefício próprio.

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

Nesse contexto e visando trazer novas contribuições a área investigação forense digital, este artigo tem o objetivo de fornecer subsídios para os profissionais de TI realizarem investigações em sistemas Unix que tenham sido afetados por Rootkits, isto é, visa construir de um processo/método de investigação para verificar a presença desses invasores, e assim, ter a possibilidade de formar as melhores respostas a incidentes provocados por eles.

1.1. Motivação

A motivação deste trabalho parte do interesse pessoal e profissional, em buscar conhecer, entender e trabalhar na área de investigação forense digital de sistemas Unix. Bem como, em

trazer ao conhecimento de profissionais de segurança da informação e do público geral a importância da investigação digital e o perigo de programas maliciosos como os rootkits.

2. Entendendo como funcionam os Rootkits

Para iniciar o processo de análise sobre rootkits é importante conhecer peculiaridades desta ferramenta. Nos sistemas Unix o usuário `root` é aquele que possui o perfil de administrador do sistema, já o rootkit, é a ferramenta que mantém ou provém esse privilégios para um invasor.

A instalação de um rootkit é normalmente o último passo de um ataque. Podem ser instalados localmente ou remotamente, via SSH, VNC, XDMCP ou qualquer outra forma de acesso remoto à sua máquina. A figura 1 exibe um exemplo da anatomia do ataque de um rootkit.

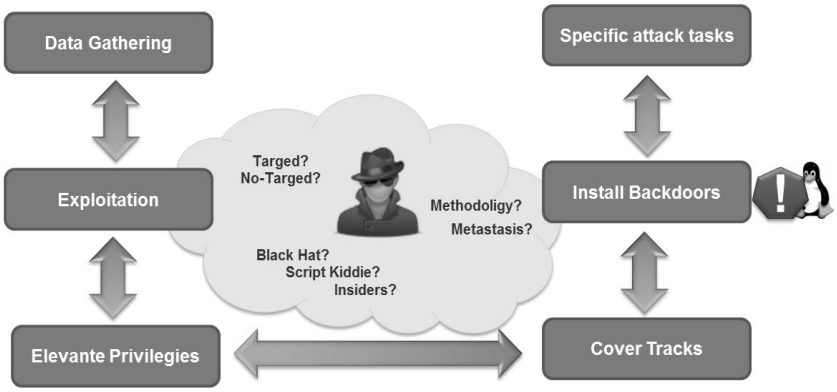

Figura 1 - Anatomia geral do ataque

Uma vez instalado, o rootkit vai modificar o comportamento da máquina alvo, mas não de modo que seja visível ao administrador. O processo do Rootkit não irá aparecer na listagem dos processos "`ls`", ou como um módulo carregado "`lsmod`" (BOVET; CESATI, 2003).

O tipo de rootkit explorado nesse artigo é tipicamente composto de versões modificadas de comandos importantes do sistema, como `ls, ps, netstat, crontab`. Os comandos substituídos podem omitir informações importantes ao administrador, tais como processos, conexões, arquivos e logs do invasor, de modo que sua presença não seja detectada. Mas também podem conter outros programas, que são úteis ao invasor.

2.1. LKM Rootkits

LKN fazem parte da 2ª geração de Rootkits (1999), surgiram com a capacidade de manipular as *system calls (ou syscalls - chamadas do sistema)* do SO diretamente por módulos carregados dinamicamente, baseados em funções de baixo nível. São utilizados em vários sistemas Unix para carregar módulos que funcionam como uma interface entre o kernel e os dispositivos físicos do computador, de modo que não geram grandes transtornos ao funcionamento do sistema (MAXWELL, 2000).

Um rootkit que trabalha em espaço de kernel pode funcionar de diversos modos, entre eles:

• *`Execution Redirection`*: Um exemplo seria um usuário chamando o `/bin/bash`, o shell padrão do Linux e o kernel ao interceptar esta execução, redireciona para um programa qualquer, em `/var/tmp/.trojan/chamaleon`. Este programa, se utilizado, poderia possuir todas as funcionalidades de um shell padrão do Linux, acrescidos de funcionalidades de um *backdoor* para possibilitar o acesso de root. O problema de detecção desta técnica encontra-se

na checagem de integridade de sistema de arquivos, pois, como o binário `/bin/bash` se encontra intacto estes programas não encontrariam nada errado.

• *`File Hiding`*: Ao invés de modificar arquivos como o comando `ls`, o atacante manipula as *syscalls* que acessam os arquivos, de modo que ocultem arquivos que o atacante não queira que sejam mostrados.

• *`Module and symbol hiding`*: os LKMs podem ser ocultados interceptando syscalls ou acessando diferentes arquivos dentro do diretório /`proc`, ou manipulando as estruturas de kernel. A ideia é a de filtrar as informações de um ataque ao administrador do sistema.

• *`Process Hiding`*: usa os métodos similares aos citados anteriormente, o kernel engana os comandos `ps` e `lsof`. P. ex., na função `main()` na linguagem C, há dois argumentos especiais que podem ser passados `argc` e `argv`. O parâmetro `argc` contém sempre pelo menos um inteiro, enquanto `argv` é um ponteiro. Nessa situação, um invasor pode ocultar suas ações em um sistema, manipulando a posição 0 em `argv[0]` de um programa mostrando um nome diferente do original quando os comandos de controle de processos forem executados.

• `Network Hiding`: o Rootkit modifica as saídas mostradas pelos programas `netstat` e `lsof`.

• `Sniffer Hiding`: altera as mensagens do kernel referentes ao uso do modo promíscuo em interfaces de rede quando um sniffer está em execução.

• *`Suckit`*: Em 2001, foi publicado na revista eletrônica Phrack, intitulado de "*Linux on-the-fly kernel patching without LKM*" (DEVIC, 2001), que seria a nova escola de rootkits. Esta é uma ferramenta que consegue modificar as funções do kernel do Linux em tempo real, sem a necessidade de carregamento de nenhum módulo de kernel, tornando-se o estado da arte, já que não havia como detectá-la mediante os processos já utilizados por programas de segurança, que localizavam e neutralizavam seus antecessores.

Uma maneira de detectar esta ferramenta foi à tentativa de auditar qualquer modificação em arquivos ou diretórios do sistema, já que, o mesmo precisa colocar seus arquivos no sistema. O `Suckit` cria uma tabela particular de *syscalls* desviando o ponto de execução do kernel para a mesma (PELÁEZ, 2004). Em suma, o que este rootkit faz, de uma maneira mais complexa é criar uma tabela própria através de comparação de bytes das informações das *syscalls* e com isso redirecionar o kernel para uma espécie de tabela própria de símbolos e reescrever o *kmem*.

O processo de localização e neutralização destes rootkits dá trabalho e exige que os administradores de sistemas tenham cuidado ao manipular o sistema invadido. As soluções iniciais são analisar os módulos carregados e a enumeração de processos no diretório /*proc*. Infelizmente, pela sua dificuldade de detecção, ainda hoje, estas ferramentas são encontradas em sistemas na internet, causando problema a administradores que não implementaram uma política de segurança em seus sistemas computacionais.

3. Como planejar uma investigação?

Esta é uma das partes mais importantes para um trabalho de investigação. Tendo em mentes as peculiaridades de um rootkit, cria-se uma necessidade de métodos direcionados para cada tipo de perícia computacional. Douglas Schweitzer (SCHWEITZER, 2003) apresenta um modelo, baseado em *checklists* que podem ser usados em análises genéricas, como um algoritmo. Isto facilita na apresentação dos resultados em juízo ou a uma diretoria de empresa a procura de respostas ao incidente ocorrido.

Outro ponto interessante na utilização de um checklist é que ele fornece uma visão de todo o processo a ser efetuado. Assim, pode-se escolher o melhor conjunto de ferramentas a utilizar para a criação de um *toolkit*.

No processo de preparação para o início de uma investigação, é importante realizar uma cópia fiel do ambiente comprometido, o que possibilita efetuar diversos testes sem que o ambiente original seja alterado. Nesse processo é interessante o uso de *hash*, que indica claramente se algo foi aberto ou violado, ou se a cópia é idêntica ou não.

4. Live Analysis

Em um primeiro contato com um sistema invadido, é necessários escolher o modo como será investigação. A Live Analysis pode ser definida como a análise feita sem o desligamento abrupto do mesmo. Esse tipo de análise dá a oportunidade de coletar informações que não estariam disponíveis caso o sistema fosse desligado.

Contudo, há a possibilidade de ainda haver um invasor na máquina, o controle da mesma é extremamente difícil. O atacante ao perceber a presença ou ação de um investigador pode tentar apagar seus rastros. Por isso é importante, copiar o conteúdo da memória principal do sistema, pois é local onde estão contidas as informações daquele momento específico do sistema computacional, e isso pode ser feito com o comando `dd`.

Em uma investigação digital, também é importante que se recupere o estado do sistema operacional naquele momento, assim, dando um maior número de informações sobre os processos que se encontram em execução naquele momento (ATÍLIO, 2003).

4.1. Comandos e Diretórios úteis

Existem várias formas de se acessar as informações relacionadas aos processos em execução em um sistema operacional Linux, onde os principais estão listados a seguir:

Comando `uptime`: obtém a informação sobre o período de tempo que a máquina está ligada, além da média de carga da máquina nos últimos 1, 5 e 15 minutos.

Comando `ps`: fornece uma lista detalhada dos processos em execução.

Comando `top`: lista os processos que estão ocupando mais CPU naquele momento.

Comando `lsof`: lista os arquivos abertos no momento da execução de um processo.

Diretório `/proc`: contém as informações de hardware e de processos em execução. Este diretório é um pseudo do sistema de arquivos, que pode ser escrito somente pelo kernel.

Cada processo cria dentro do `/proc` há um arquivo chamado `cmdline`, que é a linha de comando completa utilizada para executá-lo. O seu conteúdo pode ser manipulado para não fornecer as informações corretamente ao investigador, utilizando a técnica de manipulação do conteúdo de `argv[0]`.

Em conjunto com esta técnica, os invasores costumam executar um programa e após isto, o apagam. No Linux, mesmo que tal ação tenha sido efetuada, o binário (executável correspondente ao programa) não será excluído até que o processo que o esteja utilizando seja terminado ou o sistema operacional desligado. Para se detectar uma ação deste tipo, é possível utilizar da interface fornecida pelo `/proc`. Nesse diretório, há um arquivo de nome `exe`, que é que um link simbólico para o binário do programa que foi executado (ATÍLIO, 2003).

Comando `lsof:` pode ser uma interface de acesso ao diretório /`fd`, pois o mesmo lista todos os arquivos abertos por um processo, tornando este processo menos penoso para quem estiver realizando a investigação.

4.2. Usuários ativos e informações de *login*

Determinar quais usuários estão ativos no sistema pode fornecer informações importantes para a investigação. Isto pode ser determinado com o uso do comando `w e last`.

O comando `w` lista, entre outras informações, os usuários logados, o endereço do sistema a partir de onde fez login e o horário, além dos utilitários que o mesmo está executando.

O comando `last` fornece informações se o usuário está logado ou não, local de onde o usuário logou (no caso de login remoto), data e tempo que o mesmo ficou conectado no sistema.

4.3. Conexões de Rede

Através da análise das conexões de rede e portas TCP/UDP em atividade, é possível ter uma ideia do tipo de utilização que certa máquina está fazendo da rede em um dado momento, assim como os seus processos. Esta é a única oportunidade de coletar este tipo de informação volátil que não deixa nenhum tipo de rastro ou histórico depois que a conexão ou atividade é extinta, a não ser que cada aplicação cuide disso através da alimentação de arquivos de log, com alguma ferramenta específica, como o TCP-Wrapper.

A associação de serviços Internet, como Telnet e HTTP, e suas respectivas portas, fornecem uma ideia do tipo de atividade que gera certo tipo de conexão. Note que a interpretação da atividade de rede de uma máquina pode não ser uma tarefa trivial, pois às vezes não é possível obter informações para que possam ser formuladas hipóteses viáveis.

Uma conexão utilizando a porta 80, p. ex., pode ser um navegador ou um cavalo de Tróia utilizando uma porta que geralmente tem seu tráfego permitido na maioria dos firewalls. Sendo assim, existe a necessidade de interpretação dos dados que estão sendo transferidos. Um programa utilizado para capturar o tráfego de rede é o `tcpdump`, que permite decodificar e exibir datagramas à medida que são coletados ou armazenar os datagramas em formato binário, permitindo análise posterior (ATÍLIO, 2003).

Há também a possibilidade de usar *sniffers*, onde ocorre maior tráfego da máquina invadida. Ou ainda, pode se usar em switches gerenciáveis o recurso de *port mirroring*. Quanto maior o número de dados coletados, menor a possibilidade de que muitos datagramas sejam descartados durante a análise. Essa análise, por sua vez, é feita com programas que reconstroem e exibem as informações em um formato mais adequado. O `ethereal` permite a reprodução da sessão capturada, além da visualização dos eventos de uma maneira mais fácil do que o `tcpdump`. Há ainda, ferramentas do próprio SO que podem ajudar, como o `Arp, Netstate` e `Tracertroute`.

5. Post Mortem Analysis

O segundo modo de investigação é o *Post Mortem*, realizado após o SO ser desligado (ocasionado por uma falha do sistema ou por desligamento abrupto). É neste momento da análise que ocorre uma das fases mais importantes do processo: a análise do sistema de arquivos.

O disco rígido de um sistema computacional é uma fonte importante de evidência, pois os dados estão espalhados sobre toda a área do disco, mesmo quando excluídos. Quando excluídos, a

referência de um arquivo é removida das estruturas do sistema de arquivos, mas os dados continuam no disco. A remoção só ocorre, quando a área onde estavam é sobre escrita (SILBERSCHATZ; GALVIN, 2000) (WOODHULL, 2000).

Uma ferramenta que pode auxiliar é o `LKCD2`, que permite recuperar o estado do sistema antes de sua parada. Por *default*, ele guarda uma imagem do sistema que pode ser acessada via arquivo `/var/log/dump`.

Este log pode ser acessado via `lcrash`, contida no kit do LKCD, que tem prompt próprio onde podem ser observados, inclusive, os processos que estavam sendo executados em tempo de boot. Para utilizar esta função é necessário que o kernel tenha configurado a opção de `Sysreq`.

É importante lembrar que somente com o pedido de `Sysreq` que a ferramenta `lcrash` será ativada e irá retornar um prompt, após a máquina reiniciar, para análise do dump da memória. Esta ferramenta fornece uma possibilidade ao perito, de, no caso de um desligamento, ou um crash geral da máquina, conseguir analisar após o religamento da máquina, o momento anterior em que a mesma se encontrava (IBM, 2002).

Esta ferramenta se encontra no limiar de uma Live Analysis com a PostMortem Analysis. Isto pode ser observado, pois ela fornece um retrato do sistema anteriormente ao seu desligamento, para uma análise posterior.

5.1. Arquivos suspeitos, diretório e uso do disco

Um dos aspectos de uma invasão relacionada com rootkits é o uso de disco. Por este motivo, a preocupação com a integridade do sistema de arquivos deve ser alta. Um programa simples, que pode ser implementado é o `quick_lstat`, que é um aplicativo básico de IDS (ou para a manutenção de um sistema). Ele objetiva detectar alterações nos arquivos do sistema para identificar trojans, cavalos de troia, checar sniffers e keyloggers nos devices da rede.

O uso do `quick_lstat` ainda se mostra interessante quando houver o uso dos *`logservers`*, comum em ambientes de Cloud Computing. Outra vantagem é a possibilidade de personalização do programa, permitindo a alteração do nome dos arquivos que o programa gerencia.

Para realizar uma busca detalhada no sistema de arquivos, Raul Silez, propõe o uso do comando `find`, que aceita diversas configurações para encontrar arquivos. Variações do `find` podem levar a encontrar evidências importantes que não seriam encontradas com outros métodos.

Outra informação importante para análise é conhecer os diretórios ocultos. No Unix eles iniciam com um ponto ".", e os atacantes costumam colocar seus arquivos em diretórios nomeados com:

• ".. " dificulta a detecção devido a parecer com o nome padrão do diretório local no Linux;

• ". " dificulta a localização, já que o mesmo está oculto e sem um nome facilmente detectável;

• "..." se parece em muito com o nome padrão do diretório atual dentro do Linux;

• " " palavras vazias, não são detectadas em uma listagem rápida de um diretório.

5.2. Contagem dos hard links e total de blocos

Cada arquivo do Linux é associado um número para cada hard link. Em cada diretório há dois hard links, um correspondente ao próprio diretório (.) e outro ao diretório pai (..). Quando um atacante usa um rootkit de modo incorreto, ele tenta esconder os subdiretórios.

Em uma situação em que um diretório contém 3 subdiretórios, a contagem de hard-links deve ser 5, 2 associados ao padrão de cada diretório e 3 ligado aos diretórios contidos no mesmo. Quando um atacante esconde arquivos no sistema, a contagem será diferente da listada via `ls`.

5.3. MACTimes

Os MACTimes são fontes importantes em um processo de análise. Porém, um atacante pode adulterá-lo. Ao substituir um arquivo a informação obtida, nesses casos, é de que o arquivo não foi modificado, mas na verdade pode ser o contrário.

Uma das técnicas básicas de verificação do sistema de arquivos é a de conferir os `tamanhos` e o `timestamp` dos arquivos binário. Lembrando que nos sistemas Unix os arquivos têm três tempos: tempo de modificação (mtime), tempo de acesso (atime) e change time (ctime).

O comando `stat` pode fornecer estas informações sem problemas, mas o TCT (The Coroners ToolKit) fornece uma ferramenta mais apropriada, o comando `mactime`. O TCT é um conjunto de ferramentas para o exercício da perícia forense, em sistemas operacionais Unix Like. Raul Silez apresenta três usos deste comando para captar informações para futura referência, utilizando o comando `ls`, conforme é mostrado abaixo.

```
# ls -alRu / > access_times.txt
# ls -alRc / > change_times.txt
# ls -alR / > modification_times.txt
```

6. Estudo de Caso

O estudo de caso desenvolvido nesse artigo consiste em uma análise do comportamento de um Rootkit de *kernel* a partir da filtragem das *syscalls*.

Como dito anteriormente, um rootkit tem a função de modificar o comportamento de um SO, para torná-lo mais facilmente acessível ao invasor. Mas para que isso aconteça o invasor deve conseguir gravar seus dados em disco, e é nesse momento que se inicia a investigação. Essas modificações são as evidências são importantes para análise do comportamento desses rootkits.

Com base nessas evidências, é possível acrescentar que a filtragem `system calls execve()` se mostra importante, pois assim, é possível estudar todo o caminho efetuado por um rootkit ao comprometer um sistema computacional.

6.1. Rootkit Adore-ng

O *Adore-ng* é um dos rootkits mais conhecidos para os sistemas operacionais com base Unix. Ele tem seu funcionamento similar ao rootkit Adore, com a peculiaridade de subverter o sistema a partir de outro método. Ele substitui as funções de listagem de diretórios por rotinas próprias, provendo ao administrador as informações que o invasor quiser sobre o sistema de arquivos e o diretório `/proc` (STEALTH, 2004b).

Foi criado por Stealth[1], membro do grupo de segurança TESO[2]. O Adore-ng foi implementado como um módulo de kernel (LKM) e manipula a camada de abstração do sistema de arquivos do Linux, chamada de Virtual File System (VFS).

[1] http://www2.gmer.net/mbr/
[2] http://www.markosweb.com/www/team-teso.net/

Por ser implementado dessa maneira, ele consegue ocultar sua presença com facilidade. Fica invisível, nem o comando `ls` consegue enxergá-lo. Ou seja, o próprio kernel já estará filtrando este resultado, deixando assim o administrador sem armas, visto que não encontraria nenhum resultado aparente.

6.2. Ferramentas antes da Intrusão

O ambiente de análise forense proposto neste trabalho consiste de duas partes distintas. Uma consiste das ferramentas *AIDE*[3] e *Snoopy Logger*[4], ambas utilizadas para preparar o sistema operacional para gerar os dados para a detecção de uma intrusão no SO.

A outra consiste na escolha de ferramentas para formar um Toolkit capaz de fornecer uma plataforma de trabalho amigável e útil, para que o trabalho chegue a resultados realmente aproveitáveis. Por este motivo, nesta seção, cada ferramenta utilizada na detecção do rootkit Adore-ng será analisada para demonstrar sua importância no processo.

6.2.1. Aide

Uma das formas de efetuar um estudo e catalogar as alterações efetuadas em um disco é o uso de um IDS relacionado ao sistema de arquivos do sistema computacional. Um IDS deste tipo é importante e indicado para os diretórios de um SO, pois são alvos de um invasor em geral todos os diretórios de um sistema.

Aa opção utilizada para os trabalho, foi o software /textitAIDE. Este software é um sistema que checa a integridade de arquivos e diretórios definidos em um arquivo de configuração, onde são inseridas as políticas de atuação do mesmo. Ele ainda utiliza vários algoritmos de hash: md5, sha1, md1600, tiger, crc32, hava1 e gost.

O seu funcionamento consiste na criação de um banco de dados inicial que contém as informações com o estado do sistema no momento em que se executa o AIDE pela primeira vez.

A partir de sua segunda execução o AIDE vai comparar o estado do sistema com o banco de dados, gerando um log com as inconsistências encontradas.

Foi ainda adicionado no cron do sistema o comando do software AIDE utilizado para gerar os logs. Esta utilização do cron se faz necessária, pois o AIDE não possui um daemon específico que faz a varredura do sistema em intervalos pré-definidos (ver comando cron abaixo).

```
00 23 * * * /usr/bin/aide -C > /var/log/aide.conf
```

Assim, a utilização do cron é importante para ter uma leitura do sistema de arquivos em intervalos de tempos pré-definidos.

6.2.2. Snoopy Logger

O Snoopy Logger foi escolhido por ser capaz de gerar logs de todos os comandos executados por usuários do sistema ou apenas os comandos executados pelo usuário root.

O programa funciona como um agente, que loga todas as chamadas de executáveis, no `/var/auth/log`. Isto fornece, em uma análise post mortem, a possibilidade de saber quais

[3] http://aide.sourceforge.net
[4] http://sourceforge.net/projects/snoopylogger

executáveis foram chamados durante um certo período e assim, conseguir dados de subversão de funções do kernel, através do carregamento de LKMs, via comando `modprobe`, por exemplo.

O método utilizado pelo *Snoopy Logger* fornece dados que podem se correlacionar com outros, através da análise dos ELFs (é o formato binário, base dos executáveis mais comuns dos sistemas Unix) abertos por cada processo, através dos seus ponteiros. Esta consulta pode ser comparada a feita utilizando o comando `grep` no arquivo `System.map`, que, em geral, está localizado no diretório `/boot` das distribuições Unix (ver comando abaixo).

```
root@teste$ grep formats /boot/System.map
c039050c b formats
c03906c0 b quota_formats
c0391bc0 B cvf_formats
```

Obs.: O comando acima permite coletar ponteiros de ELF no arquivo System.map

A instalação do *Snoopy Logger* na distribuição Debian 6 foi feita através da compilação do código fonte diretamente no sistema operacional aonde o mesmo vai detectar os redirecionamentos de execução.

6.3. Ferramentas depois da Intrusão

Mesmo com um sistema concebido previamente para fornecer informações ao administrador, é necessário ter em mãos um grupo de ferramentas que possam ajudar a obter informações em um sistema invadido.

As ferramentas apresentadas nesta seção podem ser utilizadas em perícias Post Mortem como Live Analysis.

6.3.1. De-termine

De-termine é uma ferramenta desenvolvida pelo próprio criador do rootkit Adore-ng. Ela ajuda os administradores a procurar processos escondidos pelo rootkit, baseando-se nos conceitos que o mesmo utilizou ao criar o Adore-ng.

6.3.2. Chkrootkit

O chkrootkit consiste de um conjunto de ferramentas que tem como função testar a presença de rootkit instalados em um sistema computacional, usando a técnica de comparação com assinaturas.

6.3.3. Rkscan

Este é um pequeno scanner para rootkits. Ele detecta um pequeno conjunto de rootkits que são o *`Adore`* em suas versões 0.14, 0.2b e 0.24 e outro de nome *`knark`*. A importância desta ferramenta em um kit de resposta é pela sua especialização na detecção de versões específica do rootkit Adore-ng.

6.3.4. Distribuição FIRE[5]

A distribuição FIRE, anteriormente conhecida como *Biatchux*, é um Live CD que contém 196 ferramentas voltadas para a análise forense e Pen Test[6], e possui seis tipos de ferramentas

[5] http://fire.dmzs.com/

[6] Penetration Test - testes de penetração em sistemas computacionais.

básicas: Ferramentas básicas do sistema operacional; Ferramentas de forense e recuperação de dados; Resposta a incidentes; Pen Test; Associação binária estática e Escaneamento de vírus.

É uma distribuição baseada no Knoppix, utilizada atualmente para criar diversos live cds. A grande vantagem do FIRE é limitar o tempo de procura de ferramentas na internet para criar seu próprio Toolkit. O FIRE possui dois Tooltkits de análises forenses muito famosos em sua estrutura, o *TCT*, *The Coroner's Toolkit* e o *The Sleuth Kit*[7].

6.3.5. Autopsy[8]

Esta é uma ferramenta de auditoria forense, com interface web. Tem suporte aos sistemas de arquivos NTFS, FAT, UFS1/2, Ext2/3. Faz parte do *Sleuth Kit* e pode ser utilizado tanto para live analysis como para post mortem analysis. A procura de evidências pode ser feita de acordo com as seguintes técnicas: Listagem de arquivos, Conteúdo do arquivo, Hash Databases, Organização por tipos de arquivos, Timeline das ações, Procura por palavras, Análise de Meta Data, Data Unit Analisys e Detalhes do sistema de arquivos.

7. Ambiente de Análise

Para coletar as informações referentes ao comportamento dos rootkits em sistemas operacionais Linux, foi necessário a criação de um ambiente de análise (teste). Este ambiente foi criado a partir de máquinas virtuais, utilizando o software Xen Center 6.0. Estas máquinas virtuais foram instaladas utilizando a distribuição Linux DEBIAN Lenny 6, onde foram inseridos os softwares AIDE 3.4.1 e Snoopy Logger 3.4.2.

O XenCenter 6.0 permite a criação de clones das máquinas virtuais criadas. Assim, em experimentos que envolvam a subversão do sistema operacional, como o presente trabalho, é possível recuperar o sistema em seu estado inicial, para que seja possível testar as diversas interações de um *malware* com o sistema operacional hospedeiro.

Para a criação da máquina virtual, baseada no rootkit Adore-ng, foi necessário efetuar o download de uma versão operacional (versão 0.56) do mesmo, que foi encontrada em um dos sites do autor[9].

Após a configuração do Adore-ng, é necessário utilizar o comando `make`, para gerar os binários necessários ao funcionamento do rootkit. O Adore-ng é iniciado utilizando o comando `startadore`, contido no diretório com os arquivos fontes. Este aplicativo carrega os módulos necessários ao funcionamento do rootkit, para que o mesmo possa efetuar suas funções de manipulação do VFS do Linux.

Uma das funcionalidades mais interessantes desse rootkit após o carregamento é que o mesmo torna o diretório onde seus binários estão localizado, invisível. Um comando `ls` executado no local onde estão os binários deste programa, simplesmente, não irá mostrar mais este diretório, tornando a investigação mais difícil, já que tal artefato trabalha diretamente nas funções de kernel do sistema computacional.

```
root@teste:/usr/src/. # ls
adore-ng-0.53.tgz backdoor*   backdoor.c
root@teste:/usr/src/. # pwd
/usr/src/.
```

[7] http://www.porcupine.org/forensics/tct.html
[8] http://www.sleuthkit.org/autopsy/
[9] http://stealth.openwall.net/rootkits/

A ilustração acima mostra o diretório após o carregamento do rootkit Adore-ng.

Utilizando esse backdoor, pode-se perceber que ele atendeu bem aos resultados que eram esperados. O aplicativo `ava` foi utilizado para tornar o processo invisível. Veja comandos utilizados:

```
root@teste:/usr/src/. /adore-ng# ./ava i 339
Checking for adore 0.12 or higher ...
Adore 1.53 installed. Good luck.
Made PID 339 invisible.
root@atalibateste:/usr/src/. /adore-ng# ps ax | grep kswap | grep -v grep
4 ? S 0:00 [kswapd]
root@teste$:/usr/src/. /adore-ng# ls /proc | grep 339
```

A partir deste ponto, os rastros de uma intrusão efetuada com o rootkit Adore- ng será terminada.

Um invasor teria aqui um sistema apto a receber sua visita, sempre que necessário, pois haveria sempre uma porta aberta, para uma conexão sem problemas. A partir destes passos foi criada uma máquina virtual para os testes relacionados ao rootkit Adore-ng.

8. Resultados Obtidos

Uma perícia sempre é iniciada a partir de observações de funcionamentos anômalos do sistema computacional. A partir do momento em que uma equipe de forense é acionada, ela irá ao local pronta para receber uma máquina com diversos problemas ou, ainda com o invasor efetuando alguma ação dentro da mesma. No caso dos rootkits uma análise preliminar é que irá fornecer informações sobre as ações efetuadas pelo invasor nesse sistema computacional.

A ferramenta utilizada para alguns dos testes foi a FIRE. Em sua ISO, esse programa fornece a ferramenta ***chkrootkit***, que foi utilizado para demonstrar a presença de algum rootkit, que nesse aso, o resultado obtido não encontrou a presença de nenhum programa malicioso.

Isto se deve, principalmente, pelo *chkrootkit* não possuir em sua listagem as assinaturas de mudanças efetuadas no sistema pelo rootkit Adore-ng. Um detalhe a ser observado, é de que, apesar de possuir uma entrada Adore em sua listagem, esta se refere ao worm de mesmo nome. Nesse momento pode-se também perceber que infelizmente o FIRE não forneceu o conjunto de ferramentas pré-compiladas de fácil acesso, somente com a montagem do CD.

De posse dos logs obtidos pela ferramenta ***AIDE*** foi possível observar que um diretório de nome suspeito foi criado abaixo do diretório /usr/src/. Dentro deste diretório, vários arquivos suspeitos foram gravados, e que caracterizaram a intrusão efetuada com um rootkit qualquer. Os nomes dos arquivos remetem a presença do rootkit Adore-ng, no qual deverá ser o alvo de estudo de uma equipe de Forense nos próximos passos.

Como o sistema possui o ***Snoopy Logger*** instalado, uma pequena busca nos logs trouxe informações interessantes, principalmente por poderem ser comparadas com as obtidas nos logs do AIDE. De posse desses resultados foi possível visualizar a presença do rootkit Adore-ng no sistema.

De acordo com os resultados obtidos através do comando **`grep`**, foi possível observar que realmente arquivos suspeitos foram criados abaixo do diretório /usr/src. Neste diretório foram guardados códigos fonte que representam algum rootkit. Caso os profissionais forenses procurassem o diretório através do comando **`ls`**, não encontrariam, já que o invasor se preveniu tornando este diretório invisível aos comandos do Linux.

```
#cat /var/log/secure | grep adore
...
Nov 20 09:06:07 teste snoopy[226]: [root, uid:0 sid:190]:pico root/adore.ng
Nov 20 09:07:09 teste snoopy[240]: [root, uid:0 sid:190]:adore-ng/ava h 238
Nov 20 09:09:12 teste snoopy[244]: [root, uid:0 sid:190]:./startadore
Nov 20 09:11:10 teste snoopy[245]: [root, uid:0 sid:190]:insmod ./adore-ng.o
...
```

O próximo passo foi à utilização do programa **`rkscan`**. Infelizmente, nos testes efetuados, ele também não conseguiu detectar o Adore-ng. Como a versão utilizada nos testes foi a 0.56, não suporta por este scanner ele simplesmente apresentou um resultado tal qual o chkrootkit, sem valor para à investigação (ver resultado do *rkscan*).

```
teste:~$ ls
rkscan1.0.c
teste:~$ gcc -o rkscan rkscan1.0.c
rkscan1.0.c: In function 'adore_scan':
rkscan1.0.c:46: warning: comparison between pointer and integer
teste:~$ ./rkscan
-=- Rootkit Scanner -=-
-=- by Stephane.Aubert@hsc.fr -=-
Scanning for ADORE version 0.14, 0.24 and 2.0b ...
ADORE rootkit NOT DETECTED on this system.
Scanning for KNARK version 0.59 ...
KNARK rootkit NOT DETECTED on this system.
Done.
```

A última opção para detecção do rootkit Adore-ng foi o software **`de-termine`**. Esse foi único capaz de detectar a presença deste rootkit. Assim, poderá se tornar uma peça importante em um kit de resposta (ver resultado abaixo).

```
teste:~# cd determine
teste:~/determine# ls
CVS/ Makefile README detect.c detect.h main.c signatures/
teste:~/determine# make
cc -Wall -O2 -c main.c
cc -Wall -O2 -c detect.c
cc main.o detect.o -o determine
teste:~/determine# ./determine
deter-mine LKM rootkit detector. (C) 2004 Stealth
Trying to detect hidden processes ...
Process with PID 339 does not have a appropriate /proc entry. Hidden?
Done.
Scanning /dev/mem for signatures. This may take a while ...
Signature of 'Adore/Adore-ng LKM' detected!
Unusual behaivior has been detected. Please consult the removal chapter of
the README-file.
teste:~/determine#
```

Através do comando `netstat`, também nativo do Linux foi possível visualizar um estranho programa escutando na porta 20000. Os dados do comando `netstat` não são muito exatos, mas possuem a possibilidade de ser complementados com o uso do comando `lsof`.

O comando lsof lista os arquivos abertos por um processo, bem como também suas conexões de rede. Assim, foi interessante utilizar este comando para tentar conseguir chegar ao programa que está utilizando a porta 20000.

Infelizmente, devido ao uso do programa ava em paralelo com o rootkit Adore, as conexões de rede foram escondidas dos comandos do Linux, o que tornou esta tarefa um pouco mais difícil para o perito (ver resultado abaixo).

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 0.0.0.0:20000 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:37 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:113 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN
udp 0 0 0.0.0.0:512 0.0.0.0:*
udp 0 0 0.0.0.0:37 0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags Type State I-Node Path
unix 3 [ ] DGRAM 232 /dev/log
unix 2 [ ACC ] STREAM LISTENING 706 /dev/gpmctl
unix 2 [ ] DGRAM 243
teste:/usr/src/. # lsof | grep 20000
```

Uma intrusão com o rootkit Adore-ng em um ambiente de produção seria muito difícil de ser detectada com pequenas ações. A presença do AIDE se mostrou importante, pois, as mudanças em um disco rígido são necessárias em algum momento da invasão.

8.1. Análise dos resultados

Na arquitetura em que o trabalho foi desenvolvido, não foi possível levantar muito do que um ataque com rootkits pode causar ao sistema. Isto aconteceu devido aos testes efetuados serem em sistemas computacionais criados especiicamente para este estudo, não demonstrando realmente como um invasor poderia efetuar isto em um sistema real.

Apesar disto, os testes aplicadas no artigo, são as mais divulgadas em trabalhos pela internet, como citado por Antônio Marcelo, um dos responsáveis pelo projeto Honeypot-BR.

Um dos pontos interessantes a ser observar, é, que, com uma estrutura que mantenha um bom histórico do que é efetuado em uma rede durante períodos, é possível levantar com muito mais facilidade as características dos ataques.

Em relação ao Adore-ng, é importante salientar que o mesmo trabalha em espaço de kernel, efetuando suas mudanças de modo tão transparente, que até administradores muito experientes podem não detectá-lo com as auditorias de rotina.

Um dos modos mais interessantes para a detecção de ataque em sistemas computacionais é o uso de uma ferramenta como o software **Autopsy**. Ele, por suas características, pode fornecer a possibilidade de criar uma linha de tempo dos acontecimentos em um sistema computacional comprometido.

Outro detalhe importante que se pode observar é que um ataque padrão normalmente acontece com a detecção de uma vulnerabilidade em um sistema computacional e sua respectiva exploração através de um ***exploit*** ou outro programa específico. Nesses casos, o scanner mais utilizados pelos hackers para obter informações do sistemas é o **nmap.**

Assim, a partir da obtenção do shell de root a máquina se torna apta a ser controlada pelo invasor. O Adore-ng, nesse caso, age no sistema diretamente em seu núcleo. Sua principal característica é subverter as funções do kernel referentes ao VFS tornando sua detecção em uma *Live Analysis* quase impossível.

Com o software ***Authopsy*** foi possível, em uma análise post mortem, detectar os binários do Adore-ng e os códigos fontes referente ao backdoor que se encontrava instalado no SO. A partir deste software foi possível construir uma linha de tempo de como ocorreu o ataque e assim, comparando com os logs obtidos nos softwares AIDE e Snoopy Logger foi possível detectar que, o mais importante, em uma invasão deste tipo, é manter um sistema com métodos de detecção de intrusão eficazes.

9. Conclusão

A base para um sistema computacional seguro, é a intervenção humana feita de modo acertado. Assim, o administrador deve levar em conta que servidores são máquinas que precisam ter características diferentes de uma estação de trabalho.

O conhecimento das ações dos rootkits em sistemas operacionais Linux é uma arma para proteger-se de futuras intrusões, uma vez que são artefatos que ao longo dos anos estão evoluindo, e cada vez mais se faz necessário a proposta de um método para aplicação em perícias forenses relacionadas a eles.

O presente trabalho mostrou uma pequena faceta de como trabalham os rootkits. Como pôde ser visto a sua detecção se torna cada vez mais difícil aos profissionais de TI (administradores, peritos, etc.), forçando um dos mesmos a instalar em seus sistemas computacionais diversas ferramentas para auxiliá-lo na ocorrência de uma intrusão.

Em relação a trabalhos futuros, relacionados a este tema, é interessante implementar uma *honeypot* onde poderão ser estudadas as ações de invasores em um ambiente real, e os efeitos que os *rootkits* instalados teriam nos resultados de uma investigação.

Assim, os dados obtidos seriam mais condizentes com a realidade de uma invasão no mundo real. Além disto, é possível propor a mesma estrutura de trabalho para detecção de *rootkits* em sistemas Windows e nos Unixes-Like da família BSD.

Sobretudo, acredita-se que o presente artigo forneceu informações importantes sobre como trabalha um *rootkit* quando instalado em um sistema operacional e como um administrador deve proceder para defender melhor seus sistemas de ações deste tipo.

10. Bibliografia

Antirootkit.com. (2005). *Anti-Rootkit Software - Detection, Removal & Protection.* Acesso em 10 de Outubro de 2011, disponível em < http://www.antirootkit.com/software/index.htm>

Aubert, S. (2010). *Rkscan: Rootkit Scanner for Load-able Kernel-module Rootkits*. Acesso em 13 de Setembro de 2011, disponível em <http://www.hsc.fr/ressources/outils/rkscan/index.html.en>

Avertlabs. (2009). Malware Is Their Business...and Business Is Good! Acesso em 29 de Outubro de 2011, disponível em <http://www.avertlabs.com/research/blog/index.php/2009/07/22/malware-is-their-businessand-business-is-good/>

Chkrootkit. (2009). Locally checks for signs of a rootkit. Acesso em 01 de Novembro de 2011, disponível em <http://www.chkrootkit.org/>

Chuvakin, A. Peikari, C.. Security Warrior. O'Reilly, January, 2004.

Daviel, A. (2009). *Rkdet: Rootkit detector for linux*. Acesso em 2011 de Setembro de 13, disponível em <http://vancouver-webpages.com/rkdet>

dos Reis, M. A., & de Geus, P. L. (2001 a). *Computação forense: Procedimentos e padrões*. Acesso em 13 de Setembro de 2011, disponível em <http://www.las.ic.unicamp.br/paulo/papers/2001-SSI-marcelo.reis-forense.padroes.pdf>

dos Reis, M. A., & de Geus, P. L. (2001 b). Acesso em 10 de Setembro de 2011, disponível em Modelagem de Sistema Automatizado de Análise Forense: Arquitetua extensível e Protótipo Inicial: <http://dainf.ct.utfpr.edu.br/~maziero/static/ceseg/wseg02/19.pdf>

Farmer, D. Venema, W.. Computer Forensics Analysis Class Handouts. Agosto, 1999. Acesso em 10 de novembro de 2011, disponível em: <http://www.trouble.org/forensics/class.html>

Kaspersky. (2010). Kaspersky Security Bulletin 2009. Malware Evolution 2009. Acesso em 12 de Outubro de 2011, disponível em <http://www.securelist.com/en/analysis/204792100/Kaspersky_Security_Bulletin_2009_Malware_Evolution_2009>.

Macedo, G. M. (2010). Investigação Forense Digital de Rootkits em Sistemas Unix. Acesso em 12 de Novembro de 2011, disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/26345/000757798.pdf?sequence=1>.

Mandia, K., & Jones, K. J. (2009). *Carbonite: A Linux Kernel Module to aid in RootKit detection*. Acesso em 10 de Setembro de 2011, disponível em <http://www.incident-response.org/Carbonite.htm>

McAfee (2009). Inc. Research Shows Global Recession Incrense. Risks to Intellectual Property. Acesso em 2 de Outubro de 2011, disponível em <http://www.mcafee.com/us/about/press/corporate/2009/20090129_063500_j.html>

Michael G. Noblett et al. Recovering and Examining Computer Forensic Evidence. Acesso em 12 de novembro de 2011, disponível em: <http://www.fbi.gov/hq/lab/fsc/backissu/oct2000/computer.htm>.

Murilo, N., & Steding-Jessen, K. (2001). *Métodos para detecção local de rootkits e módulos de kernel maliciosos em sistema Unix*. Acesso em 13 de Setembro de 2011, disponível em <http://www.chkrootkit.org/papers/chkrootkit-ssi2001.pdf>

Penasio, C., & Marangon, S. L. (2005). *Perícia forense computacional em sistemas alterados por rootkits*. Acesso em 13 de Setembro de 2011, disponível em <http://www.lsi.usp.br/~penasio/trabalhos/PSI5007-3009850-5223770-1-V1.10.pdf>

RecGeus, P. L., Reis, M. A. (2002). Análise forense de intrusões em sistemas computacionais - Anais do I Seminário Nacional de Perícia em Crimes de Informática. Maceió, 2002.

Register (2009). Conficker seizes city's hospital network. Acesso em 15 de Outubro de 2011, disponível em <http://www.theregister.co.uk/2009/01/20/sheffield_conficker/>

Rodrigues, T. S., & Foltran, C. D. (2010). Análise de ferramentas forenses na investigação digital. *Revista de Engenharia e Tecnologia , 2*(3), 102-113.

Rootlit Hunter. (2009). Rootkit: protect your machine. Scanner rootkit. Acesso em 10 de Outubro de 2011, disponível em <http://www.rootkit.nl/projects/rootkit_hunter.html>

Shadowserver (2006). *What is Malware?* Acesso em 20 de Outubro de 2011, disponível em <http://www.shadowserver.org/wiki/pmwiki.php/Information/Malware>

Simpson, D. (2009). *Checkps: Linux rootkit detector*. Acesso em 14 de Setembro de 2011, disponível em <http://sourceforge.net/projects/checkps/>

Spitzner, L. (2000). *Know Your Enemy: The Tools and Methodologies of the Script Kiddie*. Acesso em 10 de Setembro de 2011, disponível em <http://www.linuxvoodoo.com/resources/security/enemy>

Zeppoo (2006). Antirootkit List. Acesso em 19 de Outubro de 2011, disponível em <http://www.antirootkit.com/software/Zeppoo.htm>