PERÍCIA FORENSE COMPUTACIONAL EM SISTEMAS ALTERADOS POR ROOTKITS

Claudio Penasio Junior - Número USP 5223770

Sílvio Luís Marangon - Número USP 3009850

RESUMO

Neste artigo, serão abordados os conceitos básicos sobre os *rootkits*, metodologias utilizadas por estas ferramentas e como estes podem comprometer os resultados de uma perícia forense computacional. Outros pontos abordados são quais incertezas que os *rootkits* podem gerar sobre os indícios digitais e ações específicas para perícia em sistemas alterados por *rootkits*.

ABSTRACT

In this paper, it's discussed the influence of rootkits on digital forensic investigation. Also, it's present the basic concepts about rootkit, as types, feature and detection tools.

Palavras chaves: segurança, *rootkit*, vulnerabilidade, perícia, forense.
Keywords: security, rootkit, vulnerability, digital forensic.

Introdução

Rootkit é um conjunto de softwares desenvolvidos para criar e manter um ambiente para o intruso em uma máquina comprometida, sua função é ocultar as atividades e permitir o acesso futuro para o intruso. Estes surgiram em meados da década de 1990, incorporando ferramentas de software já existentes, como os *limpadores de arquivos de Log*. O histórico da evolução dos *rootkits*, como identificado por (Chuvakin, 2003), é apresentado na seguinte lista, as datas correspondem a data identificação dos *rootkits*:

- 1989: Primeiro "limpador de Log" encontrado em um sistema *hacked*
- 1994: Encontrado o primeiro *rootkit* - SunOS kit.
- 1996: Primeiro *rootkit* para Linux.

- 1997: *LKM Trojans* propostos em "*Phrack*"
- 1998: *Patch* não LKM kernel proposto por Silvio Cesare.
- 1999: Adore LKM kit disponibilizado por TESO.
- 2000: T0mkit v8 libproc.
- 2001: KIS Trojan e SucKit.
- 2002: *Rootkits* com *Sniffer*.

Os *rootkits* disponíveis podem ser agrupados em três categorias:

- *Rootkits* binários: São compostos por um conjunto de programas que substituem programas do sistema, sendo estes versões alteradas destas ferramentas.
- *Rootkits* de *kernel*: Estes são implementados como módulos do núcleo do sistema operacional.
- *Rootkits* de biblioteca: São implementados como bibliotecas, sendo versões alteradas de bibliotecas do sistema e que substituíram as mesmas.

As funcionalidades dos *rootkits* se dividem nas seguintes categorias:

1. Manter o acesso.
2. Permitir atacar outros sistemas.
3. Destruir evidências.

Manter o acesso.

O primeiro grupo de funcionalidades, manter o acesso, é caracterizado pelos *backdoors*, tanto para manter o acesso local, quanto acesso remoto. Os métodos utilizados por este grupo de funcionalidades são:

- **Telnet ou Shell TCP.**

  O atacante se conecta ao sistema via telnet ou algum outro serviço do inetd, como rsh, ftp, etc. Estas ferramentas, diferentemente das originais não deixam registro nos arquivos de *log* do sistema, porém podem ser detectadas verificando-se as portas TCP abertas e a configuração do inetd.

- **SSH (Secure Shell).**

  O acesso é mantido através de um serviço de ssh, utilizando uma porta tcp alta. Permite comunicação encriptada e impede que sistemas IDS de rede detectem comandos suspeitos. Pode ser detectado verificando-se as portas tcp abertas, porém pode ser ocultado utilizando-se *rootkits* binários ou de *kernel*, que ocultem as conexões de rede utilizadas pelo ssh.

- **CGI Shell.**

  Este utiliza *scripts* CGI no servidor web para executar comandos, normalmente como usuário "httpd" ou "nobody", e visualizar os resultados no *webbrowser*. Este método não abre novas portas tcp, pois utiliza as conexões do servidor web. Este tipo de *shell* é utilizado pelo atacante como última opção de conexão, normalmente quando o *rootkit* já foi detectado e seus componentes principais removidos.

- **UDP listener.**

  Backdoors que utilizam UDP, baseam-se no fato de que é menos provável detectar uma porta UDP aberta. Este tipo de *backdoor* utiliza protocolo de comunicação próprio, podendo utilizar criptografia para não ser detectado por sistemas de IDS.

- **Shell/telnet reverso.**

  Um *backdoor* deste tipo abre uma conexão para a máquina do atacante, o que é melhor em situações como a presença de um *firewall*.

- **ICMP telnet.**

  ICMP telnet, como implementado pelo Loki tool, utiliza-se de mensagens ICMP para transportar comandos, utilizando a técnica de *tunelamento*. Este método possui algumas vantagens como: não ser detectado por varredura de portas, por ferramentas do sistema como o netstat e do fato que vários tipos de mensagens ICMP são normalmente permitidas pelo *firewall*. Porém esta técnica pode ser detectada pelos sistemas de IDS.

- **Tunelamento reverso de shell.**

  Quando conexões externas não são permitidas. Um *backdoor* deste tipo cria uma conexão para a máquina do atacante, utilizando a porta destinada a um serviço amplamente utilizado e que normalmente não possui restrições, como por exemplo, a porta 80 TCP utilizada pelo servidor Web, o que permitiria conexões através de proxy e firewall, sendo improvável que sistemas de IDS detecte este tipo de *backdoor*.

- **Backdoor ativado por pacote mágico.**

  Este tipo de *backdoor* é a combinação das técnicas de conexão direta e reversa, porém este tipo de backdoor só abrira uma conexão ou executara um comando quando receber um pacote contendo uma seqüência numérica especifica.

- **No-listener backdoor (baseado em sniffer).**

  Neste método o *backdoor* utiliza a técnica de *sniffing* para receber pacotes específicos de comandos do atacante, ou seja, monitora o trafego de rede, não necessitando abrir portas para comunicação. Para enviar a resposta para o atacante, o backdoor forja um pacote com endereço IP falso para dificultar o rastreamento do pacote, o que poderia ser feito através do endereço MAC, porém ficaria limitado ao mesmo seguimento de rede.

- **Cover channel backdoor.**

  Aproveitando-se do grande número de protocolos de rede e de aplicação existentes e do grande número de parâmetros utilizados por estes. Um atacante pode projeta um mecanismo de comunicação, que utilize estes protocolos já existente, para transportar de forma velada informações entre a máquina do atacante e a máquina com o backdoor. Como por exemplo, um servidor web que altere no momento da transmissão uma página web, acrescentando alguns bytes extras. Se bem projetado, estes mecanismos de comunicação podem ser impossíveis de serem detectados na pratica.

Acesso local pode ser mantido através de ferramentas do sistema alteradas, como *login*, ou de usuário especiais com acesso de root, como fazem alguns *rootktis* de *kernel*, mas que não estão presentes na lista de usuário. *Rootkits de kernel* também podem alterar a chamada do sistema *setuid()* em sistemas unix-like para alterar os privilégios de um usuário para administrador do sistema.

Permitir atacar outros sistemas.

As ferramentas de ataque dos *rootkits* também têm a função de ampliar o território do atacante. Estas ferramentas podem ser ferramentas para ataque local, remoto ou DoS (*Denial of Service*).

As ferramentas para acesso local são utilizadas para recuperar o acesso com administrador do sistema, caso este seja perdido. Os *rootkits* também possuem *sniffers* de senha para captura de senha em protocolos como: POP3, IMAP, telnet e ftp. Estes também procuram por mensagens que contenham palavras como: senha ou password. Como *sniffers* necessitam que as placas de rede operem em modo promíscuo, o comando para verificar o modo de operação é substituído

por uma versão alterada, que não indicam o modo correto de operação. A senha de usuário também pode ser capturada através da interrupção de chamadas do sistema como *read()* e *write()*, o que permitiria ao atacante obter o nome do usuário e senha em conexões SSH. Os *rootkits* também incluem várias ferramentas para quebra de senha para poderem assumir o maior número possíveis de contas de usuários legítimos.

Destruir as evidências.

A terceira e crucial funcionalidade dos *rootkits* é eliminar as evidências de um sistema recém atacado e impedir que novas evidências não sejam geradas. A remoção de evidências inclui a limpeza de vários arquivos de log do sistema, de aplicativos, histórico do *shell* e registros de auditoria, como contabilidade de processos. Muitos rootkits interrompem a execução ou substituem o syslog por uma versão alterada em sistemas Unix/Linux para impedir que registros de sua atividade sejam gerados. Geralmente a limpeza do sistema é feita automaticamente por *scripts* do *rootkit*. Rootkits de kernel podem alterar ou interceptar chamadas do sistema, o que conseqüentemente comprometeria todo o sistema, ocultando informações do administrador e dos sistemas de segurança. Normalmente os *rootkits* procuram ocultar:

- Seus arquivos.
- Seus processos.
- Conexões de rede, bem como a origem e destino das conexões.

Worms.

Worms são programas com a capacidade de se propagar pelas máquinas em uma rede de forma automática e podem ser utilizados por atacantes para ampliar seu território, automatizando as tarefas executadas pelo atacante, como a procura por máquinas vulneráveis, entre as funções que podem executar estão (Murilo, N. e Jessen, K. 2001):

- Varredura da rede a procura de sistemas e serviços vulneráveis.
- Correção da vulnerabilidade utilizada para evitar ataque de terceiros.
- Instalação de rootkit e backdoors.
- Envio de e-mail com informações do sistema para o atacante.

Componentes dos rootkits.

Os *rootkits* apresentam uma grande variedade de componentes, assim estes componentes serão agrupados segundo o tipo de rootkit.

Rootkits binários.

Os rootkits binários são compostos por uma grande variedade de programas, principalmente ferramentas do sistema utilizadas pelos administradores, que foram alteradas para ocultar as atividades do atacante. Para sistemas Unix (ou Unix-like) os principais programas que são freqüentemente alterados, segundo (CHROOTKIT), são:

amd	ftpd	netstat	su
asp	fusers	ntpd	syslogd
basename	gpm	passwd	tar
biff	grep	pidof	tcpd
chfn	hdparm	pop2d	telnetd
chsh	identd	pop3d	timed
cron	ifconfig	ps	top
date	inetd	pstree	traceroute
dirname	killall	rexed	w
du	login	rlogind	write
echo	ls	rpcinfo	wted
egrep	lsof	rshd	xinetd
env	mail	sendmail	z2
find	mingetty	slogin	
fingerd	named	sshd	

Este programas fornecem ao atacante as seguintes funcionalidades:

- Prover acesso remoto.

- Prover acesso local.
- Ocultar processos do atacante.
- Ocultar conexões de rede do atacante.
- Ocultar arquivos criados pelo *rootkit* ou pelo atacante.
- Ocultar as atividades do atacante.

Rootkits de Kernel.

Diferentemente dos *rootkits* de binários, os *rootkits* de kernel são implementados como um módulo do *kernel* do sistema operacional, chamado de Loadable Kernel Module (LKM) kit. Quando estes módulos são carregados, estes alteram as chamadas do sistema, como por exemplo, as chamadas do sistema para leitura e escrita de arquivos: open, read e write, assim o LKM *rootkit* também a capacidade de alterar as funcionalidades do *kernel* do sistema operacional. Após o sistema operacional ser alterado pelo módulo do *rootkit,* este se torna não confiável e conseqüentemente todos os compomentes do sistema que utilizem as chamadas do sistema operacional. Com alteração das chamadas do sistema, os programas do sistema irão fornecer ao administrador do sistema informações incorretas, pois estes já as recebem alteradas do sistema operacional. Também existe a possibilidade da tabela de chamadas do sistema em alguns sistemas Unix ser alterada através da imagem da memória, dispositivo /dev/mem, o que permitiria alterar o sistema mesmo que a carga de módulos esteja desabilitada.

Rootkits de biblioteca.

Os *rootkit* de biblioteca alteram bibliotecas padrão do sistema, como a libc e libproc, para obter resultados equivalentes as dos *rootkits* de *kernel,* sem a necessidade de se instalar um módulo no *kernel*.

Identificação dos rootkits.

Rootkits e worms identificados pelo programa chkrootkit (CHROOTKIT) em sistemas Linux, FreeBSD, OpenBSD, NetBSD, Solaris, HP-UX 11, Tru64 e BSDI:

1.lrk3,lrk4,lrk5,lrk6;

2.Solaris rootkit;

3.FreeBSD rootkit;

4.t0rn;

5.Ambient's Rootkit (ARK);

6.Ramen Worm;

7.rh[67]-shaper;

8.RSHA;

9.Romanian rootkit;

10.RK17;

11.Lion Worm;

12.Adore Worm;
13.LPD Worm;
14.kenny-rk;
15.Adore LKM;
16.ShitC Worm;
17.Omega Worm;
18.Wormkit Worm;
19.Maniac-RK;
20.dsc-rootkit;
21.Ducoci rootkit;
22.x.c Worm;
23.RST.b trojan;
24.duarawkz;
25.knark LKM;
26.Monkit;
27.Hidrootkit;
28.Bobkit;
29.Pizdakit;
30.t0rn v8.0;
31.Showtee;
32.Optickit;
33.T.R.K;
34.MithRa's Rootkit;
35.George;
36.SucKIT;
37.Scalper;
38.Slapper A, B, C, D;
39.OpenBSD rk v1;
40.Illogic rootkit;
41.SK rootkit.
42.sebek LKM;
43.Romanian rootkit;
44.LOC rootkit;
45.shv4 rootkit;
46.Aquatica rootkit;
47.ZK rootkit;
48.55808.A Worm;
49.TC2 Worm;
50.Volc rootkit;
51.Gold2 rootkit;
52.Anonoying rootkit;
53.Shkit rootkit;
54.AjaKit rootkit;
55.zaRwT rootkit;
56.Madalin rootkit;
57.Fu rootkit;
58.Kenga3 rootkit;
59.ESRK rootkit;

Detecção de Rootkits

A Detecção de Rootkits é uma tarefa que pode ser praticamente impossível caso o administrador da máquina não tenha instalado uma ferramenta que mantenha a integridade dos arquivos binários do sistema no momento da instalação desta máquina. As ferramentas que cumprem este papel são conhecidas como IDS – Intrusion Detection System, ou ainda no caso de sistemas Linux, LIDS – Linux Intrusion Detection System. Para a tarefa de manter a integridade dos binários do sistema, a ferramenta mais conhecida é o Tripwire [Tripwire].

O Tripwire

“O Tripwire é o IDS do Linux baseado na máquina mais conhecido. A Tripwire Inc., empresa dos desenvolvedores do Tripwire, recentemente divulgou o código-fonte do software para a versão Linux e o licenciou sob os termos da GNU General Public License”[MIT].O Tripwire é uma ferramenta de integridade de sistema do tipo plataforma cruzada. O Tripwire permite que você saiba se alguém está fazendo algo que você não queira com seu sistema. Ele cria um banco de dados seguro dos atributos dos seus arquivos e diretórios, que pode inclusive conter assinaturas digitais, este banco é normalmente armazenado em um disco somente para leitura.

O Tripwire permite a comparação dos arquivos e diretórios do disco com sua base de dados sabendo assim se foram alterados por alguém.

Normalmente o Tripwire é executado através de um agendador de tarefas, e seus resultados são enviados ao administrador do sistema. O arquivo de configuração do Tripwire permite quais arquivos e diretórios você quer rastrear, e a que nível de detalhe.

Análise manual do sistema

Uma forma para detecção manual dos vários tipos de Rootkits será discutida agora, é importante destacar que uma detecção manual pode ser mais confiável no caso de administradores extremamente experientes, uma vez que é necessário vasto conhecimento para este tipo de perícia computacional.

Arquivos de configuração dos Rootkits

Muitos *Rootkits* possuem arquivos de configuração para tornar sua utilização mais fácil ao atacante do sistema, nestes arquivos de configuração o atacante insere as características desejadas de alteração do sistema pelo *Rootkit*.

“É possível localizar a presença de um *rootkit* procurando por ocorrências de nomes de arquivos de configuração no interior de comandos alterados por um *rootkit*. As ferramentas strings, od e hexdump do Unix são normalmente usadas para esse tipo de análise” [MURILO, N. E JESSEN].

Comparação dos valores hash dos arquivos do sistema

Desde que se tenha uma cópia confiável dos binários do sistema, uma boa forma da verificação de sua integridade é a comparação dos “valores *hash* como MD5 e SHA-1 podem ser úteis na determinação de quais comandos do sistema foram alterados por *rootkits*”[MURILO, N. E JESSEN].

Chamadas de sistema

A verificação das chamadas de sistema ou system calls pode ser outra forma de detecção de *rootkit*. Esta análise se dá na verificação de chamadas de sistemas realizadas por binários

suspeitos a arquivos de configuração de *rootkits*. No Linux o comando utilizado para isso é o "strace".

Evidencias de modificação

Através da análise de informações de criação, utilização e alteração de arquivos é possível obter evidências de uma invasão do sistema, alteração do arquivo por *rootkit* ou *worm*. É importante salientar que alguns *rootkits* já conseguem alterar estas datas tornando esta análise comprometida.

No linux estas informações podem ser obtidas através do comando:

ls -l –time={atime,access,use,ctime,status} ARQUIVO

Portas abertas

Analisar quais portas estão abertas no sistema é uma boa forma de saber se há alguma porta não desejada aberta. O problema é que o netstat e o nmap são binários conhecidos dos *rootkits* e na maioria dos casos estes programas estão comprometidos. No caso de *rootkits* do tipo LKM estas portas são ocultadas diretamente no kernel. Para uma varredura eficiente de portas o ideal é realizá-la a partir de outra máquina.

Análise da tabela de símbolos do kernel

Os *rootkits* de LKM freqüentemente deixam marcas na tabela de símbolos do *kernel*, na maioria dos sistemas esta tabela está em (/dev/ksyms).

Outra alternativa na detecção dos Rootkits é a utilização de ferramentas que tentam detectar ações típicas dos *rootkit* tradicionais. A ferramenta mais conhecida para sistemas Linux deste tipo é o ***Chkrootkit*** [Chkrootkit], existem outras deste tipo e entre elas se destaca o ***Rootkit Hunter*** [Rootkit Hunter].

O Chkrootkit

O Chkrootkit é uma ferramenta que tem como função à detecção de sinais deixados por rootkits, que funciona em várias distribuições Linux.

O Chkrootkit é composto dos seguintes programas:

a)chkrootkit: é um shell script que checa os arquivos binários por possíveis modificações realizadas por rootkits.

b)ifpromisc.c: Verifica se a interface de rede está em modo promíscuo.

c)chklastlog.c: Verifica se registros de logs foram apagados.

d)chkwtmp.c: Verifica por alterações no arquivo wtmp.

e)check_wtmpx.c: Verifica por alterações no arquivo wtmpx. (Solaris)

f)chkproc.c: Verifica por sinais de *trojans* LKM.

g)chkdirs.c: Verifica por sinais de *trojans* LKM.

h)chkutmp.c: Verifica por alterações no arquivo utmp.

O Rootkit Hunter

Este software realiza a busca por assinaturas de diversos tipos de *rootkits* em sistemas Linux, O Rootkit Hunter está sob a licença GPL. No site do Rootkit Hunter [8], os desenvolvedores garantem que utilizando esta ferramenta, os sistema fica livre de 99,9% dos efeitos dos *rootkits*. Entre os testes feitos por ele estão:

- Comparação de hash MD5.

- Análise de arquivos freqüentemente utilizados por rootkits.

- Permissões erradas para arquivos binários.

- Análise de "*Strings*" suspeitas nos módulos LKM e KLD.

- Análise de arquivos ocultos.

- Varredura em arquivos textos e binários.

No site do Rootkit Hunter [8] existe uma vasta lista de Rootkits que ele detecta.

Perícia forense

Carrier e Spafford propuseram em (Carrier e Spafford, 2003) o modelo de processo de investigação digital integrado, o qual será utilizado neste artigo como guia para as discussões nas próximas seções.

O processo de investigação digital integrado é um modelo de processo de investigação que se baseia na idéia de que o computador é em si uma cena do crime, chamada de cena digital do crime, na qual aplica as técnicas de investigação de crimes. Este modelo de processo possui 17 fases organizadas em cinco grupos:

- **Prontidão (Preparação).**

  Esta é a fase de preparação e é composta por duas fases:

  - **Operacional**: Deve prover treinamento e equipamentos apropriados para a realização da investigação.

  - **Infraestrutura**: Deve garantir que os dados necessários para investigação estejam disponíveis.

- **Deployment.**

  Esta fase deve prover os mecanismos para detecção e confirmação de um incidente, sendo composta pelas fases:

  - **Detecção e Notificação**: Onde o incidente é detectado e as pessoas responsáveis são notificadas.

  - **Confirmação e Autorização**: Nesta fase deve-se obter a autorização necessária para a investigação completa do incidente.

- **Investigação da cena física do crime.**

  Nesta fase são coletadas e analisadas as evidências físicas e reconstruída as ações que ocorreram durante o incidente, sendo esta composta pelas fases:

  - **Preservação**: Nesta fase deve-se tomar as precauções necessárias para preservação da cena do crime, como: bloqueio das entradas e saídas, detenção de suspeitos e identificação de testemunhas.

  - **Inspeção**: Identificar as evidências mais obvias e frágeis, e desenvolver uma teoria inicial sobre o crime.

  - **Documentação**: Deve registrar o máximo possível de informações sobre a cena do crime, garantindo que os detalhes mais importantes sejam preservados.

  - **Procura e Coleta**: Nesta fase é realizada uma inspeção mais detalhada na cena do crime para procura de novas evidências.

  - **Reconstrução**: Envolve a organização, analise dos das evidências físicas e digitais coletadas, e o desenvolvimento de uma teoria sobre o incidente.

- **Investigação da cena digital do crime.**

Nesta fase são identificados os eventos eletrônicos que ocorreram nos sistemas e fornece-los a investigação física da cena do crime. Esta é composta pelas fases:

- **Preservação**: Em uma cena digital de crime, nesta fase deve ser tomada as providencias para preservar as evidências digitais através de medidas apropriadas, como: isolar o sistema da rede, coletar dados voláteis e identificar processos suspeitos.

- **Inspeção**: A inspeção de uma cena digital de crime geralmente ocorre em laboratório e utilizando-se copias "imagem" dos discos.

- **Documentação**: A documentação das evidências digitais deve ser feita de forma adequada e detalhadamente, e também se deve calcular o valor de hash, como MD5 e SHA-1, para garantir a autenticidade das evidências.

- **Procura e Coleta**: Neta fase, os sistemas devem ser analisados detalhadamente, utilizando-se as evidências coletadas na fase de inspeção como ponto de partida.

- **Reconstrução**: Nesta fase, são analisadas as evidências digitais, estas devem ser classificas, organizadas, deve-se identificar o seu real significado e como estas podem sustentar ou não uma teoria.

- **Apresentação**: Envolve a apresentação das evidências digitais para o grupo de investigação da cena física do crime e não inclui informações sobre outras cenas digitais de crime. A consolidação das informações obtidas em várias cenas digitais é feita pelo grupo de investigação física.

- **Revisão.**

Envolve a revisão de todas as fases de investigação e as identificações das fases devem ser melhor desenvolvidas.

Outro fator que deve ser considerado em uma perícia forense é a confiabilidade das evidencias digitais encontradas. Em (Casey, 2002), o autor apresenta uma discussão sobre os erros, incertezas e perdas de evidencias digitais, propondo uma escala de para classificação das evidencias, veja tabela 1. As evidencias digitais podem conter incertezas temporal e de origem, que podem surgir devido à falha ou má configuração do sistema ou porque foram forjadas.

Tabela 1: Níveis de incerteza de evidencias digitais

Nível de certeza	***Descrição/Indicadores***	***Qualificação***
C0	Evidencia contradiz fatos conhecidos	Incorreto
C1	Evidencia é altamente questionável	Incerteza alta

Nível de certeza	*Descrição/Indicadores*	*Qualificação*
C2	Somente uma fonte de evidencias que não é protegida contra alteração	Incerto
C3	Fonte de evidencia difícil de alterar, porém existe alguma inconsistência.	Possível
C4	Fonte da evidencia é protegida contra alteração ou existem várias fontes de evidencias não protegidas	Provável
C5	Várias fontes de evidencias protegidas contra alteração, porém existe uma certa incerteza.	Quase certo
C6	Evidencia protegida contra alteração e inquestionável	Certo

Em (Flush, 2001), é discutida a necessidade de se responder, em uma investigação de um crime digital, as seis perguntas padrão que devem ser feitas em uma investigação, ou seja: Quem, O que, Onde, Quando, Por que e Como?

Influência do Rootkits na Perícia forense

Nesta seção, será discutido como a presença de um *rootkit* em um sistema pode influenciar nos resultados de uma perícia. O modelo de investigação apresentado na seção anterior será utilizado como guia para as discussões subseqüentes para facilitar a compreensão. Não é intenção deste artigo apresentar um método especifico para investigação judicial ou corporativa, nem focado em um *rootkit* ou sistema operacional especifico.

- **Prontidão (Preparação).**

  Durante a fase de preparação, atenção especial deve ser dada ao estudo do *rootkits* e nas suas sub-fases:

  - **Operacional**: Deve prover treinamento especifico sobre os *rootkits,* equipamentos apropriados para a realização da investigação e ferramentas de software específicas para a detecção de *rootkits* devem estar disponíveis. Devido ao fato de existirem vários tipos de *rootkits*, binários, de biblioteca e de *kernel* e destes operarem das formas mais variáveis possíveis; esta fase deve prover treinamento amplo, que aborde aspectos básicos do funcionamento do sistema operacional e de sua construção, como por exemplo: construção de módulos e bibliotecas, ligação dinâmica de bibliotecas, tratamento de sinais e interrupções,

protocolos de comunicação de rede e treinamento especifico das ferramentas de detecção de *rootkits*.

•**Infraestrutura**: Como nesta fase deve-se garantir que os dados necessários para investigação estejam disponíveis. Medidas preventivas devem ser tomadas para garantir que as informações estejam disponíveis, como: gerar o valor hash para os arquivos do sistema, com ferramentas como o *Tripwire*, e monitorar o trafego de rede com ferramentas de detecção de intrusão como o Snort. Para mais detalhes sobre a detecção de *rootkits,* veja a seção *Detecção de Rootkits*.

•**Deployment.**

Nesta fase, é analisada as influencias do *rootkit*, sobre os mecanismos para detecção e confirmação de um incidente, sendo composta pelas fases:

•**Detecção e Notificação**: A detecção do incidente, ou seja, a detecção do *rootkit* pode ser feita segundo o apresentado na seção *Detecção de rootkits*, porém deve-se salientar que a detecção de um rootkit de kernel pode não ser possível em um sistema alterado, segundo (Chuvakin, 2003), enquanto este estiver em operação, pois as ferramentas de detecção de *rootkit,* podem ser "enganadas" por módulos acrescentados ao kernel. A detecção através de meios externos também pode não ser eficiente, pois a comunicação entre a máquina atacada pode ser ocultada, veja a seção *Cover channel backdoor*, não permitindo sua detecção por ferramentas de IDS. O momento entre o inicio do ataque e a finalização da instalação do *rootkit*, antes deste "limpar" seus vestígios, é o intervalo de tempo no qual o atacante pode ser mais facilmente detectado, localmente ou através de sistema de IDS, sendo este o momento de maior exposição do atacante.

•**Confirmação e Autorização**: Nesta fase deve-se obter a autorização necessária para a investigação completa do incidente, não se diferenciando de outros casos.

•**Investigação da cena física do crime.**

Nesta fase são coletadas e analisadas as evidências físicas e reconstruída as ações que ocorreram durante o incidente, sendo suas fases discutidas aqui, devido a influencia destas em fases futuras, principalmente na investigação da cena digital do crime. Esta composta pelas fases:

•**Preservação**: Nesta fase deve-se tomar as precauções necessárias para preservação da cena do crime, no caso especifico de sistemas com *rootkits,* deve-se tomar medidas para bloquear as ações do atacante e do *rootkit* o mais rapidamente possível, pois quando detectados estes podem tomar medidas de retaliação ou eliminar seus rastros. Ações que devem ser domadas são: desconectar

o equipamento de rede para impedir acesso do atacante ou desligamento do equipamento, caso dados voláteis já tenham sito obtidos. As medidas de preservação da cena do crime devem incluir a averiguação dos outros equipamentos, pois é comum entre os atacantes o conceito de ampliação de território, assim outras máquinas podem estar comprometidas e devem ser analisadas para detecção e preservação.

- **Inspeção**: A identificação das evidências mais obvias e frágeis em um caso de ataque com *rootkit* procede conforme outros casos. Tomando-se os devidos cuidados na coleta de dados voláteis, pois como os rootkit alteram o sistema, as ferramentas do sistema, bem como bibliotecas e o próprio sistema operacional não ser confiáveis para a realização esta tarefa e ferramentas externas e confiáveis devem ser utilizadas.

- **Documentação**: Procede normalmente como discutido a seção anterior.

- **Procura e Coleta**: Procede normalmente como discutido a seção anterior.

- **Reconstrução**: Na reconstrução, o responsável pela reconstrução da cena física do crime deve interagir com os peritos de cada cena digital do crime, segundo o ciclo indicado em (Carrier e Spafford, 2003, p. 8-9), requisitando, quando um *rootkit* é detectado, que outros computadores sejam inspecionados, devido à tática de expansão de território, e que as evidencias sejam analisadas adequadamente.

- **Investigação da cena digital do crime.**

Nesta fase são identificados os eventos eletrônicos que ocorreram nos sistemas e fornece-los a investigação física da cena do crime. Nas seções seguintes discutiremos como os rootkits podem influenciar uma perícia digital.

- **Preservação**: Em uma cena digital de crime, nesta fase devem ser tomadas as providências para preservar as evidências digitais através de medidas apropriadas, como: isolar o sistema da rede, coletar dados voláteis e identificar processos suspeitos. Para o caso de um *rootkit* deve-se tomar medidas especificas. como: desativar *backdoors*, processos que enviem informações sobre o sistema para o atacante, *sniffers* para captura de senha ou outro processo suspeito. A coleta de dados voláteis deve ser feita preferencialmente com ferramentas especificas, não devendo-se utilizar as ferramentas do sistema, pois estas ou as bibliotecas utilizadas por estas podem ter sido alteradas por *rootkits* binários ou de biblioteca. No caso do sistema ter sido alterado com um *rootkit* de *kernel*, pode não ser possível coletar os dados voláteis ou estes podem ser totalmente não confiáveis, pois o próprio kernel do sistema estará ocultando ou alterando os dados.

•**Inspeção**: Na inspeção de uma cena digital de crime, quando um rootkit é utilizado pelo atacante, o administrador de sistemas ou o investigador especializado deve procurar identificar qual o *rootkit* utilizado. Pois como as técnicas utilizadas por estes para ocultar evidencias muda conforme o *rootkit* utilizado, a identificação de qual rootkit foi utilizado pode auxiliar a definir os procedimentos adequados a serem executados durante as fases de inspeção, procura e coleta.

•**Documentação**: A documentação das evidências digitais, quando detectado a existência de um *rootkit* no sistema, deve conter informações sobre o *rootkit* especifico e de seu possível impacto sobre a evidencia para ser permitir a correta análise da evidencia em fases futuras.

•**Procura e Coleta**: Neta fase, os sistemas devem ser analisados detalhadamente, utilizando-se as evidências coletadas na fase de inspeção como ponto de partida. A identificação do *rootkit*, feita na fase de inspeção, permitirá ao perito identificar quais evidências digitais coletadas no sistema são confiáveis e quais não são, ou seja, após identificar o *rootkit* utilizado, o perito pode saber quais informações este oculta, altera, forja ou remove, assim podendo estimar qual o real valor de uma evidencia digital. A identificação do *rootkit* também permite identificar possíveis evidências digitais deixadas pelo atacante devido às falhas do próprio *rootkit* e da confiança que o atacante deposita neste. O conhecimento prévio do *rootkit* utilizado também pode ser útil na definição de onde procurar por evidencias, por exemplo: Se um *rootkit* troca o comando *ps* por um novo, que oculta os processo do atacante, o perito, então, saberá que para obter a informação desejada deverá utilizar um programa externo. Como os rootkits executam algumas tarefas automaticamente e outras não, a procura por evidencias pode ser mais fértil quando focada nas tarefas que devem ser executadas manualmente pelo atacante. O conhecimento detalhado do *rootkit* também pode ser útil na análise dos dados de um sistema de IDS ou os dados de um *sniffer,* utilizado para monitorar a comunicação da máquina atacada com a máquina do atacante, pois como visto na seção *Cover channel backdoor*, a comunicação entre as máquinas pode ser velada e informações “roubadas” podem estar sendo enviadas para outra máquina sem serem detectadas.

- **Reconstrução**: A análise das as evidências digitais, sua classificação, seu real significado e como estas podem sustentar ou não uma teoria, são afetados pelos *rootkits* em vários aspectos: a classificação de uma evidência pode ser rebaixada, ou seja, esta pode ser considerada incorreta ou incerta, e o seu significado pode ser totalmente incorreto ou enganoso, caso esta evidência possa ser forjada pelo *rootkit*. O perito deve estar atento em quais evidências considerar, pois a escolha de uma evidência passível de ser alterada ou que certamente é alterada pelo *rootkit* pode invalidar todo o trabalho de perícia.

- **Apresentação**: A apresentação das evidências digitais para o grupo de investigação da cena física do crime deve ressaltar a existência de um *rootkit* e seu impacto sobre as evidencias apresentadas para permitir ao grupo de investigação da cena física a consolidação e a definição de ciclos de interação com o grupo ou outros grupos de investigação digital. Como exemplo: A identificação de um *rootkit* específico em um servidor Web permitiria que o responsável pela análise dos registros da comunicação em rede analisando os métodos de comunicação deste identificar um provável destinatário das mensagens de um *backdoor*.

- **Revisão.**

  A revisão de todas as fases de investigação procede normalmente, sem diferenças significativas de outros casos.

Com relação à confiabilidade das evidências digitais, evidências que são automaticamente removidas ou alteradas pelo *rootkit* podem ser consideradas incorretas, caso não exista nenhum mecanismo de proteção. Mesmo registros que não são alteradas automaticamente podem ter sua qualificação rebaixada, pois o atacante pode altera-la manualmente. Como os *rootkits* geralmente apagam os registros em *log*, desativam sistemas de *log* ou instalam sistemas de *log trojam*, a perda de evidencias pode ser considerada elevada, dependendo apenas de quanto tempo o sistema foi atacado e das atividades do atacante. Devido às perdas e alteração de registros do sistema, a criação e análise de uma "linha de tempo" de acesso ao sistema de arquivos pode ser totalmente comprometida.

Como discutido em (Flush, 2001), sobre a necessidade de se responder, em uma investigação de um crime digital, as seis perguntas padrão em uma investigação, ou seja: Quem, O que, Onde, Quando, Por que e Como?

No caso de um sistema alterado com rootkit como estas perguntas poderiam ser respondidas:

Quem? - Como saber quem cometeu um crime, se os rootkits podem alterar as datas de acesso e modificação de arquivos, o usuário que os alterou, ocultar processos e a comunicação externa. O atacante também pode utilizar alguma técnica de “triturar os dados” para limpar a sua própria máquina, procurando eliminar evidências que o associem a um determinado ato.

O que? - Como saber o que foi roubado, alterado ou destruído, se os rootkits apagam seus vestígios? Não alterando o tempo de acesso ou alteração de um arquivo, como os programas comuns o fazem.

Onde? - Se o ataque é remoto, a cena do crime digital deixa de ser apenas local da máquina atacada e passa a incluir a máquina do atacante e outros equipamentos utilizados.

Quando? - Como saber quando ocorreu um incidente, se os rootkits alteram as datas de acesso a arquivos e ocultam sua comunicação?

Por que? Talvez não seja problema do perito.

Como? A identificação do *rootkit* pode esclarecer algo sobre como alguma informação poderia ter sido roubada, o que poder ser um ponto de partida na investigação.

Conclusão

Após o que foi discutido anteriormente, podemos verificar que os *rootkits* podem afetar as várias fases de um processo de investigação. Como os *rootkits* podem provocar a perda ou adulteração de evidencias digitais, diminuindo o número de evidencias e a confiabilidade das evidencias encontradas, os peritos devem estar atentos à existência destes e conhecer os métodos utilizados por estes para evitar ou minimizar o impacto causado, assim podendo qualificar melhor as evidências encontradas e saber onde procurar por outras evidências. O conhecimento dos rootkits fornece ao perito o conhecimento dos pontos fracos do *rootkit*, o que pode permitir ao perito conduzir melhor seu trabalho, na procura por evidências válidas.

Referências Bibliográficas

1.Chuvakin, A. An overview of unix rootkits, 2003.

2.chrootkit, (http://chkrootkit.org), acessado em 12/08/2005.

3.Murilo, N. e Jessen, K 2001, Método para detecção local de rootkits e módulos de kernel maliciosos em sistemas unix.

4.Carrier, B. Spafford, E. H. 2003, Getting Physical with the Digital Investigation Process, International Journal fo Digital Evidence, Fall 2003,Volume 2, Issue2.

5.Flush, K. J. 2001, Computer Forensic Case Study: espionage, Part 1 Just Finding the File is Not Enough, Information System Security, 2001.

6.Casey, E. 2002, Error, Uncertainty, and Loss in Digital Evidence, International Journal of Digital Evidence, volume 1, issue 2, 2002.

7.ITSF – Detecção local de Rootkits e LKMs maliciosos, (http://www.istf.com.br/vb/archive/t-4550.html). acessado em 17/08/2005.

8.Rootkit Hunter, (http://www.rootkit.nl/projects/rootkit_hunter.html), acessado em 17/08/2005.

9.Tripwire, (http://sourceforge.net/projects/tripwire/), acessado em 17/08/2005

10.MIT Massachusetts Institute of Technology, (http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-sg-pt_br-4/s1-ids-host.html) acessado em 23/08/2005