GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Advanced Incident Response, Threat Hunting, and Digital Forensics (Forensics
at http://www.giac.org/registration/gcfa

SANS GCFA Practical Submission

Version 1.3 (June 03, 2003)

By

John Banghart

© SANS Institute 2003, Author retains full rights.

Abstract:

In this paper, I will demonstrate the proper procedures and actions that are necessary to correctly analyze collected evidence so that it is admissible in a court of law. I will provide concrete examples using real world date where possible. Furthermore, I will demonstrate knowledge of legal issues surrounding forensic analysis be resolving simulated real-world events.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

TEXTUAL CONVENTIONS USED IN THIS DOCUMENT 3
ANALYSIS OF AN UNKNOWN BINARY 4
SYNOPSIS: 4
PREPARATION: 4
BINARY DETAILS: 4
Step 1: Analysis of the zip archive. *4*
Step 2: Unpacking the archive. *6*
Step 3: Verifying the integrity of the file received *8*
Step 4: Analyzing the binary file atd *9*
SOURCE ANALYSIS 16
LEGAL IMPLICATIONS 19
INTERVIEW QUESTIONS 20
ANALYSIS OF AN UNKNOWN SYSTEM 23
SYNOPSIS OF CASE FACTS 23
SYSTEM DESCRIPTION 23
HARDWARE 24
IMAGE MEDIA 25
MEDIA ANALYSIS OF SYSTEM 27
ANALYSIS OF FILE FROM /ETC 33
SUID, SGID 35
CHKROOTKIT 36
HISTORY FILES 36
STARTUP SCRIPTS 37
STRINGS 38
HIDDEN DIRECTORIES 40
USERS HOME DIRECTORY 43
DELETED FILES 45
TIMELINE 47
CONCLUSIONS AND RECOMMENDATIONS 50
LEGAL ISSUES OF INCIDENT HANDLING 53
ADDITIONAL INFORMATION AND CONCLUSIONS 55
APPENDIX A: STRACE OUTPUT OF ATD AND LOKID 57
APPENDIX B: LIST OF SUID, SGID FILES. 60
APPENDIX D: RELATED VIRGINIA STATUTES 63
REFERENCES AND CITED SOURCES 71

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Textual conventions used in this document

In order to present the information contained within this document in an easy to read manner, I have used several conventions to differentiate between different types of data.

With the exception of any direct screen captures, all input and output from the command line, or any output gathered from a web browser, is encapsulated inside a box, and uses the Courier New font. Example:

```
[john@localhost dev]$ ls -l z*
crw-rw-rw-    1 root     root       1,   5 Aug 30  2002 zero
crw-rw----    1 root     disk      27,  16 Aug 30  2002 zqft0
crw-rw----    1 root     disk      27,  17 Aug 30  2002 zqft1
crw-rw----    1 root     disk      27,  18 Aug 30  2002 zqft2
crw-rw----    1 root     disk      27,  19 Aug 30  2002 zqft3
```

Within the normal text, any programs and scripts that are referenced in relation to the source or analysis system are highlighted with the color green, and underlined. For example, instead of find, you will see find, which is a popular Unix command.

Directories and non-binary files mentioned in the normal text will be highlighted with the color purple. For example: /home/john.

URL's will appear in the color blue: http://www.sans.org.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Analysis of an Unknown Binary

Synopsis:

A binary program, retrieved from a compromised system, will be analyzed using forensic tools and procedures. Both the tools and procedures will be well documented in order to provide the best possible information concerning the origin, function, and effects of this binary program.

Preparation:

In order to insure that the analysis process did not adversely affect an important system or network, a clean PC was setup with Red Hat Linux version 7.3.

In order to accommodate possible network tests, the analysis system was connected via a 10baseT hub to a laptop running Windows XP. This laptop was initially configured with a firewall set to drop any inbound packets.

Neither system was connected to any other network by any means.

To insure integrity of the analysis tools, they were run off of a trusted CD.

Binary details:

The binary in question was provided to me on a floppy disk in a zip archive named binary_v1.2.zip. It was not initially clear if this was the original state of the file, or if it had been prepared in this fashion by the system administration personnel.

Step 1: Analysis of the zip archive.

I first needed to determine if binary_v1.2.zip was in fact a zip file. I did this by using the file command, which was located on my CD.

```
[root@localhost work]# file binary_v1.2.zip
binary_v1.2.zip: Zip archive data, at least v2.0 to extract
```

The Unix command file returns information about the type of file, as can be seen in the above example. Based on this information, I felt confident that binary_v1.2.zip was in fact a zip archive. This allowed me to choose the appropriate tools for the next step.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

First, I wanted to know what was inside the archive without actually having to open it. I accomplished this by using the zipinfo command as shown:

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

I now have information on what is inside, but what does this actually mean?

The first line simply states the name of the archive, its size in bytes, and how many files it contains.

The first file is called atd.md5. Assuming the file name/extension is accurate, this file should give me a MD5 sum of the second file atd. MD5 sums are one-way hashes of files use to insure integrity[1]. Although I couldn't be sure, my initial guess was that atd.md5 was added to the archive by the system administrator.

The first column of the display shows us the standard Unix file permissions. In this case, both files are set to "read and write" permissions for all users. This struck me as odd since in order to executed on a system, the binary would require the "execute" or "x" permission bit set. The possibility exists that permissions where changed by the system administrator in order to insure the file was not mistakenly executed. Further investigation would be necessary.

The second column shows us the version number of Zip that was used to create the archive, in this case 2.0.

The third column indicates the type of system the archive was created on. "fat" is the standard file system type of MS-DOS based systems which include many versions of Windows. Since the system administrator told me that the compromised system was a Linux system, I assumed that the archive was created on a different system after the binary had been removed from the compromised host.

The fourth column shows the uncompressed size of the file in bytes.

The fifth column indicates whether or not the zipinfo program believes the compressed file is text ("t") or binary ("b"). The results displayed meet expectations for the two files in the archive.

The sixth column indicates the compression method used in creating the archive.

Columns 7 and 8 show the date and time that the file(s) where last accessed.

[1] RSA Laboratories: http://www.rsasecurity.com/rsalabs/faq/3-6-6.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Finally, the last column shows the name of the file.

The last line in the output shows a summary of the archive.

Summary and Conclusions:

In this step I received a zip archive that appears to contain a binary file, and an MD5 that will probably contain the one-way hash that I can use to verify that the binary file I have is the same as the one that was taken from the compromised host.

I then examined the zip archive using the following tools:

- file
- zipinfo

The information gathered by these tools shows me that the zip archive was likely created on a DOS based system. Since I have been led to believe that the compromised system is a Linux host, my conclusion is that the archive was created on a different machine then the one which was compromised. The practical outcome of this is that the file modification times listed in the zipinfo output are probably those of when the files were copied/moved to the archiving machine, not those of when the files were on the compromised host.

Step 2: Unpacking the archive.

After completing the analysis of the zip archive, I am now ready to unpack the archive and begin analyzing the contents. I unzipped the archive as follows:

```
[root@localhost work]# unzip -X binary_v1.2.zip
Archive:  binary_v1.2.zip
  inflating: atd.md5
  inflating: atd
```

According the documentation for unzip on Linux, it should not change the ownership, permissions or modification times on files. To insure that this is the case, I checked the modification time using debugfs and compared it against the information I gather from zipinfo in step one.

```
debugfs:  stat atd
Inode: 309165   Type: regular    Mode:  0666   Flags: 0x0   Generation:
499235772
User:     0   Group:     0   Size: 15348
File ACL: 0    Directory ACL: 0
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
Links: 1   Blockcount: 32
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x3e650e26 -- Tue Mar  4 15:35:50 2003
atime: 0x3d653432 -- Thu Aug 22 14:57:54 2002
mtime: 0x3d653432 -- Thu Aug 22 14:57:54 2002
BLOCKS:
(0-3):634347-634350
TOTAL: 4
```

As we can see above, debugfs reports the "mtime" (modification time) as being "Aug 22 14:57:54 2002". This matches zipinfo's output of "22-Aug-02 14:57"

Debugfs also reports the user as being 0(root) and the permission as being 0666(rw-rw-rw), both of which match zipinfo.

So I am now confident that unzip hasn't changed the modification time, permissions, or owner of the files in the archive.

However, debugfs has given me an additional piece of information: the creation time. Since the creation time matches the time that I unzipped the archive, it's clear that the original creation time of the atd file has been lost.

I also notice that the "atime" and "mtime" values are very close, which supports my earlier conclusion that the MAC times for atd were all overwritten when the system administrator transferred the file from the compromised host to the archiving system.

I have also discovered that the User ID (UID) and Group ID (GID) of the file are both 0. UID provides a numerical representation of a user on the system. GID represents the primary group that a user belongs to. Groups are collections of users that have certain privileges on the system that allow them to perform whatever their assigned duties are. For example, members of a group called "webmasters", with a GID of 500, might have specific privileges to manage the web server and associated files, but nothing else.

On typical Unix systems, the 0 UID/GID is reserved for the "root" user and group. "Root" is the equivalent of "Administrator" on Windows based systems and that user has virtually unlimited rights and access to a system. This is important because it strongly suggests that the cracker who installed this file had "root" access to the machine, which is a serious breach of security.

Step 2 Summary and Conclusions:

I was able to confirm that unzip does not alter any of the file information when it unpacks an archive. It maintained the ownership, permissions, and modification time of the file.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Unfortunately, it became obvious that the MAC times associated with the binary in question were all modified when the system administrator transferred it to create the archive.

I was able to determine all this information by comparing date produced by zipinfo and debugfs.

Step 3: Verifying the integrity of the file received.

Now that I have the files out of the archive and into my working directory, I can begin to analyze them directly.

However, it is important to verify that the file I am examining is the same file that was retrieved from the compromised host.

To do this, I create my own MD5 sum as shown in this screenshot:

I now examine the value provided by the “atd.md5” file.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

A careful examination of the two values shows that they are equivalent.

Unfortunately, I have no way of verifying that the zip archive was not intercepted or altered in any way before reaching me. Therefore, the entire integrity of the archive and its contents is in question. Since I had no way of investigating this, I proceeded under the assumption that it was not modified. Under normal circumstances, data integrity would need to be better documented by on site staff.

Step 3 Summary and Conclusions:

Using the Unix program md5sum, and the contents of the file atd.md5 I was able to verify that the binary file atd I have in my possession is the same one placed into the archive by the system administrator.

Verifying data integrity is exceptionally important, particularly when prosecution is a possible avenue. If there is any chance that evidence has been tampered with in any way, it could be inadmissible in a court of law.

Step 4: Analyzing the binary file atd

Having successfully extracted and verified atd, I can now begin to examine it directly.

atd is the name of a normal Linux daemon which is designed to run jobs queued for later execution. Therefore, my first guess was that this was a modified

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

version of atd, designed to replace the original's function, but with other malicious capabilities added; what is commonly referred to as a Trojan Horse.

When a program is executed on a system, it performs various functions that have been written by the programmer in order to accomplish its function. For example, the programmer may need to determine what the current date is, so he or she will write a function to make that determination. Normally, the activities of these functions are hidden from the user because they have no bearing on the results. However, in cases where a program does not functionally normally, a debugger is used to "watch" what the program is doing in order to determine where the problem is occurring. It accomplishes this by useful special debugging "symbols" that are inserted into the program.

Staying with our date example, the function may not be able to retrieve the date, or it may always get the date wrong. In this case, the debugger would allow the developer to find the problem and fix it relatively quickly.

Normally, using a debugger to watch the execution of a program would be a great place to start because I can see exactly what it is trying to do. So I'll try loading it into the standard Unix gdb debugger.

```
[root@localhost work]# gdb
GNU gdb Red Hat Linux (5.2.1-4)
Copyright 2002 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and
you are
welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for
details.
This GDB was configured as "i386-redhat-linux".
(gdb) file atd
Reading symbols from atd...(no debugging symbols found)...done.
(gdb)
```

Unfortunately, this output shows me that this person did not leave the debugging symbols in when compiling the program. Though useful, debugging information is often removed from a program after it has been tested in order to make the program smaller and faster.

Unable to gather any information that way, I'll instead use the Unix command strings, which will scan through a binary file and print out any combination of ASCII characters that it finds.

```
[root@localhost work]# strings -a atd > strings.out
```

I've taken the output from the strings command and redirected it to a file so that I can more easily analyze it.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The entire strings output can be found in Appendix A. Here, I have listed some key elements that give me valuable information about the program:

Strings Value	Relevance
gethostbyname	This is a standard Unix function that translates a host or clients IP address into a DNS name. This suggests that the program somehow interacts with the network.
lokid: Client database full	Since lokid is not a standard Unix function, nor is it a standard Linux program, this line suggests the real name of the program is lokid. The existence of a "Client Database" suggests that this is a network program of some sort. Since there is a chance that it could be "full", the program is likely collecting some type of information.
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	This appears to be some sort of dynamic status out put. The %d and %s fields represent string substitutions and the variables they are assigned to suggest status or reporting information.
[fatal] Cannot go daemon	This appears to be an error message generated by the program. Again, this suggests the program is a daemon.
/dev/tty /tmp	These two lines correspond to existing files or directories on the Linux system. /dev/tty is a terminal device and /tmp is a world writable directory, often used to store temporary files, as it's name suggests.
lokid -p (i|u) [-v (0|1)]	This appears to be a usage message displayed to the user to show what options are available with the program. It is now almost certain that the original name of this program was lokid.
LOKI2 route [(c) 1997 guild corporation worldwide]	"LOKI2" is probably the name of a suite of programs of which our file, lokid is a part. If this information is accurate, the program was originally written in 1997 by "guild corporation worldwide"
lokid: client <%d> requested a protocol swap sending protocol update: <%d> %s [%d]	This appears to be a log or status message suggesting the program is capable of changing what protocols it is using for it's purpose.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

I now have several very useful pieces of information. First of all, I have a name: "LOKI2", with which I can begin searching for more information.

Using the Google[2] search engine, I entered "LOKI2" and found several references of which I list below.

From WindowsSecurity.com:

"LOKI2 is an information-tunneling program. It is a proof of concept work intending to draw attention to the insecurity that is present in many network protocols. In this implementation, we tunnel simple shell commands inside of ICMP_ECHO / ICMP_ECHOREPLY and DNS namelookup query / reply traffic. To the network protocol analyzer, this traffic seems like ordinary benign packets of the corresponding protocol. To the correct listener (the LOKI2 daemon) however, the packets are recognized for what they really are. Some of the features offered are: three different cryptography options and on-the-fly protocol swapping (which is a beta feature and may not be available in your area)."[3]

And this one from ISS:

"Loki is a covert-channel client/server program published in the online publication Phrack. This program is a working proof-of-concept to demonstrate that data can be transmitted somewhat surreptitiously across a network by hiding it in traffic that normally does not contain payloads. The example code can tunnel the equivalent of a Unix RCMD/RSH session in either ICMP echo request (ping) packets or UDP traffic to the DNS port. This is used as a back door into a Unix system after root access has been compromised. Presence of LOKI on a system is evidence that the system has been compromised in the past."[4]

When data is sent across a network, it uses a method call encapsulation. In practical terms, this means that the relevant data is "wrapped" in special protocol packets that tell the network where to send the information, what kind of information it is, and what to do if there is a problem. For example, when a user wishes to retrieve their email, their email client sends a request to the email server. This request is encapsulated into a network packet that gives the location of the email server, and may also include the users username and password for authentication. In response, the server will get the email message from the users account, wrap it in a similar packet with the users location, and send it back.

---

[2] Google is a popular and powerful search engine. http://www.google.com
[3] http://www.secinf.net/unix_security/LOKI2__informationtunneling_program_and_description.html
[4] http://www.iss.net/security_center/static/1452.php

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The LOKI2 program takes advantage of weaknesses in this method by wrapping one type of data in a packet that indicates it is something else. For example, it may send all the usernames and password from a system wrapped in a packet to make it look like an email message. To a system administrator, it would appear that harmless email correspondence is taking place, when in fact a serious compromise has occurred.

From forensic evidence gathered so far, this description is a match. My program appears to be a network program that transfers data and is capable of protocol switching. However, there's more work to do before I can be certain.
To determine what system libraries this file might use, I can use the Unix ldd command. ldd prints shared library dependencies.

Trying to run ldd on atd did not return any output. Since the file command told me that shared libraries are being used by atd, I immediately know that something isn't quite right.

I went back to my strings output and noticed the top 2 lines:

```
[root@localhost work]# head strings.out
/lib/ld-linux.so.1
libc.so.5
```

A quick check of the file systems shows me that these two libraries aren't installed. I located the RPM (RedHat Package Management) packages for these files on Rpmfind.net[5], and installed them. RPM is a standard format on many Linux systems that eases the installation and removal of software packages. It correctly places the files on the system so that the user or other programs can use them.

The specific packages were libc-5.3.12-27.i386.rpm and ld.so-1.9.5-8.i386.rpm. I installed these using the standard rpm command:

```
[root@localhost john]# rpm -ivh ld.so-1.9.5-8.i386.rpm
warning: ld.so-1.9.5-8.i386.rpm: V3 RSA/MD5 signature: NOKEY, key ID
cba29bf9
Preparing...                ###########################################
[100%]
   1:ld.so                  ###########################################
[100%]
[root@localhost john]# rpm -ivh libc-5.3.12-27.i386.rpm
warning: libc-5.3.12-27.i386.rpm: V3 RSA/MD5 signature: NOKEY, key ID
cba29bf9
Preparing...                ###########################################
[100%]
```

[5] http://www.rpmfind.net

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
   1:libc                   ##########################################
[100%]
[root@localhost john]#
```

I then tried the ldd command again:

```
[root@localhost work]# ldd ./atd
        libc.so.5 => /usr/i486-linux-libc5/lib/libc.so.5 (0x40012000)
```

This information confirms what I already gathered from the strings output and having verification of this is a positive result.

The next step in the analysis is actually running the program to determine what it does and what, if any, changes to the local file system it makes.

Because the program has permission 0666, I'll need to make it executable first:

```
[root@localhost work]# chmod 766 atd
[root@localhost work]# ls -l atd
-rwxrw-rw-    1 root     root        15348 Aug 22  2002 atd
```

Because I know for certain that this is a system-compromising program, but I'm still not sure what it does, I don't want to run it as root. I'll log in as a normal user and attempt to run it first:

```
[john@localhost work]$ ./atd

[fatal] invalid user identification value: Unknown error
```

The above error suggests that the program is designed to only run as a particular user or users, most likely "root" or some other account installed by the cracker. Further analysis will be necessary to determine if only root can run this program.

For now, I'll have to run it using root, so I'm going to run it through the Unix command strace. This will give a clear picture of what the program is doing. strace traces all system calls and signals a program makes while it is running.

```
[root@localhost work]# strace -f -o strace.out ./atd

LOKI2   route [(c) 1997 guild corporation worldwide]
```

This output matches what I saw from the strings output.

I'll check the running process list to insure that the program is actually running and to get it's PID.

```
[root@localhost work]# ps ax |grep atd
  660 ?        S      0:00 rpc.statd
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
 1593 ?        S      0:00 ./atd
 1597 pts/0    S      0:00 grep atd
```

Process number 660 is a known daemon that I expect to be running. Since it doesn't match the name of my program, it's unlikely that this is it.

Process number 1593 matches the name and execution string I used to start the program.

Process number 1597 is the grep portion of the command I used to generate the process list and isn't relevant.

While the program is running, I'll run lsof. This program will give me a list of all open files on the system.

I then kill the program so that I can search through the output files I have generated.

I'll first take a look at strace.out.

Strace Line	Relevance
write(2, "\nLOKI2\troute [(c) 1997 guild cor"..., 52) = 52	Here is the system call that wrote the line I saw when running the program.
fork() = 1677	This is probably an attempt to trick audit programs into losing track of the program. Using the -f option with strings allows me to follow any forked processes.
open("/dev/tty", O_RDWR) = -1 ENXIO (No such device or address)	The program tried to open /dev/tty, but there is no indication why. Since the device doesn't exist on this system, the attempt failed.
chdir("/tmp") = 0	The program changed it's working directory to /tmp. As mentioned earlier, /tmp is generally world writable which suggests that the program is going to generate output to disk at some point.
umask(0) = 022	Here the program is setting the default creation permission to 022, or 0755.

Now I'll examine the lsof output:

Lsof output	Relevance
atd 1593 root cwd DIR 3,2 4096 211745 /tmp	This confirms what I learned from strace: the program is using the /tmp directory.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Based on the output I have viewed so far, I am still not certain what the program does. Because evidence suggests that it is a network program, I'll start the program again and see if I can determine what it is doing.

With the program now running, I'll check to see if it has opened any network ports.

```
[root@localhost binary]# netstat -ap
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
raw        0      0 *:icmp                  *:*                     7           4372/atd
raw        0      0 *:255                   *:*                     7           4372/atd
```

Indeed, atd has opened 2 raw sockets. This adds to the mounting evidence that this is a network program, and is consistent with my belief that this is "LOKI2". This also provides further evidence that this program needs to run as root. On Linux systems, raw sockets can only be opened by the root user.

Raw sockets are typically used under two circumstances:

- When using new network protocols for which no other process is designed to handle.
- For protocols that do not have a user interface, such as Internet Control Message Protocol (ICMP)[6].

As we can see, ICMP is in fact the protocol being used here. This suggests that atd is using the ICMP protocol for its communication. Given the nature of the program, this isn't surprising because ICMP packets are generally harmless and would not raise any immediate concerns if spotted by a system administrator. This is because ICMP is used almost constantly by Internet connected devices to determine a variety of things, most commonly whether or not a system is available to receive information.

Source Analysis

I have a high degree of certainty at this point that the binary is "LOKI2", but to be sure, I'll need to compare it against the real "LOKI2".

Using the Google search engine, I did a search for "LOKI2" and located the source at Phrack[7], a popular web site for cracking/hacking tools.

[6] ftp://ftp.rfc-editor.org/in-notes/std/std5.txt
[7] http://www.phrack-dont-give-a-shit-about-dmca.org/show.php?p=51&a=6

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Phrack stores their source code as part of their digital magazine, so I couldn't download it directly. Instead, they provide an extraction utility, cleverly called extract to pull source code from the textual magazine.

I cut and pasted the C source code for extract into a text file on my analysis system and compiled it. I then ran extract in the directory that contained the text magazine, and sure enough, it pulled everything out. In addition to some other files that aren't relevant, I now had the source code for "LOKI2".

All attempts at compiling "LOKI2" failed with header errors. These errors indicated that the necessary programs needed to compile this program were not present on this system. Based on the information I got from running atd through strings, I knew that "LOKI2" had been written in 1997. Realizing that there had been many changes in the compiler and development environment on Red Hat Linux between 1997 and 2003, I figured that I would need to install an older version of Linux to successfully compile the program. A search on the Internet told me that I would probably need to go back to version 4.2 or earlier. I made this assumption by cross-referencing the release date of Red Hat 4.2 and the date on the "LOKI2" source.

Unfortunately, I wasn't able to find any ISO images for RH 4.2. ISO images are special files that can be used to create CD's. I am fortunate enough however to have an associate who is something of a packrat. I contacted him and he had RH 4.0. Although this wasn't exactly the version I was looking for, it was the same major version, 4.X, so I felt there was a high degree of likelihood that it would still work.

I installed the RH 4.0 onto an unused system in my test lab using the CD's I had gotten from my colleague. I transferred the "LOKI2" source code from my test system to the new 4.0 system using a standard 3.5" floppy disk.

"LOKI2" can take a couple of compile time options, the most notable of which has to do with encryption. You can chose to use strong, weak, or no encryption. I chose to use no encryption so that I could view the information being passed using a network sniffer if necessary. All other settings I left at default.

Now I have a binary of "LOKI2" that I can compare to atd.

The first thing I noticed about my newly compiled "LOKI2", is that it creates 2 binaries, not just one. Based on earlier evidence, my atd binary appears to be a renamed lokid, so I'll start with that.

Running md5sum against the 2 binaries would be pointless since I have no way of knowing the environment that atd was compiled on, meaning that the hashes

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

would be different. However, I decided to it anyway in the interest of thoroughness.

```
48e8e8ed3052cbf637e638fa82bdc566  atd
bcd8f022f784c8458e70ee57a596635f  lokid
```

Sure enough, they are quite different, just as I expected.

A better way to determine if these are the same program will be to compare the outputs of some of the tests I ran against atd, with those run against lokid.

I'll use strace to run lokid and then compare the output I have collected from atd.

The most obvious indicator is this line:

```
write(2, "\nLOKI2\troute [(c) 1997 guild c"..., 52) = 52
```

which is located in both the strace outputs.

However, there are a couple of other lines that cause some concern.

From atd:

```
execve("./atd", ["./atd"], [/* 21 vars */]) = 0
```

From lokid:

```
execve("./lokid", ["./lokid"], [/* 17 vars */]) = 0
```

Notice that the number of "vars" or "variables" that are listed by the program is greater in atd then in lokid. This suggests that atd may have been modified and additional functionality added above and beyond the standard lokid.

This suggestion is further supported the existence of this line in atd:.

```
personality(0 /* PER_??? */)      = 0
```

This appears to reference a function in atd that is not present in lokid. Clearly atd, while based on lokid, has additional capabilities.

The full strace outputs can be found in Appendix A. From this evidence, it's now clear that atd is in fact based on "LOKI2". More specifically, it is the server or daemon side of the program, called lokid. "LOKI2" relies on both a client and a server portion to transmit date via its covert channel.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Legal Implications

Now the technical analysis has been completed and the forensic evidence has been gathered, we can start to examine what the proper course of action should be.

Unfortunately, nothing in the analysis I did indicated that the program was ever run on the system. Even though "LOKI2" switched it's working directory to /tmp, it never appeared to write any files regardless of what sorts of tests I ran. If I had access to the entire compromised system, it is possible that additional evidence could have been gathered to determine if "LOKI2" had ever been run, and what the outcome was.

From a legal standpoint, this leaves me with trying to determine if the program was placed on the system as the result of unauthorized access. Specifically, was the person who placed the binary allowed to access the system and/or install software on the system without authorization from management?

In federal law, there are 3 major Acts that cover computer crime:

- Computer Fraud and Abuse Act, 18 U.S.C. §1030
- Wiretap Act, 18 U.S.C. §2511
- Electronic Communications Privacy Act, 18 U.S.C. §2701

I do know that the person is an employee, which means this falls into the category of "Intentional Conduct", as opposed to "Reckless Conduct" which would require the perpetrator be an "outsider" to the organization.

But because I can't prove that the any damage was done to the system, or that any losses resulted from the presence of the compromised binary, prosecution of this individual in a court of law could be difficult, and I'm not likely to involve law enforcement with the exception stated in Case 2, below.

I know have two possible avenues to consider:

Case 1: User has general authorization to install software on the system.

I'll carefully review the organizations security policy. If the policy was properly designed, then it should place blanket restrictions on the installation of any software not specifically related to the function, operation, or maintenance of the system. Since "LOKI2" is an obvious backdoor and well-known cracker program, it's unlikely that the security policy allows it to be installed and/or run. One possible exception would be if the system in question were used for security testing purposes. Given that the program was brought to me for analysis, I can assume that it wasn't supposed to be there, so this system was likely not designed for security testing.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Based on all this, organizational management would need to determine if the users intent was malicious, or just an innocent act of curiosity. Regardless, standing security policy should be reiterated to all users to insure proper use of systems.

Case 2: User did not have general authorization to install/run software on the system.

Under these circumstances, the ability for management to make a clear decision becomes easier. Because the user had no reason to be on the system at all, it's more likely that their intentions were malicious as opposed to innocent. Chances are they had to obtain access to the machine through some inappropriate means, such as social engineering, network password sniffers, and so on.

In addition to clearly being a violation of a well-written security/acceptable use policy, this type of intrusion is illegal. However, the seriousness of the offense depends on the nature of the system compromised. 18 U.S.C. §1030 defines a specific set of systems referred to as "protected computers":

- 18 U.S.C. §1030(e)(2): includes any U.S. government network, those used by banks and other financial institutions, and other network, domestic or foreign, that affect interstate of foreign commerce or communication of the United States.
- 18 U.S.C. §1030(e)(2)(B). Protected computers can include computers outside the United States.

If the system that was compromised meets the above criteria, then I would contact law enforcement immediately. If not, then the issue would be sent to management to make a decision on the employee's future.

Interview Questions

The analysis of the compromised binary I was given proved to be a commonly used backdoor designed to covertly send information from the host system to some remote client. Based on this, it's reasonable to assume that the perpetrator intended to access the system at some later date, or make inappropriate and possibly illegal use of any information gathered. For the purposes of this document, and I have worked under the assumption that the person was an employee. I will continue to do so in this section.

Before even starting an interview, I would want to know as much about this person as possible from management. Some specifics:

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

- Were they told they were being fired/laid off? Or is it possible they could have heard a rumor to that effect?
- How do they get along with their co-workers?
- How long have they been with the organization?
- Is there a history of rebellious or otherwise disruptive behavior?

It is important to get the above information because how the organization viewed and handled this employee plays a part in what their motivation may have been to install the backdoor and risk getting prosecuted.

As for the perpetrators themselves, I would like to get a confession to help strengthen any future prosecution in the courts, and to help reduce the amount of time and resources needed to investigate the case.

The nature of the questions I would ask in an interview would vary widely depending on the interviewee and the circumstances. Some of the more important factors:

- Are they technically savvy? That is, do they have a strong technical background?
- Do they seem angry? Scared? Amused? Etc…
- Do they talk "down" to me, or show me respect?

The list could go on. Part of being a good investigator/interviewer is the ability to tailor the interview to the situation and interviewee. Not only does this mean going into the interview with a solid plan, but you should also be able to change strategies as the interview progresses. Most people get caught between a desire to hide the truth and reveal it. A good interviewer knows how to play off this and maneuver people into providing details they might otherwise want to hide.

For the purposes of this interview, I'll assume that the individual fits a stereotypical hacker/cracker mold:

- Intelligent
- Arrogant concerning their technical abilities; e.g. they feel they are superior to their peers and their management
- Probably just trying to make a point with a "do it because I can attitude", not looking to steal corporate secrets or cause any real damage.
- Male, late teens to late twenties.

The profile gives me the ability to frame my questions.

1. "*So, <name>, I see you have been with the company for a few years. How do you like it here?*" This question will help me to ascertain how the person feels about their employer and co-workers. I will make some comment about the company myself; something generic like "*Yeah, I hear that all the IT staff here are top notch.*" My hope is to draw him into

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

discussing any grievances he has or to give me an opening. He may respond with "*Yeah, right*", or "*Maybe not ALL*". Ultimately, I want this part of the conversation to help give us a common ground and to show that I am sympathetic to whatever his situation might be.

2. "*<name>, do you ever do any work on system X?*" I want to establish that he has done work on or with the compromised system, either as a user or administrator. My forensic evidence tells me that his account accessed the system, so I know he has the potential for access. This question starts at a high level, but depending on his answer, I will start getting more specific. If he answers "*Yes*", then I might get specific about dates. If he says "*No*", I might act a little confused, but not accusatory, and discuss with him the fact that according to my records his account has accessed the system. My goal is to get him to admit that he had access to, and has logged into the compromised system.
3. "*<name>, I just read about yet another security hole in such-and-such Operating System. Can you believe that? Do you have much interest in information security?*" This question is condensed, and I would work on this during the course of conversation. I am attempting to get him to talk about their security related activities both on a personal and professional level. By showing that I have an understanding on the topic, they will be more likely to want to talk to me, either because they think I will understand their viewpoint, or because they want to show how smart they are. Either way, I've gotten an admission that they have knowledge of Information Security related tools and technologies.
4. Based on the information I get from question 3, I would want to dig deeper. *"Have you ever tested our network/systems? Do you think we are vulnerable?*" I'm continuing to play into their need to connect and share with me. But I'm hopefully taking them off the defensive by seeming to ask them for their help and opinion on the matter. Continuing on this line of discussion should get me close to where I want to be, which is the interviewee admitting they installed the backdoor, and perhaps admitting to several other acts under the pretense of helping or proving a point.
5. Lastly, if I have failed to gather enough information through the techniques described above, I would need to start turning to my actual forensic evidence. The general rule is to only reveal as little of what you know as possible, so I might start with something like *"<name>, according the log files I have gathered, your user account logged into system X at <time and date>. Was that you? Was their some issue you were trying to resolve?*" And so on.

Hopefully, by using well-known interviewing techniques, combined with solid forensic evidence, I can gather enough information to allow management to make the appropriate decision concerning the employee and the situation.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Synopsis of Case Facts

The system to be analyzed was brought to me for analysis by a colleague who had concerns about its security state.

My colleague wanted to continue to use the machine in its current state due to the large amount of proprietary data on the system. It had been left connected to the Internet for several years and several users had come and gone. One particular user had been booted from an ongoing development project under unfriendly circumstances, and the fear was that this person might have left behind sniffers, backdoors, or other tools that he could use to impede the progress of the project or to steal proprietary code. The system had been offline for a number of years, which accounts for the long lapse between the file dates and when the system was brought to me for analysis. The owner wished to restart the development process under trusted conditions.

Due to what turned out to be poor system administration practices, there were no trusted backups of the system or the code that the developers had been working on. This meant that a restore of the data to a clean system was impractical because compromised code or system files could be transferred to the new system.

By properly verifying that the host had not been comprised, my colleague could continue to use the system in its existing state, thereby saving the time and expense of moving the project and server files to another system. Alternatively, by identifying compromised files, all non-compromised files could be transferred to a clean system.

Ultimately, my analysis of the system uncovered no malicious code or otherwise compromised files. I did uncovered significant security and system administration problems that will need to be corrected by the system owner to insure a properly secured system.

System Description

This system had been used as a gaming and development server for a popular form of online game called a MUD, or Multi-User Dungeon.[8] My analysis indicated that it is a Intel processor based system, running Linux[footnote], a popular Unix style operating system.

MUD requires that a server accept connections so that players can interact with one another in a simulated environment. It is much like popular text based chat

[8] http://www.mudconnect.com/tmcfaq.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

systems, but with the added elements of adventure. For example, players may join together to explore new areas and battle monsters, like something out of fantasy novel such as "The Lord of the Rings."

The system in question was never actually completed in the sense that the MUD server was never available to the public. Its primary purpose was to serve as a development platform for the "writing" of the game. This meant that programmers logged into the system via remote connections and wrote the program code necessary to facilitate the game operation. Over the course of the development process, there were as many as 5 developers working on the program at any given time.

Prior to my receiving it, the system had been kept in a storage closet at the owner's private residence. Only residents of the home had access to it, which meant the owner and his wife. The owner brought the system to me personally when he made his request for the analysis. From that point on, until the system was returned to the owner, the system was stored at my private residence in a locked storage cabinet to which I had the only key.

Hardware

When I first received the host in my lab, I knew I had my work cut out for me on the hardware side. It was immediately evident that the system was not in its original factory state. At some point, an ATX style motherboard had been inserted into an AT style case[9]. I had confiscated the entire system and had to disassemble it to get at each part for proper identification.

I labeled each piece of the system using a simple naming scheme where:

HD = Hard Drive
VC = Video Card
NIC = Network Interface Card
FD = Floppy Drive
PS = Power Supply
CPU = Central Processing Unit

Tag Numbers	Description
HD0001	Quantum Fireball 3840AT Hard Drive, Serial # 396633517894
VC0001	Paradise`88 PVGA1A-JK Video Card, Serial # 028117201402
NIC0001	Digital DE205 NIC, Serial # TA42702379
FD0001	Panasonic F2250 3.5" Floppy Drive, Serial # JU-257A704P
PS0001	ASTEC AA14220 Power Supply, Serial # TWP35045
CPU0001	Intel Pentium 133mhz CPU, Serial # A80502133

[9] http://www.aardvarkinc.com/support/at_vs_atx.htm

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

System had 64mb of SIMM style RAM on an Intel based motherboard.

As seen in the chart above, the system was fairly typical in that it only had 1 of each part needed to build a fully functional system. It is possible, and quite frequent, that a system may have multiple hard drives (HD) or processors (CPU). If this had been the case, additional elements would have been labeled accordingly. For example, a second hard drive would have been labeled "HD0002."

Image Media

My first step was to obtain clean copies of the disk images so that they could be moved to the analysis machine for review.

The image extraction system (Red Hat Linux v7.3) contained two hard drives. Drive A contained the operating system. Drive B was blank and I first sterilized that drive to insure that any existing bits on the drive would not interfere with the images as they were being created. Sterilization of image media can be accomplished by using the popular Unix dd command in conjunction with the /dev/zero device.

```
[root@localhost root]# dd if=/dev/zero of=/dev/hdd
2370531+0 records in
2370531+0 records out
```

/dev/zero is a special Unix device that always returns \0 characters when read. In practical terms, this has the effect of overwriting every part of the hard drive with a "0", effectively and permanently erasing all existing data. This is a critical, because any residual data on a hard drive used to store forensic evidence could result in tainted, and therefore inaccurate, results.

Dd is a special Unix command that creates an exact copy of the input by reading in individual blocks of data from the source, and outputting them to a destination, in this case a file in the /images directory. Normal copy commands may skip over certain files or change permissions, timestamps, and other information that could be critical to an investigation. Equally important, dd can read the data without having to mount the source drive, meaning that the risk of corruption is minimized.

Now that the sterilization media is ready, I powered down the machine to attach the source system's drive to the extraction system. The drive I needed to attach was an IDE[footnote] type drive. I attached it by using a standard IDE cable connected to the 2nd IDE drive port on the motherboard of the analysis system.

I then powered up the machine to begin the actual imaging of the source system's drive. The analysis system was set to boot only off the primary IDE

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

drive, and the source drive was not set to mount automatically. This is important because I didn't want to accidentally boot the source drive, thereby risking contamination of the files on that drive.

After the system was up and running, I first mounted the sterilized media on the /images directory of the extraction system so that I can write the image files to it. This works by taking an entire drive, in this case the sterilized media, and assigning it to it's own directory. To the user, /images appears as a normal directory, but when files are written into that directory, they are actually being put on the sterilized drive.

Next, I need to know what partitions are contained on the source drive so that I can execute the proper dd command to write the image files.

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

Disk /dev/hdc: 128 heads, 63 sectors, 935 cylinders
Units = cylinders of 8064 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1             1       261   1052320+  83  Linux
/dev/hdc2           262       935   2717568    5  Extended
/dev/hdc5           262       327    266080+  83  Linux
/dev/hdc6           328       393    266080+  82  Linux swap
/dev/hdc7           394       935   2185312+  83  Linux
```

I need to make images of every partition except /dev/hda2, which is an extended partition that contains /dev/hd5, /dev/hda6, and /dev/hda7.[10] In this case, I chose to make copies of the individual partitions rather then the whole drive. I did this to make the data more manageable by reducing the size of each file, and so that I could later mount each partition separately for examination, reducing the risk of any cross contamination between data sources.

Now I can use dd to write the images to the sterilized media.

```
[root@localhost root]# dd if=/dev/hdc1 of=/images/hdc1.img
2104640+0 records in
2104640+0 records out
[root@localhost root]# dd if=/dev/hdc5 of=/images/hdc5.img
532160+0 records in
532160+0 records out
[root@localhost root]# dd if=/dev/hdc6 of=/images/hdc6.img
532160+0 records in
532160+0 records out
[root@localhost root]# dd if=/dev/hdc7 of=/images/hdc7.img
4370624+0 records in
4370624+0 records out
```

Now I'll use the md5sum program to insure that the images I have collected are exact duplicates of the drive partitions.

```
[root@localhost images]# md5sum /dev/hdc1 >> hdc1.md5
[root@localhost images]# md5sum /images/hdc1.img >> hdc1.md5
```

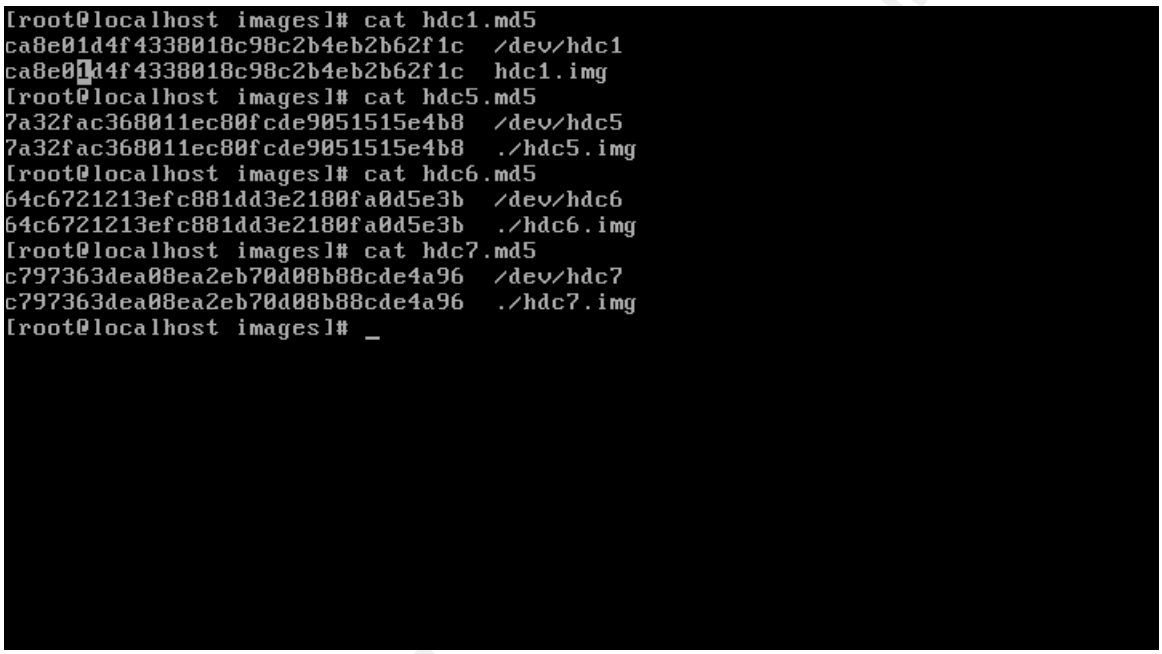
[10] http://www.linux.org/docs/ldp/howto/Large-Disk-HOWTO-13.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
[root@localhost images]# md5sum /dev/hdc5 >> hdc5.md5
[root@localhost images]# md5sum /images/hdc5.img >> hdc5.md5
[root@localhost images]# md5sum /dev/hdc6 >> hdc6.md5
[root@localhost images]# md5sum /images/hdc6.img >> hdc6.md5
[root@localhost images]# md5sum /dev/hdc7 >> hdc7.md5
[root@localhost images]# md5sum /images/hdc7.img >> hdc7.md5
```

I created a specific text file for each partition and saved both md5sum values to that file. This gives me an easy record that I can use during further analysis.

Now to compare the values contained in those files.



As we can see in the above screen capture, the MD5 hashes for each partition and image file match. This is proof that the images I have captured are identical to the actual partitions.

Media Analysis of System

To conduct the media analysis, I am going to use the industry standard programs, Task and Autopsy.[11] These two programs work in conjunction, with Task being the program that does the actually analysis, and Autopsy being the graphical front end that helps users navigate the various commands and keep data organized.

Task and Autopsy use the image files I created in previous steps and load them into what it calls its "evidence locker." From there, date can be viewed in numerous ways, as will be shown in subsequent steps. Although all of this

[11] http://www.atstake.com/research/tools/task/

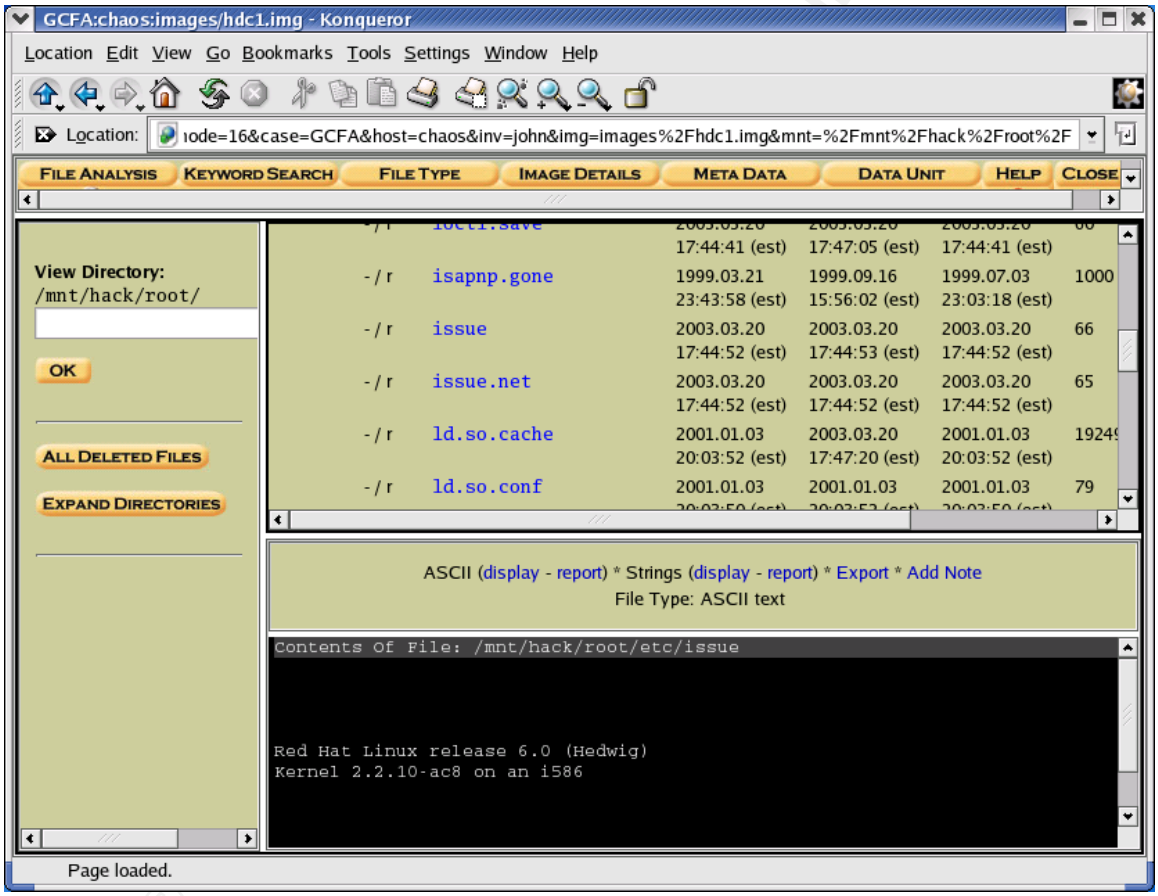
© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

functionality can be duplicated with other tools, Task and Autopsy help bring it all together in a organized and easy to use system, allowing for more time to be spent on the actual analysis.

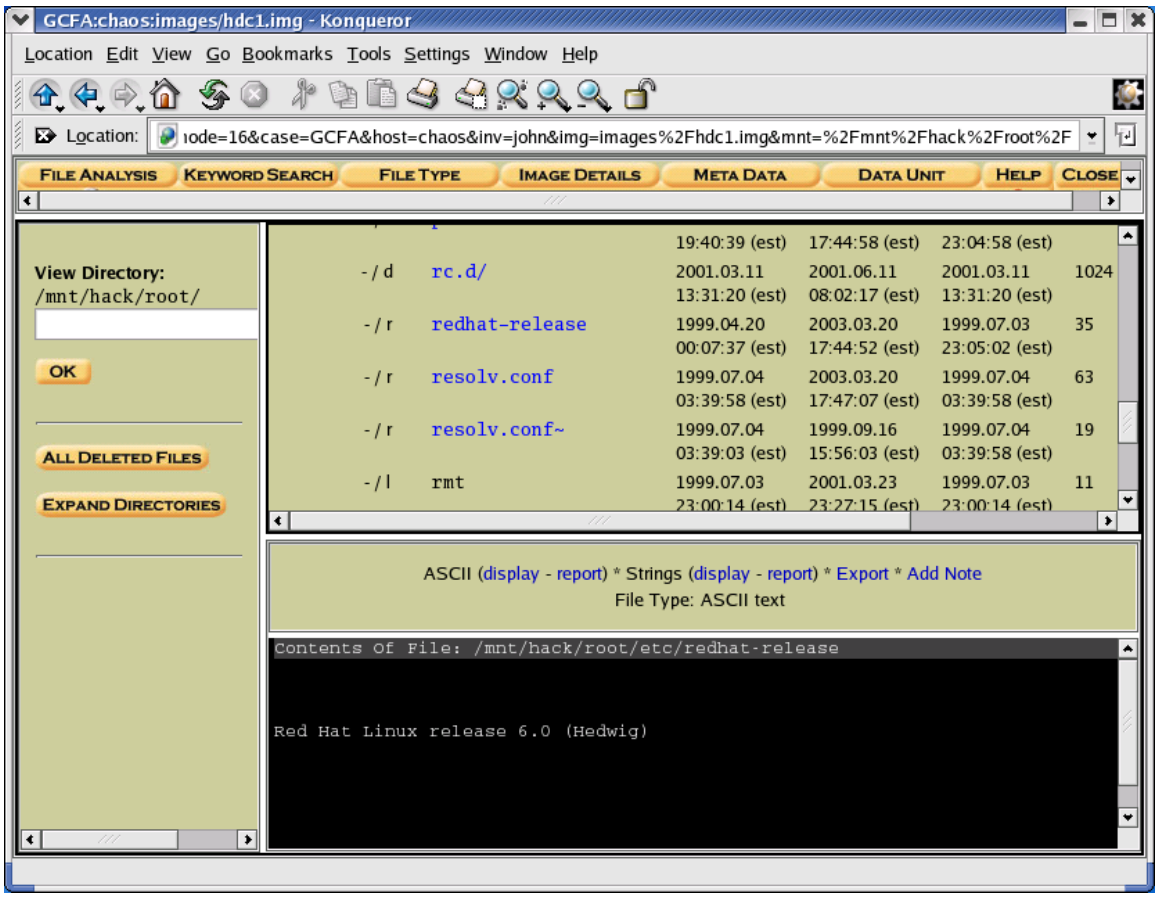
I followed the standard instructions on setting up Task and Autopsy, culminating in the addition of the hdc1.img file so that I could begin examining it.

The first thing I want to do is examine what type of system this is. My colleague has told me that this is a Linux system. Let's get some more details.

Using the Autopsy File system browser, I'll navigate to the /etc directory and check out a couple of files that normally hold version data.



© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



As you can see in the above screen shots, Autopsy provides a point-and-click interface:

1. The top bar shows buttons that take the user to different areas where analysis can be performed. In this case, we see the “File Analysis” section.
2. The left bar has a search box, plus any display options that are available for the current view.
3. The large central window is the navigation area where the user can move around on the file system.
4. The bottom window shows the actual contents of any readable file that the user has clicked on, plus additional display options.

So, in the two screen shots shown, I navigated to, and clicked on to display each of the files mentioned.

Based on the information found in these two files, we now know:

- **Operating System:** Red Hat Linux release 6.0 (Hedwig)
- **Kernel Version:** 2.2.10-ac8 on an i586

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Although it is certainly possible that a cracker could have modified these files, it is unlikely since neither file contains any information that could be used to identify the cracker or his activities. It is useful to me because I now have a better idea of what to expect from this system in terms of what files should and shouldn't exist; an important step in identifying potentially compromised binaries, log files, or other system information.

I'll come back to the /etc directory later. First, I want to see if there is any evidence located in the system logs. Typically, experienced crackers will cover their tracks by eliminating evidence from log files. Skilled crackers will remove only those entries that could identify them or their activities in hopes of avoiding detection altogether. Less experienced crackers will often remove or "zero" out log files to eliminate evidence. In the latter case, important time line information can be gathered about cracker activities, as well as what their activities may have related to, based on what log files are gone or empty.

The first thing I notice in the log directory is the wide discrepancy between the access dates among the files. In fact, some of the files have a recent date of March 20th, 2003. I confirmed with my colleague that the system had been booted up briefly on this date before being brought to my lab.

Unfortunately, this fact could create legal problems, since it is now possible that the evidence has been contaminated between the time the system was brought down under suspicion of malicious activity and the time it arrived in my lab. This could cause the system to be inadmissible in the event prosecution of any offender is undertaken. It does not hamper the investigation, since our goal is to determine what, if any, compromises have taken place. Knowing the circumstances under which the system was booted on March 20th, 2003, will help me create a valid time line.

I'll go through each log to look for entries that appear out of place, or errors that could indicate problems. Any log files not listed below were empty.

(In order to conserve space, I have shown the output in text format, rather then as screen shots from Autopsy.)

Boot.log

In boot.log, I found several entries like the following, including one on March 20th, the last time the system was booted.

```
May  7 14:42:20 chaos lpd: /etc/rc.d/rc3.d/S60lpd: Binary: command not found
May  7 14:42:20 chaos lpd: /etc/rc.d/rc3.d/S60lpd: lpd:Binary: command not found
May  7 14:42:20 chaos lpd: /etc/rc.d/rc3.d/S60lpd: lpd:lpd:#!/bin/sh: No such file or
directory
May  7 14:42:20 chaos lpd: /etc/rc.d/rc3.d/S60lpd: lpd:lpd:#: command not found
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

S60lpd is a symbolic link to the lpd file in /etc/init.d so I'll examine that file.

```
Contents Of File: /mnt/hack/root/etc/rc.d/init.d/lpd
Binary file /usr/sbin/lpd matches
lpd:Binary file /usr/sbin/lpd matches
lpd:lpd:#!/bin/sh
lpd:lpd:#
lpd:lpd:# lpd           This shell script takes care of starting and stopping
lpd:lpd:#
lpd:lpd:# chkconfig: 2345 60 60
lpd:lpd:#   It is basically a server that arbitrates print jobs to printer(s).
lpd:lpd:# processname: lpd
lpd:lpd:# config: /etc/printcap
```

Above are the top ten lines of that file. The remainders are formatted similarly.

The entire file is formatted incorrectly for a startup script. In fact, every line in that script is an invalid shell command, due to the 'lpd:lpd' at the beginning of each line. It appears that the script was generated as the output of some parsing program, possibly grep or find. Regardless, nothing in the script will actually start any program, but it does certainly prevent the lpd printing service from starting, which while possibly annoying, does not appear to constitute a serious compromise. It is more likely the result of poor procedure on the part of a system administrator.

Logfile: cron

There are 5 cron log files listed here: cron, cron.1, cron.2, cron.3, and cron.4. They all contain repeated entries like these:

```
root (05/20-03:50:00-6091) CMD (   /sbin/rmmod -as)
root (05/20-04:00:00-6093) CMD (   /sbin/rmmod -as)
root (05/20-04:01:00-6095) CMD (run-parts /etc/cron.hourly)
root (05/20-04:02:00-6097) CMD (run-parts /etc/cron.daily)
```

/sbin/rmmod –as is an expected entry. It attempts to remove any loadable modules not being used by the kernel in an effort to maintain system efficiency.[12]

run-parts /etc/cron.hourly and run-parts /etc/cron.daily will run any scripts located in the listed directories. An examination of /etc/cron.daily reveals 4 simple scripts:

- logrotate
- makewhatis.cron
- slocate.cron
- tmpwatch

All four scripts are normal and expected.

[12] Introduction to Linux kernel modules By Vans Information http://www.freeos.com/articles/4051/ 2001-05-16

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

/etc/cron.hourly was empty and therefore not running anything. Since I was there, I also checked /etc/cron.weekly and /etc/cron.monthly just to be safe. Both directories where empty.

Logfile: dmesg

dmesg contained expected entries placed there during the last boot.

Logfile: lastlog

Examination of this log shows that only the "root" account has logged in. Since I have been told that the potential crackers username was removed from the system, this is not unexpected.

Logfile: maillog

All the maillog files were empty. While the information I have concerning the history of the system indicates that it was never used as a mail server, it still had the ability to send email messages via the installed sendmail program. Therefore, it is possible that a user could have sent email messages, perhaps containing important file attachments to an Internet account and then removed all traces from the mail logs.

Logfile: messages

There are five messages log files. They all contain relevant entries like the following.

```
Jun  4 13:50:46 chaos ftpd[2776]: ANONYMOUS FTP LOGIN FROM gatekeeper.bristol.com
[207.41.40.76], chadg@cosmo.bristol.com
Jun  5 08:05:20 chaos ftpd[3291]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76],
cerwin
```

```
May 18 10:20:06 chaos PAM_pwdb[5426]: (login) session opened for user cerwin by (uid=0)
May 18 10:20:22 chaos PAM_pwdb[5426]: (login) session closed for user cerwin
```

The above entries are of particular interest because the individual who appears to be logging in is the ex-user that is of concern.

I will address these entries further in the "Time Line" section.

Logfile: secure

```
Jun  4 10:02:29 chaos login: LOGIN ON 1 BY cerwin FROM gatekeeper.bristol.com
May 18 10:20:06 chaos login: LOGIN ON 1 BY cerwin FROM cl081.usachoice.net
```

We find additional references to the user "cerwin" (the username in question) that helps verify what we already found in the messages log file.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Logfile: xferlog

```
Tue May 15 13:35:18 2001 10 x.x.x 1149724 /home/ftp/pub/Tool_-_Late
ralus_-_04_-_Mantra.mp3 b _ o r cerwin ftp 0 * c
Tue May 15 13:36:15 2001 57 x.x.x 8667602 /home/ftp/pub/Tool_-_Late
ralus_-_05_-_Schism.mp3 b _ o r cerwin ftp 0 * c
Tue May 15 13:36:38 2001 20 x.x.x 3269455 /home/ftp/pub/Tool_-_Late
ralus_-_06_-_Parabol.mp3 b _ o r cerwin ftp 0 * c
```

There are many entries in the xferlog files like the above. Clearly the user transferred a great number of files to and from the server. However, none of the files transferred appeared to be malicious. Most were images, MP3 files, or files related to his development activities.

Examination of the log files has given me some important clues as to the activity of our potential cracker. Having this information will help in directing the rest of my investigation.

Analysis of file from /etc

I now want to examine some of the files in the /etc directory, since most of the important system configuration files live in that directory.

After a careful examination of all the files in this directory, including its sub-directories, I found the following files that contained items of interest.

File: exports

```
/home/parts-mall204.249.184.230(rw,no_root_squash)
```

The exports file controls what directories are mountable as NFS partitions or directories. This is the sole entry from the file and caught my eye because it seemed out of place with what I knew the primary function of the machine to be.

Checking with my colleague revealed that he had placed that entry there as part of another project he was working on.

File: fstab

```
/dev/hda1               /                       ext2    defaults        1 1
/dev/hda7               /home                   ext2    defaults        1 2
/dev/hda5               /tmp                    ext2    defaults        1 2
/dev/hda6               swap                    swap    defaults        0 0
/dev/fd0                /mnt/floppy             ext2    noauto          0 0
/dev/cdrom              /mnt/cdrom              iso9660 noauto,ro       0 0
none                    /proc                   proc    defaults        0 0
none                    /dev/pts                devpts  mode=0622       0 0
#208.7.247.241:/mnt/linuxbackup/mnt/linuxbackupnfswsize=4096,rsize=4096,hard,intr0 0
#208.7.247.241:/home/parts-mall/mnt/parts-mallnfswsize=4096,rsize=4096,hard,intr0 0
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The contents of the fstab support our previous knowledge concerning the existing partitions on the system. The additional entries of /dev/fd0 and /dev/cdrom are for the 3.5" floppy drive and CD-ROM drive respectively, both of which are normal. I verified the existence of these two devices during my examination of the hardware.

The last line is a further expansion of the line from exports that I have already learned is a legitimate entry. The 2nd to last line is something new. Again, my colleague confirmed that he had placed the entry there and so I moved on. Both of these lines are commented out, as evidenced by the "#" proceeding each one, so they would not be mounted in any event.

File: group

```
mudadmin:x:200:mud,zxylos,cerwin,brand,strom
chaosweb:x:201:zxylos,cerwin,brand,strom,scott
muddevel:x:202:zxylos,cerwin,brand,strom
cvs:x:203:cvs-rw,cvs-read,zxylos,cerwin,brand,mud
cerwin:x:501:
```

The above lines from the group file reveal some poor security practice. Even though I have been assured that the "cerwin" username has been removed from the system, the name still appears in the groups he was a member of previously. While this doesn't pose any specific threat, it is generally good policy to remove all traces of a username after it is no longer needed on the system.

File: inetd.conf

```
cvspserver stream tcp nowait root /usr/bin/cvs cvs --allow-root=/home/cvsroot
pserver
```

This line from inetd.conf is not standard with Red Hat 6.0 systems. However, the server was used as a development platform, and cvs is a well-known development system used by groups of programmers to insure code integrity.[13]

File: Muttrc

I was surprised to find this file. Previous data from the maillog suggests that this server never sent or received any mail. Mutt is a common, text based mail reader. The Muttrc file is typically only created when the program is installed. The fact that it exists points to someone using this system for email purposes at some point.

File: passwd and shadow

```
cerwin:x:501:501::/home/cerwin:/bin/tcsh
```

[13] "Concurrent Versions System (CVS) http://www.gnu.org/software/cvs/

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

It appears that the username of the suspected cracker remains on the system as it still appears in the passwd file. The "x" in the usual password field indicates that this system is using shadow passwords[14].

```
cerwin:$1$k46yKEKe$puByGD0.k7yYqlvjv5ixn0:10828:0:99999:7:-1:1:134538468
```

This entry from shadow confirms it.

The existence of this username is group, passwd, and shadow can lead to two conclusions:

1. The person responsible for removing the account simply failed to do it, or did not follow the correct procedure for removing accounts from this system.
2. The user has gained access to the system and re-created the account.

Given some of the other things I have found concerning the maintenance of the system, Option 1 is not out of the question. Option 2 is a viable alternative as well, although I have not yet found any solid evidence that supports that conclusion.

SUID, SGID

I now want to see if I can find anything interesting about the remainder of the files on the system.

To start, I'll do a search for any files that are marked SUID, SGID. These types of files give permissions that allow unprivileged users to assume that of the super user, or "root".

This can be particularly dangerous because ordinary users can execute powerful commands that would normally be limited to the super user. This means that the user can cause serious damage or compromise on a machine without even having root access.

To accomplish this, I'm going to leave Autopsy and go back to the command line.

```
[root@localhost root]# find /mnt/hack/root \( -perm -004000 -o -perm -002000 \)
-type f -ls
 24536   14 -rwsr-xr-x   1 root     root        13208 Apr 13  1999 /mnt/hack/root/bin/su
 24547   53 -rwsr-xr-x   1 root     root        52788 Apr 17  1999
/mnt/hack/root/bin/mount
 24548   27 -rwsr-xr-x   1 root     root        26508 Apr 17  1999
/mnt/hack/root/bin/umount
 24556   16 -rwsr-xr-x   1 root     root        14804 Apr  7  1999
/mnt/hack/root/bin/ping
```

[14] Linux Administrator's Security Guide - Passwords by Kurt Seifried
http://www.windowsecurity.com/whitepapers/Linux_Administrators_Security_Guide__Passwords.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The above is the top few lines of output from the program. The remainder of the output can be found in Appendix A.

Surprisingly, I didn't find any SUID, SGID files on the system that shouldn't have been. Given what I have seen of this system previously, I was certain that I would. I say that because lazy or inexperienced system administrators will often change files in this way in order to avoid having to use tools like sudo, which can provide limited access to super user commands to regular users.[15]

CHKROOTKIT

One very useful utility is chkrootkit.[16] This program contains a list of all know root kits, or exploits, and will search through a file system to locate them. While experienced crackers can certainly get around this, many inexperienced ones, or "script kiddies"[17] as they are often called, don't think to properly hide their installed software.

Because this system did not have file integrity software, such as Tripwire, installed on it, I have no good way of determining if any key binaries have been compromised. Running chkrootkit, is an effective and easy step in the analysis process.

Appendix B contains the complete output from chkrootkit.

As we can see from the output, chkrootkit did not find anything suspicious. This is good news, although as mentioned above, not final proof that nothing has been compromised.

History Files

Unix shells maintain a command history, usually stored in a hidden file in the users directory. This is to accommodate a feature that allows a user to scroll forward and back through their previous commands in an effort to reduce the amount of typing needed. This file does not typically store all the commands that a user enters dating back to their first login, but rather stores a predefined number of commands, with a "first in, first out" type of rotation.

To look at the potential crackers .bash_history file, I'll need to mount the correct partition.

```
[root@localhost cerwin]# mount -ro,loop,nodev,noexec,noatime ./hdc7.img
/mnt/hack/root/home
```

[15] http://www.courtesan.com/sudo/
[16] http://www.chkrootkit.org/
[17] The term "script kiddies" refers specifically to the tendency of inexperienced crackers ("kiddies") to use pre-made cracking tools ("scripts").

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

There are several commands recorded in this file that are of interest:

"ls ~zxylos/"

It appears that "cerwin" was attempting to view the contents of another's users directory. There is no record as to whether or not he was successful, but this action does suggest a questionable level of ethics.

"su"

This command appears several times, although not in a row. This would seem to indicate that "cerwin" had access to the "root" account because this command allows a user to become the "root" user. Unfortunately, this means that he had access to virtually everything on the system.

Startup Scripts

I want to check on the /etc/inittab file, which can be used to start programs and sets some system defaults such as runlevels[18]. Looking at this file shows me that the system comes up in runlevel 3 by default, which is normal for Linux systems acting as servers.

I want to have a clear picture of what gets started on the system when it boots up. To do this, I'll take a look at the /etc/rc.d directory and sub-directories, which contain the startup scripts for this system.

In /etc/init.d there was one broken link, to the linuxconf program. This means that whatever program it originally linked to is no longer there. Curious, but not necessarily uncommon.

In rc.local I found 2 non-standard entries. The first one started the Apache web server:

```
# start apache
/home/apache/bin/apachectl start
```

I confirmed that the apachectl script actually exists and looks normal, so it's clear that Apache was being started every time the system booted.

The second entry looks like this:

```
# Start DOC
#/home/mud/muds/chaos/area/startup &
```

[18] The Linux Runlevel, by Doran Barton http://www.iodynamics.com/education/runlevel.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

I checked with my colleague and he told me that DOC stands for "Dominions of Chaos", the name of the MUD that they were working on. The "#" symbol indicates a comment. Comment lines don't actually get run, but are only there to help identify portions of the script. In this case, the actual MUD server has been commented out. Not malicious, but as in the case of lpd discussed earlier, this could be a minor annoyance if it is unintended.

Nothing else in this area appeared out of the ordinary.

Strings

The strings command attempts to search through a binary program and look for combinations of ASCII characters that might have some meaning. Unfortunately, I haven't found any other indications of malicious activity that might point me at specific files.

I do know that there are certain files that get run, both at startup and out of cron, so I'll run strings on these program to insure that they appear normal.

As I found in the cron log file, /sbin/rmmod gets run on a regular basis throughout the day. If a cracker wanted to disguise a program, this would be a good place to do it. This is particular true because many inexperienced system administrators don't fully understand the Linux Kernel module system[19], and wouldn't know when something was amiss.

```
[root@localhost sbin]# strings rmmod |less
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
stdout
snprintf
perror
malloc
__bzero
```

By way of example, I've included the top few lines. What we see is that the strings command has scanned through the rmmod binary and listed any combinations of ASCII (text) characters it could find. In the above list, we have references to a dynamic library and several standard C functions.

I'll run the same test on commonly compromised binary, login.

```
[root@localhost bin]# strings login |less
/lib/ld-linux.so.2
```

[19] The Linux Kernel Module system essentially loads and unloads different information into the operating kernel in order to add or remove functionality from the system. This is to help maintain efficiency by only having needed programs loaded into memory.
http://www.tldp.org/HOWTO/Kernel-HOWTO.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
__gmon_start__
libcrypt.so.1
libpam.so.0
_DYNAMIC
_GLOBAL_OFFSET_TABLE_
pam_set_item
```

I chose this program because it has considerable control over what happens when a user logs into the system. Crackers will often replace it with version that will sniff passwords, or start other programs to conduct malicious activity. I had no reason to believe this file was compromised, and my strings analysis didn't reveal anything to the contrary.

In general, strings can be used to search for all sorts of things. In particular, it is useful when attempting to identify a program whose function is unknown. Because it outputs library dependencies, C functions, comments, command line and so on, the identity of a program can often be gleaned by simply running this tool. If not, the output it generates can open up new leads and avenues of investigation.

Because this system was used as a development platform, it seemed logical that the MUD source code would be a likely target for a malicious developer. Anything from simply deleting key files to inserting backdoors that could be used to damage the game in the future were possibilities. My inclination was to search through all the source code files looking for instances of "cerwin."

The source code files for the game are not in binary format, so it is not necessary to use the strings command. Instead, I'll use a two common Unix utilities, find and grep.

```
[root@cambot src]# find . -exec grep "cerwin" {} \; -print
date    97.11.24.15.37.56;      author cerwin;  state Exp;
./RCS/cmd_olc.cc,v
date    97.12.30.21.11.08;      author cerwin;  state Exp;
./RCS/chaos.h,v
date    98.01.16.04.23.36;      author cerwin;  state Exp;
date    97.11.24.16.11.16;      author cerwin;  state Exp;
date    97.11.24.15.21.59;      author cerwin;  state Exp;
./RCS/cmd_info.cc,v
date    97.11.24.18.53.47;      author cerwin;  state Exp;
./RCS/comm.cc,v
date    97.11.24.16.26.06;      author cerwin;  state Exp;
./RCS/fight.cc,v
date    97.11.24.15.21.42;      author cerwin;  state Exp;
./RCS/commands.h,v
date    97.12.30.21.11.24;      author cerwin;  state Exp;
date    97.11.24.15.21.25;      author cerwin;  state Exp;
date    97.11.24.00.45.43;      author cerwin;  state Exp;
date    97.06.02.04.17.38;      author cerwin;  state Exp;
./RCS/interpret.cc,v
date    97.05.30.01.49.58;      author cerwin;  state Exp;
date    97.05.29.21.40.57;      author cerwin;  state Exp;
./RCS/olc.cc,v
date    97.11.23.21.05.11;      author cerwin;  state Exp;
./RCS/main.cc,v
date    97.05.30.01.49.40;      author cerwin;  state Exp;
./RCS/move.cc,v
date    97.06.05.18.08.29;      author cerwin;  state Exp;
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
date    97.06.03.05.28.31;      author cerwin;  state Exp;
date    97.06.02.04.27.43;      author cerwin;  state Exp;
./RCS/olc_cmds.cc,v
date    97.11.24.18.53.55;      author cerwin;  state Exp;
./RCS/proto.h,v
date    97.06.02.03.48.42;      author cerwin;  state Exp;
date    97.05.29.18.50.52;      author cerwin;  state Exp;
./RCS/save.cc,v
date    97.11.23.22.01.41;      author cerwin;  state Exp;
./RCS/update.cc,v
date    97.06.14.20.32.10;      author cerwin;  state Exp;
date    97.05.26.22.45.09;      author cerwin;  state Exp;
./RCS/utility.cc,v
```

As the output of this command shows, there are several areas in the source code which "cerwin" worked on. Using good programming practice, "cerwin" added comments to the areas of the program files he was working on that indicate who wrote that particular section.

I presented this information to the owner of the machine so that he could properly check the source files for any abnormalities. It is necessary for someone who is intimately familiar with the source code to perform this task, because malicious code inserted by "cerwin" may not fit any known signature that an analysis would catch. For example, "cerwin" could have written in a function that would allow him to log into the game and cause damage. Because I don't know what should and shouldn't be present in the code, I am unable to make that type of determination.

Hidden Directories

Unix based systems have the ability to "hide" files and directories by placing a "." in front of the name. For example, .myfiles. This causes the files to not be listed when a normal ls command is executed on the system. Typically, these files/directories are ones that the user does not normally need to access manually, and so "hiding" them helps reduce screen clutter. Here is an example listing containing "hidden" files and directories:

```
[root@localhost test]# ls -l
total 0
-rw-r--r--    1 root     root            0 Mar 28 10:19 forensic_files
-rw-r--r--    1 root     root            0 Mar 28 10:19 sansrules
```

In the above list, we see only 2 files.

```
[root@localhost test]# ls -al
total 8
drwxr-xr-x    2 root     root         4096 Mar 28 10:19 .
drwx------   19 john     john         4096 Mar 28 10:18 ..
-rw-r--r--    1 root     root            0 Mar 28 10:18 .bashrc
-rw-r--r--    1 root     root            0 Mar 28 10:19 forensic_files
-rw-r--r--    1 root     root            0 Mar 28 10:18 .gtkrc
-rw-r--r--    1 root     root            0 Mar 28 10:19 sansrules
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

In the above list, I have added the -a parameter to ls, which tells it to list hidden files. Now we see that there are actually 6 files in the directory. .bashrc and .gtkrc are hidden. The "." and ".." directories specify this directory and the one above it respectively.

Giving them odd, or seemingly harmless names can also hide files/directories. A popular cracker trick is to use a blank space or spaces as the directory name. Doing so makes it difficult to see that directory under casual examination. Here is an example

```
[root@localhost test]# ls -l
total 0
-rw-r--r--    1 root     root            0 Mar 28 10:21
-rw-r--r--    1 root     root            0 Mar 28 10:19 forensic_files
-rw-r--r--    1 root     root            0 Mar 28 10:19 sansrules
```

For the above, I added a file named " ". It shows up as the first file, but as you can see, the name field appears empty. For someone who wasn't looking carefully, this file could easily go unnoticed.

Crackers may also create directories that sound like they should exist, thereby fooling the less experienced user or system administrator. For example, crackers will often place files/directories in the /dev directory. This directory holds files used to access real or virtual devices on the system. Most /dev directories hold hundreds, if not thousands of entries, and many have cryptic sounding names. Our source system contains 2260 entries. Not the easiest thing to weed through, which is what crackers depend on. It's the "needle in a haystack" approach and is a perfect place to hide files.

So now that I have clearly outlined what I am looking for and why, let's see what I can find.

I'll start by searching for any "hidden" directories:

```
[root@localhost root]# find . -type d -name ".*"
.
./etc/skel/.kde
./root/.ncftp
./usr/share/control-center/.data
./usr/src/linux-2.2.5/pcmcia-cs-3.0.9/cardmgr/.depfiles
./usr/src/linux-2.2.5/pcmcia-cs-3.0.9/clients/.depfiles
./usr/src/linux-2.2.5/pcmcia-cs-3.0.9/debug-tools/.depfiles
./usr/src/linux-2.2.5/pcmcia-cs-3.0.9/flash/.depfiles
./usr/src/linux-2.2.5/pcmcia-cs-3.0.9/modules/.depfiles
```

There is nothing odd here. All the files listed above are expected. However, just to be on the safe side, I'm going to examine the contents of /root/.ncftp, since it exists in the /root directory. Ncftp is a popular ftp client, and looking in this directory may provide some interesting clues.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

A search through the files reveals a command history and 3 logs files. One of the log files contains the following entry that is of interest.

```
SESSION STARTED at:  Sun Aug  8 13:07:52 1999
   Program Version:  NcFTP 3.0.0/220 February 19 1999, 05:20 PM
   Library Version:  LibNcFTP 2.8.2 (February 18, 1999)
        Process ID:  21408
          Platform:  linux-x86
             Uname:  Linux|chaos.success.net|2.2.10-ac8|#12 Sun Jul 4 23:35:02 EDT
1999|i586
          Hostname:  chaos.success.net  (rc=2)
13:07:52  Fw: firewall.success.net  Type: 0  User: root  Pass: ********  Port: 21
13:07:52  FwExceptions: .success.net,localhost,localdomain
13:07:52  Resolving xxxxxxx.xxxxxxx.com...
13:07:52  Connecting to xxx.xxx.xxx.xxx...
13:07:53  Remote server is running NcFTPd.
13:07:53  Logging in...
13:07:53  220: xxxxxxx.xxxxxxx.com NcFTPd Server (free personal license) ready.
13:07:53  Connected to xxx.xxx.xxx.xxx.
13:07:53  Cmd: USER anonymous
13:07:54  331: Guest login ok, send your complete e-mail address as password.
13:07:54  Cmd: PASS root@chaos.success.net
13:07:55  Logging in...
13:07:55  230: You are user #1 of 3 simultaneous users allowed.
13:07:55
13:07:55       Logged in anonymously.
13:07:55  Cmd: PWD
13:07:55  257: "/" is cwd.
13:07:55  Logged in to xxx.xxx.xxx.xxx as anonymous.
13:07:55  Cmd: FEAT
13:07:55  211: Extensions supported:
13:07:55        CLNT
13:07:55        MDTM
13:07:55        MLST type;UNIX.mode;UNIX.owner;UNIX.group;size;modify
13:07:55        PASV
13:07:55        REST STREAM
13:07:55        SIZE
13:07:55        TVFS
13:07:55       End.
13:07:55  Logged in to xxxxxxx.xxxxxxx.com.
13:07:55  Cmd: CLNT NcFTP 3.0.0 linux-x86
13:07:56  200: Noted.
13:07:57  > quit
13:08:03  Cmd: QUIT
13:08:03  221: Goodbye.
SESSION ENDED at:    Sun Aug  8 13:08:03 1999
```

The above information caught my eye because someone logged in as "root" to this host and connected via ncftp to a remote machine that contained the last name of the person I suspect may have compromised this machine. I have replaced any revealing information about this person with "x's".

Unfortunately, he didn't appear to actually do anything suspicious. The other log files contained connections to Red Hat[20] and Apache.org[21] to download software updates.

[20] Red Hat produces a version of the Linux operating system, which is installed on this system. http://www.redhat.com.

[21] Apache.org makes a popular web server, typically installed by default on Linux systems. http://www.apache.org.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Next I'll search for files beginning with " ", another method crackers use, which I described earlier.

```
[root@localhost root]# find . -name "* *"
[root@localhost root]#
```

As seen above, I found no files or directories that contained empty spaces.

Finally, I'm going to peruse through the /dev directory, looking for anything out of the ordinary. As an experienced Linux system administrator, I have a good idea of what should and shouldn't be present in that directory.

```
[root@localhost dev]# find . -not -type b -not -type c -ls
  6121   35 drwxr-xr-x   5 root     root        34816 Mar 20 12:47 .
  6272    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./fb -> fb0
  6281    0 lrwxrwxrwx   1 root     root           15 Jul  3  1999 ./fd ->
../proc/self/fd
  6302    0 lrwxrwxrwx   1 root     root            4 Jul  3  1999 ./ftape -> rft0
257086   12 drwxrwxr-x   2 root     root        12288 Jul  3  1999 ./ida
  6582    0 lrwxrwxrwx   1 root     root            9 Jul  3  1999 ./isdnctrl ->
isdnctrl0
  6808    0 lrwxrwxrwx   1 root     root            5 Jul  3  1999 ./nftape -> nrft0
  6122    0 srw-------   1 root     root            0 Jun 24  2000 ./printer
 75526    1 drwxrwxr-x   2 root     root         1024 Feb 23  1999 ./pts
  7163    0 lrwxrwxrwx   1 root     root            4 Jul  3  1999 ./ramdisk -> ram0
 75527   33 drwxr-xr-x   2 root     root        32768 Jul  3  1999 ./rd
  7450    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./sg0 -> sga
  7451    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./sg1 -> sgb
  7452    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./sg2 -> sgc
  7453    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./sg3 -> sgd
  7454    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./sg4 -> sge
  7455    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./sg5 -> sgf
  7456    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./sg6 -> sgg
  7457    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./sg7 -> sgh
  7487    0 lrwxrwxrwx   1 root     root           17 Jul  3  1999 ./stderr ->
../proc/self/fd/2
  7488    0 lrwxrwxrwx   1 root     root           17 Jul  3  1999 ./stdin ->
../proc/self/fd/0
  7489    0 lrwxrwxrwx   1 root     root           17 Jul  3  1999 ./stdout ->
../proc/self/fd/1
  8448   27 -rwxr-xr-x   1 root     root        26450 Apr 17  1999 ./MAKEDEV
  8482    0 prw-------   1 root     root            0 Mar 20 12:47 ./initctl
  8504    0 lrwxrwxrwx   1 root     root            3 Jul  3  1999 ./cdrom -> hdc
```

The above command searched for anything that is not a character or block device. The output shows 3 directories and several symbolic links, but nothing suspicious or unexpected.

Users home directory

Even though I have done a thorough search for any suspicious files on the system, before I finish this part of my investigation, I want to check in the potential crackers home directory to see what files he has stored there.

```
[root@localhost cerwin]# ls -l
total 1218
drwxr-xr-x    2 501      501          1024 Oct 20  2000 arch
lrwxrwxrwx    1 501      501            22 Jul  5  1999 chaos -> /home/mudder/mud/chaos
-rw-r--r--    1 501      501          1831 Sep 27  1998 combinations.cc
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
-rw-r--r--    1 501      501        174853 Jul 19  1998 cvs.ps.gz
-rw-r--r--    1 501      501          2845 Jul 19  1998 cvs.txt
-rw-r--r--    1 501      501         15434 Jul 29  1998 cvsweb.tar.gz
-rw-r--r--    1 501      501         47104 Oct  3  2000 EL.doc
lrwxrwxrwx    1 501      501            16 Jul  5  1999 mud -> /home/mudder/mud
-rw-rw-r--    1 501      501         16725 Mar 18  1999 nedit-conf.tar.gz
-rw-rw-r--    1 501      501        583272 May  6  1999 nedit_os390-5.0.2.tar.gz
lrwxrwxrwx    1 501      501            20 Jul  5  1999 rom -> /home/mudder/mud/rom
drwxr-xr-x    3 501      501          1024 May  7  2001 shoutcast-1-8-0-linux-glibc6
-rw-r--r--    1 501      501        150722 May  7  2001 shoutcast-1-8-0-linux-
glibc6.tar.gz
drwxr-xr-x    4 501      501          1024 Oct 20  2000 src
-rw-rw-r--    1 501      501             0 Sep  8  1998 test
-rw-r--r--    1 501      501        236416 Jul 23  1997 tt++v1.64.tar.gz
lrwxrwxrwx    1 501      501            35 Jul  5  1999 web ->
/home/apache/share/htdocs/anastoria
-rwxrwxr-x    1 501      501           199 Oct 28  1998 wipe.csh
```

The chaos directory sounds ominous, but is in fact the host name of the machine, which in turn is the name of the MUD game they were developing.

Two files catch my eye: EL.doc and wipe.csh.

Using scp, a file transfer protocol included with the OpenSSH[22] package, I'll transfer EL.doc to a Windows machine, where I will first scan it using Norton Anti-virus, and then view it using Microsoft Word.

EL.doc turns out to be a completed employment application for Earthlink, Inc., a large and popular Internet Service Provider. The potential cracker we are investigating completed the application. Nothing ominous about this, although one wonders why it was located on this server.

Now to look at wipe.csh:

```
[root@localhost cerwin]# cat wipe.csh
#!/bin/csh

set DATE=$1
echo $DATE
if ( $DATE == "" ) then
        exit 1
endif

foreach i (*)
        set f=`/bin/ls -dl "$i" | grep " $DATE " | gawk '{print $9}'`
        if ( "$f" != "" ) then
            echo rm $f
        endif
end
```

On first glance, this program appears to show what files will be deleted from a given location based on their modification date. However, to be certain, I want to test it in a safe way.

[22] A suite of tools that allows for the encrypted transmission of data across a network. http://www.openssh.org/

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

To do this, I will create a test directory using a non-privileged account. I copied the contents of /usr/share/pixmaps, including subdirectories, into my test directory to insure I had a large number of real files to test against.

```
[root@localhost wipetest]# csh wipetest.csh Sep
Sep
rm accessibility-directory.png
rm accessibility-keyboard-capplet.png
rm advanced-directory.png
```

A quick check shows that the files still exist. So wipetest.csh is a small utility that shows what files will be deleted without actually removing them. While a cracker could potentially use this to delete files, it isn't SUID/SGID, and doesn't actually do anything malicious.

Deleted Files

It is a commonly held misconception that when you delete a file off a system it is gone for good. The truth is that what is removed is the "pointer" to the file, not the file itself.

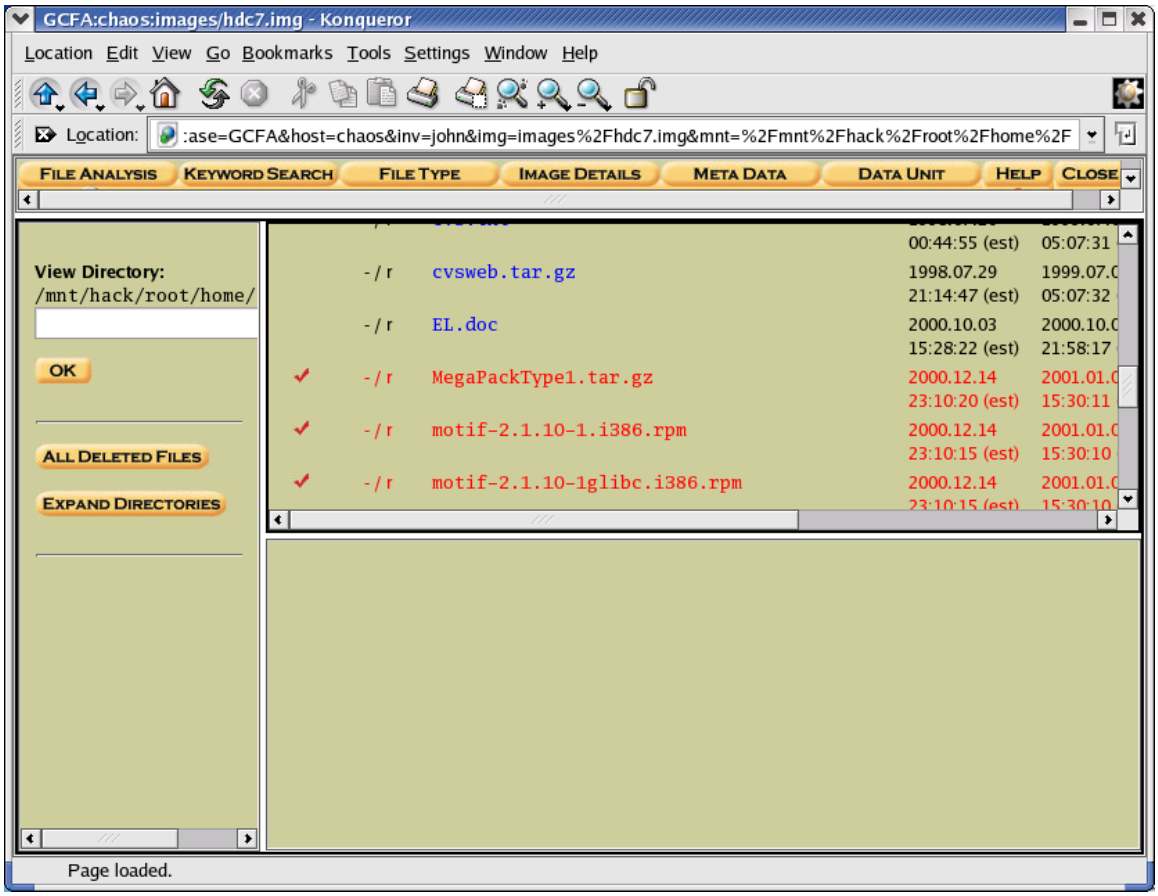
To put it simply, most file systems use inodes to act as pointers to files. An inode is like a number on the front of a PO Box at the post office. It tells you where to get a particular persons mail, which is contained inside. Even if you remove the label from the PO box, the mail is still inside until I new number is put there and new mail inserted, pushing out the old.

On a computer, the concept is very much the same. From a practical standpoint, this means that data that has been "deleted" will continue to reside on the disk until new data is written over top of it. With the proper tools, I can look for files that have been deleted but still exist and extract them for review. In technical terms, these files are said to be in "unallocated space" because the operating system no long has space "allocated", or set aside for them, but has not overwritten them either.

Unfortunately, experienced hackers know all about how this works, and simple means exist to "truly" delete files. This is accomplished by writing null data over the area of the disk you want to delete from. To stick with our example, this is like removing the PO box label and then pushing a bunch of junk mail into the box.

To help locate and recover deleted files, I'm going to go back to Autopsy, which has built in functionality to work with files in unallocated space, as described above.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



In the above screenshot, the files/directories shown in red are the “deleted”, or “unallocated” ones. Those in blue are normal. The red files can be recovered by Autopsy, as I will show below.

The first deleted file I came across that contained anything interesting was the xferlog.5. Although Autopsy gives me the ability to view this file within it’s own windows, I want to actually undelete it so that I can make a copy for future reference.

Since this is a text file, the “Export” command in Autopsy simply displays the contents of the file, which I can then “cut and paste” into my own file in the evidence directory I have created. If this had been a binary file, Autopsy would allow me to save it out in its raw format which I could then run tests against.

```
Wed May  9 20:20:12 2001 18 x.x.x 58061 /home/apache/htdocs/Chevy/crest-marking.jpg b _ i
r scott ftp 0 * c
Wed May  9 20:21:02 2001 46 x.x.x 145595 /home/apache/htdocs/Chevy/dash-drivers-
position.jpg b _ i r scott ftp 0 * c
Wed May  9 20:21:31 2001 25 x.x.x 79457 /home/apache/htdocs/Chevy/door-panel-wear.jpg b _
i r scott ftp 0 * c
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The above entries from that file, shown in Autopsy, indicate that another user, "scott", was transferring files not related to the development project to and from this server. In this case, they appear to be pictures of a car.

The remaining deleted files were of no interest.

Timeline

(The complete time line can be found in Appendix E.)

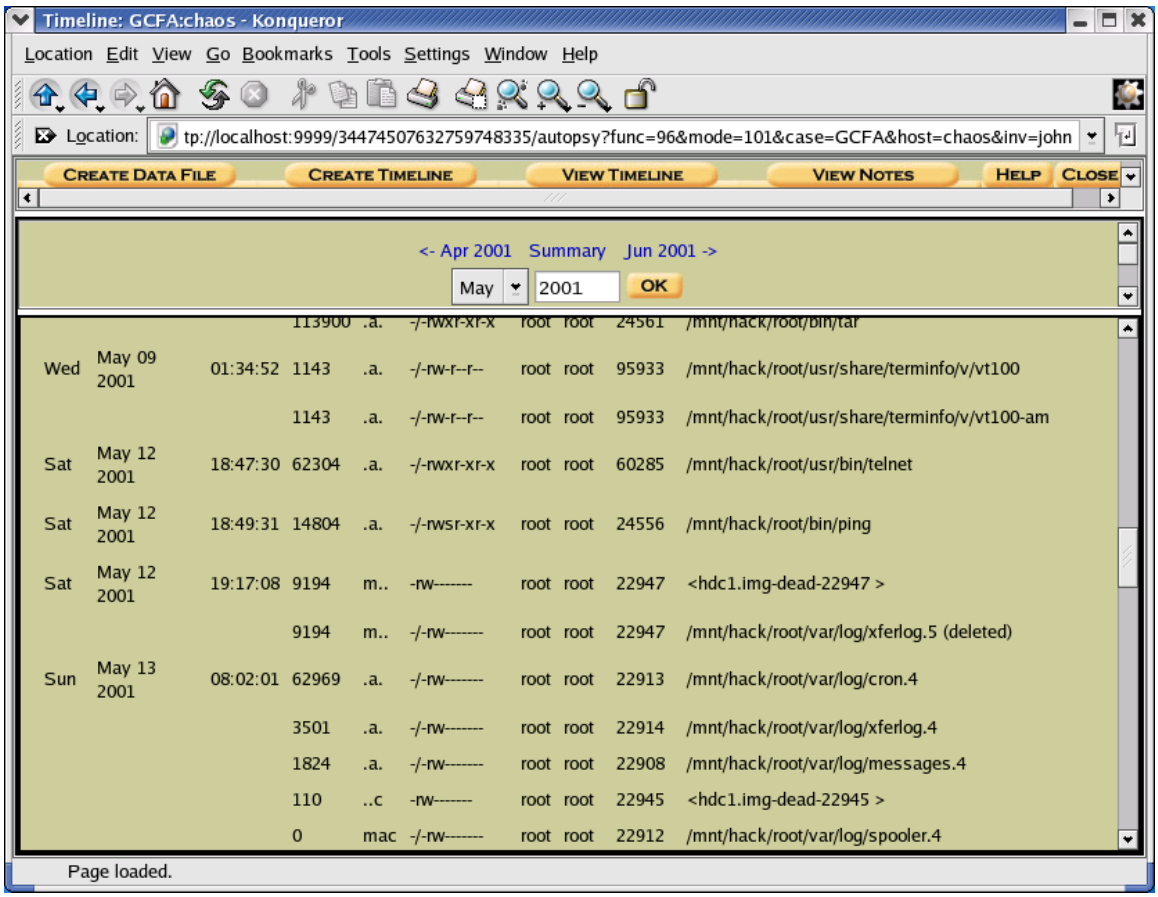
An important step in any investigation is establishing a time line of events. A valid and comprehensive time line:

- Helps trace the path the cracker has taken during his/her time on the system
- Provide a means by which evidence collected from the system can be correlated to evidence collected from other sources
- Can show what files were modified since the time the system was originally installed, thereby indicating possibly compromised binaries.

Task/Autopsy has a built in Time Line Analysis tool that I am going to run on my image.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.



Armed with this data, I want to first determine when this machine was installed so that I can set my baseline.

Browsing through the time line, I come across April 17th, 1999, and July 3rd, 1999, where there are thousands of files being created on the system. Based on this date, it appears that the system was originally installed on April 19th, and then upgraded to its current state on July 3rd. Further analysis of the time line showed that the system was last used on March 30th, 2003, a fact I had already determined from the log files, and confirmed with the owner of the system.

Since most of the system files were modified by the upgrade on July 3rd, I want to search for any files that were modified since that date. In particular, I want to look for binaries or system configuration files, since these are the most likely to be used in any type of compromise.

I'll start by searching for files that are owned by the username of the potential cracker, "cerwin".

```
[root@localhost root]# find . -type f -user 501 |sort -nr
[root@localhost root]#
```

"501" is the user id of "cerwin" in the systems /etc/passwd file.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

The good news is that “cerwin” doesn’t appear to own any system files. However, given the systems lackluster security state, it is quite likely he could have had access to the “root” account.

Therefore, I’ll build a list of executable programs, owned by “root”, that have been modified since July 3rd. To sort through those files properly, I’ll need to know what the modification time for each file is, so I’ll add some parameters to find that will help accomplish that.

```
find . -type f -user root -perm +111 -printf "%TY%Tm%Td%TH%TM%TS %h/%f/\n" |sort -nr
20010530131816 ./etc/rc.d/rc.local/
20010408054656 ./etc/rc.d/init.d/lpd/
20010311085234 ./etc/rc.d/init.d/rstatd/
20010311085234 ./etc/rc.d/init.d/nfs/
20010311083120 ./usr/bin/mingetty/
20010311083120 ./etc/rc.d/rc.sysinit/
20010127055350 ./usr/sbin/ndc/
20010127055350 ./usr/sbin/named-xfer/
20010127055350 ./usr/sbin/named/
20010127055350 ./usr/sbin/irpd/
20010127055350 ./usr/sbin/dnskeygen/
20010127055348 ./usr/sbin/named-bootconf/
20010127055348 ./etc/rc.d/init.d/named/
```

Not a very large list. There are some modifications to the startup scripts in /etc/rc.d/init.d, one of which is lpd. During an earlier stage of the analysis, I found lpd to be non-executable due to large numbers of syntax errors.

An examination of rstatd, nfs, and named doesn’t reveal anything malicious. I have gathered other evidence that suggests NFS was running on this system at some point, so only the modification to named seems out of the ordinary. Once again, I check with the owner of system, and he confirmed that he had been “playing around” with running a DNS server on this system during the timeframe indicated above, so the results are expected.

Also, I see that irpd was installed on the system. This binary isn’t familiar to me, but a quick Google search indicates that it is an Internet Radio Protocol daemon.[23]

This daemon is used to stream MP3 audio files across the network. My earlier examination of the log files showed me that “cerwin” was uploading MP3’s to this server. The presence of irpd also suggests that he may have been streaming this audio out to others. This is a violation of the systems intended purpose and no doubt something the owner will want to watch out for in the future.

I also need to check to see what files “cerwin” might own on the remainder of the system.

[23] http://www.linux.org/apps/AppId_7453.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
[root@localhost root]# find . -type f -user 501 -printf '%t\t%u\t%h%f\n' |sort -nr
```

This command is similar to the previous find that I ran, except that in this case, I have specified a particular output format that will show me the dates that the found files were last modified along with the username and filename.

The command returned a list of 1110 files within the /home directory, nearly all of which are located within a web server directory.

Besides that I have already mentioned above, the time line tells me a few other things:

- The earliest file modified by "cerwin" on the system was on August 24th, 1999, roughly one month after the server was updated to it's current state. This information is corroborated by the system owner. Interesting enough, these initial files appear to related to the actual MUD development. Every subsequent file does not appear to be related to MUD development.
- The last modified file on the system by "cerwin" was June 5th, 2001, also corroborated by the system owner.
- Over the course of nearly 2 years, "cerwin" uploaded hundreds of personal images to the server into a directory that was accessible to the web server, suggesting that his photos were serving as some form of personal web site.
- As mentioned above, he also used the server for storing and possibly playing music files across the Internet.
- None of the files owned by "cerwin" appear to be malicious

Conclusions and Recommendations

When I first received the system to be analyzed, I didn't have a clear direction as to what may have been the issue. Instead, I was simply told there was the "chance" the system had been compromised that the owner wanted to insure that no such compromise had occurred. Because a trusted backup was not available, simply restoring the needed data onto a clean system was not an option. The system also suffered from poor system administration that resulted in several possible security problems. This meant that virtually any file on the system could be compromised, including the game source code.

To achieve a thorough analysis:

- I properly cataloged and tagged the hardware
- I identified the type of system I was working on so that I would have a better idea of what types of files to look for and check.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

- I analyzed system and log files looking for any evidence of malicious activity.
- I searched for hidden directories and compromised binaries
- I traced the users activities
- I looked for and analyzed files that had been deleted from the system, but that were still accessible.

Through all this, I discovered some interesting facts:

- I couldn't find any evidence that any known cracker tools had been installed or used on the system.
- I learned that the system had not been properly administrated, either from an operational or security standpoint. This was made evident by the fact that the "cerwin" user account had not yet been removed, and by various permission and syntax errors located throughout the system.
- I learned that "cerwin" had used the system for transferring and possibly streaming MP3 audio files, something that was not the intended purpose of the system.

Based on what I found, I returned the system to its owner along with the following recommendations:

- Despite the fact that no serious malicious activity had been found, the system was in a questionable state of operability and should therefore be replaced.
    - The MUD source code should be carefully reviewed by a trusted party(s) to insure that no malicious code exists.
    - All proprietary data should be backed up to storage media pending changes to the system.
    - The hardware, though functional, was quite old by computing standards, and serious improvements in performance could be gained by upgrading.
    - The existing operating system should be replaced with the most recent version, and all patches and updates should be applied on a regular basis to insure maximum possible security, functionality and performance.
    - A formal security and acceptable use policy should be written and provided to every user on the system. This policy should clearly outline the procedures necessary to properly add and remove user accounts, and should set up and clearly defined access policy for data on the system.
    - All Internet accessible ports should include "bannered", meaning that a message should be displayed to users informing them of their rights, or lack thereof, on the system.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

- It is clear that “cerwin” was not using the server for much more then his own personal gain, and booting him from the team was an appropriate measure.

Finally, this analysis proved that collecting evidence from a system is both challenging and time-consuming, but that it can identify issues that are not immediately obvious. Because said issues could have serious potential consequences for the operation and security of the system, a forensic analysis is well worth the time and effort.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

During the course of my normal duties as a system administrator at a regional ISP, I received a phone call from someone identifying himself as Special Agent John Doe of the Federal Bureau of Investigation (FBI.) I properly confirmed his identity before proceeding with any discussion.

Agent Doe informed me that an account on a system I am responsible for was used to crack into a government computer. He then asked me to analyze my log files in order to corroborate his information, and to provide him with any additional information I might be able to garner.

18 U.S.C. §1030, The Computer Fraud and Abuse Act[24], criminalizes the act of causing damage to a "protected computer". A "protected computer" is defined as "any U.S. government network, those used by banks and other financial institutions, and other networks, domestic or foreign, that affect interstate or foreign commerce of communication of the United States" (18. U.S.C. § 1030(e)(2)).

Since I have been informed that a government, and therefore "protected", system has been attacked, this Act clearly applies.

The Wiretap Act (18 U.S.C. §2511)[25] puts restrictions on what information can be monitored, selected, or shared. Generally speaking, the Wiretap Act prohibits the intentional interception, use, or disclosure of wire and electronic communications unless a statutory exception applies. (18 U.S.C. § 2511(1). These statutory exceptions cover a wide range of possibilities, but in this case, the "provider exception" (18 U.S.C. § 2511(2)(a)(i) applies. This exception gives me, the provider, the ability to conduct reasonable activities in order to protect my rights or property. Since it appears that my system was used in the commission of an illegal act, and since I want to protect my company and system from any future liability or damage, I am within my rights to examine the log files.

The Electronic Communications Privacy Act (ECPA), 18 U.S.C. § 2701-12 also has bearing on what I can provide to Agent Doe. Generally, the ECPA provides privacy rights for customers of and subscribers of computer network service providers. An ISP is considered a "network service provider", and my company offers its services to the public, an important factor in determining what statutes may apply.

Log files are part of a category of information defined in the ECPA as "Stored Communications Associated Data." Whether or not I can share this information is addressed directly by 18 U.S.C. § 2702(b)(5) which reads: [the disclosure] "may be necessarily incident to the rendition of the service or to the protection of

[24] http://cio.doe.gov/Documents/CFA.HTM
[25] http://cio.doe.gov/Documents/ECPA.HTM

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

the rights or property of the provider of that service." My company is the provider of the service, and if someone is using our network to conduct criminal acts, then we have a legitimate right to take reasonable measures to deal with that. Unfortunately, none of this adds up to much in this case, because according to my log files, no malicious activity has taken place. A valid user account was on during that time, but I have no knowledge of what the user did, legal or otherwise. Without further legal authorization, I am unable to tell Agent Doe anything other then the fact that a user was on.

In October of 2001, following the tragic terrorist attack on September of that same year, Congress passed the US Patriot Act[26], which greatly expands the powers that law enforcement agencies have when it comes to computer crime and surveillance of electronic communications. Relevant to this issue:

- "it allows ISPs to voluntarily hand over all "non-content" information to law enforcement with no need for any court order or subpoena. sec. 212."[27]

What this means is that if I so choose, I can confirm to Agent Doe that the incident occurred, and any other non-subscriber information. An example of information I can't provide would be an email message.

Agent Doe accepts that answer, but wishes to explore this further. To do so, he'll need to get a court order, or possibly just a subpoena to gather more data from my system. Of particular interest to him are what the ECPA refers to as "Stored Content of Communications", which constitutes e-mail, IRC logs, and other types of text-based communications that may be stored on my system.

So that evidence is not lost while the court order is obtained, Agent Doe requests that I make a bit copy of the current log files and e-mail folder of the user account. Maintaining backup copies is well within my rights as a service provider, and can be done without viewing the content, thereby avoiding privacy issues. To help insure legality, I will request that Agent Doe send me a preservations letter, which is allowed under the ECPA, §2703(f). This letter is a formal request that I hold on to relevant evidence for at least 90 days, thereby giving the Agent adequate time to get the court documents he needs to proceed further.

Since I have no reason not to cooperate, I make the copies onto a CD-R and store it in a secure location until Agent Doe contacts me again.

Assuming Agent Doe obtains the proper court order compelling me to give him all the information I have, I can provide it to him knowing I have complied completely with the law.

---

[26] http://www.eff.org/Privacy/Surveillance/Terrorism_militias/hr3162.php
[27] http://www.eff.org/Privacy/Surveillance/Terrorism_militias/20011031_eff_usa_patriot_analysis.html

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

What if I had looked at my logs and discovered that someone had gained unauthorized access to my system, and then used it to attack the government computer?

Under these circumstances, several important things change.

First of all, unauthorized access is covered under 18 U.S.C. § 1030, the Computer Fraud and Abuse Act and in my case, Virginia Code (V.C) § 18.2-152.4 (See Appendix D). This means that my rights have been violated and I have increased power to take protective and investigative actions[28].

Since I now have concerns about this user account, I'm going to take some time to review other sources of information I have at my disposal. These include:

- Router logs. Combined with subscriber logs that will give me the user's IP address at the time of the incident, I can gather information about what sites the user may have visited.
- If Agent Doe has provided me with an address of the system that was attacked, I can place a filter on the Intrusion Detection System (IDS) to alert me if that address appears again.
- Collecting and reviewing information from my IDS, host or network based.

Under the terms of the Provider Exception to the Wiretap Act, mentioned above, I have full authority to investigate the activities of this user account in order to protect myself.

However, because the exception to the Wiretap Act applies only to me and not the government, Agent Doe will still need to obtain a court order to get the information I have collected[29].

Additional Information and Conclusions

When a computer crime occurs, there are 4 major Acts that a system or security administrator must consider before taking action:

- The Computer Fraud and Abuse Act, 18 U.S.C. §1030
- The Wiretap Act, 18 U.S.C. §2511
- The Electronic Communications Privacy Act, 18 U.S.C. 2701
- The U.S. Patriot Act

[28] United States v. Mullins, 992 F.2d 1472, 1478 (9th Cir. 1993)
[29] United States v. McLaren, 957 F. Supp. 215, 219 (M.D. Fla. 1997)

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Combined, these 4 Acts try to strike a balance between protecting the rights of the network owner, protecting the privacy of the network user, and giving law enforcement sufficient power to investigate and prosecute.

It is equally important for administrators to understand their state laws as well, which while often similar, may provide additional restrictions on what actions one can take, and what avenues of prosecution are available.

When available, system administrators should always consult corporate legal consul before communicating with law enforcement. Also, it is important to clearly outline corporate policy on reporting security incidents. Many organizations may wish to avoid the publicity that comes with reporting. However, it is important to remember that law enforcement generally will work as hard as they can to protect the identity of a compromised company. Failure to prosecute may result in additional attacks because crackers will learn that their actions will not have any serious consequences.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix A: Strace output of atd and lokid

Key identifying lines have been highlighted in red.

Atd:

```
1322  execve("./atd", ["./atd"], [/* 21 vars */]) = 0
1322  old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x40007000
1322  mprotect(0x40000000, 21772, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
1322  mprotect(0x8048000, 13604, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
1322  stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=56082, ...}) = 0
1322  open("/etc/ld.so.cache", O_RDONLY) = 3
1322  old_mmap(NULL, 56082, PROT_READ, MAP_SHARED, 3, 0) = 0x40008000
1322  close(3)                          = 0
1322  stat("/etc/ld.so.preload", 0xbffffae0) = -1 ENOENT (No such file or direct
ory)
1322  open("/usr/i486-linux-libc5/lib/libc.so.5", O_RDONLY) = 3
1322  read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0(k\1\000"..., 4096
) = 4096
1322  old_mmap(NULL, 823296, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40
016000
1322  old_mmap(0x40016000, 592037, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 3
, 0) = 0x40016000
1322  old_mmap(0x400a7000, 23728, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3
, 0x90000) = 0x400a7000
1322  old_mmap(0x400ad000, 201876, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|M
AP_ANONYMOUS, -1, 0) = 0x400ad000
1322  close(3)                          = 0
1322  mprotect(0x40016000, 592037, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
1322  munmap(0x40008000, 56082)         = 0
1322  mprotect(0x8048000, 13604, PROT_READ|PROT_EXEC) = 0
1322  mprotect(0x40016000, 592037, PROT_READ|PROT_EXEC) = 0
1322  mprotect(0x40000000, 21772, PROT_READ|PROT_EXEC) = 0
1322  personality(0 /* PER_??? */)      = 0
1322  geteuid()                         = 0
1322  getuid()                          = 0
1322  getgid()                          = 0
1322  getegid()                         = 0
1322  geteuid()                         = 0
1322  getuid()                          = 0
1322  brk(0x804c818)                    = 0x804c818
1322  brk(0x804d000)                    = 0x804d000
1322  open("/usr/share/locale/en_US.UTF-8/LC_MESSAGES", O_RDONLY) = -1 ENOENT (N
o such file or directory)
1322  stat("/etc/locale/C/libc.cat", 0xbffff604) = -1 ENOENT (No such file or di
rectory)
1322  stat("/usr/lib/locale/C/libc.cat", 0xbffff604) = -1 ENOENT (No such file o
r directory)
1322  stat("/usr/lib/locale/libc/C", 0xbffff604) = -1 ENOENT (No such file or di
rectory)
1322  stat("/usr/share/locale/C/libc.cat", 0xbffff604) = -1 ENOENT (No such file
 or directory)
1322  stat("/usr/local/share/locale/C/libc.cat", 0xbffff604) = -1 ENOENT (No suc
h file or directory)
1322  socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
1322  sigaction(SIGUSR1, {0x804a6b0, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SI
G_DFL}, 0x42028c48) = 0
1322  socket(PF_INET, SOCK_RAW, IPPROTO_RAW) = 4
1322  setsockopt(4, SOL_IP, IP_HDRINCL, [1], 4) = 0
1322  getpid()                          = 1322
1322  getpid()                          = 1322
1322  shmget(1564, 240, IPC_CREAT|0)    = 0
1322  semget(1746, 1, IPC_CREAT|0x180|0600) = 0
1322  shmat(0, 0, 0)                    = 0x40008000
1322  write(2, "\nLOKI2\troute [(c) 1997 guild cor"..., 52) = 52
1322  time([1049209617])                = 1049209617
1322  close(0)                          = 0
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
1322  sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}, 0x42028c48) = 0
1322  sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}, 0x42028c48) = 0
1322  sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}, 0x42028c48) = 0
1322  fork()                            = 1323
1322  close(4 <unfinished ...>
1323  --- SIGSTOP (Stopped (signal)) ---
1322  <... close resumed> )             = 0
1323  setsid( <unfinished ...>
1322  close(3 <unfinished ...>
1323  <... setsid resumed> )            = 1323
1322  <... close resumed> )             = 0
1323  open("/dev/tty", O_RDWR)          = -1 ENXIO (No such device or address)
1322  semop(0, 0xbffffa7c, 2 <unfinished ...>
1323  chdir("/tmp" <unfinished ...>
1322  <... semop resumed> )             = 0
1323  <... chdir resumed> )             = 0
1322  shmdt(0x40008000 <unfinished ...>
1323  umask(0 <unfinished ...>
1322  <... shmdt resumed> )             = 0
1323  <... umask resumed> )             = 022
1322  semop(0, 0xbffffa7c, 1 <unfinished ...>
1323  sigaction(SIGALRM, {0x8049218, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT},  <u
nfinished ...>
1322  <... semop resumed> )             = 0
1322  _exit(0)                          = ?
1323  <... sigaction resumed> {SIG_DFL}, 0x42028c48) = 0
1323  alarm(3600)                       = 0
1323  sigaction(SIGCHLD, {0x8049900, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SI
G_DFL}, 0x42028c48) = 0
1323  read(3,
```

Lokid:

```
497   execve("./lokid", ["./lokid"], [/* 17 vars */]) = 0
497   mmap(0, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x
40006000
497   mprotect(0x8048000, 13972, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
497   stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=4420, ...}) = 0
497   open("/etc/ld.so.cache", O_RDONLY) = 4
497   mmap(0, 4420, PROT_READ, MAP_SHARED, 4, 0) = 0x40007000
497   close(4)                          = 0
497   open("/lib/libc.so.5.3.12", O_RDONLY) = 4
497   read(4, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3"..., 4096) = 4096
497   mmap(0, 831488, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40009000
497   mmap(0x40009000, 599414, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0)
 = 0x40009000
497   mmap(0x4009c000, 22884, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 4, 0x
92000) = 0x4009c000
497   mmap(0x400a2000, 200952, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_A
NONYMOUS, -1, 0) = 0x400a2000
497   close(4)                          = 0
497   mprotect(0x40009000, 599414, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
497   munmap(0x40007000, 4420)          = 0
497   mprotect(0x8048000, 13972, PROT_READ|PROT_EXEC) = 0
497   mprotect(0x40009000, 599414, PROT_READ|PROT_EXEC) = 0
497   personality(PER_LINUX)            = 0
497   geteuid()                         = 0
497   getuid()                          = 0
497   brk(0x804c988)                    = 0x804c988
497   brk(0x804d000)                    = 0x804d000
497   open("/usr/share/locale/C/LC_MESSAGES", O_RDONLY) = -1 ENOENT (No such fil
e or directory)
497   stat("/etc/locale/C/libc.cat", 0xbffff88c) = -1 ENOENT (No such file or di
rectory)
497   stat("/usr/lib/locale/C/libc.cat", 0xbffff88c) = -1 ENOENT (No such file o
r directory)
497   stat("/usr/lib/locale/libc/C", 0xbffff88c) = -1 ENOENT (No such file or di
rectory)
497   stat("/usr/share/locale/C/libc.cat", 0xbffff88c) = -1 ENOENT (No such file
```


© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
 or directory)
497   stat("/usr/local/share/locale/C/libc.cat", 0xbffff88c) = -1 ENOENT (No suc
h file or directory)
497   socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 4
497   sigaction(SIGUSR1, {0x804a820, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SI
G_DFL}) = 0
497   socket(PF_INET, SOCK_RAW, IPPROTO_RAW) = 5
497   setsockopt(5, IPPROTO_IP3, [1], 4) = 0
497   getpid()                          = 497
497   getpid()                          = 497
497   shmget(739, 240, IPC_CREAT|0)     = 2
497   semget(921, 1, IPC_CREAT|0x180|0600) = 2
497   shmat(2, 0, 0)                    = 0x40007000
497   write(2, "\nLOKI2\troute [(c) 1997 guild c"..., 52) = 52
497   time([1049208640])                = 1049208640
497   close(0)                          = 0
497   sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}) = 0
497   sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}) = 0
497   sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}) = 0
497   fork()                            = 498
497   close(5)                          = 0
497   close(4)                          = 0
497   semop(0x2, 0x2, 0, 0xbffffd08)    = 0
497   shmdt(0x40007000)                 = 0
497   semop(0x2, 0x1, 0, 0xbffffd08)    = 0
497   _exit(0)                          = ?
498   setsid()                          = 498
498   open("/dev/tty", O_RDWR)          = -1 ENXIO (No such device or address)
498   chdir("/tmp")                     = 0
498   umask(0)                          = 022
498   sigaction(SIGALRM, {0x8049290, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SI
G_DFL}) = 0
498   alarm(3600)                       = 0
498   sigaction(SIGCHLD, {0x80499f0, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SI
G_DFL}) = 0
498   read(4,
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix B: List of SUID, SGID files.

```
[root@localhost root]# find /mnt/hack/root \( -perm -004000 -o -perm -002000 \)
-type f -ls
 24536   14 -rwsr-xr-x   1 root     root        13208 Apr 13  1999 /mnt/hack/root/bin/su
 24547   53 -rwsr-xr-x   1 root     root        52788 Apr 17  1999 /mnt/hack/root/bin/mount
 24548   27 -rwsr-xr-x   1 root     root        26508 Apr 17  1999 /mnt/hack/root/bin/umount
 24556   16 -rwsr-xr-x   1 root     root        14804 Apr  7  1999 /mnt/hack/root/bin/ping
 24567   21 -rwsr-xr-x   1 root     root        20164 Apr 17  1999 /mnt/hack/root/bin/login
 42864    4 -rwxr-sr-x   1 root     root         3860 Apr 19  1999 /mnt/hack/root/sbin/netreport
 42876   11 -rwsr-xr-x   1 root     root        10708 Apr 19  1999 /mnt/hack/root/sbin/cardctl
 42887   47 -r-sr-xr-x   1 root     root        46472 Apr 17  1999 /mnt/hack/root/sbin/pwdb_chkpwd
 59206   34 -rwsr-xr-x   1 root     root        33120 Mar 21  1999 /mnt/hack/root/usr/bin/at
 59332   31 -rwsr-xr-x   1 root     root        30560 Apr 15  1999 /mnt/hack/root/usr/bin/chage
 59334   30 -rwsr-xr-x   1 root     root        29492 Apr 15  1999 /mnt/hack/root/usr/bin/gpasswd
 59523   66 ---x--s--x   1 root     games       66284 Apr 10  1999 /mnt/hack/root/usr/bin/gnibbles
 59524   29 ---x--s--x   1 root     games       27736 Apr 10  1999 /mnt/hack/root/usr/bin/gnobots
 59525   75 ---x--s--x   1 root     games       74812 Apr 10  1999 /mnt/hack/root/usr/bin/gnobots2
 59526   51 ---x--s--x   1 root     games       50864 Apr 10  1999 /mnt/hack/root/usr/bin/gnome-
stones
 59527   71 ---x--s--x   1 root     games       70880 Apr 10  1999 /mnt/hack/root/usr/bin/gnomine
 59528   26 ---x--s--x   1 root     games       25556 Apr 10  1999
/mnt/hack/root/usr/bin/gnotravex
 59529  227 ---x--s--x   1 root     games      230584 Apr 10  1999 /mnt/hack/root/usr/bin/gtali
 59530   25 ---x--s--x   1 root     games       23676 Apr 10  1999 /mnt/hack/root/usr/bin/gturing
 59531   47 ---x--s--x   1 root     games       46492 Apr 10  1999 /mnt/hack/root/usr/bin/iagno
 59532   40 ---x--s--x   1 root     games       39268 Apr 10  1999 /mnt/hack/root/usr/bin/mahjongg
 59533   22 ---x--s--x   1 root     games       21076 Apr 10  1999 /mnt/hack/root/usr/bin/same-
gnome
 59583    4 -rwsr-xr-x   1 root     root         3208 Mar 22  1999 /mnt/hack/root/usr/bin/disable-
paste
 59974   17 -r-sr-sr-x   1 root     lp          15816 Mar 22  1999 /mnt/hack/root/usr/bin/lpq
 59975   17 -r-sr-sr-x   1 root     lp          15768 Mar 22  1999 /mnt/hack/root/usr/bin/lpr
 59976   17 -r-sr-sr-x   1 root     lp          16216 Mar 22  1999 /mnt/hack/root/usr/bin/lprm
 59985   33 -rwxr-sr-x   1 root     man         32320 Apr  9  1999 /mnt/hack/root/usr/bin/man
 60139   11 -r-s--x--x   1 root     root        10704 Apr 14  1999 /mnt/hack/root/usr/bin/passwd
 60155  509 -rws--x--x   2 root     root       517916 Apr  6  1999 /mnt/hack/root/usr/bin/suidperl
 60155  509 -rws--x--x   2 root     root       517916 Apr  6  1999
/mnt/hack/root/usr/bin/sperl5.00503
 60171   12 -rwxr-sr-x   1 root     mail        11432 Apr  6  1999 /mnt/hack/root/usr/bin/lockfile
 60173   64 -rwsr-sr-x   1 root     mail        64468 Apr  6  1999 /mnt/hack/root/usr/bin/procmail
 60209   15 -rwsr-xr-x   1 root     root        14036 Apr 15  1999 /mnt/hack/root/usr/bin/rcp
 60211   11 -rwsr-xr-x   1 root     root        10516 Apr 15  1999 /mnt/hack/root/usr/bin/rlogin
 60212    8 -rwsr-xr-x   1 root     root         7780 Apr 15  1999 /mnt/hack/root/usr/bin/rsh
 60252   16 -rwxr-sr-x   1 root     slocate     15032 Apr 19  1999 /mnt/hack/root/usr/bin/slocate
 60277    7 -r-xr-sr-x   1 root     tty          6212 Apr 17  1999 /mnt/hack/root/usr/bin/wall
 60325   15 -rws--x--x   1 root     root        14088 Apr 17  1999 /mnt/hack/root/usr/bin/chfn
 60326   15 -rws--x--x   1 root     root        13800 Apr 17  1999 /mnt/hack/root/usr/bin/chsh
 60342    6 -rws--x--x   1 root     root         5576 Apr 17  1999 /mnt/hack/root/usr/bin/newgrp
 60352    9 -rwxr-sr-x   1 root     tty          8392 Apr 17  1999 /mnt/hack/root/usr/bin/write
 60357   22 -rwsr-xr-x   1 root     root        20920 Apr 14  1999 /mnt/hack/root/usr/bin/crontab
142838   17 -rwxr-sr-x   1 root     102         15523 Apr  8  1999
/mnt/hack/root/usr/sbin/utempter
142839    9 -rwxr-sr-x   1 root     102          8376 Apr 15  1999 /mnt/hack/root/usr/sbin/gnome-
pty-helper
142841    6 -rwsr-xr-x   1 root     root         5736 Apr 19  1999
/mnt/hack/root/usr/sbin/usernetctl
142853   25 -rwxr-sr-x   1 root     lp          24104 Mar 22  1999 /mnt/hack/root/usr/sbin/lpc
142906  296 -rwsr-sr-x   1 root     root       299364 Apr 19  1999
/mnt/hack/root/usr/sbin/sendmail
142927   18 -rwsr-xr-x   1 root     bin         16488 Mar 22  1999
/mnt/hack/root/usr/sbin/traceroute
142931   11 -rwsr-xr-x   1 root     root        10708 Apr 12  1999
/mnt/hack/root/usr/sbin/userhelper
177482   35 -rwsr-xr-x   1 root     root        34131 Apr 16  1999
/mnt/hack/root/usr/libexec/pt_chown
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix C: Output of chkrootkit command.

```
[root@localhost chkrootkit-0.39a]# ./chkrootkit -r /mnt/hack/root
ROOTDIR is `/mnt/hack/root/'
Checking `amd'... not found
Checking `basename'... not infected
Checking `biff'... not found
Checking `chfn'... not infected
Checking `chsh'... not infected
Checking `cron'... not infected
Checking `date'... not infected
Checking `du'... not infected
Checking `dirname'... not infected
Checking `echo'... not infected
Checking `egrep'... not infected
Checking `env'... not infected
Checking `find'... not infected
Checking `fingerd'... not infected
Checking `gpm'... not infected
Checking `grep'... not infected
Checking `hdparm'... not infected
Checking `su'... not infected
Checking `ifconfig'... not infected
Checking `inetd'... not infected
Checking `inetdconf'... not infected
Checking `identd'... not infected
Checking `killall'... not infected
Checking `ldsopreload'... not infected
Checking `login'... not infected
Checking `ls'... not infected
Checking `lsof'... not infected
Checking `mail'... not infected
Checking `mingetty'... not infected
Checking `netstat'... not infected
Checking `named'... not infected
Checking `passwd'... not infected
Checking `pidof'... not infected
Checking `pop2'... not found
Checking `pop3'... not found
Checking `ps'... not infected
Checking `pstree'... not infected
Checking `rpcinfo'... not infected
Checking `rlogind'... not infected
Checking `rshd'... not infected
Checking `slogin'... not found
Checking `sendmail'... not infected
Checking `sshd'... not found
Checking `syslogd'... not infected
Checking `tar'... not infected
Checking `tcpd'... /usr/bin/strings: /mnt/hack/root//mnt/hack/root/usr/sbin/tcpd: No such
file or directory
not infected
Checking `tcpdump'... not infected
Checking `top'... not infected
Checking `telnetd'... not infected
Checking `timed'... not infected
Checking `traceroute'... not infected
Checking `w'... not infected
Checking `write'... not infected
Checking `aliens'... no suspect files
Searching for sniffer's logs, it may take a while... nothing found
Searching for HiDrootkit's default dir... nothing found
Searching for t0rn's default files and dirs... nothing found
Searching for t0rn's v8 defaults... nothing found
Searching for Lion Worm default files and dirs... nothing found
Searching for RSHA's default files and dir... nothing found
Searching for RH-Sharpe's default files... nothing found
Searching for Ambient's rootkit (ark) default files and dirs... nothing found
Searching for suspicious files and dirs, it may take a while...
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

```
/mnt/hack/root/usr/lib/git/.gitrc.aixterm /mnt/hack/root/usr/lib/git/.gitrc.common
/mnt/hack/root/usr/lib/git/.gitrc.hft /mnt/hack/root/usr/lib/git/.gitrc.hpterm
/mnt/hack/root/usr/lib/git/.gitrc.hp /mnt/hack/root/usr/lib/git/.gitrc.iris-ansi-net
/mnt/hack/root/usr/lib/git/.gitrc.ansi /mnt/hack/root/usr/lib/git/.gitrc.iris-ansi
/mnt/hack/root/usr/lib/git/.gitrc.linux /mnt/hack/root/usr/lib/git/.gitrc.console
/mnt/hack/root/usr/lib/git/.gitrc.mach /mnt/hack/root/usr/lib/git/.gitrc.minix
/mnt/hack/root/usr/lib/git/.gitrc.sun-cmd /mnt/hack/root/usr/lib/git/.gitrc.eterm
/mnt/hack/root/usr/lib/git/.gitrc.generic /mnt/hack/root/usr/lib/git/.gitrc.pc3
/mnt/hack/root/usr/lib/git/.gitrc.sun /mnt/hack/root/usr/lib/git/.gitrc.thix
/mnt/hack/root/usr/lib/git/.gitrc.vt102 /mnt/hack/root/usr/lib/git/.gitrc.vt420
/mnt/hack/root/usr/lib/git/.gitrc.screen /mnt/hack/root/usr/lib/git/.gitrc.vt100
/mnt/hack/root/usr/lib/git/.gitrc.vt125 /mnt/hack/root/usr/lib/git/.gitrc.vt200
/mnt/hack/root/usr/lib/git/.gitrc.vt201 /mnt/hack/root/usr/lib/git/.gitrc.vt220
/mnt/hack/root/usr/lib/git/.gitrc.vt240 /mnt/hack/root/usr/lib/git/.gitrc.vt300
/mnt/hack/root/usr/lib/git/.gitrc.vt320 /mnt/hack/root/usr/lib/git/.gitrc.vt400
/mnt/hack/root/usr/lib/git/.gitrc.xterm-color /mnt/hack/root/usr/lib/git/.gitrc.dtterm
/mnt/hack/root/usr/lib/git/.gitrc.xterms /mnt/hack/root/usr/lib/git/.gitrc.xterm
/mnt/hack/root/usr/lib/linuxconf/install/gnome/.directory
/mnt/hack/root/usr/lib/linuxconf/install/gnome/.order
/mnt/hack/root/usr/lib/perl5/5.00503/i386-linux/.packlist
/mnt/hack/root/usr/lib/perl5/site_perl/5.005/i386-linux/auto/MD5/.packlist
/mnt/hack/root/lib/modules/2.2.5-15/.rhkmvtag

Searching for LPD Worm files and dirs... nothing found
Searching for Ramen Worm files and dirs... nothing found
Searching for Maniac files and dirs... nothing found
Searching for RK17 files and dirs... nothing found
Searching for Ducoci rootkit... nothing found
Searching for Adore Worm... nothing found
Searching for ShitC Worm... nothing found
Searching for Omega Worm... nothing found
Searching for Sadmind/IIS Worm... nothing found
Searching for MonKit... nothing found
Searching for Showtee... nothing found
Searching for OpticKit... nothing found
Searching for T.R.K... nothing found
Searching for Mithra... nothing found
Searching for LOC rootkit ... nothing found
Searching for Romanian rootkit ... nothing found
Searching for anomalies in shell history files... nothing found
Checking `asp'... not infected
Checking `bindshell'... not tested
Checking `lkm'... not tested
Checking `rexedcs'... not found
Checking `sniffer'... not tested
Checking `wted'... nothing deleted
Checking `scalper'... not infected
Checking `slapper'... not infected
Checking `z2'...
nothing deleted
```

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix D: Related Virginia Statutes[30]

§ 18.2-152.3. Computer fraud.
Any person who uses a computer or computer network without authority and with the intent to:
1. Obtain property or services by false pretenses;
2. Embezzle or commit larceny; or
3. Convert the property of another shall be guilty of the crime of computer fraud. If the value of the property or services obtained is $200 or more, the crime of computer fraud shall be punishable as a Class 5 felony. Where the value of the property or services obtained is less than $200, the crime of computer fraud shall be punishable as a Class 1 misdemeanor.
(1984, c. 751; 1985, c. 322.)

§ 18.2-152.1. Short title.
This article shall be known and may be cited as the "Virginia Computer Crimes Act."
(1984, c. 751.)

§ 18.2-152.4. Computer trespass; penalty.
A. It shall be unlawful for any person to use a computer or computer network without authority and with the intent to:
1. Temporarily or permanently remove, halt, or otherwise disable any computer data, computer programs, or computer software from a computer or computer network;
2. Cause a computer to malfunction, regardless of how long the malfunction persists;
3. Alter or erase any computer data, computer programs, or computer software;
4. Effect the creation or alteration of a financial instrument or of an electronic transfer of funds;
5. Cause physical injury to the property of another;
6. Make or cause to be made an unauthorized copy, in any form, including, but not limited to, any printed or electronic form of computer data, computer programs, or computer software residing in, communicated by, or produced by a computer or computer network; or
7. Falsify or forge electronic mail transmission information or other routing information in any manner in connection with the transmission of unsolicited bulk electronic mail through or into the computer network of an electronic mail service provider or its subscribers.
B. It shall be unlawful for any person knowingly to sell, give or otherwise distribute or possess with the intent to sell, give or distribute software which (i) is primarily designed or produced for the purpose of facilitating or enabling the falsification of electronic mail transmission information or other routing information; (ii) has only limited commercially significant purpose or use other

[30] http://leg1.state.va.us/cgi-bin/legp504.exe?000+cod+TOC

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

than to facilitate or enable the falsification of electronic mail transmission information or other routing information; or (iii) is marketed by that person or another acting in concert with that person with that person's knowledge for use in facilitating or enabling the falsification of electronic mail transmission information or other routing information.
C. Any person who violates this section shall be guilty of computer trespass, which offense shall be punishable as a Class 3 misdemeanor. If there is damage to the property of another valued at $2,500 or more caused by such person's reckless disregard for the consequences of his act in violation of this section, the offense shall be punished as a Class 1 misdemeanor. If there is damage to the property of another valued at $2,500 or more caused by such person's malicious act in violation of this section, the offense shall be punishable as a Class 6 felony.
D. Nothing in this section shall be construed to interfere with or prohibit terms or conditions in a contract or license related to computers, computer data, computer networks, computer operations, computer programs, computer services, or computer software or to create any liability by reason of terms or conditions adopted by, or technical measures implemented by, a Virginia-based electronic mail service provider to prevent the transmission of unsolicited electronic mail in violation of this article. Nothing in this section shall be construed to prohibit the monitoring of computer usage of, the otherwise lawful copying of data of, or the denial of computer or Internet access to a minor by a parent or legal guardian of the minor.
(1984, c. 751; 1985, c. 322; 1990, c. 663; 1998, c. 892; 1999, cc. 886, 904, 905; 2002, c. 195.)

§ 18.2-152.7. Personal trespass by computer.
A. A person is guilty of the crime of personal trespass by computer when he uses a computer or computer network without authority and with the intent to cause physical injury to an individual.
B. If committed maliciously, the crime of personal trespass by computer shall be punishable as a Class 3 felony. If such act be done unlawfully but not maliciously, the crime of personal trespass by computer shall be punishable as a Class 1 misdemeanor.
(1984, c. 751; 1985, c. 322.)

§ 18.2-152.2. Definitions.
For purposes of this article:
"Computer" means an electronic, magnetic, optical, hydraulic or organic device or group of devices which, pursuant to a computer program, to human instruction, or to permanent instructions contained in the device or group of devices, can automatically perform computer operations with or on computer data and can communicate the results to another computer or to a person. The term "computer" includes any connected or directly related device, equipment, or

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

facility which enables the computer to store, retrieve or communicate computer programs, computer data or the results of computer operations to or from a person, another computer or another device.
"Computer data" means any representation of information, knowledge, facts, concepts, or instructions which is being prepared or has been prepared and is intended to be processed, is being processed, or has been processed in a computer or computer network. "Computer data" may be in any form, whether readable only by a computer or only by a human or by either, including, but not limited to, computer printouts, magnetic storage media, punched cards, or stored internally in the memory of the computer.
"Computer network" means two or more computers connected by a network.
"Computer operation" means arithmetic, logical, monitoring, storage or retrieval functions and any combination thereof, and includes, but is not limited to, communication with, storage of data to, or retrieval of data from any device or human hand manipulation of electronic or magnetic impulses. A "computer operation" for a particular computer may also be any function for which that computer was generally designed.
"Computer program" means an ordered set of data representing coded instructions or statements that, when executed by a computer, causes the computer to perform one or more computer operations.
"Computer services" means computer time or services, including data processing services, Internet services, electronic mail services, electronic message services, or information or data stored in connection therewith.
"Computer software" means a set of computer programs, procedures and associated documentation concerned with computer data or with the operation of a computer, computer program, or computer network.
"Electronic mail service provider" means any person who (i) is an intermediary in sending or receiving electronic mail and (ii) provides to end-users of electronic mail services the ability to send or receive electronic mail.
"Financial instrument" includes, but is not limited to, any check, draft, warrant, money order, note, certificate of deposit, letter of credit, bill of exchange, credit or debit card, transaction authorization mechanism, marketable security, or any computerized representation thereof.
"Network" means any combination of digital transmission facilities and packet switches, routers, and similar equipment interconnected to enable the exchange of computer data.
"Owner" means an owner or lessee of a computer or a computer network or an owner, lessee, or licensee of computer data, computer programs, or computer software.
"Person" shall include any individual, partnership, association, corporation or joint venture.
"Property" shall include:
1. Real property;
2. Computers and computer networks;
3. Financial instruments, computer data, computer programs, computer software and all other personal property regardless of whether they are:

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

a. Tangible or intangible;
b. In a format readable by humans or by a computer;
c. In transit between computers or within a computer network or between any devices which comprise a computer; or
d. Located on any paper or in any device on which it is stored by a computer or by a human; and
4. Computer services.
A person "uses" a computer or computer network when he:
1. Attempts to cause or causes a computer or computer network to perform or to stop performing computer operations;
2. Attempts to cause or causes the withholding or denial of the use of a computer, computer network, computer program, computer data or computer software to another user; or
3. Attempts to cause or causes another person to put false information into a computer.
A person is "without authority" when (i) he has no right or permission of the owner to use a computer or he uses a computer in a manner exceeding such right or permission or (ii) he uses a computer, a computer network, or the computer services of an electronic mail service provider to transmit unsolicited bulk electronic mail in contravention of the authority granted by or in violation of the policies set by the electronic mail service provider. Transmission of electronic mail from an organization to its members shall not be deemed to be unsolicited bulk electronic mail.
(1984, c. 751; 1999, cc. 886, 904, 905; 2000, c. 627.)

© SANS Institute 2003, Author retains full rights

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

Appendix E: Timeline of Events

Due to the large number of files found, I have limited this list to showing groupings of modifications, and any key pieces that are relevant to the investigation. A vertical "…" indicates that there were numerous files modified within this same directory between the date/time listed.

In case where a filename is displayed, it means that "cerwin" modified this file on the date in question.

Aug 24th, 1999

21:15:34 1999 501 ./cvsroot/chaos/srcupdate.c,v
21:48:52 1999 501 ./cvsroot/chaos/srcmagic2.c,v

Aug 25th, 1999

12:35:34 1999 501 ./cvsroot/chaos/srcconst.c,v

Aug 26th, 1999

20:43:04 1999 501 ./cvsroot/chaos/srcrecycle.c,v

Jan 1st, 2000

00:00:00 2000 501 ./apache/htdocs/Chevy/ETown-5-2001/1P1010001.JPG
.
.
.
00:00:00 2000 501 ./apache/htdocs/Chevy/ETown-5-2001/7P1010017.JPG

March 1st, 2000

21:27:48 2000 501 ./apache/htdocs/CerwinCobraR-2.jpg
21:33:11 2000 501 ./apache/htdocs/CerwinCobraR-1.jpg

May 29th, 2000

20:54:40 2000 501 ./apache/htdocs/Cerwin/MyPhotosaudi-1-1635x1070.jpg
.
.
.
21:52:51 2000 501 ./apache/htdocs/Cerwin/MyPhotossaleen-5-1656x1111.jpg

May 30th, 2000

14:30:45 2000 501 ./apache/htdocs/Cerwin/MyPhotosintegra-1-1631x1079.jpg
.
.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

.
23:37:05 2000 501 ./apache/htdocs/Cerwin/MyPhotosaudi.html

June 7th, 2000

20:57:17 2000 501 ./apache/htdocs/Cerwin/MyPhotosporsche-1-1668x1138.jpg
.
.
.
00:48:28 2000 501 ./apache/htdocs/Cerwin/MyPhotosmy_car-5-254x169.jpg8

Aug 22nd, 2000

09:18:54 2000 501 ./apache/htdocs/Cerwin/MyPhotosspeed1.jpg
.
.
.
09:19:36 2000 501 ./apache/htdocs/Cerwin/MyPhotosranger-5-628x424.jpg

Sep 27th, 2000

09:25:31 2000 501 ./apache/htdocs/Cerwin/MyPhotoslowered-6-1609x1090.jpg
.
.
.
09:26:38 2000 501 ./apache/htdocs/Cerwin/MyPhotossprings-8-629x426.jpg

Oct 27th, 2000

15:32:04 2000 501 ./apache/htdocs/Cerwintabasco.mpeg

Nov 27th, 2000

17:14:11 2000 501 ./apache/htdocs/Cerwinjetta.jpg

Jan 4th, 2001

20:33:40 2001 501./cerwin/shoutcast-1-8-0-linux-glibc6/contentscpromo.mp3
20:36:42 2001 501 ./cerwin/shoutcast-1-8-0-linux-glibc6README
21:45:51 2001 501 ./cerwin/shoutcast-1-8-0-linux-glibc6sc_serv

Feb 9th, 2001

18:52:35 2001 501 ./apache/htdocs/Cerwin/MyPhotosdragster-1-249x166.jpg

March 8th, 2001

11:08:10 2001 501 ./apache/htdocs/radio/reviewsTS-940S.pdf
.
.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

.
16:31:12 2001 501 ./ftp/pubtrue_romance-dvd.rip.divx.avi

May 15th, 2001

13:15:55 chaos ftpd[4209]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76], cerwin
13:16:45 chaos ftpd[4212]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76], cerwin

May 18th, 2001

10:20:06 chaos PAM_pwdb[5426]: (login) session opened for user cerwin by (uid=0)
10:20:22 chaos PAM_pwdb[5426]: (login) session closed for user cerwin

May 22nd, 2001

09:05:37 chaos PAM_pwdb[7290]: (login) session opened for user cerwin by (uid=0)
09:05:39 chaos PAM_pwdb[7290]: (login) session closed for user cerwin
09:21:45 chaos PAM_pwdb[7308]: (login) session opened for user cerwin by (uid=0)
09:23:18 chaos PAM_pwdb[7332]: 1 authentication failure; cerwin(uid=501) -> root for su service
09:23:21 chaos PAM_pwdb[7333]: (su) session opened for user root by cerwin(uid=501)
09:34:08 chaos PAM_pwdb[7308]: (login) session closed for user cerwin

May 29th, 2001

15:04:24 chaos ftpd[10418]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76], cerwin
15:05:07 chaos ftpd[10420]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76], cerwin
15:05:10 2001 1 gatekeeper.bristol.com 84708 /home/apache/htdocs/Cerwin/MyPhotos/tokico-2-629x426.jpg b _ i r cerwin ftp 0 * c
15:05:10 2001 501 ./apache/htdocs/Cerwin/MyPhotostokico-2-629x426.jpg
.
.
.
16:40:10 2001 501 ./apache/htdocs/Cerwin/MyPhotoslowered-6-251x170.jpg
16:39:41 chaos ftpd[10465]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76], cerwin
16:40:01 chaos ftpd[10467]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76], cerwin

May 30th, 2001

10:31:18 chaos PAM_pwdb[502]: (login) session opened for user cerwin by (uid=0)
10:31:43 chaos PAM_pwdb[502]: (login) session closed for user cerwin
13:17:16 chaos PAM_pwdb[579]: (login) session opened for user cerwin by (uid=0)
13:17:38 chaos PAM_pwdb[593]: (su) session opened for user root by cerwin(uid=501)
13:38:39 chaos PAM_pwdb[579]: (login) session closed for user cerwin

June 4th, 2001

10:02:29 chaos PAM_pwdb[2788]: (login) session opened for user cerwin by (uid=0)@
10:03:04 chaos PAM_pwdb[2807]: 1 authentication failure; cerwin(uid=501) -> root for su service
10:03:09 chaos PAM_pwdb[2808]: (su) session opened for user root by cerwin(uid=501)
10:06:29 chaos PAM_pwdb[2788]: (login) session closed for user cerwin
10:08:09 chaos PAM_pwdb[2851]: (login) session opened for user cerwin by (uid=0)

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

10:08:22 chaos PAM_pwdb[2866]: 1 authentication failure; cerwin(uid=501) -> root for su service
10:08:24 chaos PAM_pwdb[2867]: (su) session opened for user root by cerwin(uid=501)
10:19:46 chaos PAM_pwdb[2851]: (login) session closed for user cerwin

June 5th, 2001

08:05:20 chaos ftpd[3291]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76], cerwin
08:05:44 chaos ftpd[3294]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76], cerwin
14:39:09 chaos ftpd[3392]: FTP LOGIN FROM gatekeeper.bristol.com [207.41.40.76], cerwin

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.

References and Cited Sources

1. RSA Laboratories: http://www.rsasecurity.com/rsalabs/faq/3-6-6.html
2. Google is a popular and powerful search engine. http://www.google.com
3.http://www.secinf.net/unix_security/LOKI2__informationtunneling_program_and_description.html
4. http://www.iss.net/security_center/static/1452.php
5. http://www.rpmfind.net
6. http://www.phrack-dont-give-a-shit-about-dmca.org/show.php?p=51&a=6
7. http://www.mudconnect.com/tmcfaq.html
8. http://www.aardvarkinc.com/support/at_vs_atx.htm
9. http://www.linux.org/docs/ldp/howto/Large-Disk-HOWTO-13.html
10. http://www.atstake.com/research/tools/task/
11. Introduction to Linux kernel modules By Vans Information http://www.freeos.com/articles/4051/ 2001-05-16
12. "Concurrent Versions System (CVS) http://www.gnu.org/software/cvs/
13. Linux Administrator's Security Guide - Passwords by Kurt Seifried http://www.windowsecurity.com/whitepapers/Linux_Administrators_Security_Guide__Passwords.html
14. http://www.courtesan.com/sudo/
15. http://www.chkrootkit.org/
16. The term "script kiddies" refers specifically to the tendency of inexperienced crackers ("kiddies) to use pre-made cracking tools ("scripts").
17. A popular distributed networking client designed to help find extraterrestrial life. http://setiathome.ssl.berkeley.edu/
18. The Linux Runlevel, by Doran Barton http://www.iodynamics.com/education/runlevel.html
19. The Linux Kernel Module system essentially loads and unloads different information into the operating kernel in order to add or remove functionality from the system. This is to help maintain efficiency by only having needed programs loaded into memory. http://www.tldp.org/HOWTO/Kernel-HOWTO.html
20. Red Hat produces a version of the Linux operating system, which is installed on this system. http://www.redhat.com.
21. Apache.org makes a popular web server, typically installed by default on Linux systems. http://www.apache.org.
22. A suite of tools that allows for the encrypted transmission of data across a network. http://www.openssh.org/
23. http://www.linux.org/apps/AppId_7453.html
24. http://cio.doe.gov/Documents/CFA.HTM
25. http://cio.doe.gov/Documents/ECPA.HTM
26. http://www.eff.org/Privacy/Surveillance/Terrorism_militias/hr3162.php
27.http://www.eff.org/Privacy/Surveillance/Terrorism_militias/20011031_eff_usa_patriot_analysis.html
28. United States v. Mullins, 992 F.2d 1472, 1478 (9th Cir. 1993)
29. United States v. McLaren, 957 F. Supp. 215, 219 (M.D. Fla. 1997)
30. http://leg1.state.va.us/cgi-bin/legp504.exe?000+cod+TOC

© SANS Institute 2003, As part of GIAC practical repository. Author retains full rights.