Global Information Assurance Certification Paper

SANS GIAC
Computer Forensic Analysts
Certification: v 1.2

J. Philip Craiger

March 29, 2003

Part 1

Analysis of an Unknown Binary: Loki ICMP Covert Channel Daemon

Abstract

This paper describes the forensic procedures applied to a binary of an unknown origin or function. Static and dynamic analyses were conducted on the binary. The analyses indicate that the binary is a well-known ICMP covert channeling executable. The source code was located on the Internet in a well-known online hacker's journal, and an attempt was made to compile the code. Although unsuccessful, the article and source code provided enough information to allow me to track down the author of the code. The legal implications of running the binary are discussed, and several interview questions are provided.

Static Analysis

Described below are the steps required to conduct a static analysis of the unknown binary.

Step 1. Calculate the MD5 hash of the binary.

[root@whammo sans.cert]# md5sum binary_1.2.zip > b1.2.zip.md5

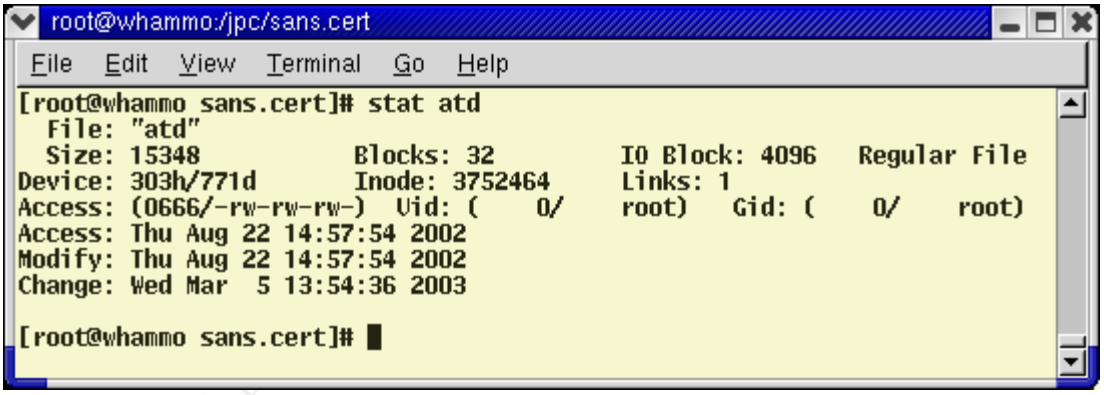
Rationale: Normally, this would not be needed because typically we are only interested in the cryptographic hash of the binary itself. Nevertheless, it doesn't hurt to have it -- just in case.

Step 2. Unzip the compressed file.

[root@whammo sans.cert]# unzip binary_1.2.zip

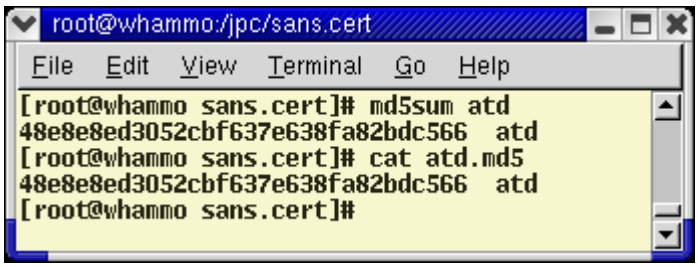
Rationale: Self explanatory. This step resulted in the extraction of two files: *atd* and *atd.md5*. The latter file contained an MD5 hash of (apparently) *atd*.

Step 3. Calculate the MAC times of the binary.



Rationale: Determine a timeline as to when the program might have last been accessed (run), compiled (modified), etc. I used the UNIX utility *stat* to identify the modified, accessed, & changed times for the binary. Results of the *stat* command indicate that the last time the program was accessed was Thursday, August 22 2002, at 2:57 in the afternoon (14:57 in military time). The modified time was the same. The changed time was the time the binary was copied to my own hard drive.

Step 3. Calculate the MD5 cryptographic hash of the binary, and compare to the hash included in the atd.md5 file.



Rationale: To test the integrity of the file, that is, making sure that it has not been changed in any way. Note that if I had used this command prior to the *stat* command the accessed time of the binary would have changed.

The MD5 for the extracted file was then compared against the MD5 hash that accompanied the unknown binary. As Figure 2 illustrates the cryptographic hashes are the same, indicating the integrity of the binary.

Step 5. Determine the file type.

[root@whammo sans.cert]# file atd > atd.file

Rationale: The file command allows me to determine the type of file in question. In this case, the command indicated the unknown binary was of type:

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

The "dynamically linked" term indicates that the binary requires certain library files to function properly. Accordingly, some of the utilities used in the dynamic analysis section should assist in identifying those libraries (e.g., *lsof*, *strace*). The 'stripped' means that the binary has been stripped of any debugging information, which is unfortunate as it often contains clues as to the binary's functions.

I also ran *readelf* against the file to obtain more specific information regarding the executables internals. According to itrs man page, *readelf* displays information about one or more ELF format object files.

[root@whammo sans.cert]# readelf -h atd
ELF Header:

Magic:	7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
Class:	ELF32
Data:	2's complement, little endian
Version:	1 (current)
OS/ABI:	UNIX - System V
ABI Version:	0
Type:	EXEC (Executable file)
Machine:	Intel 80386

Version:	0x1
Entry point address:	0x8048db0
Start of program headers:	52 (bytes into file)
Start of section headers:	14508 (bytes into file)
Flags:	0x0
Size of this header:	52 (bytes)
Size of program headers:	32 (bytes)
Number of program headers:	5
Size of section headers:	40 (bytes)
Number of section headers:	21
Section header string table index:	20

Significant results are the Magic number, which the file command uses to identify the type of file; the file class; the OS; machine; and the entry point address. The results confirm the results from the *file* command.

Step 6. Use the strings UNIX utility to extract human-readable strings from the binary in order to acquire clues about its function, author, etc.

[root@whammo sans.cert]# strings -a atd > atd.strings

Rationale: The *strings* command is able to write out any human readable text from a binary source. The *strings* command found several useful pieces of information within the unknown binary. Some of this information was used to identify the location of the source code on the Internet, as well as the author of the program.

Step 7. Analyze strings from step 6 to determine the binary's function

The UNIX utility '*strings –a*' was run against the binary to extract the human-readable strings. The reason for this procedure is that binaries often contain human-readable strings (user messages, help, error messages, etc.), that can be used as keywords to search the Internet to find the associated source code.

By far the most common unique string from the results of this procedure was the term 'lokid.' This term lead me to believe that the program was part of a well-known Loki ICMP covert channel application. The 'd' on the end was a clue that this was a daemon, as most UNIX daemons have a 'd' on the end of their names, although this was pure speculation.

Brief Tangent: What the Hell is "Loki?"

In cracking parlance Loki is a Trojan horse that provides a covert channeling (back door) for tunneling information over ICMP, specifically, using ICMP type ECHO_REPLY (0) and ECHO (8). A covert channel provides a surreptitious channel for the communication of signals (Pfleeger & Pfleeger, 2003; Miller, J., 1988).

Daemon9 (route), the author of both Phrack articles as well as the Loki source code, writes that Loki is simple a form of steganography (i.e., hiding information within another media).

> *"Loki, Norse God of deceit and trickery, the 'Lord of Misrule' was well known for his subversive behavior. Inversion and reversal of all sorts was typical for him. Due to its clandestine nature, we chose to name this project after him." (Daemon9, 1996)*

Back to the Strings Analysis

Partial results of the output from the *strings* command are provided below. Some of the strings are clues as to the functionality provided by the Loki daemon. For example, the string in line 5 appears to indicate that cryptography is available (actually, more than one type of cryptography is available). Line 13 appears to be feedback to a user on the proper use of the command:

lokid -p (i|u [-v (0|1)]

Line 14 is perhaps the most interesting and seemingly unique because it appears to tell us the name of the program as well as the author (daemon9 AKA route), as well as the year that the program was written (or at least copyrighted):

LOKI2 route [(c) 1997 guild corporation worldwide]

I believed this string to be unique enough to warrant its use in a Google search (described later).

1.lokid: Client database full

2.lokid version:

3.remote interface:

4.active transport:

5.active cryptography:

6.server uptime:

7.lokid: inactive client <%d> expired from list [%d]

8.@[fatal] shared mem segment request error

9.lokid -p (i|u) [-v (0|1)]

10.LOKI2 route [(c) 1997 guild corporation worldwide]

11.lokid: server is currently at capacity. Try again later

12.lokid: Cannot add key

13.lokid: popen

14.lokid: client <%d> requested an all kill

15.lokid: clean exit (killed at client request)

16.lokid: cannot locate client entry in database

17.lokid: client <%d> freed from list [%d]

18.lokid: unsupported or unknown command string

19.lokid: client <%d> requested a protocol swap

20.sending protocol update: <%d> %s [%d]

21.lokid: transport protocol changed to %s

22.GCC: (GNU) 2.7.2.1

Line 22 appears to indicate that the GNU compiler version 2.7.2.1 was used to compile the binary. This is a helpful clue as later I will attempt to compile the source code, which may or may not be successful depending upon the libraries and compiler used to create the original program. Therefore, knowing that this binary was compiled successfully with the gcc 2.7.2.1 is helpful.

Speculation as why the name 'atd' was chosen

The name 'atd' was probably chosen because it would replace the real *at* daemon. *at* is a UNIX utility that allows users to specify commands that should be run at a specific time or date. For example, the command:

[root@whammo sans.cert]# at 5PM bkup.sh

would run the backup shell file at 5PM. Because the *at* daemon is probably run by default on startup on most UNIX/Linux systems, so that even if the host is shutdown, on reboot the Trojaned atd (Loki) would restart. Also, the fact that the real atd was replaced by a Trojan might not found, most likely because *at* is not a command that is used as often as, for example, cron jobs, which most likely would be noticed if a cron job was not run.

Step 8: Using results from strings as keywords to a Google search

I used the term "LOKI2 route [(c) 1997 guild corporation worldwide]" as the keyword to a Google search. One of the returned hits was from Phrack Magazine, an online hacker's journal, specifically, Volume 7, Issue 51 September 01, 1997. This issue contained an article 06 of 17: LOKI2 (the implementation) daemon9 route@infonexus.com. This article contained source code that appeared to be the same, or at least highly similar, to the strings derived from the binary.

Clearly daemon9, the author of the article and source code, is a pseudonym; however, it is a clue that can be used to perhaps determine the author of the program. I did some reconnaissance by visiting www.infonexus.com. The only person apparently associated

with the website is Mike Schiffman, a picture of whom is provided below (from the web site).

According to the web site this is Mike Schiffman. Mike appears to be a developer/programmer with various interests. Wanting to know if Mike was the author of Loki I emailed Mike (mike@infonexus.com, according to the web site) and straight forwardly asked:

From: Dr. Philip Craiger <philip_craiger@yahoo.com>
Reply-To: philip@craiger.net
To: mike@infonexus.com
Subject: Question regarding Phrack article
Date: 05 Mar 2003 10:47:10 -0600

Hi Mike,

I was reading an article from Phrack magazine about the LOKI2 ICMP covert channeling program. It's author appears to be:

daemon9 <route@infonexus.com>

I was wondering if you are 'that' daemon9?

Thanks.

Philip

Of course, I could have emailed route@infonexus and asked "who are you in real life?" However, I decided to try the aforementioned strategy first to see if Mike admits to being the author.

Several minutes later I received the following reply:

From: Mike Schiffman <mike@infonexus.com>
To: philip@craiger.net
Subject: RE: Question regarding Phrack article
Date: Wed, 5 Mar 2003 08:53:21 -0800

I am.

-----Original Message-----
From: Dr. Philip Craiger [mailto:philip_craiger@yahoo.com]
Sent: Wednesday, March 05, 2003 8:47 AM
To: mike@infonexus.com
Subject: Question regarding Phrack article

Hi Mike,

I was reading an article from Phrack magazine about the LOKI2 ICMP covert channeling program. It's author appears to be:

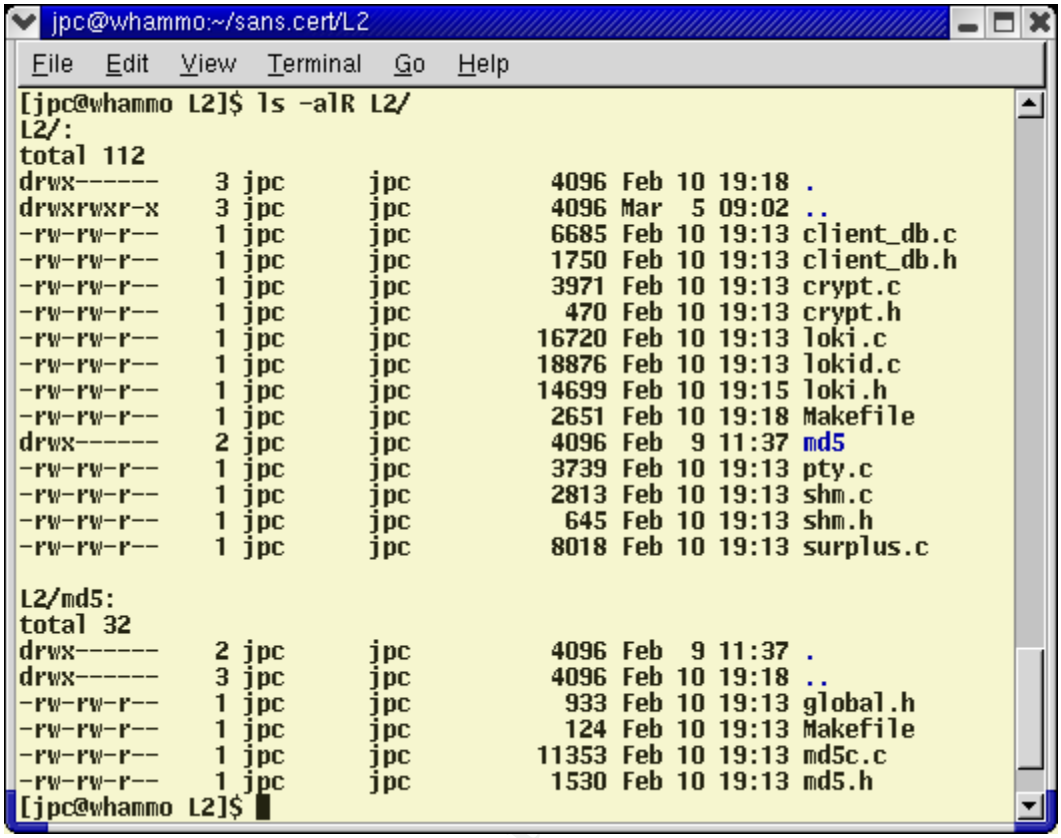
Step 9. Download the article containing the source code.

I downloaded the article containing the Loki source code. The article contains a detailed description of the Loki program, the technologies used, and a graphical depiction of the workings of the program as well as the source code.

NOTE: daemon9 is the author of a prequel Phrack article (http://www.phrack.org/show.php?p=49&a=6) that discussed the background on ICMP, basic firewall theory and covert channels, how Loki attempts to subvert detection, as well as a discussion on how to detect and prevent the use of covert channels. It's a very worthy read.

Step 10. Extract the source code from the article

The source code for Loki is embedded in the Phrack article. Phrack provides source code for an 'extract' executable that is run against the Phrack article which automatically extracts the source code and places it in appropriate directories. Accordingly I downloaded and compiled the 'extract.c' program from the Phrack site, and ran it against the text from the Phrack article. The resulting files and directory structure are displayed below.



Next I attempted to compile the source code on an air-gapped Dell P2 laptop running Redhat 8.0 using the instructions included in the Phrack article and the GNU compiler version 3.2. The source did not compile. The first error message indicated a problem with the *signal.h* header file. Given the modification date on the file, August 2000, from the *stat* command, I surmised that the libraries had changed and therefore needed to use an older version or an older version of the GNU compiler or both. I subsequently installed Redhat 7.1 and attempted to compile the source code. I received the same error. I repeated the process with Redhat 6.2, Redhat 5.2, and FreeBSD 4.5. Each attempt resulted in a failure with the same error message.

I went back to the original Phrack article and noted that the copyright date of 1996, 1997, which predates Redhat 5.2. I then attempted to download an even older version of Redhat (3 or 4) however, I could find none.

Next I took the source code to a knowledgeable C coder to see if he could identify and correct the problem. After several attempts to rework the code he also surmised that I needed to use older version of the libraries.

Although I could not compile the code, I believe there is a 99% chance that the source code is exactly the same as the binary provided by SANS. Given enough time I could probably, with the help of experienced coders (which I am not) brute force the problem

and get it to compile, that would most likely render the issue of comparing the MD5s moot.

Dynamic Binary Analysis

Overview

How Loki affects a system can only be determined by running the program and noting its effects. Questions to be answered include: What are the footprints when installed? How is the filesystem affected by the execution of the program? Does the program use or reference other files? These questions will be answered by using an experimental methodology using a control and experimental condition so that the two could be compared. The control condition involved booting a Linux Redhat 6.2 system prior to running the Loki daemon and capturing relevant information such as running processes, open sockets, and so forth. Next, the system is rebooted and then the Loki daemon is executed. Subsequently, running the same commands captures the same information. This serves as the experimental condition. The control condition serves as a baseline so that it can be more easily determined the effect that the Loki daemon has on the system.

I used four UNIX utilities to determine how the Loki daemon affects the system. These commands include *ps*, which displays running processes; *netstat*, which displays open sockets; *lsof*, which displays open files, and *strace*, which traces the program as it is executed.

The explanation accompanying the Loki source code indicates that it has been tested with Linux, FreeBSD, and OpenBSD. It specifically states that it has been tested with Linux kernel 2.2. Having used Linux for a little over a year (and only using a kernel > 2.4) I did a little research and found that Redhat 6.2 used a 2.2 kernel. I installed a Redhat 6.2 using the workstation installation on my old Dell P2 Laptop. I did no post-installation hardening to the system, such as turning off unneeded services, or adding or removing any software, to simulate a raw system that might be encountered on the Internet.

The *uname* command indicates the following the for Redhat 6.2 system:

```
Linux localhost.localdomain 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i686 unknown
```

Once installed I renamed the real *at* daemon and copied the Trojaned *atd* to /usr/sbin directory, where it normally resides. I checked to ensure that the *atd* was scheduled to load at run levels 3 and 5 by running the command: /sbin/chkconfig –list.

Below I compare and contrast the results of running each command across the control and experimental conditions.

Running processes

The running processes are displayed using the command *ps -aux*. The -a flag means select all running processes, including other users; -u means select by effective user ID; -x means include processes without controlling ttys.

Control Condition

```
USER   PID %CPU %MEM   VSZ  RSS TTY STAT START  TIME  COMMAND
root     1  8.8  0.1  1120              476 ?   S      11:52  0:06 init [3]
root     2  0.0  0.0     0    0 ?                    SW   11:52  0:00 [kflushd]
root     3  0.0  0.0     0    0 ?                    SW   11:52  0:00 [kupdate]
root     4  0.0  0.0     0    0 ?                    SW   11:52  0:00 [kpiod]
root     5  0.0  0.0     0    0 ?                    SW   11:52  0:00 [kswapd]
root     6  0.0  0.0     0    0 ?                    SW<  11:52  0:00 [mdrecoveryd]
bin    264  0.0  0.1  1212  420 ?                    S    11:53  0:00 portmap
root   279  0.0  0.0     0    0 ?                    SW   11:53  0:00 [lockd]
// Sections Deleted
nobody 377  0.0  0.1  1292  628 ?                    S    11:53   0:00 identd -e -o
nobody 380  0.0  0.1  1292  628 ?                    S    11:53   0:00 identd -e -o
nobody 381  0.0  0.1  1292  628 ?                    S    11:53  0:00 identd -e -o
nobody 383  0.0  0.1  1292  628 ?                    S    11:53  0:00 identd -e -o
nobody 384  0.0  0.1  1292  628 ?                    S    11:53  0:00 identd -e -o
daemon 395  0.0  0.1  1144  496 ?                    S    11:53  0:00 /usr/sbin/atd
root   409  0.1  0.1  1328  620 ?                    S    11:53  0:00 crond
root   424  0.0  0.1  1212  592 ?                    S    11:53  0:00 /sbin/cardmgr
root   438  0.0  0.1  1144  488 ?                    S    11:53  0:00 inetd
root   452  0.0  0.1  1204  532 ?                    S    11:53  0:00 lpd
root   496  0.0  0.2  2128 1124 ?                    S    11:53  0:00 sendmail:
root   511  0.0  0.1  1144  452 ?                    S    11:53  0:00 gpm -t ps/2
root   525  1.4  1.0  6036 3952 ?                    S    11:53  0:00 httpd
nobody 532  0.0  1.0  6132 4024 ?                    S    11:53  0:00 httpd
nobody 538  0.0  1.0  6132 4024 ?                    S    11:53  0:00 httpd
nobody 539  0.0  1.0  6132 4024 ?                    S    11:53  0:00 httpd
// Sections Deleted
```

```
root   602  0.3  0.2  1696  936 tty1          S    11:53   0:00 -bash
root   623  0.0  0.2  1696  936 tty1          S    11:53   0:00 -bash
root   625  0.0  0.2  2508  828 tty1          R    11:53   0:00 ps -aux
```

Trojaned System

Next the Loki daemon was executed and the ps -aux was run.

```
USER   PID %CPU %MEM   VSZ  RSS TTY STAT START   TIME COMMAND
root     1  2.9  0.1  1120  476 ?   S              11:45   0:06 init [3]
root     2  0.0  0.0     0    0 ?   SW             11:45   0:00 [kflushd]
root     3  0.0  0.0     0    0 ?   SW             11:45   0:00 [kupdate]
root     4  0.0  0.0     0    0 ?   SW             11:45   0:00 [kpiod]
root     5  0.0  0.0     0    0 ?   SW             11:45   0:00 [kswapd]
root     6  0.0  0.0     0    0 ?   SW<            11:45   0:00 [mdrecoveryd]
bin    264  0.0  0.1  1212  420 ?   S              11:45   0:00 portmap
root   279  0.0  0.0     0    0 ?   SW             11:45   0:00 [lockd]
// Sections Deleted
root   280  0.0  0.0     0    0 ?   SW             11:45   0:00 [rpciod]
nobody 380  0.0  0.1  1292  628 ?   S              11:45   0:00 identd -e -o
nobody 382  0.0  0.1  1292  628 ?   S              11:45   0:00 identd -e -o
nobody 383  0.0  0.1  1292  628 ?   S              11:45   0:00 identd -e -o
nobody 384  0.0  0.1  1292  628 ?   S              11:45   0:00 identd -e -o
root   395  0.0  0.0   868  304 ?   S              11:45   0:00 /usr/sbin/atd
root   409  0.0  0.1  1328  620 ?   S              11:45   0:00 crond
root   424  0.0  0.1  1212  592 ?   S              11:45   0:00 /sbin/cardmgr
root   438  0.0  0.1  1144  488 ?   S              11:45   0:00 inetd
root   452  0.0  0.1  1204  532 ?   S              11:45   0:00 lpd
nobody 544  0.0  1.0  6132 4024 ?   S              11:45   0:00 httpd
nobody 545  0.0  1.0  6132 4024 ?   S              11:45   0:00 httpd
nobody 546  0.0  1.0  6132 4024 ?   S              11:45   0:00 httpd
// Sections Deleted
root  602  0.0  0.2  1696  936 tty1 S              11:47   0:00 -bash
root  629  0.0  0.2  1696  936 tty1 S              11:48   0:00 -bash
```

root 631 0.0 0.2 2508 828 tty1 R 11:48 0:00 ps -aux

Results

When the Loki daemon is run as a startup service -- as the at daemon is intended -- it shows up in the list of running processes as running from the /usr/sbin directory, it's normal location. Consequently, nothing appears suspicious about the daemon from looking at the list of running processes.

Netstat: List of Open Sockets

The *netstat* command is used to print network connections, routing tables, interface statistics, and the like. The specific command run was: netstat -pan: the -p means to show the process ID and name of the program to which each socket belongs; the -a means show both listening and non-listening processes; and -n means to show numerical addresses instead of trying to determine symbolic host.

Control Condition

Active Internet connections (servers and established)

Proto	Recv-Q	SQ	Local Add	Foreign Add	State	ID/Program name
tcp	0	0	0.0.0.0:80	0.0.0.0:*	LISTEN	520/httpd
tcp	0	0	0.0.0.0:25	0.0.0.0:*	LISTEN	491/sendmail: accept
tcp	0	0	0.0.0.0:515	0.0.0.0:*	LISTEN	447/lpd
tcp	0	0	0.0.0.0:98	0.0.0.0:*	LISTEN	433/inetd
tcp	0	0	0.0.0.0:79	0.0.0.0:*	LISTEN	433/inetd
tcp	0	0	0.0.0.0:513	0.0.0.0:*	LISTEN	433/inetd
tcp	0	0	0.0.0.0:514	0.0.0.0:*	LISTEN	433/inetd
tcp	0	0	0.0.0.0:23	0.0.0.0:*	LISTEN	433/inetd
tcp	0	0	0.0.0.0:21	0.0.0.0:*	LISTEN	433/inetd
tcp	0	0	0.0.0.0:113	0.0.0.0:*	LISTEN	372/identd
tcp	0	0	0.0.0.0:887	0.0.0.0:*	LISTEN	284/rpc.statd
tcp	0	0	0.0.0.0:1024	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	259/portmap
udp	0	0	0.0.0.0:518	0.0.0.0:*		433/inetd
udp	0	0	0.0.0.0:517	0.0.0.0:*		433/inetd
udp	0	0	0.0.0.0:885	0.0.0.0:*		284/rpc.statd
udp	0	0	0.0.0.0:1024	0.0.0.0:*		-

```
udp   0      0        0.0.0.0:111  0.0.0.0:*              259/portmap
raw   0      0        0.0.0.0:1    0.0.0.0:*        7     -
raw   0      0        0.0.0.0:6    0.0.0.0:*        7     -
```

Note that the real *at* daemon does not show up the list of open and/or listening sockets.

Trojaned System

```
Active Internet connections (servers and established)
Proto RQSQ Local Add    Foreign Add        State  PID/Program name
tcp   0  0       0.0.0.0:80   0.0.0.0:*          LISTEN  525/httpd
tcp   0  0       0.0.0.0:25   0.0.0.0:*          LISTEN  496/sendmail: accept
tcp   0  0       0.0.0.0:515  0.0.0.0:*          LISTEN  452/lpd
tcp   0  0       0.0.0.0:98   0.0.0.0:*          LISTEN  438/inetd
tcp   0  0       0.0.0.0:79   0.0.0.0:*          LISTEN  438/inetd
tcp   0  0       0.0.0.0:513  0.0.0.0:*          LISTEN  438/inetd
tcp   0  0       0.0.0.0:514  0.0.0.0:*          LISTEN  438/inetd
tcp   0  0       0.0.0.0:23   0.0.0.0:*          LISTEN  438/inetd
tcp   0  0       0.0.0.0:21   0.0.0.0:*          LISTEN  438/inetd
tcp    0 0       0.0.0.0:113  0.0.0.0:*          LISTEN  377/identd
tcp    0 0       0.0.0.0:892  0.0.0.0:*          LISTEN  289/rpc.statd
tcp    0 0       0.0.0.0:1024 0.0.0.0:*          LISTEN  -
tcp    0 0       0.0.0.0:111  0.0.0.0:*          LISTEN  264/portmap
udp    0 0       0.0.0.0:518  0.0.0.0:*                  438/inetd
udp    0 0       0.0.0.0:517  0.0.0.0:*                  438/inetd
udp    0 0       0.0.0.0:890  0.0.0.0:*                  289/rpc.statd
udp    0 0       0.0.0.0:1024 0.0.0.0:*                  -
udp    0 0       0.0.0.0:111  0.0.0.0:*                  264/portmap
raw    0 0       0.0.0.0:255  0.0.0.0:*          7       395/atd
raw    0 0       0.0.0.0:1    0.0.0.0:*          7       395/atd
raw    0 0       0.0.0.0:1    0.0.0.0:*          7       -
raw    0 0       0.0.0.0:6    0.0.0.0:*          7       -
```

The Trojan Loki daemon (PID 395 above) opens two raw (i.e., ICMP) sockets, 255

and 1. (The two bottom lines in the output indicate that the kernel is listening on ports 1 & 6, and is typical). The fact that it would show up in this list makes sense because Loki functions as a covert ICMP channel (ergo the raw sockets as opposed to UDP or TCP).

The reason Loki opens two sockets is explained in the Phrack article. The below is the illustration from the Phrack article. Note that the first generation Loki daemon forks resulting in a second generation daemon, and the second generation also forks into a third generation daemon, while the first generation daemon exits. The second and third generation daemons appear to handle requests from the Loki client through separate threads, ergo the two open raw ports, 1 & 255.

LSOF: List of Open Files

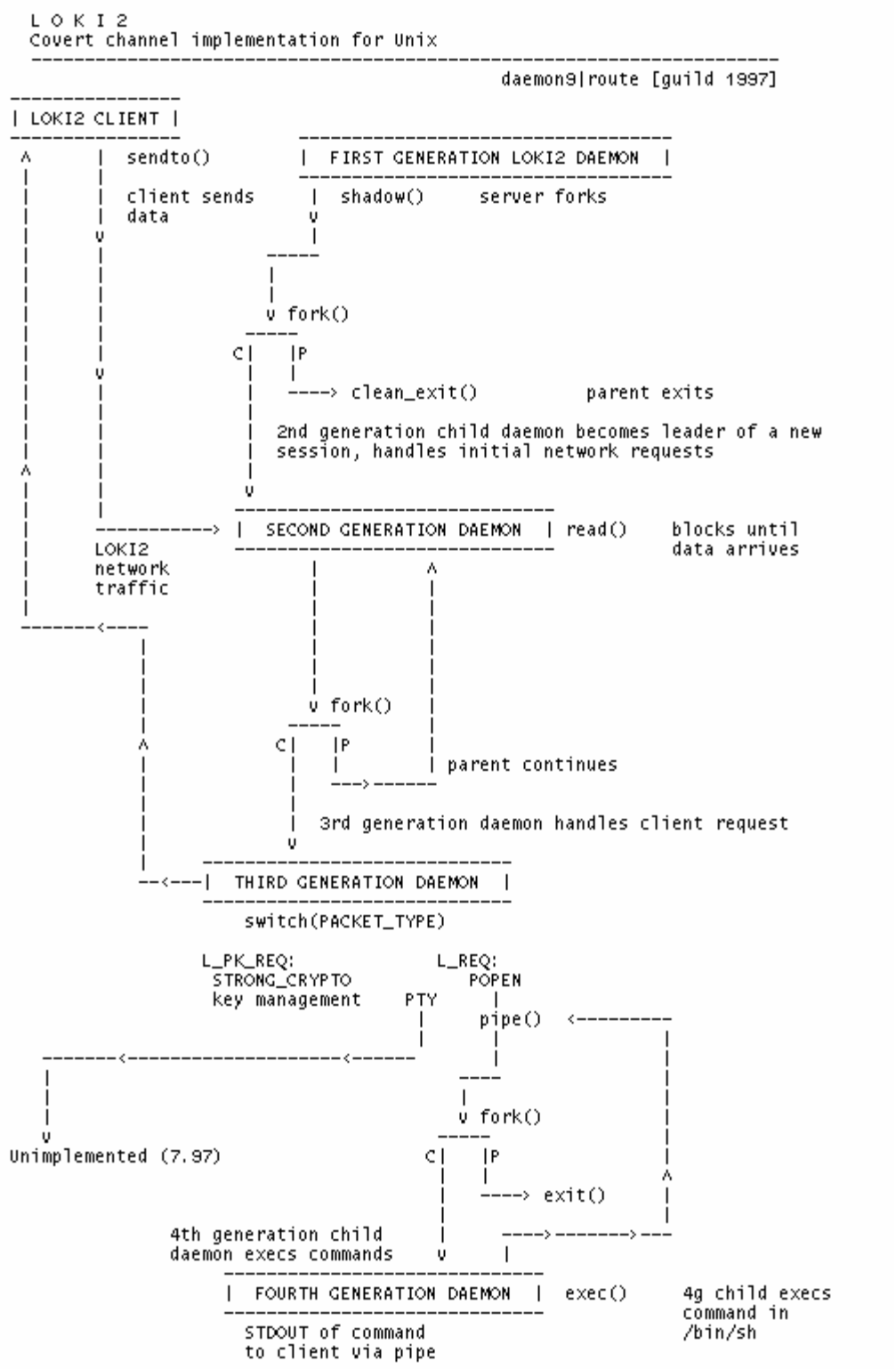
The *lsof* command indicates that files that are currently open and being used by running process or applications. The *lsof* was run both pre- and post-daemon. The entire results are several pages long, therefore, only the differences between the two listings are indicated below:

LSOF for the Control Condition

// lines deleted

```
COM  PID USER   FD   TYPE DEV  SIZE     NODE   NAME
atd  390 root  cwd    DIR 3,5    4096   506951 /var/spool/at
atd  390 root  rtd    DIR 3,5    4096        2 /
atd  390 root  txt    REG 3,5   13916  1291893 /usr/sbin/atd
atd  390 root  mem    REG 3,5  340663   425154 /lib/ld-2.1.3.so
atd  390 root  mem    REG 3,5 4101324   425161 /lib/libc-2.1.3.so
atd  390 root  mem    REG 3,5  246652   425192 /lib/libnss_files-2.1.3.so
atd  390 root    0u   CHR 1,3          196237 /dev/null
atd  390 root    1u   CHR 1,3          196237 /dev/null
atd  390 root    2u   CHR 1,3          196237 /dev/null
atd  390 root    3uW  REG 3,5       4  1488214 /var/run/atd.pid
atd  390 root    7r  FIFO 0,0            354 pipe
atd  390 root    8w  FIFO 0,0            354 pipe
atd  390 root   21w   CHR 1,3          196237 /dev/null
```

The output above describes the files that the at daemon uses when it is running. Note the three libraries opened.



LSOF for Trojaned System

// Lines deleted

```
COD  PID USER   FD   TYPE     DEVICE    SIZE     NODE NAME
atd  395 root  cwd    DIR 3,5    4096   179873 /tmp
atd  395 root  rtd    DIR 3,5    4096        2 /
atd  395 root  txt    REG 3,5   15348   539723 /usr/sbin/atd
atd  395 root  mem    REG 3,5   25386   425240 /lib/ld-linux.so.1.9.5
atd  395 root  mem    REG 3,5  699832  1275640 /usr/i486-linux-libc5/lib/libc.so.5.3.12
atd  395 root    1w  FIFO 0,0            415 pipe
atd  395 root    2w  FIFO 0,0            416 pipe
atd  395 root    3u   raw               419 00000000:0001->00000000:0000 st=07
atd  395 root    4u   raw               420 00000000:00FF->00000000:0000 st=07
atd  395 root    7r  FIFO 0,0            417 pipe
atd  395 root    8w  FIFO 0,0            417 pipe
atd  395 root   21w   CHR 1,3          196237 /dev/null
```

The list of open files for the Trojaned system looks very different. First, it uses entirely different library files, and it also shows the two raw sockets at ports 1 (HEX = :0001) and 255 (HEX = :00FF). Given the dissimilar functionality of the at & Loki daemons it is not unexpected to see that they use entirely different library files.

MAC Timeline

Recall that the Loki binary was copied to an air-gapped system running a freshly installed
(unpatched) copy of Redhat 6.2 using the server install. It was subsequently run to determine what files, if any, change on the hard drive. Once Loki was executed, the system was shut down and then restarted using FIRE (a bootable Linux CD that includes TASK and many other forensics applications).

TASK tools were used to determine what files are accessed, modified, or changed when the Loki is run. The first command run is *fls*, which lists files and directory names in a forensics image, along with each MAC time.

```
[root@FIRE]# fls -rf linux-ext2 -m / /dev/hda5 > /data/atd.mac
```

The command line above asks that *fls* work recursively (-r) on a EXT2 file system, getting MAC times for each file and directory encountered. Next inodes (including unallocated) are captured along with MAC times using the *ils* command.

[root@FIRE]# ils -f linux-ext2 -m /dev/hda5 >> /data/atd.mac

This command line above creates MAC times for each inode on /dev/hda5, the partition where the Loki daemon was executed. This output is concatenated onto the output from the previous command. Finally, the MAC times are formatted in human readable format using the *mactime* command.

[root@FIRE]# mactime -b /data/atd.mac 3/1/2003 > /data/atd.mactimes

The -b flag indicates the body file to be used with the *mactime* utility, the date 3/1/2003 indicates that I am only interested in the files with any of M,A,C on or after that date. Partial results of the analysis are indicated below.

// Dozens of lines deleted

Wed Mar 12 2003 16:32:38

699832 .a. -/-rwxr-xr-x 0 0 957427 /usr/i486-linux-libc5/lib/libc.so.5.3.12

25386 .a. -/-rwxr-xr-x 0 0 421913 /lib/ld-linux.so.1.9.5

44 .a. -/-rw-r--r-- 0 0 194693
/usr/share/locale/en_US/LC_MESSAGES/SYS_LC_MESSAGES

14 .a. l/lrwxrwxrwx 0 0 957452 /usr/i486-linux-libc5/lib/libc.so.5 -> libc.so.5.3.12bz.so.Y

17 .a. l/lrwxrwxrwx 0 0 421912 /lib/ld-linux.so.1 -> ld-linux.so.1.9.5

12210 .a. -/-rw-r--r-- 0 0 308565 /etc/ld.so.cache

15348 .a. -/-rwxr-xr-x 0 0 535481 /root/atd

The time and date, Wed Mar 12 2003 16:32:38, is when the Loki daemon was executed. Note that each is a library. The *libc.so.5* library includes all the standard C commands (e.g., *printf*, *read*, *open*, etc.) as well as system calls. This library has since been replaced by a version 6. Library *ld-lilnux.so.1* is the loader for the aforementioned library. Other files opened, including *ld.so.cache* and SYS_LC_MESSAGES are discussed more fully in the *strace* analysis below.

As the results of this analysis illustrate, the Loki server accesses several files during execution. Unfortunately (for the forensic analysis), they are files that are probably accessed by hundreds (thousands) of executables during the normal course of operation. Because of this fact it will make it more difficult to determine whether the Loki server was run on the system.

Tracing System Calls

The final part of dynamic analysis involved tracing the use of system calls by a running process. This was done with the Linux utility *strace*. *strace* acts like a wiretap between a program and the operating system, displaying information about files or networks that are accessed, memory access, and other types of system calls (Mandia & Prosise, 2002).

The exact command used to run *strace* was as follows:

[root@FIRE]# strace –o atd.out ./atd

The –o flag indicates that the trace should be written to the file *atd.out*. The results of *strace* are provided below.

1. execve("./atd", ["./atd"], [/* 19 vars */]) = 0
2. old_mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40007000
3. mprotect(0x40000000, 21772, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
4. mprotect(0x8048000, 13604, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
5. stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=12210, ...}) = 0
6. open("/etc/ld.so.cache", O_RDONLY) = 4
7. old_mmap(NULL, 12210, PROT_READ, MAP_SHARED, 4, 0) = 0x40008000
8. close(4) = 0
9. stat("/etc/ld.so.preload", 0xbffffb70) = -1 ENOENT (No such file or directory)
10. open("/usr/i486-linux-libc5/lib/libc.so.5", O_RDONLY) = 4
11. read(4, "\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
12. old_mmap(NULL, 823296, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x4000b000
13. old_mmap(0x4000b000, 592037, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0) = 0x4000b000
14. old_mmap(0x4009c000, 23728, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 4, 0x90000) = 0x4009c000
15. old_mmap(0x400a2000, 201876, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400a2000
16. close(4) = 0
17. mprotect(0x4000b000, 592037, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
18. munmap(0x40008000, 12210) = 0

19. mprotect(0x8048000, 13604, PROT_READ|PROT_EXEC) = 0
20. mprotect(0x4000b000, 592037, PROT_READ|PROT_EXEC) = 0
21. mprotect(0x40000000, 21772, PROT_READ|PROT_EXEC) = 0
22. personality(PER_LINUX) = 0
23. geteuid() = 0
24. getuid() = 0
25. getgid() = 0
26. getegid() = 0
27. geteuid() = 0
28. getuid() = 0
29. brk(0x804c818) = 0x804c818
30. brk(0x804d000) = 0x804d000
31. open("/usr/share/locale/en_US/LC_MESSAGES", O_RDONLY) = 4
32. fstat(4, {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
33. close(4) = 0
34. open("/usr/share/locale/en_US/LC_MESSAGES/SYS_LC_MESSAGES", O_RDONLY) = 4
35. fstat(4, {st_mode=S_IFREG|0644, st_size=44, ...}) = 0
36. old_mmap(NULL, 44, PROT_READ, MAP_PRIVATE, 4, 0) = 0x40008000
37. munmap(0x40008000, 44) = 0
38. close(4) = 0
39. stat("/etc/locale/C/libc.cat", 0xbffff694) = -1 ENOENT (No such file or directory)
40. stat("/usr/lib/locale/C/libc.cat", 0xbffff694) = -1 ENOENT (No such file or directory)
41. stat("/usr/lib/locale/libc/C", 0xbffff694) = -1 ENOENT (No such file or directory)
42. stat("/usr/share/locale/C/libc.cat", 0xbffff694) = -1 ENOENT (No such file or directory)
43. stat("/usr/local/share/locale/C/libc.cat", 0xbffff694) = -1 ENOENT (No such file or directory)
44. socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 4
45. sigaction(SIGUSR1, {0x804a6b0, [], SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SIG_DFL}, 0x4003ac68) = 0
46. socket(PF_INET, SOCK_RAW, IPPROTO_RAW) = 5
47. setsockopt(5, SOL_IP, IP_HDRINCL, [1], 4) = 0
48. getpid() = 625
49. getpid() = 625
50. shmget(867, 240, IPC_CREAT|0) = 2
51. semget(1049, 1, IPC_CREAT|0x180|0600) = 1
52. shmat(2, 0, 0) = 0x40008000
53. write(2, "\nLOKI2\troute [(c) 1997 guild cor"..., 52) = 52
54. time([1047438538]) = 1047438538

55. close(0) = 0
56. (SIGTTOU, {SIG_IGN}, {SIG_DFL}, 0x4003ac68) = 0
57. sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}, 0x4003ac68) = 0
58. sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}, 0x4003ac68) = 0
59. fork() = 626
60. close(5) = 0
61. close(4) = 0
62. semop(1, 0xbffffb0c, 2) = 0
63. shmdt(0x40008000) = 0
64. semop(1, 0xbffffb0c, 1) = 0
65. _exit(0) = ?

Strace Analysis

According to Mandia & Prosise (2002), the first 22 lines are "basically standard for all dynamically linked Executable Linked Format (ELF) executables." (p. 414). Nevertheless, I will briefly describe their meaning (given I went to the trouble of understanding their function).

The results of *strace* support the results of the MAC time analysis with respect to evidence of the files that are touched/open by Loki, including *ld.so.1, ld.so.5, ld.so.cache*, etc. However, *strace* provides far more information than the other dynamic analysis utility, especially in terms of things that the executable apparently needs to function properly, but cannon find on the running system. *ld.so.cache* is the first file that is searched for (line 4), and if found, opened read-only (line 5) and subsequently closed (line 8). ld.so.cache contains a compiled list of candidate libraries previously found in the augmented library path. It appears that *ld.so.cache* is opened for read-only to determine where the executable needs to find dynamically linked library files necessary for its operation.

Line 9 indicates that the file /etc/ld.so.preload was searched for, but not found on the system. According to Internet resources this file contains the names of libraries, which cause the system to preload this shared library every time a dynamically linked application is run (http://archives.neohapsis.com/archives/incidents/2002-01/0086.html). This source also indicates that this file is part of the source of rootkit vulnerability.

Line 34 indicates that the file SYS_LC_MESSAGES is opened read-only as file descriptor 4 (and subsequently close on line 33. Files in the /locate/en_US directory are base files for English language localization and are part of the *glibc* library, which is the standard C library (according to information on freshmeat.net). I viewed the contents of this file but was unable to ascertain its purpose (it contained what appeared to almost be grep-like commands).

I searched the Internet (via Google) using the keywords
"/usr/share/locale/en_US/LC_MESSAGES/SYS_LC_MESSAGES"
and "SYS_LC_MESSAGES." Although the keyword searched returned several hundreds (or thousands) of hits I could find no a precise description as to exactly what the file contains.

At line 39 we see that Loki attempts to find the file *libc.cat* by searching through several directories. The file is not found in any of the four directories where are searched. I attempted to manually search for the file, in case the file happened to reside in a directory not included within the four searches [find / -name "libc.cat" –print], but it did not appear that the file is included in the standard install. I used the keyword "*libc.cat*" as well as "What is *libc.cat*" for a Google search. I found several references to the file, however, no precise description as to its function.

Two raw (ICMP) sockets are opened as descriptors 4 & 5 at lines 44 & 46, respectively. We are aware of this fact from both the results of running the *netstat* as well as *lsof* utilities.

Lines 48 & 49 indicate that the running process received a process ID of 625. Also, it appears that when the process forks (line 59), the fork receives the process ID of 626.

Line 53 of the *strace* output shows that the phrase "LOKI2 route [(c) 1997 guild corporation worldwide]" is printed to standard out (the file descriptor 2). Line 65 indicates that the process exits.

Conclusion

The *strace* utility provided new as well as corroborating evidence regarding Loki's functionality. There are several places in the output that indicate that the process failed to find (apparently) needed files. This information could not be found with the other utilities employed in the dynamic analysis.

One interesting note is that the Loki doesn't appear to open any files on the hard drive for writing. Thus, it appears to leave little evidence of its execution outside of the opening of several library and associated files, files that used commonly by numerous other executables. The only real pieces of evidence of Loki is itself (the executable), and the fact that it writes out to standard error the executables name, author, and copyright date.

Had I been able to compile the original source code it would have been informative to place the code on an air-gapped network (two computers connected via switch, for example) and run both the server and several clients to determine how they react to the fact that several files could not be found (at least on a Redhat 6.2 system).

Legal Implications

The legal implications of the use of a covert channeling system are interesting are several levels. First I will discuss the applicability of the Computer Fraud and Abuse Act of 1987, followed by the Nebraska Computer Crimes Act (1995). First I will discuss whether Loki leaves any forensic evidence of it being executed on a system.

Evidence Loki Was Run

Recall that the purpose of Loki is to use ICMP packets as a covert channel of communications. One might assume that finding a running Loki client or server would be evidence of someone who attempting to thwart regular/acceptable communications channels. Under some circumstances one can imagine that the use of covert channels for communication may be necessary and ethical, albeit perhaps illegal, such as dissidents in oppressed political regimes.

Political dissidents' aside, however, the use of a covert channel is likely to be problematic in an organization. The use of a covert channel in an organizational setting is likely to indicate that someone is willfully subverting official organizational policy on communications, and probably violating the acceptable use policy. (It is unlikely that any acceptable use policy *specifically* addresses the use of covert channels, although perhaps some government intelligence or Department of Defense agencies include such provisions).

The dynamic analysis did not appear to demonstrate that the Loki daemon leaves anything in the way of evidence that it was ever run (unless, of course, someone finds that it is running during the computers operation). The evidence that makes it clear that we are not dealing with the usual *at* daemon is found in the *netstat, lsof,* and *strace* results. In each of these analyses there were two raw sockets opened by the Loki daemon. The normal *at* daemon would not be associated with two ICMP sockets.

There is no strong, clear-cut evidence that the Loki daemon has been run after it is killed. For example, it does not appear to open any files for writing (e.g., temporary, scratch files), which may be undeleted with appropriate forensics techniques (using *debugfs*, or any of the TCT or TASK tools). Several portions of the dynamic analysis did show that the daemon used several library files; however there are hundreds or thousands of executables that would use these library files, therefore, the behavior is not unique to the Loki daemon.

Loki server daemon **does** print out its name, author, and copyright date to the screen. If that information were found on the screen, or in some log file, then that could be considered evidence that the file was run, especially if the executable were also found on the hard drive.

Identifying Loki by Understanding Normal ICMP Traffic

One way to determine if an ICMP covert channel is in use is by comparing it to normal ICMP traffic. According to Mandia & Prosise (2002) normal ping traffic works accordingly:

1. Normal ICMP traffic begins with an ICMP echo request, followed by single ICMP echo reply
2. ICMP sequence number increments by one for each echo request sent
3. ICMP traffic contains a payload with predictable traffic
4. ECHO replies contain a payload that is identical to that of the corresponding request

Any network behavior violating these heuristics may be evidence of a covert channel. Point 1 above is fairly straightforward: ICMP *echo replies* must be preceded by a single associated *echo reply*. Regarding point 2, the authors demonstrated that Loki traffic has a distinctive signature instead of the normal incremented sequence number: *cdab.* Points 3 and 4 can be checked through logging or other more intrusive means (sniffing via *tcpdump*, *Ethereal*, etc.). Because ICMP covert channel is used as a backdoor the packets will contain human readable commands, which will clearly look as such.

Loki as a Violation of Acceptable Use Policies

The extent to which use of Loki violates an organization's acceptable use policy depends upon how well the policy is written. A good acceptable use policy will have a 'that which is not expressly allowed is denied.' This is a fail-safe policy given it would be difficult or impossible to list all of the possible types of illegal or unacceptable uses or abuses of a computer network.

Loki as a Violation of Federal Statutes

A person installing a Loki server on a computer without the owner's authorization could be indicted under the 1987 Computer Fraud and Abuse Act (CFAA: DOJ, 1987). The CFAA is part of 18 U.S.C. 1030, Fraud and Related Activity in Connection with Computers. The CFAA defines penalties for any person who gains unauthorized access to a protected computer, and in so doing causes damage or loss. The CFAA is fairly broad in its interpretation of illegal and unauthorized access to 'protected computers' (defined below). There appear to be several conditions under which a person installing Loki on a computer could fall under the CFAA. For example, articles 1 through 4 of the CFAA specify that whoever:

> **(1)** having knowingly accessed a computer without authorization or exceeding authorized access, and by means of such conduct having obtained information that has been determined by the United States Government pursuant to an Executive order or statute to require protection against unauthorized disclosure for reasons of national defense or foreign relations, or any restricted data, as defined in paragraph y of section 11 of the Atomic Energy Act of 1954.

Punishment: Fine or imprisonment of not more than 10 years, as long as the person has not been convicted of a similar crime under this heading. If convicted of more than one crime, subject to fine or imprisonment for not more than 20 years, or both.

(2) intentionally accesses a computer without authorization or exceeds authorized access…

(3) intentionally, without authorization to access any nonpublic computer of a department or agency of the United States, accesses such a computer of that department or agency that is exclusively for the use of the Government of the United States;

(4) knowingly and with intent to defraud, accesses a protected computer without authorization, or exceeds authorized access, and by means of such conduct furthers the intended fraud and obtains anything of value

Punishment for Sections 2 and 3: Fine or imprisonment for not more than one year, or both, unless person has been convicted previously under the statute, where the penalty is fine or imprisonment for not more than 10 years, or both.

Sections 5 (A) and (B) extend the conditions to situations where computer code or programs are transmitted using computers that are involved in the conduct of interstate commerce. It would not be difficult to argue that any web server or computer with a connection to the web may well fall under the rubric "used in the conduct of interstate commerce."

To paraphrase 5 (B), whoever: through means of a computer used in interstate commerce or communication, knowingly causes the transmission of a program, information, code, or command to a computer or computer system causes damage to computer system or network; or withhold or deny the use of the system or network; and if the transmission of the harmful component of the program, information, code, or command occurred without the authorization of the persons or entities who own or are responsible for the computer system receiving the program, information, code, or command; and causes loss or damage to one or more other persons of a value aggregating $1,000 or more during any 1-year period.

Section 5B appears to be directly applicable to a situation involving the unauthorized installation of Loki. Clearly it involves the transmission of code or a program, and it can be construed as causing damage based upon the definition of damage in the statute (see definition below). The value to the organization would probably quickly rise to $1,000 based upon the time it would take for several technical staff to conduct forensic analysis to determine what the program did, how it affected other programs (even thought it doesn't), cleaning the system, and even reinstalling the system from backups.

Civil Penalties

A person can also be subject to *civil penalties* as defined in section **(12)(G)** of the CFAA:

> Any person who suffers damage or loss by reason of a violation of this section may maintain a civil action against the violator to obtain compensatory damages and injunctive relief or other equitable relief. A civil action for a violation of this section may be brought only if the conduct involves 1 of the factors set forth in clause (i), (ii), (iii), (iv), or (v) of subsection (a)(5)(B).

Definition of Legal Terms

To determine whether the installation of Loki can be prosecuted under the articles of the CFAA one must define the key terms included within the statute. Recall that the CFAA defines penalties for any person who gains unauthorized access to a protected computer, and in so doing causes damage and loss. The CFAA defines the following terms:

- The term '*protected computer*' as (A) exclusively for the use of a financial institution or the United States Government, or (B) which is used in interstate or foreign commerce or communications, including a computer located outside the United States that is used in a manner that affects interstate or foreign commerce or communication of the United States;
- The term "*exceeds authorized access*" means to access a computer with authorization and to use such access to obtain or alter information in the computer that the accesser is not entitled so to obtain or alter;
- The term "*damage*" means any impairment to the integrity or availability of data, a program, a system, or information;
- The term "*loss*" includes any reasonable cost to any victim, including the cost of responding to an offense, conducting a damage assessment, and restoring the data, program, system, or information to its condition prior to the offense, and any revenue lost, cost incurred, or other consequential damages incurred because of interruption of service; and
- The term "*person*" means any individual, firm, corporation, educational institution, financial institution, governmental entity, or legal or other entity.

Nebraska Computer Crimes Act

Nebraska Criminal Code, Section 28-13, defines the Nebraska Computer Crimes Act (1995: http://downloads.securityfocus.com/library/nebraska.html). The NE state statute covers the use of computers in fraudulent transactions, denial of service, disclosures of passwords or other means of authentication or crimes against intellectual property. Clearly it is not as far reaching as the CFAA although it appears to be more specific than its Federal counterpart.

The Nebraska Computer Crimes Law states that the penalty for use of a computer to defraud is punished by a fine of not more than $10,000, or by imprisonment for not more than five years, or both. The penalty for disclosure of passwords, codes, or other means of authentication shall be punished by a fine of not more than $1,000, or by imprisonment for not more than six months, or both.

An offense against intellectual property is defined as the "intentional destruction, insertion or modification, without consent, of intellectual property; or disclosure, use, copying, taking or accessing, without consent, of intellectual property."

Whoever commits an offense against intellectual property shall be punished, upon conviction, by:

- A fine of not more than $ 1,000, or by
- imprisonment for not more than six months, or
- both.

However, when the damage or loss amounts to a value of $100 or more, the offender may be punished, upon convictions by:

- a fine of not more than $10,000, or
- imprisonment for not more than 5 years, or
- both.

The extent to which any of the Nebraska's statute would be applicable to the installation of Loki would depend upon the situation. If the backdoor was used to steal passwords, or download documents (considered intellectual property) or used to defraud others, then the perpetrator would be subject to the aforementioned penalties.

Interview Questions

There are several well known interviewing techniques that could be used for interview purposes. Two of which I studied include the Reid and the Kubark Techniques.

Reid Technique

The "Reid Technique" is used by law enforcement when interviewing suspects prior to answering this question. The following explanation was taken directly from the Reid website (http://www.reid.com/critic-technique.html).

> "The Reid Technique is a non-accusatory interview that is conducted before the investigator engages in an accusatory interrogation. The technique attempts to cull information from the interviewee by offering moral justification for the suspect's alleged crime. The motive for the admission of guilt I offered in the form of alternative choices provided the suspect. For example: "Did you commit the crime alone or did your friend help you?""

Kubark Technique

The Kubark Manual (http://www.mindcontrolforums.com/kubark.htm) was originally written for interrogating spies during the cold war era. The techniques I have used for this paper are the non coercive variety. The Kubark manual includes a set of coercive techniques, such as arrest, detention, pain, deprivation of sensory stimuli, etc., when the non coercive types do not work.

The Pitches

The first pitch is based on the Reid technique (which is also covered in the Kubark Manual) of allowing the interviewee to save face and empathize with them, giving them a way out.

Pitch 1: "I understand that recently you had a falling out with your manager over some work procedures, and that she threatened to fire you. This occurred a week prior to this incident. That must have been humiliating given that you've been a systems administrator for much longer than your boss. I also understand that you were logged into the system from a remote location at the time that the rootkit was planted on the system. I also understand that several of your friends and co-workers that you used to code for a particular rootkit. Did you place the rootkit on the system because you were angry at your manager, or just to test the security of your system?"

The second pitch is based on "Nobody Loves You" from the Kubark Manual. The Kubark Manual explains this pitch thusly:

"An interrogatee who is withholding items of no grave consequence to himself may sometimes be persuaded to talk by the simple tactic of pointing out that to date all of the information about his case has come from persons other than himself. The interrogator wants to be fair. He recognizes that some of the denouncers may have been biased or malicious. In any case, there is bound to be some slanting of the facts unless the interrogatee redresses the balance. The source owes it to himself to be sure that the interrogator hears both sides of the story."

Pitch 2: "I've been talking to several of your colleagues and they've indicated that they believe that you are the one that placed the back door on the system. They have also implied that perhaps you were the source of several other security problems [a bluff]. I just want to get the facts, and hear both sides of the story so that we can get this mess cleaned up ASAP."

The third pitch will be based upon the "The All-Seeing Eye (or Confession is Good for the Soul)" from the Kubark manual:

"The interrogator who already knows part of the story explains to the source that the purpose of the questioning is not to gain information; the interrogator knows everything already. His real purpose is to test the sincerity (reliability, honor, etc.) of the source. The interrogator then asks a few questions to which he knows the answers. If the subject lies, he is informed firmly and dispassionately that he has lied. By skilled manipulation of the known, the questioner can convince a naive subject that all his secrets are out and that further resistance would be not only pointless but dangerous. If this technique does not work very quickly, it must be dropped before the interrogatee learns the true limits of the questioner's knowledge."

Pitch 3: I already have everything I need to know from several very reliable sources. Now I want to hear everything from you. Don't skip any details. I know you were logged in from your girlfriends home account, that's been verified. Now, start at the beginning ..."

Pitch four is based upon "Ivan Is a Dope." This pitch appears to be applicable when there may be two suspects, or the suspect is working for someone. From the Kubark Manual:

"It may be useful to point out to a hostile agent that the cover story was ill-contrived, that the other service botched the job, that it is typical of the other service to ignore the welfare of its agents. The interrogator may personalize this pitch by explaining that he has been impressed by the agent's courage and intelligence. He sells the agent the idea that the interrogator, not his old service, represents a true friend, who understands him and will look after his welfare."

Pitch 4. I don't believe you story. It doesn't ring true. No one believes it. It's time to come clean. We know you are working for "Fred." Fred doesn't give a crap about you. He's let other workers go to jail, and you know what, he could not care less. We know you aren't the leader, you were just following orders. Why don't you start at the beginning and tell me everything. If you cooperate, I can see what I can do to lighten your sentence."

The final pitch is called "overstating the case." In this pitch the interrogator exaggerates what the suspect has believed to have done in hopes that the suspect will admit to only the lesser offense, and deny the larger one.

Pitch 5: "So now we find that not only did you install the Loki server on the system, but it appears that you also installed several logic bombs, as well as destroyed several backup tapes. We were looking at maybe a fine with the Loki server; however, with these additional charges you are looking at serious jail time."

References

Pfleeger, C.P., & Pfleeger, S.L. (2003). *Security in computing*. Prentice-Hall: New York.

Miller, J. (1988). *20 Years of Covert Channel Modeling and Analysis*. Unpublished Paper.

John Reid Interviewing Technique. http://www.reid.com/critic-technique.html.

The Kubark Manual. http://www.mindcontrolforums.com/kubark.htm

Phrack P51-06: LOKI2. http://www.phrack.org/phrack/51/P51-06

Whalley, I., Arnold, B., Chess, D., Morar, J., Segal, A., & Swimmer, M. (2001). *An Environment for Controlled Worm Replication and Analysis or: Internet-inna-Box*. http://www.research.ibm.com/antivirus/SciPapers/VB2000INW.htm

Merchant, C., & Stewart, J. (July 10, 2002). *Detecting and Containing IRC-Controlled Trojans: When Firewalls, AV, and IDS Are Not Enough*. LURHQ Corporation Secure Operations Center. http://www.megasecurity.org/Trojaninfo/IRC%20controled%20Trojans.html

Daemon9 (route) (August, 1996). Project Loki. *Phrack Magazine*. Volume Seven, Issue Forty-Nine. http://www.phrack.org/show.php?p=49&a=6

DOJ (July, 2002). *Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations*. Computer Crime and Intellectual Property Section, Criminal Division, United States Department of Justice. http://www.cybercrime.gov/s&smanual2002.htm

Acceptable Use Policies. http://www.rice.edu/armadillo/acceptable.html

Computer Fraud and Abuse Act. http://www.usdoj.gov/criminal/cybercrime/1030NEW.htm

Computer Crime and Intellectual Property Section (CCIPS), Federal Computer Intrusion. Laws http://www.usdoj.gov/criminal/cybercrime/cclaws.html

Nebraska Computer Crime Laws. http://www.phreak.org/archives/The_Hacker_Chronicles_II/law/ne_law.sta

Mandia, K, & Prosise, C. (2002). *Incident response: Investigating computer crime*. Osborne.

Part 2

Forensic Analysis of a Discarded University Computer System

ABSTRACT

This document describes the forensic analysis of a 'retired' university personal computer system. The system was a P5-2000 Gateway system with a three GB hard drive running Microsoft Windows 95 and DOS 6. The forensic procedures were comprised of an analysis of allocated, slack, and unallocated space. The results of the analysis suggested there had been five users with accounts (user ids) on the computer over a four year period of time, and that the computer had once been used by employees of the cashiering/student accounting department.

Below I provide a detailed explanation of these forensic procedures. The analysis resulted in the recovery of several personal as well as work-related documents, social security numbers, and user id/password combinations. Given the nature of the information found all identifying has been sanitized, or described in summary, in order to protect users' privacy, as well as those whose names are mentioned in the files.

Synopsis of Case Facts

Every year the university selectively replaces computers in various departments. The replacement of computers usually occurs on a three- to five- year rotating basis. The specific procedure for the replacement of computers is handled thusly:

1. A department, either academic or administrative, is identified as eligible for new computers.
2. Computer Services identifies a minimum set of requirements for a new computer system.
3. The order goes out for bid, and a vendor is subsequently selected.
4. New computers arrive.
5. A user, a departmental member, faculty member, or staff member from Computer Services is designated as responsible for preparing the old computer for transition from university use. (One might expect that this procedure would include a secure deletion of the hard drive would occur at this time.)
6. The old computers, including CPU, monitor, keyboard, and mouse, are taken to a warehouse.
7. At a later time the university holds auctions for the systems.

We requisitioned approximately 20 of these computers for the purpose of providing incoming freshmen and high school students with the opportunity of acquiring hands-on experience with computer hardware.

An inventory of these systems found all to be Pentium class machines, P5-200, 250, or 300 systems built around 1997-98. The hard drives were in the range of two to three GB in size.

I selected a system randomly for this assignment. Below I provide a detailed explanation of the forensic procedures that I applied to this system. The analysis resulted in the recovery of several personal as well as work-related documents, social security numbers, and user id/password combinations. Given the nature of the information found all identifying has been sanitized, or described in summary, to protect users' privacy, as well as those whose names are mentioned in the files.

Hardware

The hardware selected for this forensic analysis was the following:

Tag # Description
001 Gateway P5-2000, Pentium 1, 200MHz, SN 0007055495, MD: 4/30/97
002 Western Digital Caviar 32500, Model: WDAC32500-32H SN: WT3603304173

The system included:

- 17" Gateway monitor;
- 3.5" floppy drive ;
- Teac CD-ROM drive;
- internal sound card;
- internal Matrox graphics card;
- Microsoft mouse, and;
- Gateway keyboard.

Forensic Analysis Machine

The forensic machine is a dual boot machine that purchased specifically for forensic analysis with money from a modest grant. The machine is a P4-2.2GhZ system with two hard drives: an 80GB Western Digital hard drive for the operating systems, and a 200GB Western Digital forensic evidence hard drive, both of which spin at 7200 RPM. The Linux partition runs Redhat 8, and the Windows partition runs Windows XP. The evidence drive is partitioned into several different file systems, Linux-ext3, FAT32, or NTFS. The forensic machine includes a LG DVD-RW and a Hewlett-Packard CD-RW.

The hard drive was removed from the Gateway computer and tagged. I made a bit-copy (image) of the hard drive using an external USB hard drive mounter. The procedures employed in the imaging of the hard drive are described below.

Forensic Software

The Linux partition includes many forensic tools, including:

- The @stake Sleuth Kit (TASK) v. 1.60 (Carrier, 2003), a replacement and extension of The Coroner's Toolkit (Farmer & Venema, 1998) that supports the analysis of Linux (e.g., ext2 & 3) as well as FAT and NTFS file systems;
- The Autopsy Browser v. 1.70 (Carrier, 2003), a graphical front-end to TASK;
- mac_daddy (Lee, 2003), a Perl script for calculating MAC times;
- mac-robber, a command line utility for calculating MAC times (Carrier, 2000); and
- John the Ripper, a UNIX/NT password cracker.

The Windows partition includes

- The Forensic Toolkit (www.accessdata.com) ;
- EnCase (www.guidancesoftware.com)
- NTI's command line suite of forensic tools (www.nti.com);
- Registrar Lite, a utility for viewing Window's registry files; and
- PWLTOOL, a *.PWL password cracker (old Windows password files).

The forensic analysis required the use of both Linux and Window's forensic tools.

Image Media

The first step is to mount the hard drive read-only and then calculate the MD5 cryptographic hash.

md5sum /dev/sda1 > /sans.cert/evidence.md5

//initial step calculates the MD5 cryptographic hash

dd if=/dev/zero of=/dev/hdb2

// write a series of '0's to evidence/forensic hard drive to ensure that no information preexisting information remains

dd if=/dev/sda1 of=/mnt/evidence/uno.dispose.dd bs=1M

// create the bit-copy image with a block size of 1MB

md5sum uno.dispose.dd >> /sans.cert/evidence.md5

// calculate the cryptographic hash and compare to the hash of the original

cat ./evidence.md5

ef4e023769f1ff1ab895f37cf2205b52 /dev/sda1

ef4e023769f1ff1ab895f37cf2205b52 uno.dispose.dd

Analysis of the System

Evidence can be found anywhere on a disk. This includes allocated space, that is, in files residing on the disk, as well as unallocated space, when files have been deleted. Evidence may also be found in slack space; the space at the end of a cluster that a file does not use. If a file overwrites a previously used cluster, and the file does not use the entire cluster, then evidence within the slack space may be retrieved.

The procedures for identifying and retrieving evidence will differ depending upon where the evidence resides. Files in allocated space may be searched by mounting the disk image (read-only), and then using common search techniques, such as *find* or *grep* under UNIX, to search for specific file names or keywords within files, respectively. The unallocated space is a special animal because it can hold information from deleted files. Special forensic procedures are required to access this evidence. Unallocated space will be searched last.

Windows provides standard directories where it places files which it controls, such as cookies, recently access files, or temporary files, and so on. The outline below provides a brief description of this search space.

- Swap (win386.swp)

- Located anywhere
  - *.tmp files located anywhere (temporary files)
  - *.emf files located anywhere (e.g., print spool files)
  - *.txt (text)
  - inbox.mbx (mailbox files)
  - *.PWL (password files)
  - Usual files holding evidence
    - *.doc, *.wp5, *.ppt, *.xls, *.mba, etc.
- Internet history
- Windows registry files
  - System.dat
  - User.dat
- Any files in the following directories
  - C:\temp
  - C:\tmp
  - C:\windows\temp
  - C:\windows\spool
  - C:\windows\recent
  - C:\windows\cookies
  - C:\windows\favorites
  - C:\windows\Temporary Internet Files
  - C:\Program Files
  - C:\My Documents

This outline served as a blueprint for the forensic procedures and their description below.

Filesystem Overview

Specific information regarding the filesystem can be established using TASK's *fsstat* command. According to its man page, the *fsstat* command display details of a file system. The command indicated that the system was a FAT16 system with a cluster size of 32K, and contains the traditional two FATs (file allocation tables) and a root directory. Data begins at sector 537.

```
# fsstat -f fat16 uno.dispose.dd

FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT
OEM: MSWIN4.1
Volume ID: 595812119
Volume Label:
File System Type (super block): FAT16

META-DATA INFORMATION
```

```
----------------------------------------
Range: 2 - 65922050
Root Directory: 2

CONTENT-DATA INFORMATION
----------------------------------------
Sector Size: 512
Cluster Size: 32768
Sector of First Cluster: 537
Total Sector Range: 0 - 4120639
FAT 0 Range: 1 - 252
FAT 1 Range: 253 - 504
Data Area Sector Range: 505 - 4120639

// lines deleted which display the fat allocation
```

Note the large cluster size: each allocation unit is comprised of 64 sectors of size 512bytes. This is the largest cluster size recommended by Microsoft. Forensically speaking, the larger the cluster size, the greater chance of finding interesting tidbits of evidence in the slack space. Recall that the allocation unit, i.e., the cluster size, is the smallest amount of disk space that any file (of any size) will be allocated. Thus, a file comprised of a single ASCII character would be allocated a whopping 32768 byte cluster. Accordingly, on average, each file will waste approximately 1/2 of its allocation unit. This will be important when slack space is analyzed later.

Timeline Analysis

I created a timeline first for two reasons. The first reason was to create an exhaustive list of every file on the system. The second reason was to establish who was using the programs and files, at what time, and so on. I used command line utilities from the @stake Sleuth kit (TASK, v1.6). Creating a timeline using TASK is a three step process First *fls* is run against the bit-copy. According to its man page *fls* creates a timeline listing for existing files:

```
# fls -rf fat16 -m / uno.dispose.dd > /sans.cert/uno.timeline
```

The -r flag indicates that *fls* should work recursively through the files, the -f flag indicates the type of file system (fat16), and -m indicates that the MAC (modified, accessed, creagted) times should be generated.

The second command run is *ils*. According to its man page *ils* lists inode information (or in this case, directory entry listings):

```
# ils -f fat16 -m uno.dispose.dd >> /sans.cert/uno.timeline
```

The results of this command are appended to the results from *fls*. The final step is to format the results in a human-readable format, done so by the *mactime* utility. According to its man page *mactime* creates an ASCII time line of file activity. The exact command line is:

mactime -b /sans.cert/uno.timeline

The -b indicates the body file to be used that contains the derived MAC times, in this case it is the combined output from the *fls* and *ils* command. If an investigator is interested in only a specific time period, say Jan 1 2003 until April 4, 2003, then a specific time period can be specified. For example:

mactime -b /sans.cert/uno.timeline 1/1/2003-4/1/2003

Because I was not interested in constraining the timeline I included no constraints.

The entire timeline is attached as a zipped file (it is several hundred pages long).

Timeline Results

The timeline begins on Wed Dec 31 1969 18:00:00, clearly an impossible date for any PC timeline given that the first PCs did not exist until 1981. According to the timeline 1537 files were created (and a few accessed) at this date and time, which if not for the highly incorrect date, might lead one to infer that this was the installation time. Because the system clock was not correctly set during installation it is impossible to determine the actual date and time of installation.

Wed Dec 31 1969 18:00:00

```
48640 ..c -/-rwxrwxrwx 0  0  4230668  /WINDOWS/SYSTEM/VIEWERS/VSAMI.DLL
   545 ..c -/-rwxrwxrwx 0  0  671276   /WINDOWS/DOSPRMPT.PIF
78965 ..c -/-rwxrwxrwx 0  0  980528   /WINDOWS/SYSTEM/VMM32/_HELL.VXD (deleted)
2629 .ac -/-rwxrwxrwx  0  0  4106769   /PROGRA~1/ONLINE~1/AT&T/SIGNBTN.GIF
4336 ..c -/-rwxrwxrwx   0  0    706409   /WINDOWS/SYSTEM/ADDREG.EXE
55920 ..c -/-rwxrwxrwx 0  0    706058  /WINDOWS/SYSTEM/MSPRINT.DLL
69194 ..c -/-rwxrwxrwx 0 0          603  /DOS/MSBACKFR.OVL
    57 .ac -/-rwxrwxrwx 0 0   3310093  /WINDOWS/SHELLNEW/WORDPFCT.WPG
88064 ..c -/-rwxrwxrwx 0 0     706308 /WINDOWS/SYSTEM/AVIFIL32.DLL
18999 .ac -/-rwxrwxrwx 0 0  1157651 /WINDOWS/SYSTEM/IOSUBSYS/HSFLOP.PDR
90521 ..c -/-rwxrwxrwx 0 0     980526 /WINDOWS/SYSTEM/VMM32/_ONFIGMG.VXD (deleted)
```

5632 ..c -/-rwxrwxrwx 0 0 706311 /WINDOWS/SYSTEM/DCIMAN32.DLL

282112 ..c -/-rwxrwxrwx 0 0 706348 /WINDOWS/SYSTEM/D3DRM.DLL

11264 ..c -/-rwxrwxrwx 0 0 706182 /WINDOWS/SYSTEM/AWRAMB32.DLL

17993 ..c -/-rwxrwxrwx 0 0 980490 /WINDOWS/SYSTEM/VMM32/_BIOS.VXD (deleted)

[HUNDREDS OF LINES DELETED]

The second date appearing is Fri Jan 04 1980 00:00:00. Again, this would be an incorrect date. There are several hundred files with this date/timestamp.

The fact that the system clock was set incorrectly several times will make it difficult, and impossible in some cases, to make valid inferences regarding system usage. This anomaly is discussed later in more depth.

Password Files

Perhaps the most interesting forensic questions are: who was using the system, and for what purposes? Answering these questions requires establishing who the users were. Unfortunately, Windows 95 does not partition the user space as does NT/2000, and XP, therefore, other means are required to establish user identity. This was accomplished by examining the password files found in the Windows directory.

There were five password files (Windows *.PWL files) in the *\windows* directory. Because different users would leave different fingerprints on the filesystem I a) cracked the passwords and b) established the MAC times for the password files, which would allow me to establish when the user first started on the system (the creation date of the *.PWL file), when the user last modified his/her password (the modification date of the *.PWL), and the last time a user logged onto the system, given by the access time.

Users (sanitized) user IDs are:

User ID	Password
T*****	T******* (same as username)
H***	“1234”
S*****	“welcome”
TW**	“blockout”
V*****	FAILED

Four of the five passwords were broken with a dictionary attack, using a dictionary with over 3 million entries. The fifth password (user V) proved more difficult. When the dictionary attack failed for user V I subsequently ran several brute force attacks, including:

1. Maximum of four characters using the character set of upper/lower case letters and numbers

2. Same as #1 only adding special characters
3. Maximum of five characters using the character set of upper/lower case letters, numbers and special characters
4. Maximum of 8 characters using "smart force" search (heuristically driven search for plausible combinations of letters.)
5. Numbers 0-9 with 9 letters, hoping that user V used her social security number.

None of the aforementioned attempts succeeded in breaking the password for user V.

User T's PWL password file included several cached passwords:

Server:'UNOBFSRV'
Password:'t*******'
Share path:'WWW\HTMLCASH'
Password:'zyac074'
Mail:'MAPI'
Password:'MAPI'

The 'zyac074' password will reappear multiple times later when I analyze the unallocated and slack space of the hard drive.

I was able to establish the entire user's first and last names from various documents and files remaining on the hard drive. The names will not be divulged due to privacy concerns.

User Timeline

Next I used the MAC times of the password files to establish when each of the users was first and last logged onto the system. I speculated that the creation date of each password file would be a good guess as to the initial date that the user first logged in, and that the access time would be the last time the user logged in. Each of the users is considered below in the (apparent) order in which they gained accessed to the computer.

User T

Wed Dec 03 1997 13:39:38 758 ..c -/-rwxrwxrwx 0 0 671421 /WINDOWS/T.PWL

Tue Feb 24 1998 16:00:16 758 m.. -/-rwxrwxrwx 0 0 671421 /WINDOWS/T.PWL

Tue Jun 23 1998 01:00:00 758 .a. -/-rwxrwxrwx 0 0 671421 /WINDOWS/T.PWL

User T was the first account created on the computer, in December of 1997. She changed her password on Feb 24, 1998, and last logged in on June 23 1998. She did not change her password for 5 months. At some point in time user T changed her last name, based upon the fact that her last name as indicated on her user ID was different from the name she used to sign documents.

User TW

Tue Jun 23 1998 16:50:18 688 ..c -/-rwxrwxrwx 0 0 671627 /WINDOWS/TW.PWL

Tue Jun 23 1998 16:50:34 688 m.. -/-rwxrwxrwx 0 0 671627 /WINDOWS/TW.PWL

Tue Jul 07 1998 01:00:00 688 .a. -/-rwxrwxrwx 0 0 671627 /WINDOWS/TW.PWL

User TW's account was created on June 23, 1998. He/she last logged in on July 7 of 1998, and the modification date indicates that he/she never changed her/his password. TW accessed the computer for a mere three weeks during the summer of 1998. I could find no evidence on the evidence drive that could be attributed to TW.

User S

Tue Jun 30 1998 12:10:32 730 ..c -/-rwxrwxrwx 0 0 671630 /WINDOWS/S.PWL

Mon Dec 07 1998 08:10:10 730 m.. -/-rwxrwxrwx 0 0 671630 /WINDOWS/S.PWL

Sun Dec 27 1998 00:00:00 730 .a. -/-rwxrwxrwx 0 0 671630 /WINDOWS/S.PWL

User S's password was created on June 30, 1998. She modified the password on December 7, 1998, and was last logged in on December 27. Interestingly, user H's account was created the same day (see below).

User V

Fri Jan 04 1980 00:00:00 706 .a. -/-rwxrwxrwx 0 0 671651 /WINDOWS/V.PWL

Fri Jan 04 1980 17:01:58 706 m.. -/-rwxrwxrwx 0 0 671651 /WINDOWS/V.PWL

Sun Jan 24 1999 21:08:58 706 ..c -/-rwxrwxrwx 0 0 671651 /WINDOWS/V.PWL

The last time that user V, a female, logged in is speculative. Clearly the date is set incorrectly. It does indicate that user V was the last account to be created on the computer, Sun Jan 24, 1999, which interestingly, is the last time that user H was logged in.

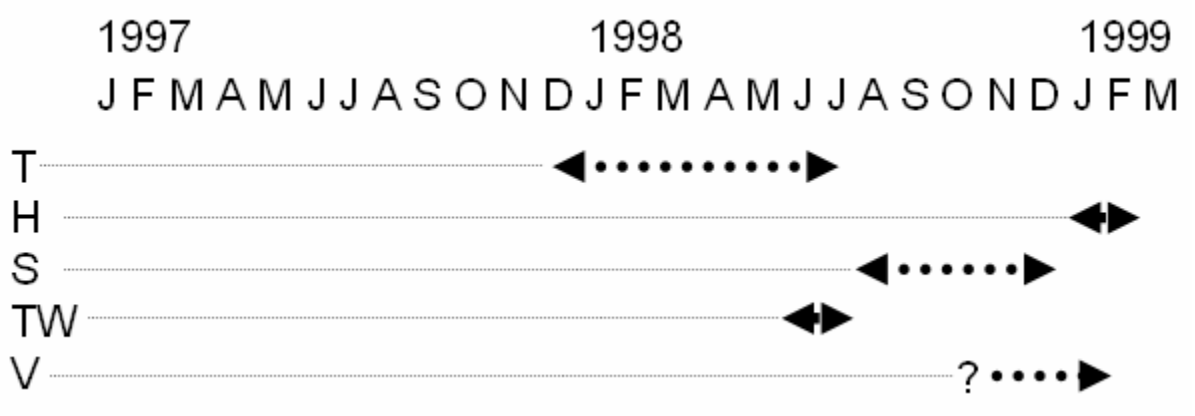
User H

Sun Dec 27 1998 14:18:44 688 ..c -/-rwxrwxrwx 0 0 671262 /WINDOWS/H.PWL

Sun Dec 27 1998 14:18:46 688 m.. -/-rwxrwxrwx 0 0 671262 /WINDOWS/H.PWL

Sun Jan 24 1999 00:00:00 688 .a. -/-rwxrwxrwx 0 0 671262 /WINDOWS/H.PWL

User H's account was created Dec 27, 1998, and was last logged in Sun Jan 24, 1999. According to the modification date he never changed his password.



This timeline clearly indicates that the two users with the most access over time were user T and user S.

Operating System

I mounted the disk image in read-only mode and did some reconnaissance to determine what programs or files remained.

mount -t vfat -o ro,loop,noexec,noatime,nodev /dev/sda1 /mnt/evidence

I quickly found that the hard drive contained what appeared to be the entire Windows 95 application. It also contained Microsoft Office 97, Lotus Notes, several games, as well as several dozen files of possible evidentiary value found in several locations. The operating system and applications versions were established by viewing several text files, or using a hex viewer to view documents to view the signatures, or viewing binary files.

- Windows 95
- DOS 6
- Internet Explorer 4
    - Microsoft Office 8
    - Word 8
    - Excel 8
    - PowerPoint 8
    - Access 7.53
- Lotus Notes Client 4.6
- Several games

Start-up Files

Start-up files were examined for any anomalous entries. Start-up files included:

- autoexec.bat (root directory)
- config.sys (root directory)

- win.ini (c:\windows directory)
- system.ini c:\windows directory)

Nothing strange or interesting was found in any of the start-up files.

Program Files Directory

The “Program Files” directory contained a full compliment of software. An examination of each of the directories found that what appeared to be the full installation of each software package.

```
[jpc@whammo Program Files]# ls -al
total 608
dr-xr-xr-x    19 32768 Dec  3  1997 .
drwxr-xr-x   22 16384 Dec 31  1969 ..
dr-xr-xr-x     3 32768 Dec  3  1997          Accessories
drwxr-xr-x     6 32768 Dec  5  1997          Common Files
drwxr-xr-x     3 32768 Dec  5  1997          Dr Solomon's
drwxr-xr-x     2 32768 Dec  3  1997          ICW-Internet Connection Wizard
dr-xr-xr-x     5 32768 Dec  3  1997          Internet Explorer
drwxr-xr-x     3 32768 Jul  2  1998          Internet Mail and News
drwxr-xr-x    12 32768 Dec 11  1997          Microsoft FrontPage
drwxr-xr-x     6 32768 Dec  5  1997          Microsoft Office
dr-xr-xr-x     2 32768 Dec  3  1997          NetMeeting
drwxr-xr-x     5 32768 Dec  3  1997          Online Services
drwxr-xr-x     7 32768 Dec  4  1997          Personal Communications
dr-xr-xr-x    2 32768 Dec  3  1997          The Microsoft Network
drwxr-xr-x 13 32768 Dec  5  1997          Uninstall Information
dr-xr-xr-x    2 32768 Dec  3  1997          Windows Messaging
drwxr-xr-x    2 32768 Dec 18  1998          backpack
drwxr-xr-x    3 32768 Dec  3  1997          plus!
drwxr-xr-x    6 32768 Dec 11  1997          websvr
```

Program Installation Dates

Windows and DOS Installation

It was clear from the MAC timeline that the system clock was quite often set incorrectly. Recall that the first time on the timeline is Wednesday, December 31, 1969. The fact that there were many Window's system files and directories created at this time might lead one to believe that this was when Windows was first installed.

Of course, Windows could not have been installed in 1969. I suspect that the Gateway computer came with DOS and Windows installed as they usually do. I also would expect that the workers at Gateway who perform these installations would correctly set the system clock. Interestingly, the modification dates of the Windows installation files were December 3, 1997 (see below), which could mean that Windows 95 was updated on that date.

```
Wed Dec 03 1997 11:56:46  32768 m.. d/drwxrwxrwx  0 0         6 /DOS
Wed Dec 03 1997 12:02:44        71 m.. -/-rwxrwxrwx    0 0 242184   /PCD/CONFIG.SYS
Wed Dec 03 1997 12:11:30  32768 m.. d/drwxrwxrwx  0 0        11 /PCD
Wed Dec 03 1997 12:11:34        99 m.. -/-rwxrwxrwx   0 0         9 /CONFIG.SYS
Wed Dec 03 1997 12:11:36        98 m.. -/-rwxrwxrwx   0 0        10 /AUTOEXEC.BAT
Wed Dec 03 1997 12:25:48        22 m.. -/-r-xr-xr-x       0  0         5  /MSDOS.---
Wed Dec 03 1997 12:54:34  32768 m.. d/drwxrwxrwx 0  0        18  /WINDOWS
Wed Dec 03 1997 12:54:56    2389 m.. -/-rwxrwxrwx   0  0        20  /NETLOG.TXT
Wed Dec 03 1997 12:55:14  32768 m.. d/dr-xr-xr-x    0  0  671237   /WINDOWS/INF
Wed Dec 03 1997 12:55:16
      32768 m.. d/drwxrwxrwx 0  0 671239  /WINDOWS/COMMAND
      32768 m.. d/drwxrwxrwx 0  0  671238   /WINDOWS/SYSTEM
       2474 m.. -/-rwxrwxrwx   0   0  706053   /WINDOWS/SYSTEM/DLCNDI.DLL
```

Accoring to web site ComputerHope (http://www.computerhope.com/whow.htm) Windows 95 had several updates:

- Windows 95 4.00.950 Original release of Windows 95.
- Windows 95 4.00.950 A Original release of Windows 95 with Service Pack 1 or OEM Service Release 1.
- Windows 95 4.00.950 B Second release of Windows 95 / OSR2 (Has Fat32).
- Windows 95 4.00.950 C Second release of Windows 95 / OSR2 with FAT32, USB and AGP support.

Therefore, it is likely that the modification dates of December 3, 1997 are most likely one of these Windows 95 updates.

Office 97 Installation

Office 97 was installed on Friday, July 11, 1997. A portion of the listing of the timeline associated with the installation is provided below.

Fri Jul 11 1997 01:00:00

5324560 m.c -/-rwxrwxrwx 0 0 2228780 /PROGRA~1/MICROS~1/OFFICE/WINWORD.EXE
34806 mac -/-rwxrwxrwx 0 0 7143966 /PROGRA~1/MICROS~1/CLIPART/POPULAR/CHAMPGNE.WMF
114706 m.c -/-rwxrwxrwx 0 0 2228835 /PROGRA~1/MICROS~1/OFFICE/XLMAIN8.CNT
9878 mac -/-rwxrwxrwx 0 0 7143963 /PROGRA~1/MICROS~1/CLIPART/POPULAR/BRICK.WMF
500736 m.c -/-rwxrwxrwx 0 0 2228869 /PROGRA~1/MICROS~1/OFFICE/MSAIN800.DLL
5604624 m.c -/-rwxrwxrwx 0 0 2228821 /PROGRA~1/MICROS~1/OFFICE/EXCEL.EXE
1597200 m.c -/-rwxrwxrwx 0 0 7053847 /PROGRA~1/COMMON~1/MICROS~1/VBA/VBA332.DLL
18402 mac -/-rwxrwxrwx 0 0 2228904 /PROGRA~1/MICROS~1/OFFICE/OFREAD8.TXT
[HUNDREDS OF LINES DELETED]

Internet Explorer 4 Update or Installation

Internet Explorer 4 was installed, or version 3 was updated, on Thursday, Sept 18, 1997. Only a portion of the MAC timeline is provided below.

Thu Sep 18 1997 01:00:00
92672 m.c -/-rwxrwxrwx 0 0 706233 /WINDOWS/SYSTEM/URL.DLL
6144 m.c -/-rwxrwxrwx 0 0 706537 /WINDOWS/SYSTEM/COMCAT.DLL
17075 m.c -/-rwxrwxrwx 0 0 706731 /WINDOWS/SYSTEM/IE4UINIT.INF
486000 m.c -/-rwxrwxrwx 0 0 706792 /WINDOWS/SYSTEM/IE4TOUR.DLL
46746 mac -/-rwxrwxrwx 0 0 738924 /WINDOWS/HELP/IEXPLORE.HLP
35232 mac -/-rwxrwxrwx 0 0 2063888 /PROGRA~1/INTERN~1/ACTSETUP.EXE
1478 m.c -/-rwxrwxrwx 0 0 686855 /WINDOWS/INF/INETFIND.INF
83200 m.c -/-rwxrwxrwx 0 0 706787 /WINDOWS/SYSTEM/CRYPTDLG.DLL
22528 m.c -/-rwxrwxrwx 0 0 706742 /WINDOWS/SYSTEM/CFGMGR32.DLL
482576 m.c -/-rwxrwxrwx 0 0 706549 /WINDOWS/SYSTEM/JSCRIPT.DLL
18371 mac -/-rwxrwxrwx 0 0 11065864 /PROGRA~1/INTERN~1/SETUP/IE4.INF
77728 m.c -/-rwxrwxrwx 0 0 706759 /WINDOWS/SYSTEM/IETIMER.OCX
106352 m.c -/-rwxrwxrwx 0 0 2063887 /PROGRA~1/INTERN~1/IE4.DLL
106352 mac -/-rwxrwxrwx 0 0 11065866 /PROGRA~1/INTERN~1/SETUP/IE4.DLL

Lotus Notes Installation

Lotus Notes 4.6 was installed on Monday, September 15, 1997. Only a small portion of the MAC timeline is provided below.

Mon Sep 15 1997 20:08:48
13824 m.c -/-rwxrwxrwx 0 0 25471633 /NOTES/nntediff.dll (NNTEDIFF.DLL)
Mon Sep 15 1997 20:08:52
47104 m.c -/-rwxrwxrwx 0 0 25471911 /NOTES/naldaemn.exe (NALDAEMN.EXE)
Mon Sep 15 1997 20:08:54
95232 m.c -/-rwxrwxrwx 0 0 25471527 /NOTES/namgr.exe (NAMGR.EXE)
Mon Sep 15 1997 20:08:56
4096 m.c -/-rwxrwxrwx 0 0 25471937 /NOTES/namhook.dll (NAMHOOK.DLL)
Mon Sep 15 1997 20:09:00
28672 m.c -/-rwxrwxrwx 0 0 25471607 /NOTES/ltssb01.dll (LTSSB01.DLL)
Mon Sep 15 1997 20:09:02
63488 m.c -/-rwxrwxrwx 0 0 25471919 /NOTES/nwrdaemn.exe (NWRDAEMN.EXE)
Mon Sep 15 1997 20:09:08
43520 m.c -/-rwxrwxrwx 0 0 25471651 /NOTES/nchronos.exe (NCHRONOS.EXE)
Mon Sep 15 1997 20:09:16
11264 m.c -/-rwxrwxrwx 0 0 25471695 /NOTES/ndbnotes.dll (NDBNOTES.DLL)
Mon Sep 15 1997 20:09:18
5120 m.c -/-rwxrwxrwx 0 0 25471691 /NOTES/nntcheck.dll (NNTCHECK.DLL)
Mon Sep 15 1997 20:09:26
40960 m.c -/-rwxrwxrwx 0 0 25471711 /NOTES/getgroup.exe (GETGROUP.EXE)
Mon Sep 15 1997 20:11:46
702464 m.c -/-rwxrwxrwx 0 0 25471537 /NOTES/neditfax.dll (NEDITFAX.DLL)
Mon Sep 15 1997 20:12:10
36864 m.c -/-rwxrwxrwx 0 0 25471609 /NOTES/mksyd.exe (MKSYD.EXE)

// hundreds of additional lines deleted

Last Days of the Computer

According to the MAC timeline the last day the computer was accessed was Wednesday, December 29, 1999. (As will be demonstrated later, this simply cannot be the case, as a letter is found whose contents describe events transpiring in November of 2000). On that day there is evidence that someone was manipulating some of the personal files on the computer. Although one cannot put too much faith in the dates and times on these files, it appears that someone was working on Christmas day. A description of the nature of these personal files is provided later.

Fri Dec 17 1999 00:00:00
33792 .a. -/-rwxrwxrwx 0 0 14851598 /DATA/To The Honorable Senators Kerry and Hagel.doc
Fri Dec 17 1999 21:26:58
33792 ..c -/-rwxrwxrwx 0 0 14851598 /DATA/To The Honorable Senators Kerry and Hagel.doc
Fri Dec 17 1999 21:27:00
33792 m.. -/-rwxrwxrwx 0 0 14851598 /DATA/To The Honorable Senators Kerry and Hagel.doc
38912 .a. -/-rwxrwxrwx 0 0 14851601 /DATA/To my dearest Son.doc
Sat Dec 25 1999 01:06:16
38912 ..c -/-rwxrwxrwx 0 0 14851601 /DATA/To my dearest Son.doc
Sat Dec 25 1999 10:53:16
38912 m.. -/-rwxrwxrwx 0 0 14851601 /DATA/To my dearest Son.doc
Wed Dec 29 1999 00:00:00
22016 .a. -/-rwxrwxrwx 0 0 14851603 /DATA/Dear Bill.doc
Wed Dec 29 1999 05:51:38
22016 ..c -/-rwxrwxrwx 0 0 14851603 /DATA/Dear Bill.doc
Wed Dec 29 1999 05:51:40
22016 m.. -/-rwxrwxrwx 0 0 14851603 /DATA/Dear Bill.doc

FILE ANALYSIS

Windows Swap File

The Window's swap file was found in C:\windows\win386.swp. Unfortunately, the file was 0 bytes, and therefore contained no data. Data previously contained within the swap file might still be accessible in the unallocated or slack space.

C:\windows\temp

The Window's \temp directory contained several files, including eight *.tmp files. However, each was 0 bytes, and therefore, provided no evidentiary information.

-rwxr-xr-x 1 root root 0 Apr 16 1999 ~dfa37.tmp
-rwxr-xr-x 1 root root 0 Jan 4 1980 ~dfc458.tmp
-rwxr-xr-x 1 root root 0 Apr 16 1999 ~dff613.tmp
-rwxr-xr-x 1 root root 0 Mar 3 1999 offA0B1.TMP
-rwxr-xr-x 1 root root 0 Mar 3 1999 offA0B4.TMP
-rwxr-xr-x 1 root root 0 Mar 3 1999 offB1D5.TMP

```
-rwxr-xr-x   1 root     root            0 Dec 30  1998 offC2F5.TMP
-rwxr-xr-x   1 root     root            0 Dec 30  1998 offC302.TMP
```

Root Directory

The root directory contained many files of possible evidentiary value. These files are identified in bold. The dates associated are the creation dates (from the –c flag on the ls command).

```
# ls –lct

drwxr-xr-x   2 root     root        32768 Apr 16  1999 mc3
drwxr-xr-x   4 root     root        32768 Apr 16  1999 sl
-rwxr-xr-x   1 root     root        32768 Apr 16  1999 shroud.doc
-rwxr-xr-x   1 root     root        19456 Dec 29  1998 Doc1.doc
-rwxr-xr-x   1 root     root           65 Dec 18  1998    config.w40
-r-xr-xr-x   1 root     root         1652 Dec 18  1998    msdos.w40
drwxr-xr-x   3 root     root        32768 Dec 18  1998 ~mssetup.t
drwxr-xr-x   2 root     root        32768 Dec 15  1998 shtlman
drwxr-xr-x   2 root     root        32768 Dec 15  1998 epiacd.w95
-rwxr-xr-x   1 root     root        70577 Dec 15  1998 detlog.txt
-rwxr-xr-x   1 root     root            0 Dec  9  1998 config.bak
drwxr-xr-x   2 root     root        32768 Sep 17 1998 rockymt
-rwxr-xr-x   1 root     root          187 Jul  7  1998 Autoexec.ttw.bat
-rwxr-xr-x   1 root     root          238 Jul  7  1998 Config.ttw
-rwxr-xr-x   1 root     root          183 Jul  7  1998 autoexec.syd
-rwxr-xr-x   1 root     root          234 Jul  7  1998 Config.sys.syd
drwxr-xr-x   6 root     root        32768 Jul  2  1998 notes
-rwxr-xr-x   1 root     root        35665 Jul  2  1998 uninst.log
-rwxr-xr-x   1 root     root          154 Jun 30  1998 Autoexec.old.bat
drwxr-xr-x   2 root     root        32768 Jan  8  1998 ewan
drwxr-xr-x   2 root     root        32768 Dec 11  1997 temp
drwxr-xr-x   2 root     root        32768 Dec 11  1997 data
drwxr-xr-x   5 root     root        32768 Dec 11  1997 webshare
drwxr-xr-x   3 root     root        32768 Dec  9  1997 internet
drwxr-xr-x   2 root     root        32768 Dec  9  1997 My Documents
```

```
-rwxr-xr-x   1 root     root        21669 Dec  8  1997 nec_ide.sys
-rwxr-xr-x   1 root     root        25361 Dec  8  1997 mscdex.exe
drwxr-xr-x   2 root     root        32768 Dec  8  1997 bin
drwxr-xr-x   2 root     root        32768 Dec  8  1997 dev
-rwxr-xr-x   1 root     root        88484 Dec  5  1997 detlog.old
-rwxr-xr-x   1 root     root        62901 Dec  5  1997 findviru.rpt
-rwxr-xr-x   1 root     root          483 Dec  5  1997 autoexec.bak
-rwxr-xr-x   1 root     root          485 Dec  5  1997 autoexec.w40
drwxr-xr-x   2 root     root        32768 Dec  3  1997 recycled
-rwxr-xr-x   1 root     root        59672 Dec  3  1997 setuplog.txt
-rwxr-xr-x   1 root     root         5168 Jan  4  1980 ffastun.ffa
-rwxr-xr-x   1 root     root       167936 Jan  4  1980 ffastun.ffo
-rwxr-xr-x   1 root     root       364544 Jan  4  1980 ffastun0.ffx
-rwxr-xr-x   1 root     root        98304 Jan  4  1980 ffastun.ffl
-rwxr-xr-x   1 root     root        16896 Jan  4  1980 My Dearest Darling.doc
dr-xr-xr-x  19 root     root        32768 Jan  1  1980 Program Files
-rwxr-xr-x   1 root     root           98 Jan  1  1980 autoexec.bat
-rwxr-xr-x   1 root     root        24767 Jan  1  1980 bootlog.prv
-rwxr-xr-x   1 root     root        24767 Jan  1  1980 bootlog.txt
-r-xr-xr-x   1 root     root        54619 Jan  1  1980 command.com
-rwxr-xr-x   1 root     root        93812 Jan  1  1980 command.w40
-rwxr-xr-x   1 root     root           99 Jan  1  1980 config.sys
drwxr-xr-x   2 root     root        32768 Jan  1  1980 dos
-rwxr-xr-x   1 root     root        98304 Jan  1  1980 file0001.chk
-rwxr-xr-x   1 root     root        32768 Jan  1  1980 file0002.chk
-rwxr-xr-x   1 root     root        32768 Jan  1  1980 file0003.chk
-rwxr-xr-x   1 root     root        98304 Jan  1  1980 file0004.chk
-rwxr-xr-x   1 root     root        32768 Jan  1  1980 file0005.chk
-rwxr-xr-x   1 root     root        32768 Jan  1  1980 file0006.chk
-rwxr-xr-x   1 root     root       524288 Jan  1  1980 file0007.chk
-r-xr-xr-x   1 root     root        40566 Jan  1  1980 io.sys
-rwxr-xr-x   1 root     root        29078 Jan  1  1980 logo.sys
-rwxr-xr-x   1 root     root           22 Jan  1  1980 msdos.---
-r-xr-xr-x   1 root     root        38138 Jan  1  1980 msdos.sys
-rwxr-xr-x   1 root     root         2389 Jan  1  1980 netlog.txt
drwxr-xr-x   2 root     root        32768 Jan  1  1980 pcd
-rwxr-xr-x   1 root     root          447 Jan  1  1980 scandisk.log
```

```
-r-xr-xr-x    1 root     root          5166 Jan  1  1980 suhdlog.dat
-r-xr-xr-x    1 root     root        517892 Jan  1  1980 system.1st
-rwxr-xr-x    1 root     root          9349 Jan  1  1980 wina20.386
-r-xr-xr-x    1 root     root        214836 Jan  1  1980 winboot.sys
drwxr-xr-x   34 root     root         32768 Jan  1  1980 windows
```

The files in bold were considered of potential 'evidentiary value.' That is, files that could tell possibly tell me something significant about the use of the computer. In some instances this guess was correct, and others not.

The contents of the *autoexec.bat* and associated backups appeared normal, as did the contents of the *config.sys* files and its associated backups.

The **.chk* files are lost clusters that are retrieved by DOS *chkdsk* utility. These were viewed with a hex viewer. Nothing of interest was found in any of the seven files.

The *bootlog.txt* file consisted of several hundred lines showing the drivers loaded on bootup.

```
[001553C3] Loading Device = \DEV\SLCD.SYS
[001553FC] LoadFailed     = \DEV\SLCD.SYS
[001553FC] Loading Device = C:\WINDOWS\HIMEM.SYS
[001553FE] LoadSuccess    = C:\WINDOWS\HIMEM.SYS
[001553FE] Loading Device = C:\WINDOWS\IFSHLP.SYS
[00155400] LoadSuccess    = C:\WINDOWS\IFSHLP.SYS
[00155400] Loading Device = C:\WINDOWS\SETVER.EXE
[00155402] LoadSuccess    = C:\WINDOWS\SETVER.EXE
[00155408] C:\BIN\MSCDEX.EXE[00155408]  starting
[Several Hundred Lines Deleted]
```

Nothing obvious was found regarding the possibility of Trojans or backdoors being loaded, of course, only a more thorough examination including a dynamic analysis, would be required to validate this assumption.

The directory *\webshare* contained some work-related web sites. Apparently the users of this computer were responsible for maintaining a portion of the web site for cashiering/student accounting.

The directories *\data* and *\My Documents* contained a wealth of information. The *\data* directory contained highly personal – non work related – files, whereas the *\My Document* directory contained a dozen or so Access databases, several of which contained personal identifying information. The contents of these directories are described below.

Two Microsoft Word files were found in the root directory: *shroud.doc* and *Doc1.doc*. Although the document's properties indicated that the author of the document was user

S, its creation time indicates that the user associated with this file is user V, the only apparent user at the time.

```
Fri Apr 16 1999 21:17:48    32768 ..c -/-rwxrwxrwx  0  0    82       /shroud.doc
Fri Apr 16 1999 21:17:50    32768 m.. -/-rwxrwxrwx 0  0    82       /shroud.doc
```

The file contained a description of the Shroud of Turin. The title of the document is "Hungry for His Presence," and according to text within the document, was downloaded from http://www.hungryweb.net.

The document Doc1.doc was an empty document.

Word 97 GUID

Microsoft Word 97 included a Globally Unique Identifier in each Word document. This ID could be used to trace a document to a particular copy of Word. The GUID for the copy of Word used to create the documents on this hard drive was established through the use of a hex editor to view the Word documents:

GUID CFC69D43-56AC-11BD-8B0D-E666CGC5E42B

Contents of c:\data\

A user created a directory called *\data* and placed several personal files here. Except for a single file (KEYS.doc) each file was of a personal nature. The files are listed below.

```
[jpc@whammo data]# ls -al
total 512
drwxr-xr-x    2 32768        Dec 11  1997 .
drwxr-xr-x   22 16384        Dec 31  1969 ..
-rwxr-xr-x    1 22016        Dec 29  1999 Dear Bill.doc
-rwxr-xr-x    1 38912        Dec 29  1998 KEYS.doc
-rwxr-xr-x    1 78848        Jul  8  1998 PROMISSORY NOTE.doc
-rwxr-xr-x    1 21504        Jan  4  1980 The Story.doc
-rwxr-xr-x    1 22528        Jan  4  1980 The.doc (empty file)
-rwxr-xr-x    1 33792        Dec 17  1999 To The Honorable Senators Kerry and Hagel.doc
-rwxr-xr-x    1 38912        Dec 25  1999 To my dearest Son.doc
-rwxr-xr-x    1 35840        Jan  4  1980 faith.doc
```

The contents of the files are as follows:

Dear Bill.doc

Although the Word properties dialog indicates that the author of this document is user S, this information doesn't fit with the MAC timelines. The MAC times for this document indicate the file was created, modified, and last accessed on the last day the computer was accessed. However, the only user accessing the system at that time was user H.

The author of the letter describes his/her current financial situation, and requests a loan of $2400 from Bill. The author admits he/she gambles frequently at the casinos, and apparently has won a lot of money.

To my dearest Son.doc

This is a letter from a father to a son in which the father discusses similarities in a number of painful moments throughout their lives. Because the author is male and the MAC times associated with the letter, the apparent author of the letter is user H.

To The Honorable Senators Kerry and Hagel.doc

This is a letter to two Nebraska U.S. Senators. The letter, signed by user V, relates the fact that she is the mother of a daughter who was subject of a police investigation. User V accuses the local police department of improper actions based on a search of her daughter's house.

PROMISSORY NOTE.doc

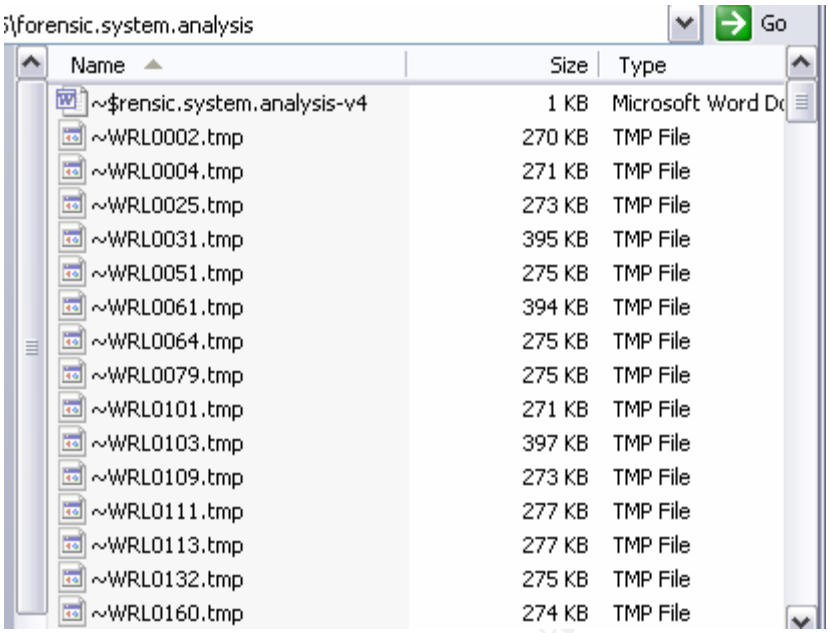
This is a template letter from student accounting to be completed by students owing money to the university.

The Story.doc

A story about growing up in Nebraska.

faith.doc

This is a consoling letter written to a family who lost a child. Dozens of copies of this letter were found in unallocated as well as slack space. This was most likely because it was saved multiple times by the author: Explanation: Each time a file is saved Microsoft Word creates a temporary file. The figure below illustrates this phenomenon as I'm writing this document.



Once the document is closed Word deletes the temporary files. However, this information remains on the hard drive until overwritten.

Contents of C:\My Documents

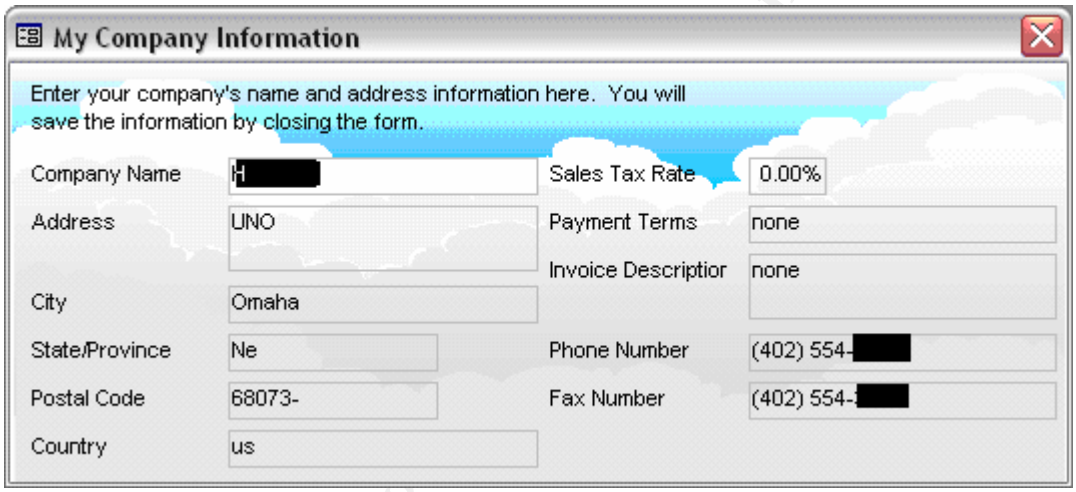
Eighteen files remained in the *\My Documents* directory. Each file was a Microsoft Access database. One of the databases was named after the first name of User H.

```
drwxr-xr-x   2 32768       Dec  9  1997 .
drwxr-xr-x  22 16384       Dec 31  1969 ..
-rwxr-xr-x   1 628736      Jan  4  1980  Asset Tracking1.mdb
-rwxr-xr-x   1 1085440     Dec 31  1998  Inventory Control1.mdb
-rwxr-xr-x   1 1087488     Apr 19  1999  Inventory Control2.mdb
-rwxr-xr-x   1 1277952     Dec 31  1998  Service Call Management1.mdb
-rwxr-xr-x   1 71680       Dec 27  1998  db1.mdb
-rwxr-xr-x   1 61440       Nov 21  1999  db10.mdb
-rwxr-xr-x   1 196608      Jan  4  1980  db11.mdb
-rwxr-xr-x   1 137216      Jan  4  1980  db12.mdb
-rwxr-xr-x   1 153600      Jan  4  1980  db13.mdb
-rwxr-xr-x   1 71680       Mar 30  1999  db2.mdb
-rwxr-xr-x   1 61440       Dec 28  1998  db3.mdb
-rwxr-xr-x   1 59392       Dec 28  1998  db4.mdb
-rwxr-xr-x   1 145408      Dec 28  1998  db5.mdb
-rwxr-xr-x   1 67584       Apr 19  1999  db6.mdb
```

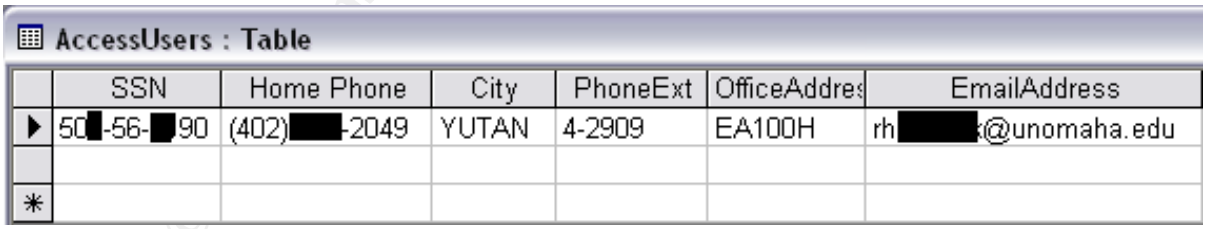
-rwxr-xr-x	1 73728	Apr 19 1999	db7.mdb
-rwxr-xr-x	1 73728	Jun 27 1999	db8.mdb
-rwxr-xr-x	1 77824	Aug 1 1999	db9.mdb
-rwxr-xr-x	1 258048	Feb 8 1999	user H.mdb

The databases were void of information, except for some 'scratch' information. Moreover, each database had the exact same interface except for a few changes. One might speculate that these were databases that had been created for a class.

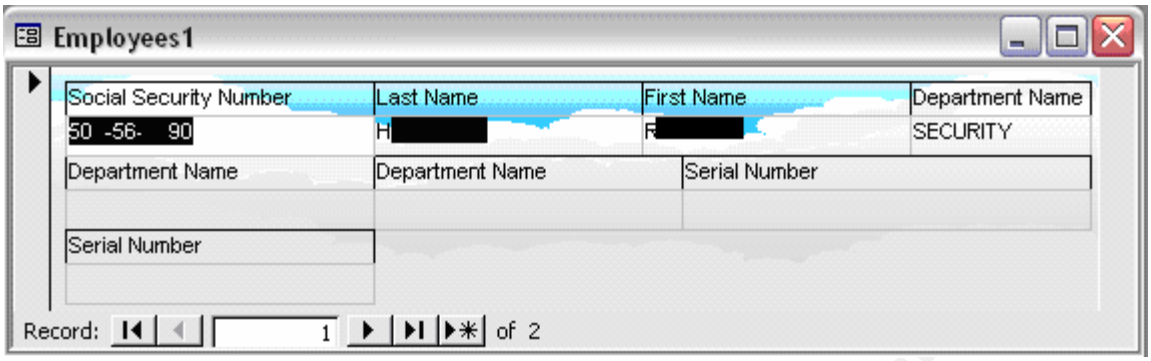
Below is a screenshot from db4.mdb. It appears that user H was the creator of this database.



A second database (db5.mdb) was even more revealing, containing user H's (apparent) social security number, home phone, office and email addresses.



Finally, db10.mdb contained user H's social security number and full name.



Temporary Internet Files

The *\windows\Temporary Internet Files* directory contains four subdirectories as expected. When a user visits a web site with a browser, the files downloaded are cached within the four subdirectories in turn. For example, if a user visits sites A, B, C &, D, then a copy of page A would be placed in the first subdirectory, B in the second, C in the third, and D in the fourth. A copy of the next web page would be placed in A, and so on..

An examination of this directory was accomplished in two steps. First I extracted the human-readable strings from the index.dat file to determine which users were visiting which sites.

strings –a windows/Temporary Internet Files/index.dat > /sans.cert/tif.strings.

A typical record within the index.dat file looks like the following:

URL
http://www3.infoseek.com/Topic?tid=502
Topic.htm
HTTP/1.0 200 OK
MIME-Version: 1.0
Content-length: 23754
Content-type: text/html
~U:t*****

Each record displays the URL, the protocol used in the download, the number of bytes of the download, type of content, and the user's account which accessed the site. The above example indicates that user T accessed www.infoseek.com, a popular search engine at that time.

There was nothing particularly inflammatory or interesting about the web sites visited. The only interesting evidence found was that, apparently, only User T and S browsed the Internet, given that there's were the only names appearing in the index.dat file.

The second procedure involved manually loading each of the files in each of the four subdirectories of *\windows\Temporary Internet Files* into a browser for viewing. Several interesting pieces of evidence were found. First, some user was interested in California State University at Fullerton. Several files associated with Cal State Fullerton were found, including files containing information on tuition rates, and also a login page (see figure). The tuition rates page for the University of Nebraska @ Omaha were visited for 1996-97 (an old page) and from 1997-98. One could speculate that user T had a son or daughter attending California State University, and one at the University of Nebraska @ Omaha. However, because this computer was used by employees in student accounting, this is not definitive (one might expect employees in this department to view these web pages as part of their job).

There was only a single record found in the index.dat file relating to Cal State Fullerton.

URL
http://bfa.fullerton.edu/images/Visa.gif
Visa.gif
HTTP/1.1 200 OK
Content-Type: image/gif
ETag: "80f5ad6b7ddfbc1:dcc"
Content-Length: 654

Unfortunately, no name is associated with this record.

A second interesting finding is that the temporary internet folders contained several files that indicate search keywords and results.

- Yahoo:
  - Search Keyword: *automated gifs*

- USWESTDEX:
  - Search Keyword: *Bellevue Nebraska*
- Unspecified search page:
  - Search Keyword: *graphics*
- USWESTDEX:
  - Search Keywords: *Joan G**** of Piedmont, OH*

User S was the only user associated with the uswestdex site (according to records in the index.dat file), therefore, it is reasonable to conclude that she was the one who searched for Bellevue, NE and Joan G***** of Piedmont OH.

> http://www.uswestdex.com/img/index_11.gif
> index_11.gif
> HTTP/1.0 200 OK
> Content-length: 2800
> Content-type: image/gif
> **~U:s*******

Cookies

The *c:\windows\cookies* directory holds cookies, small text files placed on the user's computer by a server. Cookies provide evidence of web sites users have visited.

```
[jpc@whammo cookies]$ ls -lut
total 1376
-rwxr-xr-x   1 90 Dec  7  1998   USER.S@net-trak_stats(1).txt
-rwxr-xr-x   1 203 Dec  7  1998  USER.S@home_microsoft(2).txt
-rwxr-xr-x   1 113 Nov 30  1998 USER.S@wp_uswestdex(1).txt
-rwxr-xr-x   1 190 Nov 30  1998 USER.S@www_uswestdex(2).txt
-rwxr-xr-x   1 94 Nov 12  1998   USER.S@abc.txt
-rwxr-xr-x   1 69 Oct 27  1998   USER.S@bfast.txt
-rwxr-xr-x   1 91 Oct 27  1998   USER.S@linkexchange.txt
-rwxr-xr-x   1 92 Oct 27  1998   USER.S@homearts.txt
-rwxr-xr-x   1 103 Oct 27  1998  USER.S@sidewalk_msn.txt
-rwxr-xr-x   1 73 Oct 26  1998   USER.S@doubleclick.txt
-rwxr-xr-x   1 99 Oct 26  1998   USER.S@4.txt
-rwxr-xr-x   1 181 Oct 26  1998  USER.S@switchboard(1).txt
-rwxr-xr-x   1 80 Oct 26  1998   USER.S@altavista.txt
-rwxr-xr-x   1 189 Sep 29  1998 USER.S@www_uswestdex.txt
-rwxr-xr-x   1 103 Sep 29  1998 USER.S@home_microsoft(1).txt
-rwxr-xr-x   1 111 Sep 22  1998 USER.S@www_drwhitakersvitamins.txt
```

```
-rwxr-xr-x   1 155 Sep 21  1998 USER.S@netfind_aol(1).txt
-rwxr-xr-x   1 98 Sep 21  1998   USER.S@preferences.txt
-rwxr-xr-x   1 184 Sep 21  1998 anyuser@www_uswestdex(1).txt
-rwxr-xr-x   1 99 Sep 21  1998   anyuser@web3_yp_uswest(1).txt
-rwxr-xr-x   1 77 Aug 26  1998   USER.S@yahoo.txt
-rwxr-xr-x   1 125 Aug 18  1998 USER.S@www_windowsmedia(1).txt
-rwxr-xr-x   1 343 Jul 31  1998   USER.S@yp_yahoo(1).txt
-rwxr-xr-x   1 100 Jul 13  1998   USER.S@web3_yp_uswest(1).txt
-rwxr-xr-x   1 147 Jul 13  1998   USER.S@excite(1).txt
-rwxr-xr-x   1 106 Jul 13  1998   USER.S@infoseek.txt
-rwxr-xr-x   1 100 Jul  2  1998   USER.S@microsoft(1).txt
-rwxr-xr-x   1 94 Jul  2  1998     USER.S@msn.txt
-rwxr-xr-x   1 76 Feb  6  1998    USER.T@yahoo.txt
-rwxr-xr-x   1 160 Jan 21  1998  USER.T@msn(1).txt
-rwxr-xr-x   1 102 Jan 21  1998  USER.T@home_microsoft.txt
-rwxr-xr-x   1 147 Jan 20  1998  USER.T@excite(1).txt
-rwxr-xr-x   1 98 Jan 19  1998   USER.T@preferences(1).txt
-rwxr-xr-x   1 73 Jan  9  1998    USER.T@doubleclick.txt
-rwxr-xr-x   1 91 Dec 18  1997   USER.T@linkexchange.txt
-rwxr-xr-x   1 106 Dec 18  1997 USER.T@infoseek.txt
-rwxr-xr-x   1 109 Dec 18  1997 USER.T@warnerbros_entertaindom.txt
-rwxr-xr-x   1 84 Dec 12  1997   USER.T@gateway2000.txt
-rwxr-xr-x   1 83 Dec 12  1997   USER.T@www_gateway.txt
-rwxr-xr-x   1 87 Dec  5  1997   USER.T@netscape.txt
-rwxr-xr-x   1 8192 Dec  5  1997          mm2048.dat
-rwxr-xr-x   1 8192 Dec  5  1997          mm256.dat
-rwxr-xr-x   1 32768 Jan  4  1980 index.dat
```

The timelines for user T and S's cookies are consistent with the timeline acquired from an examination of their password files. Interestingly, these were the only users who had cookies, which is consistent with findings from the index.dat from the Window's Temporary Internet Files directory.

Printer Spool Files

Eighty print spool files were still accessible within unallocated space. Example print spool files include:

```
Thu Nov 19 1998 15:44:18
161 ..c -rwxrwxrwx 0 0 505371   <uno.dispose.dd-_0022.SHD-dead-505371>
161 ..c -/-rwxrwxrwx 0 0 505371   /WINDOWS/SPOOL/PRINTERS/_0022.SHD (deleted)
Thu Nov 19 1998 15:44:20
161 m.. -/-rwxrwxrwx 0 0 505371   /WINDOWS/SPOOL/PRINTERS/_0022.SHD (deleted)
161 m.. -rwxrwxrwx 0 0 505371   <uno.dispose.dd-_0022.SHD-dead-505371>
Thu Nov 19 1998 15:45:22
162 ..c -/-rwxrwxrwx 0 0 505372   /WINDOWS/SPOOL/PRINTERS/_0023.SHD (deleted)
Thu Nov 19 1998 15:45:24
162 m.. -/-rwxrwxrwx 0 0 505372   /WINDOWS/SPOOL/PRINTERS/_0023.SHD (deleted)
Thu Nov 19 1998 16:10:30
164 m.. -/-rwxrwxrwx 0 0 505390   /WINDOWS/SPOOL/PRINTERS/_0041.SHD (deleted)
```

I retrieved these files using TASK's *icat* command line utility.

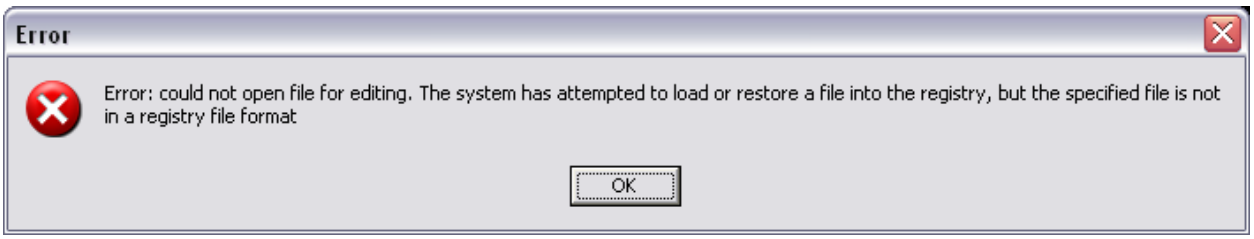
```
# icat –f fat16 /sans.cert/uno.dispose.dd 505390 > /sans.cert/505390.icat
```

The 505390 is the inode number of the file. Unfortunately nothing in the way of interesting evidence was revealed in these files.

Windows Registry File

The Window's registry file holds a wealth of information regarding the system and its users. As mentioned previous, Windows 95 does not partition the user space (as do the more recent versions of Windows). Therefore, there are only two files associated with the Windows registry as opposed to two per user. These files are *system.dat* and *user.dat*, found in the *\windows* directory. Possible evidentiary information found in the Windows 95 version of the registry includes: user information; MRU (most recently used files); installed programs; deleted programs; etc.

I copied these files, along with their associated backups found in the same directory (system.da0 and user.da0) to a FAT32 partition, for analysis under Windows XP using Resplendence Registrar, a powerful registry editor.



I first attempted to load *system.dat* into Resplendence Registrar, and received the error message displayed above. I next attempted to load *user.dat*, but received the same error message. I used Google to search the Internet to determine what the problem is, but unfortunate, could not find a solution. This is unfortunate as the registry is an important part of any forensic analysis on a Windows system.

Given I couldn't use the preferred method of accessing the contents of the registry files, I opened up the files using a hex editor. Unfortunately, it was very difficult, or impossible, to interpret the contents with any degree of accuracy.

UNALLOCATED SPACE

Unallocated space consists of disk clusters that: (a) have never been allocated for use, or (b) were previously allocated for a file, but were marked for reallocation due to a files deletion. Forensically speaking (b) is the more interesting situation because data residing in those clusters will be available for retrieval until overwritten.

The TASK command line utility *dls* was used to extract data from the unallocated clusters.

```
//check unallocated space

# dls -f fat uno.dispose.dd > uno.dispose.dls

# strings -a uno.dispose.dls > uno.dispose.dls.strings

# grep -A 5 -B 5 -i ssn uno.dispose.dls.strings | less

# grep -A 5 -B 5 -i pass uno.dispose.dls.strings | less

# grep -A 5 -B 5 -i sex uno.dispose.dls.strings | less

# grep -A 5 -B 5 -i porn uno.dispose.dls.strings | less
```

The –A 5 and –B 5 flags indicate to return 5 lines after the match and five lines before, to provide context. The –i flag indicates to ignore case.

The 'ssn' and 'pass' keywords were chosen to determine if any personal information was located within the files (there was). The 'sex' and 'porn' keywords were chosen to determine the 'browsing habits' of the users.

The only interesting information found in the unallocated space was user id/password combinations:

---

user=zyac074
password=^EXL<DEIqE4!pp
login=^L03^T05^M^Slogin:^S^W^U^M^SPassword:^S^W^P^M
logout=^L03^T05^Mexit^M^Slogin:^S

stopserver=^L03^T05^C^X^X^X^X^X^M^S\x24 ^S^R00|^S# ^S^R00|^S%
^S^R00|^Slogin:^S^R00
startserver=^L03^T10^C^X^X^X^X^X^H^H^H^H^H^H^Mterm -
x^M^SOK^M^J^S^R00|^Slogin:^S^R01

Recall that the PWL file was cracked for user T, and that her password for a Window's share was 'zyac074.'

---

telnet
xxxcsh00 [sanitized]
zyac074
XL<DElqE4!pp
login:
Password:
exit

It appears that user T used telnet to connect to a university computer. Note that the 'zyac074' and 'XL<DElqE4!pp' appear again. I used the domain internet groper (dig) utility to determine the identity of 'xxxcsh00':

```
[jpc@whammo jpc]$ dig xxxcsh00.xxxxxxx.edu
; <<>> DiG 9.2.1 <<>> unocshxx.unomaha.edu
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35784
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2
;; QUESTION SECTION:
;xxxcsh00.xxxxxxx.edu.          IN      A
;; ANSWER SECTION:
unocshxx.unomaha.edu.  86400   IN      CNAME   xxxcsh00.eab.xxxxxxx.edu.
xxxcsh00.eab.unomaha.edu. 86400 IN     A       137.xx.xx.xx
;; AUTHORITY SECTION:
eab.xxxxxxx.edu.       86400   IN      NS      dns-1.xxxxxxxx.edu.
eab.xxxxxxx.edu.       86400   IN      NS      dns-2.xxxxxxx.edu.
```

The 'eab.xxxxxxx.edu' is a computer in the university's administration building. This is consistent given the other evidence that indicates this person works somewhere within financial aid office.

```
xprot=WTERMCRC
user=post
password=^EVL,D_)UG55/p
login=^L03^T05^M^Slogin:^S^W^U^M^SPassword:^S^W^P^M
logout=^L03^T05^Mexit^M^Slogin:^S
startserver=^L03^T10^C^X^X^X^X^X^H^H^H^H^H^H^Mterm -
x^M^SOK^M^J^S^R00|^Slogin:^S
^R01
stopserver=^L03^T05^C^X^X^X^X^X^M^S\x24 ^S^R00|^S# ^S^R00|^S%
^S^R00|^Slogin:^S^
```

A Google search for "WTERMCRC" indicated that it was some type of file transfer protocol in the realm of zmodem, xmodem, ymodem, ftp, etc.

```
xprot=WTERMCRC
user=lookup
password=^EVL,D_)UG55/p
login=^L03^T05^M^Slogin:^S^W^U^M^SPassword:^S^W^P^M
logout=^L03^T05^Mexit^M^Slogin:^S
startserver=^L03^T10^C^X^X^X^X^X^H^H^H^H^H^H^Mterm -
x^M^SOK^M^J^S^R00|^Slogin:^S
```

We see two new users, 'lookup' and 'post', which could be automated processes or scripts of various kinds that are run unattended (or maybe not). Nevertheless, note that they both use the same password.

There were several files found in unallocated space. Here are the most interesting ones:

1. A letter about a Korean student drowning in the university pool.
2. An email about an "Inside Edition" TV story about a scam to abduct women in parking lots.
3. A letter regarding the 2000 election and all the hubbub surrounding it.

The last letter is telling about the system clock. This story had to have been written after November, 2000. Yet the last date in the MAC timelines was December, 1999.

Other less interesting things found include a list of faculty names, and a list of what appears to be student names. No other personal information was associated with the list of names.

SLACK SPACE ANALYSIS

I searched the slack space with TASK's *dls* command using the –s switch. Recall that slack space is the space at the end of a cluster that is not used by a file. For example, if a file is 1 character (in ASCII, 1 byte), then the cluster current cluster size (recall, 32768 bytes) allows for slack space amounting to 32768 – 1 = 32767 bytes. That's a lot of wasted space; however, from a forensic standpoint it can be a goldmine.

// check the slack space only

dls -f fat -s uno.dispose.dd > uno.dispose.slack

strings -a uno.dispose.slack > uno.dispose.slack.strings

grep -A 5 -B 5 -i ssn uno.dispose.slack.strings | less

grep -A 5 -B 5 -i pass uno.dispose.slack.strings | less

grep -A 5 -B 5 -i sex uno.dispose.slack.strings | less

grep -A 5 -B 5 -i porn uno.dispose.slack.strings | less

The results were somewhat similar as to that found in unallocated space. For example, the same user/password combinations were found.

---

xprot=WTERMCRC
user=lookup
password=^EVL,D_)UG55/p
login=^L03^T05^M^Slogin:^S^W^U^M^SPassword:^S^W^P^M
logout=^L03^T05^Mexit^M^Slogin:^S
startserver=^L03^T10^C^X^X^X^X^X^H^H^H^H^H^H^Mterm -
x^M^SOK^M^J^S^R00|^Slogin:^S^R01

---

bsdel=OFF
col132=OFF
wru=
xprot=WTERMCRC
user=zyac074
password=^EE4,CGlqEs!pp
login=^L03^T05^M^Slogin:^S^W^U^M^SPassword:^S^W^P^M
logout=^L03^T05^Mexit^M^Slogin:^S

stopserver=^L03^T05^C^X^X^X^X^X^M^S\x24 ^S^R00|^S# ^S^R00|^S%
^S^R00|^Slogin:^S^R00

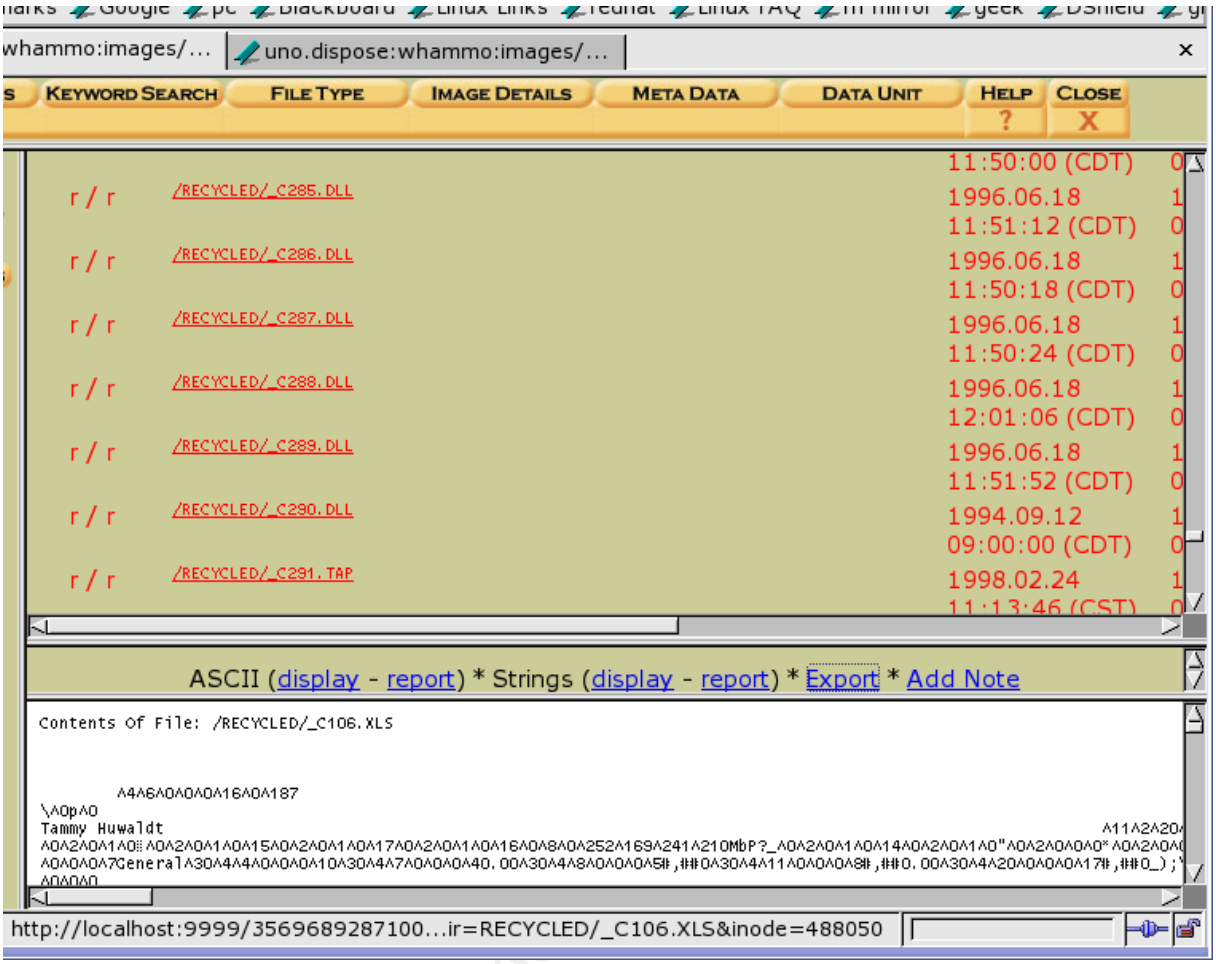
telnet
unocsh00
zyac074
E4,CGlqEs!pp
login:
Password:
exit

Recycle Bin

The recycle bin contained no files except for the desktop.ini, which contained the following:

[.ShellClassInfo] CLSID={645FF040-5081-101B-9F08-00AA002F954E}

This indicated that someone had 'emptied' the recycle bin at some point in time. I used the Autopsy Browser (Carrier, 2003) to determine what files, if any, had ever resided in the recycle bin.



Autopsy showed that there were several hundred deleted files, 295 of which had once resided in the Recycle Bin (see figure above). Note that when a file is 'deleted' using the Recycle Bin, a copy of the file is placed in the Recycle Bin and renamed by replacing the original name with the drive letter where the file resided, plus the order in which the file was placed in the Recycle Bin, and the original extension. Therefore, a typical file with possible evidentiary value (due to the file extension name) might be C068.doc, C191.xls, etc.

Several dozen of these files were Microsoft Office files, including Word and Excel documents. I used Autopsy's 'export' function to retrieve these documents and OpenOffice 1.02 to view them.

Because most of these recovered documents were of a work-related nature and contained identifying information (particularly student names) I will only summarize them. The Excel documents contained information on Pell grant applications and other financial applications. The Word documents were letters to students regarding their university grants or accounts. User T signed the Word documents to students.

Conclusions

This document describes the forensic analysis of a 'moth-balled' university personal computer system. The forensic procedures were comprised of an analysis of allocated, slack, and unallocated space. The results of the analysis suggested there had been five users with accounts (user ids) on the computer over a four year period of time, and that the computer had once been used by employees of the cashiering/student accounting department.

A significant number of work-related as well as personal files were recovered from the hard drive. Interestingly, the work-related documents had been diligently deleted; however, they were easily retrieved using the Autopsy browser. Alternatively, sensitive personal documents remained on the computer. It would be interesting to find out why personal documents were not deleted as well.

Personal identifying information was easily retrieving, including; first and last names of former users, social security numbers, addresses, email login IDs, and phone numbers.

There were several unfortunate realities. First, the MAC timeline indicated there were several hundred files with impossible dates (from 1969 until the early 1990s, well before Windows 95). This made it impossible to determine the exact dates of the installation or creation of some files, including the initial installation of the software. Care should be taken when relying on any of the MAC time due to this anomaly. Finally, I was unable to import the Windows 95 registry files *system.dat* and *user.dat* into a registry editor, which likely would have held information of significant evidentiary value.

References

ComputerHope (1999). (http://www.computerhope.com/whow.htm)
Carrier, B. (2003). The @stake Sleuth Kit (TASK). http://www.atstake.com/research/
Carrier, B. (2003). Autopsy Browser http://www.atstake.com/research/
Lee, R. (2003). Mac-Daddy. http://www.incident-response.org/mac_daddy.html
John the Ripper. (2002). http://www.openwall.com/john/

Part 3

Legal Issues of Incident Handling

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

The provider has the legal obligation to protect the privacy of the individual. Two Federal Statutes may be applicable in this situation: *The Electronic Communications Privacy Act Sources*: (United States Department of Justice, July, 2002, Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations, http://www.usdoj.gov/criminal/cybercrime/s&smanual2002.htm), and the *U.S. Patriot Act* (Source: http://www.ala.org/washoff/patriot.html).

Electronic Communications Privacy Act

The 18 U.S.C. 2703, ECPA creates statutory privacy rights for customers of ISPs, imposing restrictions and granting rights to system operators and government agents with regard to disclosing stored customer information and stored communications.

Two important pieces of information regarding the scenario that determine how the provider should handle it are a) it is a law enforcement (government) agent who is requesting the information, and b) the fact that the provider is a "public provider." Had the provider been a private provider, then it could have voluntarily disclosed the information requested. Private providers are not subject to ECPA. Or, had the calling party NOT been a government agent (law enforcement), then the provider could have voluntarily disclosed the information.

The ECPA provides for voluntary disclosure of non-content customer records to government agencies. When:

(1) The disclosure "may be necessarily incident to the rendition of the service or to the protection of the rights or property of the provider of that service," § 2702(c)(3);
(2) The provider "reasonably believes that an emergency involving immediate danger of death of serious physical injury to any person" justifies disclosure, § 2702(c)(4); or
(3) the disclosure is made with the consent of the intended recipient, or pursuant to a court order or legal process § 2702(c)(1)-(2).

"In general, these exceptions permit disclosure by a provider to the public when the needs of public safety and service providers outweigh privacy concerns of customers, or else when disclosure is unlikely to pose a serious threat to privacy interests."

U.S Patriot Act

The U.S. Patriot Act (USC 2703) may also be applicable to this situation. (Source: http://www.ala.org/washoff/patriot.html) According the 2703 (c):

(1) A governmental entity may require a provider of electronic communication service or remote computing service to disclose a record or other information pertaining to a subscriber to or customer of such service (not including the contents of communications) only when the governmental entity—
   a. obtains a warrant issued using the procedures described in the Federal Rules of Criminal Procedure by a court with jurisdiction over the offense under investigation or equivalent State warrant;
   b. obtains a court order for such disclosure under subsection (d) of this section;
   c. has the consent of the subscriber or customer to such disclosure; or
   d. submits a formal written request relevant to a law enforcement investigation concerning telemarketing fraud for the name, address, and place of business of a subscriber or customer of such provider, which subscriber or customer is engaged in telemarketing (as such term is defined in section 2325 of this title); or
   e. seeks information under paragraph (2).

(2) A provider of electronic communication service or remote computing service shall disclose to a governmental entity the—
   a. name;
   b. address;
   c. local and long distance telephone connection records, or records of session times and durations;
   d. length of service (including start date) and types of service utilized;
   e. telephone or instrument number or other subscriber number or identity, including any temporarily assigned network address; and
   f. means and source of payment for such service (including any credit card or bank account number), of a subscriber to or customer of such service when the governmental entity uses an administrative subpoena authorized by a Federal or State statute or a Federal or State grand jury or trial subpoena or any means available under paragraph (1).

Nebraska Computer Crimes Act

The Nebraska Computer Crimes Act does not provide guidance on the situation in this case, at least with respect to the obligations of the provider. However, it does deal with the suspected hacker in this case. (Source: http://statutes.unicam.state.ne.us)

Conclusion

The provider would not be able to provide the information requested by the law enforcement agent, given they are a public provider and covered by the ECPA. The only item of information that the provider could provide is the fact that that person is a customer, and the fact that they have logs for the customer at the time of the incident.

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

Phone, FAX, or email them and make a request that they preserve the logs:

From: (United States Department of Justice, July, 2002, Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations, http://www.usdoj.gov/criminal/cybercrime/s&smanual2002.htm

Preservation of Evidence under 18 U.S.C. § 2703(f)

"Agents may direct providers to preserve existing records pending the issuance of compulsory legal process. … ECPA permits the government to direct providers to "freeze" stored records and communications pursuant to 18 U.S.C. § 2703(f).

Specifically, § 2703(f)(1) states:

A provider of wire or electronic communication service or a remote computing service, upon the request of a governmental entity, shall take all necessary steps to preserve records and other evidence in its possession pending the issuance of a court order or other process."

This request may be made in the form of phone call, FAX, or email. According to DOJ (2002) "a fax or an e-mail is better practice because it both provides a paper record and guards against miscommunication. Upon receipt of the government's request, the provider must retain the records for 90 days, renewable for another 90-day period upon a government request."

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

Assuming that the ISP is considered a 'public provider,' then the ECPA has five mechanisms that a "government entity" may take to compel providers to disclose the requested information. Listed below, in ascending order of required threshold are the mechanisms:

a) Subpoena;
b) Subpoena with prior notice to the customer;
c) § 2703(d) Court order;
d) § 2703(d) Court order with prior notice to the customer, and;
e) Search warrant

(Source: United States Department of Justice, July, 2002, Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations, http://www.usdoj.gov/criminal/cybercrime/s&smanual2002.htm)

Question: What other investigative activity are you permitted to conduct at this time?

The provider is allowed to review any other logs, such as intrusion detection, if applicable. Also, the provider may contact upstream providers if it can be determined that the customer logged in from another ISP, as long as the ISP upstream is not an agent of the government. The Federal Wiretap Act and the Pen/Trap statute allow service providers to use pen/trap devices on their own networks without a court order.

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

There is a provider exception in the ECPA which allows the provider to voluntarily disclose information in the "may be necessarily incident to the rendition of the service or to the protection of the rights or property of the provider of that service" § 2702(b)(5). It would not be difficult to argue that it is in the provider's interest to call in law enforcement and provide them with the information to protect their customers.

From: (United States Department of Justice, July, 2002, Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations, http://www.usdoj.gov/criminal/cybercrime/s&smanual2002.htm