Global Information Assurance Certification Paper

Loki & the Honeypot: Forensic Analyses

GIAC Certified Forensic Analyst (GCFA)
Practical Assignment Version 1.2
Matthew Geiger

Summary

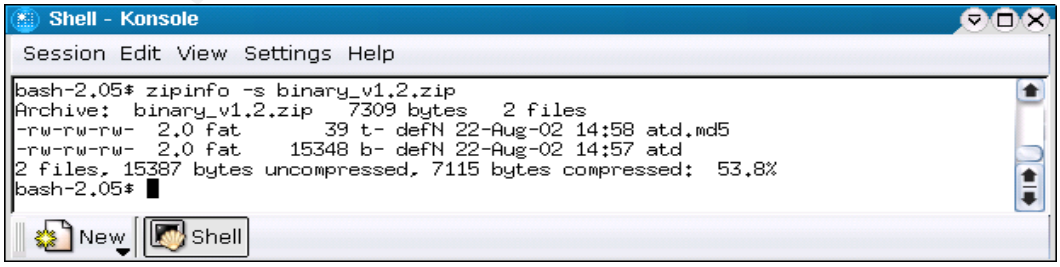
A three-part paper covering forensic examination techniques, analytical processes and related legal issues. The first pa rt examines an unknown binary to determine its nature and function, whether it can be identified and what forensically acceptable details of its history can be determined. Legal implications are also discussed.
Part two outlines the forensic analysis of a system, a potentially compromised honeypot. The examination pursues evidence of a compromise with an emphasis on the ability to demonstrate conclusions and present corroborating data in a legally sound fashion.
Part three considers local computer crime and privacy laws, in this case those of Singapore, in the context of a given scenario.

Part 1: Unknown Binary

Hunting the Norse God of Deception

Investigative Platform:
Fully updated and patched install of RedHat 7.2 running on a 1.7Ghz P4 system. The computer's network connection was unplugged prior to analysis. Tripwire was run to build a protected database of hashes of system binaries and configuration files to determine any were altered in the process of analyzing the binary. Precautions like these make sense when dealing with any suspected piece of executable malware.

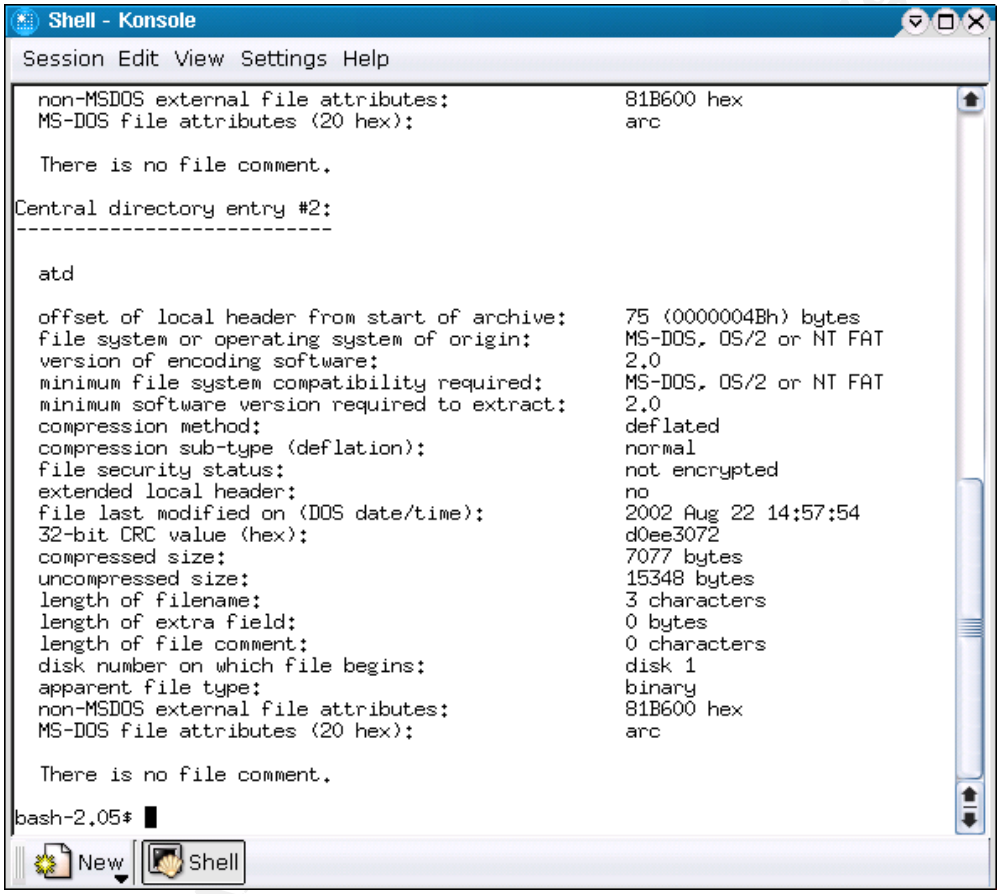
The target file for analysis was delivered in a compressed file named binary_v1.2.zip. The archive format is that same as used by zip utilities common to the Unix and Windows worlds. After verifying the integrity of the archive with the *unzip -t* command, I obtained a Unix-style listing of its contents with *zipinfo -s binary_v1.2.zip*.



A few points are worth noting:

- The archive contains two files: atd it considers a binary and atd.md5 i s considered text (and seems likely to contain the binary's MD5 hash value for verification).
- The executable bit isn't set for either file.
- The modification date for both is the same Aug. 22, 2002, with the MD5 hash file created a minute later.
- The archive was created on a DOS FAT filesystem, which has relevance for the accuracy and relevance of the two preceding points.

The *zipinfo -v* command generates a fuller set of data associated with the file in the archive.



As root, I extracted the two files fro m the archive using *unzip -X* to attempt to preserve the original UID/GID information for the binary. I also used the same command as an ordinary user. In both cases, the UID/GID on the files extracted were the same as those of the user calling *unzip* with n o errors thrown. With the -X option, *unzip* alerts if it can't create the UID/GID associated with the file because the calling user doesn't have sufficient privileges.

Hence, the fact that when the files are extracted with -X as root they have UID/GID of 0 and when extracted as user they have the user's UID/GID, without alerts in either case, indicates the files in the archive lost their associated *nix filesystem UID/GID info, probably the result of archiving on a DOS -based system.

The *gzip* utility won't work as another route to recovering additional metadata. According to its man page: "Files created by zip can be uncompressed by gzip only if they have a single member compressed with the 'deflation' method." An examination running WinZip under Windows XP Pro doesn't reveal any additional information that may have been stored in a proprietary format, either.

Details of the file:

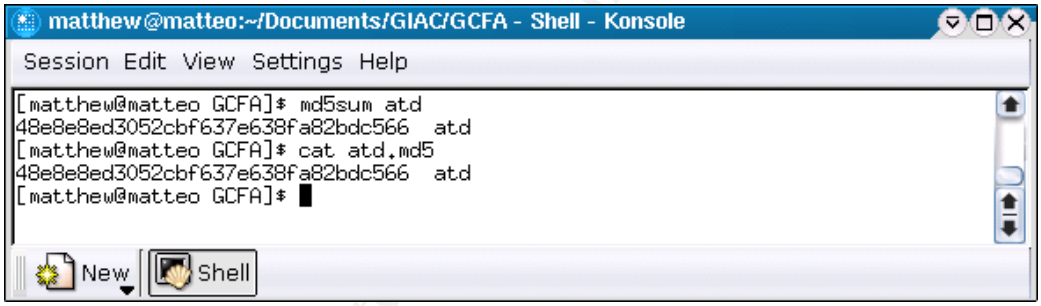
Filename: program has been named *atd*, which is the name of the *nix job - scheduling daemon that allows users to set up tasks and scripts to run at a predetermined time. (This differs from cron in that it's not used for recurring jobs.)

MACTime: not in UNIX MAC format because it was zipped in a Windows environment. Modification time appears above: 2002 Aug 22 14:57:54. Access time and creation time are set to the time of extraction when the file is extracted and aren't available from querying this archive.

File Owner: Set to the user/group who ran the extraction. Unix filesystem metadata wasn't preserved when archived from Windows filesystem.

File Size: 15,348 bytes

MD5 hash: 48e8e8ed3052cbf637e638fa82bdc566 (This matches that in the atd.md5 file also in the zip archive).



Key words:

Running the strings utility against the file presents a number of results inconsistent with the program being the *atd* job daemon and points to its identity instead as being *lokid*, the Loki covert communications daemon. Here's a selection of the relevant ASCII strings, with comments in italics:

`getprotobynumber` *-- part of a call to translate servic ename to port*
`gethostbyname` *-- similar, but for name resolution*
`inet_addr` *-- another string that wouldn't appear in the genuine atd, which doesn't internally handle network communications*

Following are the text components of a number of informational mess ages that point to the program's functionality:

```
lokid: Client database full
DEBUG: stat_client nono
lokid version: %s
remote interface: %s
```

```
active transport: %s
active cryptography: %s  -- Loki supports several types of encryption
server uptime: %.02f minute s
client ID: %d
packets written: %ld
bytes written: %ld
requests: %d
N@[fatal] cannot catch SIGALRM lokid: inactive client <%d>
expired from list [%d]
¬@[fatal] shared mem segment request error
lokid -p (i|u) [ -v (0|1) ]  -- here's a nice usage prompt
[fatal] forking error lokid: server is currently at capacity.
Try again later
lokid: Cannot add key
lokid: popen
/quit all lokid: client <%d> requested an all kill
sending L_QUIT: <%d> %s lokid: clean exit (killed at client
request)
/quit lokid: cannot locate  client entry in database
lokid: client <%d> freed from list [%d]
[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 chan ged to %s  -- Loki experimentally
allowed changing from ICMP to UDP (typically to port 53 to simulate DNS
traffic).
```

An Introduction to Loki:

Loki, named after a Norse God renowned as a shape -shifter and instigator of conflict (see the primer on Norse Myth ology by Sweden's Luleä University referenced below), is a program first described in issue 49 of Phrack magazine in the article "Project Loki: ICMP Tunneling". The program is designed to provide a covert channel of communications using ICMP "ping" packets . It subsequently incorporated a UDP channel that could be disguised as DNS traffic.

From the introduction of the Phrack article:

> "...many firewalls and networks consider ping traffic to be benign and will allow it to pass through, unmolested. This proj ect explores why that practice can be insecure. Ignoring the obvious threat of the done -to-death denial of service attack, use of ping traffic can open up covert channels through the networks in which it is allowed."

.. and from the Luleä University prim er on Norse Mythology:

> "Loki transgresses boundaries not only as shape -shifter but also as transgressor of gender boundaries, being able to change his sex at will.''

It appears this isn't even the first use of the name for malware. There's a precedent in DOS, according to McAfee's AVERT virus database:

> "The Loki, Loki-1237 or Merde-5, virus was received in October, 1992. Its origin or point of isolation are unknown. Loki is a memory resident infector of .COM and .EXE programs, including COMMAND.COM."

ICMP is a connectionless network messaging protocol, designed to carry information such the availability and status of hosts, routes and services. The Loki program uses the often-overlooked capability of ICMP "ping" packets to carry a data payload to establish a communications channel with the protocol. These are the ICMP_ECHO and ICMP_ECHOREPLY packets that can carry about 50 bytes of data each.

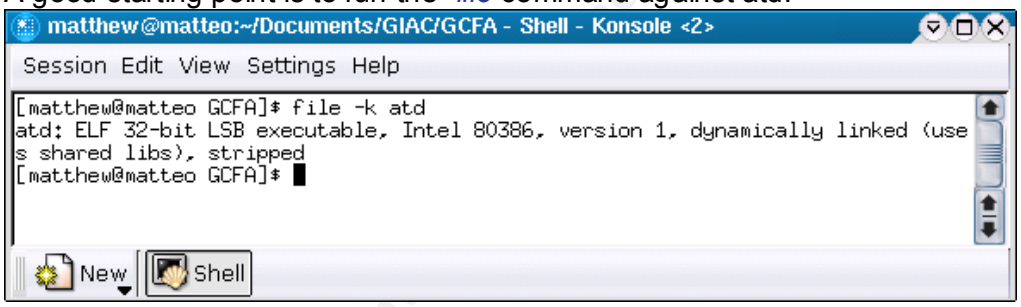
In the words of its authors, who use the handles daemon9 and alhambra:

> "If ICMP_ECHO traffic is allowed, then thi s channel exists. If this channel exists, then it is unbeatable for a backdoor (once the system is compromised)."

Disassembly and Analysis:

To examine the suspect file and build evidence that it is indeed the Loki daemon as suggested by our strings probe , we can use a number of binary file utilities and forensic tools.

A good starting point is to run the *file* command against atd:

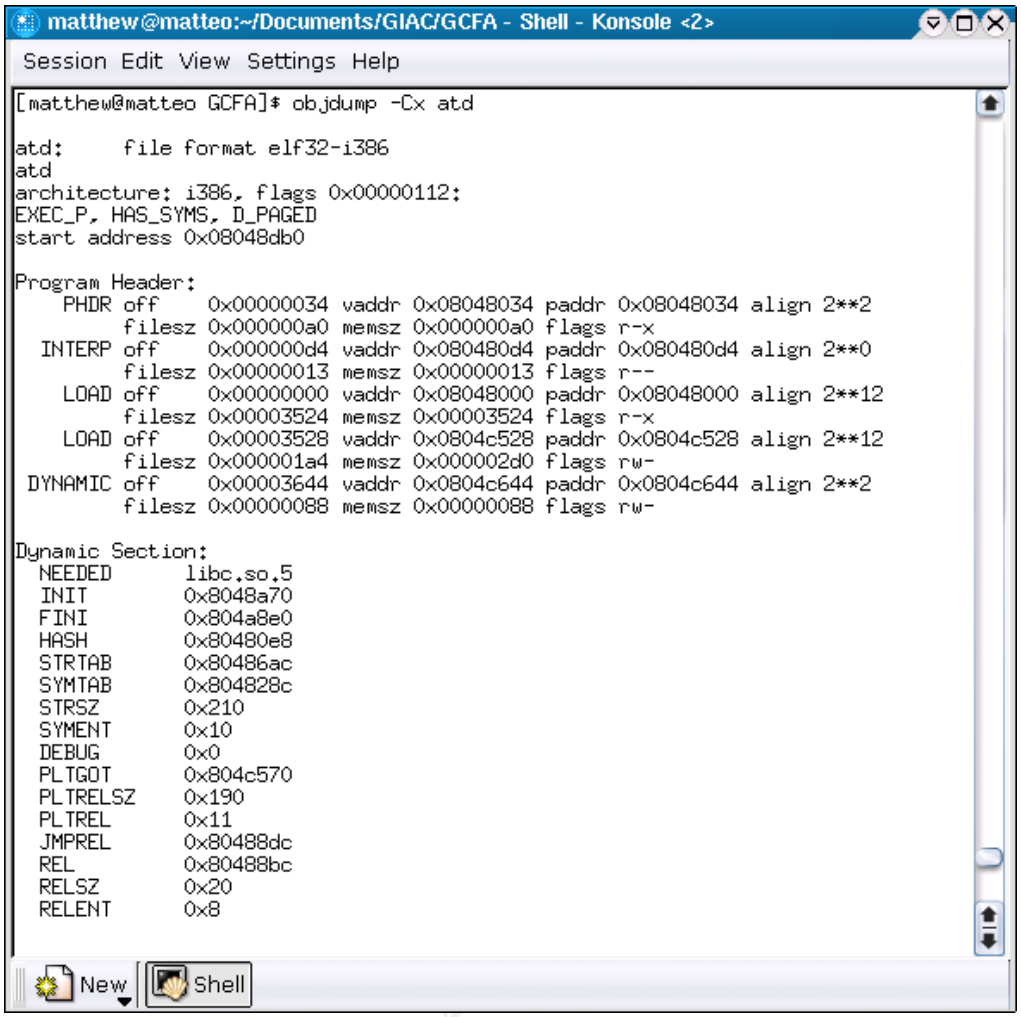


This tells us the file is an executable in the common Unix Executable and Linking Format, which relies on shared libraries on the host system that are external to the file. It's also stripped of symbols from object files. This points us in a couple of other productive directions.

Since it's dynamically linked, *ldd* should show us the shared libraries the program relies on. Setting the executable bit (*chmod +x*) on the target is first required to run *ldd* against the file. On the investigative platform, however, *ldd* complains it can't find the library or libraries linked.

Using *nm* to find object symbols is not going to get us any where because the binary is stripped.

So, we turn to *objdump* and initially run it with the -Cx arguments against atd.



This shows us a linked library needed is libc.so.5, a component of the standard C library, but an earlier version than that installed on the investigative platform, which uses libc.so.6. The program may also rely on the ld -Linux linker library if it was compiled under Linux. Running *objdump* with the -Csx options gives a far more verbose structural view of the program that indicates it u ses the ld-linux.so.1 library. The investigative system uses ld -linux.so.2. Text in the .comment section of the file, disclosed with *objdump -Csx*, indicates it was compiled with the Gnu GCC version 2.7.2.1. That compares with the 2.96 -major version of GCC on the examination system.

The older version of GCC shipped with the Red Hat 4.2 Linux release in mid -1997, which also had the 2.0.30 - 2.0.35 kernels and libc.so.5 -series C libraries. At this point, we might work on the assumption that this program was c ompiled on a Linux platform of circa 1997 vintage (which has no bearing on when the program was actually compiled, just the type of platform used.)

In all, this means run -time analysis will require a different investigative platform or modification of the existing one. Otherwise, the program will fail to find the shared library handler and the shared library it expects, both likely to be terminal errors for a

dynamically linked program like this. The same issues mean we can't step through the program fully in a debugger like *gdb,* at present.

In fact, trying to run the program under *gdb* results in the following:

```
Program exited with code 01.
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
warning: shared library handler failed to enable breakpoint
You can't do that without a process to debug.
```

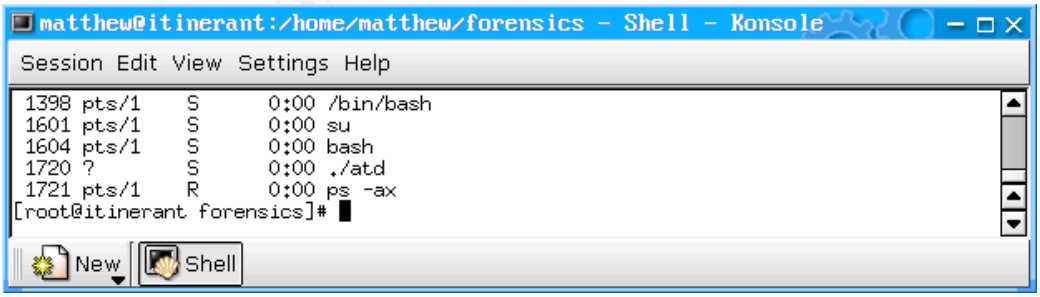
The best way to pursue this further might be a stock install of RedHat 4.2 or a system of similar vintage be cause, apart from the lib C and dynamic loader conflicts, it's possible that at the kernel and module levels further differences would interfere with successful execution -- especially on a program written and compiled for a kernel two MAJOR versions behin d modern kernels.

Building a Retrograde Platform:

Nonetheless, I decided to try to adapt a more modern platform to attempt a run -time analysis. For this, I chose a second system: an off -network laptop, also running a RedHat 7.2 install. After I hunted d own the antique RPMs from the RedHat 4.2 distribution obtained from ftp://mirror.pacific.net.au, I installed libc -5.3.12-18.i386.rpm, resolving a conflict with the /etc/nsswitch.conf configuration file by using the "--excludepath /etc/" option for *rpm*. I also installed the ld -so-1.7.14-4.i386.rpm.

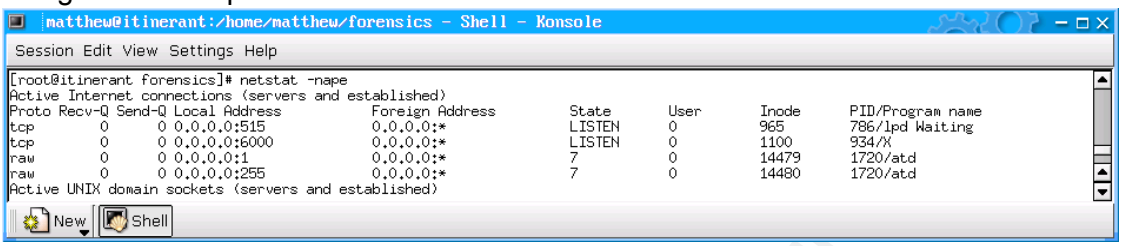
Conflicts with two binaries /usr/bin/ldd and /sbin/ldconfig I handled by copying the two current binaries to backup locations and installing the ld dynamic loader package with *rpm*'s --force option. After these i nstallations, I refreshed the Tripwire database, also applied to this machine, to trap any critical file modifications once the program was run.

This is what I considered the minimum for a decent shot at getting the program to execute ... and it did.

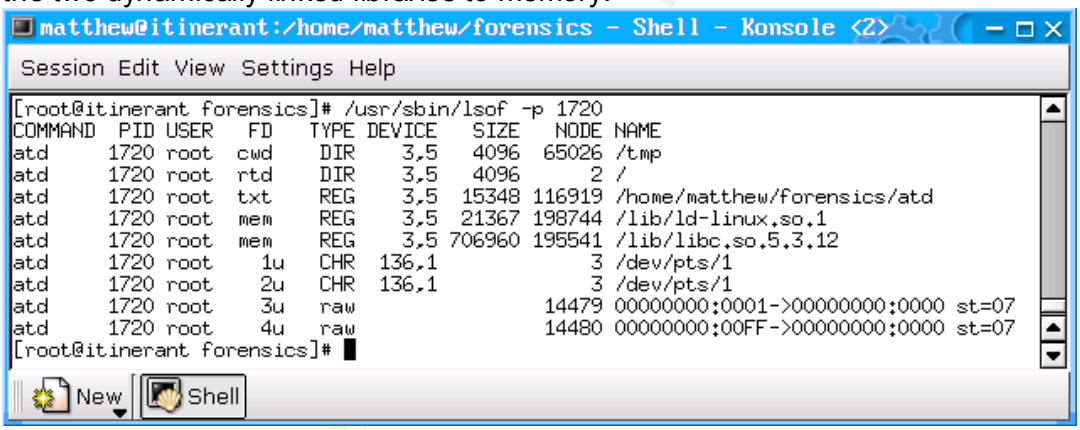


As shown in this partial *ps* listing, the program has gone daemon and is running with the PID 1720. Using the -aux switches reveals a little more information, including the fact that the daemon has a small system footprint, using just 0.4% of memory on this 64MB RAM laptop. Tripwire didn't register any modification of system binaries or key config files.

Next, running *netstat* shows two raw IP sockets opened, one associated with ICMP on port 1 and one on port 255, both are tied to the PID of *atd* and are running as root -- the program complained when execution was attempted without super -user privileges. The state information isn't really meaningful for a raw socket. Because ICMP packets are defined by type instead of being bound to a service port, the second port assignment of 255 is interesting but not necessarily meaningful beyond being a second open socket.



Using *lsof* to look at the files and handles opened by the suspect binary's process shows it has set its current working directory as /tmp and designat es the standard / as its root directory. The binary itself is shown running -- the “txt” file descriptor is a little misleading in that it refers to program text (i.e. code and data), according to the man page for *lsof*, rather than a text file. It also sho ws that the program has mapped the two dynamically linked libraries to memory.



The next four items are the file descriptor number of special files created by the program, with the “u” indicating read -write access. The four appear to be paired off, with one character special file for each raw socket, which would stand to reason to take and manipulate the I/O for each socket. This gives us confirmation of the two open raw sockets reported by *netstat* and, indeed, shows us the same information: one socket is listening on port 1 and one is on FF, the hexadecimal equivalent of 255. State is listed as 7, also. The character special files' associated device names -- /dev/pts/1 -- is that of one of the Unix pseudo -TTY devices which handle STD I/O from, for exampl e, shell consoles.

Positive Identification:

So far, the findings are all consistent with the description for Loki -- a program that opens a covert communication channel to allow remote access to a system. There are some other obvious paths to pursue for further validation of its identity. One is to compare the suspect Loki daemon with one compiled from Loki source code.

Another is to compile the client and attempt to communicate with the running daemon.

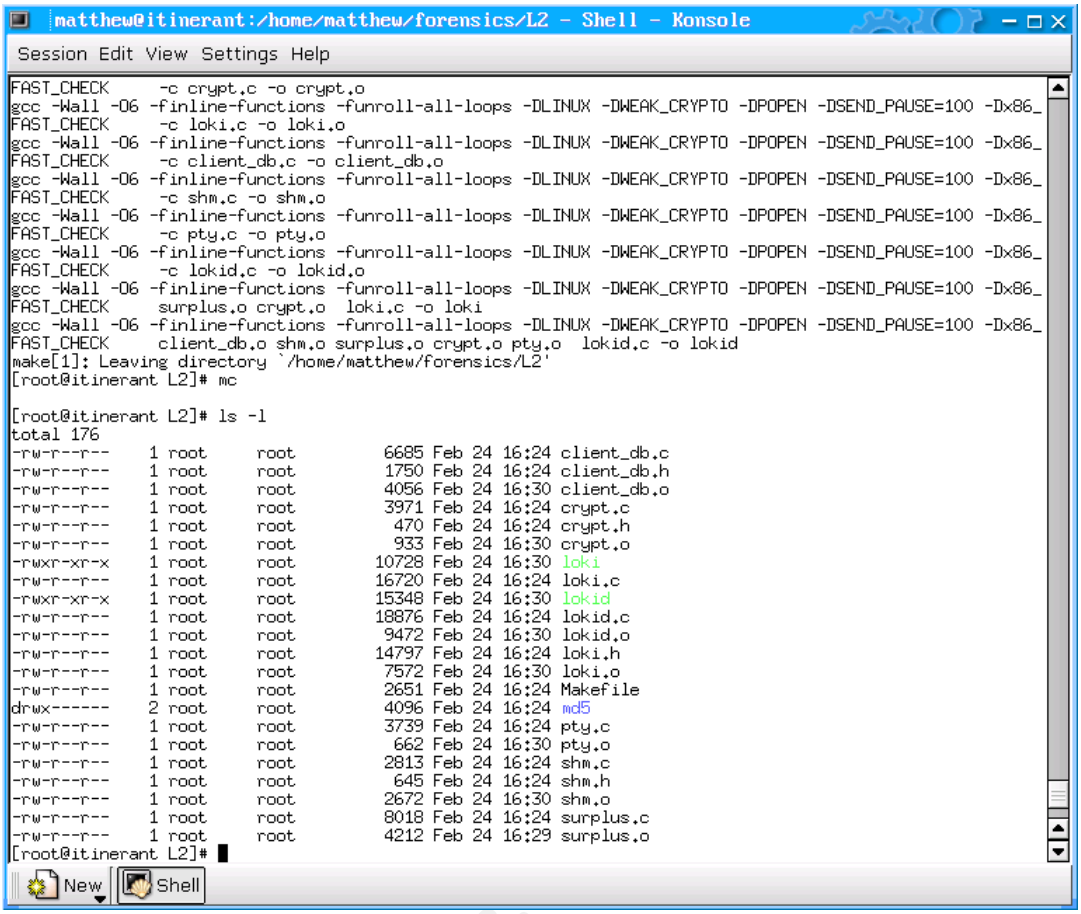
Both may fall short of 100% definitive answers. A daemon-to-daemon comparison in search of matching MD5 hashes could be very time-consuming -- even fruitless -- because minor differences in compiler platforms, compiling environments and compile-time options would lead to program variations that may not significantly affect functionality but would ensure non-matching checksums. Still, comparing the two would allow assessments of how similar they were in content and system function, using tools such as the ones already applied above, and others such as *diff*.

Attempting a connection from a known Loki client also has some drawbacks. If it doesn't work, it doesn't necessarily mean it's not a functional *lokid*, it may mean it's just not functional on the evaluating system. If it does work, it doesn't necessarily mean the binary contains solely a Loki daemon, just that it does offer the functionality of Lokid. Still, this approach is probably the fastest to return some probative value, i.e. if this does open a covert communication backdoor, then -- regardless of what else it does -- it does that.

Taking the belt and suspenders approach, I decided to do both. First, on the same testing platform with the selected RH4.2 libraries, I installed the same distribution's stock binutils-2.7.0.2-4.i386.rpm package (resolving minor conflicts), kernel-headers-2.0.30-2.i386.rpm, libc-devel-5.3.12-18.i386.rpm and the gcc-2.7.2.1-2.i386.rpm package, which should allow us to mimic the core of the compiling environment under which the suspect program was made.

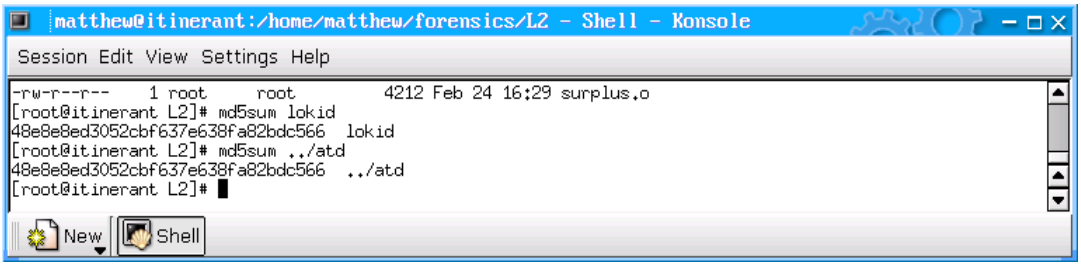
While researching Loki, I had located its source code in Phrack issue 51, in the article "L O K I 2 (the implementation)". Now, using the *extract.c* program compiled from that issue, I extract the source tree from which to build Loki, which is created in a directory called L2. In that directory, I compile the program with the *make Linux* command, leaving the Makefile options at their default, including XOR 'encryption' and no debugging info. The make process, as configured, also strips symbols.

This creates two executables, *loki* and *lokid*. I get a little buzz when I realize the lokid binary is also 15,348 bytes. This could be the jackpot.



I run md5sum against the newly created *lokid* and -- BINGO -- it matches the MD5 hash on the suspect binary. This is a grand -slam. It means the suspect binary is, byte for byte, identical to the *lokid* just built using the RedHat 4.2 compiling platform I cobbled together. MD5 hashes are created by what's known as a one -way algorithm, which calculates a 128-bit hash value from any given input . That gives us 2^{128} different possible hash values, and the one -way nature of the algorithm makes deriving the input from the hash what cryptographers refer to as "difficult" and what the rest of the world refers to as "basically impossible", with current levels of computational power.

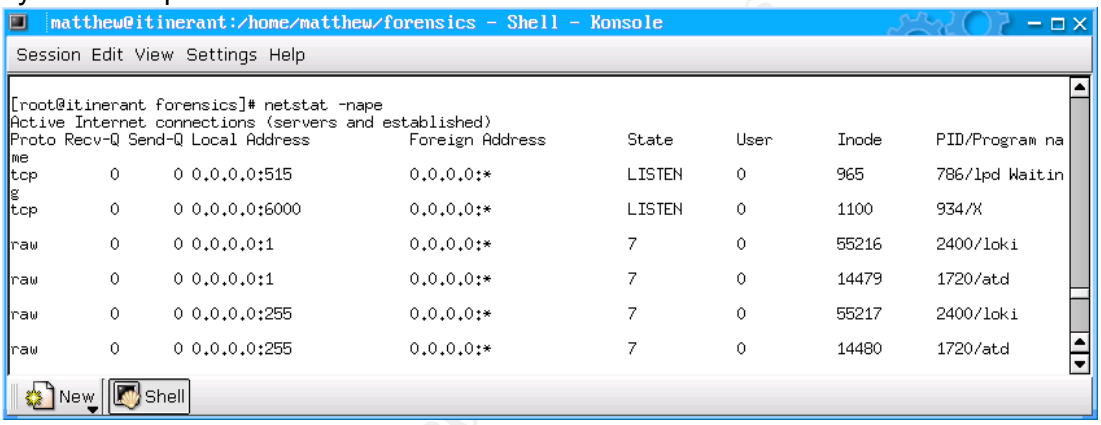
You would have to try, on average, 2^{127} different inputs to find one with a specific hash value. That means its "basically impossible" to find or develop a program with an MD5 hash that matches that of another. It also mak es the chances of two different input files accidentally possessing matching MD5 hashes (a so -called collision) infinitesimally small.



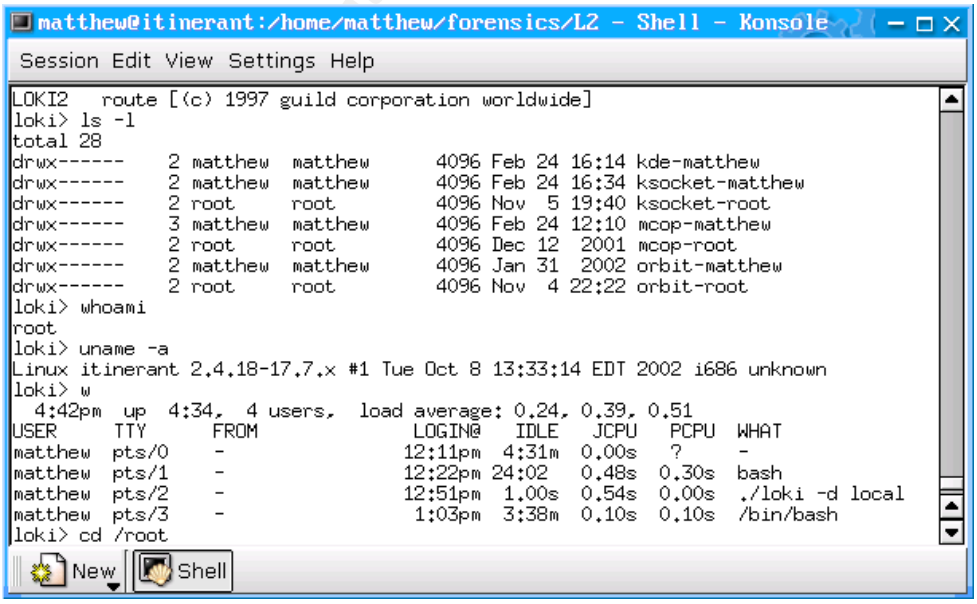
Although there is some migration toward SHA -1 as an even more robust hashing standard, MD5 is still relied upon and widely accepted as definitive. We have a positive ID.

A Test-Drive:

I crank up the *loki* client and test a connection with the running lokid on the test system's loopback interface. It works as advertised:



Below is a screenshot of a Loki session. The f irst directory listed is the /tmp that Loki sets as its working directory. Then below there's a smattering of bash commands to test the Loki shell.



Analysis & Conclusions:

For further analysis, one could set up another system with the necessary configuration to run the *loki* client (or daemon) to sniff the traffic between the two to verify its format. On a private network that isn't likely to see much ICMP traffic, using TCPDump to grab all ICMP packets wouldn't be impractical. But as far as identifying the program and its function is concerned, there's no real need to take this further.

As a back-door, Loki leaves a relatively small footprint -- ironically in part because it doesn't seek active concealment on the host. It incorporates no process hiding or subversion functions nor does it execute scripts, trojan binaries, etc at start-up. It's not a method of compromising a host -- it's a method of controlling a compromised host. Because it needs elevated privileges to execute, finding it running on a system suggests that has happened already, unless it was installed by someone with the authority to do so.

Depending on how a system administrator is used to running *netstat*, *lokid* may not be apparent in the listing, particularly if he's looking only for TCP and UDP listeners (i.e. *netstat -tupan*). On the plus side, "an active installation of *lokid* will often result in many zombie copies of the process left around, due to bugs in the program. This can be used as a clue," according to a 1997 report on Loki by Internet Security Systems, Inc. I made no attempt to verify this on the test-bed system.

Tracing ownership solely from the compiled binary given for this exercise is impractical, especially because of the file metadata lost in the zip archiving process. The original system needs to be subjected to a broader forensic examination to determine more about the file's provenance. String searches on the daemon didn't reveal hard-coded information that would tie it to anyone other that its creators. Indeed, the fact that the *lokid* binary I compiled from source hashes out to the same MD5 value means there can't be any individualized information in the discovered program.

We can determine some of the system requirements needed for the client and daemon to successfully execute. Note, however, that although vintage libraries are required to run it, the program ran quite happily on the latest RedHat kernel 2.4.18 -- so the system itself needn't be that old. An examination of the original system or a drive image would be needed to determine if this file was executed and by whom. To do that could take a number of avenues, including process listing, *netstat* and *lsof*, an analysis of data recovered from *lokid*'s /tmp working directory (which may need to be undeleted or pulled from unallocated inodes), a MACtime timeline, a memory dump analysis, attaching strace to the daemon, and even monitoring /dev/pts/x.

I wouldn't leap to the assumption that because the executable bit wasn't set on the atd binary in the archive, that it wasn't executed. That change could be accounted for by the manner in which the archive was created on the DOS system, and the archiving program used. Again a review of the original file, in situ, would be needed.

To determine ownership, you'd need to take that data as a starting point and pursue a forensic analysis, looking for evidence of a remote compromise or a local one.

Checking for a rootkit or deleted artifacts from compiling *lokid*, together with bash history files, would be a good start. *Lokid* doesn't care who connects with it and doesn't perform authentication, by design according to its authors. In a remote compromise, the cracker may build access control into a wrapper that provides more of a clue to identity, i.e. who does it allow to connect. Pulling the source address off ICMP packets to the host would also offers clues to the identity of *lokid*'s controller.

Given that many border routers and firewalls these days filter certain types of ICMP traffic and may munge ICMP data payloads, it makes sense to also consider a machine inside the private network as running the *loki* client. Both a scan of internal network traffic looking for Loki ICMP packets and, possibly, a search of other systems for the client program are possible responses.

Legal Implications:

Under Singapore's Computer Misuse Act of 1994, which is the main specific legislation covering computer crime on this island nation, anyone accessing a computer or service without authorization "shall be guilty of an offence and shall be liable on conviction to a fine not exceeding $10,000 or to imprisonment for a term not exceeding 3 years or to both and, in the case of a second or subsequent conviction, to a fine not exceeding $20,000 or to imprisonment for a term not exceeding 5 years or to both. [21/98]"

The penalties are much larger if the computer belongs to a bank, a sensitive government agency or is related to the operation of any protected infrastructure. (See the Legal Issues in Part 3 for a detailed discussion of this and other parts of the act). Likewise, damage to or interference with the operation of a computer are separate offenses under the act.

In addition, if we can demonstrate through packet sniffers, logs or forensic analysis, that any file was copied from the computer, it would be another crime that carries "a fine not exceeding $5,000 or to imprisonment for a term not exceeding 2 years or to both and, in the case of a second or subsequent conviction, to a fine not exceeding $10,000 or to imprisonment for a term not exceeding 3 years or to both. [21/98]," according the to act.

In terms of both a potential legal defense and in terms of how typical corporate policy might be applied, the nature of the Loki program makes it much harder for an insider who might have authority to access the systems to justify installing it.

First and foremost, there's no built-in authentication or access control for connecting to the daemon, which provides root level control of the system on which it runs. That's hugely irresponsible and should violate the most basic of corporate computer-use policies. It's certainly a hard software choice to justify to peers, managers or a jury.

The program intentionally shuns the reliability and stability of TCP connections and uses the "stateless" ICMP protocol. Stateless protocols have not way of determining whether transmissions arrive intact and – in the case of ICMP -- whether the packets arrive in correct order for reassembly. ICMP's only advantage is that it's not

considered a protocol that carries data a nd, thus, may be allowed through firewalls or otherwise avoid detection. (Less common today). The same is true of the Loki binary's alternative use of UDP packets that can simulate port 53 DNS traffic. This creates a definite presumption of skullduggery.

Interview Approach:

In questioning a person suspected of using the Loki program, the approach will obviously depend on the circumstances and who it is. Assuming the person is an insider, I'd want to establish control of the system/account and then angle i n non-threateningly on his activity, while being understanding and offering the interviewee a means to justify his actions.

Potential lines of questioning to pursue:

1. I'm looking into some weird stuff going on the network and trying to make sure that there's no laxness in how our policy is being applied. It's not that I think you have given out sensitive data, but I need to confirm with you that you've kept account information private and haven't let anyone log into to the system under your identity?

2. That's good. Now, I need your help in understanding what's going on here. I know I like to fool around with some underground tools myself, just to better inform myself of what's going on in that world. But look at it from my perspective. I'm under pressure fro m management to resolve this, so I need some details and explanations. That way we can settle this issue before the situation escalates. Help me out here.

3. Here's the problem. I don't think the higher -ups have a good understanding of the situation, and the y're inclined to hit the panic button whenever they run across anything that's unusual. If I can't end their uncertainty and doubts now, I think it's very likely they're going to turn outside the company and bring in police and lawyers. That's just going t o get ugly fast. Let's put an end to it before that happens and help me describe what the true situation is and why it shouldn't be such a big deal.

4. It looks like some of the activity we're seeing is coming from your system, and I noticed some underground tools on there. You could really help me out by not running them on our network any more. Is that OK? The one thing that really triggers our IDS is the Loki traffic. I know it can cut through a lot of red -tape for remote administration, but that does make some of the management nervous. So, that's definitely not going to continue, right?

5. Look my job is to know everything that happens on this network. This has developed into a situation where the only way to stop it from spinning out of control is if I can get details of what's going on and why. There's just too much here to go unexplained, and I'm concerned that someone is going to lose their job if we can't get answers quickly. You can see how there'd be a lot of nervousness, right? What's the best way to explain what's going on here?

References

Phrack's home page
http://www.phrack.org/

Phrack issue 49, where Project Loki was outlined
http://www.phrack.org/show. php?p=49

Phrack issue 51, for Loki source code and extract.c code
http://www.phrack.org/show.php?p=51

RedHat 4.2 Distribution
ftp://mirror.pacific.net.au/linux/redhat/redhat/linux/4.2/en/os/i386/RedHat/RPMS/

The Character of Loki (in Norse Mythology)
http://www.luth.se/luth/present/sweden/history/gods/johannes/4.the_character_of_loki

Internet Security Systems on Loki
http://www.iss.net/security_center/static/1452.php

McAfee Security's A VERT virus database entry on the Loki virus:
http://vil.nai.com/vil/content/v_724.htm

The National District Attorneys Association on presenting computer forensic concepts in court
http://www.ndaa.org/publications/newsletters/update_volume_15_number_9_2002.html

Also see the Linux man pages for lsof, gunzip and zip, available online at:
lsof(8)
http://linux.ctyme.com/man/man1163.htm
gzip(8)
http://linux.ctyme.com/man/man0777.htm
zip(1)
http://linux.ctyme.com/man/man3282.htm

Part 2: Forensic Analysis of a System

80 Hours in the Life of a Honeypot

Background:
The target of this investigation was a honeypot system, with a nearly default install of RedHat 7.2. The computer was connected be hind an IPTables-based firewall system, configured to forward connections to services running on the honeypot and perform a full packet capture of all traffic via tcpdump. The remainder of the setup duplicated more of the challenges faced in a typical fore nsic examination in that no IDS system was installed nor was the honeypot configured with remote logging or a file-integrity system, like Tripwire. The use of a relatively recent RedHat distribution – 7.2 was released in the second half of 2001 – shortens the list of vulnerabilities compared to the 6.2 version often employed for honeypots but may be more representative of the vintage and configuration of systems deployed.

Firewall logs were periodically monitored to assess activity on the honeypot, as were the honeypot's own logs via the console. The system was online via a cable -modem connection for less than two days before yielding evidence of a potential compromise and was left connected for an additional 31 hours to gather more information from any further activity before being taken offline for investigation. At all times, the system was in a controlled -access location to which only unauthorized personnel were allowed entry. An examination of the packet capture file was postponed until after the forensic investigation of the system.

UDP NTP packets were the only traffic allowed directly between the honeypot and the firewall system, which was also configured to block outbound connections from unprivileged ports (>1204) on the honeypot destined for any p rivileged ports. Other outbound connections originating from the honeypot were to be logged and evaluated for blocking on an individual basis. The firewall box ran no services open to the Internet (except for a source -IP filtered NTP daemon) and was monito red either from the console or from a separate private interface via SSH. Tripwire was installed and set up to monitor configuration files and binaries on the firewall. Arpwatch was installed and configured to alert to any IP address change or ARP -spoofing attempt via the honeypot. The firewall was in the same controlled -access area as the honeypot.

The Honeypot:

- Hardware:
  - A Compaq Presario 1655 laptop with
    - a 266Mhz P2 processor,
    - 64MB of RAM,
    - internal CD-ROM and 1.44MB floppy drives,
    - a PCMCIA 10/100MB Ethernet adapter,
    - a 4 GB Hitachi hard drive

- OS:
  - A stock install of RedHat 7.2 obtained as iso images from a RedHat mirror site,
    - GPG-signed MD5 hashes of the images used to verify the downloaded isos.
    - The only package installed that wasn't on the stock distribution was an upgraded kernel 2.4.9-13 (RPM obtained from same mirror, GPG sig checked) in order to support the specific PCMCIA network card employed.

- Configuration:
  - Prior to the installation, a CD-bootable Linux distribution was used to zero out the hard drive using *dd if=/dev/zero of=/dev/hda*.
  - Of the hard drive's 4 GB, about 2.5 GB were partitioned as follows:
    - / -- 1 GB
    - /boot -- 107 MB
    - /home -- 1 GB
    - swap -- 390 MB
  - The non-swap partitions were formatted as EXT2, not the journaling EXT3 format suggested by the installation program. The remaining space was left unpartitioned.
  - The following services were installed:
    - ftpd (WU-FTP 2.6.1)
    - ntpd (ver 4.1.0)
    - smtpd (Sendmail 8.11.6)
    - fingerd (ver 1.5)
    - xinetd (ver 2.3.3)
    - sshd (OpenSSH 2.9p-2)
    - telnetd (ver 1.24)
    - httpd (Apache 1.3.20)
    - rpc.statd (rstatd v. 1.2)
    - portmap (v 4.0)

Potential Compromise:

The honeypot was christened darth and given the private IP address of 192.168.0.25. It was made available for Internet access around 10 p.m. on March 13, and all services (except for Sendmail) were configured to accept external connections.

After less than 48 hours online in which the honeypot had seen a number of scans, the firewall logs showed a flurry of FTP traffic. After several more hours with l ittle access, the honeypot's own logs were briefly reviewed from the console and showed several segmentation faults thrown by the FTP daemon. A compromise was considered possible. The system was left online for 31 more hours to give its potential cracker further attempts to access it and leave further traces of activity.

Forensic Seizure:
On Mar. 16 at about 9:27 p.m. Singapore time, I started a forensic examination of the honeypot system. The system was running, on mains power and was connected to the network via a PCMCIA Ethernet card. The Ethernet card's connector has an LED array that shows when the link is active, its speed and whether there is traffic on the link. It wasn't registering any network traffic but showed the interface to be up and connected to the network. The honeypot's screen display showed a standard RedHat login prompt.

I inserted the Forensic and Incident Response Environment (FIRE) 0.3.5b bootable Linux CD-ROM distribution. I also inserted a floppy with a script I modified from one bundled with the FIRE distribution for gathering information from a live system (see appendix).

I log in as root and check the time on the honeypot, which registers as 21:32:20 Sun March 16. I mount both the CDROM, which has a selection of statically com piled Linux system binaries, and the floppy. The use of static, trusted binaries is important to avoid relying on the target system's libraries and related files, which may have be modified. Using the trusted *netstat* binary from the CDROM, I execute a *netstat -pan* command, sending the output to a file called "connected.netstat" on the floppy, to gather information on any current connections. Immediately after that, I disconnect the honeypot from the network and connect directly via a crossover network cable to my forensic acquisition system.

Forensic Workstation:

- Hardware:
  - A Dell Inspiron 4150
    - a 2 GHz P4 processor,
    - 384MB of RAM,
    - internal CD R/W-DVD ROM,
    - 20 GB hard drive

- OS:
  - Dual-boost system with
    - RedHat 8.0, patched and up to date on roughly half the drive
    - XP Pro, patched and up to date on a single NTFS partition on the remainder

To the forensic system's eth0 interface, I add a virtual IP address on the same subnet as the honeypot using *ifconfig eth0 add 192.168.0.2 up* .
I start netcat listening on por t 5055 and direct output to the file "process -dump". On the honeypot, I cd to the mounted floppy and execute the script to gather volatile system information, piping the output through netcat to my forensic system (*./volatile-cap.sh | /mnt/cdrom/statbins/l inux2.2_x86/nc 192.168.0.2 5055*).

There's always a tradeoff between altering the state of the system and retrieving volatile data, such as /proc info and contents of RAM. There's no easy answer to the question of when to conduct a live seizure and when to pull the plug right away. Factors such as the system's importance, role and likelihood of any further damage

come into play. In this case, I decided there was potentially greater value in seizing key live information first. Using an in-the-can script can help minimize time-based changes to the system (i.e. cron-driven chores) from the point of seizure to the point of halting it. The script also attempts to order data-gathering to maximize the useful retrieval of volatile data. It also cuts down on typing mistakes that could prove damaging.

After that task terminates, I start netcat listening again on the forensic system and direct output to the file "darth-core". On the honeypot system, I cat the contents of /proc/kcore, piping the output of this memory dump through netcat.

A time comparison between the two systems at this point shows: darth 21:52:08; forensic workstation 21:47:07.

I unmount the floppy and remove it, remove the battery from the laptop, unplug the network cable and pull the power cord, taking down the system hard to freeze the contents of the hard drive with no further changes.

Working With the Dead System:

At this point I copy the netstat information from the floppy onto the forensic system and create MD5 hashes of the files generated in the initial, volatile data acquisition stage.

```
28277091b0d115850a5352679922e8a6  connected.netstat
62e21ef040e70e9e4289fd5803cac9fb  darth -core
19cb0f1840a65065ec4548b9e23d4299  process -dump
```

Next, I boot directly from the FIRE CD-ROM, which is a fully self-contained modern-kernel Linux environment that – apart from a useful selection of static binaries – contains a variety of forensic, incident response and even pen-testing tools. After a little twiddling I get the system to recognize the PCMCIA network card, assign the interface an IP address and hook up the crossover cable to the forensic system.

I list the partitions on the honeypot's /dev/hda using *fdisk -l*. Using *md5sum*, I calculate hashes for all the relevant partitions. Then, I launch another netcat session on the forensic system, and using dd (*dd if=/dev/hda1 | netcat 10.1.1.50 5055*) copy the first of the system partitions /dev/hda1 across, where it's stored as an image file "darth-hda1.img" on the forensic system. The process is repeated for the remaining system partitions and the swap partition. The entire exercise avoids mounting the partitions on the honeypot system or altering the data on the original hard drive, which now must be secured and preserved to maintain its evidentiary value.

The originally calculated MD5 hashes for the partitions are netcatted across and compared with hashes *md5sum* computes for the copied partitions images. They match, assuring that the mirrored images are faithful copies of the partitions. (See the discussion of MD5 hashes in the previous section).

Orignal:

```
5b972c8fabe8906c4f049a0ee65bf353  /dev/hda1
fc3dd7d9a69f48ffff1ae62eb5064376  /dev/hda2
```

```
b55a5c40c5bcfc4ee56ae372738912f7  /dev/hda3
622fd877592fb5544e1252562436499b  /dev/hda5
```

Mirrors:

```
5b972c8fabe8906c4f049a0 ee65bf353  darth -hda1.img
fc3dd7d9a69f48ffff1ae62eb5064376  darth  -hda2.img
b55a5c40c5bcfc4ee56ae372738912f7  darth  -hda3.img
622fd877592fb5544e1252562436499b  darth  -hda5.img
```

Preserving the Evidence:

After disconnecting the honeypot from the network and po wer connections again, I conduct a hardware inventory and stick initialed evidence tags to the components involved.

Custodial record of hardware:

Tag#20030316 -GIAC-Darth-MG10	A 4 GB Hitachi HDD Model DK227A -41 with Serial#ZH14345321
Tag#20030316 -GIAC-Darth-MG20	A PCMCIA Prolink 10/100 Ethernet card Model PFE500c with Serial#KPC00202465
Tag#20030316 -GIAC-Darth-MG30	The laptop, a charcoal -gray Compaq Presario 1655 has no chassis serial number that's externally visible. Using an engraving pen, I carve the tag number (20030316 -GIAC-Darth-MG30) into the bottom of the laptop case, just above the product label.

The computer is placed in a locked, fire -resistant filing cabinet in a controlled area. Access to the area and cabinet key is restricted to authorized personnel, who have to sign in and out.

First View:

I make a quick survey of who was connected to the system when I logged in and executed *netstat*. The "connected.netstat" file shows only ports listening for configured services and no connections to the system. There's no indication our suspected cracker was on the system at the time it was last accessible to the Internet. This is good because otherwise he may have noted the console log -in and worked to cover any tracks. It's for this reason that I watch ed for Ethernet link activity for a period before logging in.

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address  State   PID/Program name
tcp        0      0 0.0.0.0:1024   0.0.0.0:*       LISTEN  546/   rpc.statd
tcp        0      0 0.0.0.0:513    0.0.0.0:*       LISTEN  879/xinetd
tcp        0      0 0.0.0.0:514    0.0.0.0:*       LISTEN  879/xinetd
tcp        0      0 0.0.0.0:79     0.0.0.0:*       LISTEN  879/xinetd
tcp        0      0 0.0.0.0:111    0.0.0.0:*       LISTEN  518/portmap
tcp        0      0 0.0.0.0:80     0.0.0.0:*       LISTEN  1889/httpd
tcp        0      0 0.0.0.0:81     0.0.0.0:*       LISTEN  1889/httpd
tcp        0      0 0.0.0.0:21     0.0.0.0:*       LISTEN  879/xinetd
tcp        0      0 0.0.0.0:22     0.0.0.0:*       LISTEN  810/sshd
```

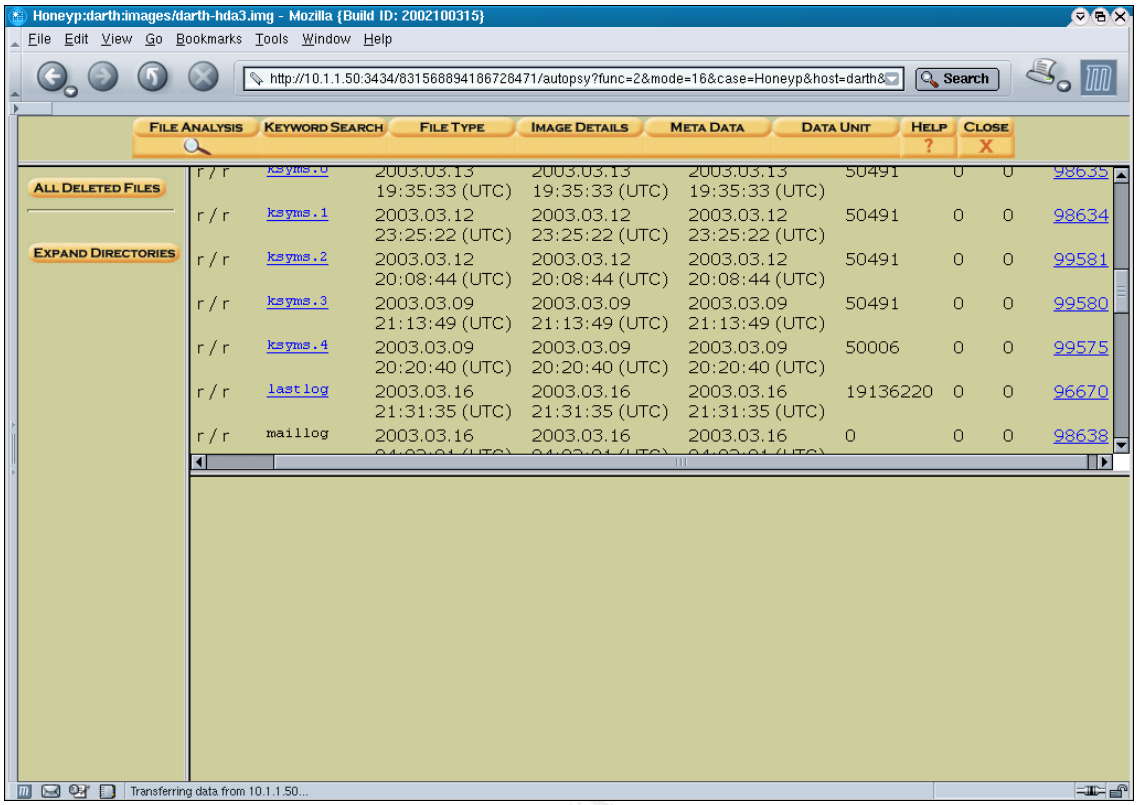
```
tcp        0      0 0.0.0.0:23      0.0.0.0:*       LISTEN  879/xinetd
tcp        0      0 127.0.0.1:2 5   0.0.0.0:*        LISTEN
1511/sendmail:acce
tcp        0      0 0.0.0.0:443     0.0.0.0:*       LISTEN  1889/httpd
udp        0      0 0.0.0.0:1024    0.0.0.0:*               546/rpc.statd
udp        0      0 0.0.0.0:1025    0.0.0.0:*                 679/ntpd
udp        0      0 0.0.0.0:517     0.0.0.0:*               879/xinetd
udp        0      0 0.0.0.0:722     0.0.0.0:*               546/rpc.statd
udp        0      0 0.0.0.0:111     0.0.0.0:*               518/p   ortmap
udp        0      0 127.0.0.1:123   0.0.0.0:*               679/ntpd
udp        0      0 0.0.0.0:123     0.0.0.0:*               679/ntpd

Active UNIX domain sockets (servers and established)
Proto RefCnt Flags    Type   State      I-Node PID/Program name    Path
unix  11     [ ]     DGRAM              854    493/syslogd         /dev/log
unix  2      [ ACC ] STREAM LISTENING   1323   975/gpm             /dev/gpmctl
unix  2      [ ]     DGRAM             13487  4873/login   -- root
unix  2      [ ]     DGRAM              2260  1511/sendmail: acce
unix  2      [ ]     DGRAM              1367  1016/crond
unix  2      [ ]     DGRAM              1217   879/xinetd
unix  2      [ ]     DGRAM              1120   771/cardmgr
unix  2      [ ]     DGRAM              1042   679/ntpd
unix  2      [ ]     DGRAM              1027   659/apmd
unix  2      [ ]     DGRAM               911   546/rpc.statd
unix  2      [ ]     DGRAM               863   498/klogd
unix  2      [ ]     STREAM CONNECTED    631     1/init [3]
```

Filesystem Autopsy:

To examine the filesystem images, I opted to use The @stake Sleuth Kit (TASK) ver 1.60 and the companion Autopsy browser 1.70. (The TASK project, led by Brian Carrier, uses much of th e code from The Coroners Toolkit developed by Dan Farmer and Wietse Venema.) Used together on a Linux platform, the forensic tools allow the safe examination of images of a range of filesystems from the Windows and Unix worlds. Mounting the filesystems isn 't necessary, and Autopsy, a graphical front -end for TASK, helps provide a framework for and speeds up many of the analytical operations.

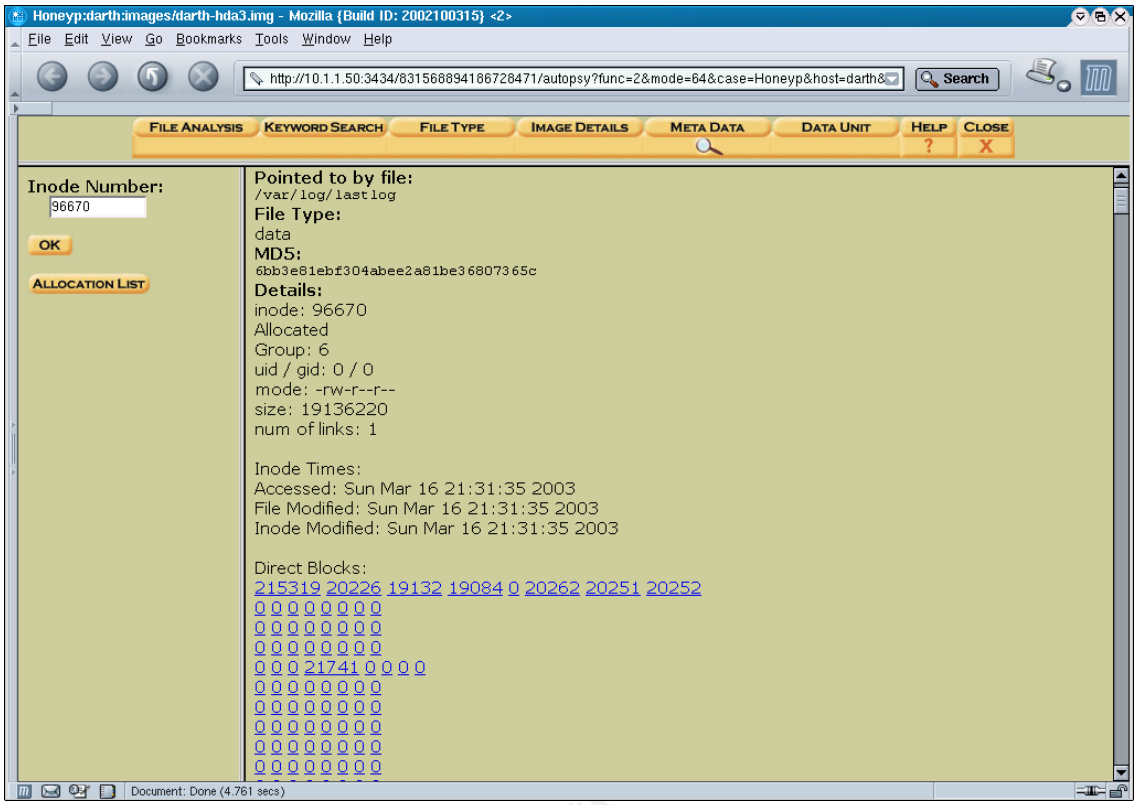
First, I have to start a case and record myself as investigator, then set up a host file for the compromised system, darth. The host file also is where the timezone of the system is set, in this case UTC -8. Then, I have to configure the images associated with the host. Only images that are filesystem partitions (not swap partitions) can be selected for analysis. I regis ter the image files of the three partitions: “/”, “/boot” and “/home”. To look at the swap image and the /proc/core file, I can use some of the TASK tools but will also employ other methods.

One of the first things I want to view is the system's logs. Whi le these are highly unreliable on a compromised host – any number of automated hacker tools exist to clean logs – often they can provide clues anyway. I select the root partition, click the "File Analysis" button, and browse to the /var/log/ directory. Eve n before examining the log files, I see one thing that piques my curiosity: the lastlog file, which contains data on users' login histories, is enormous, about 19 MB. It's showing it was last modified, accessed and inode data changed at the same time, 9:31 :35 p.m., close to when I logged into the system console as root to start the forensic seizure.



This isn't a text file like most other logs, but I click on the link to view its contents anyway. That proves a mistake. The disk starts thrashing and the Autopsy browser process bogs down and stalls.

After a couple of minutes, I kill the process and start again. This time I look at the data contained in the inode associated with the file and see the problem. The inode data seems corrupted, listing Block 0 as a data block for the file scores of times, along with a few valid data blocks.



At this point, I don't know if the corruption occurred because of the harsh shutdown or as a byproduct of a compromise. I select the “export contents” button to copy th e raw data blocks into a file “images -darth-hda3.img-inode96670.raw” for later analysis.

Log Analysis:

Then I look at the main “messages” logs. The syslog daemon appears to have been running until the system was taken off -line, recording my actions as I started to capture volatile data and including my disconnecting the network wire and reconnecting it to the forensic workstation.

```
Mar 16 21:31:35 darth login(pam_unix)[4873]: session opened for user root
by LOGIN(uid=0)
Mar 16 21:31:35 darth   -- root[4873]: ROOT LOGIN ON tty3
Mar 16 21:32:23 darth kernel: ide -floppy driver 0.97.sv
Mar 16 21:32:23 darth kernel: hdc: ATAPI 24X CD -ROM DVD-RAM drive, 128kB
Cache, DMA
Mar 16 21:32:23 darth kernel: Uniform CD -ROM driver Revision: 3.12
Mar 16 21:36:51 darth kerne l: eth0: lost link beat
Mar 16 21:37:03 darth kernel: eth0: found link beat
Mar 16 21:37:03 darth kernel: eth0: autonegotiation complete: 100baseT   -FD
selected
Mar 16 21:54:00 darth kernel: eth0: lost link beat
```

Reviewing earlier activity, I find the logs s how little obvious evidence of tampering – no blank lines, stray spaces or other formatting errors. Timestamps seem logical, except that ftp daemon messages seem to be on a different timezone setting, but

track in terms of minutes and seconds. After some c hecking, I discover the configuration of this version of WU -FTP on RedHat appears to need some customization to be set to the same timezone as the system but that it doesn't affect accuracy of logging. The time difference is -8 hours for the ftpd log entri es. Onward.

The log shows several FTP failures to login as root, some incomplete connections and some successes as anonymous users. Then, I see the following:

```
Mar 15 14:00:29 darth ftpd[4372]: FTP session closed
Mar 15 14:03:51 darth ftpd[4383]: FTP sess  ion closed
Mar 15 06:07:24 darth ftpd[4387]: ANONYMOUS FTP LOGIN FROM 202.101.xx.xxx
[202.101.xx.xxx], mozilla@
Mar 15 06:08:59 darth ftpd[4387]: exiting on signal 11: Segmentation fault
Mar 15 06:08:59 darth ftpd[4387]: exiting on signal 11: Segmentation    fault
Mar 15 14:09:54 darth ftpd[4388]: FTP session closed
Mar 15 06:10:01 darth ftpd[4389]: ANONYMOUS FTP LOGIN FROM 202.101.xx.xxx
[202.101.xx.xxx], mozilla@
Mar 15 14:10:41 darth kernel: eth0: interrupt(s) dropped!
Mar 15 06:11:12 darth ftpd[4389]: exit  ing on signal 11: Segmentation fault
Mar 15 21:17:20 darth login(pam_unix)[1061]: session opened for user root
by LOGIN(uid=0)
Mar 15 21:17:20 darth   -- root[1061]: ROOT LOGIN ON tty3
Mar 15 21:30:18 darth login(pam_unix)[1061]: session closed for user roo   t
```

Without prior connections, entries like the first two FTP session closed entries usually signal incomplete connections, such as scanners and banner harvesters. Then, there's an anonymous login for which the remote host gave the password (supposed to be an e-mail address) as mozilla@. That's at 14:07:24 local time (remember the timezone issue). A minute and 35 seconds later the ftp daemon throws a segfault (repeated in the logs).

Another anonymous login from the same IP address and using the same passwo rd starts a little more than a minute later, and within 41 seconds an interrupt dropped message appears (not clear if this relates to the potential exploit being run or to a little touchiness in the PCMCIA driver, the first version supporting this particul ar network card). In any case, about a minute and 11 seconds after login, *ftpd* reports another segfault. Now, this is never a good sign – it basically means the program accessed a memory location that it wasn't assigned – and can be an indicator of a buffer overflow-style exploit. The subsequent root login at 21:17 is from the console and corresponds with a known monitoring of the honeypot system.

One more FTP connection attempt is listed as a session -closed failure hours later in the logs. No other connec tions are recorded, but of course we have to distrust this.

Next, I review the other logs. The /var/log/secure logs show the FTP connections, along with earlier, failed attempts to login via the telnet and ssh services. No more -- nor fewer – FTP connections are reported. Xinetd, the daemon that is responsible for starting ftpd processes when connections are requested, reports that the longest of the sessions from the 202.101.xx.xxx IP address lasted exactly 100 seconds, probably long enough to start a bac kdoor process. And again, even this information is suspect.

Challenger #1?

The xferlog, which records FTP file transfers, shows us the results of an earlier anonymous session: three transfers of the ftpd system's chrooted passwd file to the same IP address, although different from the one associated with the segfaults (nasXXXX-120.mystarhub.com.sg corresponds to 203.117.xx.xxx). This is a pretty lame attempt at compromise. That file contains just a handful of default usernames "root, operator" etc and no password information. Our would -be miscreant gets slightly higher points for humor in the selection of an e -mail address to submit at the password prompt in one attempt. These transfers all preceded the FTP segfault sessions time by about 13 hours.

```
Sat Mar 15 00:46:29 2003 1 nasXXXX.mystarhub.com.sg 79
/var/ftp/etc/passwd a _ o a ftp ftp 0 * c
Sat Mar 15 00:47:13 2003 1 nasXXXX.mystarhub.com.sg 79
/var/ftp/etc/passwd a _ o a ftp ftp 0 * c
Sat Mar 15 00:53:45 2003 1 nasXXXX.mystarhub.com.sg 79
/var/ftp/ etc/passwd a _ o a bill.gates@microsoft.com ftp 0 * c
```

The rest of the logs don't reveal much suspicious. Still, we at least have one area that merits thorough examination, the FTP connections that triggered segfaults. And, it's time to step back and cover some other basics before digging deeper there.

Back to Basics:

First, I check the information that I dumped from the still running honeypot using trusted, static binaries. I'm keen to see if the output from the honeypot's own *ps* command differs significantly from that generated by the statically compiled *lsof*. Any differences would indicate processes were hidden on the system. In addition, the *lsof* listing, which shows not just processes running but also their associated open files, would reveal processes using unusual or inappropriate libraries and other files.

```
==========================
List of running processes from honeypot's ps:
==========================
  F S UID   PID PPID C  PRI  NI ADDR SZ WCHAN STIME TTY TIME     CMD
100 S root  1   0    0  68    0   -  353 do_sel Mar13 ?  00:00:04 init [3]
040 S root  2   1    0  69   0    -   0  contex Mar13 ?  00:00:00 [keventd]
040 S root  3   1    0  69   0    -   0  apm_ma Mar13 ?  00:00:00 [kapm -idled]
040 S root  4   0    0  79  19    -   0  ksofti Mar13 ?   00:00:00 [ksoftirqd_CPU0]
040 S root  5   0    0  69   0    -   0  kswapd Mar13 ?  00:00:01 [kswapd]
040 S root  6   0    0  69   0    -   0  krecla Mar13 ?  00:00:00 [kreclaimd]
040 S root  7   0    0  69   0    -   0  bdflus Mar13 ?  00:00:00 [bdflush]
040 S root  8   0    0  69   0    -   0  kupdat Mar13 ?  00:00:00 [kupdated]
040 S root  9   1    0  59  -20   -   0  md_thr Mar13 ?  00:00:00 [mdrecoveryd]
040 S root 80   1    0  69   0    -   0  end    Mar13 ?  00:00:00 [khubd]
140 S root 493  1    0  69   0    -  368 do_sel Mar13 ?  00:00:00 syslogd  -m 0
140 S root 498  1    0  69   0    -  493 do_sys Mar13 ?  00:00:00 klogd  -2
140 S rpc  518  1    0  69   0    -  389 do_pol Mar13 ?  00:00:00 portmap
140 S rpcuser546 1   0  69   0    -  399 do_sel Mar13 ?  00:00: 00 rpc.statd
140 S root 659  1    0  68   0    -  349 do_sel Mar13 ?  00:00:00 /usr/sbin/apmd   -p
10 -w 5 -W -P /etc/sysconfig/apm -scripts/apmscript
140 S ntp  679  1    0  69   0    -  482 do_sel Mar13 ?  00:00:00 ntpd  -U ntp
140 S root 771  1    0  69   0    -  383 do_sel Mar13 ?  00:00:00 /sbin/cardmgr
140 S root 810  1    0  69   0    -  669 do_sel Mar13 ?  00:00:02 /usr/sbin/sshd
140 S root 879  1    0  69   0    -  566 do_sel Mar13 ?  00:00:00 xinetd  -stayalive
-reuse -pidfile /var/run/xinetd.pid
```

```
140 S root  975  1    0  69   0    -  360 do_sel Mar13 ?  00:00:00 gpm  -t ps/2 -m
/dev/mouse
040 S root 1016 1    0  66   0    -  396 nanosl Mar13 ?  00:00:00 crond
040 S daemon 1052 1  0  69   0    -  361 nanosl Mar13 ?  00:00:00 /usr/sbin/atd
100 S root 1060 1    0  6 9   0    -  346 read_c Mar13 tty2 00:00:00 /sbin/mingetty
tty2
100 S root 1062 1    0  69   0    -  346 read_c Mar13 tty4 00:00:00 /sbin/mingetty
tty4
100 S root 1063 1    0  69   0    -  346 read_c Mar13 tty5 00:00:00 /sbin/mingetty
tty5
100 S root 1064 1     0  69   0    -  346 read_c Mar13 tty6 00:00:00 /sbin/mingetty
tty6
140 S root 1511 1    0  69   0    - 1324 do_sel Mar13 ?  00:00:00 sendmail:
accepting connections
140 S root 1889 1    0  69   0    - 11595 do_sel Mar13 ? 00:00:05 /usr/sbin/httpd   -
DHAVE_ACCE SS -DHAVE_PROXY -DHAVE_AUTH_ANON -DHAVE_ACTIONS -DHAVE_ALIAS -DHAVE_ASIS
-DHAVE_AUTH -DHAVE_AUTOINDEX -DHAVE_AUTH_DB -DHAVE_AUTH_DBM -DHAVE_CERN_META -
DHAVE_CGI -DHAVE_DIGEST -DHAVE_DIR -DHAVE_ENV -DHAVE_EXAMPLE -DHAVE_EXPIRES -
DHAVE_HEADERS -DHAVE_IMAP -DHAVE_INCLUDE -DHAVE_INFO -DHAVE_LOG_AGENT -
DHAVE_LOG_CONFIG -DHAVE_LOG_REFERER -DHAVE_MIME -DHAVE_MIME_MAGIC -
DHAVE_MMAP_STATIC -DHAVE_NEGOTIATION -DHAVE_REWRITE -DHAVE_SETENVIF -DHAVE_SPELING
-DHAVE_STATUS -DHAVE_UNIQUE_ID -DHAVE_USERDIR -DHAVE_USERTRACK -DHAVE_VHOST_ALIAS
-DHAVE_DAV -DHAVE_PERL -DHAVE_SSL -DHAVE_PHP4
100 S root 4034 1    0  69   0    -  346 read_c Mar15 tty1 00:00:00 /sbin/mingetty
tty1
100 S root 4873 1    0  69   0    -  580 wait4  Mar15 tty3 00:00:00 login  -- root
140 S apache 5314 1889  0 69  0    - 11606 do_sel 04:02 ? 00:00:00 /usr/sbin/httpd
(REPEATED ARUGMENTS REMOVED FOR BREVITY, SEE PID 1889)
140 S apache 5315 1889 0 69  0    - 11606 semop 04:02 ?  00:00:00 /usr/sbin/httpd
(REPEATED ARUGMENTS REMOVED FOR BREVITY, SEE PID 1889)
140 S apache 5316 1889 0 69  0    - 11606 semop 04:02 ?  00:00:00 /usr/sbin/httpd
(REPEATED ARUGMENTS REMOVED FOR BREVITY, SEE PID 1889)
140 S apache 5317 1889 0 69  0    - 11606 semop 04:02 ?  00:00:00 /usr/sbin/httpd
(REPEATED ARUGMENTS REMOVED FOR BREVITY, SEE P  ID 1889)
140 S apache 5321 1889 0 69  0    - 11606 semop 04:02 ?  00:00:00 /usr/sbin/httpd
(REPEATED ARUGMENTS REMOVED FOR BREVITY, SEE PID 1889)
100 S root 8698 4873 0   70  0    -  614 wait4  21:31 tty3 00:00:00  -bash
040 S root 8761 8698 1   71  0    -  614 wait4  21:43 tty3 00:00:00  -bash
000 S root 8762 8698 0   71  0    -  457 do_sel 21:43 tty3 00:00:00 ./nc
192.168.0.2 5055
000 R root 8798 8761 0   75  0    -  770 -      21:43 tty3 00:00:00 ps  -eflwwww
```

This compares with the lsof output below (with some listings truncated, as noted, for space reasons):

```
============
lsof output:
============
COMMAND    PID USER   FD   TYPE  DEVICE    SIZE    NODE NAME
init         1 root  cwd    DIR     3,3    4096       2 /
init         1 root  rtd    DIR     3,3    4096         2 /
init         1 root  txt    REG     3,3   26636    1361 /sbin/init
<<< KERNEL PROCESS LIST TRUNCATED TO SAVE SPACE >>>>
khubd       80 root   10u  FIFO     3,3           70337 /dev/initctl
syslogd    493 root  cwd    DIR     3,3    4096       2   /
syslogd    493 root  rtd    DIR     3,3    4096       2 /
syslogd    493 root  txt    REG     3,3   26972    1204 /sbin/syslogd
syslogd    493 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
syslogd    493 root  mem    REG     3,3 5772268     3  72 /lib/i686/libc -
2.2.4.so
```

```
syslogd    493 root  mem    REG     3,3  261460   96716 /lib/libnss_files   -
2.2.4.so
syslogd    493 root    0u  unix 0xc3de6a80          854 /dev/log
syslogd    493 root    2w   REG     3,3     586   98636 /var/log/messages
syslogd    493 root    3w   REG     3,3       0   98637 /var/log/secure
syslogd    493 root    4w   REG     3,3       0   98638 /var/log/maillog
syslogd    493 root    5w   REG     3,3   26157   98641 /var/log/cron
syslogd    493 root    6w   REG     3,3       0    98639 /var/log/spooler
syslogd    493 root    7w   REG     3,3       0   98640 /var/log/boot.log
syslogd    493 root    8w   REG     3,3       0   35717
/var/log/news/news.crit
syslogd    493 root    9w   REG     3,3       0   35718
/var/log/news/news.er r
syslogd    493 root   10w   REG     3,3       0   35716
/var/log/news/news.notice
klogd      498 root  cwd    DIR     3,3    4096       2 /
klogd      498 root  rtd    DIR     3,3    4096       2 /
klogd      498 root  txt    REG     3,3   20476    1203   /sbin/klogd
klogd      498 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
klogd      498 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
klogd      498 root    0r   REG     0,2       0    4116 /proc/kmsg
klogd      498 root    1 u  unix 0xc3de6040            863 socket
klogd      498 root    2r   REG     3,1  438077      25 /boot/System.map   -
2.4.9-13
portmap    518 root  cwd    DIR     3,3    4096       2 /
portmap    518 root  rtd    DIR     3,3    4096       2 /
portmap    518 r oot  txt    REG     3,3   29724    1579 /sbin/portmap
portmap    518 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
portmap    518 root  mem    REG     3,3  436784   96700 /lib/libnsl   -
2.2.4.so
portmap    518 root  mem    REG     3,3 5772268       372 /lib/i686/libc -
2.2.4.so
portmap    518 root  mem    REG     3,3  261460   96716 /lib/libnss_files   -
2.2.4.so
portmap    518 root    0u   CHR     1,3           66419 /dev/null
portmap    518 root    1u   CHR     1,3           66419 /dev/null
portmap    51 8 root    2u   CHR     1,3           66419 /dev/null
portmap    518 root    3u  IPv4     883             UDP *:sunrpc
portmap    518 root    4u  IPv4     884             TCP *:sunrpc (LISTEN)
rpc.statd  546 root  cwd    DIR     3,3    4096   35567 /var/li  b/nfs/statd
rpc.statd  546 root  rtd    DIR     3,3    4096       2 /
rpc.statd  546 root  txt    REG     3,3   23576    1624 /sbin/rpc.statd
rpc.statd  546 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
rpc.statd  546 root  mem    REG     3,3   436784   96700 /lib/libnsl -
2.2.4.so
rpc.statd  546 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
rpc.statd  546 root  mem    REG     3,3  261460   96716 /lib/libnss_files   -
2.2.4.so
rpc.statd  546 root  mem    REG     3,3  355236   96724
/lib/libnss_nisplus -2.2.4.so
rpc.statd  546 root    0u   CHR     1,3           66419 /dev/null
rpc.statd  546 root    1u   CHR     1,3           66419 /dev/null
rpc.statd  546 root    2u   CHR     1,3           66419 /dev/null
rpc.statd  546 root    3u  uni x 0xc3a3bac0           911 socket
rpc.statd  546 root    4u  IPv4     912             UDP *:722
rpc.statd  546 root    5u  IPv4     926             UDP *:1024
rpc.statd  546 root    6u  IPv4     929             TCP *:1024 (LISTEN)
apmd       659 root  c wd    DIR     3,3    4096       2 /
apmd       659 root  rtd    DIR     3,3    4096       2 /
```

```
apmd       659 root  txt    REG     3,3   16444   98633 /usr/sbin/apmd
apmd       659 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
apmd       659 roo t  mem    REG     3,3 5772268     372 /lib/i686/libc  -
2.2.4.so
apmd       659 root    0u   CHR    10,134         65338 /dev/apm_bios
apmd       659 root    1u  unix 0xc130e060           1027 socket
ntpd       679 root  cwd    DIR     3,3    4096       2 /
ntpd       679 root  rtd    DIR     3,3    4096       2 /
ntpd       679 root  txt    REG     3,3  252044   99203 /usr/sbin/ntpd
ntpd       679 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
ntpd       679 root  mem    REG     3,3  622317     37  4 /lib/i686/libm -
2.2.4.so
ntpd       679 root  mem    REG     3,3   44851   19616
/usr/lib/libcap.so.1.10
ntpd       679 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
ntpd       679 root  mem    REG     3,3  261460   96716 /lib/libnss_fi   les-
2.2.4.so
ntpd       679 root    3u  unix 0xc3a83a80          1042 socket
ntpd       679 root    4u  IPv4    1047             UDP *:ntp
ntpd       679 root    5u  IPv4    1048             UDP darth:ntp
ntpd       679 root    7u  IPv4    1049               UDP *:1025
cardmgr    771 root  cwd    DIR     3,3    4096  113778 /etc/pcmcia
cardmgr    771 root  rtd    DIR     3,3    4096       2 /
cardmgr    771 root  txt    REG     3,3   37260    1497 /sbin/cardmgr
cardmgr    771 root  mem    REG     3,3  4851  71   96682 /lib/ld -2.2.4.so
cardmgr    771 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
cardmgr    771 root    0u  unix 0xc3a64aa0         1120 socket
cardmgr    771 root    1u   CHR   254,0          113882 /var/lib/pcmcia/cm   -
771 (delet ed)
cardmgr    771 root    2u   CHR   254,1          113883 /var/lib/pcmcia/cm   -
771 (deleted)
sshd       810 root  cwd    DIR     3,3    4096       2 /
sshd       810 root  rtd    DIR     3,3    4096       2 /
sshd       810 root  txt    REG     3,3  246220     99461 /usr/sbin/sshd
sshd       810 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
sshd       810 root  mem    REG     3,3   35424   98372 /lib/libpam.so.0.75
sshd       810 root  mem    REG     3,3   65997   96695 /lib/libdl  -2.2.4.so
sshd       810 root  mem    REG     3,3   59618   19585
/usr/lib/libz.so.1.1.3
sshd       810 root  mem    REG     3,3  436784   96700 /lib/libnsl   -
2.2.4.so
sshd       810 root  mem    REG     3,3   47872   96735 /lib/libutil   -
2.2.4.so
sshd       810 root  mem    R EG     3,3  918752   97319
/lib/libcrypto.so.0.9.6b
sshd       810 root  mem    REG     3,3 5772268    372 /lib/i686/libc   -
2.2.4.so
sshd       810 root    0u   CHR     1,3           66419 /dev/null
sshd       810 root    1u   CHR     1,3           66419 /de  v/null
sshd       810 root    2u   CHR     1,3           66419 /dev/null
sshd       810 root    3u  IPv4    1163             TCP *:ssh (LISTEN)
xinetd     879 root  cwd    DIR     3,3    4096       2 /
xinetd     879 root  rtd    DIR     3,3    4096         2 /
xinetd     879 root  txt    REG     3,3  143660   99309 /usr/sbin/xinetd
xinetd     879 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
xinetd     879 root  mem    REG     3,3  436784   96700 /lib/libnsl   -
2.2.4.so
xinetd     879 root  mem    R EG     3,3  622317     374 /lib/i686/libm  -
2.2.4.so
```

```
xinetd     879 root  mem    REG     3,3   85115   96693 /lib/libcrypt   -
2.2.4.so
xinetd     879 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
xinetd     879 root  mem    REG     3,3  26146  0   96716 /lib/libnss_files -
2.2.4.so
xinetd     879 root  mem    REG     3,3  355236   96724
/lib/libnss_nisplus -2.2.4.so
xinetd     879 root  mem    REG     3,3   72296   96713 /lib/libnss_dns   -
2.2.4.so
xinetd     879 root  mem    REG     3,3  261196   967  29 /lib/libresolv -
2.2.4.so
xinetd     879 root    0r   CHR     1,3           66419 /dev/null
xinetd     879 root    1r   CHR     1,3           66419 /dev/null
xinetd     879 root    2r   CHR     1,3           66419 /dev/null
xinetd     879 root    3u  IPv4     1220             TCP *:finger (LISTEN)
xinetd     879 root    4u  IPv4    1221            TCP *:login (LISTEN)
xinetd     879 root    5u  IPv4    1222            TCP *:shell (LISTEN)
xinetd     879 root    6u  unix 0xc38e4060          1217 socket
xinetd     879 root    7u  IPv4    1223            UDP *:talk
xinetd     879 root    8u  IPv4    1224            TCP *:telnet (LISTEN)
xinetd     879 root    9u  IPv4    1225            TCP *:ftp (LISTEN)
gpm        975 root  cwd    DIR     3,3    4096       2 /
gpm        975 root  rtd    DIR     3,3    4096       2 /
gpm        975 root  txt    REG     3,3   62556   98452 /usr/sbin/gpm
gpm        975 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
gpm        975 root  mem    REG     3,3 577226  8     372 /lib/i686/libc -
2.2.4.so
gpm        975 root    0w   CHR     5,1           65373 /dev/console
gpm        975 root    1u   REG     3,3        4   2879 /var/run/gpm8me73r
(deleted)
gpm        975 root    2u   CHR    10,1           66537 /dev/psaux
gpm        975 root    3u  unix 0xc04f6a80         1323 /dev/gpmctl
crond     1016 root  cwd    DIR     3,3    4096     368 /var/spool
crond     1016 root  rtd    DIR     3,3    4096       2 /
crond     1016 root  txt    REG     3,3   21852   99169 /usr/sbi  n/crond
crond     1016 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
crond     1016 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
crond     1016 root  mem    REG     3,3  261460   96716 /lib/libnss_files   -
2.2.4.so
crond     10 16 root  mem    REG     3,3  355236   96724
/lib/libnss_nisplus -2.2.4.so
crond     1016 root  mem    REG     3,3  436784   96700 /lib/libnsl   -
2.2.4.so
crond     1016 root    0u   CHR     5,1           65373 /dev/console
crond     1016 root    1w  FIFO     0 ,0            1363 pipe
crond     1016 root    2w  FIFO     0,0            1364 pipe
crond     1016 root    3u   REG     3,3       5    2883 /var/run/crond.pid
crond     1016 root    4u  unix 0xc3656080           1367 socket
atd       1052 root  cwd    DI R     3,3    4096  113097 /var/spool/at
atd       1052 root  rtd    DIR     3,3    4096       2 /
atd       1052 root  txt    REG     3,3   14832   97291 /usr/sbin/atd
atd       1052 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
atd       1052  root  mem    REG     3,3 5772268      372 /lib/i686/libc  -
2.2.4.so
atd       1052 root  mem    REG     3,3  261460   96716 /lib/libnss_files   -
2.2.4.so
atd       1052 root    0u   CHR     1,3           66419 /dev/null
atd       1052 root    1u   CHR     1,3            66419 /dev/null
atd       1052 root    2u   CHR     1,3           66419 /dev/null
```

```
atd       1052 root    3u   REG     3,3       5    3077 /var/run/atd.pid
mingetty  1060 root  cwd    DIR     3,3    4096       2 /
mingetty  1060 root  rtd    DIR      3,3    4096       2 /
<<<< MINGETTY TRUNCATED AFTER LIST OF OPENED FILES WAS REVIEWED >>>>>
sendmail  1511 root  cwd    DIR     3,3    4096  113624 /var/spool/mqueue
sendmail  1511 root  rtd    DIR     3,3    4096       2 /
sendmail  1511 root  txt    RE G     3,3  451076   98544 /usr/sbin/sendmail
sendmail  1511 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
sendmail  1511 root  mem    REG     3,3    8371   98417
/usr/lib/sasl/libanonymous.so.1.0.15
sendmail  1511 root  mem    REG     3,3  7548  70   96740 /lib/libdb -3.2.so
sendmail  1511 root  mem    REG     3,3  261196   96729 /lib/libresolv   -
2.2.4.so
sendmail  1511 root  mem    REG     3,3   85115   96693 /lib/libcrypt   -
2.2.4.so
sendmail  1511 root  mem    REG     3,3  436784   96700 /lib/libnsl   -
2.2.4.so
sendmail  1511 root  mem    REG     3,3  200192   98491
/lib/libldap.so.2.0.6
sendmail  1511 root  mem    REG     3,3   44728   98489
/lib/liblber.so.2.0.6
sendmail  1511 root  mem    REG     3,3   48683   19578
/usr/lib/libsasl.so.7.1.8
sendmail  1511 root  mem    REG     3,3   30114   16567
/usr/lib/libgdbm.so.2.0.0
sendmail  1511 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
sendmail  1511 root  mem    REG     3,3   80016  113594
/usr/kerberos/lib/libkrb4.so.2.0
sendmail  1511  root  mem    REG     3,3   17552  113586
/usr/kerberos/lib/libdes425.so.3.0
sendmail  1511 root  mem    REG     3,3  425493  113595
/usr/kerberos/lib/libkrb5.so.3.0
sendmail  1511 root  mem    REG     3,3   78193  113590
/usr/kerberos/lib/libk5crypto.so.3  .0
sendmail  1511 root  mem    REG     3,3    8713  113585
/usr/kerberos/lib/libcom_err.so.3.0
sendmail  1511 root  mem    REG     3,3  207036   97320
/lib/libssl.so.0.9.6b
sendmail  1511 root  mem    REG     3,3  918752   97319
/lib/libcrypto.so.0.9.6b
sendmail  1511 root  mem    REG     3,3   65997   96695 /lib/libdl   -2.2.4.so
sendmail  1511 root  mem    REG     3,3   35424   98372 /lib/libpam.so.0.75
sendmail  1511 root  mem    REG     3,3  261460   96716 /lib/libnss_files   -
2.2.4.so
sendmail  1511 root  me m    REG     3,3  355236   96724
/lib/libnss_nisplus -2.2.4.so
sendmail  1511 root  mem    REG     3,3   13708   98439
/usr/lib/sasl/libcrammd5.so.1.0.15
sendmail  1511 root  mem    REG     3,3   33637   98442
/usr/lib/sasl/libdigestmd5.so.0.0.17
sendmail  1511 root  mem    REG     3,3   10778   98445
/usr/lib/sasl/liblogin.so.0.0.5
sendmail  1511 root  mem    REG     3,3   10368   98448
/usr/lib/sasl/libplain.so.1.0.14
sendmail  1511 root    0r   CHR     1,3           66419 /dev/null
sendmail  1511 root     1w   CHR     1,3           66419 /dev/null
sendmail  1511 root    2w   CHR     1,3           66419 /dev/null
sendmail  1511 root    3u  unix 0xc3a64060         2260 socket
sendmail  1511 root    4u  IPv4    2261             TCP darth:smtp (LISTEN)
httpd     1889 root  cwd    DIR     3,3    4096       2 /
```

```
httpd     1889 root  rtd    DIR     3,3    4096       2 /
httpd     1889 root  txt    REG     3,3  330668   99562 /usr/sbin/httpd
<<<< HTTPD & CHILDREN TRUNCATED >>>>>>
httpd     1889 root   23w   REG     3 ,3       0  113881
/var/log/httpd/ssl_request_log
mingetty  4034 root  cwd    DIR     3,3    4096       2 /
mingetty  4034 root  rtd    DIR     3,3    4096       2 /
mingetty  4034 root  txt    REG     3,3    8636     441 /sbin/mingetty
mingetty  4034 root   mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
mingetty  4034 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
mingetty  4034 root    0u   CHR     4,1           69140 /dev/tty1
mingetty  4034 root    1u   CHR     4,1           69140   /dev/tty1
mingetty  4034 root    2u   CHR     4,1           69140 /dev/tty1
login     4873 root  cwd    DIR     3,3    4096       2 /
login     4873 root  rtd    DIR     3,3    4096       2 /
login     4873 root  txt    REG     3,3   17740    1391 /bin/log   in
login     4873 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
login     4873 root  mem    REG     3,3    6144  113495
/lib/security/pam_nologin.so
login     4873 root  mem    REG     3,3   85115   96693 /lib/libcrypt   -
2.2.4.so
login     4873 r oot  mem    REG     3,3   35424   98372 /lib/libpam.so.0.75
login     4873 root  mem    REG     3,3   65997   96695 /lib/libdl   -2.2.4.so
login     4873 root  mem    REG     3,3   12079   98373
/lib/libpam_misc.so.0.75
login     4873 root  mem    REG     3,3  5772268     372 /lib/i686/libc -
2.2.4.so
login     4873 root  mem    REG     3,3    7841  113500
/lib/security/pam_securetty.so
login     4873 root  mem    REG     3,3   12060  113502
/lib/security/pam_stack.so
login     4873 root  mem    REG     3,3   501  55  113478
/lib/security/pam_console.so
login     4873 root  mem    REG     3,3    4994  113480
/lib/security/pam_deny.so
login     4873 root  mem    REG     3,3  182363   16569 /usr/lib/libglib   -
1.2.so.0.0.10
login     4873 root  mem    REG     3,3  261460     96716 /lib/libnss_files -
2.2.4.so
login     4873 root  mem    REG     3,3   13025  113481
/lib/security/pam_env.so
login     4873 root  mem    REG     3,3   47819  113506
/lib/security/pam_unix.so
login     4873 root  mem    REG     3,3  436784   96700 /  lib/libnsl -
2.2.4.so
login     4873 root  mem    REG     3,3   14636  113479
/lib/security/pam_cracklib.so
login     4873 root  mem    REG     3,3   69084   16549
/usr/lib/libcrack.so.2.7
login     4873 root  mem    REG     3,3   13778  113489
/lib/security /pam_limits.so
login     4873 root  mem    REG     3,3  355236   96724
/lib/libnss_nisplus -2.2.4.so
login     4873 root    0u   CHR     4,3           69162 /dev/tty3
login     4873 root    1u   CHR     4,3           69162 /dev/tty3
login     4873 root    2 u   CHR     4,3           69162 /dev/tty3
login     4873 root    3u  unix 0xc311fa80          13487 socket
httpd     5314 root  cwd    DIR     3,3    4096       2 /
httpd     5314 root  rtd    DIR     3,3    4096       2 /
<<<<<<<< HTTP CHILDREN TRUNCATED  >>>>>>>>>>
```

```
httpd     5321 root   24w   REG     3,3       0  113886
/var/log/httpd/ssl_mutex.1888
bash      8698 root  cwd    DIR    22,0    8192  141654
/mnt/cdrom/statbins/linux2.2_x86
bash      8698 root  rtd    DIR     3,3    4096       2 /
bash      86 98 root  txt    REG     3,3  519964     807 /bin/bash
bash      8698 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
bash      8698 root  mem    REG     3,3   11832   96841
/lib/libtermcap.so.2.0.8
bash      8698 root  mem    REG     3,3   65997    96695 /lib/libdl -2.2.4.so
bash      8698 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
bash      8698 root  mem    REG     3,3  261460   96716 /lib/libnss_files   -
2.2.4.so
bash      8698 root  mem    REG     3,3  355236   96724
/lib/libnss_nisplus-2.2.4.so
bash      8698 root  mem    REG     3,3  436784   96700 /lib/libnsl   -
2.2.4.so
bash      8698 root  mem    REG     3,3  173408   16544
/usr/lib/locale/en_US/LC_CTYPE
bash      8698 root    0u   CHR     4,3           69162 /dev/tty3
bash      8698 root    1u   CHR     4,3           69162 /dev/tty3
bash      8698 root    2u   CHR     4,3           69162 /dev/tty3
bash      8698 root  255u   CHR     4,3           69162 /dev/tty3
bash      8761 root  cwd    DIR    22,0    8192  141654
/mnt/cdrom/statbins/linux2.2_x86
bash      8761 root  rtd    DIR     3,3    4096       2 /
bash      8761 root  txt    REG     3,3  519964     807 /bin/bash
bash      8761 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
bash      8761 root  mem    REG     3,3   11832   96841
/lib/libtermcap.so.2.0.8
bash      8761 root  mem    REG     3,3   65997   96695 /lib/libdl   -2.2.4.so
bash      8761 root  mem    REG     3,3 5772268     372 /lib/i686/libc   -
2.2.4.so
bash      8761 root  mem    REG     3,3  261460   967  16 /lib/libnss_files -
2.2.4.so
bash      8761 root  mem    REG     3,3  355236   96724
/lib/libnss_nisplus -2.2.4.so
bash      8761 root  mem    REG     3,3  436784   96700 /lib/libnsl   -
2.2.4.so
bash      8761 root  mem    REG     3,3  173408   16544
/usr/lib/locale/en_US/LC_CTYPE
bash      8761 root    0u   CHR     4,3           69162 /dev/tty3
bash      8761 root    1w  FIFO     0,0           13853 pipe
bash      8761 root    2u   CHR     4,3           69162 /dev/tty3
bash      8761 root  255r   REG     2,0     3944   4 /mnt/floppy/volatile -
cap.sh
nc        8762 root  cwd    DIR    22,0    8192  141654
/mnt/cdrom/statbins/linux2.2_x86
nc        8762 root  rtd    DIR     3,3    4096       2 /
nc        8762 root  txt    REG    22,0  262068  157696
/mnt/cdrom/statbins/linux2.2_x86/nc
nc        8762 root  mem    REG     3,3  261460   96716 /lib/libnss_files   -
2.2.4.so
nc        8762 root  mem    REG     3,3 1282588   96691 /lib/libc  -2.2.4.so
nc        8762 root  mem    REG     3,3  485171   96682 /lib/ld  -2.2.4.so
nc        8762 root  mem    REG     3,3  355236   96724
/lib/libnss_nisplus -2.2.4.so
nc        8762 root  mem    REG     3,3  436784   96700 /lib/libnsl   -
2.2.4.so
nc        8762 root    0r  FIFO     0,0           13853 pipe
```

```
nc        8762 root    1u   CHR       4,3           69162 /dev/tty3
nc        8762 root    2u   CHR      4,3           69162 /dev/tty3
nc        8762 root    3u  IPv4   13859            TCP darth:1045   -
>192.168.0.2:5055 (ESTABLISHED)
lsof      8832 root  cwd    DIR    22,0    8192  141654
/mnt/cdrom/statbins/linux2.2_x86
lsof      8832 root  rtd    DIR     3,3    4096       2 /
lsof      8832 root  txt    REG    22,0  349492  156928
/mnt/cdrom/statbins/linux2.2_x86/lsof
lsof      8832 root    0u   CHR     4,3           69162 /dev/tty3
lsof      8832 root    1w  FIFO     0,0           13853 pipe
lsof      8832 root    2u   CHR     4,3           69162 /dev/tty3
lsof      8832 root    3r   DIR     0,2       0       1 /proc
lsof      8832 root    4r   DIR     0,2       0 578813960 /proc/8832/fd
lsof      8832 root    5w  FIFO     0,0           16091 pipe
lsof      8832 root    6r  FIFO     0,0           16092 pipe
lsof      8833 root  cwd    DIR    22,0    8192  141654
/mnt/cdrom/statbins/linux2.2_x86
lsof      8833 root  rtd    DIR     3,3    4096       2 /
lsof      8833 root  txt    REG    22,0  349492  156928
/mnt/cdrom/statbins/linux2.2_x86/lsof
lsof      8833 root    4r  FIFO     0,0           16091 pipe
lsof      8833 root    7w  FIFO     0,0           16092 pipe
```

Configuration Check:

Checking process IDs reveals no rogue processes and no unusual looking files or devices associated with running processes. This can be suspect to the degree that some of the regular working files could be trojaned or a kernel module rootkit may have been able to partially mask output to lsof. But there's nothing here that points to backdoor processes running on the system when analysis was started. That still leaves plenty open for investigation, including whether any of the system's legitimate services and configurations were altered to allow access to an intruder.

Next stop, the /etc directory tree. A check of the following key files in the directory found no unusual entries or modifications: crontab, group, group -, gshadow, gshadow-, hosts.allow, hosts.deny, init tab, mtab, passwd, passwd -, shadow, shadow-, syslog.conf. The ftp-related configuration files (ftpaccess, ftpgroups, ftphosts, ftpusers) also had default entries. Burrowing into the configuration directories for cron functions, Apache (httpd.conf), the pam authentication modules, ssh and xinet.d revealed no tampering either. The rc.local and rc.sysinit scripts, in the rc.d directory, were also examined for non-standard entries.

Timeline:

Although all of the above files are prime candidates for alteration to open holes in the system, it's impossible to anticipate all the alternatives for subverting legitimate functions or introducing illegitimate ones. At this point, looking for clues about what WAS done to the system is best. An excellent starting point for that is to build a timeline of file activity from the Unix MACtimes of the files. Again, there's nothing stopping someone who has gained root privileges on a system from altering MACtimes, but doing so with leaving **any** clue (i.e. files that should have different modification times all having identical ones) isn't trivial. MAC stands for modification,

access and changed (the last refers to a change in inode information for the file, not the file itself).

TASK and Autopsy automate the generation of a time line, which is a two-step process. First, a data file is gleaned from the inode information of the target partition image, in this case the root partition. I specify the inclusion of data from unallocated files and metadata. Then, a text-based timeline file is generated from the data file, for a specified date range and optionally allowing the substitution of numeric file ownership data with the users and groups from the /etc/passwd and /etc/group files.

Looking at the timestamps of some marker files, such as /tmp/install.log, reveals when the operating system was installed – in the morning of March 10, in this case. Since we know the system was put on line around 10 pm on March 13, we can start the timeline there. In fact, the first data that shows up around that time is the access of the ping command, as network connectivity was being verified. That's followed by configuration of Sendmail, which shows the aliases database being rebuilt (note M,A and C all change) along with numerous other settings checked (i.e. /etc/mail/local-host-names) and the mail log updated.

```
Thu Mar 13 2003 21:57:57
23436 .a. -/-rwsr-xr-x 0   0   427      /bin/ping
Thu Mar 13 2003 22:04:05
12288 .a. -/-rw-r--r-- 0   0   113626   /etc/mail/access.db
Thu Mar 13 2003 22:04:23
0 .a. -/-rw-r--r-- 0   0   113617   /etc/mail/mailertable
Thu Mar 13 2003 22:04:25
12288 .a. -/-rw-r--r-- 0   0   113628   /etc/mail/mailertable.db
Thu Mar 13 2003 22:04:30
0 .a. -/-rw-r--r-- 0   0   113619   /etc/mail/statistics
Thu Mar 13 2003 22:04:40
345 .a. -/-rw-r--r-- 0   0   113613   /etc/mail/access
Thu Mar 13 2003 22:05:12
20 .a. -/-rw-r--r-- 0   0   83520    /etc/sysconfig/sendmail
30 .a. -/-rw-r--r-- 0   0   84299    /etc/sysconfig/network
16 .a. l/lrwxrwxrwx 0   0   98537    /usr/bin/newaliases   ->
../sbin/sendmail
Thu Mar 13 2003 22:05:13
120860 .a. -/-rwxr-xr-x 0   0   99312    /usr/bin/make
12288 mac -/-rw-r--r-- 0   0   70334    /etc/aliases.db
611 .a. -/-rw-r--r-- 0   0   113612   /etc/mail/Makefile
451076 .a. -/-r-sr-xr-x 0   0   98544    /usr/sbin /sendmail
1048 .a. -/-rw-r--r-- 0   0   70332    /etc/aliases
Thu Mar 13 2003 22:05:14
1830 .a. -/-rwxr-xr-x 0   0   83519    /etc/rc.d/init.d/sendmail
0 mac -/-rw-r--r-- 0   0   2877     /var/lock/subsys/sendmail
10368 .a. -/-rwxr-xr-x 0   0   98448    /u sr/lib/sasl/libplain.so.1.0.14
22 .a. l/lrwxrwxrwx 0   0   98415    /usr/lib/sasl/libanonymous.so    ->
libanonymous.so.1.0.15
64 .a. -/-rw-r--r-- 0   0   113616   /etc/mail/local -host-names
10778 .a. -/-rwxr-xr-x 0   0   98445    /usr/lib/sasl/liblogin.so.0.  0.5
13708 .a. -/-rwxr-xr-x 0   0   98439    /usr/lib/sasl/libcrammd5.so.1.0.15
20 .a. l/lrwxrwxrwx 0   0   98437    /usr/lib/sasl/libcrammd5.so    ->
libcrammd5.so.1.0.15
127 .a. -/-rw-r--r-- 0   0   113620   /etc/mail/trusted -users
22 .a. l/lrwxrwxrwx 0   0    98440    /usr/lib/sasl/libdigestmd5.so   ->
libdigestmd5.so.0.0.17
33 mac -/-rw-r--r-- 0   0   2878     /var/run/sendmail.pid
```

```
18 .a. l/lrwxrwxrwx 0   0   98446    /usr/lib/sasl/libplain.so    ->
libplain.so.1.0.14
8371 .a. -/-rwxr-xr-x 0   0   98417    /usr/l ib/sasl/libanonymous.so.1.0.15
46302 .a. -/-rw-r--r-- 0   0   70333    /etc/sendmail.cf
17 .a. l/lrwxrwxrwx 0   0   98443    /usr/lib/sasl/liblogin.so    ->
liblogin.so.0.0.5
33637 .a. -/-rwxr-xr-x 0   0   98442 /usr/lib/sasl/libdigestmd5.so.0.0.17
19 .a. -/-rw-r--r-- 0   0   98540    /usr/lib/sasl/Sendmail.conf
1599 m.c -/-rw------- 0   0   97380    /var/log/maillog.1
```

Where metadata such as the filename hasn't been preserved, the timeline points to the inode that's been freed, allowing the inode and the bloc k table it contains to be investigated. The format looks like this:

```
Thu Mar 13 2003 22:36:53
35 ma. -rw-r--r--  0    0   35343    <darth -hda3.img -dead-35343>
```

Wading Through Time:

When wading through a timeline, it's good to keep a couple of points in mi nd. Only that last time the MA or C element was changed will be recorded. That means subsequent access of or alterations to files on the system will knock them out of the timeline for an earlier event, even if they were accessed or changed at that earlier time, too. And, many Unix -based systems, such as the RedHat distribution on the honeypot, run periodic filesystem sweeps for dead files and to rebuild databases, such as the one used for the slocate utility. It's important to know the time of these automated runs (on RedHat the default is 4:02 am) to understand why the last access time for so many files is grouped around that time.

These runs will look like the following, with many files touched in the same second:

```
Sun Mar 16 2003 04:02:21
4096 .a. d/drwx r-xr-x  0   0   115802   /usr/lib/squid/errors/Japanese
4096 .a. d/drwxr -xr-x  0   0   115929   /usr/lib/squid/errors/Russian  -1251
4096 .a. d/drwxr -xr-x  0   0   80978    /usr/share/doc/bzip2  -1.0.1
4096 .a. d/drwxr -xr-x  0   0   80989    /usr/share/doc/ipr  oute-2.2.4
4096 .a. d/drwxr -xr-x  0   0   32865    /usr/share/doc/bash  -2.05/misc
4096 .a. d/drwxr -xr-x  0   0   80673    /usr/share/doc/cracklib  -2.7
4096 .a. d/drwxr -xr-x  0   0   80709    /usr/share/doc/parted  -1.4.16
4096 .a. d/drwxr -xr-x  0   0   32948     /usr/share/doc/logrotate -3.5.9
4096 .a. d/drwxr -xr-x  0   0   35949   /usr/lib/squid/errors/Russian  -koi8-r
4096 .a. d/drwxr -xr-x  0   0   80686    /usr/share/doc/db3  -3.2.9
4096 .a. d/drwxr -xr-x  0   0   16036    /usr/share
4096 .a. d/drwxr -xr-x  0   0    80689    /usr/share/doc/e2fsprogs  -1.23
4096 .a. d/drwxr -xr-x  0   0   116088   /usr/lib/squid/icons
4096 .a. d/drwxr -xr-x  0   0   113100   /usr/share/doc/findutils  -4.1.7
4096 .a. d/drwxr -xr-x  0   0   65193    /usr/share/doc/ed  -0.2
4096 .a. d/drwxr -xr-x  0   0   32950    /usr/share/doc/MAKEDEV  -3.2
4096 .a. d/drwxr -xr-x  0   0   32852    /usr/share/doc/newt  -0.50.33
4096 .a. d/drwxr -xr-x  0   0   19915 /usr/lib/squid/errors/Simplify_Chinese
4096 .a. d/drwxr -xr-x  0   0   32843  /usr/share/doc/shadow  -utils-20000902
4096 .a. d/drwxr -xr-x  0   0   65209    /usr/share/doc/at  -3.1.8
4096 .a. d/drwxr -xr-x  0   0   80650    /usr/share/doc/glibc  -2.2.4
4096 .a. d/drwxr -xr-x  0   0   115770   /usr/lib/squid/errors/German
4096 .a. d/drwxr -xr-x  0   0   115898   /usr/lib/ squid/errors/Romanian
4096 .a. d/drwxr -xr-x  0   0   35917    /usr/lib/squid/errors/Hungarian
4096 .a. d/drwxr -xr-x  0   0   113121   /usr/share/doc/gawk  -3.1.0
```

```
4096 .a. d/drwxr-xr-x  0   0   115866   /usr/lib/squid/errors/Portuguese
4096 .a. d/drwxr-xr-x  0   0   51932    /usr/lib/squid/errors/Estonian
4096 .a. d/drwxr-xr-x  0   0   64773  /usr/share/doc/bash -2.05/scripts.noah
4096 .a. d/drwxr-xr-x  0   0   16556    /usr/share/doc/db3 -3.2.9/images
4096 .a. d/drwxr-xr-x  0   0   80698    /usr/share/doc/glib -1.2.10
4096 .a. d/drwxr-xr-x  0   0   48608    /usr/share/doc/mingetty -0.9.4
4096 .a. d/drwxr-xr-x  0   0   16786    /usr/share/doc/bash -2.05/bashdb
4096 .a. d/drwxr-xr-x  0   0   64770    /usr/share/doc/bash -2.05/scripts
4096 .a. d/drwxr-xr-x  0   0   648 21  /usr/share/doc/iproute -
2.2.4/examples/diffserv
4096 .a. d/drwxr-xr-x  0   0   80164    /usr/share/doc
4096 .a. d/drwxr-xr-x  0   0   96197    /usr/sbin
4096 .a. d/drwxr-xr-x  0   0   65187    /usr/share/doc/cpio -2.4.2
4096 .a. d/drwxr-xr-x  0   0   521 5    /usr/libexec/awk
4096 .a. d/drwxr-xr-x  0   0   1995    /usr/lib/squid/errors/Swedish
4096 .a. d/drwxr-xr-x  0   0   8558    /usr/share/doc/HTML
```

In the following snippet, we can see traces of an attempt a little after midnight Friday local time (early Saturday morning) to log in via telnet. A new instance of *in.telnetd* binary, like other net services controlled by *xinetd*, is called by *xinetd* in response to a connection attempt. Our logs tell us the telnet session failed. And we see other traces a few minutes later of the anonymous FTP session in which the worthless /var/ftp/etc/passwd file was downloaded. After that transaction, we see the transfer log file dutifully updated (M and C times changed for it). Note that it's /var/log/xferlog.1 in the timel ine (with the appended '.1') thanks to the weekly *logrotate* that ran at 4:22 a.m. on the following day. The same is true of other logs.

```
Sat Mar 15 2003 00:41:21
37708 .a. -/-rwxr-xr-x  0   0   99473    /usr/sbin/in.telnetd
16 .a. l/lrwxrwxrwx  0   0   96 736   /lib/libutil.so.1  -> libutil -2.2.4.so
47872 .a. -/-rwxr-xr-x  0   0   96735    /lib/libutil -2.2.4.so
Sat Mar 15 2003 00:41:22
54 .a. -/-rw-r--r--  0   0   65295    /etc/issue.net
Sat Mar 15 2003 00:45:27
11504 .a. -/-rwxr-xr-x  0   0   113490   /lib /security/pam_listfile.so
6742 .a. -/-rwxr-xr-x  0   0   113501   /lib/security/pam_shells.so
363 .a. -/-rw-r--r--  0   0   114965   /etc/pam.d/ftp
Sat Mar 15 2003 00:53:44
79 .a. -/-r--r--r--  0   0   35735    /var/ftp/etc/passwd
Sat Mar 15 2003 00:53:45
324 m.c -/-rw-------  0   0   99552    /var/log/xferlog.1
```

At the time of the FTP login that generates the segmentation faults, we find some traces of that activity, namely the FTP users and hosts files are checked, probably by the FTP daemon processing th e logon attempt. Note that the /var/ftp directory file is modified along with its inode data. I'll tag this for further investigation a little later:

```
Sat Mar 15 2003 14:10:00
150 .a. -/-rw-------  0   0   70393    /etc/ftpusers
104 .a. -/-rw-------  0   0   70392    /etc/ftphosts
4096 mac -/-rw-r--r--  0   0   3084     /var/run/ftp.rips -all
Sat Mar 15 2003 14:11:12
4096 m.c d/drwxr-xr-x  0   0   35731    /var/ftp
```

The next traces on the timeline are of a honeypot supervisor logging in on the console and br inging up Midnight Commander.

```
Sat Mar 15 2003 21:17:23
110438 .a. -/-rwxr-xr-x  0   0   6746    /lib/libext2fs.so.2.4
16 .a. l/lrwxrwxrwx  0   0   6745    /lib/libext2fs.so.2   ->
libext2fs.so.2.4
649 .a. -/-rwxr-xr-x  0   0   9269    /usr/lib/mc/extfs/sfs.  ini
0 m.c -/-rw-r--r--  0   0   6007    /root/.cedit/cooledit.temp
6284 .a. -/-rwxr-xr-x  0   0   14520   /usr/lib/mc/bin/cons.saver
17 .a. l/lrwxrwxrwx  0   0   6741    /lib/libcom_err.so.2   ->
libcom_err.so.2.0
2785 .a. -/-rw-r--r--  0   0   5998    /root /.mc/ini
0 m.c -/-rw-r--r--  0   0   6006    /root/.cedit/cooledit.block
8456 .a. -/-rwxr-xr-x  0   0   6742    /lib/libcom_err.so.2.0
16 .a. l/lrwxrwxrwx  0   0   9579    /usr/lib/libgpm.so.1   ->
libgpm.so.1.18.0
526508 .a. -/-rwxr-xr-x  0   0   9255    /u sr/bin/mc
0 m.c -/-rw-r--r--  0   0   6005    /root/.cedit/cooledit.error
```

There's no indication of any other activity that left traces in the filesystem around that time, which isn't conclusive because MAC times can be tampered with and file access times may be overwritten by subsequent activity. Still, it does tend to suggest that if the WU-FTP segfaults coincided with a successful compromise, not much else was done on the system at the time. The more file activity, the harder it would be to ensure no trace of it remained in MAC times.

Tracking the timeline further, there's little of interest until what appears to be traces of another FTP connection at 10:21 p.m. the same day. There are a couple of things to note here. This is the last time the FTP daemo n file is accessed according to the timeline (*xinetd* would have started another *ftpd* process in response to a connection request). Also, it appears that this session didn't get very far. That's because we don't see the /etc/ftpusers and /etc/ftphosts files touched again (and if we did, we wouldn't have seen evidence of them being accessed in the previous FTP session), just the /etc/ftpaccess file. That file is referenced every time a FTP daemon process is started. The ftpusers and ftphosts files are checked during the FTP login process. There's no other evidence of filesystem activity around this time, so I tentatively mark this as another scan or banner-grab session.

```
Sat Mar 15 2003 22:21:47
464 .a. -/-rw-------  0   0   70390    /etc/ftpconversions
172668 .a. -/-rwxr-xr-x  1   1   99547    /usr/sbin/in.ftpd
1657 .a. -/-rw-------  0   0   65328    /etc/ftpaccess
90766 m.c -/-rw-------  0   0   97378    /var/log/messages.1
4096 mac -/-rw-r--r--  0   0   3083    /var/run/ftp.pids -all
Sat Mar 15 2003 22 :21:48
3280 m.c -/-rw-------  0   0   97379    /var/log/secure.1
```

The Witching Hour:

After this, nothing interesting appears in the timeline before 4:02 a.m. on Sunday, the witching hour that marks the commencement of the series of housekeeping scripts on RedHat Linux installations. This steps all over the filesystem MAC times, and because it's the end of week, automated weekly tasks – including the log file rotation – add to the mess 20 minutes later. On the plus side, the vast majority of files affected by the processes are simply accessed, not altered, meaning reviewing the

timeline for files that were modified or had inode data changed in that period not as daunting. That review yields nothing unusual.

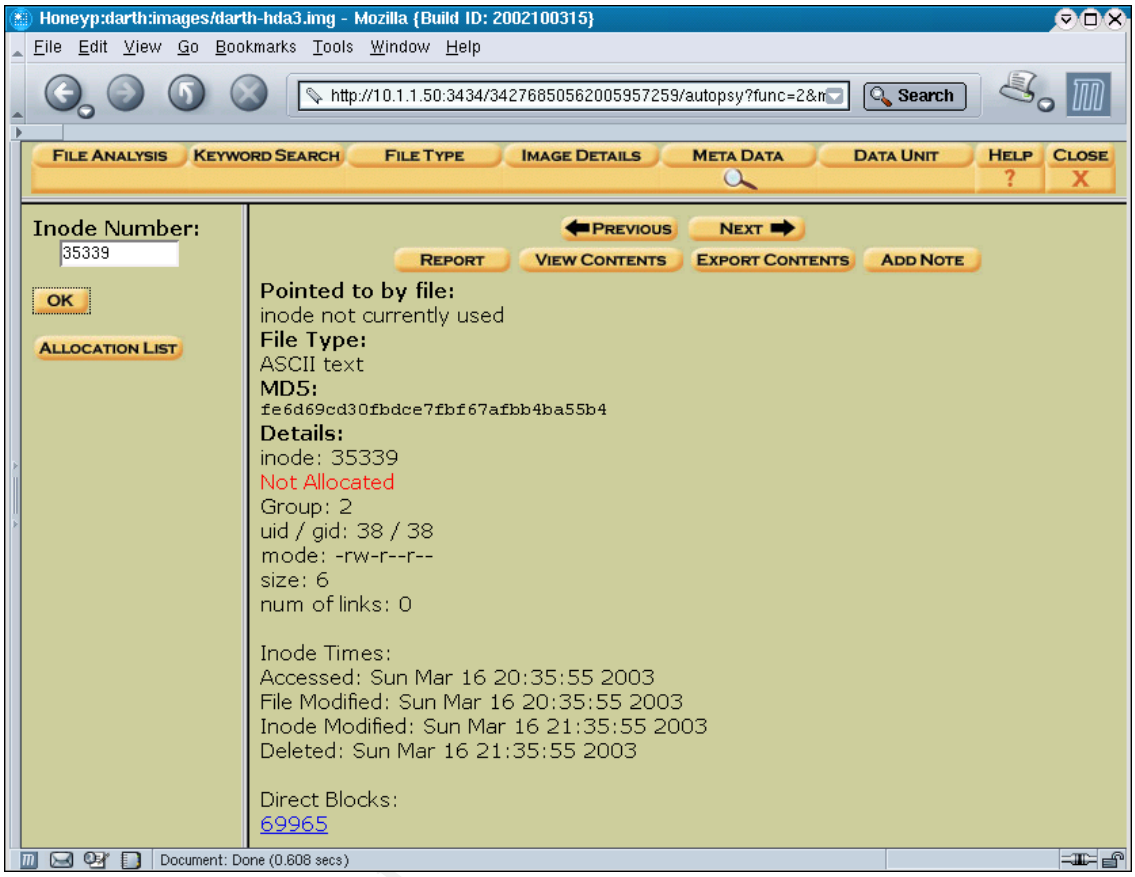
The next entry of interest is a trace that suggest s another connection attempt, a reading of the tcp-wrappers-style “hosts.allow” and “hosts.deny” files. It's now about 22 hours since the *ftpd* segfault session, so I wonder whether this – if it is the same person responsible for the earlier session – suggests someone whose Internet access hews to some schedule. However, no other MAC timestamps show up in the same period and no other log information points to this time, so it doesn't add much to the picture for now.

```
Sun Mar 16 2003 11:59:43
161 .a. -/-rw-r--r--  0   0   64622    /etc/hosts.allow
347 .a. -/-rw-r--r--  0   0   64623    /etc/hosts.deny
```

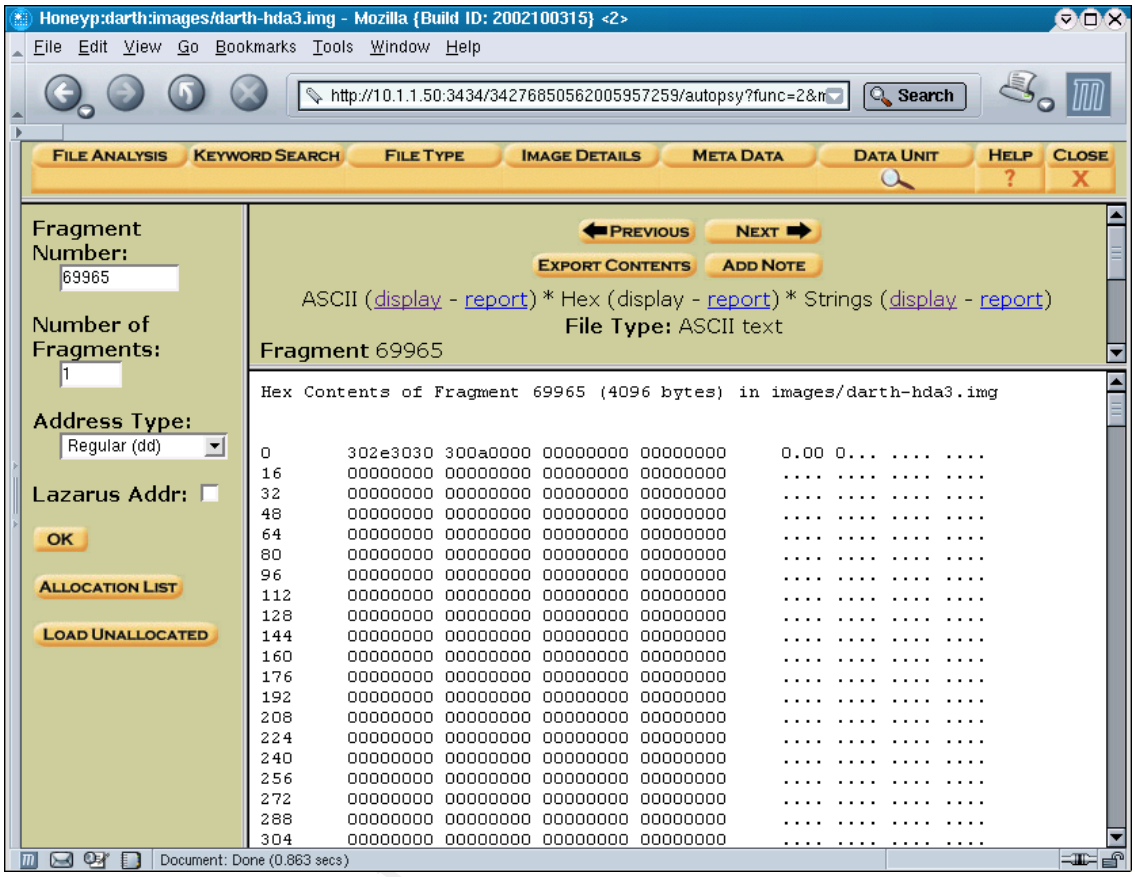
Next of possible interest is the modification of a tiny file around 8:35 p.m. The file was subsequently deleted, and its meta-data structure updated to reflect this. In general, the times associated with the file are saved in the structure until it is reallocated.

```
Sun Mar 16 2003 20:35:55
6 ma. -rw-r--r-- 38    38    35339    <darth-hda3.img-dead-35339>
```

The information from the timeline report can still be use d to find the block(s) on which the file sits, as long as its contents weren't overwritten, using TASK and Autopsy's ability to directly reference inode metadata. Entering 35339 into the inode search number, shows us the remaining meta data about the file:



One interesting note is the file was deleted exactly an hour after it appears as modified in the timeline. I follow the link to the associated file block and see it contains only the value 0.000, with the rest of the block still zeroed out from the p re-installation wiping:



Fast-forwarding through the timeline, confirms what this was ...

```
Sun Mar 16 2003 21:35:55
6 mac -/-rw-r--r-- 38  38   35340    /etc/ntp/drift
6 ..c -rw-r--r-- 38  38    35339    <darth -hda3.img-dead-35339>
```

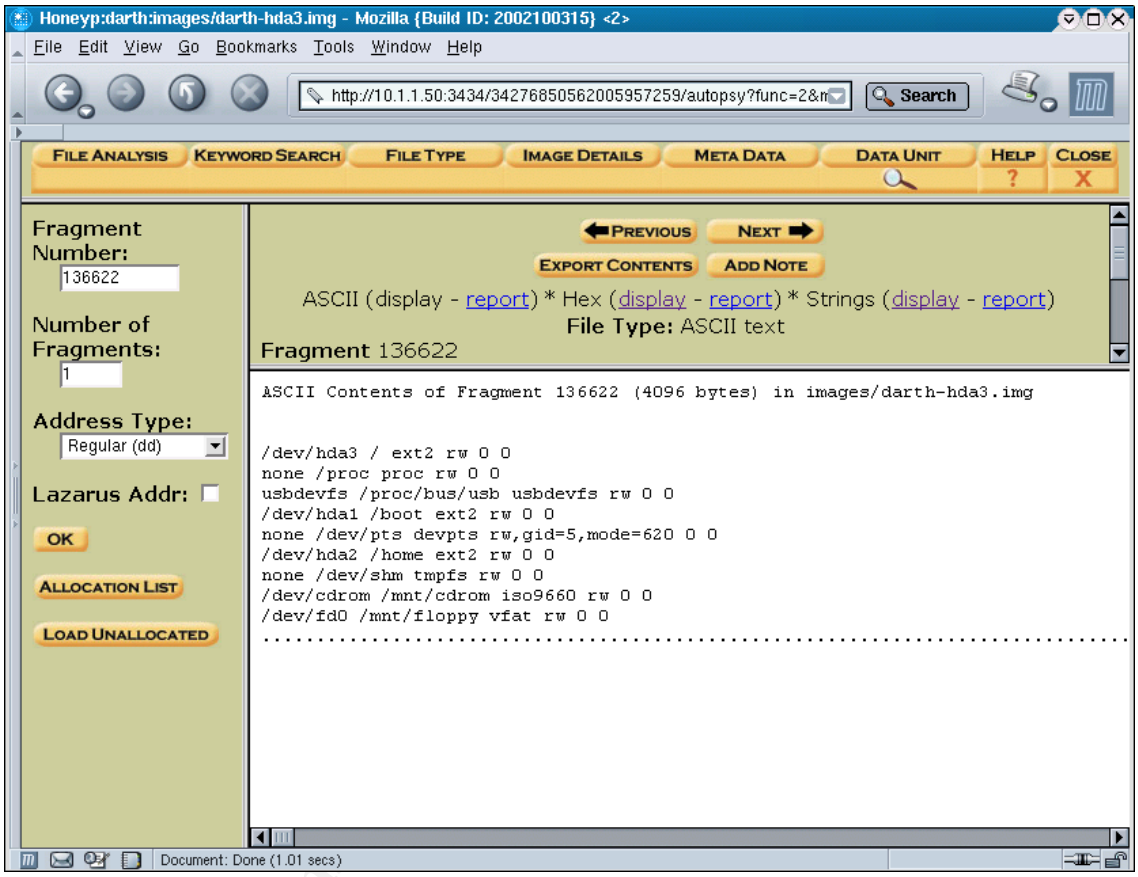
.... the latest entr y to the Network Time Protocol daemon's drift file, which keeps track of clock drift for the system and which was updated and saved, using the next inode.

There's no other interesting activity in the timeline until the final login to the system to grab volatile information prior to seizure. That bears a close review for signs whether any of the information -gathering activity triggered a response, perhaps by a malicious process watching for specific activity. None appears.

An entry in the final seconds of the timeline that shows a change to the /etc directory entry, on review, proves to be tied to the unmounting of the floppy drive right before the plug is pulled. That altered the /etc/mtab file and the information referencing it in the directory structure.

```
Sun Mar 16 2003 21:53:20
4096 m.c d/drwxr-xr-x   0     0     64131     /etc
28380 .a. -/-rwsr-xr-x  0     0     1247      /bin/umount
0 mac ----------        0     0     64135     <darth -hda3.img-dead-64135>
282 .ac -rw-r--r--      0     0     64133     <darth -hda3.img-dead-64133>
249 mac -/-rw-r--r--    0     0     70448     /etc/mtab
```

We can find the old mtab file on block 136622, pointed to by inode 64133:

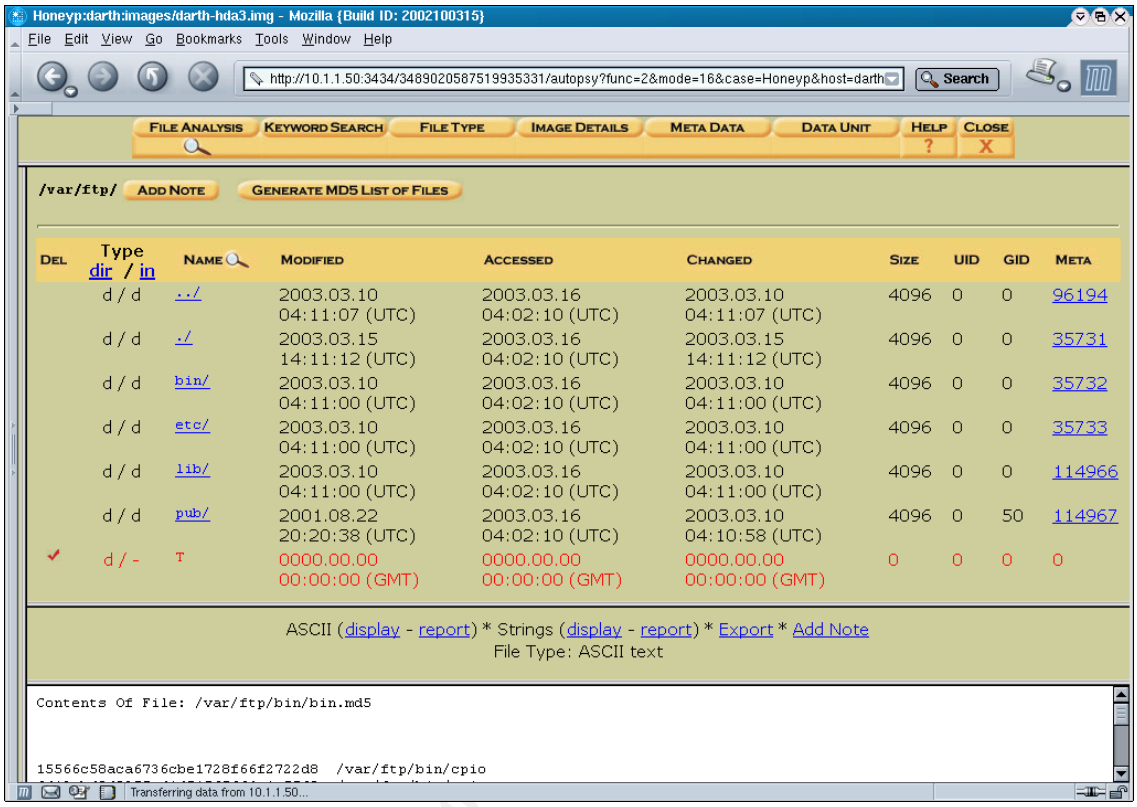


Pursuing Leads:

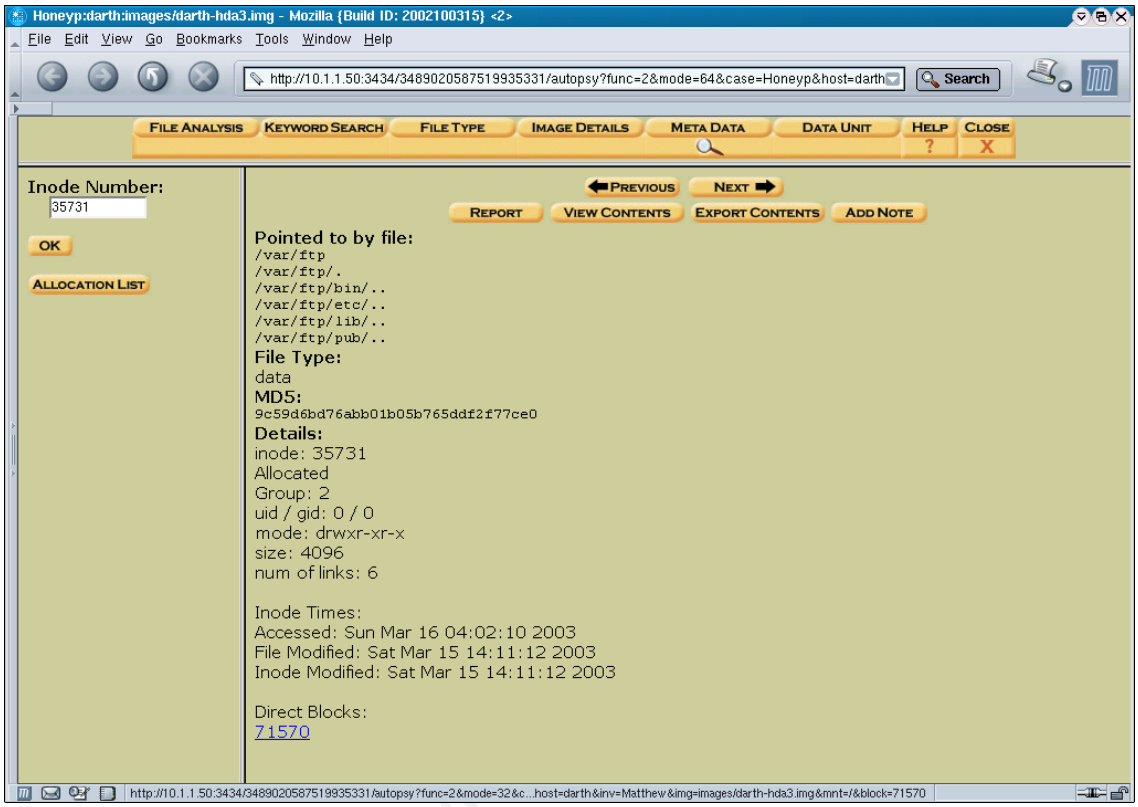
The timeline hasn't revealed any strong indications of things amiss on the filesystem, but it has left us with some areas to examine, especially when combined with information gleaned from the log files.

First, I browse to the /var/ftp directory on which the M_C times changed around the time of ftp access that led to a segfault. The changes sug gest the directory's contents or content information changed, which considering that it's the working directory for the ftp daemon isn't surprising. The Autopsy browser automatically shows information on deleted files (highlighted in red) when it's availab le, and provides links to data blocks and inodes referenced by the data, making it easy to pursue areas of interest.

The FTP directory shows a phantom directory called “T” with no other inode data including MAC times. It also shows the single -dot “.” special directory information has M_C time changes corresponding to the change in the parent directory and the segfault records.



Following the link to its metadata, shows entries that correspond to the directory structure and a reference to a data block, one of those holding the directory information:



The M_C time changes suggest an alteration to the content of the directory tree, but no corresponding file or subdirectory MAC time appears. The other file and directory contents of the /var/ftp director y tree appear normal. It's possible the change is a byproduct of the segfaults and daemon restarts, even a temporary file the metadata for which no longer survives.

Widening the Search:

At this stage, with no further strong leads to follow other than the suggestion of a possible overflow exploit run against WU -FTP, it makes sense to step back a little again and look in other likely areas for evidence of a compromise.

Autopsy's file analysis menu provides an easy way to find data on all deleted files on a partition, and using that to survey files deleted on the system revealed nothing interesting that hadn't already appeared on the timeline. Indeed, files deleted during the period under examination on the timeline would have appeared there. Similar searches of deleted files in the /boot and /home partitions reveals nothing interesting.

Next to check is the /root directory, where I search the bash and other shell (MC in this case) history files and look for other clues like a .ssh directory that would refle ct the configuration of the OpenSSH secure shell for the root user. The

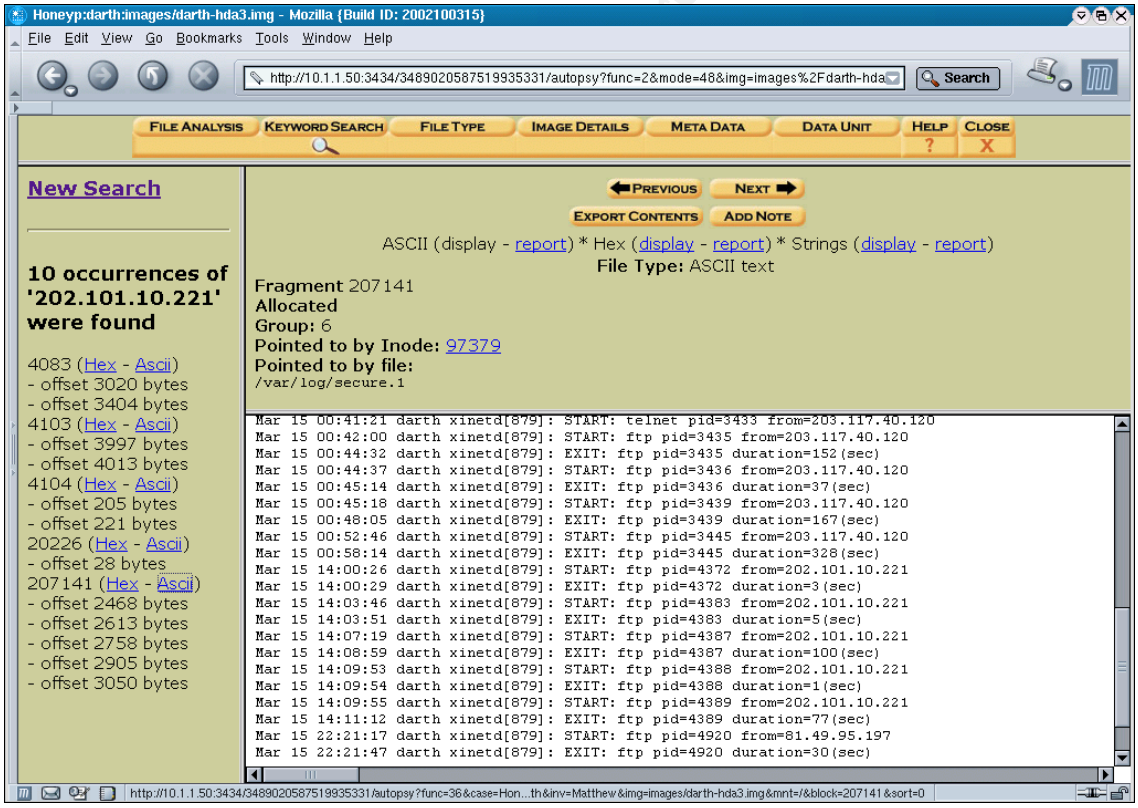
/root/.bash_history file reveals little other activity than recent console monitoring and earlier configuration work, along with a strong preference for the Midnight Commander.

```
mc
mc
mc
mc
umount /mnt/cdrom
mc
shutdown -r now
ping 192.168.0.1
./ntpd start
date
date
date
mc
mc
ping yahoo.com
mc
exit
ps
ping yahoo.com
ifconfig
mc
exit
ping yahoo.com
/etc/init.d/sendmail start
ps -ax
/etc/init.d/httpd restart
telnet 192.168.0.25 80
telnet 127.0.0.1 80
wget http://192.168.0.25/index.html
wget http://127.0.0.1/index.html
mc
mc
mc
/etc/init.d/httpd restart
wget http://127.0.0.1/index.html
wget http://192.168.0.25/index.html
mc
mc
mc
mc
mc
mc
mc
exit
ps -ax
mc
cd /home
ls
exit
ps -ax
netstat -tupan
netstat -tupan | less
netstat -tupan | less
mc
exit
```

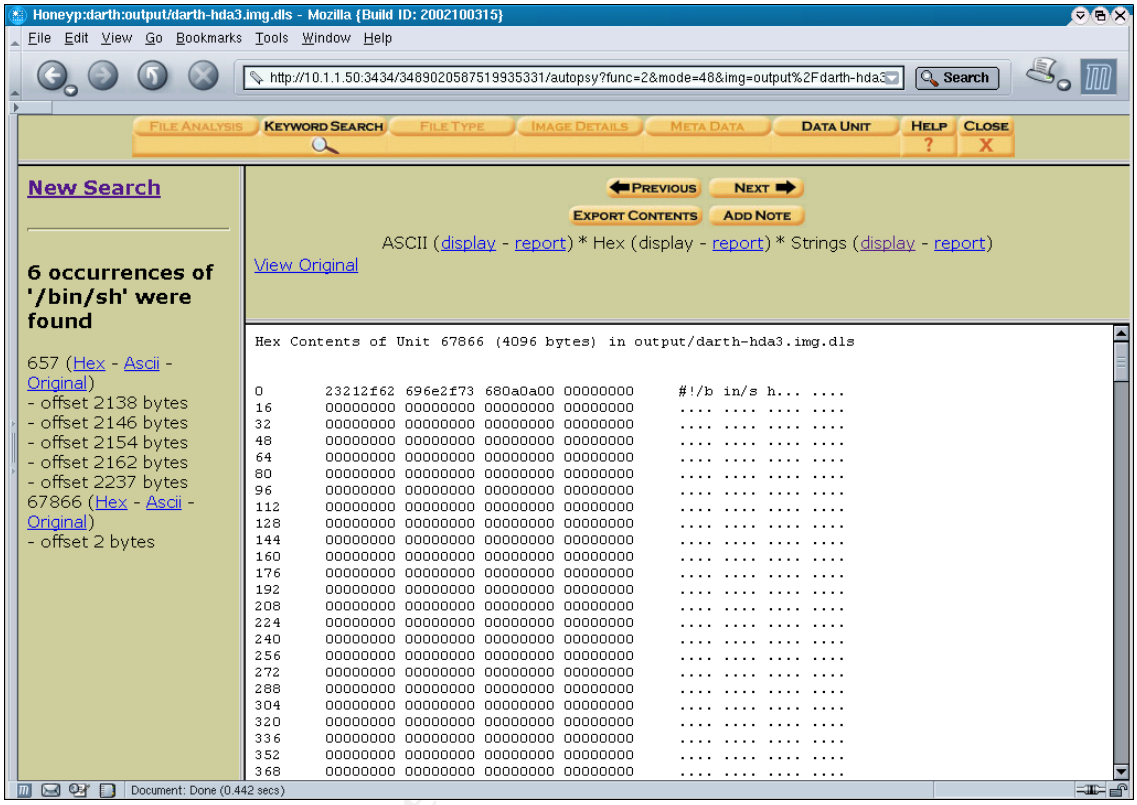
String Searches:
Autopsy and TASK also greatly speed up searching for key ASCII strings by generating string files both from the listed filesystem and from unallocated space on each partition, wh ich allows for much faster ad -hoc searches for ASCII content than parsing through the original data blocks each time. Each strings file, like pretty much all other TASK output, is MD5 hashed to ensure its validity. Autopsy provides an interface to check th e MD5 hashes on all output files.

Although I intend to run a lengthier, specialized string search later, I take advantage of the Autopsy interface to run a few quick text searches for things like the IP addresses from which connections were made to the ho neypot system. Often exploit packages will use hard -coded IP addresses. What's more, searching deleted space for suspect IP addresses is a good way to find snippets of purged log files.

The searches against (217.89.xx.xx, 61.8.228.xxx, 218.1.xx.xxx, 200.8 5.x.xx, 203.117.xx.xxx, 202.101.xx.xxx, 81.49.xx.xxx) turn up nothing in the unallocated data blocks. In the regular filesystem, they're found only in log files, and nowhere on the /boot and /home partitions.



Searching unallocated space for the string "/bin/sh", used to invoke the system shell, yields two data blocks containing it: one is part of a text file containing configuration instructions for portmapper, the other is a lone reference with the sh -bang (#!) that signals the start of a shell script , in an otherwise empty data block. Examining the 10 blocks on either side doesn't reveal any relationship or useful context.



I ran the *strings* ASCII-extraction utility against the exported data in the usable blocks of the var/log/lastlog file (images -darth-hda3.img-inode96670.raw), which contained text from a record of an FTP session, including the IP address associated with the segfault. It contained little else in ASCII, other than an entry for the last TTY login on tty3. I performed a more extensive keyword search using the strings files extracted from each partition by TASK and similar files created with *strings* from the /proc/core dump and from the swap partition.

Running *grep –f /home/matthew/wordlist* against *darth-core.str > word-results-core* and against the other string files allowed the use of a custom word list to accelerate the process. That list included searching for more instances of data already revealed (the login id of mozilla@ may help find instances of deleted log files), the originat ing IP address (again for any hard -coded access wrappers or core memory traces). It also includes key strings for hotmail and yahoo e -mail addresses, profane language often associated with root kits and other hacker scripts, and for keywords associated with rootkits, as identified by Chkrootkit.org

Rootkits are typically prepackaged and scripted tools that install trojaned binaries in the place of legitimate ones, subvert system processes (including logging), open backdoors and seek to hide all traces of t hemselves.

Chkrootkit.org's excellent *chkrootkit* utility for detecting the presence of known rootkits on systems makes use of string signatures and known filename searches for part of its evaluation process. For example, from the search for HKRK:

```
if ${egrep} "\.hk" /etc/rc.d/init.d/network 2>/dev/null ; then
echo "Warning: /etc/rc.d/init.d/network INFECTED"
```

Most of my custom word list's keywords related to rootkits were gleaned from the source for *chkrootkit*. Also included are keywords associat ed with popular IRC bots such as Eggdrop and Energymech, along with other tools like sniffers (i.e. Sniffit) and scanners (i.e. NMAP).

Wading through the output of the searches yielded nothing of interest – other than discovering that the developers of a lot of legitimate Linux programs use yahoo.com and hotmail addresses.

Working with Images:

To pursue further analysis and circumvent some of the limitations of Autopsy, we first mount the images as loopback filesystems so that we can use them like regula r block devices. The command *mount -ro,loop,nodev,noexec,noatime /forensics/image-file /forensics/Honeyp/darth/mnt/mountpoint* allows us to mount an image as a read-only one, with no alteration to its filesystem data, and protects us from accidentally execu ting a script or binary during our exam. This is repeated for the two other partition images.

Looking for hidden or obscured directories, I first run a search for all directories starting with an "." character, which hides the directory from normal ls out put.

find /forensics/Honeyp/darth/mnt/ -name "." -type d -ls* yields this:

```
64129    4 drwxrwxr-x   3 matthew  matthew  4096 Mar  9 21:22
/forensics/Honeyp/darth/mnt/home/sam/.mc
35995    4 drwxr-xr-x   3 root     root     4096 Mar 15 21:30
/forensics/Honeyp/darth/mnt/root/.mc
36000    4 drwx------   2 root     root     4096 Mar  9 20:50
/forensics/Honeyp/darth/mnt/root/.cedit
3080     4 drwx------   2 root     root     4096 Mar 12 21:18
/forensics/Honeyp/darth/mnt/root/.links
```

All of these directories hav e been reviewed and cleared, so let's move on to directories starting with a space (*find /forensics/Honeyp/darth/mnt/ -name " *" -type d -ls*) or a squiggle (*find /forensics/Honeyp/darth/mnt/ -name "~*" -type d -ls*). Neither search yields any directories.

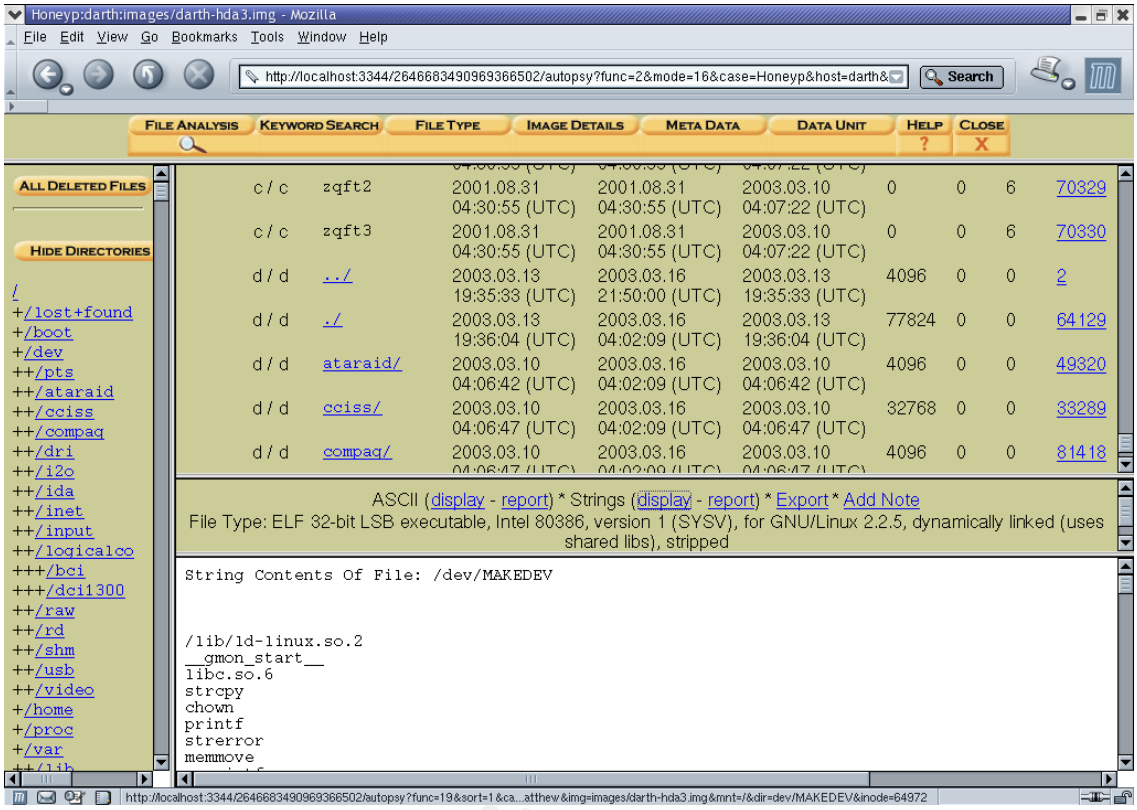
Another common hiding place is the /dev directory, where a teeming crowd of entries pointing to character, block and other system devices helps obscure other entities. Here the key is to look at file types and exclude those types that point to devices. *find /forensics/Honeyp/darth/mnt/dev -not -type c -not -type b -ls* gives us:

```
70447 0 srw-rw-rw-   1 root  root   0 Mar 13 19:35
/forensics/Honeyp/darth/mnt/dev/log
 64972   20  -rwxr-xr-x   1 root  root  16600 Aug 31  2001
/forensics/Honeyp/darth/mnt/dev/MAK EDEV
 65374 0 lrwxrwxrwx   1 root  root  13 Mar 10 04:06
/forensics/Honeyp/darth/mnt/dev/core  -> ../proc/kcore
 65411 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:06
/forensics/Honeyp/darth/mnt/dev/fb  -> fb0
 65444 0 lrwxrwxrwx   1 root  root  15 Mar 10 04:06
/forensics/Honeyp/darth/mnt/dev/fd  -> ../proc/self/fd
 65721 0 lrwxrwxrwx   1 root  root   4 Mar 10 04:06
/forensics/Honeyp/darth/mnt/dev/ftape  -> qft0
 17259 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/inet/arp  -> udp
 17268 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/inet/rip  -> udp
 66073 0 lrwxrwxrwx   1 root  root   7 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/ip  -> inet/ip
 66083 0 lrwxrwxrwx   1 root  root   9 Mar 10 04:07
/forensic s/Honeyp/darth/mnt/dev/ipip  -> inet/ipip
 66178 0 lrwxrwxrwx   1 root  root   9 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/js0  -> input/js0
 66158 0 lrwxrwxrwx   1 root  root   9 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/isdnctrl  -> isdnctrl0
 66179 0 lrwxrwxrwx   1 root  root   9 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/js1  -> input/js1
 66180 0 lrwxrwxrwx   1 root  root   9 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/js2  -> input/js2
 66181 0 lrwxrwxrwx   1 root  root   9 Mar 10 04:07
/forensics /Honeyp/darth/mnt/dev/js3  -> input/js3
 66817 0 lrwxrwxrwx   1 root  root  6 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/radio  -> radio0
 66822 0 lrwxrwxrwx   1 root  root   4 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/ram  -> ram1
 66843 0 lrwxrwxrwx    1 root  root   4 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/ramdisk  -> ram0
 66846 0 lrwxrwxrwx   1 root  root  10 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/rawip  -> inet/rawip
 66866 0 lrwxrwxrwx   1 root  root   6 Mar 10 04:07
/forensics/Honeyp/dar th/mnt/dev/sbpcd  -> sbpcd0
 68957 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sga  -> sg0
 68958 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgb  -> sg1
 68959 0 lrwxrwxrwx   1 root  root   3  Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgc  -> sg2
 68960 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgd  -> sg3
 68961 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sge  -> sg4
 68962 0 l rwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgf  -> sg5
 68963 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgg  -> sg6
 68964 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/ mnt/dev/sgh  -> sg7
 68965 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgi  -> sg8
 68966 0 lrwxrwxrwx   1 root  root   3 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgj  -> sg9
```

```
 68967 0 lrwxrwxrwx   1 root  root   4 Mar 10 0  4:07
/forensics/Honeyp/darth/mnt/dev/sgk   -> sg10
 68968 0 lrwxrwxrwx   1 root  root   4 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgl   -> sg11
 68969 0 lrwxrwxrwx   1 root  root   4 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgm   -> sg12
 68970 0 lrwxrw xrwx   1 root  root   4 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgn   -> sg13
 68971 0 lrwxrwxrwx   1 root  root   4 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/sgo   -> sg14
 68972 0 lrwxrwxrwx   1 root  root   4 Mar 10 04:07
/forensics/Honeyp/darth/mnt /dev/sgp -> sg15
 70044 0 lrwxrwxrwx   1 root  root   4 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/vbi   -> vbi0
 69123 0 lrwxrwxrwx   1 root  root  17 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/stderr   -> ../proc/self/fd/2
 69124 0 lrwxrwxrwx   1 root  r oot  17 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/stdin   -> ../proc/self/fd/0
 69125 0 lrwxrwxrwx   1 root  root  17 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/stdout   -> ../proc/self/fd/1
 69127 0 lrwxrwxrwx   1 root  root   4 Mar 10 04:07
/forensics/H oneyp/darth/mnt/dev/systty  -> tty0
 70192 0 lrwxrwxrwx   1 root  root   9 Mar 10 04:07
/forensics/Honeyp/darth/mnt/dev/winradio   -> winradio0
 70430 0 lrwxrwxrwx   1 root  root   5 Mar 10 04:18
/forensics/Honeyp/darth/mnt/dev/mouse   -> psaux
 70337 0 prw-------   1 root  root   0 Mar 12 23:23
/forensics/Honeyp/darth/mnt/dev/initctl
 70446 0 lrwxrwxrwx   1 root  root   8 Mar  9 20:20
/forensics/Honeyp/darth/mnt/dev/cdrom   -> /dev/hdc
 64132 0 srwx------   1 root  root  0 Mar 13 19:36
/forensics/Honeyp/darth/mn t/dev/gpmctl
```

Most of the output listed are links to other device names (and a couple of /proc entries), which can easily be reviewed and eliminated. There are two type "s" devices, serial devices designated differently from other character devices (both O K) and the MAKEDEV file, an executable used in the creation of the /dev directory structure.

Another way to do this is via the Autopsy browser, which allows sorting /dev entries by type by clicking on either link at the top of the file type column. Here's an example:



File Hunt:

Next comes a search for binaries and other executables which have the SUID bit set. These can be dangerous because they can allow an ordinary user to elevate his privileges on the system to those of the SUID program, usually ro ot. A user who has been able to designate a shell as SUID root would have full system control, and it's this sort of privilege-elevation backdoor we're looking for here.

find /forensics/Honeyp/darth/mnt/ -perm -004000 -type f -ls gives us:

```
96809  772 -rws--x--x   2 root  root    785372 Aug 10  2001
/forensics/Honeyp/darth/mnt/usr/bin/suidperl
 96809  772 -rws--x--x   2 root  root    785372 Aug 10  2001
/forensics/Honeyp/darth/mnt/usr/bin/sperl5.6.0
 96816   36 -rwsr-xr-x   1 root  root  34476 Aug 28  2001
/forensics/Honeyp/darth/mnt/usr/bin/chage
 96818   36 -rwsr-xr-x   1 root  root  36208 Aug 28  2001
/forensics/Honeyp/darth/mnt/usr/bin/gpasswd
 97287   40 -rwsr-xr-x   1 root  root  37580 Aug  3  2001
/forensics/Honeyp/darth/mnt/usr/bin/at
 98456   16 -r-s--x--x   1 root  root  13476 Aug  7  2001
/forensics/Honeyp/darth/mnt/usr/bin/passwd
 98558   16 -rws--x--x   1 root  root  13136 Aug 27  2001
/forensics/Honeyp/darth/mnt/usr/bin/chfn
 98559   16 -rws--x--x   1 root  root  12484 Aug 27  2001
/forensics/H oneyp/darth/mnt/usr/bin/chsh
```

```
 98577 8 -rws--x--x   1 root  root   5456 Aug 27  2001
/forensics/Honeyp/darth/mnt/usr/bin/newgrp
 99168   24 -rwsr-xr-x   1 root  root  21280 Jun 25  2001
/forensics/Honeyp/darth/mnt/usr/bin/crontab
 99353  212 -rwsr-xr-x   1 root  root    209948 Sep  6  2001
/forensics/Honeyp/darth/mnt/usr/bin/ssh
 99364   16 -rwsr-xr-x   1 root  root  14588 Jul 24  2001
/forensics/Honeyp/darth/mnt/usr/bin/rcp
 99366   12 -rwsr-xr-x   1 root  root  10940 Jul 24  2001
/forensics/Honeyp/darth/mn t/usr/bin/rlogin
 99367 8 -rwsr-xr-x   1 root  root   7932 Jul 24  2001
/forensics/Honeyp/darth/mnt/usr/bin/rsh
 99500   32 -r-sr-x---   1 root  news  29116 Jul 25  2001
/forensics/Honeyp/darth/mnt/usr/bin/inndstart
 99526   60 -r-sr-x---   1 uucp  news  5 3817 Jul 25  2001
/forensics/Honeyp/darth/mnt/usr/bin/rnews
 99539   28 -r-sr-x---   1 root  news  25436 Jul 25  2001
/forensics/Honeyp/darth/mnt/usr/bin/startinnfeed
 96759   20 -rwsr-xr-x   1 root  root  18444 Aug 28  2001
/forensics/Honeyp/darth/mnt/usr /sbin/ping6
 96763   12 -rwsr-xr-x   1 root  root   9804 Aug 28  2001
/forensics/Honeyp/darth/mnt/usr/sbin/traceroute6
 98544  448 -r-sr-xr-x   1 root  root    451076 Aug 31  2001
/forensics/Honeyp/darth/mnt/usr/sbin/sendmail
 98627 8 -rwsr-xr-x   1 root  root   6340 Sep  9  2001
/forensics/Honeyp/darth/mnt/usr/sbin/usernetctl
 99379   20 -rwsr-xr-x   1 root  root  20120 Jun 26  2001
/forensics/Honeyp/darth/mnt/usr/sbin/traceroute
 99565   12 -r-s--x---   1 root  apache   11244 Sep  6  2001
/forensics/Honey p/darth/mnt/usr/sbin/suexec
   427   24 -rwsr-xr-x   1 root  root  23436 Aug 28  2001
/forensics/Honeyp/darth/mnt/bin/ping
  1246   64 -rwsr-xr-x   1 root  root  57628 Jul 25  2001
/forensics/Honeyp/darth/mnt/bin/mount
  1247   28 -rwsr-xr-x   1 root  root   28380 Jul 25  2001
/forensics/Honeyp/darth/mnt/bin/umount
  1315   20 -rwsr-xr-x   1 root  root  18452 Jul 24  2001
/forensics/Honeyp/darth/mnt/bin/su
  1285   16 -r-sr-xr-x   1 root  root  15088 Sep 25  2001
/forensics/Honeyp/darth/mnt/sbin/pwdb_chkpwd
  1286   16 -r-sr-xr-x   1 root  root  15672 Sep 25  2001
/forensics/Honeyp/darth/mnt/sbin/unix_chkpwd
```

A search (*find /forensics/Honeyp/darth/mnt/ -perm -002000 -type f -ls*) for set GID files returns:

```
97325   16 -rwxr-sr-x   1 root  mail  12500 Jun 30  20 01
/forensics/Honeyp/darth/mnt/usr/bin/lockfile
 97376   28 -rwxr-sr-x   1 root  slocate  25020 Jun 25  2001
/forensics/Honeyp/darth/mnt/usr/bin/slocate
 98550 8 -r-xr-sr-x   1 root  tty    6444 Aug 29  2001
/forensics/Honeyp/darth/mnt/usr/bin/wall
 98588   12 -rwxr-sr-x   1 root  tty    8744 Aug 27  2001
/forensics/Honeyp/darth/mnt/usr/bin/write
 97546 8 -rwxr-sr-x   1 root  utmp   6604 Jun 25  2001
/forensics/Honeyp/darth/mnt/usr/sbin/utempter
  1482 8 -rwxr-sr-x   1 root  root   4120 Sep  9  2001
/forens ics/Honeyp/darth/mnt/sbin/netreport
```

All of these actually check out as SUID or SGID programs for this RedHat distribution, as it was installed, and the file modification times aren't recent.

That leads us to a check of any binaries with modification time s more recent that the system's install date, as referenced by /tmp/install.log with an Mtime of 4:17 a.m. on March 10. Although the timeline output examined earlier would have contained any files with Mtimes more recent than when the system was brought on line, we can see if any binaries claim to be more recent than its installation with: *find /forensics/Honeyp/darth/mnt/ -newer /forensics/Honeyp/darth/mnt/tmp/install.log -type f -user root -perm +111 -ls*. This search produces no results.

Calling on RPM:

While MACtime information is something a cracker may well bother to alter and can employ some scripted tools to help, another in-built method of file verification is harder to tamper with – the RedHat Package Manager database. This database is updated whenever files are installed from rpm packages with information about the file size, ownership, permissions and an MD5 hash value for each file. Altering database values for individual files isn't as readily done as changing MACtimes.

Making a review of the file database on the compromised system easier, RPM allows a filesystem's relative root to be set via command line, and the database it polls is relative to that, i.e. the one on our loopback mounted honeypot filesystem. Issuing the command *rpm -Va --root /forensics/Honeyp/darth/mnt/* delivers the following output:

```
.M......   /proc
S.5....T c /etc/crontab
S.5....T c /etc/syslog.conf
.M....G.   /dev/tty1
.M....G.   /dev/tty2
.M....G.   /dev/tty4
.M....G.   /dev/tty5
.M....G.   /dev/tty6
S.5....T c /etc/pam. d/system-auth
S.5....T c /etc/openldap/ldap.conf
S.5....T c /etc/krb.conf
S.5....T c /etc/ldap.conf
S.5....T c /etc/ssh/sshd_config
S.5....T c /etc/xinetd.d/talk
.M......   /usr/bin/filter
S.5....T c /etc/xinetd.d/wu -ftpd
S.5....T c /etc/php.ini
..5....T c /etc/mime.types
.......T c /etc/krb5.conf
S.5....T c /etc/sysconfig/pcmcia
S.5....T c /etc/ntp.conf
.....UG.   /var/run/radvd
S.5....T c /etc/xinetd.d/finger
S.5....T c /etc/xinetd.d/rlogin
S.5....T c /etc/xinetd.d/rsh
S.5....T c /etc/xinetd.d/telnet
..5....T c /etc/httpd/conf/httpd.conf
.....UG.   /var/log/squid
.....UG.   /var/spool/squid

The key to the results:
```

```
S file Size differs
M Mode differs (includes permissions and file type)
5 MD5 sum differs
D Device major/minor number mis -match
L readLink(2) path mis -match
U User ownership differs
G Group ownership differs
T mTime differs
```

The output shows a number of files have been altered since installation via *rpm*. Most are configuration files that were altered during setup, but the list is short enough to make double-checking the config files smart. The only binary altered is */usr/bin/filter*, which is a mail and news spam filtering utility. In the RedHat 7.2 distribution, this comes with the "cleanfeed" package for the news server installation. The file hasn't changed, only its permissions, which suggests this isn't an important issue. A little investigation shows that some versions of */usr/bin/filter* were the target of an exploit that leveraged the fact the file used SGID permissions – perhaps a post-installation script tightened those permissions. Certainly, the file doesn't show up in our list of SUID/SGID files.

Otherwise, there's nothing here to suggest binaries or other important files have been trojaned or replaced. As a final check, I browse in Autopsy to the main binary directories /bin, /sbin, /usr/bin, and /usr/sbin and look at the inode numbers associated with the installed executables. Within each directory, the inodes all are bunched in a relatively tight series, as one would expect from files installed at the same time. Although it wouldn't be impossible to slot a modified binary into the same or nearby inodes, most intruders won't go to the trouble, and an outlier among the inode numbers would be another tell-tale.

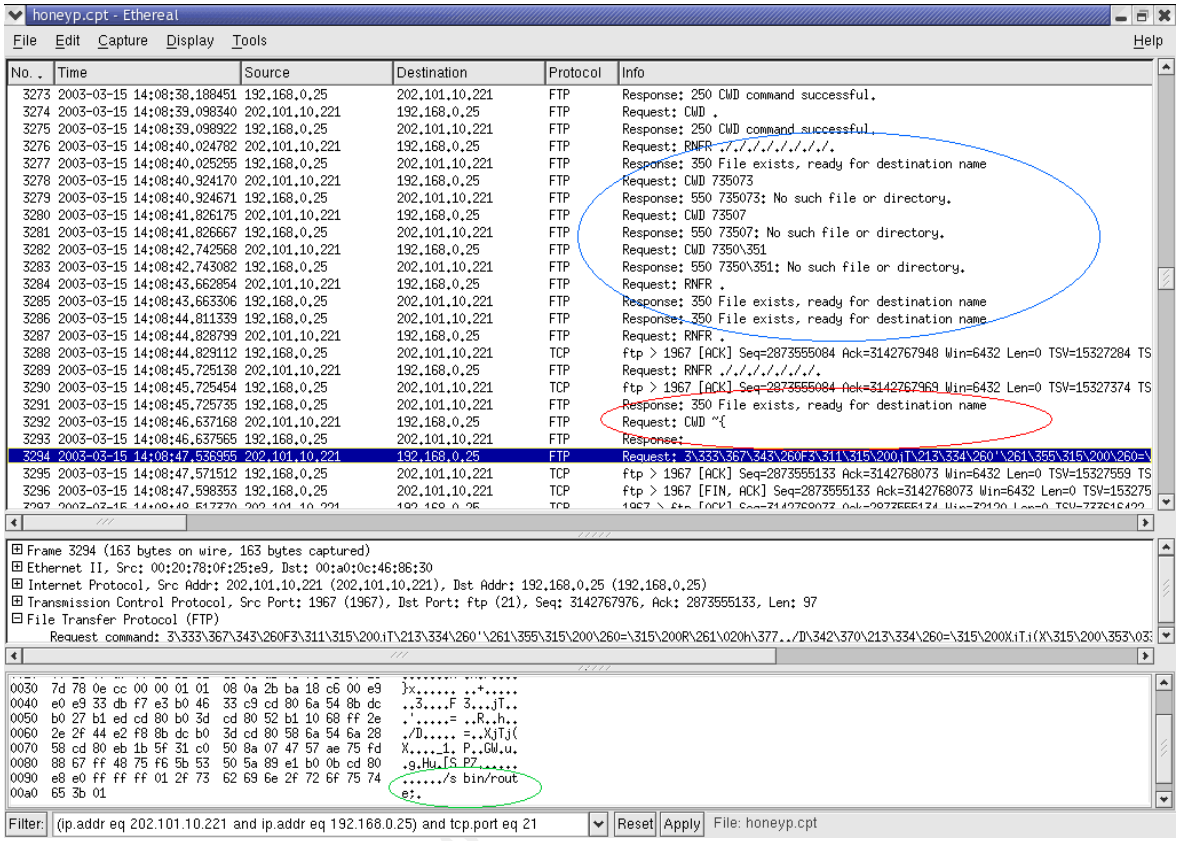
Lastly, I boot from the FIRE CDROM on the honeypot one more time and dd the unpartitioned space in the extended partition /dev/hda4, skipping past the blocks allocated to the /dev/hda5 swap partition with the skip= option, across via netcat to an image file on the forensic system. I check that the remaining space is still zeroed out using *cmp /dev/zero darth-hda4.img*, which reports any differences or just terminates with an EOF message. Nothing is hidden there.

Conclusion:

It's frequently more difficult to demonstrate that a system isn't compromised than it is to find evidence of one. Still, all of the analysis in this case suggests attempts to exploit vulnerable services on the honeypot system didn't lead to a working compromise. There's not much evidence of why not – although we know why the potential attacker who tried to download the system passwd file would have failed. Certainly, the WU-FTP version installed was vulnerable to remote exploits, as were a number of the other network services as configured, including OpenSSH and Apache, according to the CERT database.

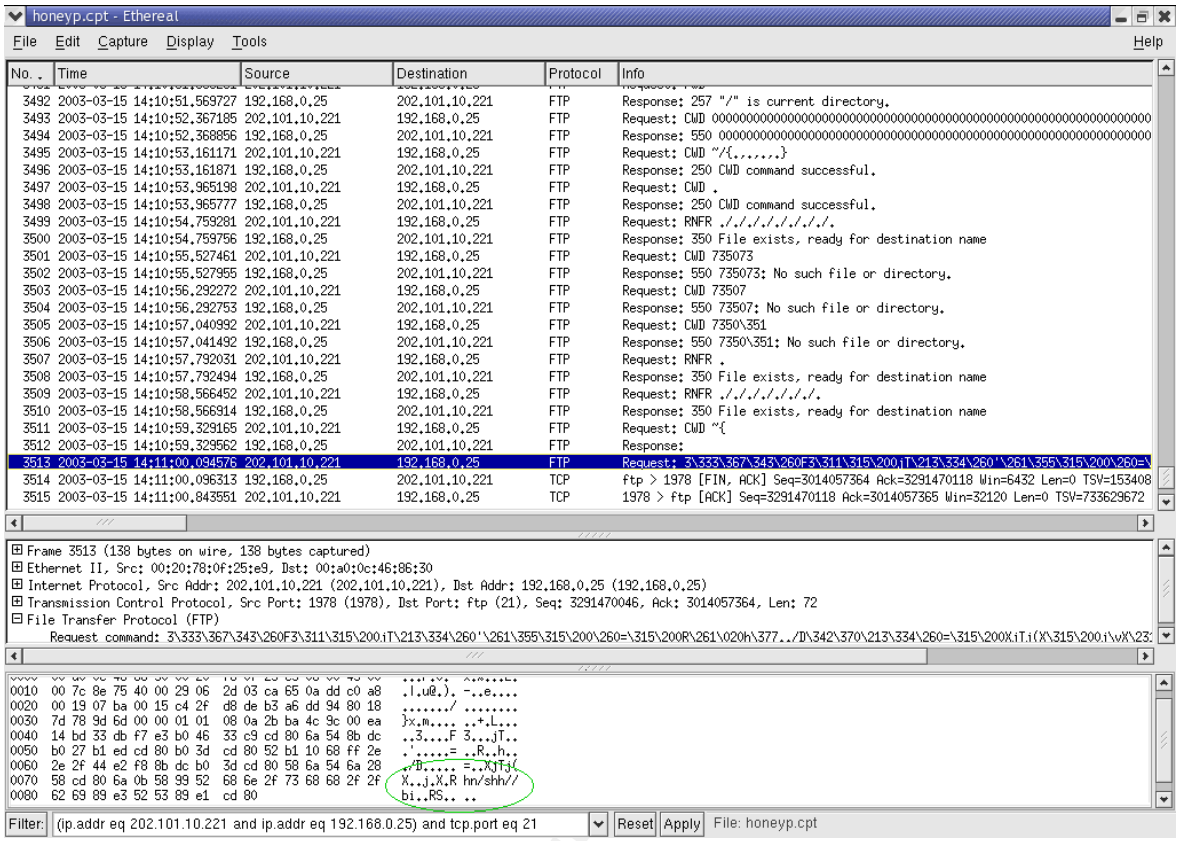
For a definitive view of what happened during the honeypot's relatively short life on the Internet, we need to turn to the packet capture on the firewalled system. Using Ethereal to analyze and reassemble traffic, we see a couple of half-hearted password-guessing attempts (i.e. root / root) on Telnet, shortly prior to the FTP

session that transferred the hollow /var/ftp/etc/passwd files. There's a number of other presumed scans, FTP serv ice requests that were never pursued. Then, there's the first FTP connection that results in a segfault:



From the speed of the interchange, it's pretty clear this is a scripted attack. Prior to this screen, the RNFR (rename -from) command is heavily rep eated (without it's expected companion rename -to), and in the blue circle on this screen you can see the digits 7350 repeated with the CWD command. It turns out this is a signature of 7350wurm, a scripted attack on various WU -FTP versions up to 2.6.1. The numerals also stand, in hacker scrip, for Teso, linking the exploit to a group known as Team Teso, authors of a number of exploits. This particular attack is run against a globbing buffer overflow hole (red circle) that exists in this WU -FTP version.

The copy of 7350wurm source code (also claimed by Teso) I found at the Packetstorm Security archives closely parallels the action in this session – except at the end, with the packet highlighted in blue and detailed in the two lower windows. This is the shellc ode payload packet, but this varies from the 7350wurm code in the archive in that instead of calling a shell it appears to try to execute the /sbin/route command (circled in green). Without any arguments, this would display the target's routing table. The attempt fails, and another attack is launched:



This time the shellcode comes straight from the 7350wurm script, culminating with an attempt to launch a shell. However, this also fails, as the following session transcript (honeypot FTP server is in blue) shows in greater detail:

```
220 darth FTP server (Version wu -2.6.1-18) ready.
USER ftp
331 Guest login ok, send your complete e  -mail address as password.
PASS mozilla@
230 Guest login ok, access restrictions apply.
RNFR ././
350 File exists, ready for dest ination name
RNFR ././
350 File exists, ready for destination name
RNFR ././
<<<<< TRUNCATED FOR BREVITY >>>>>>>
RNFR ././
350 File exists, ready for destination name
PWD
257 "/" is current directory.
CWD 0000000000000000000000000000000000000000000000000000  00000<SNIP>
500 0000000000000000 <SNIP> ë_èíÿÿÿ: File name too long.
<<<<< OVERFLOW TRUNCATED >>>>>>>
CWD ~/{.,.,.,.}
250 CWD command successful.
CWD .
250 CWD command successful.
RNFR ././././././././.
350 File exists, ready for destination name
CWD 73507 3
```

```
550 735073: No such file or directory.
CWD 73507
550 73507: No such file or directory.
CWD 7350é
550 7350é: No such file or directory.
RNFR .
350 File exists, ready for destination name
RNFR ./././././././.
350 File exists, ready for destination name
CWD ~{
sP
3Û÷ã°F3ÉÍ€jT‹Ü°'±íÍ€°=Í€R±_h ÿ../Dâø‹Ü°=Í €XjTj(XÍ€j
X™Rhn/shh//bi‰ãRS‰áÍ €unset HISTFILE;id;uname  -a;
```

And here is the shellcode at the end of the session, which attempts to open a shell as root, stop recording the command history (with unset HISTFILE), veri fy user and group (with id) and obtain basic information about the operating system (uname –a).

The overflow seems to work. After the malformed globbing instruction (~{) is sent, the FTP server responds by returning the exploit's marker: sP (note this t ranslates into hexadecimal notation as 0x7350). That tells the scripted attack to push through the appropriate shellcode. Still, it doesn't execute. The source code, where the authors leave a To-Do entry, offers a clue to the reason:

```
TODO: fix chroot brea k on 2.4.x series (somewhere between 2.4.6 and
 *       2.4.13 they changed chroot behaviour.
```

It seems possible the kernel upgrade used to accommodate the PCMCIA Ethernet card may have thrown the shellcode out in this case. In any case, the attacker – who had been connecting from a system out of Shanghai, China, according to a whois search against the IP address – doesn't appear to return in the next 31 hours to refine the exploit. Subsequent traffic to the honeypot system includes a number of service scans but no fresh attempts to crack the box. There's no evidence of a working compromise.

References

CERT Advisories for remote exploits on:
Apache 1.3.20
http://www.cert.org/advisories/CA -2002-17.html
OpenSSH 2.9p -2
http://www.cert.org/advisories/CA -2001-35.html
WU-FTP 2.6.1
http://www.cert.org/advisories/CA -2001-33.html
Sendmail 8.11.6
http://www.cert.org/advisories/CA -2003-07.html

FIRE, the Forensic and Incident Response Environment
http://fire.dmzs.com /?section=main

Chkrootkit.org, home of chkrootkit
http://www.chkrootkit.org/

7350wurm in the Packetstorm Security archives
http://packetst ormsecurity.nl/0205 -exploits/7350wurm.c

Autopsy and TASK, now renamed the Sleuth Kit
http://www.sleuthkit.org/index.php

The Coroners Toolkit
http://www.porcupine.org/forensics/tct.html

A discussion of WU -FTP timestamps in logs
http://hypermail.linklord.com/wuftpd -questions.old/2001/Nov/0081.html

Man pages for:
cmp
http://nodevice.com/sections/ManIndex/man0165.html
nc
http://www.spirit.com/Resources/nc.txt
logrotate
http://nodevice.com/sections/ManIndex/man0735.html

Part 3: Legal Issues

Scenario

I am the system administrator for fake.net.sg, Singapore's smallest Internet Service Provider, which provides dial-up and broadband Internet access to businesses and individuals.

On Feb. 27 2003, I receive a call from a person who identifies herself as Officer Chin Lee Bee, with the Singapore police Criminal Investigation Division's computer crime branch. She says they have evidence of unauthorized access of a Ministry of Defense computer that appears to have originated from our network. She gives me enough information to sift through our logs to see if we have comparable records of activity and whether it originated from our network or elsewhere.

At this point, I tell her I will call back, giving me time to review our logs and an excuse to verify her identity by calling the CID switchboard's listed number, which I do. It turns out Officer Chin is the second in charge of the computer crime branch, underscoring the potential seriousness of the incident.

A review of our records shows a valid user account logged in via a dialup account at the time in question, initiating connections that correspond to the suspicious activity.

Discussion

Q1: What, if any, information can I provide to the law enforcement officer over the phone during the initial contact?

Under Singapore law, I *can* provide any information the officer requests in this instance, including details of the log files and our other records, as well as the identity of the account holder. That is, there is no specific privacy law in Singapore that requires I not disclose such information unless the police officer has a special writ or court order.

In fact, according to privacy researchers Caslon Analytics Pty Ltd (http://www.caslon.com.au), there is no general data protection or privacy law in Singapore. The closest relevant guide is from the government National Internet Advisory Committee, which in 1998 released an "E-Commerce Code for the Protection of Personal Information and Communications of Consumers of Internet Commerce". The code, to which there is no legal requirement for compliance, seeks to set standards to limit collection and unauthorized use of personal information in e-commerce transactions. There is legistation that applies to records associated with electronic commercial transactions, called the Electronic Transactions Act, which

requires those be protected and not disclosed to unauthorized parties. That wouldn't seem to apply in this case, since no transaction defined by the act occurred.

This said, my action would be limited by our internal policy. And there may be overriding legal protection not related to on line privacy. For example, the user account could belong to another government body, whose activity might be covered by the Official Secrets Act (economic data and military information are some examples of its application). I would also want to verify tha t information disclosure didn't violate Singapore's banking secrecy laws.

It so happens that our privacy policy is modeled after that of Pacific Internet, one of the island nation's largest ISP's. Pacific Internet's policy (http://www.pacific.net.sg/article.php?id=1747) says, in part:

> 3. We will not reveal subscriber information to any external party except with your prior consent, or where required under law.

Let's see what the law require s me to do. The main specific legislation in Singapore the covers computer crime is the Computer Misuse Act of 1994, as amended in 1998. Much of the 16-section act focuses on illegal access to and "damage" of computer systems. To wit:

Part III Section 15. "Power of police officer to access computer and data"

1) A police officer or a person authorised in writing by the Commissioner of Police shall:

   a) be entitled at any time to –
      i) have access to and inspect and check the operation of any computer to which this section applies;
      ii) use or cause to be used any such computer to search any data contained in or available to such computer; or
      iii) have access to any information, code or technology which has the capability of retransforming or unscrambling encrypted data conta ined or available to such computer into readable and comprehensible format or text for the purpose of investigating any offence under this Act or any other offence which has been disclosed in the course of the lawful exercise of the powers under this secti on;

   b) be entitled to require –
      i) the person by whom or on whose behalf, the police officer or investigation officer has reasonable cause to suspect, any computer to which this section applies is or has been used; or
      ii) any person having charge of, or otherwise c oncerned with the operation of, such computer,

   to provide him with such reasonable technical and other assistance as he may require for the purposes of paragraph (a); or

c) be entitled to require any person in possession of decryption information to grant him access to such decryption information necessary to decrypt data required for the purpose of investigating any such offence. [21/98]

2) This section shall apply to a computer which a police officer or a person authorised in writing by the Commissioner of Police has reasonable cause to suspect is or has been in use in connection with any offence under this Act or any other offence which has been disclosed in the course of the lawful exercise of the powers under this section. [21/98]

3) The powers referred to in paragraphs (a) (ii) and (iii) and (c) of subsection (1) shall not be exercised except with the consent of the Public Prosecutor. [21/98]

Since our computers are suspected of having been used to commit an offense under the act, this law would apply as per subsection (2) of Section 15. That would entitle a police officer to "access, inspect and check the operation of" any or our computers thought to be involved at any time and under her own authority. It's not clear what the boundaries of "access, inspect and check" are, but since the two subsequent sections specifically cover searching for data on the system and decrypting information stored there, I'm assuming those activities aren't authorized. To compel me to searching and produce data from the computer -- or to force me to allow authorized officers to do so -- requires, not a court order, but "the consent of the Public Prosecutor".

At this point, I'd probably want to talk to our lawyers about whether the ability to "access, inspect and check the operation of" our computers would mean access to our log files and other usage records. Since sharing non-identifying information in those wouldn't violate our privacy policy, there's little to stop me from providing them after getting legal advice. Matching that to the identity of a client or clients appears likely to violate our privacy policy, unless the officer demonstrates she has the consent of the prosecutor's office.

There's no indication that, assuming I verified the officer's identity, she would have to be present in person to request my compliance with the act's provisions at least until she needed to demonstrate the authority of the prosecutor.

Q2: What must the law enforcement officer do to ensure you to preserve this evidence, if there is a delay in obtaining any required legal authority?

Under subsection 1, paragraph (b) (ii) of the same Section 15, I'm obliged to provide all reasonable assistance to aid an officer in carrying out the tasks in paragraph (a), which includes searching for data and decrypting files. There's no mention that an officer must first be authorized to carry out the two tasks before I'm obliged to assist. In other words, as soon as I'm informed of the intention to obtain legal authority to search, I may be compelled to take all reasonable steps to help prepare for that, including presumably preserving evidence. It's also possible that under "access,

inspect and check the operation of" an officer may be empowered to ensure data protection.

What's more, penalties are stiff if I fail to comply and assist. Again, from the Computer Misuse Act, Part III Section 15:

> (4) Any person who obstructs the lawful exercise of the powers under subsection (1) (a) or who fails to comply with a request under subsection (1) (b) or (c) shall be guilty of an offence and shall be liable on conviction to a fine not exceeding $10,000 or to imprisonment for a term not exceeding 3 years or to both. [21/98]

and

Part III Section 16 says:

> Any police officer may arrest without warrant any person reasonably suspected of committing an offence under this Act.

By failing to help with a legitimate request, I'd be open to immediate arrest and prosecution.

Q3: What legal authority, if any, does the law enforcement officer need to provide to you in order for you to send him your logs?

To require that I search for and provide data from a system under my control, including logs, account information and other data necessary for investigating the incident, the police officer would need to demonstrate the consent of the Public Prosecutor's office, as per Section 15, subsection 3 of the act.

In understanding Singapore's approach to weighing individual privacy vs state interests, the policy statements of former Prime Minister Lee Kuan Yew provide a useful framework. Lee, educated as a lawyer, is the architect of modern Singapore and remains its Senior Minister, a cabinet -level post created for him. As the nation's leader from independence from England until the 1990s, he left his imprint on much of public policy.

Speaking in 1986, he said:

> "I am often accused of interfering in the private lives of citizens. Yet, if I did not, had I not done that, we wouldn't be here today. And I say without the slightest remorse, that we wouldn't be here, we would not have made economic progress, if we had not intervened on very personal matters - who your neighbor is, how you live, the noise you make, how you spit, or what language you use.
> We decide what is right, never mind what the people think. That's another problem."

Q4: What other "investigative" activity are you permitted to conduct at this time?

Before Officer Chin called me, this question would have largely been governed by our company policy, save for the areas noted above as being covered by banking secrecy laws, government secrecy laws and the Electronic Transactions Act. After being contacted and being informed of an investigation, I'd also be constrained by two other factors:

1) needing to comply with authorized requests from police. In other words, they could demand that -- within reason and my abilities -- I do much of the system "investigation" work, such as retrieving records and logs, decrypting data archives, explaining our network and system.
2) anything I do while not under the instruction of the police ca n't corrupt records or in any other way interfere with the investigation, or I risk the penalties outlined in section 4.

Q5: How would your actions change if your logs disclosed a hacker gained unauthorized access to your system at some point, created a n account for him/her to use, and used THAT account to hack into the government system?

Under these circumstances, there'd be no right to privacy for the person who used that account under our company policy. Indeed, the unauthorized access and use of our network would itself be a crime under the Computer Misuse Act, which says:

Part II, Section 6

1) Subject to subsection (2), any person who knowingly

   a) secures access without authority to any computer for the purpose of obtaining, directly or indirectly, any computer service;
   b) intercepts or causes to be intercepted without authority, directly or indirectly, any function of a computer by means of an electro -magnetic, acoustic, mechanical or other device; or
   c) uses or causes to be used, directly or indirectly, the computer or any other device for the purpose of committing an offence under paragraph (a) or (b),

   shall be guilty of an offence and shall be liable on conviction to a fine not exceeding $10,000 or to imprisonment for a term not exceeding 3 years or to bo th and, in the case of a second or subsequent conviction, to a fine not exceeding $20,000 or to imprisonment for a term not exceeding 5 years or to both. [21/98]

2) If any damage is caused as a result of an offence under this section, a person convicted of the offence shall be liable to a fine not exceeding $50,000 or to imprisonment for a term not exceeding 7 years or to both. [21/98]

Damage is defined in Part I Section 2, subsection 1, as follows:

> "damage" means, except for the purposes of section 13, an y impairment to a computer or the integrity or availability of data, a program or system, or information, that –
>
> a) causes loss aggregating at least $10,000 in value, or such other amount as the Minister may, by notification in the Gazette, prescribe except that any loss incurred or accrued more than one year after the date of the offence in question shall not be taken into account;
> b) modifies or impairs, or potentially modifies or impairs, the medical examination, diagnosis, treatment or care of one or more p ersons;
> c) causes or threatens physical injury or death to any person; or
> d) threatens public health or public safety;

Part III Section 13 allows courts to determine damages separately for purposes of ordering compensation paid to the victim.

If I discovered that our network was also illegally accessed by the individual who attacked the government computer system, the police would be likely to expand the investigation to include our network compromise – especially since it would compound the crime committed by the cracker and allow more severe sentencing.

Singapore has successfully prosecuted cases of illegal access to ISP's computers under the Computer Misuse Act. In 2000, 17 -year-old Muhammad Nuzaihan Kamal Luddin compromised servers at two Internet service p roviders. He was convicted and sentenced to four months in jail under the act, local media reported.

In this instance, the punishment for accessing a government computer without authorization is much stiffer, with a fine of up to 100,000 Singapore dollars and as long as a 20-year prison sentence. The computer, being a defense department one, is deemed a "protected computer" under the act.

Part II Section 9 "Enhanced punishment for offences involving protected computers"

1) Where access to any protected comp uter is obtained in the course of the commission of an offence under section 3, 5, 6 or 7, the person convicted of such an offence shall, in lieu of the punishment prescribed in those sections, be liable on conviction to a fine not exceeding $100,000 or to imprisonment for a term not exceeding 20 years or to both. [21/98]

2) For the purposes of subsection (1), a computer shall be treated as a "protected computer" if the person committing the offence knew, or ought reasonably to have known, that the computer o r program or data is used directly in connection with or necessary for –

   a) the security, defence or international relations of Singapore;

b) the existence or identity of a confidential source of information relating to the enforcement of a criminal law;
c) the provision of services directly related to communications infrastructure, banking and financial services, public utilities, public transportation or public key infrastructure; or
d) the protection of public safety including systems related to essential emergency services such as police, civil defence and medical services. [21/98]

3) For the purposes of any prosecution under this section, it shall be presumed, until the contrary is proved, that the accused has the requisite knowledge referred to in subsection (2) if there is, in respect of the computer, program or data, an electronic or other warning exhibited to the accused stating that unauthorised access to that computer, program or data attracts an enhanced penalty under this section.[6C [21/98]

The last subsection means that, assuming the defense department computer bannered its services correctly, anyone breaking into the system will be assumed to know that it was a protected system under the act.

References

Singapore's Computer Misuse Act
http://www.lawnet.com.sg/freeaccess/CMA.htm

Caslon Analytics Privacy Guide
http://www.caslon.com.au/privacyguide6.htm

Pacific Internet
http://www.pacific.net.sg

Singapore's National Internet Advisory Committee
http://www.mda.gov.sg/committees/i_niac.html

Summary of Singapore's Electronic Transaction s Act
http://www.egovaspac.org/topics/legislation/downloads/Summary_of_Singapore_Electronic_Transactions_Act.pdf

Then-Prime Minister Lee Kuan Yew on individual rights in Singapore
http://unpan1.un.org/intradoc/groups/public/documents/apcity/unpan002553.pdf

Appendix – Script for volatile dat a capture

```
#CURDIR=`./linux2.2_x86/pwd`
BINDIR="/mnt/cdrom/statbins/linux2.2_x86"

$BINDIR/echo "========="
$BINDIR/echo "cpu info:"
$BINDIR/echo "========="
$BINDIR/cat /proc/cpuinfo
$BINDIR/echo

$BINDIR/echo "========="
$BINDIR/echo "version:"
$BINDIR/e cho "========="
$BINDIR/cat /proc/version
$BINDIR/echo

$BINDIR/echo "==================="
$BINDIR/echo "kernel boot params:"
$BINDIR/echo "==================="
$BINDIR/cat /proc/cmdline
$BINDIR/echo

$BINDIR/echo "================================="
$BINDIR/echo "Local shell environment variables:"
$BINDIR/echo "================================="
$BINDIR/env
$BINDIR/echo

$BINDIR/echo "========================="
$BINDIR/echo "users, runtime history:"
$BINDIR/echo "========================="
$BINDIR/who
$BINDIR/echo

$BINDIR/echo "=========================="
$BINDIR/echo "List of running processes from honeypot's ps:"
$BINDIR/echo "=========================="
ps -eflwwww

$BINDIR/echo
$BINDIR/echo "====================="
$BINDIR/echo "Network interface i nfo"
$BINDIR/echo "====================="
$BINDIR/ifconfig -a
$BINDIR/echo
$BINDIR/ifconfig -s
$BINDIR/echo

$BINDIR/echo
$BINDIR/echo "=================="
$BINDIR/echo "arp table entries:"
$BINDIR/echo "=================="
$BINDIR/arp -n
$BINDIR/echo

$BINDIR/echo
$BINDIR/echo "================================="
$BINDIR/echo "netstat output(current connections)"
$BINDIR/echo "================================="
$BINDIR/netstat -anp
```

```
$BINDIR/echo

$BINDIR/echo "=============="
$BINDIR/echo "routing table :"
$BINDIR/echo "=============="
$BINDIR/netstat  -rn
$BINDIR/echo

$BINDIR/echo "============================"
$BINDIR/echo "listening ports (via lsof):"
$BINDIR/echo "============================"
$BINDIR/lsof  -P -i -n
$BINDIR/echo

$BINDIR/echo "========== =="
$BINDIR/echo "lsof output:"
$BINDIR/echo "============"
$BINDIR/lsof
$BINDIR/echo

$BINDIR/echo "============"
$BINDIR/echo "Memory info:"
$BINDIR/echo "============"
$BINDIR/cat /proc/meminfo
$BINDIR/echo
$BINDIR/echo

$BINDIR/echo "============"
$BINDIR/echo "Module info:"
$BINDIR/echo "============"
$BINDIR/cat /proc/modules
$BINDIR/echo
$BINDIR/echo

$BINDIR/echo "==========="
$BINDIR/echo "Mount info:"
$BINDIR/echo "==========="
$BINDIR/cat /proc/mounts
$BINDIR/echo

$BINDIR/echo "==========="
$BINDIR/echo "Swap info:"
$BINDIR/echo "=========="
$BINDIR/cat /proc/swaps
$BINDIR/echo
$BINDIR/echo "==========="
$BINDIR/echo "   fstab   "
$BINDIR/echo "==========="
$BINDIR/cat /etc/fstab
$BINDIR/echo
$BINDIR/echo
```