Chapter 3

Postmortem Forensics

Discovering and Extracting Malware and Associated Artifacts from Linux Systems

Solutions in this Chapter

INTRODUCTION

If live system analysis can be considered surgery, forensic examination of Linux systems can be considered an autopsy of a computer impacted by malware. Trace evidence relating to a particular piece of malware may be found in various locations on the hard drive of a compromised host, including files, configuration entries, records in system logs, and associated date stamps. Forensic examination of such trace evidence on a Linux system is an important part of analyzing malicious code, providing context and additional information that help us address important questions about a malware incident, including how malware was placed on the system, what it did, and what remote systems were involved.

This chapter provides a repeatable approach to conducting forensic examinations in malware incidents, increasing the consistency across multiple computers, and enabling others to evaluate the process and results. Employing this approach, with a measure of critical thinking on the part of a digital investigator, can uncover information necessary to discover how malware was placed on the system (a.k.a. the intrusion vector), to determine

Malware Forensics Field Guide for Linux Systems. http://dx.doi.org/10.1016/B978-1-59749-470-0.00003-6

malware functionality and its primary purpose (e.g., password theft, data theft, remote control), and to detect other infected systems. This forensic examination process can be applied to both a compromised host and a test system purposely infected with malware, to learn more about the behavior of the malicious code.

Investigative Considerations

- In the past, it was relatively straightforward to uncover traces of malware on the file system and in configuration scripts of a compromised Linux computer. More recently, attackers have been employing anti-forensic techniques to conceal their activities or make malicious files blend in with legitimate ones. For instance, intruders may backdate the inode change time (ctime) date-time stamps on a malicious file to have the same values as a legitimate system file. Intruders also take banners and other characteristics from a legitimate service and compile them into a trojanized version to make it as similar as possible to the legitimate one. Therefore, digital investigators should be alert for misinformation on compromised systems.
- Modern malware is being designed to leave limited traces on the compromised host and store more information in memory rather than on disk. A methodical approach to forensic examination, looking carefully at the system from all perspectives, increases the chances of uncovering footprints that the intruder failed to hide.

Analysis Tip

System Administration versus Forensics

System administrators of Linux systems are often very knowledgeable and, when they find malware on a system, they know enough about their systems to start remediating the problem. However, editing or moving files to "fix" the problem alters crucial evidence, making it more difficult to reconstruct activities related to a malware incident. Therefore, to avoid making matters worse, a forensic duplicate of the compromised system should be acquired before system administrators make alterations.

LINUX FORENSIC ANALYSIS OVERVIEW

☑ ***After a forensic duplicate of a compromised system has been acquired, employ a consistent forensic examination approach to extract the maximum amount of information relating to the malware incident.***

▶ The hard drive of a Linux computer can contain traces of malware in various places and forms, including malicious files, configuration scripts, log files, Web browser history, and remnants of installation and execution such as system logs and command history. In addition, forensic examination of a compromised Linux computer can reveal manipulation such as log deletion and date-time tampering. Some of this information has associated date-time stamps that can be useful for determining when the initial compromise occurred and what happened subsequently. The following general approach is designed to extract the maximum amount of information related to a malware incident:

- Search for Known Malware
- Survey Installed Programs
- Inspect Executables
- Review Services, Modules, and Auto-start Locations
- Review Scheduled Jobs
- Examine Logs (system logs, AntiVirus logs, Web browser history, etc.)
- Review User Accounts
- Examine File System
- Examine Configuration Files
- Perform keyword searches for any specific, known details relating to a malware incident. Useful keywords may come from other forms of analysis, including memory forensics and analysis of the malware itself.
- Harvest available metadata including file system date-time stamps, modification times of configuration files, e-mails, entries in Web browser history, system logs, and other logs such as those created by AntiVirus, crash dump monitoring, and patch management programs. Use this information to determine when the malware incident occurred and what else was done to the system around that time, ultimately generating a time line of potentially malicious events.
- Look for common indicators of anti-forensics including file system date-time stamp alteration, log manipulation, and log deletion.
- Look for links to other systems that may be involved.
- Look for data that should not be on the system such as directories full of illegal materials and software or data stolen from other organizations.

▶ These goals are provided as a guideline and not as a checklist for performing Linux forensic analysis. No single approach can address all situations, and some of these goals may not apply in certain cases. In addition, the specific implementation will depend on the tools that are used and the type of malware involved. Some malware may leave traces in novel or unexpected places on a Linux computer, including in the BIOS or Firmware. Ultimately, the success of the investigation depends on the abilities of the digital investigator to apply digital forensic techniques and adapt them to new challenges.

Analysis Tip

Correlating Key Findings

As noted in prior chapters, knowing the time period of the incident and knowing what evidence of malware was observed can help digital investigators develop a strategy for scouring compromised computers for relevant digital evidence. Therefore, prior to performing forensic analysis of a compromised computer, it is advisable to review all information from the Field Interview Questions in Chapter 1 to avoid wasted effort and missed opportunities. Findings from other data sources, such as memory dumps and network logs, can also help focus the forensic analysis (i.e., the compromised computer was sending packets to a Russian IP address, providing an IP address to search for in a given time frame). Similarly, the results of static and dynamic analysis covered in later chapters can help guide forensic analysis of a compromised computer. So, the analysis of one malware specimen may lead to further forensic examination of the compromised host, which uncovers additional malware that requires further analysis; this cyclical analysis ultimately leads to a comprehensive reconstruction of the incident. In addition, as new traces of malicious activity are uncovered through forensic examination of a compromised system, it is important to document them in a manner that facilitates forensic analysis. One effective approach is to insert new findings into a time line of events that gradually expands as the forensic analysis proceeds. This is particularly useful when dealing with multiple compromised computers. By generating a single time line for all systems, forensic analysts are more likely to observe relationships and gaps.

Investigative Considerations

- It is generally unrealistic to perform a blind review on certain structures that are too large or too complex to analyze without some investigative leads. Therefore, it is important to use all of the information available from other sources to direct a forensic analysis of the compromised system, including interview notes, spearphishing e-mails, volatile data, memory dumps, and logs from the system and network.
- Most file system forensic tools do not provide full metadata from an EXT4 file system. When dealing with malware that likely manipulated date-time stamps, it may be necessary to extract additional attributes from inodes for comparison with the common EXT attributes. Tools for extracting attributes from EXT entries such as The Sleuth Kit and Autopsy GUI shown in Figure 3.1 are presented in the Toolbox section at the end of this chapter.

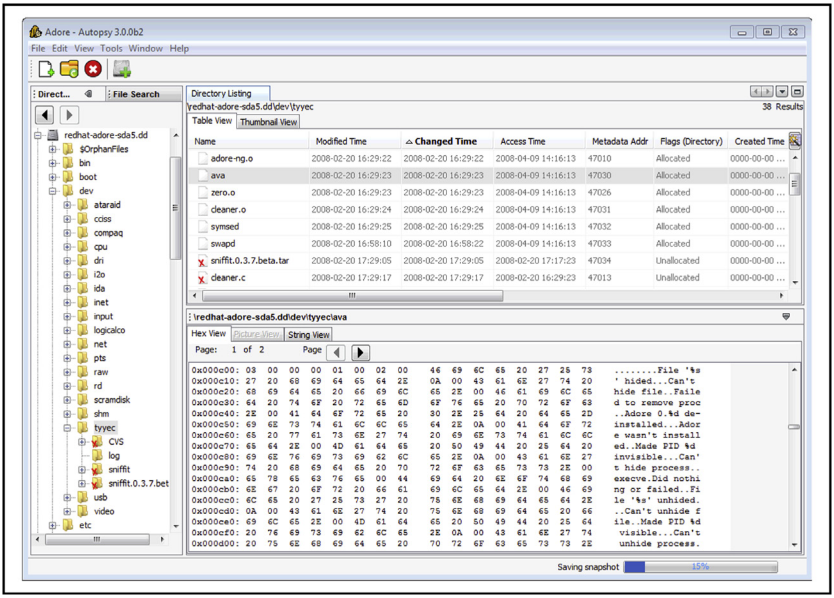


FIGURE 3.1–Linux system being examined using The Sleuth Kit Autopsy GUI

- It is important to look in all areas of a Linux system where traces of malware might be found, even if a quick look in a few common places reveals obvious signs of infection. There may be multiple types of malware on a computer, with more obvious signs of infection presenting a kind of smoke screen that may distract from more subtle traces of compromise. Being thorough, and correlating other information sources (e.g., initial incident reports, network logs) with traces found on the system, reduces the risk that more subtle items will be overlooked.
- No one approach or tool can serve all needs in a forensic examination. To avoid mistakes and missed opportunities, it is necessary to compare the results of multiple tools, to employ different analysis techniques, and to verify important findings manually.

☑ ***In addition to employing forensic tools, mount the forensic duplicate as a logical volume to support additional analysis.***

▶ Although forensic tools can support sophisticated analysis, they cannot solve every problem relating to a malware incident. For instance, running AntiVirus software and rootkit detection tools against files on the compromised system is an important step in examining a compromised host. Figure 3.2 shows the loopback interface being used to mount a forensic duplicate so that it is accessible as a logical volume on the forensic examination system without altering the original evidentiary data. 🛠

```
# mount -o loop,ro,noatime,noexec adore-sda5.dd /mnt/examine

OR

# losetup -r /dev/loop1 adore-sda5.dd
# mount /dev/loop1 /mnt/examine -o loop,ro,noatime,noexec
# ls /mnt/examine
bin    dev   home     lib          misc   opt    root   tftpboot   usr
boot   etc   initrd   lost+found   mnt    proc   sbin   tmp        var
```

FIGURE 3.2–Linux loopback interface used to mount a forensic duplicate

Additional utilities such as FTK Imager, EnCase modules, and Daemon Tools (www.daemon-tools.cc) for mounting a forensic duplicate are discussed in the Tool Box section at the end of this chapter.

Analysis Tip

Trust but Verify

When mounting a forensic duplicate via the Linux loopback interface or using any other method, it is advisable to perform a test run in order to confirm that it does not alter the forensic duplicate. This verification process can be as simple as comparing the MD5 value of the forensic duplicate before and after mounting the file system and performing simple operations such as copying files. Some versions of Linux or some mounting methods may not prevent all changes, particularly when processes are being run as root.

MALWARE DISCOVERY AND EXTRACTION FROM A LINUX SYSTEM

▶ Employing a methodical approach to examining areas of the compromised system that are most likely to contain traces of malware installation and use increases the chances that all traces of a compromise will be uncovered, especially when performed with feedback from the static and dynamic analysis covered in Chapters 5 and 6.

Search for Known Malware

☑ ***Use characteristics from known malware to scour the file system for the same or similar items on the compromised computer.***

▶ Many intruders will use easily recognizable programs such as known rootkits, keystroke monitoring programs, sniffers, and anti-forensic tools (e.g., `touch2`, `shsniff`, `sshgrab`). There are several approaches to locating known malware on a forensic duplicate of a compromised computer.

- **Hashe and File Characteristics**: Searching a forensic duplicate of a compromised system for hash values matching known malware may identify other files with the same data but different names. In addition to using a hash database such as NSRL, another approach to identifying malicious code is to look for deviations from known good configurations of the system. Some Linux systems have a feature to verify the integrity of many installed components, providing an effective way to identify unusual or out of place files. For instance, `rpm -Va` on Linux is designed to verify all packages that were installed using RedHat Package Manager. For instance, the results of this verification process in the T0rnkit scenario are shown in Figure 3.3 to show binaries that have different filesize (S), mode (M), and MD5 (5) than expected. Some of these binaries also have discrepancies in the user (U), group (G), and modified time (T). With `rpm` it is also possible to specify a known good database using the `--dbpath` option, when there are concerns that the database on the subject system is not trustworthy.

```
# rpm -Va --root=/mntpath/evidence | grep SM5
SM5..UG.   /sbin/syslogd
SM5..UG.   /usr/bin/find
SM5....T c /etc/conf.linuxconf
SM5..UG.   /usr/sbin/lsof
SM5..UG.   /bin/netstat
SM5..UG.   /sbin/ifconfig
SM5..UGT   /usr/bin/ssh
SM5..UG.   /usr/bin/slocate
SM5..UG.   /bin/ls
SM5..UG.   /usr/bin/dir
SM5..UG.   /usr/bin/md5sum
SM5..UG.   /bin/ps
SM5..UG.   /usr/bin/top
SM5..UG.   /usr/bin/pstree
SM5....T c /etc/ssh/sshd_config
```

FIGURE 3.3–T0rnkit rootkit files found using RPM verify

- **Rootkit Detectors**: Tools such as Rootkit Hunter[1] and chkrootkit[2] have been developed to look for known malicious code on Linux systems. These programs contain a regularly updated database of known malware, and can be used to scan a forensic duplicate. Many of the rootkit checks can be run against a mounted image as shown in Figure 3.4, but some checks can only be performed on a running system, such as scanning running processes for malware. Be aware that these rootkit scanning tools may only detect rootkit files that are in a specific, default location. Therefore, a specific rootkit may not be detected by these scanning tools if the files

1 http://rkhunter.sourceforge.net.
2 http://www.chkrootkit.org/.

are not in the expected location (false negative). These scanning tools also often have false positive hits, flagging legitimate files as possible rootkit components.

```
# rkhunter --check -r /media/_root -l /evidence/rkhunter.log
[ Rootkit Hunter version 1.3.8 ]
Checking system commands...
Performing 'strings' command checks
Checking 'strings' command                                [ OK ]

  Performing file properties checks
    Checking for prerequisites                            [ Warning ]
    /media/_root/sbin/chkconfig                           [ Warning ]
<excerpted for brevity>

Checking for rootkits...
  Performing check of known rootkit files and directories
    55808 Trojan - Variant A                             [ Not found ]
    ADM Worm                                             [ Not found ]
    AjaKit Rootkit                                       [ Not found ]
    Adore Rootkit                                        [ Warning ]

  Performing additional rootkit checks
    Suckit Rookit additional checks                       [ OK ]
    Checking for possible rootkit files                  [ Warning ]
    Checking for possible rootkit strings                [ Warning ]

=====================

Rootkit checks...
    Rootkits checked : 227
    Possible rootkits: 3
    Rootkit names    : Adore, Tuxtendo, Rootkit component

One or more warnings have been found while checking the system.
Please check the log file (/evidence/rkhunter.log)
```

FIGURE 3.4–Scanning a target drive image with rkhunter

- **AntiVirus**: Using updated AntiVirus programs to scan files within a forensic duplicate of a compromised system may identify known malware. To increase the chances of detecting malware, multiple AntiVirus programs can be used with any heuristic capabilities enabled. Such scanning is commonly performed by mounting a forensic duplicate on the examination system and configuring AntiVirus software to scan the mounted volume as shown in Figure 3.5 using Clam AntiVirus.[3] Another AntiVirus program for Linux is F-Prot.[4]

[3] http://www.clamav.net/.
[4] http://www.f-prot.com.

```
# clamscan -d /examination/clamdb -r -i -l
clamscan.log /mnt/evidence

----------- SCAN SUMMARY -----------
Known viruses: 1256684
Engine version: 0.97.3
Scanned directories: 20
Scanned files: 46
Infected files: 1
Data scanned: 0.29 MB
Data read: 3340.26 MB (ratio 0.00:1)
Time: 6.046 sec (0 m 6 s)
```

FIGURE 3.5–Clam AntiVirus software scanning a mounted forensic duplicate

- **Piecewise Comparison**: When known malware files are available for comparison purposes, a tool such as `frag_find`[5] can be used to search for parts of the reference dataset on the compromised system. In addition, a piecewise comparison tool such as `ssdeep`[6] may reveal malware files that are largely similar with slight variations. Using the matching mode, with a list of fuzzy hashes of known malware, may find specimens that are not detected with an exact hash match or by current anti-virus definitions (e.g., when embedded IP addresses change).

Analysis Tip

Existing Security Software Logs

Given the prevalence of security monitoring software, it is advisable to review any logs that were created by AntiVirus software or other programs that were running on the compromised system for indications of malware. Many AntiVirus programs have logging and quarantine features that can provide information about detected malware. When a system is running Tripwire or other system integrity checking tools that monitor the system for alterations, daily reports might exist showing which files were added, changed, and deleted during a malware incident.

- **Keywords**: Searching for IRC commands and other traits commonly seen in malware, and any characteristics that have been uncovered during the digital investigation (e.g., IP addresses observed in network-level logs) may uncover malicious files on the system. Strings within core system components can reveal that they have been trojanized by the intruder. For instance, Figure 3.6 shows a shared library from a compromised system

---

5 https://github.com/simsong/frag_find (part of the NPS Bloom filter package).
6 http://ssdeep.sourceforge.net.

with unusual functions named `proc_hackinit` and `proc_istrojaned`, `fp_hack`, `hack_list` and `proc_childofhidden`, which demonstrates that "trojan," "hack," and "hidden" may be useful keywords when investigating some malware incidents.

```
from_gid·getgrgid·bad_user_access_length·openproc·opendir·closeproc·closedir·
freeproc·status2proc·sscanf·stat2proc·strrchr·statm2proc·nulls2sep·file2str·f
ile2strvec·readproc·readdir·strcat·proc_istrojaned·ps_readproc·look_up_our_se
lf·getpid·LookupPID·readproctree·readproctab·freeproctab·list_signals·stdout·
_IO_putc·get_signal·get_signal2·status·uptime·_exit·lseek·Hertz·four_cpu_numb
ers·loadavg·meminfo·read_total_main·procps_version·display_version·sprint_upt
ime·time·localtime·setutent·getutent·endutent·av·print_uptime·pname·hname·pro
c_addpid·pidsinuse·pids·pid·proc_hackinit·xor_buf·h_tmp·fp_hack·tmp_str·fgets
·hack_list·strp·strtok·proc_childofhidden·libc.so.6·___brk_addr·__curbrk·__en
viron·atexit·_etext·_edata·__bss_start·_end·libproc.so.2.0.6·GLIBC_2.1·GLIBC_
2.0
```

FIGURE 3.6–Extract from a trojanized shared library (/lib/libproc.so.2.0.6) with unusual function names

Investigative Considerations

- Some malware provides an installation option to delete the executable from disk after loading into memory. Therefore, in addition to scanning logical files, it can be worthwhile to carve all executables out of the swap partition and unallocated space in order to scan them using AntiVirus software as well, particularly when malware has been deleted by the intruder (or by AntiVirus software that was running on the compromised system).
- Some malware is specifically designed to avoid detection by hash values, AntiVirus signatures, rootkit detection software, or other similarity characteristics. Therefore, the absence of evidence in an AntiVirus scan or hash analysis should not be interpreted as evidence that no malware is on the system. For example, the Phalanx2 rootkit periodically changes the name of its executables and now stores its components and TTY sniffer logs in a randomly named directory. For instance, in one incident the `/etc/khubd.p2` directory contained files related to the Phalanx2 rootkit shown in Figure 3.7.[7] However, every part of the rootkit and hidden directory is subject to change in later versions of Phalanx2, including the location and names of files.

```
-rw-r--r--  1 root  root   1356 Jul 24 19:58 .p2rc
-rwxr-xr-x  1 root  root   561032 Jul 24 19:58 .phalanx2*
-rwxr-xr-x  1 root  root   7637 Jul 28 15:04 .sniff*
-rw-r--r--  1 root  53746  1063 Jul 24 20:56 sshgrab.py
```

FIGURE 3.7–Phalanx2 rootkit and TTY sniffer components located in a hidden directory

[7] http://hep.uchicago.edu/admin/report_072808.html.

- Given that intruders can make a trojanized application look very similar to the legitimate one that was originally installed on the compromised system, it is advisable to compare critical applications such as SSH with the original package obtained from a trusted source. Any discrepancies between the MD5 hash values of SSH binaries on a compromised system and those from a trusted distribution of the same version warrant further investigation.
- If backups of the compromised system exist, they can be used to create a customized hashset of the system at various points in time. Such a customized hashset can be used to determine which files were added or changed since the backup was created. In one case, intruders made a trojanized SSH package indistinguishable from the original, legitimate package, making it necessary to perform hashset comparisons with files from backups. This comparison also helped narrow down the time frame of the intrusion, because the trojanized files were on a backup from February but not an earlier backup from January.
- Keyword searches for common characteristics in malware can also trigger on AntiVirus definition files, resulting in false positives.

Survey Installed Programs and Potentially Suspicious Executables

☑ ***Review the programs that are installed on the compromised system for potentially malicious applications.***

▶ Surveying the names and installation dates of programs and executable files that were installed on the compromised computer may reveal ones that are suspicious, as well as legitimate programs that can be used to gain remote access or to facilitate data theft.

- This process does not require in-depth analysis of each program. Instead look for items that are unexpected, questionable, or were installed around the time of the incident.
- Many applications for Linux systems are distributed as "packages" that automate their installation. On Debian-based systems, the `/var/lib/dpkg/status` file contains details about installed packages and the `/var/log/dpkg.log` file records information when a package is installed. For instance, entries in the dpkg.log file on an Ubuntu system revealing that `nmap` was installed are shown in Figure 3.8. On RedHat and related Linux distributions the `rpm -qa --root=/mntpath/var/lib/rpm` command will list the contents of an RPM database on a subject systems.

```
# tail -15 /mntpath/var/log/dpkg.log
2012-06-12 14:48:20 startup archives unpack
2012-06-12 14:48:22 install nmap <none> 5.21-1.1
2012-06-12 14:48:22 status half-installed nmap 5.21-1.1
2012-06-12 14:48:23 status triggers-pending man-db 2.6.0.2-2
2012-06-12 14:48:23 status half-installed nmap 5.21-1.1
2012-06-12 14:48:23 status unpacked nmap 5.21-1.1
2012-06-12 14:48:23 status unpacked nmap 5.21-1.1
2012-06-12 14:48:23 trigproc man-db 2.6.0.2-2 2.6.0.2-2
2012-06-12 14:48:23 status half-configured man-db 2.6.0.2-2
2012-06-12 14:48:27 status installed man-db 2.6.0.2-2
2012-06-12 14:48:28 startup packages configure
2012-06-12 14:48:28 configure nmap 5.21-1.1 <none>
2012-06-12 14:48:28 status unpacked nmap 5.21-1.1
2012-06-12 14:48:28 status half-configured nmap 5.21-1.1
2012-06-12 14:48:28 status installed nmap 5.21-1.1
```

FIGURE 3.8–Log entries (`/var/log/dpkg.log`) showing installation of potentially malicious program (nmap) on a Debian-based Linux system (Ubuntu)

- Not all installed programs will be listed by the above commands because some applications are not available as packages for certain systems and must be installed from source. Therefore, a review of locations such as `/usr/local` and `/opt` may reveal other applications that have been compiled and installed from source code. On RedHat and related Linux distributions the command `find /mntpath/sbin -exec rpm -qf {} \; | grep "is not"` command will list all executables in the `/sbin` directory on a mounted forensic duplicate that are not associated with a package.
- A malicious program may be apparent from a file in the file system (e.g., sniffer logs, RAR files, or configuration scripts). For example, Figure 3.9 shows sniffer logs on a compromised system that network traffic is being recorded by malware on the system.

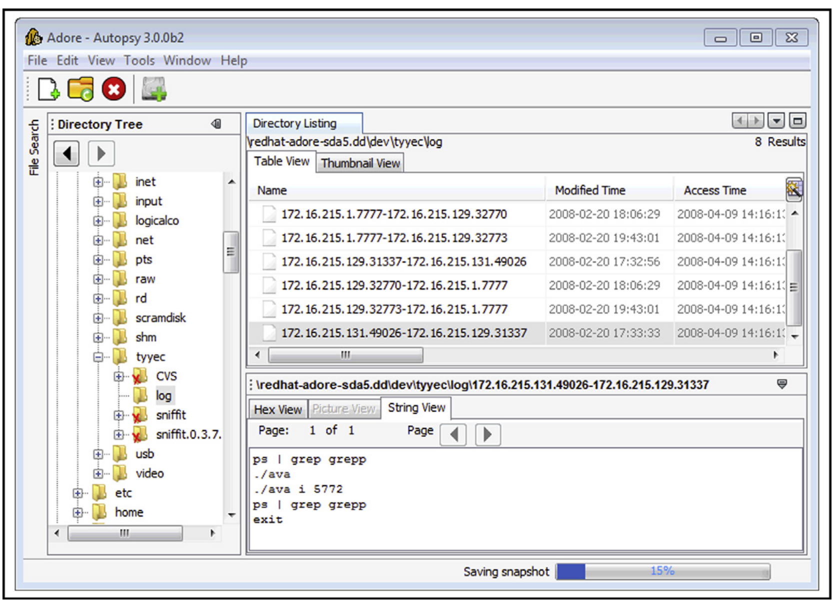


FIGURE 3.9–Sniffer logs on a compromised system viewed using The Sleuth Kit

- Legitimate programs installed on a computer can also play a role in malware incidents. For instance, PGP or remote desktop programs (e.g., X) installed on a system may be normal in certain environments, but its availability may have enabled intruders to use it for malicious purposes such as encrypting sensitive information before stealing it over the network. Coordination with the victim organization can help determine if these are legitimate typical business use applications. Even so, keep in mind that they could be abused/utilized by the intruder and examination of associated logs may be fruitful.

Analysis Tip

Look for Recently Installed or Out-of-Place Executables
Not all installed programs will be listed by the above commands because intruders might put executables in unexpected locations. Therefore, it may be necessary to look for recently installed programs that coincide with the timing of the malware incident, or use clues from other parts of the investigation to focus attention on potentially suspicious applications. In addition, look for executable files in user home directories and other locations that are commonly accessed by users but that do not normally contain executables.

Investigative Considerations

- Reviewing every potential executable on a computer is a time-consuming process and an important file may be missed in the mass of information. Digital investigators can generally narrow their focus to a particular time period or region of the file system in order to reduce the number of files that need to be reviewed for suspicious characteristics. In addition, look for executable files in locations that are commonly accessed by users but that do not normally contain executables such as an IRC bot running from a compromised user account.
- Malware on Linux systems is often simply a modified version of a legitimate system binary, making it more difficult to distinguish. However, digital investigators may find malware that has been Base64 encoded or packed using common methods such as UPX or Burneye.
- The increase in "spearphishing attacks," which employ social engineering to trick users to click on e-mail attachments, combined with malware embedded in Adobe PDFs as discussed in Chapter 5 means that digital investigators need to expand searches for malware to include objects embedded in documents and e-mail attachments.

Inspect Services, Modules, Auto-Starting Locations, and Scheduled Jobs

☑ ***Look for references to malware in the various startup locations on compromised systems to determine how malware managed to remain running on a Linux system after reboots.***

▶ To remain running after reboots, malware is usually relaunched using some persistence mechanism available in the various startup methods on a Linux system, including services, drivers, scheduled tasks, and other startup locations.

- **Scheduled Tasks**: Some malware uses the Linux cronjob scheduler to periodically execute and maintain persistence on the system. Therefore, it is important to look for malicious code that has been scheduled to execute in the `/var/spool/cron/crontabs` and `/var/spool/cron/atjobs` configuration files.
- **Services**: It is extremely common for malware to entrench itself as a new, unauthorized service. Linux has a number of scripts that are used to start services as the computer boots. The initialization startup script `/etc/inittab` calls other scripts such as `rc.sysinit` and various startup scripts under the `/etc/rc.d/` directory, or `/etc/rc.boot/` in some older versions. On other versions of Linux, such as Debian, startup scripts are stored in the `/etc/init.d/` directory. In addition, some common services are enabled in `/etc/inetd.conf` or `/etc/xinetd/` depending on the version of Linux. Digital investigators should inspect each of these startup scripts for anomalous entries. For example, in one intrusion, the backdoor was restarted whenever the compromised system rebooted by placing the entries in Figure 3.10 at the end of the `/etc/rc.d/rc.sysinit` system startup file.

```
# Xntps (NTPv3 daemon) startup..
/usr/sbin/xntps -q
# Xntps (NTPv3 deamon) check..
/usr/sbin/xntpsc 1>/dev/null 2>/dev/null
```

FIGURE 3.10–Malicious entries in `/etc/rc.d/rc.sysinit` file to restart backdoor on reboot

The Phalanx2 rootkit is launched from a separate startup script under the `/etc/rc.d/` directory with the same randomly generated name as the hidden directory where the rootkit components are stored. Be warned that Phalanx2 also hides the startup script from users on the system, making forensic examination of the file system an important part of such malware investigations.

- **Kernel Modules**: On Linux systems, kernel modules are commonly used as rootkit components to malware packages. Kernel modules are loaded when the system boots up based on the configuration information in the `/lib/modules/'uname -r'` and `/etc/modprobe.d` directories, and the `/etc/modprobe` or `/etc/modprobe.conf` file. These areas should be inspected for items that are related to malware.
- **Autostart Locations**: There are several configuration files that Linux uses to automatically launch an executable when a user logs into the system that may contain traces of malware. Items in the `/etc/profile.d` directory and the `/etc/profile` and `/etc/bash.bashrc` files are executed when any user account logs in and may be of interest in malware incident. In addition, each user account has individual configuration files (`~/.bashrc`, `~/.bash_profile` and `~/.config/autostart`) that can contain entries to execute malware when a specific user account logs into the system.

Investigative Considerations

- Check all programs that are specified in startup scripts to verify that they are correct and have not been replaced by trojanized programs.
- Intruders sometimes enable services that were previously disabled, so it is also important to check for legitimate services that should be disabled.

Examine Logs

☑ ***Look in all available log files on the compromised system for traces of malicious execution and associated activities such as creation of a new service.***

▶ Linux systems maintain a variety of logs that record system events and user account activities. The main log on a Linux system is generally called `messages` or `syslog`, and the `security` log records security-specific events. Some Linux systems also have audit subsystems (e.g., SELinux) configured to record specific events such as changes to configuration files. The degree of detail in these logs varies, depending on how logging is configured on a given machine.

- **System Logs**: Logon events recorded in the system and security logs, including logons via the network, can reveal that malware or an intruder gained access to a compromised system via a given account at a specific time. Other events around the time of a malware infection can be captured in system logs, including the creation of a new service or new accounts around the time of an incident. Most Linux logs are in plain text and can be searched using a variety of tools, including `grep` and Splunk[8] with the ability to filter on specific types of events.

Certain attacks create distinctive patterns in logs that may reveal the vector of attack. For instance, buffer overflow attacks may cause many log entries to be generated with lengthy input strings as shown in Figure 3.11 from the `messages` log.

```
Apr  8 07:47:26 localhost SERVER[5151]: Dispatch_input: bad request line
'BBàóÿ¿áóÿ¿âóÿ¿ãóÿ¿XXXXXXXXXXXXXXXXXXX00000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000480000000107383508 8security000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000061Û1É1À°FÍ€‰å1Ò²f‰Ð1É‰ËC‰]øC‰]ôK‰Mü□MôÍ€1É‰EôCf‰]
ìfÇEî^O'‰Mð□Eì‰EøÆEü^P‰Ð□MôÍ€‰ÐCCÍ€‰ÐCÍ€‰Ã1É²?‰ÐÍ€‰ÐAÍ€ë^X^‰u^H1À^F^G‰E^L°^K‰
ó□M^H□U^LÍ€èãÿÿÿ/bin/sh'
```

FIGURE 3.11–Log entry showing buffer overflow attack against a server to launch a command shell

This log entry shows the successful buffer overflow had "/bin/sh" at the end, causing the system to launch a command shell that the intruder used to gain unauthorized access to the system with root level privileges.

- **Web Browser History**: The records of Web browsing activity on a compromised computer can reveal access to malicious Web sites and subsequent download of malware. In addition, some malware leaves traces in the Web browser history when it spreads to other machines on the network. Firefox is a common Web browser on Linux systems and historical records of browser events are stored in a user profile under the `~/.mozilla/firefox` directory for each user account.
- **Command History**: As detailed in Chapter 1, many Linux systems are configured to maintain a command history for each user account (e.g., `.bash_history`, `.history`, `.sh_history`). Figure 3.12 shows a command history from a Linux system that had its entire hard drive copied over the network using netcat. Although entries in a command history file are not time stamped (unless available in memory dumps as discussed in Chapter 2), it may be possible to correlate some entries with the last accessed dates of the associated executables, in an effort to determine when the events recorded in the command history log occurred. Some Linux systems maintain process accounting (`pacct`) logs, which can be viewed using the `lastcomm` command. These logs record every command that was executed on the system along with the time and user account.

[8] http://www.splunk.com/.

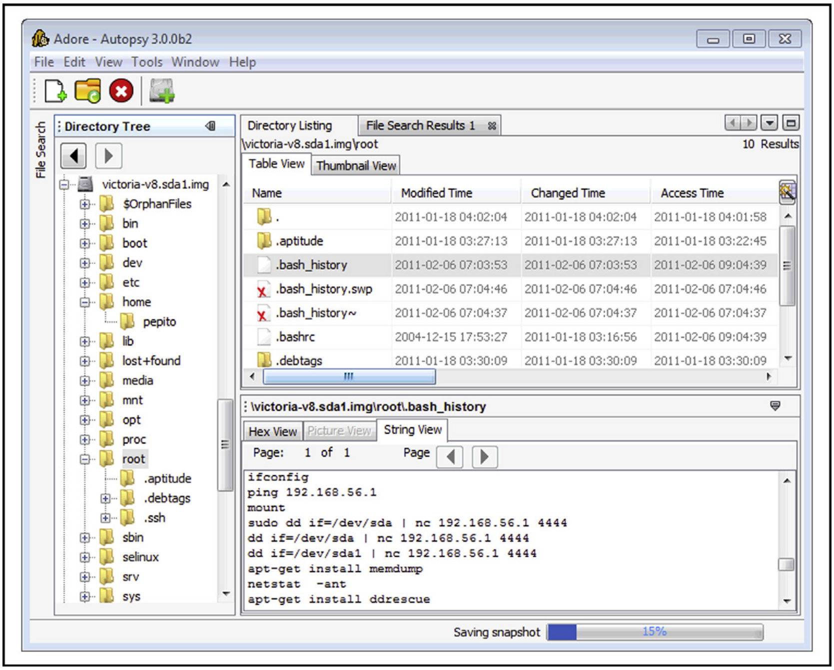


FIGURE 3.12–Command history contents viewed using The Sleuth Kit and Autopsy GUI

- **Desktop Firewall Logs**: Linux host-based firewalls such as IPtables and other security programs (e.g., `tcp_wrappers`) function at the packet level, catching each packet before it is processed by higher level applications and, therefore, may be configured to create very detailed logs of malicious activities on a compromised system.
- **AntiVirus Logs**: When a Linux system is compromised, AntiVirus software may detect and even block some malicious activities. Such events will be recorded in a log file with associated date-time stamps (e.g., under `/var/log/clamav/` for ClamAV), and any quarantined items may still be stored by the AntiVirus software in a holding area.
- **Crash Dump**: When configured, the `abrt` service can capture information about programs that crashed and produced debug information. When `abrtd` traps a crashing program, it creates a file named `coredump` (under `/var/spool/abrt` by default) containing memory contents from the crash, which may provide useful information such as attacker IP addresses.

Investigative Considerations

- Log files can reveal connections from other computers that provide links to other systems on the network that may be compromised.

- Not all programs make an entry in Linux logs in all cases, and malware installed by intruders generally bypass the standard logging mechanisms.
- Linux system logs and audit subsystems may be disabled or deleted in an intrusion or malware incident. In fact, because logs on Linux systems generally contain some of the most useful information about malicious activities, intruders routinely delete them. Therefore, when examining available log files, it is important to look for gaps or out of order entries that might be an indication of deletion or tampering. Because Linux generates logs on a regular basis during normal operation, a system that is not shut down frequently, such as a server, should not have prolonged gaps in logs. For instance, when logs are loaded into Splunk, a histogram of events by day is generated automatically and can show a gap that suggests log deletion. In addition, it is generally advisable to search unallocated space for deleted log entries as discussed in the Examine Linux File System later in this chapter.
- Keep in mind that log entries of buffer overflows merely show that a buffer overflow attack occurred, and not that the attack was successful. To determine whether the attack was successful, it is necessary to examine activities on the system following the attack.
- Rootkits and trojanized services have a tendency to be unstable and crash periodically. Even if a service such as the ABRT package is not installed, kernel activity logs (e.g., `dmesg`, `kern.log`, `klog`) can show that a particular service crashed repeatedly, potentially indicating that an unstable trojanized version was installed.

Analysis Tip

Centralized Syslog Server

In some enterprise environments, syslog servers are relied on to capture logging and so local security event logging is sparse on individual Linux computers. Given the volume of logs on a syslog server, there may be a retention period of just a few days and digital investigators must preserve those logs quickly or risk losing this information.

Review User Accounts and Logon Activities

☑ ***Verify that all accounts used to access the system are legitimate accounts and determine when these accounts were used to log onto the compromised system.***

▶ Look for the unauthorized creation of new accounts on the compromised system, accounts with no passwords, or existing accounts added to Administrator groups.

- **Unauthorized Account Creation**: Examine the `/etc/passwd`, `/etc/shadow` and security logs for unusual names or accounts created and/or used in close proximity to known unauthorized events.
- **Administrator Groups**: It is advisable to check `/etc/sudoers` files for unexpected accounts being granted administrative access and check `/etc/groups` for unusual groups and for user accounts that are not supposed to be in local or domain-level administrator groups. In addition, consult with system administrators to determine whether a centralized authorization mechanism is used (e.g., NIS, Kerberos).
- **Weak/Blank Passwords**: In some situations it may be necessary to look for accounts with no passwords or easily guessed passwords. A variety of tools are designed for this purpose, including John the Ripper[9] and Cain & Abel.[10] Rainbow tables are created by precomputing the hash representation of passwords and creating a lookup table to accelerate the process of checking for weak passwords.[11]

Investigative Considerations

- Failed authentication attempts, including `sudo` attempts, can be important when repeated efforts were made to guess the passwords. In one investigation, after gaining access to a Linux server via a normal user account, the intruders used `sudo` repeatedly until they guessed the password of an account with root privileges. The multiple failed `sudo` attempts were captured in system logs, but the intruders deleted these logs after obtaining root. The deleted log entries were salvaged by performing a keyword search of unallocated space.
- Malware or intruders may overwrite log entries to eliminate trace evidence of unauthorized activities. Therefore, keep in mind that activities may have occurred that are not evident from available and salvaged logs, and it may be necessary to pay greater attention to details and correlation of information from multiple sources to get a more complete understanding of a malware incident. In such situations, a centralized syslog server or network-level logs such as NetFlow can be invaluable for filling in gaps of activities on a compromised host.

---

9 www.openwall.com/john/.
10 http://www.oxid.it/cain.html.
11 http://project-rainbowcrack.com or http://www.antsight.com.

Analysis Tip

Correlation with Logons

Combine a review of user accounts with a review of Linux security logs on the system to determine logon times, dates of account creation, and other activities related to user account activity on the compromised system. This can reveal unauthorized access, including logons via SSH or other remote access methods

EXAMINE LINUX FILE SYSTEM

☑ ***Explore the file system for traces left by malware.***

▶ File system data structures can provide substantial amounts of information related to a malware incident, including the timing of events and the actual content of malware. Various software applications for performing forensic examination are available but some have significant limitations when applied to Linux file systems. Therefore, it is necessary to become familiar with tools that are specifically designed for Linux forensic examination, and to double check important findings using multiple tools. In addition, malware is increasingly being designed to thwart file system analysis. Some malware alter date-time stamps on malicious files to make it more difficult to find them with time line analysis. Other malicious code is designed to only store certain information in memory to minimize the amount of data stored in the file system. To deal with such anti-forensic techniques, it is necessary to pay careful attention to time line analysis of file system date-time stamps and to files stored in common locations where malware might be found.

- One of the first challenges is to determine what time periods to focus on initially. An approach is to use the `mactime` histogram feature in the Sleuth Kit to find spikes in activity as shown in Figure 3.13. The output of this command shows the most file system activity on April 7, 2004, when the operating system was installed, and reveals a spike in activity on April 8, 2004, around 07:00 and 08:00, which corresponds to the installation of a rootkit.

```
# mactime -b /tornkit/body -i hour index.hourly 04/01/2004-
04/30/2004
      Hourly Summary for Timeline of /tornkit/body
      Wed Apr 07 2004 09:00:00: 43511
      Wed Apr 07 2004 13:00:00: 95
      Wed Apr 07 2004 10:00:00: 4507
      Wed Apr 07 2004 14:00:00: 4036
      Thu Apr 08 2004 07:00:00: 6023
      Thu Apr 08 2004 08:00:00: 312
```

FIGURE 3.13–Histogram of file system date-time stamps created using mactime

- Search for file types that attackers commonly use to aggregate and exfiltrate information. For example, if PGP files are not commonly used in the victim environment, searching for .asc file extensions and PGP headers may reveal activities related to the intrusion.
- Review the contents of the `/usr/sbin` and `/sbin` directories for files with date-time stamps around the time of the incident, scripts that are not normally located in these directories (e.g., .sh or .php scripts), or executables not associated with any known application (hash analysis can assist in this type of review to exclude known files).
- Since many of the items in the `/dev` directory are special files that refer to a block or character device (containing a "b" or "c" in the file permissions), digital investigators may find malware by looking for normal (non-special) files and directories.
- Look for unusual or hidden files and directories, such as ".. " (dot dot space) or "..^G " (dot dot control-G), as these can be used to conceal tools and information stored on the system.
- Intruders sometimes leave setuid copies of `/bin/sh` on a system to allow them root level access at a later time. Digital investigators can use the following commands to find setuid root files on the entire file system:

  ```
  find /mnt/evidence -user root -perm -04000 -print
  ```
- When one piece of malware is found in a particular directory (e.g., `/dev` or `/tmp`), an inspection of other files in that directory may reveal additional malware, sniffer logs, configuration files, and stolen files.
- Looking for files that should not be on the compromised system (e.g., illegal music libraries, warez, etc.) can be a starting point for further analysis. For instance, the location of such files, or the dates such files were placed on the system, can narrow the focus of forensic analysis to a particular area or time period.
- Time line analysis is one of the most powerful techniques for organizing and analyzing file system information. Combining date-time stamps of malware-related files and system-related files such as startup scripts and application configuration files can lead to an illuminating reconstruction of events surrounding a malware incident, including the initial vector of attack and subsequent entrenchment and data theft.

> 🛠 Tools for generating time lines from Linux file systems, including plaso, which incorporates log entries, are discussed in the Tool Box section.

- Review date-time stamps of deleted inodes for large numbers of files being deleted around the same time, which might indicate malicious activity such as installation of a rootkit or trojanized service.
- Because inodes are allocated on a next available basis, malicious files placed on the system at around the same time may be assigned consecutive inodes. Therefore, after one component of malware is located, it can be productive to inspect neighboring inodes. A corollary of such inode analysis is to look for files with out-of-place inodes among system binaries (Altheide and Casey, 2010). For instance, as shown in Figure 3.14, if malware was placed in `/bin` or `/sbin` directories, or if an application was replaced with a trojanized version, the inode number may appear as an outlier because the new inode number would not be similar to inode numbers of the other, original files.

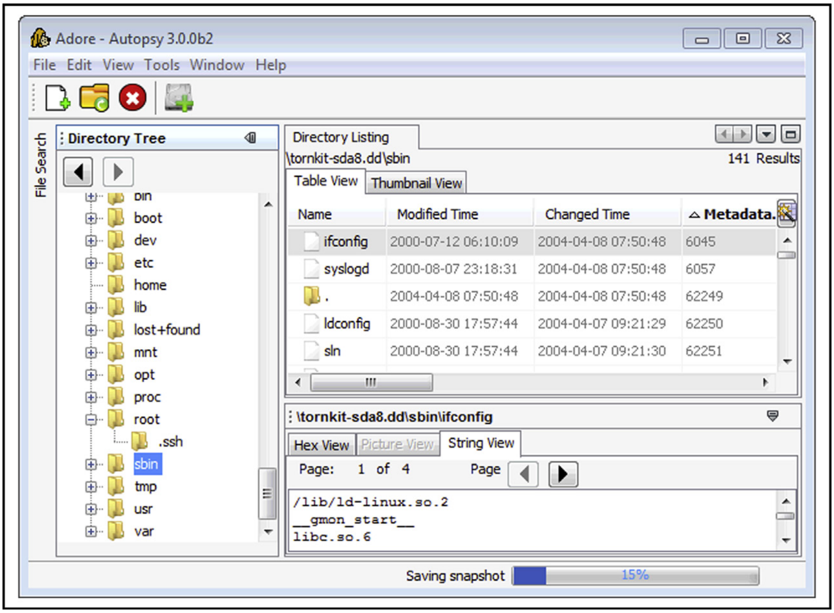


FIGURE 3.14–Trojanized binaries ifconfig and syslogd in /sbin have inode numbers that differ significantly from the majority of other (legitimate) binaries in this directory

- Some digital forensic tools sort directory entries alphabetically rather than keeping them in their original order. This can be significant when malware creates a directory and the entry is appended to the end of the directory listing. For example, Figure 3.15 shows the Digital Forensic Framework displaying the contents of the `/dev` directory in the left window pane with entries listed in the order that they exist within the directory file rather than ordered alphabetically (the `tyyec` entry was added last and contains adore rootkit files). In this situation, the fact that the directory is last can be helpful in determining that it was created recently, even if date-time stamps have been altered using anti-forensic methods.

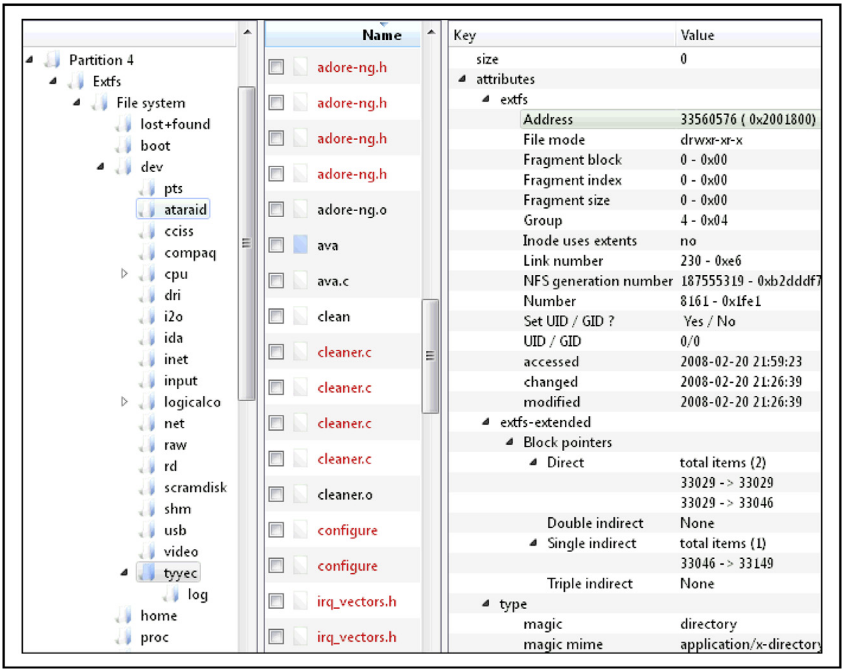


FIGURE 3.15–Rootkit directory displayed using the Digital Forensics Framework, which retains directory order

- Once malware is identified on a Linux system, examine the file permissions to determine their owner and, if the owner is not root, look for other files owned by the offending account.

Investigative Considerations

- It is often possible to narrow down the time period when malicious activity occurred on a computer, in which case digital investigators can create a time line of events on the system to identify malware and related components, such as keystroke capture logs.
- There are many forensic techniques for examining Linux file systems that require a familiarity with the underlying data structures such as inode tables and journal entries. Therefore, to reduce the risk of overlooking important information, for each important file and time period in a malware incident, it is advisable to look in a methodical and comprehensive manner for patterns in related/surrounding inodes, directory entries, filenames, and journal entries using Linux forensic tools.

- Although it is becoming more common for the modified time (mtime) of a file to be falsified by malware, the inode change time (ctime) is not typically updated. Therefore, discrepancies between the mtime and ctime may indicate that date-time stamps have been artificially manipulated (e.g., an mtime before the ctime).
- The journal on EXT3 and EXT4 contains references to file system records that can be examined using the `jls` and `jcat` utilities in TSK.[12]
- The increasing use of anti-forensic techniques in malware is making it more difficult to find traces on the file system. To mitigate this challenge, use all of the information available from other sources to direct a forensic analysis of the file system, including memory and logs.

EXAMINE APPLICATION TRACES

☑ ***Scour files associated with applications for traces of usage related to malware.***

▶ Linux systems do not have a central repository of information like the Windows Registry, but individual applications maintain files that can contain traces of activities related to malicious activities. Some common examples of applications traces are summarized below.

- **SSH**: Connections to systems made using SSH to and from a compromised system result in entries being made in files for each user account (`~/.ssh/authorized_keys` and `~/.ssh/known_keys`). These entries can reveal the hostname or IP address of the remote hosts as shown in Figure 3.16.
- **Gnome Desktop**: User accounts may have a `~/.recently-used.xbel` file that contains information about files that were recently accessed using applications running in the Gnome desktop.
- **VIM**: User accounts may have a `~/.viminfo` file that contains details about the use of VIM, including search string history and paths to files that were opened using vim.
- **Open Office**: Recent files.
- **MySQL:** User accounts may have a `~/.mysql_history` file that contains queries executed using MySQL.
- **Less:** User accounts may have a `~/.lesshst` file that contains details about the use of less, including search string history and shell commands executed via less.

---

[12] Gregorio Narváez "Taking advantage of Ext3 journaling file system in a forensic investigation," http://www.sans.org/reading_room/whitepapers/forensics/advantage-ext3-journaling-file-system-forensic-investigation_2011.

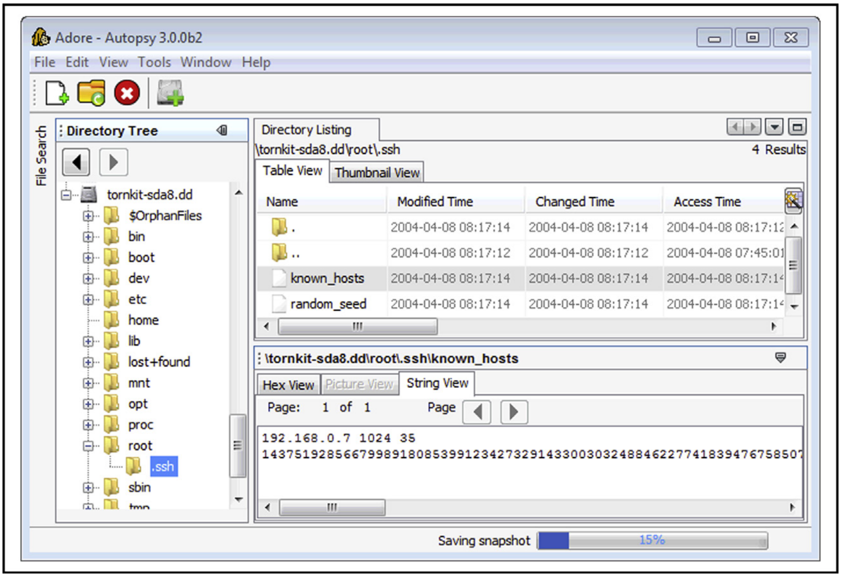


FIGURE 3.16–SSH usage remnants in `known_hosts` for the root account viewed using The Sleuth Kit

Investigative Considerations

- Given the variety of applications that can be used on Linux systems, it is not feasible to create a comprehensive list of application traces. An effective approach to finding other application traces is to search for application files created or modified around the time of the malware incident.

KEYWORD SEARCHING

☑ ***Search for distinctive keywords each time such an item is uncovered during forensic analysis.***

▶ Searching for keywords is effective when you know what you are looking for but do not know where to find it on the compromised system. There are certain features of a malware incident that are sufficiently distinctive to warrant a broad search of the system for related information. Such distinctive items include:

- **Malware Characteristics**: Names of tools that are commonly used by intruders and strings that are associated with known malware can be used as keywords (e.g., trojan, hack, sniff). Some of the rootkit scanning tools have file names that are commonly associated with known malware but only searches for these in active files, not in unallocated space. Some rootkits have their own configuration files that specify what will be hidden, including process names and IP addresses. Such configuration files can provide keywords that are useful for finding other malicious files or activities on the compromised system and in network traffic. Searching a compromised system for strings associated with malware can help find files that are related to the incident as shown in Figures 3.17 and 3.18 for the Adore rootkit.

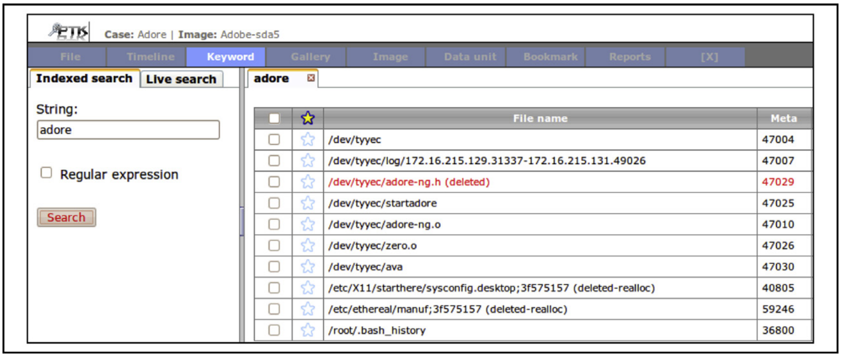


FIGURE 3.17–Keyword searching for the string “adore” using PTK indexed search[13]

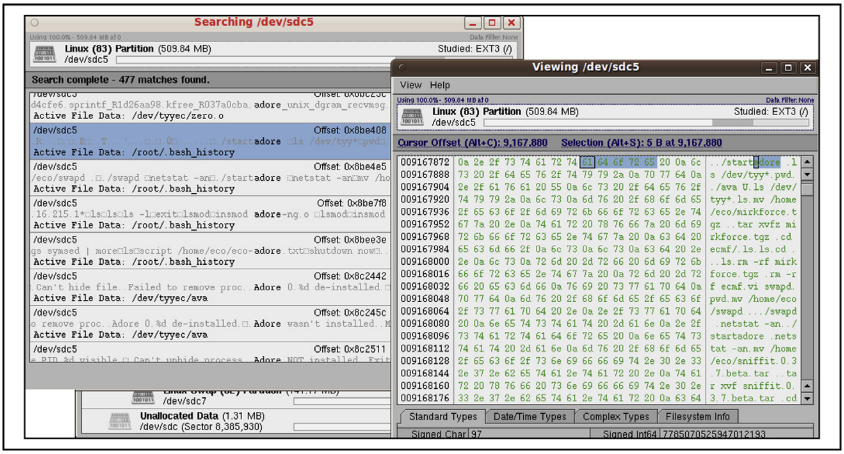


FIGURE 3.18–Keyword searching for the string “adore” using SMART forensic tool[14]

13 www.dflabs.com.
14 www.asrdata.com.

- **Command-Line Arguments**: Looking for commands that malware use to execute processes on or obtain information from other systems on the network or to exfiltrate data can reveal additional information related to the intrusion (e.g., `openvpn`, `vncviewer`).
- **IP Addresses**: IP addresses may be stored in the human readable dot decimal format (e.g., 172.16.157.136) in both ASCII and Unicode formats, and can be represented in hex (e.g., `ac 10 9d 88`) both in little and big endian formats. Therefore, it might be necessary to construct multiple keywords for a single IP address.
- **URLs**: Use of standard character encoding in URLs such as %20 for space and %2E for a "." can impact keyword searching. Therefore it might be necessary to construct multiple keywords for a single URL.
- **Hostnames**: Hostnames of computers used to establish remote connections with a compromised system may be found in various locations, including system logs.
- **Passphrases**: Searching for passphrases and encryption keys associated with malicious code can uncover additional information related to malware.
- **File Characteristics**: File extensions and headers of file types commonly used to steal data (e.g., .asc, .rar, .7z) can find evidence of data theft.
- **Date-Time Stamps**: System logs that have been deleted during a malware incident may still exist in unallocated space. Using the date-time stamp formats that are common in system logs, it is possible to search unallocated space for deleted log entries with date-time stamps around the period of the malware incidents. The command in Figure 3.19 searches unallocated space of a forensic duplicate for any entry dated November 13, and prints the byte offset for each matching line.

```
# blkls -A /evidence/phalanx2.dd | strings –t d | grep "Nov 13"
```

FIGURE 3.19–Salvaging deleted log entries dated Nov 13 by searching for strings in unallocated space that is extracted from a forensic duplicate using the `blkls` utility from The Sleuth Kit

Analysis Tip

Search Smart

The use of partitions in Linux to group different types of data can make keyword searching more effective. For instance, rather than scouring the entire hard drive, digital investigators may be able to recover all deleted log entries by simply searching the partition that contains log files.

FORENSIC RECONSTRUCTION OF COMPROMISED LINUX SYSTEMS

☑ ***Performing a comprehensive forensic reconstruction can provide digital investigators with a detailed understanding of the malware incident.***

▶ Although it may seem counterintuitive to start creating a time line before beginning a forensic examination, there is a strong rationale for this practice. Performing temporal analysis of available information related to a malware incident should be treated as an analytical tool, not just a byproduct of a forensic examination. Even the simple act of developing a time line of events can reveal the method of infection and subsequent malicious actions on the system. Therefore, as each trace of malware is uncovered, any temporal information should be inserted into a time line until the analyst has a comprehensive reconstruction of what occurred. When multiple digital investigators are examining available data sources, it is important to combine everyone's findings into a shared time line in order to obtain visibility of the overall incident.

▶ Interacting with malware in its native environment can be useful for developing a better understanding of how the malware functions. Functional analysis of a compromised Linux system involves creating a bootable clone of the system and examining it in action.

- One approach to creating a bootable clone is using Live View. The snapshot feature in VMWare gives digital investigators a great degree of latitude for dynamic analysis on the actual victim clone image. Another approach to performing functional reconstruction is to restore a forensic duplicate onto a hard drive and insert the restored drive into the original hardware. This is necessary when malware detects that it is running in a virtualized environment and take evasive action to thwart forensic examination. Some malware may look for characteristics that are specific to the compromised system such as the network interface address (MAC). Therefore, using a forensic duplicate/clone may be necessary depending on the sophistication of the malware.
- As an example of the usefulness of functional analysis, consider a system compromised with the Adore rootkit. In this instance, the malware was found in the `/dev/tyyec` directory, which was hidden (not visible on the live system) but was observed during forensic analysis, and the digital investigator used a bootable clone of the compromised system to observe the functionality of two associated utilities as shown in Figure 3.20. Changing the directory into the hidden directory and typing `ls` reveals components of the Adore rootkit files. Running the main Adore program displays the usage, including an uninstall option.

```
# cd /dev/tyyec
# ls
adore-ng.o  ava  cleaner.o  log  relink  startadore  swapd
symsed  zero.o
```

```
# ./ava

Usage: ./ava {h,u,r,R,i,v,U} [file or PID]

       I print info (secret UID etc)
       h hide file
       u unhide file
       r execute as root
       R remove PID forever
       U uninstall adore
       i make PID invisible
       v make PID visible

# ./ava U
Checking for adore  0.12 or higher ...
Adore 0.41 de-installed.
Adore 1.41 installed. Good luck.
```

FIGURE 3.20–Performing functional analysis of Adore rootkit on forensic duplicate loaded into VMWare using Live View

- After uninstalling the Adore rootkit from the resuscitated subject system, the port 31337 that was previously hidden is now visible and clearly associated with the "klogd" process as shown in Figure 3.21.

```
# netstat -anp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
State       PID/Program name
tcp        0      0 0.0.0.0:32768           0.0.0.0:*
LISTEN      561/rpc.statd
tcp        0      0 127.0.0.1:32769         0.0.0.0:*
LISTEN      694/xinetd
tcp        0      0 0.0.0.0:31337           0.0.0.0:*
LISTEN      5961/klogd -x
tcp        0      0 0.0.0.0:111             0.0.0.0:*
LISTEN      542/portmap
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      680/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*
LISTEN      717/sendmail: accep
udp        0      0 0.0.0.0:32768           0.0.0.0:*
561/rpc.statd
udp        0      0 0.0.0.0:68              0.0.0.0:*
468/dhclient
udp        0      0 0.0.0.0:111             0.0.0.0:*
542/portmap
```

FIGURE 3.21–Previously hidden port 31337 revealed during functional analysis of the Adore rootkit on a resuscitated subject system

- Furthermore, a process named "grepp" that was not previously visible, is now displayed in the ps output as shown in Figure 3.22.

```
# /media/cdrom/Linux-IR/ps auxeww | grep grepp
root       5772  0.0  0.2  1684  552 ?        S    17:31   0:01 grepp -t
172.16.@ PATH=/usr/bin:/bin:/usr/sbin:/sbin PWD=/dev/tyyec/log SHLVL=1
_=/usr/bin/grepp OLDPWD=/dev/tyyec
```

FIGURE 3.22–Previously hidden process grepp revealed during functional analysis of the Adore rootkit on a resuscitated subject system

Investigative Considerations

- In some situations, malware defense mechanisms may utilize characteristics of the hardware on a compromised computer such as MAC address, in which case it may be necessary to use a clone hard drive in the exact hardware of the compromised system from which the forensic duplicate was obtained.

ADVANCED MALWARE DISCOVERY AND EXTRACTION FROM A LINUX SYSTEM

☑ ***Perform targeted remote scan of all hosts on the network for specific indicators of the malware.***

- Since the *Malware Forensics* textbook was published in 2008, more tools have been developed to address the increasing problem of malware designed to circumvent information security best practices and propagate within a network, enabling criminals to steal data from corporations and individuals despite intrusion detection systems and firewalls.
- Some tools, such as the OSSEC Rootcheck,[15] can be used to check every computer that is managed by an organization for specific features of malware and report the scan results to a central location. When dealing with malware that is not covered by the OSSEC default configuration, this tool can be configured to look for specific files or strings known to be associated with malware. Even when searching for specific malware, it can be informative to include all default OSSEC Rootcheck configuration options, finding malware that was not the focus of the investigation.
- Other COTS remote forensic tools such as EnCase Enterprise, F-Response, FTK Enterprise, and SecondLook can be configured to examine files and/or memory on remote systems for characteristics related to specific malware. For example, the SecondLook Enterprise Edition can be used to scan a remote system that is configured to run the agent and `pmad.ko` modules using the command line (`secondlook-cli -t secondlook@compromisedserver.orgx.net info`) or via the GUI as shown in

[15] http://www.ossec.net/en/rootcheck.html.

FIGURE 3.23–Detecting the jynx2 rootkit on a Linux system using SecondLook

Figure 3.23. Additional coverage of memory analysis techniques and tools, including SecondLook, are covered in Chapter 2.

- In addition, some groups that specialize in intrusion investigation have developed customized tools to examine remote systems for traces of malicious code. For instance, it is sometimes possible to use information obtained from the malware analysis process discussed in Chapter 5 to develop a network-based scanner that "knocks on the door" of remote systems on a network in order to determine whether the specific rootkit is present.

CONCLUSIONS

- If malware is present on a system, it can be found by applying the forensic examination approach outlined in this chapter. Following such a methodical, documented approach will uncover the majority of trace evidence relating to malware incident and has the added benefit of being repeatable each time a forensic examination is performed. By conducting each forensic examination in a consistent manner, documenting each step along the way, digital investigators will be in a better position when their work is evaluated by other practitioners or in a court of law.

- As more trace evidence is found on a compromised system, it can be combined to create a temporal, functional, and relational reconstruction of the malware incident. In addition, information recovered from compromised hosts can be correlated with network-level logs and memory, as well as the malicious code itself, to obtain a more comprehensive picture of the malware incident.
- Use characteristics extracted from one compromised host to search other systems on the network for similar traces of compromise.

Pitfalls to Avoid

Stepping in Evidence

⃠ Do not perform the steps outlined in this chapter on the original system.

- ☑ Create a forensic duplicate of the hard drive from the original system and perform all analysis on a working copy of this data. In this way, no alterations are made to the original evidence during the forensic examination.
- ☑ Make working copies of the forensic duplicate to ensure that any corruption or problems that arise during a forensic examination does not ruin the only copy of the forensic duplicate.

Missed or Forgotten Evidence

⃠ Do not skip a step in the forensic examination process for the sake of expediency.

- ☑ Make an investigative plan, and then follow it. This will ensure that you include all necessary procedures.
- ☑ Be methodical, reviewing each area of the system that may contain trace evidence of malware.
- ☑ Document what you find as you perform your work so that it is not lost or forgotten later. Waiting to complete documentation later generally leads to failure because details are missed or forgotten in the fast pace of an investigation.
- ☑ Combine information from all available data sources into a shared time line of events related to the incident.

Failure to Incorporate Relevant Information from Other Sources

⃠ Do not assume that you have full information about the incident or that a single person performed the initial incident review and response.

- ☑ Determine all of the people who performed field interviews, volatile data preservation, and log analysis, and obtain any information they gathered. Incorporate such information into the overall time line that represents the entire incident.
- ☑ Review documentation such as the Field Interview notes for information that can help focus and direct the forensic examination. If a particular individual did not maintain documentation of their work and findings, speak with them to obtain details.

This page intentionally left blank

FIELD NOTES: LINUX SYSTEM EXAMINATIONS

Note: This document is not intended as a checklist, but rather as a guide to increase consistency of forensic examination of compromised Linux systems. When dealing with multiple compromised computer systems, it may be necessary to tabulate the results of each individual examination into a single document or spreadsheet.

Case Number:		**Date/Time:**	
Examiner Name:		**Client Name:**	
Organization/Company:		**Address:**	
Incident Type:	❒Trojan Horse ❒Bot ❒Logic Bomb ❒Sniffer	❒Worm ❒Scareware/Rogue AV ❒Keylogger ❒Other:	❒Virus ❒Rootkit ❒Ransomware ❒Unknown
System Information:		**Make/Model:**	
Operating System:	**Forensic Duplication Method:** ❍Postmortem acquisition ❍Live console acquisition ❍Live remote acquisition	**Network State:** ❍Connected to Internet ❍Connected to Intranet ❍Disconnected	
Role of System: ❑Workstation: ❑Web Server:	 ❑Credit Card Processing System: ❑Other:		
FORENSIC DUPLICATE			
Physical Hard Drive Acquisition: ❑Acquired ❑Date/Time : ❑File Name: ❑Size: ❑MD5 Value: ❑SHA1 Value: ❑Tool used:	 ❑Not Acquired [Reason]:		

KNOWN MALWARE:

Note: AntiVirus software may quarantine known malware in a compressed/encoded format.

❑**File/Folder Identified:**

❍Method of identification (e.g., Hashset, AntiVirus):

____

❒File Name:
❒Inode Change/Birth date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑**File/Folder Identified:**

❍Method of identification (e.g., Hashset, AntiVirus):

____

❒File Name:
❒Inode Change/Birth date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑**File/Folder Identified:**

❍Method of identification (e.g., Hashset, AntiVirus):

____

❒File Name:
❒Inode Change/Birth date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

SUSPICIOUS INSTALLED PROGRAMS:

❑**Application name and description:**

____

❍Software installation path:

❑**Application name and description:**

____

❍Software installation path:

SUSPICIOUS E-MAILS AND ATTACHMENTS:

❑**E-mail:**
❍Sender address:
❍Originating IP:
❍Attachment name:
❍Attachment description:
____
____

❑**E-mail:**
❍Sender address:
❍Originating IP:
❍Attachment name:
❍Attachment description:
____
____

SUSPECT EXECUTABLE FILES:

❑**File/Directory Identified:**

❍Method of identification (e.g., stripped, unique string):

____

❒File Name:
❒Inode Change/Birth date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑**File/Directory Identified:**

❍Method of identification (e.g., stripped, unique string):

____

❒File Name:
❒Inode Change/Birth date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑**File/ Directory Identified:**

❍Method of identification (e.g., stripped, unique string):

____

❒File Name:
❒Inode Change/Birth date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

MALICIOUS AUTO-STARTS:

❑**Auto-start description:**

____

____

❍Auto-start location:

❑**Auto-start description:**

____

____

❍Auto-start location:

QUESTIONABLE USER ACCOUNTS:

❑**User account ________________ on the system:**
❍Date of account creation:
❍Login date
❍Shares, files, or other resources accessed by the user account:
❍Processes associated with the user account:
❍Network activity attributable to the user account:
❍Passphrases associated with the user account:

❑**User account ________________ on the system:**
❍Date of account creation:
❍Login date
❍Shares, files, or other resources accessed by the user account:
❍Processes associated with the user account:
❍Network activity attributable to the user account:
❍Passphrases associated with the user account:

SCHEDULED TASKS:

❑**Scheduled Tasks Examined**
❑**Tasks Scheduled on the System**
❍Yes
❍No
❑**Suspicious Task(s) Identified:**
❍Yes
❍No

❑**Suspicious Task(s)**
❍Task Name:
❐Scheduled Run Time:
❐Status:
❐Description:
❍Task Name:
❐Scheduled Run Time:
❐Status:
❐Description:

SUSPICIOUS SERVICES:

❑**Services Examined**
❑**Suspicious Services(s) Identified:**
❍Yes
❍No
❑**Suspicious Service Identified:**
❍Service Name:
❐ Associated executable path:
❐ Associated startup script date-time stamps:
❑**Suspicious Service Identified:**
❍Service Name:
❐ Associated executable path:
❐ Associated startup script date-time stamps:

FILE SYSTEM CLUES

Artifacts to Look for on Storage Media:

Notes:

FILE SYSTEM ENTRIES:

❑**File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒Creation Date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑ **File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒Creation Date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑**File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒Creation Date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑**File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒ Creation Date-time stamp:
❒ File location on system (path):
❒File location on system (clusters):

❑**File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒Creation Date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑**File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒Creation Date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑**File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒Creation Date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑**File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒Creation Date-time stamp:
❒Handle Value:
❒File location on system:

❑**File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒Creation Date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

❑ **File/Directory Identified:**
❍Opened Remotely/❍Opened Locally
❒File Name:
❒Creation Date-time stamp:
❒File location on system (path):
❒File location on system (clusters):

HOST-BASED LOGS

AntiVirus Logs:

❑**AntiVirus Type:**
❑**AntiVirus log location:**
❑**AntiVirus log entry description:**

____

❍Detection date:
❍File name:
❍Malware name:
❍AntiVirus action:

❑**AntiVirus log entry description:**

____

❍Detection date:
❍File name:
❍Malware name:
❍AntiVirus action:

❑**AntiVirus log entry description:**

____

❍Detection date:
❍File name:
❍Malware name:
❍AntiVirus action:

LINUX SYSTEM LOGS:

❑**Log Entry Identified:**
❍Security/❍System/❍ Other ______________
❐Event type:
❐Source:
❐Creation Date-time stamp:
❐Associated account/computer:
❐Description:

❑**Log Entry Identified:**
❍Security/❍System/❍Other ______________
❐Event type:
❐Source:
❐Creation Date-time stamp:
❐Associated account/computer:
❐Description:

❑**Log Entry Identified:**
❍Security/❍System/❍Other ______________
❐Event type:
❐Source:
❐Creation Date-time stamp:
❐Associated account/computer:
❐Description:

❑**Log Entry Identified:**
❍Security/❍System/❍Other ______________
❐Event type:
❐ Source:
❐ Creation Date-time stamp:
❐ Associated account/computer:
❐ Description:

❑ **Log Entry Identified:**
❍Security/❍System/❍ \Other ______________
❐Event type:
❐Source:
❐Creation Date-time stamp:
❐Associated account/computer:
❐Description:

❑**Log Entry Identified:**
❍Security/❍System/❍Other ______________
❐Event type:
❐Source:
❐Creation Date-time stamp:
❐Associated account/computer:
❐Description:

❑**Log Entry Identified:**
❍Security/❍System/❍Other ______________
❐Event type:
❐Source:
❐Creation Date-time stamp:
❐Associated account/computer:
❐Description:

❑**Log Entry Identified:**
❍Security/❍System/❍Other ______________
❐Event type:
❐Source:
❐Creation Date-time stamp:
❐Associated account/computer:
❐Description:

❑**Log Entry Identified:**
❍Security/❍System/❍Other ______________
❐Event type:
❐Source:
❐Creation Date-time stamp:
❐Associated account/computer:
❐Description:

❑**Log Entry Identified:**
❍Security/❍System/❍Other ______________
❐Event type:
❐Source:
❐Creation Date-time stamp:
❐Associated account/computer:
❐Description:

WEB BROWSER HISTORY:

❑**Suspicious Web Site Identified:**
❍Name:
❐URL:
❐Last Visited Date-time stamp:
❐Description:

❑**Suspicious Web Site Identified:**
❍ Name:
❐URL:
❐Last Visited Date-time stamp:
❐Description

❑**Suspicious Web Site Identified:**
❍Name:
❐URL:
❐Last Visited Date-time stamp:
❐Description:

❑**Suspicious Web Site Identified:**
❍Name:
❐URL:
❐Last Visited Date-time stamp:
❐Description:

HOST-BASED FIREWALL LOGS:

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________________
❍Protocol:
❐TCP
❐UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________________
❍Protocol:
❐TCP
❐UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________________
❍Protocol:
❐TCP
❐UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________________
❍Protocol:
❐TCP
❐UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________________
❍Protocol:
❐TCP
❐UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________________
❍Protocol:
❐TCP
❐UDP

CRASH DUMP LOGS:

❑**Crash dump:**
❍File name:
❍Creation date-time stamp:
❍File location on system (path):
❍File location on system (cluster):
❒Description:
____

❑**Crash dump:**
❍File name:
❍Creation date-time stamp:
❍File location on system (path):
❍File location on system (cluster):
❒Description:
____

NETWORK CLUES

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________
❍Protocol:
❒TCP
❒UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________
❍Protocol:
❒TCP
❒UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________
❍Protocol:
❒TCP
❒UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________
❍Protocol:
❒TCP
❒UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________
❍Protocol:
❒TCP
❒UDP

❑**IP Address Found:**
❍Local IP Address: ___.___.___.___ Port Number: ____
❍Remote IP Address: ___.___.___.___Port Number: ___
❍Remote Host Name:_____________________
❍Protocol:
❒TCP
❒UDP

WEB SITE/URLS/E-MAIL ADDRESSES:

❑**Suspicious Web Site/URL/E-mail Identified:**
❍Name:
❒Description

❑ **Suspicious Web Site/URL/E-mail Identified:**
❍Name:
❒Description

❑**Suspicious Web Site/URL/E-mail Identified:**
❍Name:
❒Description:

❑**Suspicious Web Site/URL/E-mail Identified:**
❍Name:
❒Description:

LINKAGE TO OTHER COMPROMISED SYSTEMS:

❑**Association with other compromised system:**
❍IP address:
❍Name:
❒Description

❑**Association with other compromised system:**
❍IP address:
❍Name:
❒Description

❑**Association with other compromised system:**
❍IP address:
❍Name:
❒Description:

❑**Association with other compromised system:**
❍IP address:
❍Name:
❒Description:

SEARCH FOR KEYWORDS/ARTIFACTS	
Keyword Search Results:	
❑**Keyword:**	❑**Keyword:**
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❑**Keyword:**	❑**Keyword:**
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❑**Keyword:**	❑**Keyword:**
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____
❍Search hit description: _____________ Location: ____	❍Search hit description: _____________ Location: ____

This page intentionally left blank

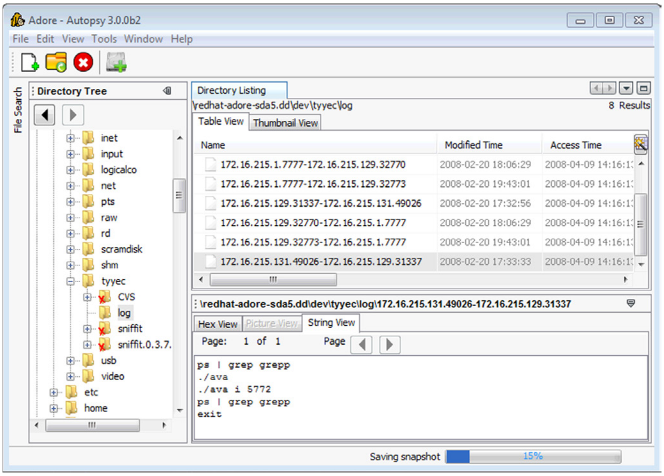
Malware Forensic Tool Box

Forensic Examination Tools for Linux Systems

In this chapter we discussed approaches to interpreting data structures in memory on Linux systems. There are a number of forensic analysis tools that you should be aware of and familiar with. In this section, we explore these tool alternatives, often demonstrating their functionality. This section can also simply be used as a "tool quick reference" or "cheat sheet" as there will inevitably be an instance during an investigation where having an additional tool that is useful for a particular function would be beneficial, but while responding in the field you will have little time to conduct research for or regarding the tool(s). It is important to perform your own testing and validation of these tools to ensure that they work as expected in your environment and for your specific needs.

FORENSIC TOOL SUITES

Name: ***The Sleuth kit & Autopsy***
Author/Distributor: Brian Carrier and Open Source Collaborators
Page Reference: 43
Available From: http://www.sleuthkit.org
Description: The Sleuth kit is a free open source suite of forensic utilities that has a GUI called Autopsy. This tool suite has strong support for Linux file systems and can be used to examine the full details of inodes and other data structures. The Sleuth kit has a plugin framework that supports automated processing The Autopsy GUI for The Sleuth kit is shown herewith a Linux file system:



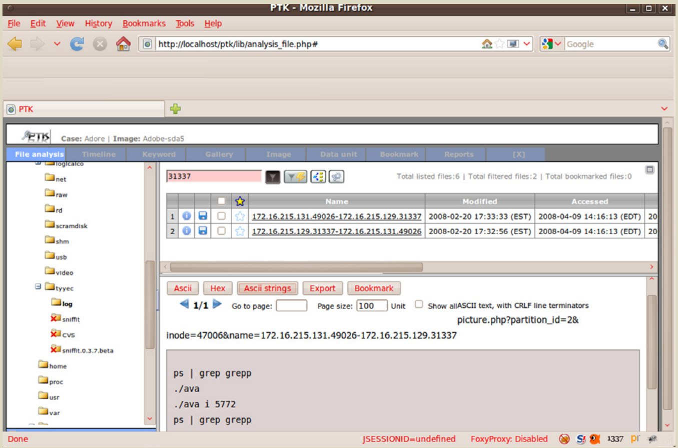
Name: ***PTK***

Page Reference: 26

Author/Distributor: DFLabs

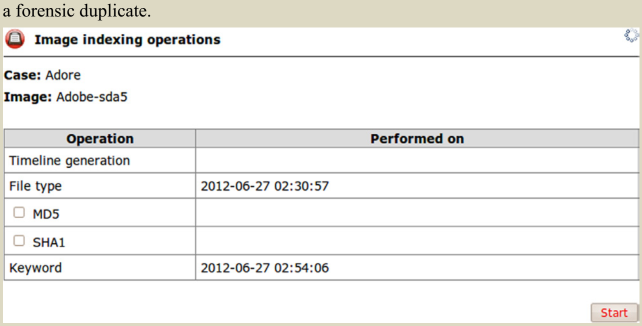
Available From: http://www.dflabs.com

Description: The PTK suite builds on The Sleuth kit framework to provide added functionality, including keyword indexing and signature matching. This tool uses a database to provide stability and flexibility, saving processing results between uses.

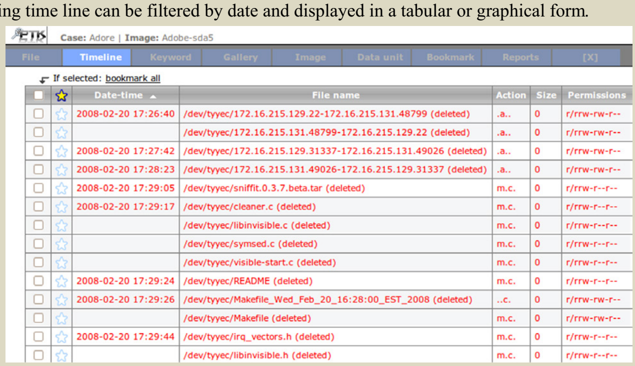


Additional Options:

PTK has options to index forensic duplicate for keyword searching, to create a file system time line, calculate file hashes, and perform signature/header analysis as shown here in the indexing operations screen for a forensic duplicate.



The resulting time line can be filtered by date and displayed in a tabular or graphical form.



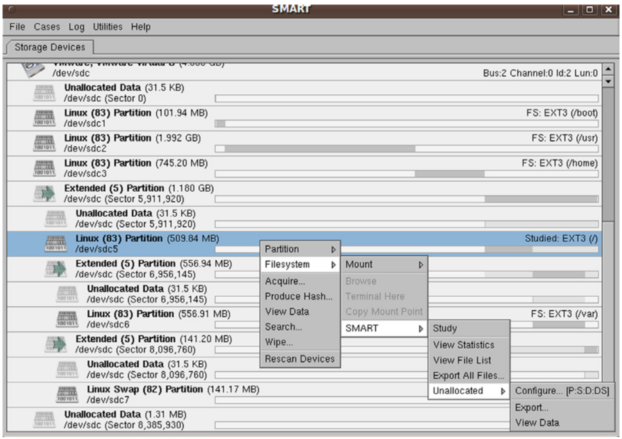
Case: Adore | Image: Adobe-sda5

File | Timeline | Keyword | Gallery | Image | Data unit | Bookmark | Reports | [X]

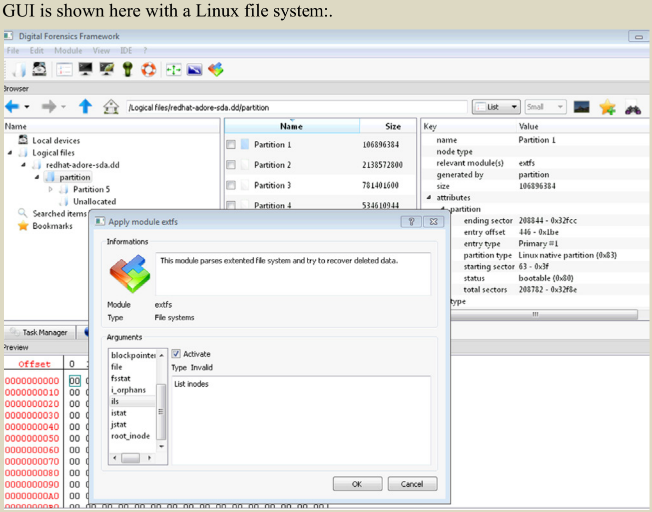
If selected: bookmark all

Date-time	File name	Action	Size	Permissions
2008-02-20 17:26:40	/dev/tyyec/172.16.215.129.22-172.16.215.131.48799 (deleted)	.a..	0	r/rrw-rw-r--
	/dev/tyyec/172.16.215.131.48799-172.16.215.129.22 (deleted)	.a..	0	r/rrw-rw-r--
2008-02-20 17:27:42	/dev/tyyec/172.16.215.129.31337-172.16.215.131.49026 (deleted)	.a..	0	r/rrw-rw-r--
2008-02-20 17:28:23	/dev/tyyec/172.16.215.131.49026-172.16.215.129.31337 (deleted)	.a..	0	r/rrw-rw-r--
2008-02-20 17:29:05	/dev/tyyec/sniffit.0.3.7.beta.tar (deleted)	m.c.	0	r/rrw-r--r--
2008-02-20 17:29:17	/dev/tyyec/cleaner.c (deleted)	m.c.	0	r/rrw-r--r--
	/dev/tyyec/libinvisible.c (deleted)	m.c.	0	r/rrw-r--r--
	/dev/tyyec/symsed.c (deleted)	m.c.	0	r/rrw-r--r--
	/dev/tyyec/visible-start.c (deleted)	m.c.	0	r/rrw-r--r--
2008-02-20 17:29:24	/dev/tyyec/README (deleted)	m.c.	0	r/rrw-r--r--
2008-02-20 17:29:26	/dev/tyyec/Makefile_Wed_Feb_20_16:28:00_EST_2008 (deleted)	..c.	0	r/rrw-rw-r--
	/dev/tyyec/Makefile (deleted)	m.c.	0	r/rrw-rw-r--
2008-02-20 17:29:44	/dev/tyyec/irq_vectors.h (deleted)	m.c.	0	r/rrw-r--r--
	/dev/tyyec/libinvisible.h (deleted)	m.c.	0	r/rrw-r--r--

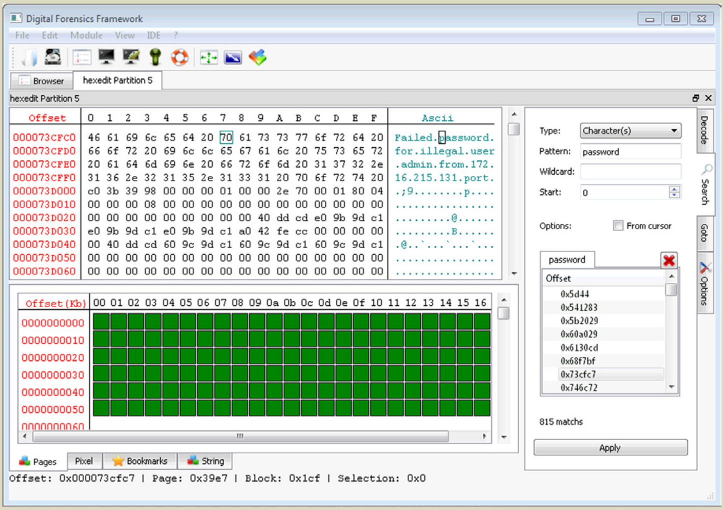
Name: ***SMART***
Page Reference: 26
Author/Distributor: ASR Data
Available From: http://www.asrdata.com
Description: The SMART tool can be used to perform an examination of a Linux file system, including browsing directories and keyword searching of active and unallocated space. This tool does not display names of recoverable deleted files that are still referenced in a Linux file system, but does provide access to unallocated space, which contains the content of deleted files. The SMART GUI is shown below with a Linux file system and several examination options.



Name: Digital Forensics Framework
Page Reference: 23
Author/Distributor: DFF
Available From: http://www.digital-forensic.org/
Description: The Digital Forensics Framework is a free open source tool that has strong support for Linux file systems. The DFF has a plugin framework that supports the development and integration of customized features. The DFF GUI is shown here with a Linux file system:.



Features and Plugins:
DFF has a variety of features, including keyword searching shown below, and uses a plugin approach to adding capabilities.



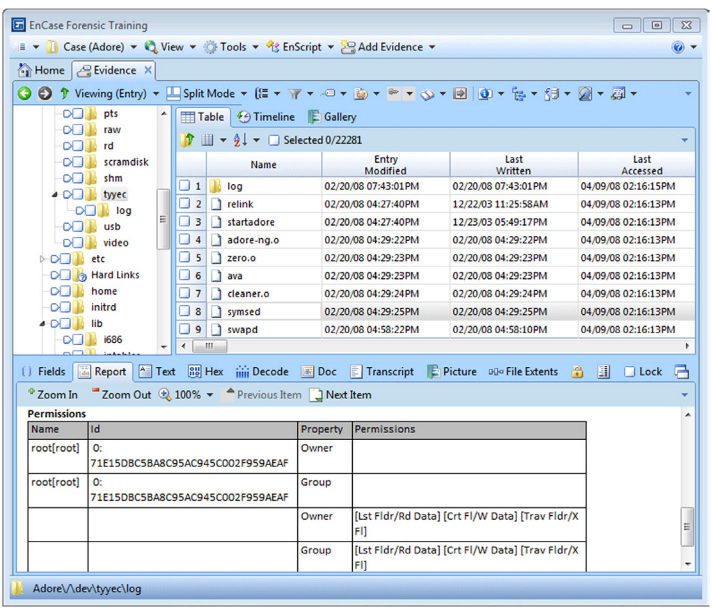
Name: ***EnCase***

Page Reference: 6

Author/Distributor: Guidance Software

Available From: http://www.guidancesoftware.com

Description: EnCase is a commercial integrated digital forensic examination program that has a wide range of features for examining forensic duplicates of storage media. This tool has limited support for Linux file systems but does not provide access to the full range of file system metadata:



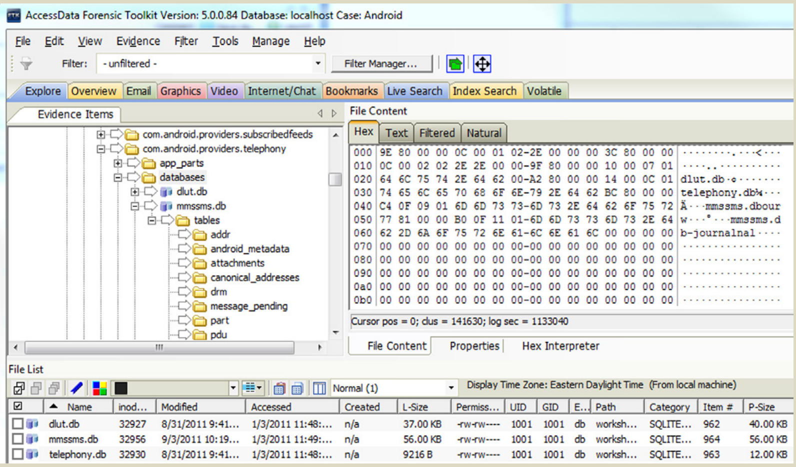
Name: ***FTK***

Page Reference: 6

Author/Distributor: AccessData

Available From: http://www.accessdata.com

Description: FTK is a commercial integrated digital forensic examination program that has a wide range of features for examining forensic duplicates of storage media. This tool has strong Linux files system support as shown in the following figure, displaying inode metadata in full detail. In addition to parsing and displaying common file systems, FTK recovers deleted files and performs indexing to facilitate keyword searching.



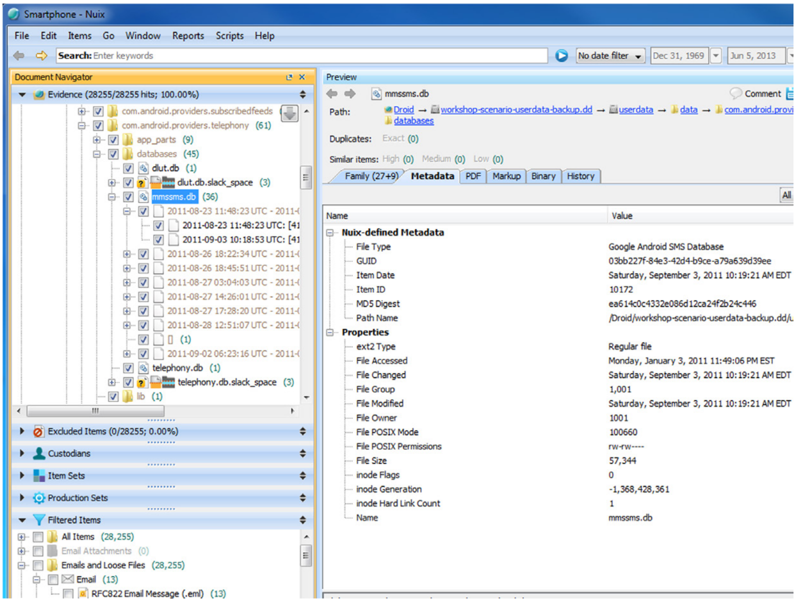
Name: ***Nuix***

Author/Distributor: Nuix

Page Reference: 6

Available From: http://www.nuix.com

Description: Nuix is a suite of commercial digital forensic programs for extracting information from forensic duplicates of storage media, categorizing content, and performing correlation. This tool has strong Linux file system support, including EXT, and Android devices as shown in the following figure, displaying detailed inode metadata. Correlation can be performed between activities on a single system, or across multiple systems to create an overall viewpoint of activities in an investigation. In addition to parsing and displaying various file formats, including e-mail and chat communications, Nuix recovers deleted file and performs indexing to facilitate keyword searching. Data extracted using Nuix can be displayed and analyzed visually using temporal information, file type, and other characteristics.



TIMELINE GENERATION

Name: ***plaso***

Page Reference: 21

Author/Distributor: Kristo Gudjonsson

Available From: https://code.google.com/p/plaso/ and http://plaso.kiddaland.net

Description: The `log2timeline` and `psort` tools are part of a free open source suite called plaso that extracts information from a variety of logs and other date-time stamps data sources and consolidates the information in a comprehensive time line for review. This tool suite can be used to process individual files or an entire mounted file system to extract information from supported file formats. For example, the following command processes a forensic duplicate of a Linux system, creating a database named "`l2timeline.db`" that can be examined using `psort` (e.g., to extract items between August 16–18, 2013 in this example), and other tools in the plaso suite:

```
% log2timeline -i -f linux -z EST5EDT l2timeline.db host1.dd
<cut for length>
% psort -o L2tcsv l2timeline.db host1.dd \
-t 2013-08-16 -T 2013-08-18 -w output.csv
```

SELECTED READINGS

Books

Altheide, C. & Carvey, H. (2011). Digital Forensics with Open Source Tools. Burlington, MA: Syngress.

Carrier, B. (2005). File System Forensic Analysis. Reading, MA: Addison-Wesley Professional.

Casey, E. (2011). Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet (3rd edition). San Diego, CA: Academic Press.

Casey, E. (2009). Handbook of Digital Forensics and Investigation. San Diego, CA: Academic Press.

Papers

An analysis of Ext4 for digital forensics DFRWS2012 Conference Proceedings. Retrieved from, http://www.dfrws.org/2012/proceedings/DFRWS2012-13.pdf.

Eckstein, K. (2004). Forensics for advanced Unix file systems. In: IEEE/USMA information assurance workshop. p. 377–85.

Eckstein, K. & Jahnke M. (2005). Data hiding in journaling file systems. Digital Forensic Research Workshop (DFRWS). p. 1–8.

Swenson C, Phillips R, & Shenoi S. (2007). File system journal forensics. In: Advances in digital forensics III. IFIP international federation for information processing, vol. 242. Boston: Springer. p. 231–44.

This page intentionally left blank