11/1/2018

A Study on Linux Forensics

By: Gustavo Amarchand, Keanu

Munn, and Samantha Renicker

Abstract

In the field of computer forensics investigators must be familiar with many different systems and tools to uncover reliable evidence. For that reason, familiarity with Linux whether it be to conduct an analysis on a Linux machine or to use Linux as a platform for investigation is necessary. Linux as a platform for conducing analysis provides a variety of tools that will allow an investigator to uncover the truth. It is also important that an investigator also be aware of the necessary methods and concepts that are required to audit a Linux machine. Furthermore, adequate knowledge of the Linux file system journal is a critical factor for finding evidence on a Linux machine.

Linux Forensics & Opensource Tools

In the world of computer forensics there are a variety of tools and options available for investigators to choose from. Because the goal of a computer investigator is to uncover the truth using tools that are open source and verifiable removes possible layers of abstraction that can keep the investigator from the truth that is needed [1]. One such tool known as “dcfldd” is a modified version of the Linux dd utility, developed by the DoD computer forensics library that has a whole host of features for computer forensics professionals to take advantage of [9]. Another such tool used for the purpose of file system analysis would be the “Sleuth Kit” and “Autopsy” an open source Linux based toolset that allows investigators to easily analyze drive sectors and create hashed evidence files [2] .

In general, the main objective for a computer forensics expert is to uncover the truth in the form of evidence from suspect devices. The issue one could argue is that in using closed source programs there is often no way to verify bugfixes or issues with the current tools version as this type of information is not disclosed to the users of the tools. This could lead to skewed results or distortion in the process of evidence gathering [1]. Using an open source tool such as Sleuth Kit for example an investigator could verify the legitimacy of the program and how it operates in many different ways, one of which would be the public record of bug fixes another option would be being able to compare the older versions of the source code to the newer releases, and the third option would be verifying the actual function of the code therefor being able to present how it is the tool was able to legitimately produce the evidence [1]. In using a closed source tool, the investigator is placing layers of abstraction between themselves and the evidence and with each of these layers the potential that the evidence could be distorted or have errors could increase.

Dcfldd was a forensics-oriented version of the Unix dd utility that was developed by the Department of Defense computer forensics lab in 2002 [9]. The standard dd utility has the primary purpose of converting and copying files and is often used for backing up the boot sectors of hard drives. When talking about dcfldd the main differences that are provided by the tool vs the standard dd are hashing, hash comparisons of the source image and the created copy image, flexible naming conventions for the split image files, and the ability to direct output to multiple locations simultaneously [9]. Dcfldd is an important tool in the case of Linux system-based forensics as being able to make valid images of suspect drives is essential to conducting any form of proper forensics investigation.

Another great tool for forensics use on the Linux platform Sleuth Kit and Autopsy. Sleuth kit is a toolset used for finding evidence through analysis of file systems, comprised of tools for examining DOS, BSD, Mac partitions, Sun slices, and GPT disks [2]. Because all these tools are command line tools Autopsy was created as a graphical suit to allow for easier use of the tools. During analysis. Because a full analysis will often require more than just file and volume system analysis the modularity of the Sleuth Kit will allow for the inclusion of other tools to further search for more data. In searching through Sleuth kit tools the ability to look up file hashes through a hash database is provided [7]. As well as being able to display the details and contents of all NTFS attributes including Alternate Data Streams.

In the world of computer forensics, the tools that investigators choose to use must be able to allow confidence in the evidence that is provided. Using Linux and open source tools as a means of conducting investigations can potentially allow for more credible or provable evidence through verification of the function of those tools.

Linux Auditing & Logging

Computer forensic investigators often look to auditing systems and logs that are in place on a host machine to try to glean any useful information for their investigations. Because of this it is important for an investigator to be familiar with the concepts and methodologies that are involved in auditing and logging, especially on a Linux system. Auditing refers to the actions that one wishes to monitor on a system or application for security purposes. Logging refers to logging all activity that occurs in an application or on a system [3]. Both tools can be very useful for a forensic investigator in their endeavors to uncover evidence or attempts to gain a better understanding of what kind of crime was committed on a suspect's computer. Linux has a lot of useful logs in place that can help forensic investigators, and Linux has an interesting audit system that is installed on most distributions by default [5].

As mentioned previously, Linux by default has a plethora of useful terminal commands and logs already in place for forensic investigators to take advantage of when analyzing Linux systems. Let's start with some simple commands such as "history" and "/usr/bin/w". The command "history" shows the last commands executed by the current user logged in, and "usr/bin/w" allows to see who is currently logged in and what each user is performing [8]. These commands are a simple way to get an understanding of what might've immediately occurred on a system, but log files are much more useful for an investigator because they usually contain the exact moment when a suspicious event might've occurred. For example, the "/var/log/faillog" contains every failed user login attempt, which can be a useful log for trying to determine if a bad actor was trying to break into a system [5]. Another example is the "/var/log/lpr.log" which shows a record of all items that have been printed from the machine. This could be a great tool for corporate espionage cases and providing additional evidence in a court of law to solidify the claim of what information was stolen. These are just some of the basic logs and commands that can be performed; however, it is important to realize that logs can easily be compromised and edited to hide the tracks of hackers. Which is why it is important to have an advanced auditing system in place.

Many Linux distributions have installed by default the Linux Audit system which is a great tool for tracking security related events.

> The Audit system consists of two main parts: the user-space applications and utilities, and the kernel-side system call processing. The kernel component receives system calls from user-space applications and filters them through one of the three filters: user, task, or exit. Once a system call passes through one of these filters, it is sent through the exclude filter, which, based on the Audit rule configuration, sends it to the Audit daemon for further processing [6].

The Audit system was designed with security in mind and enables a user to audit and record a lot of useful information in it's log files. For example, in a typical Audit system log file one might find: the date/time and type of event along with the outcome of said event, sensitivity labels of subjects and objects, identity of the user who triggered event, changes to any trusted database like "/etc/passwd", and modifications to Audit configuration and attempts to access Audit log files [6]. This is not an exhausted list of what the Audit system is capable of, but it shows just how much useful information can be ascertained from the Linux Audit system.

Auditing and logging are valuable tools of information for forensic investigators. They contain information about all events that occur on a system and the user who triggered said events. Linux is aware of this and has provided multiple tools to provide efficient and secure auditing/log solutions.

Linux File System Journal Forensics

File system journals are used to cache data that reside on the disk before it is sent to the file system. This is done to ensure nothing is lost if there is power loss or a system malfunction; however, if an event like that occurs, the journal can be replayed to allow for write operations to be completed. The journal could be analyzed and used to recover overwritten or deleted data files because previous file writes are typically stored in the journal for a limited time. After the ext2 version of Linux, journaling has been implemented into the more recent file systems such as ReiserFS, ext3, and ext4. I will be focusing solely on the journaling forensics of the ext3 version file system. The ext3 file system is the third installment of the Linux extension file systems and the first to implement the journaling feature. Ext3 uses the superblock structure, so the size of the journal just depends on the size of the file system. Each entry that is logged into the journal gets put in a rotational queue, all entries have a commit block and a descriptor block.

When changes are made to files within the file system, they are logged into the journal and these changes can be discovered by using a data recovery tool. When using the default journal settings in ext3, the only changes to the file system that are recorded in the journal are the ones made to the metadata. This means that all the metadata is stored in the journal when editing a file. Rather than just metadata being logged, all data can be logged by using the mount command option, data=journal. A Java-based data recovery tool can be used to analyze journal entries and look for modified, deleted, or overwritten files. The data recovery tool uses a FileInputStream object to open a file system that is passed as a command line argument and it reads the file byte by byte. Upon opening the file, information is extracted from the superblock and stored for future use [4]. Depending on the entries that are logged into the journal inode, an array of block objects holds those entries for all the journal blocks that are on the system. The tool can then search the superblock to find out what type of block each entry is: commit, descriptor, or data. After figuring out the block type, the tool can recognize what kind of entry was made (such as a modification), it can see the address of it, the order of how the file was manipulated, and if there is a copy of it. Once the tool has checked for edits that were made, it looks in each journal block for information that may have been deleted, so it can be restored. There are two conditions that a block can satisfy for information to be restored: being a filled data block that is not being used anymore or when information that is in the journal version of the block is different from the current version being used. If one of these are met that means a deleted file was discovered and, the recovery tool can locate information on the block including, the file system address, the block of which it was an older version and which condition it met, along with the contents of the block [4].

The implementation of journals into the ext3 version of Linux was a great way to help keep data stored all in one place. Along with all of the data being stored in one place, that makes using a data recovery tool much easier to use with journals. Data recovery tools can be used to search journal blocks for modifications, deletions, and overwritten files. The tool takes many steps to determine what type of data is recovered from each block and how it is recovered based on the type of edit that was made to a file in the journal block.

References

1. Altheide, C., & Carvey, H. A. (2011). *Digital forensics with open source tools*. Amsterdam: Syngress.
2. Carrier, B. (2015). *File system forensic analysis*. Upper Saddle River, NJ: Addison-Wesely.
3. Clarke, G. (2018). *CompTIA Security + Certification Study Guide* . New York: McGraw-Hill Education.
4. Craiger, P., & Shenoi, S. (2007). Advances in Digital Forensics III. Retrieved November 1, 2018, from https://link.springer.com/content/pdf/10.1007/978-0-387-73742-3_16.pdf Pages 231-242.
5. Easttom, C. (2018). *System Forensics, Investigation, and Response.* Burlington: Jones & Bartlett Learning.
6. Jahoda, M., Krátký, R., Prpič, M., Čapek, T., Wadeley, S., Ruseva, Y., & Svoboda, M. (2018). *Chapter 7. System Auditing*. Retrieved from Redhat Customer Portal: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/chap-system_auditing
7. Nikkel, B. J. (2009). Forensic analysis of GPT disks and GUID partition tables. Digital Investigation, 6(1-2), 39-47. doi:10.1016/j.diin.2009.07.001
8. Wazi, W. (2012, November 27). *Linux Auditing 101*. Retrieved from Network World:https://www.networkworld.com/article/2224092/opensource-subnet/linux-auditing-101.html
9. Tutorial: How to use dcfldd instead of dd. (2014, November 03). Retrieved October 20, 2018, from https://stefanoprenna.com/blog/2014/03/02/tutorial-how-to-use-dcfldd-instead-of-dd/