Linux Security

Gustavo Amarchand, Patrick Brown, Thomas Mahoney

Saint Leo University

Abstract

Linux is an open source Operating system that is for the most part freely available to the public. Due to its customizability and cost to performance benefits Linux has quickly been adopted by users and companies alike for use in applications such as servers and workstation. As the spread of Linux continues it is important for security specialists to understand the platform and the security issues that affect the platform as well. This paper seeks to first educate the users on what the Linux platforms is and what it offers to the user or company. And it then will expand upon some common or recent vulnerabilities that Linux faces due to the way it functions. After explaining some exploits, the paper will then seek to explain some hardening solutions that are available on the platform.

Why Linux?

To first understand why Linux has a vast amount of security flaws, that can scare new users away, we must first understand what Linux is, and why it's so popular today. Linux was created by, then college student, Linus Torvalds. His main motive for the creation of Linux revolved around his dissatisfaction with MS-DOS, and the high prices of UNIX back in 1991 (What is Linux?, p. 1). Linux is an open source software, that being said, it can be used and changed by users at their will (Brooke, 2000, p. 1). The interesting thing about Linux is that, when enough changes are done to the source code by one corporation/individual they then have their own distribution of Linux, which is completely unique compared to others. Now, when we get into specifics, such as fine prints, we can see where Linux really does shine compared to Windows and Microsoft. According to Linus Torvalds, "Linux's key is freedom--freedom from 'Microsoft's control and fine print'" (Brooke, 2000, p. 1). With that being said, clearly Linux holds users data more valuable compared to Microsoft, because of the lack of corporate involvement.

Now, let's dive right into some of the claims as to why Linux can and cannot be considered secure. Both of the following claims stem from the same thought process, that being that since Linux is a open-source project, anyone can look at the code and identify flaws quickly. However, since everyone can view the source code, others claim that hackers have an easier time at finding exploits and attacks (Spafford & Wilson, 2004, p. 1). However, to debunk these claims, the article goes on to state that, "Furthermore, careful analysis leads to the conclusion that security is unrelated to whether the software is proprietary or open source" (Spafford & Wilson, 2004, p. 1). Now with this being said, all operating systems are vulnerable to some attack in one way or another. To pick Linux and Windows, as just two examples, both see a wide

variety of attacks and exploits that are used everyday. For example, Windows does have a higher and more common risk of being attacked by worms and viruses, while Linux has a much higher chance of being attacked by a “root kit”, both of which can be found online (Spafford & Wilson, 2004, p. 1).

Ultimately when it comes down to the common argument of “Linux versus Windows”, you would always come out with the outcome that it greatly depends on what the user is looking to accomplish, and how the user wants to go about doing it. Because, according to Spafford & Wilson, neither open-source nor proprietary paradigms offer any kind of silver bullet for security and quality (2004, p. 1). Since both can experience just as many exploits and attacks as the other, it really comes down to user needs, not security, and by no means is Linux the most secure operating system, however neither is Windows.

Vulnerabilities in Linux

The use of Linux based systems is ever growing, used as personal computers, servers, and mobile operating system it would be difficult to interact with modern technology and services without using Linux at some point in the exchange. While there are benefits to using a Linux based system as a server or workstation there still lies the possibility that malicious entities will target these systems for monetary and personal gain. So, it is important to be aware of exploits or vulnerabilities that have been found in the Linux kernel in order to better secure and protect the various devices that may be running the Operating system.

In order to fully understand the exploits being covered, it is necessary to go over what a stack clash pertains to. Each program running on a machine uses a region of memory referred to as a stack, as the programs needs for memory increase the stack will grow automatically in order to meet the need of the program (Law, 2017). In the case of a stack clash when the memory

region grows too large and gets too close to another region in memory the program can become confused as to which region belongs to it leaving it vulnerable to an attacker taking advantage of this and overwriting the stack with the other memory region or vice versa (Law, 2017). Using this method an attacker can cause the execution of malicious code depending on what has been written to the stack. Because of the underlying architecture of many Unix based systems this issue can be found on most Linux distributions, BSD distributions, and Solaris (Law, 2017).

In the case of the exploit System Down discovered on Jan 9th, 2019 by Qualys vulnerability and Malware Research Lab. The System Down vulnerability is comprised of three separate exploits CVE-2018-16864, CVE-2018-16865, and CVE-2018-16866. In this case, the exploits CVE-2018-16864, CVE-2018-16865, and CVE-2018-16866 take advantage of memory leaks and a form of stack clashing (Qualys Security Advisory, 2019). These exploits target journald which is a part of systemd which is an essential initialization system that runs on nearly all Linux based distributions that are currently in use (Linode, 2018). Systemd is a very important tool that is used to not only initialize a Linux system but also interfaces with the kernel directly (Linode, 2018). “In using the System Down exploit an attacker can gain a full root level shell in 10 minutes on a 32-bit system and 70 minutes on a 64bit system” (Qualys Security Advisory, 2019, System Down: A systemd-journald exploit, Summary). This can be performed by causing a memory leak by exploiting the journald utility of systemd when this is done it allows the attacker to write instructions directly into memory.

The exploit by Qualis known as CVE-2018-16865 is known to work reliably on both 64-bit machines and 32-bit machines (Qualys Security Advisory, 2019). Like its predecessor CVE-2018-16864, CVE-2018-16865 works very similarly to a standard stack clash however it is far simpler then CVE-2018-16864. When implementing this exploit the attacker no longer needs to

clash the stack or move the pointer to the begging of the stack. Instead using an alloca() of 4GB all the attacker needs to do is jump the guard page and smash the stack at the end (Qualys Security Advisory, 2019). This is all caused by the alloca() of 4GB as it is much larger then what is expected by journald and allows the attacker to jump from journald's main memory stack into the mmap region and from there the attacker can then overwrite a target function pointer to instead execute malicious code (Qualys Security Advisory, 2019). But for CVE-2018-16865 to work the attacker needs to know the memory address of journald's stack pointer before the jump so for that an information leak is needed which is provided by CVE-2018-16866.

To round out the system down exploits there is the necessary piece of CVE-2018-16866, while the last two exploits functioned as memory stack clashes CVE-2018-16866 is instead a memory leak that also uses systemd as its target. The exploit CVE-2018-16866 occurs when an attacker causes a purposeful out of bound read error in journald (Qualys Security Advisory, 2019). When a syslog message is sent to journald formatted to end with the `:` character this causes journald to improperly handle the terminator that is placed on the syslog causing the pointer in the underlying code to point out of bounds and then subsequently writing and out of bounds log to journald that allows the attacker to then have the ability to read the out of bound string (Qualys Security Advisory, 2019). With this information, the attacker would then be able to ascertain the location of journald's stack pointer and then using the large alloc() jump from exploit CVE-2018-16865 the attacker would then be able to compromise the system.

Linux Hardening

Making an operating system more secure, is a process called hardening. Hardening is defined as Making an operating system more secure. It often requires numerous

actions such as configuring system and network components properly, deleting unused files and applying the latest patches.

This is especially necessary in the linux operating system, as furthered by Mohan where he states “linux is capable of high end security; however, out-of-the-box configurations must be altered to meet the security needs of most businesses with an internet presence.” (Krishnamurthy, 2008 pg.18) which shows how good a linux machine can be if configured the right way, to suite the needs of your company.

Achieving this level of security is not easy by any means, to achieve a high level of security there are a lot steps involved in getting there, in his article titled Securing and Hardening Red†Hat Linux Dhar outlines five main areas that one could look at when looking to harden your linux machine, these include: physical security, installation & initial configuration, Account security, system security and using security tools.

When it comes to physical security, “Physical security is the first step in securing your Linux server. If a malicious user has physical access to your server, all the other security measures you have put in place are moot” (Dhar 2006, pg1) which is a very important thing to note, if the physical security of your linux machine, or any machine for that matter is bypassed then any security setting you have will be essentially useless.

The next step in the basic hardening process comes with your installation & configuration, as stated by, “Giving adequate thought and care to the installation process ensures a system that is trouble free and secure.” (Dhar 2006, pg 2) which basically can set the tone for your security issues in the future, if you are to have any. This process can involve, hard drive partitioning, package installation and boot loader security.

The account security step, is also very vital in the case that you don't want many users to have root access to a system, lest that system be compromised and give root access to someone who shouldn't have root access. This is mainly done through scanning users with superuser privileges and especially restricting remote root login.

The main area where operating system hardening occurs, is in the system security phase. In this phase you would do things such as "disabling nonessential services, mounting partitions properly, and securing various daemons running on the server." (Dhar 2006, pg 6) to begin but Krishnamurthy also states that other key steps in system security include "updating the system, disabling unnecessary services, locking down ports, logging, and maintenance." (Krishnamurthy 2008, pg18) The commonality between the two is very important, because disabling unnecessary services is a very important part of the process, since one of the biggest ideas to have when doing security for a system, is that if it isn't explicitly needed, turn it off, which holds true in this case.

Finally, the last aspect of linux hardening is the use of security tools, these are usually third-party software that help make the system more secure, the main one to make not of however, is "Bastille Linux, a set of Perl scripts, which attempts to carry out automated hardening of a Linux system. It is comprehensive, instructive, and user friendly." (Dhar 2006, pg8) this tool is also talked about by Krishnamurthy, where he even states that "Bastille is powerful and can save administrators time from configuring each individual file and program throughout the operating system." (Krishnamurthy 2008, pg32) which just further shows how useful security tools can be in the hardening process of any linux machine.

References

Law, J. (2017, September 22). Stack Clash Mitigation in GCC - Background. Retrieved February 18, 2019, from https://developers.redhat.com/blog/2017/09/25/stack-clash-mitigation-gcc-background/

System Down: A systemd-journald exploit. (2019, January 9). Retrieved February 1, 2019, from https://www.qualys.com/2019/01/09/system-down/system-down.txt

The Stack Clash. (2017, June 28). Retrieved March 1, 2019, from https://blog.qualys.com/securitylabs/2017/06/19/the-stack-clash

What is systemd? (2018, September 12). Retrieved March 3, 2019, from https://www.linode.com/docs/quick-answers/linux-essentials/what-is-systemd/

Krishnamurthy Madwachar, M. (2008). How to Cheat at Securing Linux. Burlington, MA: Syngress. Retrieved from https://saintleo.idm.oclc.org/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=e000xna&AN=230834&site=ehost-live&scope=site

Dhar, S. (2006). Securing and Hardening Red† Hat Linux. Information Systems Security, 15(1), 21–30. https://doi-org.saintleo.idm.oclc.org/10.1201/1086.1065898X/45926.15.1.20060301/92682.5

What is Linux? The history and background of the world's most popular Open Source Operating System. (n.d.). Retrieved from https://www.linuxtraining academy.com/what-is-linux/

Brooke, B. (2000). What's All This About Linux? *Hispanic*, *13*(6), 60. Retrieved from https://saintleo.idm.oclc.org/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=3207934&site=ehost-live

Spafford, E. H., & Wilson, D. L. (2004, September 24). Whether Linux or Windows, No Software Is Secure. *Chronicle of Higher Education*, pp. B21–B22. Retrieved from https://saintleo.idm.oclc.org/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=14622879&site=ehost-live