Privilege Escalation in Linux Operating System

Hammad Anjum
Department of Cyber Security
Air University
Islamabad, Pakistan
190797@students.au.edu.pk

Ahtsham Saeed
Department of Cyber Security
Air University
Islamabad, Pakistan
190817@students.au.edu.pk

Faizan Khokhar
Department of Cyber Security
Air University
Islamabad, Pakistan
190852@students.au.edu.pk

M.Aryan
Department of Cyber Security
Air University
Islamabad, Pakistan
190813@students.au.edu.pk

I. Introduction

LINUX is an operating system or a kernel distributed under an open-source license. Its functionality list is quite like UNIX. The kernel is a program at the heart of the Linux operating system that takes care of fundamental stuff, like letting hardware communicate with software.In this paper we are using Kali Linux Operating system.

Kali Linux is a Debian-based Linux distribution focused on advanced penetration testing and ethical hacking. It contains several hundred tools which are aimed at a wide range of information security tasks, such as penetration testing, security examinations, computer forensics and reverse engineering.

Attackers can leverage certain techniques to elevate their privileges, which can be further utilized by you to protect your systems and applications from intrusion and attacks.

Industrial use of Linux.

The Linux kernel is used in more smart phones than any other system (Moblin, Meego, Maemo, Android, Tizen etc.), Web Servers, Routers-Linksys famously used the Linux kernel in its routers allowing it to be used with a free GPL license giving way to awesome projects such as DD-WRT, GPS devices-TomTom famously also uses Linux in its systems, Medical equipment to battleships Linux runs more devices than you can think of in any single blog post. There used to be a site called Linux Devices which archived all the places where Linux is used, but sadly it no longer is around.

Kali Linux has evolved not only into the information security professional's platform of choice, but truly into an industrial-grade, world-class, mature, secure, and enterprise-ready operating system distribution.

Kali Linux always provides a customized recent Linux kernel, based on the version in Debian Unstable. This ensures solid hardware support, especially for a wide range of wireless devices. The kernel is patched for wireless injection support since many wireless security assessment tools rely on this feature.

An operating system is responsible for giving programs access to the resources they need when they run. Different users will have different access privileges to files, devices, and other system resources. The operating system must enforce these access privileges. Users should not be able to read or modify data if a policy prohibits them from doing so. Processor time and system memory are also finite resources that need to be allocated across all process in a fair manner that conforms to established policies.

Access control refers to the policies and mechanisms that control who is permitted access to resources and what they can do to those resources.

To enforce security policies and protect resources, we need to know who they apply to. This requires authenticating the user. Authentication is the process of getting and validating a user's identity. After that, ,the system can authorize the user's access to a desired resource. Authorization determines whether an authenticated user is permitted access to a resource and is based on the security policy. A policy is the definition of what is or is not permissible in the organization. A protection mechanism enforces security policies.

Privilege escalation is the act of exploiting a bug, a design flaw, or a configuration oversight in an operating system or software application to gain elevated access to resources that are normally protected from an application or user.

In this paper we will be showing different ways of how an attacker can take advantage of permissions in a Linux operating system. We will be using the Kali Linux distribution of Linux OS that is configured with some permissions that can be exploited to escalate the privileges from one user to another and ultimately become the root user with the highest privileges.

Our contribution to exploitable Linux permissions are as follows:

• Python library hijacking in Linux along with Sudoers permission.

• Exploiting SUID permission on a command owned by root user.

• Exploiting system wide cronjobs that runs a scheduled task as root user.

With the help of above exploitable Linux permission, in this paper we are performing Privilege Escalation in Linux Operating System. Penetration testers and attackers alike often focus on identifying and exploiting weak Linux services and configurations due to their relative stability and also because of their prominence on modern Linux systems.

So as in On the other hand Unlike most mainstream operating systems, Kali Linux is a rolling distribution, which means that you will receive updates every single day.

II. Literature View

We have studied several research papers relating to the privilege escalation and based upon that we have implemented the privilege escalation on Linux OS after exploiting the vulnerabilities. The problem statements highlighted by various papers include It addresses attacks that exploit vulnerable interfaces of trusted operating systems, security critical vulnerable links , Detection and checking of privileges, non-privileged users can escalate permissions by invoking poorly designed higher-privileged users that do not sufficiently protect their interfaces. Kirin is a tool that analyses Manifest files of applications to ensure that granted permissions comply to a system-wide policy. Kirin can be used to prohibit installation of applications which request security-critical combination of permission. SELinux is implemented on top of the Linux Security Modules (LSM) framework which provides a common interface for the Linux kernel for a lot of different security systems, not only SELinux (Wright et al, 2002) and as such, SELinux must adhere to a common interface. This is different from several other security systems, for instance Gr security which is implemented as replacements for the default security model of the kernel (Fox et al, 2008) instead of add-ons to it. Linux kernel denies access to a resource, there is no way for SELinux to override this to provide access to the resource. If a file has permissions which disallow editing of the file by an ordinary user, and we want to enable a specific user to edit that file, then we must first give that user access to write to the file using the normal Linux kernel security model, and can not rely on SELinux to grant this permission. Systrace follows a principle in which is limits a programs access by restricting the system calls which that program makes. Permissions are not enforced specifically for files, but the relevant system calls for opening and writing to a file can be caught by Systrace and then access granted or denied based on those system calls. For writing and reading to files, the Systrace program will keep track of which file is opened so that it can be referred to by name in the policy files. Systrace follows a principle in which is limits a programs access by restricting the system calls which that program makes. Permissions are not enforced specifically for files, but the relevant system calls for opening and writing to a file can be caught by Systrace and then access granted or denied based on those system calls. For writing and reading to files, the Systrace program will keep track of which file is opened so that it can be referred to by name in the policy files.

III. Proposed Methodology

We have 3 users in the system Host (kali), Root and the Guest (hammad) user. Based on the access and the permission of the guest user we will slide into the host user and then become the root user by exploiting some vulnerabilities that are in this system.

Vulnerabilities that we have exploited:

· Sudoers permission to execute a python file (guest to host).

· SUID permission on copy command (host to root)

· Writable shell script in system wide crontabs (host to root While Linux OS can be an effective means to escalate privileges, they can often result in system instability because they tamper with the very foundation of the operating system. Furthermore, kernel exploits are less likely to be successful in environments with thorough patching practices. For these reasons, penetration testers and attackers alike often focus on identifying and exploiting weak Linux services and configurations due to their relative stability and also because of their prominence on modern Linux systems. This section will examine several techniques that can be employed to hijack privileged applications and also examine effective remediation methods.

Attackers are particularly interested in compromising sudo users, that is, users who can execute sudo commands. Sudo, which stands for “substitute user do,” allows a Linux user to run a command as another user, typically root. If an attacker can compromise a user who has sudo rights, he can potentially execute arbitrary commands with root privileges. Administrators are generally aware that they need to properly manage their sudo users just as they manage the root account to prevent sudo misuse. However, critical vulnerabilities are frequently introduced by well-meaning administrators because they installed a poorly configured third party application or issued sudo rights without awareness of command execution vulnerabilities.

In our case the guest user had a python file in their home directory and only has sudo permissions to run that python file as the Host user (kali). The guest user took advantage of this by doing python library hijacking. This means that user created a python file with the same name as that of the module imported in the actual python file and placed some malicious code in it such that it spawns a shell. So whenever the guest user runs the python file as the kali user, instead of importing the original python module the malicious one will be imported which will spawn a shell to guest user of the kali (host) user.

The Infamous SUID, which stands for set owner user ID, is a Linux feature that allows users to execute a file with the permissions of a specified user. For example, the Linux ping command typically requires root permissions in order to open raw network sockets. By marking the ping program as SUID with the owner as root, ping executes with root privileges anytime a low privilege user executes the program. Linux systems also feature the SGID, or set group ID, which allows programs to run as a specified group; for simplicity, this section will focus on SUID, but these techniques also apply to SGID. SUID is a feature that, when used properly, actually enhances Linux security. The problem is that administrators may unwittingly introduce dangerous SUID configurations when they install third party applications or make logical configuration changes.

In our case, when the host (kali) user searches for files and commands with SUID permission enabled, we found out that the copy command has SUID permission is enabled and since the root is the owner of this command we will execute it as root. So we take advantage of this by copying the contents of the shadow file (shadow file stores the password hashes of all users in the system and is not accessible to anyone except root) to another file, let's say password.txt, created by the host user. Then the host user generates the hash of a password and replaces it with the hash of the root password in in the password.txt file. Then again the copy command is used to change the content of the shadow file by copying the content of the password.txt file in the shadow file, allowing us to successfully change the password of the root user.

Crontab (CRON TABLE) is a file which contains the schedule of cron entries to be run and at specified times. File location varies by operating systems. Cron job or cron schedule is a specific set of execution instructions specifying day, time and command to execute. Crontab can have multiple execution statements.

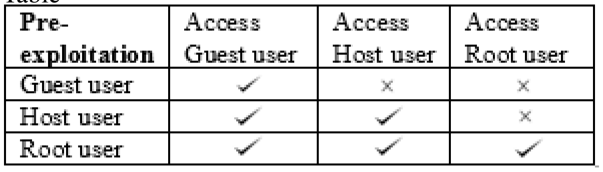
Cron tabs, if not configured properly can be exploited to get root privilege. Cron tabs generally run with root privileges. If we can successfully tamper any script or binary which are defined in the cron jobs then we can execute arbitrary code with root privilege. Any script or binaries in cron jobs which are writable and we can write over the cron file itself.

In our case, when we check the crontabs we can see that we have a cronjob that executes a shell script after every one minute in a Linux directory, when the permissions of that file are checked we got to know that its world readable/writeable/executable meaning that the host user can modify it. A malicious command is added such that whenever the cronjob runs (worst case one minute) as root it will enable SUID permission on the /bin/bash command. Allowing the host user to simply run the /bin/bash –p command to get shell of the root user.

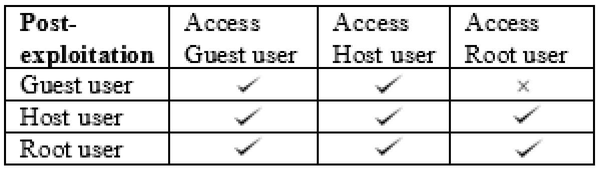
IV. Results

Here we conclude a successful privilege escalation attack on any Linux based systems. We showed how python library hijacking can be used along with Sudoers permission to escalate our privileges. After this we looked at how the SUID permissions on certain commands or executables in Linux can be taken advantage of for escalating privileges. We also looked at cronjobs (which is another widely used feature in Linux OS that is used for scheduling tasks) that consisted of a world readable/writable/executable shell script owned by the root user and was taken advantage by editing it ultimately allowing us to escalate our privileges. Now you know how attackers can leverage certain techniques to elevate their privileges, which can be further utilized by you to protect your systems and applications from intrusion and attacks. Following is the pre-exploitation and post-exploitation comparison of how we have escalated our privileges from one user to another.

Table



Pre-exploitation	Access Guest user	Access Host user	Access Root user
Guest user	✓	×	×
Host user	✓	✓	×
Root user	✓	✓	✓



Post-exploitation	Access Guest user	Access Host user	Access Root user
Guest user	✓	✓	×
Host user	✓	✓	✓
Root user	✓	✓	✓

V. Conclusion

Our project has examined several Linux privilege escalation techniques that are in active use as of the date of this publication. While the Linux community has made great progress in securing their systems, these exploits demonstrate that critical vulnerabilities are still present in the Linux kernel, the operating system, and user level applications. Many of the privilege escalation techniques discussed will remain viable for the foreseeable future, as they exploit foundational capabilities of the Linux operating system. This fact reinforces the importance of identifying, validating, and remediating Linux privilege escalation vulnerabilities. Linux systems such as production servers, embedded devices, and cloud infrastructure are often critical requirements for an organization to operate. If these devices are compromised, the safety of the organization is at stake. Therefore, the administrators should take ownership of the security of these devices and harden them accordingly. In essence, if administrators stay current on patches, carefully audits and privileged programs and users, and follows secure computing practices, they can dramatically reduce their susceptibility to privilege escalation attacks and ultimately enhance their Linux security posture.

VI. References

[1] Long, M., 2019. Attack and defend: Linux privilege escalation techniques of 2016. STI graduate student research. SANS Technology Institute.

[2] Öberg, J., 2009. Comparative study of operating system security using SELinux and Systrace.