Privilege Escalation

Our initial access to a remote server is usually in the context of a low-privileged user, which would not give us complete access over the box. To gain full access, we will need to find an internal/local vulnerability that would escalate our privileges to the `root` user on `Linux` or the `administrator`/`SYSTEM` user on `Windows`. Let us walk through some common methods of escalating our privileges.

PrivEsc Checklists

Once we gain initial access to a box, we want to thoroughly enumerate the box to find any potential vulnerabilities we can exploit to achieve a higher privilege level. We can find many checklists and cheat sheets online that have a collection of checks we can run and the commands to run these checks. One excellent resource is HackTricks, which has an excellent checklist for both Linux and Windows local privilege escalation. Another excellent repository is PayloadsAllTheThings, which also has checklists for both Linux and Windows. We must start experimenting with various commands and techniques and get familiar with them to understand multiple weaknesses that can lead to escalating our privileges.

Enumeration Scripts

Many of the above commands may be automatically run with a script to go through the report and look for any weaknesses. We can run many scripts to automatically enumerate the server by running common commands that return any interesting findings. Some of the common Linux enumeration scripts include LinEnum and linuxprivchecker, and for Windows include Seatbelt and JAWS.

Another useful tool we may use for server enumeration is the Privilege Escalation Awesome Scripts SUITE (PEASS), as it is well maintained to remain up to date and includes scripts for enumerating both Linux and Windows.

Note: These scripts will run many commands known for identifying vulnerabilities and create a lot of "noise" that may trigger anti-virus software or security monitoring software that looks for these types of events. This may prevent the scripts from running or even trigger an alarm that the system has been compromised. In some instances, we may want to do a manual enumeration instead of running scripts.

Let us take an example of running the Linux script from `PEASS` called `LinPEAS`:

```
MichaelLuka@htb[/htb]$ ./linpeas.sh
...SNIP...

Linux Privesc Checklist: https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist
 LEYEND:
  RED/YELLOW: 99% a PE vector
  RED: You must take a look at it
  LightCyan: Users with console
  Blue: Users without console & mounted devs
  Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
  LightMangenta: Your username


====================================( Basic information )=====================================
OS: Linux version 3.9.0-73-generic
User & Groups: uid=33(www-data) gid=33(www-data) groups=33(www-data)
...SNIP...
```

As we can see, once the script runs, it starts collecting information and displaying it in an excellent report. Let us discuss some of the vulnerabilities that we should look for in the output from these scripts.

Kernel Exploits

Whenever we encounter a server running an old operating system, we should start by looking for potential kernel vulnerabilities that may exist. Suppose the server is not being maintained with the latest updates and patches. In that case, it is likely vulnerable to specific kernel exploits found on unpatched versions of Linux and Windows.

For example, the above script showed us the Linux version to be `3.9.0-73-generic`. If we Google exploits for this version or use `searchsploit`, we would find a `CVE-2016-5195`, otherwise known as `DirtyCow`. We can search for and download the DirtyCow exploit and run it on the server to gain root access.

The same concept also applies to Windows, as there are many vulnerabilities in unpatched/older versions of Windows, with various vulnerabilities that can be used for privilege escalation. We should keep in mind that kernel exploits can cause system instability, and we should take great care before running them on production systems. It is best to try them in a lab environment and only run them on production systems with explicit approval and coordination with our client.

Vulnerable Software

Another thing we should look for is installed software. For example, we can use the `dpkg -l` command on Linux or look at `C:\Program Files` in Windows to see what software is installed on the system. We should look for public exploits for any installed software, especially if any older versions are in use, containing unpatched vulnerabilities.

User Privileges

Another critical aspect to look for after gaining access to a server is the privileges available to the user we have access to. Suppose we are allowed to run specific commands as root (or as another user). In that case, we may be able to escalate our privileges to root/system users or gain access as a different user. Below are some common ways to exploit certain user privileges:

1. Sudo
2. SUID
3. Windows Token Privileges

The `sudo` command in Linux allows a user to execute commands as a different user. It is usually used to allow lower privileged users to execute commands as root without giving them access to the root user. This is generally done as specific commands can only be run as root 'like `tcpdump`' or allow the user to access certain root-only directories. We can check what `sudo` privileges we have with the `sudo -l` command:

```
MichaelLuka@htb[/htb]$ sudo -l

[sudo] password for user1:
...SNIP...

User user1 may run the following commands on ExampleServer:
    (ALL : ALL) ALL
```

The above output says that we can run all commands with `sudo`, which gives us complete access, and we can use the `su` command with `sudo` to switch to the root user:

```
MichaelLuka@htb[/htb]$ sudo su -

[sudo] password for user1:
whoami
root
```

The above command requires a password to run any commands with `sudo`. There are certain occasions where we may be allowed to execute certain applications, or all applications, without having to provide a password:

```
MichaelLuka@htb[/htb]$ sudo -l

    (user : user) NOPASSWD: /bin/echo
```

The `NOPASSWD` entry shows that the `/bin/echo` command can be executed without a password. This would be useful if we gained access to the server through a vulnerability and did not have the user's password. As it says `user`, we can run `sudo` as that user and not as root. To do so, we can specify the user with `-u user`:

```
MichaelLuka@htb[/htb]$ sudo -u user /bin/echo Hello World!

    Hello World!
```

Once we find a particular application we can run with `sudo`, we can look for ways to exploit it to get a shell as the root user. GTFOBins contains a list of commands and how they can be exploited through `sudo`. We can search for the application we have `sudo` privilege over, and if it exists, it may tell us the exact command we should execute to gain root access using the `sudo` privilege we have.

LOLBAS also contains a list of Windows applications which we may be able to leverage to perform certain functions, like downloading files or executing commands in the context of a privileged user.

Scheduled Tasks

In both Linux and Windows, there are methods to have scripts run at specific intervals to carry out a task. Some examples are having an anti-virus scan running every hour or a backup script that runs every 30 minutes. There are usually two ways to take advantage of scheduled tasks (Windows) or cron jobs (Linux) to escalate our privileges:

1. Add new scheduled tasks/cron jobs
2. Trick them to execute a malicious software

The easiest way is to check if we are allowed to add new scheduled tasks. In Linux, a common form of maintaining scheduled tasks is through `Cron Jobs`. There are specific directories that we may be able to utilize to add new cron jobs if we have the `write` permissions over them. These include:

1. `/etc/crontab`
2. `/etc/cron.d`
3. `/var/spool/cron/crontabs/root`

If we can write to a directory called by a cron job, we can write a bash script with a reverse shell command, which should send us a reverse shell when executed.

Exposed Credentials

Next, we can look for files we can read and see if they contain any exposed credentials. This is very common with `configuration` files, `log` files, and user history files (`bash_history` in Linux and `PSReadLine` in Windows). The enumeration scripts we discussed at the beginning usually look for potential passwords in files and provide them to us, as below:

```
...SNIP...
[+] Searching passwords in config PHP files
[+] Finding passwords inside logs (limit 70)
...SNIP...
/var/www/html/config.php: $conn = new mysqli(localhost, 'db_user', 'password123');
```

As we can see, the database password '`password123`' is exposed, which would allow us to log in to the local `mysql` databases and look for interesting information. We may also check for `Password Reuse`, as the system user may have used their password for the databases, which may allow us to use the same password to switch to that user, as follows:

```
MichaelLuka@htb[/htb]$ su -

Password: password123
whoami

root
```

We may also use the user credentials to `ssh` into the server as that user.

SSH Keys

Finally, let us discuss SSH keys. If we have read access over the `.ssh` directory for a specific user, we may read their private ssh keys found in `/home/user/.ssh/id_rsa` or `/root/.ssh/id_rsa`, and use it to log in to the server. If we can read the `/root/.ssh/` directory and can read the `id_rsa` file, we can copy it to our machine and use the `-i` flag to log in with it:

```
MichaelLuka@htb[/htb]$ vim id_rsa
MichaelLuka@htb[/htb]$ chmod 600 id_rsa
MichaelLuka@htb[/htb]$ ssh user@10.10.10.10 -i id_rsa

root@remotehost#
```

Note that we used the command 'chmod 600 id_rsa' on the key after we created it on our machine to change the file's permissions to be more restrictive. If ssh keys have lax permissions, i.e., maybe read by other people, the ssh server would prevent them from working.

If we find ourselves with write access to a users`/.ssh/` directory, we can place our public key in the user's ssh directory at `/home/user/.ssh/authorized_keys`. This technique is usually used to gain ssh access after gaining a shell as that user. The current SSH configuration will not accept keys written by other users, so it will only work if we have already gained control over that user. We must first create a new key with `ssh-keygen` and the `-f` flag to specify the output file:

```
MichaelLuka@htb[/htb]$ ssh-keygen -f key

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): *******
Enter same passphrase again: *******

Your identification has been saved in key
Your public key has been saved in key.pub
The key fingerprint is:
SHA256:...SNIP... user@parrot
The key's randomart image is:
+---[RSA 3072]----+
|   ..o.++.+      |
...SNIP...
|     . ..oo+.    |
+----[SHA256]-----+
```

This will give us two files: `key` (which we will use with `ssh -i`) and `key.pub`, which we will copy to the remote machine. Let us copy `key.pub`, then on the remote machine, we will add it into `/root/.ssh/authorized_keys`:

```
user@remotehost$ echo "ssh-rsa AAAAB...SNIP...M= user@parrot" >> /root/.ssh/authorized_keys
```

Now, the remote server should allow us to log in as that user by using our private key:

```
MichaelLuka@htb[/htb]$ ssh root@10.10.10.10 -i key

root@remotehost#
```

As we can see, we can now ssh in as the user `root`. The Linux Privilege Escalation and the Windows Privilege Escalation modules go into more details on how to use each of these methods for Privilege Escalation, and many others as well.

Start Instance

1 / 1 spawns left

Waiting to start...

Questions

Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

Target: Click here to spawn the target system!

SSH to with user "user1" and password "password1"

+ 1 SSH into the server above with the provided credentials, and use the '-p xxxxxx' to specify the port shown above. Once you login, try to find a way to move to 'user2', to get the flag in '/home/user2/flag.txt'.

Submit your answer here...

Submit

Hint

+ 1 Once you gain access to 'user2', try to find a way to escalate your privileges to root, to get the flag in '/root/flag.txt'.

Submit your answer here...

Submit

Hint

Previous Next

Cheat Sheet

Go to Questions

Table of Contents

- Privilege Escalation
- Transferring Files

Getting Started with Hack The Box (HTB)

- Starting Out
- Navigating HTB

Attacking Your First Box

- Nibbles - Enumeration
- Nibbles - Web Footprinting
- Nibbles - Initial Foothold
- Nibbles - Privilege Escalation
- Nibbles - Alternate User Method - Metasploit

Problem Solving

- Common Pitfalls
- Getting Help

What's Next?

- Next Steps
- Knowledge Check

My Workstation

OFFLINE

Start Instance

1 / 1 spawns left