SIGPwny

SP2025 Week 02 • 2025-02-04

Linux Privilege Escalation

Ronan Boyarski

Table of Contents

- Linux OS Theory
  - Permissions model
  - Shell scripts, cron jobs, sudo
  - SSH
- Common Kernel Exploits
  - DirtyCOW, PwnKit
- Exploiting misconfigurations
  - SUID & Sudo misconfigurations
  - Exposed/vulnerable scripts/cron jobs
  - Exposed confidential information
- Manipulating /etc/passwd & /etc/shadow
- Advanced: SSH Hijacking & DevOps attacks

File System Permissions

File System Permissions

- Open up your kali and run `ls -l`
  - The stuff on the left is the permissions

Example Listing:

```
drwxrwxr-x  2 ronan ronan     4096 Sep 29 20:58 Payloads
-rwx------  1 ronan ronan 24383064 Sep 29 21:28 QUIET_DROP.bin
```

File System Permissions

Example Listing:

- File type (**d**irectory or not)

```
drwxrwxr-x  2 ronan ronan     4096 Sep 29 20:58 Payloads
-rwx------  1 ronan ronan 24383064 Sep 29 21:28 QUIET_DROP.bin
```

File System Permissions

Example Listing:

- Permissions applying to owner
- Read, Write, eXecute

```
drwxrwxr-x  2 ronan ronan     4096 Sep 29 20:58 Payloads
-rwx------  1 ronan ronan 24383064 Sep 29 21:28 QUIET_DROP.bin
```

File System Permissions

Example Listing:

- Permissions applying to the same user group
- Read, Write, eXecute (top), none (bottom)

```
drwxrwxr-x  2 ronan ronan     4096 Sep 29 20:58 Payloads
-rwx------  1 ronan ronan 24383064 Sep 29 21:28 QUIET_DROP.bin
```

File System Permissions

Example Listing:

- Permissions applying to everyone else
- Read & eXecute (top), none (bottom)

```
drwxrwxr-x  2 ronan ronan     4096 Sep 29 20:58 Payloads
-rwx------  1 ronan ronan 24383064 Sep 29 21:28 QUIET_DROP.bin
```

File System Permissions

Example Listing:

- User Owner (me)

```
drwxrwxr-x  2 ronan ronan     4096 Sep 29 20:58 Payloads
-rwx------  1 ronan ronan 24383064 Sep 29 21:28 QUIET_DROP.bin
```

File System Permissions

Example Listing:

- Group Owner (also me)

```
drwxrwxr-x  2 ronan ronan     4096 Sep 29 20:58 Payloads
-rwx------  1 ronan ronan 24383064 Sep 29 21:28 QUIET_DROP.bin
```

File System Permissions

Special Permissions

- SUID - when this is set, the file is always executed as if it is executed by its owner user
- SGID - when this is set, the file is always executed as if it is executed by its group owner
- Attributes - can be manipulated with chattr
  - You can set a file to be immutable. This is a neat (although cheesy) trick commonly used in King of the Hill scenarios.

File System Permissions

Changing File Permissions

- Octal for Read (4), Write (2), Execute (1), then repeat for each category
- Use the **`chmod`** command
- If I want to grant everything for everyone, I could do **`chmod 777 file`**
- If I want read + write for only me, I could do **`chmod 600 file`**

Sudo, Shell Scripts, Cron

Sudo

- Superuser Do - allows us to run commands as root
- Root has all of the privileges over everything
- Not all users will have sudo rights
- Can check what our sudo rights are with **`sudo -l`**
    - This will require a password in most instances
- Sudo can be done with a password required, or with no password required
- You can also check `/etc/sudoers`, but this requires sudo rights, so it's pretty self-defeating in a hacking context

Switching Users

- If we have another user's password, we can run `su username`
- If we want to login as root, you want to do `sudo su`, NOT `su root` (the latter would require a root password to be set)
- You can **cat /etc/passwd** to view all users on the system
  - This is, somewhat counterintuitively, not where the passwords are stored
  - The password hashes are in `/etc/shadow`, which is obviously not world-readable

Shell Scripts

- These run shell commands, end with the **.sh** file extension
- Can be very simple or very complex
- Often used for things like installs, but can be used administratively
- It's worth checking them for things like credentials (as they may perform remote logins or send passwords)
- Alternatively, if we have write access to a shell script that another user is running, that's an easy win

Cron Jobs

- Essentially scheduled tasks
- Can be in a number of places
  - `/etc/crontab, /etc/cron.*, /var/spool/cron/crontabs`
- Safest is probably in the /var/spool/cron/crontabs/ directory, as unprivileged users can edit with **crontab -e** but cannot read the directory
- You can always `cat /etc/crontab` and `ls -l /etc/cron*`
- The asterisks signify how often they run
- Check for the ability to write to any privileged cron jobs

SSH

- Stands for Secure Shell (yes it's actually secure)
- However, still very useful for attackers, as remote access is always good
  - Requires a valid login on the target system, usually cannot SSH as root (this is configuration-dependent)
- Syntax: **`ssh user@host`**
- We can also use SSH for port forwarding, either fixed or dynamic
  - I won't go over specifics here but it's something to keep in mind
- This is very useful for lateral movement and persistence

SSH

- You can also do lateral movement by SSH'ing in with a private key instead of a username or password
- So, always be on the lookout for exposed SSH keys, which will sometimes be called **id_rsa**
- Check for authorized_keys files as well as every **.ssh** directory that you can get into
- `chmod 600 id_rsa`
- `ssh -i id_rsa user@host`

Privilege Escalation

Low Effort Kernel Exploits

- When you land on a linux system, start checking OS versions
  - `uname -a`
- Then, you can start googling or searching exploit-db for known exploits for the linux kernel version or the specific distro version
- You can alternatively use LinPEAS 🤮
- Common easy wins for old versions of linux are PwnKit and DirtyCOW for ancient versions of linux
  - These will come up in Boot2Root sometimes, and pretty much every linux box that hasn't been updated since 2021 will be susceptible to PwnKit (when's the last time you ran `sudo apt-get update`?)

Exploiting SUID

- There are many binaries that are not safe to have the SUID bit set
  - Can check them with GTFObins
- Using these is one of the absolute easiest ways to escalate privileges and is going to be common in beginner-level Boot2Root CTFs
  - Does sometimes happen in the real world though
- To show all SUID binaries, you can run:

```
find / -perm -u=s -type f 2>/dev/null
```

Exploiting sudo nopasswd

- If you login, run **`sudo -l`**, and see that you can run ALL with nopasswd, then congratulations, you win
- If there are only specific binaries that can be run with sudo, it's worth referencing GTFObins for specific techniques
- Also be sure to check their file system permissions, because being able to write to them results in an instant win

Exposed Vulnerable Files

- If you find an exposed script that you can write to that belongs to or is being run by a privileged user, be sure to overwrite that with something that will give you a shell
- Always make sure to look around the filesystem and check for any files that could contain credentials that you can read
  - A common but often overlooked one is looking for local and database credentials in config files for web servers
  - After all, you (usually) need a password to connect to the local SQL server, and people would never reuse passwords, right?
  - Of course the usual suspects still apply when it comes to looking for exposed credentials

/etc/passwd & /etc/shadow

- `/etc/passwd` contains the list of users while `/etc/shadow` contains their password hashes
- If you have write access to `/etc/shadow`, you can obtain root access trivially
  - Just make a new password with **`openssl passwd -6 -salt xyz pwned`**
  - Then update the shadow file to contain the new hash in an appropriate format
- If you have read access to `/etc/shadow`, get the password and shadow files, then on your Kali machine run **`unshadow passwd shadow > crackme`**

/etc/passwd & /etc/shadow cont'd

- You can then crack the resulting file with hashcat or John the Ripper (hashcat is preferred but in a VM it makes little difference)
  - You can just run `john crackme -wordlist=/usr/share/wordlists/rockyou.txt`
  - Usually you can use `hashcat -m 1800 crackme -w /usr/share/wordlists/rockyou.txt`
- There are much more advanced cracking/wordlist techniques that are out of scope for this meeting, but I would encourage researching rules, brute force, and keymapping at a minimum

Advanced: SSH Hijacking

SSH Hijacking (ControlMaster)

- Technique where we use an existing connection to compromise a different machine
- The exact technique depends on what software is running, but I will show examples for ControlMaster & SSH-Agent
- ControlMaster enables sharing of multiple SSH sessions over a single network connection (set in ~/.ssh/config)
- When the victim SSH's into the target server through the machine we compromised, we can then SSH into that server

SSH Hijacking (SSH-Agent)

- The client must have the following line in ~/.ssh/config
  - `ForwardAgent Yes`
- The intermediate box (the one that you're on) must have this line
  - `AllowAgentForwarding Yes`
- Then, if `SSH-Agent` is running, and a user SSH's into the target box through the intermediate box (the one that you're on), you can now SSH into the target box as them

Advanced: Attacking DevOps

Ansible

- Ansible is a configuration engine that allows IT personnel to dynamically push configurations and resources to a bunch of remote computers
- The ansible controller can connect to nodes and **run arbitrary python and shell commands**
- We can enumerate if ansible is on the box by just running the `ansible` command
- You can view the ansible host inventory with `cat /etc/ansible/hosts`

Ansible Execution & Playbooks

- You can execute commands for a given ansible group with `ansible <group name> -a "<command>"`
  - You can also add the **-become** flag to run as root
- Ansible uses **playbooks** which are conventionally stored in `/opt/playbooks`
- These can be executed with `ansible-playbook <playbook>.yml`
  - A lot of times these will contain credentials and you can just cat them

Ansible Vaults

- These can be cracked with the following workflow
    - `ansible2john playbook.yml > crackme`
    - `hashcat crackme –force –hash-type 16900 /usr/share/wordlists/rockyou.txt -r /usr/share/hashcat/rules/best64.rule`
    - `cat pw.txt | ansible-vault decrypt`

Ansible Misconfigurations

- If we have write access to a playbook, we can overwrite it to make a backdoor (this is the same as write access to any other script)
- There may be some sensitive data leakage to `/var/log/syslog` in the form of module parameters, which will often contain usernames and passwords

Attacking DevOps Review

- There are many more systems to cover, but this should give a good example of what we can think about when attacking DevOps
- We're looking for the same types of general misconfigurations as general Linux Privilege Escalation, but now our target isn't just the local machine, but everything in our downstream
- Even if we can't root the current machine, getting another user to execute for example a playbook or a backdoored binary is an equally important win

GOAD Time

Next Meetings

2025-02-06 • **This Thursday**

- Linux Hardening Review

2025-02-11 • **Next Tuesday**

- Windows Privilege Escalation Review

2025-02-13 • **Next Thursday**

- Elastic EDR Installation for GOAD