Linux Privilege Escalation

Presented By: Hassan Al Achek

What is the goal of privilege escalation?

- Our ultimate goal with privilege escalation in Linux is to gain a shell running as the <u>root</u> user.
- Privilege escalation can be simple (e.g. a kernel exploit) or require a lot of reconnaissance on the compromised system.

Understanding Permissions in Linux:

permissions in Linux is a relationship between users, groups, files and directories.

- users: multiple groups
- groups: multiple users

Every file and directory defines its permissions in terms of a user, a group, and "others"(all users).

Users:

- The core of the Linux security system is the user account
- Each individual who accesses a Linux system should have a unique user account assigned
- The users' permissions to objects on the system depend on the user account they log in with
- User permissions are tracked using a user ID (often called a UID), which is assigned to an account when it's created
- The Linux system uses special files and utilities to track and manage user accounts on the system.
- The root user is a special type of account in linux system (with UID 0)

/etc/passwd file

```
❯ kali   ~
❱ cat /etc/passwd
```

File: **/etc/passwd**

```
 1  root:x:0:0:root:/root:/bin/bash
 2  daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
 3  bin:x:2:2:bin:/bin:/usr/sbin/nologin
 4  sys:x:3:3:sys:/dev:/usr/sbin/nologin
 5  sync:x:4:65534:sync:/bin:/bin/sync
 6  games:x:5:60:games:/usr/games:/usr/sbin/nologin
 7  man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
 8  lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
 9  mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10  news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11  uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12  proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13  www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14  backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15  list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16  irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
17  gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18  nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19  _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
20  systemd-timesync:x:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
21  systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
22  systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
23  mysql:x:104:110:MySQL Server,,,:/nonexistent:/bin/false
24  tss:x:105:111:TPM software stack,,,:/var/lib/tpm:/bin/false
25  strongswan:x:106:65534::/var/lib/strongswan:/usr/sbin/nologin
26  ntp:x:107:112::/nonexistent:/usr/sbin/nologin
27  messagebus:x:108:113::/nonexistent:/usr/sbin/nologin
28  redsocks:x:109:114::/var/run/redsocks:/usr/sbin/nologin
```
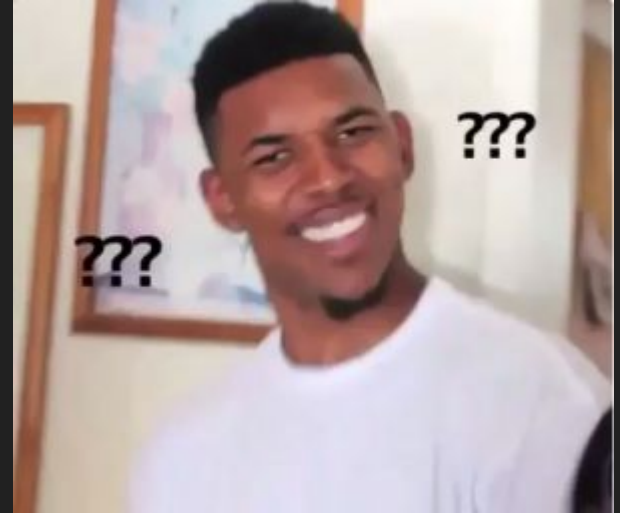
/etc/passwd

`root:x:0:0:root:/root:/bin/bash`

User Name	password	UID	GID	a text description	HOME directory	Shell
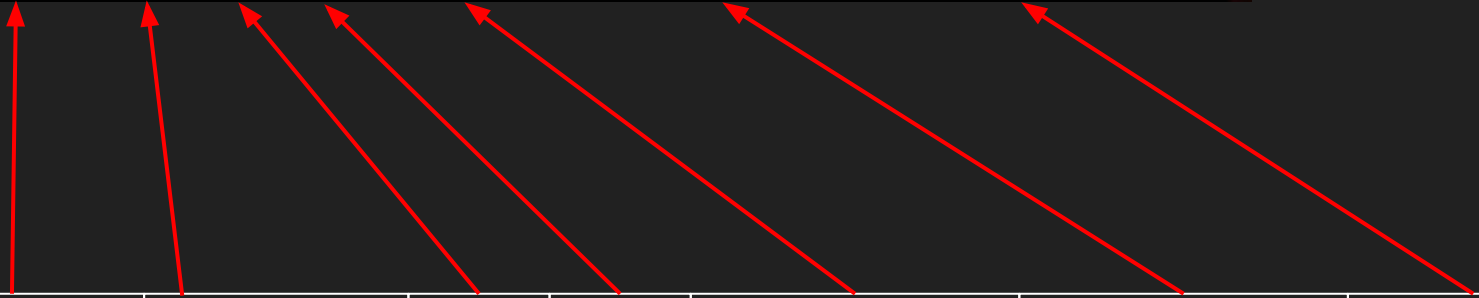
/etc/shadow

But wait Hassan, is the password 'x' for all users

:) wait let's discuss the /etc/shadow

/etc/shadow

File: **/etc/shadow**

```
 1  root:$y$j9T$ZgLWwGZCw/rAZ2/3fBGvX/$3nm/A1EgduaePDpYMZAp.L0CtVASamPXQuvuTERCSTA:18713:0:99999:7:::
 2  daemon:*:18393:0:99999:7:::
 3  bin:*:18393:0:99999:7:::
 4  sys:*:18393:0:99999:7:::
 5  sync:*:18393:0:99999:7:::
 6  games:*:18393:0:99999:7:::
 7  man:*:18393:0:99999:7:::
 8  lp:*:18393:0:99999:7:::
 9  mail:*:18393:0:99999:7:::
10  news:*:18393:0:99999:7:::
11  uucp:*:18393:0:99999:7:::
12  proxy:*:18393:0:99999:7:::
13  www-data:*:18393:0:99999:7:::
14  backup:*:18393:0:99999:7:::
15  list:*:18393:0:99999:7:::
16  irc:*:18393:0:99999:7:::
17  gnats:*:18393:0:99999:7:::
18  nobody:*:18393:0:99999:7:::
19  _apt:*:18393:0:99999:7:::
20  systemd-timesync:*:18393:0:99999:7:::
21  systemd-network:*:18393:0:99999:7:::
22  systemd-resolve:*:18393:0:99999:7:::
23  mysql:!:18393:0:99999:7:::
24  tss:*:18393:0:99999:7:::
```

:

/etc/shadow

```
root:$y$j9T$ZgLWwGZCw/rAZ2/3fBGvX/$3nm/A1EgduaePDpYMZAp.L0CtVASamPXQuvuTERCSTA:18713:0:99999:7:::
```

Login Name

The Encrypted Password

Groups

- Group permissions allow multiple users to share a common set of permissions for an object on the system, such as a file, directory, or device
- Each group has a unique GID, which, like UIDs, is a unique numerical value on the system. Along with the GID, each group has a unique group name.
- Groups are configured in the /etc/group file

/etc/group

```
  1    root:x:0:
   1   daemon:x:1:
   2   bin:x:2:
   3   sys:x:3:
   4   adm:x:4:
   5   tty:x:5:
   6   disk:x:6:
   7   lp:x:7:
   8   mail:x:8:
   9   news:x:9:
  10   uucp:x:10:
  11   man:x:12:
  12   proxy:x:13:
  13   kmem:x:15:
  14   dialout:x:20:
  15   fax:x:21:
  16   voice:x:22:
  17   cdrom:x:24:kali
  18   floppy:x:25:kali
  19   tape:x:26:
  20   sudo:x:27:kali,hassan
  21   audio:x:29:pulse,kali
  22   dip:x:30:kali
"/etc/group" [readonly] 94L, 1385B
```

/etc/group

Group Name	Group Password	Group ID (GID)	Uses in the group

Note: The group password allows a non-group member to temporarily become a member of the group by using the password.

Files

To list file permission we will use ls -l command

```
drwxr-xr-x kali kali    4 KB Thu Mar 25 13:55:56 2021 PHPCodes
drwxr-xr-x kali kali    4 KB Sat Mar 13 23:50:11 2021 Pictures
drwxr-xr-x kali kali    4 KB Thu Mar 25 13:46:54 2021 ProjectsTools
drwxr-xr-x kali kali    4 KB Mon May 11 10:40:39 2020 Public
drwxr-xr-x kali kali    4 KB Fri Oct 23 14:20:50 2020 PycharmProjects
drwxr-xr-x kali kali    4 KB Fri Feb 12 13:15:57 2021 scripts
drwxr-xr-x kali kali    4 KB Thu Sep 24 22:49:19 2020 snap
drwxr-xr-x kali kali    4 KB Mon May 11 10:40:39 2020 Templates
drwxr-xr-x kali kali    4 KB Thu Mar 25 14:20:01 2021 tools
drwxr-xr-x kali kali    4 KB Sun Jan  3 23:54:42 2021 Videos
.rw-r--r-- kali kali  202 B  Sat Mar 20 00:40:57 2021 backupngrok.txt
.rw-r--r-- kali kali  6.1 KB Sat Mar 27 18:36:03 2021 bash.backup
.rw-r--r-- kali kali  3.3 MB Sun Mar 21 18:06:56 2021 harvard.txt
.rw-r--r-- kali kali   38 B  Thu Mar 25 13:43:19 2021 removed.txt
.rw-r--r-- kali kali  319 B  Sun Mar 21 16:34:12 2021 WebScarab.properties
```

Files

- file, d directory	owner	group	others

Note: Only owner of the file can change permissions

All files & directories have a single owner and a group

File Permissions

Read - r - when set, the file contents can be read

Write - w - when set, the file contents can be modified

Execute - x - when set, the file can be executed (i.e. run as some kind of process)

Directory Permission

Execute – when set, the directory can be entered. Without this permission, neither the read nor write permissions will work

Read – when set, the directory contents can be listed

Write – when set, files and subdirectories can be created in the directory

SUID and SGID

- The set user id (SUID): When a file is executed by a user, the program runs under the permissions of the file owner.
- The set group id (SGID): For a file, the program runs under the permissions of the file group. For a directory, new files created in the directory use the directory group as the default group.
- SUID/SGID permissions are represented by an 's' in the execute position.
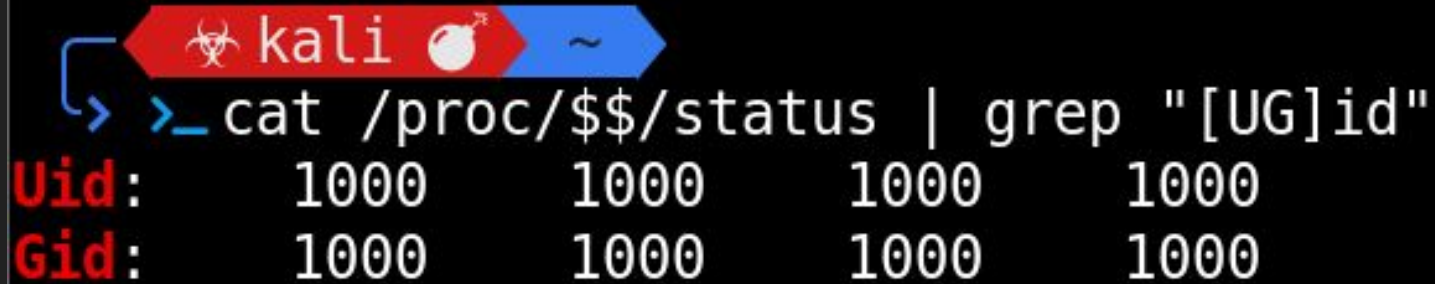
Real, Effective, and Saved UID/GID

- Each user has three user IDs in Linux (real, effective, and saved)
- Real user ID: is who they actually are (check /etc/passwd)
- Effective user ID: is normally equal to their real ID, however when executing a process as another user, the effective ID is set to that user's real ID.
- the effective ID is used in most access control decisions to verify a user, and commands such as whoami use the effective ID.
- Saved user ID: is used to ensure that SUID processes can temporarily switch a user's effective ID back to their real ID and back again without losing track of the original effective ID.

Print real and effective user / group IDs:

```
┌──(kali㉿kali)-[~]
└─$ id
uid=1000(kali) gid=1000(kali) groups=1000(kali),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),109(netdev),117(bluetooth),129(lpadmin),134(scanner)
```

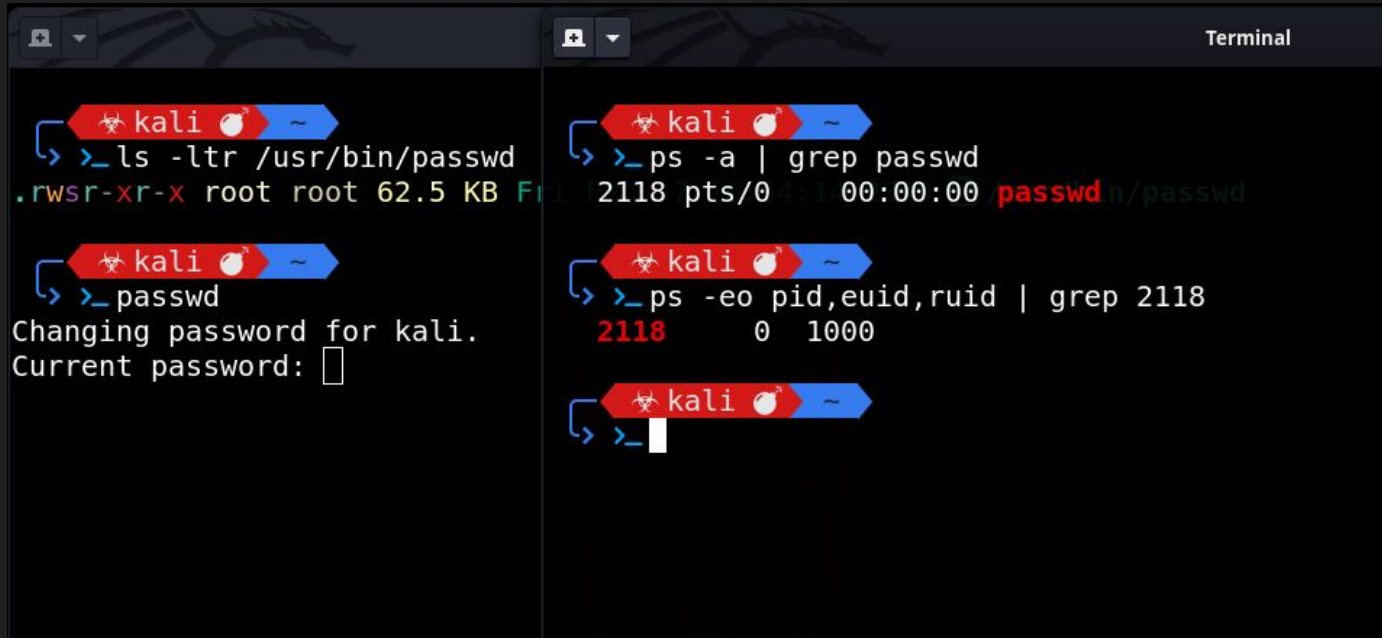Print real, effective, saved, and file system user / group IDs of the current process (i.e. our shell):


```
┌──(kali㉿~)
└─> ╾cat /proc/$$/status | grep "[UG]id"
Uid:    1000    1000    1000    1000
Gid:    1000    1000    1000    1000
```

Viewing Permission

EUID and RUID

Kernel Exploitation

What is a Kernel?

- Kernels are the core of any operating system.
- Think of it as a layer between application software and the actual computer hardware.
- The kernel has complete control over the operating system.
- Exploiting a kernel vulnerability can result in execution as the root user.
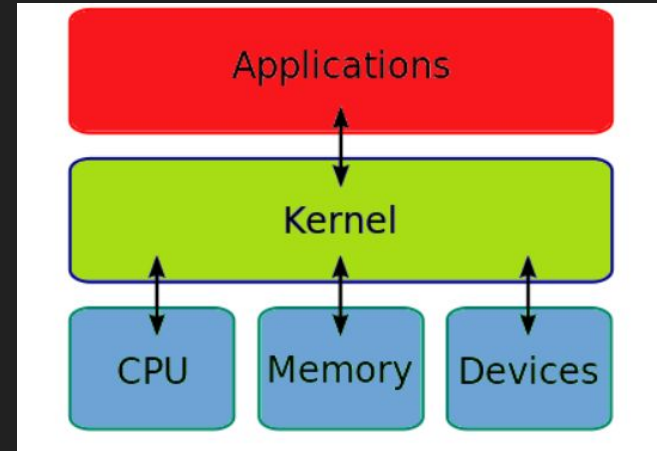
How To Find Kernel Exploitation?

1. Enumeration of kernel version (uname -a)
2. Search for exploit (google, exploit database, or using searchsploit)
3. Compile and run
4. Get Root privilege or Crash the system :)

Enumeration

```
┌──(kali㉿~)
└─> >_ uname -a
Linux kali 5.10.0-kali5-amd64 #1 SMP Debian 5.10.24-1kali1 (2021-03-23) x86_64 GNU/Linux
```

Searchsploit

```
┌──(kali㉿~)
└─> >_ searchsploit linux kernel 5.10.24 priv esc
```

Demo time

Links

Dirty Cow: [PTRACE_POKEDATA variant of CVE-2016-5195 · GitHub](#)

Linux exploit suggester 1: [GitHub - mzet-/linux-exploit-suggester: Linux privilege escalation auditing tool](#)

Linux exploit suggester 2: [GitHub - jondonas/linux-exploit-suggester-2: Next-Generation Linux Kernel Exploit Suggester](#)

Weak File Permissions

- Certain system files can be taken advantage of to perform privilege escalation if the permissions on them are too weak.
- If a system file has confidential information we can read, it may be used to gain access to the root account.
- If a system file can be written to, we may be able to modify the way the operating system works and gain root access that way.

Find Command And Manual Enumeration

Find Command And Manual Enumeration

1. Find writable file: find /etc -maxdepth 1 -writable -type f
2. Find readable file: find /etc -maxdepth 1 -readable -type f
3. Find all directories which can be written to:

   find / -executable -writable -type d 2> /dev/null

Readable /etc/shadow

- By Default Readable by root only
- If we are able to read the content of /etc/shadow
- We can try to crack the hash of the root user
- Using HashCat or John

john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

Readable /etc/shadow

~# unshadow passwd shadow > crackme

~# john --wordlist<=custom wordlist file> <file to crack>

~# john --wordlist=/usr/share/wordlists/rockyou.txt crackme

```
┌──(kali㉿ ~)
└─>_ unshadow  pass hash > crackme
┌──(kali㉿ ~)
└─>_ cat crackme
```

```
     File: crackme

  1  kali:$6$pPKWGpwcuEgoOF0z$GFSBP6xJLpcTZlBwyo.RbRRitHXZ6HZeSsQPSP4xrBYRAce.iqshkVcDjgod5LpF8OYPdBRRdc11zE.s0rcAF.:1000:1000:,,,:
     /home/kali:/bin/bash
```

Readable /etc/shadow


```
┌──(kali㉿ )-[~]
└─$ john --wordlist=wordlist.txt crackme
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA224"
Use the "--format=HMAC-SHA224" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
No password hashes left to crack (see FAQ)

┌──(kali㉿ )-[~]
└─$ john --show crackme
kali:kali:1000:1000:,,,:/home/kali:/bin/bash

1 password hash cracked, 0 left
```

Writable /etc/shadow

Generate a new SHA-512 password hash:

~# mkpasswd -m sha-512 newpassword

Edit the /etc/shadow and replace the root user's password hash with the one we generated.

~# su #switch user

```
kali:$6$pPKWGpwcuEgoOF0z$GFSBP6xJLpcTZlBwyo.RbRRitHXZ6HZeSsQPSP4xrBYRAce.iqshkVcDjgod5LpF8OYPdBRRdc11zE.s0rcAF.:18393:0:99999:7:::
```

```
⟲ kali 🐙 ~
└─>_ mkpasswd -m sha-512 newpassword
$6$36Wpvh7pR2qKZ/iQ$JUwQAxFPJAe37effE/rJqY3H.ilzfV3/D.1965RX9WSJwAlrtsyZyBublw5j3kBtZZla.h.uTpgeb/3QhX5t3/
```

Sudo

What is sudo?

sudo is a program which lets users run other programs with the security privileges of other users. By default, that other user will be root.

A user generally needs to enter their password to use sudo, and they must be permitted access via rule(s) in the /etc/sudoers file.

List programs a user is allowed (and disallowed) to run:

~$ sudo -l


```
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED....and...
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
```

Known Password

~$ sudo su

Password:

~#whoami

Root

~#

If su program is not allowed

~$ sudo -s   # -s for shell

~$ sudo -i   # -i run login shell as the target user

~$ sudo /bin/bash   # run the /bin/bash

~$ sudo passwd   # passwd run passwd

Shell Escape Sequences

GTFOBins

Even if we are restricted to running certain programs via sudo, it is sometimes possible to "escape" the program and spawn a shell.

GTFOBins ☆ Star 4.446

GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.

The project collects legitimate functions of Unix binaries that can be abused to ~~get the f**k~~ break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.

It is important to note that this is **not** a list of exploits, and the programs listed here are not vulnerable per se, rather, GTFOBins is a compendium about how to live off the land when you only have certain binaries available.

GTFOBins is a collaborative project created by Emilio Pinna and Andrea Cardaci where everyone can contribute with additional binaries and techniques.

If you are looking for Windows binaries you should visit LOLBAS.

SUID and SGID Executable

SUID files get executed with the privileges of the file owner.

SGID files get executed with the privileges of the file group.

If the file is owned by root, it gets executed with root privileges, and we may be able to use it to escalate privileges.

We can use the following find command to locate files with the SUID or SGID bits set:

~$ find / -type f -a \(-perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2> /dev/null

Cron Jobs

Cron jobs are programs or scripts which users can schedule to run at specific times or intervals.

Cron jobs run with the security level of the user who owns them.

By default, cron jobs are run using the /bin/sh shell, with

limited environment variables.
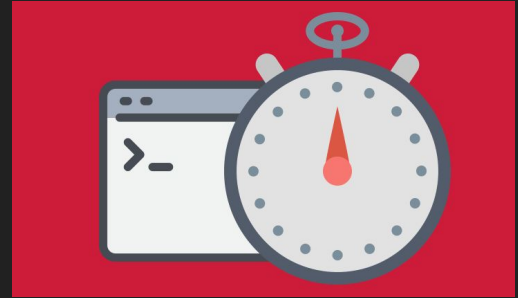
Cron Jobs

Cron table files (crontabs) store the configuration for cron jobs.

User crontabs are usually located in /var/spool/cron/ or /var/spool/cron/crontabs/

The system-wide crontab is located at /etc/crontab.

File Permissions

Misconfiguration of file permissions associated with cron jobs can lead to easy privilege escalation.

If we can write to a program or script which gets run as part of a cron job, we can replace it with our own code.

View the contents of the system-wide crontab:

~# cat /etc/crontab

Try Hack Me Room

https://tryhackme.com/room/linuxprivesc

1. Create Account
2. Download Vpn Configuration File
3. Run sudo openvpn username.ovpn
4. Verify Connection by typing ifconfig tun0
5. SSH: ssh username@targetIp then enter password
6. Get SSH Session Enjoy ;)

Thanks You :)

Tools For Automation

[GitHub - diego-treitos/linux-smart-enumeration: Linux enumeration tool for pentesting and CTFs with verbosity levels](#)

[GitHub - rebootuser/LinEnum: Scripted Local Linux Enumeration & Privilege Escalation Checks](#)

[GTFOBins](#)

Bash scripting: https://devhints.io/bash

[GitHub - sagishahar/lpeworkshop: Windows / Linux Local Privilege Escalation Workshop](#)

[privilege-escalation-awesome-scripts-suite/linPEAS at master · carlospolop/privilege-escalation-awesome-scripts-suite · GitHub](#)