Program Misuse

Privilege Escalation

Yan Shoshitaishvili
Arizona State University

Recall the Linux permission model

Every process has a user ID and a GID.

```
yans@pwn ~/pwn $ id
uid=1000(yans) gid=1000(yans) groups=1000(yans),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(
sambashare),136(docker)
```

Every file and directory is owned by a user and group.

```
yans@pwn ~/pwn $ ls -l /bin/cat
-rwxr-xr-x 1 root root 43416 Sep  5  2019 /bin/cat
```

Child processes inherit from parent processes.

```
yans@pwn ~/pwn $ cat getuid.c
int main() { printf("UID: %d\n", getuid()); }
yans@pwn ~/pwn $ gcc -w -o getuid getuid.c
yans@pwn ~/pwn $ id
uid=1000(yans) gid=1000(yans) groups=1000(yans),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(
sambashare),136(docker)
yans@pwn ~/pwn $ ./getuid
UID: 1000
```

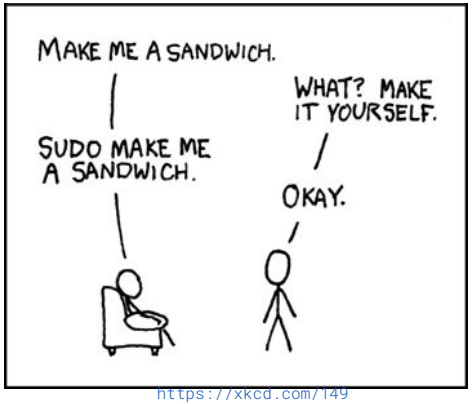
Some UIDs are better than others...

UID 0 is the Linux administrator user, root. Roughly speaking, you need root for:

- Installing software.
- Loading device drivers.
- Shutting down, rebooting.
- Changing system-wide settings.

But if you're UID 1000, how do you become UID 0?

Privilege Elevation



https://xkcd.com/149

One way to elevate your privileges is to run an *suid binary*.

SUID is a bit in the Linux permission model:

SUID	User Read	User Write	User Exec	SGID	Group Read	Group Write	Group Exec	Sticky	World Read	World Write	World Exec

SUID: execute with the *e*UID of the file owner rather than the parent process.
SGID: execute with the *e*GID of the file owner rather than the parent process.
Sticky: used for shared directories to limit file removal to file owners.

Common examples of SUID binaries: sudo, su, newgrp

Quick Detour: *e* UID?

Three different type of user and group IDs:

Effective (eUID, eGID): the UID/GID used for most access checks.
Real (UID, GID): the "real" UID (or GID) of the process owner, used for things such as signal checks.
Saved: a UID/GID that your process *could* switch its eUID/eGID to. Used for temporarily dropping privileges.

With great power comes great responsibility...

eUID 0 is powerful. Aside from system management, root can (by default):

Open any file!
- Including things in the special /proc filesystem!
- And device-backed files!

Execute any program.
Assume any other UID or GID.
Debug any program.

Obviously, this can be a security disaster...

Privilege Escalation

Privilege Escalation is a class of exploit in which the attacker elevates their privileges to (generally) root level.

Typical flow:

1. Gain a foothold on the system (vulnerable network service, intended shell access, code in app context, etc).
2. Identify a vulnerable privileged service.
3. Exploit the privileged service to gain its privileges.

Example: if an SUID binary has a security problem, an attacker can use it in a *privilege escalation* attack:

Security Woes

Who would be this careless?

1. Vulnerabilities in SUID binaries, such as sudo:
    a. CVE-2019-14287: privilege escalation under certain configurations.
    b. CVE-2018-10852: permission misconfiguration leading to privilege escalation
    c. CVE-2017-1000367: improper input sanitization leading to command execution
    d. CVE-2016-7076: privilege escalation under certain invocation scenarios
    e. CVE-2016-7091: privileged information disclosure
    f. CVE-2015-5602: privilege escalation under certain configurations
    g. CVE-2014-0106: bypass of configuration restrictions
2. Unnecessary SUDOing (or running as root by other means) other software.
    a. Depressingly common in course grading systems, other shared server management software.
    b. Too common with containerization (docker's default user is root).
3. OS-level vulnerabilities (stay tuned!).

Practice Problems!

This module's practice problems:

1. Connect to pwn.college.
2. Select a program path to unnecessarily SUID.
    a. TONS of programs can be chosen, but not everything is viable to read the flag with...
3. Use this program to read /flag, if you can!

Let's demo a few:

- `/bin/cat`
- `/usr/bin/more`
- `/usr/bin/find`