Using setuid (Lab)

coursera.org/learn/linux-tools-for-developers/supplement/1CpX6/using-setuid-lab

Exercise

Normally, programs are run with the privileges of the user who is executing the program. Occasionally, it may make sense to have normal users have expanded capabilities they would not normally have, such as the ability to start or stop a network interface, or edit a file owned by the superuser.

By setting the **setuid** (set user ID) flag on an executable file, you modify this normal behavior by giving the program the access rights of the owner rather than the user of the program.

We should emphasize that this is generally a bad idea and is to be avoided in most circumstances. It is often better to write a daemon program with lesser privileges for this kind of use.

Suppose we have the following C program (**writeit.c,** provided to save typing for download with the solution below) which attempts to overwrite a file in the current directory, **afile**:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
15
16
17
18
19
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <stdlib.h>
#include <sys/stat.h>

int main(int argc, char *argv[])
{
        int fd, rc;
        char *buffer = "TESTING A WRITE";

        fd = open("./afile", O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
        rc = write(fd, buffer, strlen(buffer));
        printf("wrote %d bytes\n", rc);
        close(fd);
        exit(EXIT_SUCCESS);
}
```

which can be compiled simply by doing:

```
1
```

```
$ make writeit
```

or

```
1
 $ gcc -o writeit writeit.c
```

If you try to run this program on a file owned by root, you will get the following sequence of events:

```
1
2
3
4
5
$ sudo touch afile
$ ./writeit afile
wrote -1 bytes
$ sudo ./writeit afile
wrote 15 bytes
```

Thus, the root user was able to overwrite the file it owned, but a normal user could not.

Note that changing the owner of **writeit** to root will not help:

```
1

2

3

$ sudo chown root.root writeit

$ ./writeit

wrote -1 bytes
```

By setting the **setuid** bit you can make any normal user able to do this:

```
1

2

3

$ sudo chmod +s writeit

$ ./writeit

wrote 15 bytes
```

You may be asking why we did not just write a script to do such an operation, rather than to write and compile an executable program? The reason is that under Linux if you change the **setuid** bit on such an executable, it will not do anything unless you actually change the **setuid** bit on the shell, which would be downright stupid.

Solution

You can see a solution for this exercise here:

lab_setuidSH File
Download file
lab_writeC File
Download file