Set-UID Lab

Erik Olsen - Infosec Club

What is a UID?

- User Identifier
- Every user in linux has a unique UID
- When a user executes a program that program is given the UID of the user
- Used for access control

UIDs can be found in the `/etc/passwd` file

The UID for the `root` user is always `0`.

The UID for my `kali` user happens to be `1000`.

The /etc/shadow file

Contains the hashed+salted passwords for all of the users on the system

```
kali:$y$j9T$HlGqUlZv2BS4c1t9LSd8f0$.s8R4LbyUjkADBNmIQIKETz68lWbRLdVWTXE
QbWvQB7:18693:0:99999:7:::
```

```
┌──(kali㉿kali)-[~]
└─$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1577 Mar  7 14:17 /etc/shadow
```

```
┌──(kali㉿kali)-[~]
└─$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

```
┌──(root💀kali)-[/home/kali]
└─# cat /etc/shadow
root:!:18693:0:99999:7:::
daemon:*:18693:0:99999:7:::
bin:*:18693:0:99999:7:::
```

I have a 6 character alphanumeric password - The first person to crack it gets a free raspberry pi (email me: eolsen@syr.edu)

What is a Set-UID program?

- A program that will temporarily change its UID, regardless of the UID of the user who ran it, to perform a privileged action.
- Example: `/bin/passwd`
  - If a user changes their password, the shadow file must be modified.
  - No one, except root, should be allowed to modify the shadow file.
  - `/bin/passwd` temporarily sets its UID to 0 to modify the shadow file when a user changes their password.

Making a Set-UID program

```
┌──(kali㉿kali)-[~/Documents/SetUIDLab]
└─$ ls -l myprogram
-rwxr-xr-x 1 kali kali 16656 Mar  7 15:39 myprogram

┌──(kali㉿kali)-[~/Documents/SetUIDLab]
└─$ sudo chown root myprogram

┌──(kali㉿kali)-[~/Documents/SetUIDLab]
└─$ sudo chmod 4755 myprogram

┌──(kali㉿kali)-[~/Documents/SetUIDLab]
└─$ ls -l myprogram
-rwsr-xr-x 1 root kali 16656 Mar  7 15:39 myprogram
```

Lab: Setup 5 min

1. Clone the lab repository

```
┌──(kali㉿kali)-[~/Documents]
└─$ git clone https://github.com/eriknj99/SetUIDLab.git
```

2. Execute `enable-vulnerability.sh`

```
┌──(kali㉿kali)-[~/Documents/SetUIDLab]
└─$ ./enable-vulnerability.sh
Making the system vulnerable to Set-UID attacks ...
sudo ln -sf /bin/zsh /bin/sh
Done.
Make sure you run disable-vulnerability.sh when you are done with the lab
```

Lab: Part 1

```
┌──(kali㉿kali)
└─$ whoami
kali
```

```
┌──(kali㉿kali)-[
└─$ id -u kali
1000
```

```
┌──(kali㉿kali)-[~/
└─$ id -u $(whoami)
1000
```

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

//PrintMyUID.c

int main(){
  printf("Hello! My UID is:\n");
  system("id -u $(whoami)");

  printf("\nwhich corresponds to the user: \n");
  system("whoami");

  return 0;

}
```

Lab: Part 1

15 min

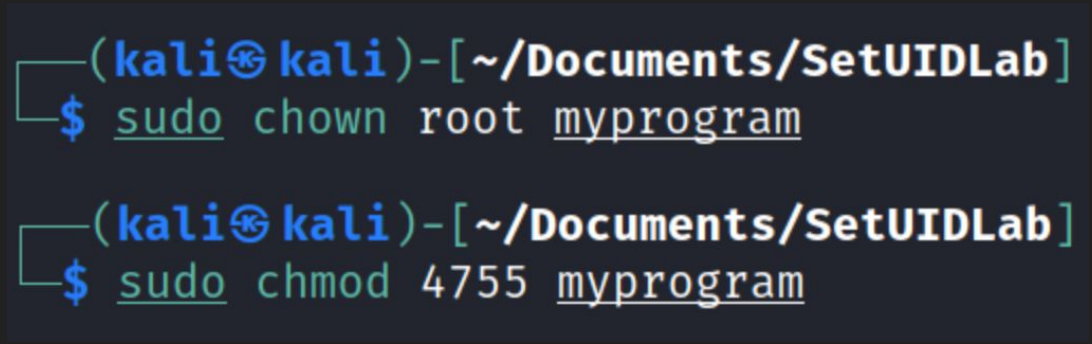
Task 1: Compile `PrintMyUID.c` into the program `PrintMyUID`. (Hint: Use `gcc`)

Task 2: Run the `PrintMyUID` program.

Task 3: Change `PrintMyUID` to a root Set-UID program.

Task 4: Run the `PrintMyUID` program again and observe the changes.

Task 5 (Optional): Modify `PrintMyUID.c` to print the contents of the `/etc/shadow` file. Repeat steps 1-4.



Lab: Part 1 Demonstration

Lab: Part 2

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
//-----Exploitable.c-----


char *v[3];
char *command;

int main(int argc, char *argv[]){
  // argc : The number of command line arguments
  // argv : An array of string (size argc) containing all command line arguments.

  if(argc > 1){

    printf("The contents of the file %s are:\n", argv[1]);

    // Concatinate /bin/cat with argv[1] and set command equal to the result.
    // Equivelent to command = "/bin/cat" + argv[1] in python
    v[0] = "/bin/cat";
    v[1] = argv[1];
    command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
    sprintf(command, "%s %s", v[0], v[1]);


    //Execute the command
    system(command);


  }else{
    printf("Please enter a filename as a command line argument.\n");
    printf("Example: $ Exploitable /file/to/print \n");
  }


  return 0;
}
```

Lab: Part 2 15 min

Task 1: Compile `Exploitable.c` and make it a Set-UID program with `compile-exploitable.sh`

```
┌──(kali㉿kali)-[~/Documents/SetUIDLab]
└─$ ./compile-exploitable.sh
Compiling Exploitable.c...
Making Exploitable a Set-UID program...
done.
```

Task 2: Make the `Exploitable` program execute arbitrary commands as root using only command line arguments.

Lab: Part 2 Demonstration

Lab: Clean Up

1. Execute `disable-vulnerability.sh` to patch the vulnerability we enabled earlier.

Thank You!