Assignment1: SEED Labs – Environment Variable and Set-UID Program

Mayur Suresh

CSC 302-01: Computer Security

Date: February 19th, 2023

Task 4: Environment Variables and system()

For this task we will see the behavior of using the system() function with another program. We will use this function to execute a command. Unlike the execve() function which will directly execute a command the system() function will invoke the `/bin/`sh -c` command.

We will first run and compile a program that utilizes the system() function and we see the environment variables below.

```
seed@pcvm471-2:~$ cat task4.c

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

int main() {
  system("/usr/bin/env");
  return 0 ;
}
seed@pcvm471-2:~$
```

```
seed@pcvm471-2:~$ ./a.out
SHELL=/bin/bash
PWD=/home/seed
LOGNAME=seed
XDG_SESSION_TYPE=tty
MOTD_SHOWN=pam
HOME=/home/seed
LANG=en_US.UTF-8
SSH_CONNECTION=172.58.232.74 41953 155.98.38.249 22
XDG_SESSION_CLASS=user
TERM=xterm-256color
USER=seed
SHLVL=1
XDG_SESSION_ID=1871
XDG_RUNTIME_DIR=/run/user/38503
SSH_CLIENT=172.58.232.74 41953 22
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:.
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/38503/bus
SSH_TTY=/dev/pts/1
_=/usr/bin/env
OLDPWD=/home/seed
seed@pcvm471-2:~$
```

Task 5: Environment Variables and Set-UID Programs

Set-UID is a critical security mechanism in Unix OS, as it assumes the owner's privileges when running. It can gain root privileges then this program is run if the owner is in root.

We first write a program that will print out the environment variables with this process and compiled as shown below.

```
PS C:\Users\sures> cat task5.c
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

int main() {
  int i = 0;
  while (environ[i] != NULL) {
    printf("%s\n", environ[i]);
    i++;
  }
}

PS C:\Users\sures>
```

```
seed@pcvm471-2:~$ ./a.out
SHELL=/bin/bash
PWD=/home/seed
LOGNAME=seed
XDG_SESSION_TYPE=tty
MOTD_SHOWN=pam
HOME=/home/seed
LANG=en_US.UTF-8
SSH_CONNECTION=172.58.232.74 22737 155.98.38.249 22
XDG_SESSION_CLASS=user
TERM=xterm-256color
USER=seed
SHLVL=1
XDG_SESSION_ID=1876
XDG_RUNTIME_DIR=/run/user/38503
SSH_CLIENT=172.58.232.74 22737 22
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi
n:/usr/games:/usr/local/games:.
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/38503/bus
SSH_TTY=/dev/pts/2
_=./a.out
seed@pcvm471-2:~$
```

We then run the following commands with the binary file outputted from the program.

```
seed@pcvm471-2:~$ sudo chown root a.out
seed@pcvm471-2:~$ sudo chmod 4755 a.out
```

We all so see that we have switched the ownership to root below.

```
seed@pcvm471-2:~$ ls -l a.out
-rwsr-xr-x 1 root seed 16768 Feb 19 16:04 a.out
seed@pcvm471-2:~$
```

When then using the export commands set the PATH, LD_LIBRARY_PATH and a PATH defined by use like below and then see it listed in the environment variables.

```
seed@pcvm471-2:~$ ./a.out
SHELL=/bin/bash
PWD=/home/seed
LOGNAME=seed
XDG_SESSION_TYPE=tty
MOTD_SHOWN=pam
HOME=/home/seed
LANG=en_US.UTF-8
SSH_CONNECTION=172.58.232.74 52546 155.98.38.249 22
XDG_SESSION_CLASS=user
TERM=xterm-256color
USER=seed
MY_CATS=i love cats
SHLVL=1
XDG_SESSION_ID=1885
XDG_RUNTIME_DIR=/run/user/38503
SSH_CLIENT=172.58.232.74 52546 22
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi
n:/usr/games:/usr/local/games:.
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/38503/bus
SSH_TTY=/dev/pts/4
_=./a.out
seed@pcvm471-2:~$
```

We now run the Set-UID again with the same process as before and wee see that nothing happens. I am very surprised by this because I was under the assumption that I would be sent over to root, but I stayed in seed. I now run the binary file again and only see the environment variables displayed again.

Task 6: Environment Variables and Set-UID Programs

For this task we will change the path address as it is dangerous to call system() inside a Set-UID program.

First, we will change the PATH using export, setting it to the `/home/seed` directory to the beginning of the PATH environment variable. As shown with the environment variables below.

```
seed@pcvm471-2:~$ export PATH=/home/seed:$PATH
seed@pcvm471-2:~$ ./a.out
SHELL=/bin/bash
PWD=/home/seed
LOGNAME=seed
XDG_SESSION_TYPE=tty
MOTD_SHOWN=pam
HOME=/home/seed
LANG=en_US.UTF-8
SSH_CONNECTION=71.230.28.39 51082 155.98.38.249 22
XDG_SESSION_CLASS=user
TERM=xterm-256color
USER=seed
SHLVL=1
XDG_SESSION_ID=1917
XDG_RUNTIME_DIR=/run/user/38503
SSH_CLIENT=71.230.28.39 51082 22
PATH=/home/seed:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/gam
es:.
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/38503/bus
SSH_TTY=/dev/pts/0
_=./a.out
seed@pcvm471-2:~$
```

We now set up a Set-UID program that executes the `/bin/sh` command. We compile the program, this program changes the owner to the root with the chown/chmod commands and this makes it a Set-UID program.

```
seed@pcvm471-2:~$ nano task6.c
seed@pcvm471-2:~$ gcc task6.c
task6.c: In function 'main':
task6.c:3:3: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
    3 |   system("ls");
      |   ^~~~~~
seed@pcvm471-2:~$ sudo chown root a.out
[sudo] password for seed:
seed@pcvm471-2:~$ sudo chmod 4755 a.out
seed@pcvm471-2:~$ ls -l a.out
-rwsr-xr-x 1 root seed 16696 Feb 19 19:06 a.out
seed@pcvm471-2:~$
```

When we run the binary file produced by the program, we see that it executes as it supposed showing the files under the current directory shown below.

```
seed@pcvm471-2:~$ a.out
a.out                mem_layout.c        myshell222                          stack
badfile              mem_layout.o        myshell23232323                     stack.c
exploit              mem_layout_print    peda                                stack.c.save
exploit.c            mem_layout_print.c  peda-session-gdb_stack.txt           strcpy_overflow
file                 myshe               peda-session-gdb_strcpy_overflow.txt strcpy_overflow.c
find_myshell.c       myshell             peda-session-stack.txt              task4.c
gdb_stack            myshell0            ret_to_libc_exploit.c               task5.c
gdb_strcpy_overflow  myshell1            setuid_lab                          task6.c
kcats                myshell2            shellcode
mem_layout           myshell22           shellcode.c
seed@pcvm471-2:~$
```

We next go ahead and change the command called from `/bin/sh` to one of our own. I will choose the `pwd` command to run this program as shown below. We see that with the system() and changing the owner from seed to root that it acts just like a Set-UID program would behave.

```
seed@pcvm471-2:~$ gcc task6.c
task6.c: In function 'main':
task6.c:3:3: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
    3 |   system("pwd");
      |   ^~~~~~
seed@pcvm471-2:~$ sudo chown root a.out
seed@pcvm471-2:~$ sudo chmod 4755 a.out
seed@pcvm471-2:~$ ls -l a.out
-rwsr-xr-x 1 root seed 16696 Feb 19 19:16 a.out
seed@pcvm471-2:~$ a.out
/home/seed
seed@pcvm471-2:~$
```

```
seed@pcvm471-2:~$ cat task6.c

int main() {
  system("pwd");
  return 0;
}
seed@pcvm471-2:~$
```

Task 7: The LD_PRELOAD Environment Variable and Set-UID Programs

For this task we will see how the Set-UID program deals with many environment variables such as LD_LIBRARY_PATH and the LD_PRELOAD.

We will now see how these environment variables changes the behavior of dynamic loader/linker when running a program. First, we build a dynamic link library with a program that overrides the sleep() function in libc.

```
seed@pcvm471-2:~$ nano mylib.c
seed@pcvm471-2:~$ gcc -fPIC -g -c mylib.c
seed@pcvm471-2:~$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
seed@pcvm471-2:~$ ls
a.out                mem_layout.c        myshell22                           stack
badfile              mem_layout.o        myshell222                          stack.c
exploit              mem_layout_print    myshell23232323                     stack.c.save
exploit.c            mem_layout_print.c  peda                                strcpy_overflow
file                 mylib.c             peda-session-gdb_stack.txt          strcpy_overflow.c
find_myshell.c       mylib.o             peda-session-gdb_strcpy_overflow.txt task4.c
gdb_stack            myshe               peda-session-stack.txt              task5.c
gdb_strcpy_overflow  myshell             ret_to_libc_exploit.c               task6.c
kcats                myshell0            setuid_lab
libmylib.so.1.0.1    myshell1            shellcode
mem_layout           myshell2            shellcode.c
seed@pcvm471-2:~$
```

Then we set the LD_RELOAD to the ` ./libmylib.so.1.0.1` address. Next, we compile the program the program myprog.

We now run the myprog program which is a normal program in many ways to see the difference. We first run the program as a normal user, then make the myprog a Set-UID root program by switching its owner from seed to root and then export the LD_PRELOAD environment variable and then run the program.

```
seed@pcvm471-2:~$ a.out
I am not sleeping!
seed@pcvm471-2:~$
```

1) We first run it normally

```
seed@pcvm471-2:~$ sudo chmod 4755 myprog
seed@pcvm471-2:~$ sudo chown root myprog
seed@pcvm471-2:~$ sudo chmod 4755 myprog
seed@pcvm471-2:~$ ls -l myprog
-rwsr-xr-x 1 root seed 16696 Feb 19 20:49 myprog
seed@pcvm471-2:~$ myprog
seed@pcvm471-2:~$
```

2) We then make it a Set-UID program and run it as a normal user.

```
root@pcvm471-2:/home/seed# export LD_PRELOAD=./libmylib.so.1.0.1
root@pcvm471-2:/home/seed# ./myprog
I am not sleeping!
root@pcvm471-2:/home/seed#
```

3) Finally, we export the variable LD_PRELOAD in the root and run it again

There is a difference in between all these scenarios because it is all about the environment variables. In the first step be export the LD_PRELOAD when we are in seed and then we run it and this is why we are able to see the string. However, when we switch the program's owner to the root and run it like in second step we are not able to see the string as we only exported the environment variable in the seed at that time. So we go into the root export the environment variable and run it we see that string again. So, it all comes down to then the LD_PRELOAD is set.

Task 8: The LD_PRELOAD Environment Variable and Set-UID Programs

In this task we will work with a scenario where we use the program given is compiled and used to explain how to help Bob.

First we create a catall.c and compile the program and change its owner to root. We see below running the program after making it a Set-UID program by giving it a sample file. We are able to see the file from the root.

```
seed@pcvm471-2:~$ catall myprog.c
/* myprog.c */
#include <unistd.h>

int main() {
  sleep(1);
  return 0;
}
seed@pcvm471-2:~$
```

But with just with the program provided Bob is not able to modify or edit any files. Hence, Bob is not able to delete a file that is not writable to us. This program does not give us permission or the ability when the program is on in the root, because the file specified is not owned by the root.

We no switch from the system() to the execve() and compile it after making it a Set-UID program.

```
seed@pcvm471-2:~$ cat catall.c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    char *v[3];
    char *command;

    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }

    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;

    command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
    sprintf(command, "%s %s", v[0], v[1]);

    // Use only one of the followings.
    //system(command);
    execve(v[0], v, NULL);

    return 0;
}
```

```
seed@pcvm471-2:~$ sudo chown root catall
seed@pcvm471-2:~$ sudo chmod 4755 catall
seed@pcvm471-2:~$ catall myprog.c
/* myprog.c */
#include <unistd.h>

int main() {
  sleep(1);
  return 0;
}
seed@pcvm471-2:~$
```

Even after replacing the system() with execve() there is no change as all it does is start a new process but does not give anymore permissions. So, this program still only gives us the access to read a file and not edit files or delete them as we are not root users. So, the abilities with using the system() is shared by execve().