Nick Carter

Lab 1 Report - Environment Variable and Set-UID

CIS 518 02

September 25, 2023

Task 1

Shows the environment variables

```
[09/14/23]seed@VM:~$ printenv PWD
/home/seed
[09/14/23]seed@VM:~$ env | grep PWD
PWD=/home/seed
[09/14/23]seed@VM:~$
```

Exports foo as an environment variable as ‘test string’ and then unsets it.

```
[09/14/23]seed@VM:~$ export foo='test string'
[09/14/23]seed@VM:~$ printenv foo
test string
[09/14/23]seed@VM:~$ unset foo
[09/14/23]seed@VM:~$ printenv foo
[09/14/23]seed@VM:~$
```

Task 2

Step 1:

```
[09/14/23]seed@VM:~$ gcc myprintenv.c
[09/14/23]seed@VM:~$ ls
a.out       Desktop    Music         Pictures  Templates
cap_leak.c  Documents  myenv.c       Public    Videos
catall.c    Downloads  myprintenv.c  Share
[09/14/23]seed@VM:~$ a.out > file
[09/14/23]seed@VM:~$
```

“file” is the child processes.

Step 2:

```
21        // printenv();
22        exit(0);
23      default:  /* parent process */
24         printenv();
```

```
[09/14/23]seed@VM:~$ gcc myprintenv.c
[09/14/23]seed@VM:~$ a.out > file2
```

“file2” is the parent processes.

Step 3:

There are no differences, which means everything was inherited. The child is an exact copy of the parent. fork() creates a child process that inherits all the environment variables from the parent process.

```
[09/14/23]seed@VM:~$ diff file file2
[09/14/23]seed@VM:~$
```

Task 3

Step 1:

There is no output.

```
[09/19/23]seed@VM:~$ gcc myenv.c
[09/19/23]seed@VM:~$ ./a.out
```

Step 2:

Output after making this change includes the environment variables for this process.

```
12   execve("/usr/bin/env", argv, environ);
```

Step 3:

The new program gets its environment variables due to the “environ” argument we added earlier to the execve() function. This argument was NULL before, meaning it would not retrieve the environment variables from any process, giving us the option to inherit the environment variables.

Task 4

system.c contains the program provided in Task 4, I redirected the output to a file called systemtest and printed the contents of the output. This shows the environment variables which shows that using system() will still provide the environment variables, just by executing a different command instead of being directly executed like execve().

```
[09/19/23]seed@VM:~$ gcc -g system.c
[09/20/23]seed@VM:~$ a.out > systemtest
[09/20/23]seed@VM:~$ cat systemtest
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=1945
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
MANAGERPID=1729
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COLORTERM=truecolor
IM_CONFIG_PHASE=1
LOGNAME=seed
JOURNAL_STREAM=9:36099
```

Task 5

Step 1:

Created the program.

```
[09/20/23]seed@VM:~$ touch task5program.c
[09/20/23]seed@VM:~$ vim task5program.c
[09/20/23]seed@VM:~$ cat task5program.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++;
        }
}
```

Step 2:

```
[09/20/23]seed@VM:~$ gcc -g task5program.c
[09/20/23]seed@VM:~$ sudo chown root task5program.c
[09/20/23]seed@VM:~$ sudo chmod 4755 task5program.c
```

Gives the program root privilege and makes it a UID program.

Step 3:

Exported PATH, LD_LIBRARY_PATH, and TASK5.

```
[09/21/23]seed@VM:~$ export PATH=/home/seed:$PATH

export LD_LIBRARY_PATH=/home/seed:$LD_LIBRARY_PAT

[09/21/23]seed@VM:~$ export TASK5='Task 5 env variable'
```

After running the program, we can see that the LD_LIBRARY_PATH environment variable is not inherited.

```
[09/21/23]seed@VM:~$ diff env_result task5_result
43d42
< LD_LIBRARY_PATH=/home/seed:
50c49
< _=/usr/bin/env
---
> _=/home/seed/a.out
```

Task 6

Created a “virus,” edited the environment variable of ls to our virus. Now our executable changes ls to our virus.

```
[09/21/23]seed@VM:~$ touch task6virus.c

[09/21/23]seed@VM:~$ gcc task6virus.c > ls

[09/21/23]seed@VM:~$ ls
a.out       env_result  myls.c        systemtest      test2
cap_leak.c  file        myprintenv.c  task5program.c  Videos
catall.c    file2       Pictures      task5_result
Desktop     ls          Public        task6virus.c
Documents   Music       Share         Templates
Downloads   myenv.c     system.c      test1
[09/21/23]seed@VM:~$ a.out
VIRUS!!![09/21/23]seed@VM:~$
```

Task 7

Step 1:

1. Created the mylib.c program as instructed.

```
[09/21/23]seed@VM:~$ touch mylib.c

[09/22/23]seed@VM:~$ cat mylib.c
#include <stdio.h>
void sleep (int s)
{
        /* If this is invoked by a privileged program,
         * you can do damages here! */
        printf("I am not sleeping!\n");
}
```

2. Compiled the program with the compiling instructions.

```
[09/22/23]seed@VM:~$ gcc -fPIC -g -c mylib.c
[09/22/23]seed@VM:~$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
```

3. Set the LD_PRELOAD environment variable.

```
[09/22/23]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
```

4. Wrote the myprog.c program and compiled it in the same directory as libmylib.so.1.0.1.

```
[09/22/23]seed@VM:~$ touch myprog.c
```

```
[09/22/23]seed@VM:~$ cat myprog.c
/* myprog.c */
#include <unistd.h>
int main()
{
        sleep(1);
        return 0;
}
```

```
[09/22/23]seed@VM:~$ gcc -g myprog.c
```

Step 2:

myprog.c as a regular program, normal user:

```
[09/22/23]seed@VM:~$ a.out
I am not sleeping!
```

myprog.c as Set-UID, normal user:

```
[09/22/23]seed@VM:~$ sudo chown root myprog.c
[09/22/23]seed@VM:~$ sudo chmod 4755 myprog.c
[09/22/23]seed@VM:~$ gcc myprog.c
[09/22/23]seed@VM:~$ a.out
I am not sleeping!
```

myprog.c as Set-UID with LD_PRELOAD exported in root:

```
[09/22/23]seed@VM:~$ sudo chown root myprog.c
[09/22/23]seed@VM:~$ gcc -g myprog.c
[09/22/23]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/22/23]seed@VM:~$ gcc -g myprog.c
[09/22/23]seed@VM:~$ a.out
I am not sleeping!
```

myprog.c as Set-UID user1 program, export LD_PRELOAD in different user:

```
[09/22/23]seed@VM:~$ su user1
Password:
user1@VM:/home/seed$ export LD_PRELOAD=./libmylib.so.1.0.1

user1@VM:/home/seed$ gcc myprog.c
/usr/bin/ld: cannot open output file a.out: Permission denied
collect2: error: ld returned 1 exit status
```

Step 3:

The only different behavior I got was on the last one where I was denied permission which didn't allow me to run the myprog.c program. It is clear that in this instance, the child process was not inherited as we can see the exit status. This means that our LD_PRELOAD environment variable that was exported did not get inherited. The steps for this part were a little confusing so I'm not entirely sure if I did everything correctly.

Task 8

Step 1:

Created and compiled the catall.c program and made it a root-owned Set-UID program. Bob can compromise the integrity of the system and can do more than read the files.

Bob would have to feed a string, as I've done in the screenshot, and two shell commands will run. One of these provides Bob with a root shell which would give him root access to modify what he wants. They state that the SEED VM has a countermeasure for this, but we can see that where we are given a normal shell after feeding a string, that would normally be a root shell.

```
[09/22/23]seed@VM:~$ vim catall.c
[09/22/23]seed@VM:~$ gcc -g catall.c
[09/22/23]seed@VM:~$ cat catall.c
#include <stdio.h>
#include <stdlib.h>
#include <string.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 following
        system(command);
        //execve(v[0], v, NULL);

        return 0;
}
[09/22/23]seed@VM:~$ sudo chown root catall.c
[09/22/23]seed@VM:~$ sudo chmod 4755 catall.c
```

```
[09/22/23]seed@VM:~$ a.out "aa;/bin/sh"
/bin/cat: aa: No such file or directory
$
```

Step 2:

Commented out system(command) and uncommented execve, compiled catall.c, made it root and Set-UID, and attempted the same attack as the prior step.

The attack does not work because execve does not ask a shell program for the command, but rather asks the operating system directly. This means the string that is

fed is treated as an argument and not a command and simply states that the

file/directory does not exist.

```
//system(command);
execve(v[0], v, NULL);
```

Doesn't exist.

```
[09/22/23]seed@VM:~$ vim catall.c
[09/22/23]seed@VM:~$ gcc -g catall.c
catall.c: In function ‘main’:
catall.c:21:2: warning: implicit declaration of fu
   21 |  execve(v[0], v, NULL);
      |  ^~~~~~
[09/22/23]seed@VM:~$ sudo chown root catall.c
[09/22/23]seed@VM:~$ sudo chmod 4755 catall.c
[09/22/23]seed@VM:~$ a.out "aa;/bin/sh"
/bin/cat: 'aa;/bin/sh': No such file or directory
[09/22/23]seed@VM:~$
```

Task 9

Created cap_leak.c

```
[09/22/23]seed@VM:~$ touch cap_leak.c
```

Created the file /etc/zzz

```
[09/22/23]seed@VM:~$ sudo touch /etc/zzz
```

```
[09/22/23]seed@VM:~$ cat cap_leak.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>


void main()
{
        int fd;
        char *v[2];

        /* Assume that /etc/zzz is an important system file,
        * and it is owned by root with permission 0644.
        * Before running this program, you should create
        * the file /etc/zzz first. */
        fd = open("/etc/zzz", O_RDWR | O_APPEND);
        if (fd == -1) {
                printf("Cannot open /etc/zzz\n");
                exit(0);
        }

        // Print out the file descriptor value
        printf("fd is %d\n", fd);

        // Permanently disable the privilege by making the
        // effective uid the same as the real uid
        setuid(getuid());

        // Execute /bin/sh
        v[0] = "/bin/sh"; v[1] = 0;
        execve(v[0], v, 0);
}
```

Compiled it.

```
[09/22/23]seed@VM:~$ gcc -g cap_leak.c
```

Gave root and Set-UID

```
[09/22/23]seed@VM:~$ sudo chown root cap_leak.c
[09/22/23]seed@VM:~$ sudo chmod 4755 cap_leak.c
```

At first you cannot edit the /etc/zzz file because of the root user and permissions it is given. However, after running our program we can see the file descriptor (3) which then allows us to write to this file if we direct it towards the file descriptor (3).

```
fd is 3

echo this file has been edited >& 3

[09/22/23]seed@VM:~$ cat /etc/zzz
this file has been edited
```