EECS565 Intro to Computer and Information Security

Mini Project 3

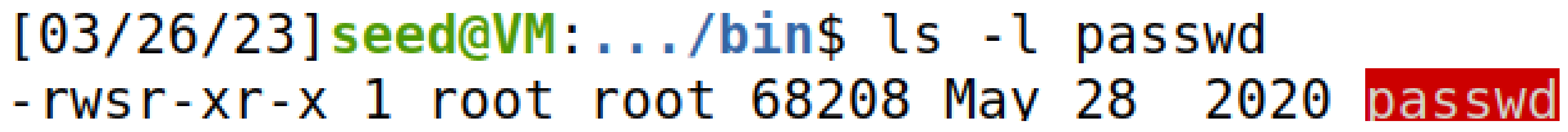
Set-UID Program Vulnerability

Outline

- **Task 1: Explore SetUID Programs**
- **Task 2: Exploring Environment Variables**
  - **Manipulating** Environment Variables
  - Passing Environment Variables **from Parent Process to Child Process**
  - Environment Variables and **execve()**
  - Environment Variables and **system()**
- **Task 3: Environment Variables and Set-UID Programs**
  - Use Environment Variables to Affect Set-UID Programs
  - The PATH Environment Variable

Task 1: Explore SetUID Programs

- Explore a few Set-UID programs: *passwd, chsh,* and *sudo*
- Run these programs in their default location (/bin or /usr/bin directories)
- Copy the program to the directory of your choice (e.g., Desktop or Downloads)
  - Hint: *cp* command to copy the file to a new directory
- Run these programs again
  - Hint: *ls –l filename* to check the permissions of one file.



```
[03/26/23]seed@VM:.../bin$ ls -l passwd
-rwsr-xr-x 1 root root 68208 May 28  2020 passwd
```

  - Did the programs work appropriately in both cases?

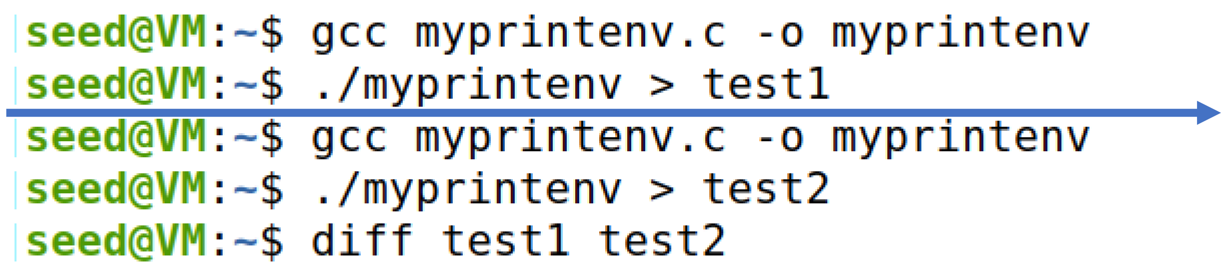
Task 2: Exploring Environment Variables

2.1 Manipulating Environment Variables

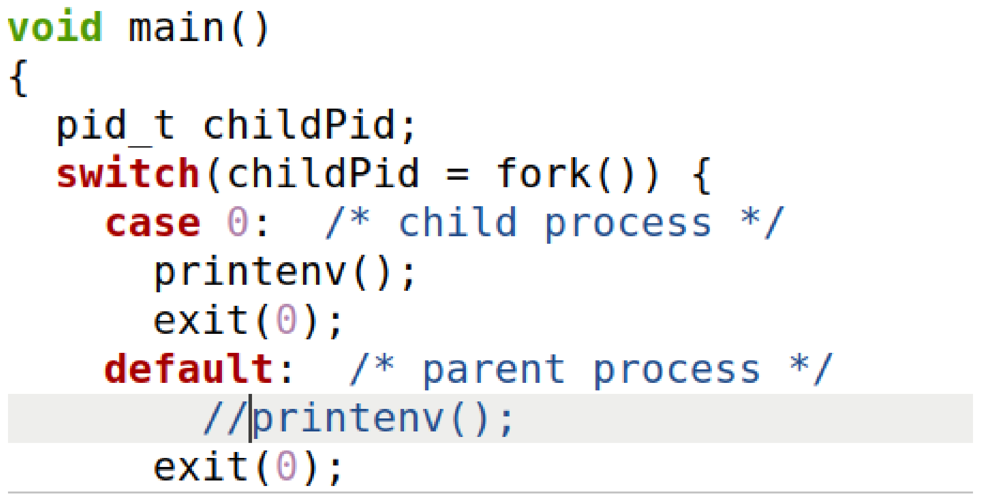
- Set an environment variable
  - *export aaa=bbb*.
- Unset an environment variable
  - *unset aaa*
- Show all the Environment Variables
  - *printenv*
- Show a specific Environment Variable aaa
  - *printenv aaa* or *env | grep aaa*

Task 2: Exploring Environment Variables

2.2 Passing Environment Variables from Parent Process to Child Process



```
seed@VM:~$ gcc myprintenv.c -o myprintenv
seed@VM:~$ ./myprintenv > test1
seed@VM:~$ gcc myprintenv.c -o myprintenv
seed@VM:~$ ./myprintenv > test2
seed@VM:~$ diff test1 test2
```



```
void main()
{
  pid_t childPid;
  switch(childPid = fork()) {
    case 0:  /* child process */
      printenv();
      exit(0);
    default:  /* parent process */
        //printenv();
      exit(0);
```

Hints:

- Keep running in the same terminal.
- Use the same program name.
- *Diff* command to compare

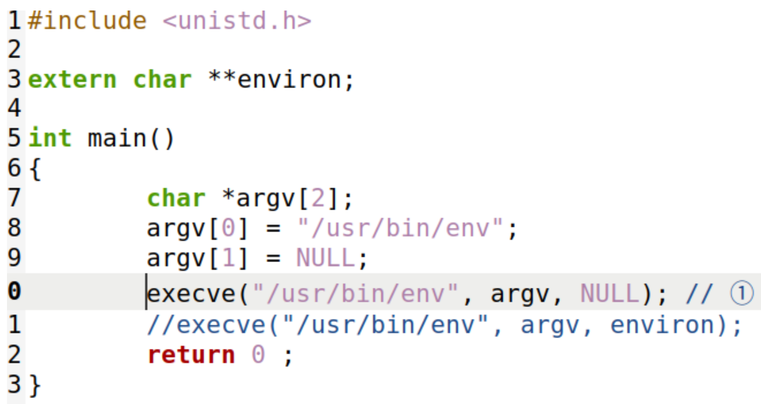
Hints:

- Only comment one printenv() in turns.

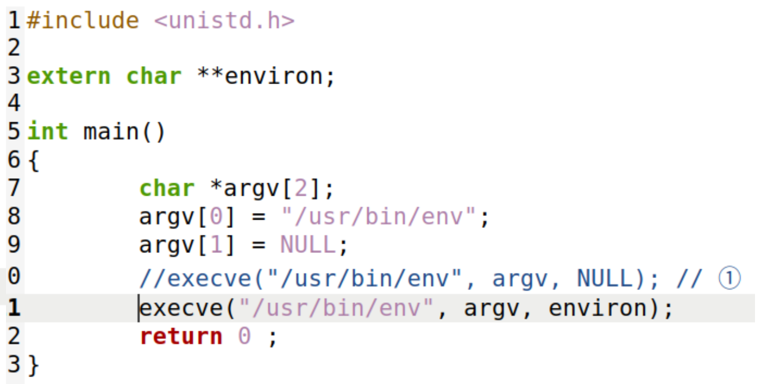
Task 2: Exploring Environment Variables

2.3 Environment Variables and execve()

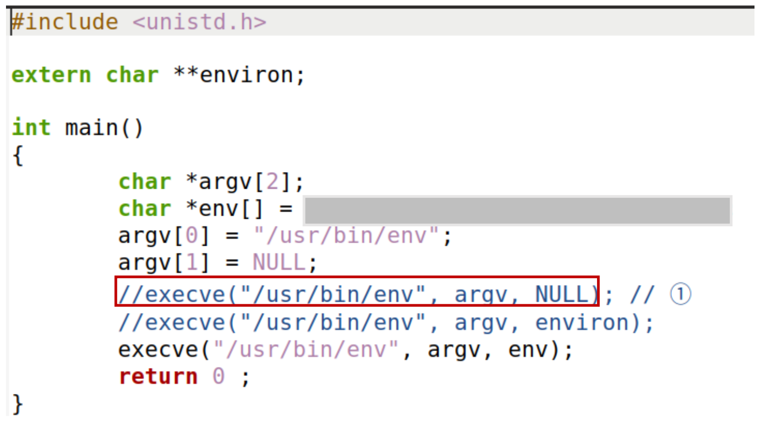
- ❑ Directly run the execve(), what is the result?
- ❑ How about that an array of string pointers is passed with variables?
- ❑ How can we control the environment of the child by passing our own null terminated array of char *'s?



```
#include <unistd.h>

extern char **environ;

int main()
{
        char *argv[2];
        argv[0] = "/usr/bin/env";
        argv[1] = NULL;
        execve("/usr/bin/env", argv, NULL); // ①
        //execve("/usr/bin/env", argv, environ);
        return 0 ;
}
```



```
#include <unistd.h>

extern char **environ;

int main()
{
        char *argv[2];
        argv[0] = "/usr/bin/env";
        argv[1] = NULL;
        //execve("/usr/bin/env", argv, NULL); // ①
        execve("/usr/bin/env", argv, environ);
        return 0 ;
}
```



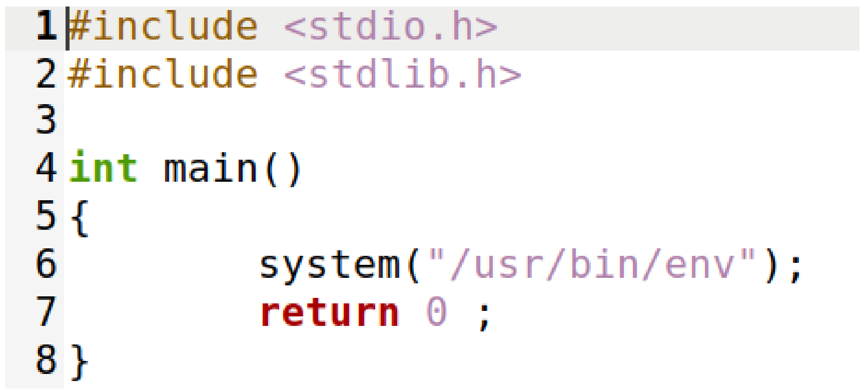
```
#include <unistd.h>

extern char **environ;

int main()
{
        char *argv[2];
        char *env[] = [redacted]
        argv[0] = "/usr/bin/env";
        argv[1] = NULL;
        //execve("/usr/bin/env", argv, NULL); // ①
        //execve("/usr/bin/env", argv, environ);
        execve("/usr/bin/env", argv, env);
        return 0 ;
}
```

Task 2: Exploring Environment Variables

2.4 Environment Variables and system()

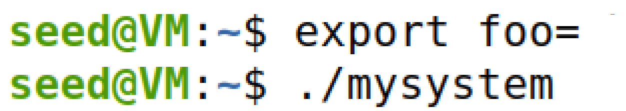
- The *system()* function uses execl() to execute /bin/sh. To do so, *execl()* calls *execve()* and pass the environment variables array to it.

- mysystem.c: call *system()* to get the *env.*
- *Compile and run the program to show the result.*



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

int main()
{
        system("/usr/bin/env");
        return 0 ;
}
```

- Set a customized environment variable and rerun the program. Show and analyze the result.

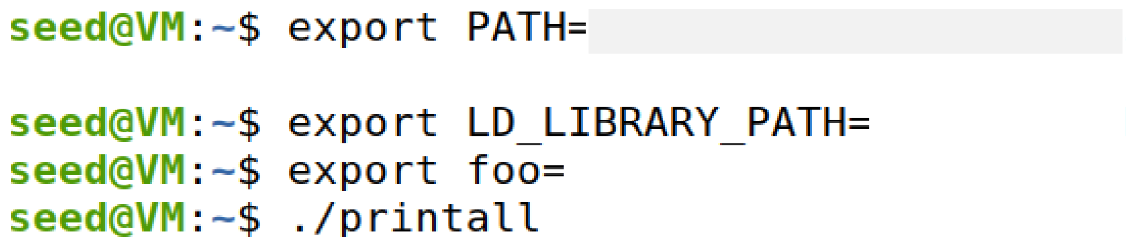


```
seed@VM:~$ export foo=
seed@VM:~$ ./mysystem
```

Task 3: Environment Variables and Set-UID Programs

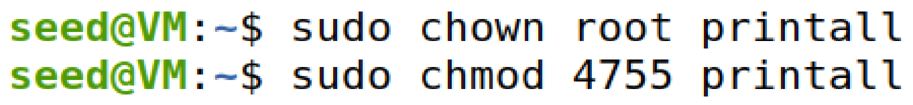
3.1 Use Environment Variables to Affect Set-UID Programs

- Set the Environment Variables in a normal user account(SEED)



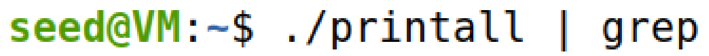
```
seed@VM:~$ export PATH=

seed@VM:~$ export LD_LIBRARY_PATH=
seed@VM:~$ export foo=
seed@VM:~$ ./printall
```

- Change the program’s ownership to root and make it a Set-UID Program



```
seed@VM:~$ sudo chown root printall
seed@VM:~$ sudo chmod 4755 printall
```

- Check the values of these Environment Variables



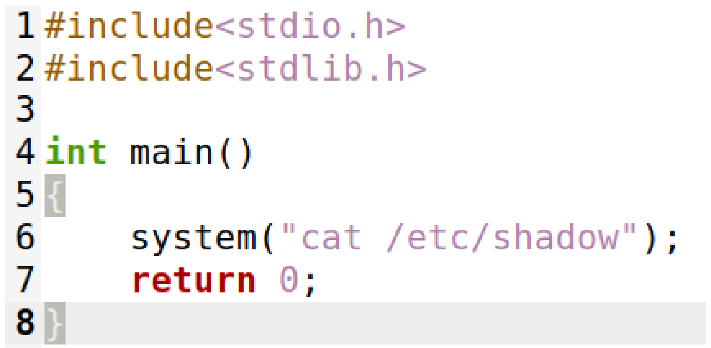
```
seed@VM:~$ ./printall | grep
```

- PATH
- LD_LAIBRARY_PATH
- foo

Task 3: Environment Variables and Set-UID Programs

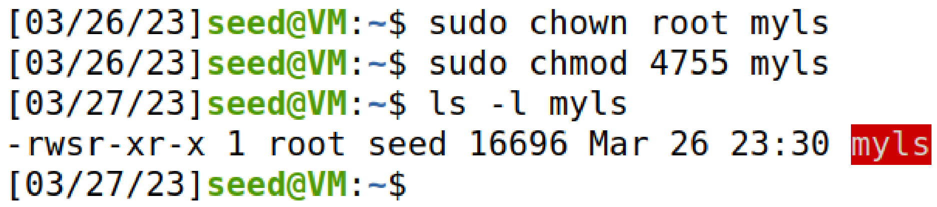
3.2 The PATH Environment Variable

- Can you get the Set-UID program to run a malicious command?



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

int main()
{
    system("cat /etc/shadow");
    return 0;
}
```

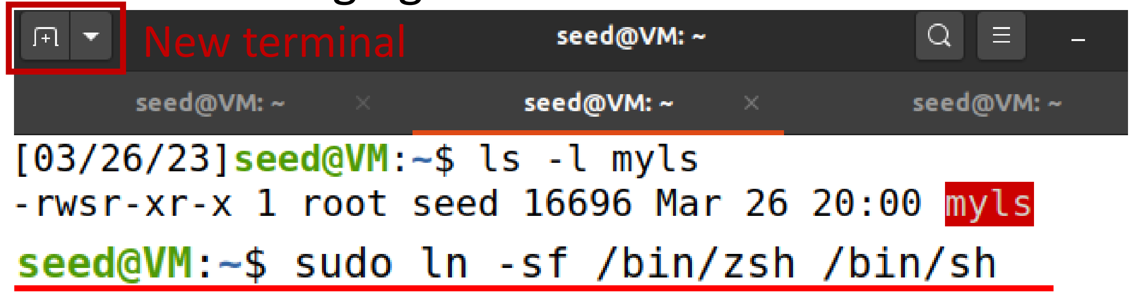
- Don't forget:



```
[03/26/23]seed@VM:~$ sudo chown root myls
[03/26/23]seed@VM:~$ sudo chmod 4755 myls
[03/27/23]seed@VM:~$ ls -l myls
-rwsr-xr-x 1 root seed 16696 Mar 26 23:30 myls
[03/27/23]seed@VM:~$
```

Hints:

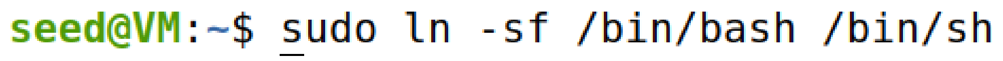
- After changing the shell, open a new terminal to make the changing work.



```
[03/26/23]seed@VM:~$ ls -l myls
-rwsr-xr-x 1 root seed 16696 Mar 26 20:00 myls
seed@VM:~$ sudo ln -sf /bin/zsh /bin/sh
```

Change shell to zsh

- The changing is permanent, to change back to bash, use the command below:



```
seed@VM:~$ sudo ln -sf /bin/bash /bin/sh
```

Acknowledgements

- We thank the SEED Lab for sharing the lab design and slides.