Environment Variable and `Set-UID` Program Lab

Xinyi Li

February 12, 2020

Task 1

```
seed@VM:~$ printenv
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
...
```

Task 2

I find that their environment variables are exactly the same.

Task 3

When the third arguments set as `NULL`, no output. But when it turns to `environ`, all environment variables are printed.

Because `environ` is the **user environment** which holds those environment variables.

Task 4

Yes, but with different orders somehow.

Task 5

Yes, all environment variables set above are changed (if all commands are done in the **same** user shell)

Task 6

use `gcc` to compile the code with the argument `-o ls`, then append current folder into the environment variable `PATH`.

Task 7

- Swtich to `root`: `su`
- No need to actually add user `user1`, just type `sudo chown user1 ./myprog.out` under current user shell.
- regular program, normal user: **no sleep**
- `Set-UID` root program, normal user: actually sleep
- `Set-UID` root program, root user: **no sleep**
- `Set-UID user1` program, **another**(non-root,non-`user1`) normal user: actually sleep

Add some codes in original `myprog.c`:

```
#include <unistd.h>
#include <stdio.h>
#include <string.h>
extern char **environ;

void printenv(const char *var_name)
{
    int i = 0;
    while (environ[i] != NULL)
    {
        if (strstr(environ[i], var_name) != NULL)
        {
            printf("%s\n", environ[i]);
            break;
        }
        else
        {
            i++;
        }
    }
}

int main()
{
    printenv("LD_PRELOAD");
    sleep(1);
```

```
    return 0;
}
```

So when `myprog.out` runs we can see the real value of `LD_PRELOAD` in the environment where the process is called. (Sometimes `stdout` is the shell of the actual owner so that nothing will be printed on this shell)

Besides, use `echo $LD_PRELOAD` to print the environment of the user shell.

It shows that the process runs in the environment of the owner rather than that of the shell user.

When the process runs with `LD_PRELOAD=./libmylib.so.1.0.1`, it will execute what defined in `mylib.c`. The `Set-UID` just runs in the environment of the owner instead of the caller.

Task 8

Step 1

Yes, I can. Assumed the file to view named `example1` and the file to remove maliciously is `/path/critical`, I can use the program (assume it is `a.out`) as

```
seed@~$ ./a.out "example1; /bin/sh"
$ rm  /path/critical
```

Step 2

No. Using the same command will, you will get the error message `"No such file or directory"`. `execve()` simply treat the second argument as arguments of a command rather than invoking `/bin/sh` as `system()` do.

Task 9

The file is modified. The child process forked from the parent can also inherit the privileges to manipulate some critical files even if the parent process is terminated.