Assignment 3

Comp 4580

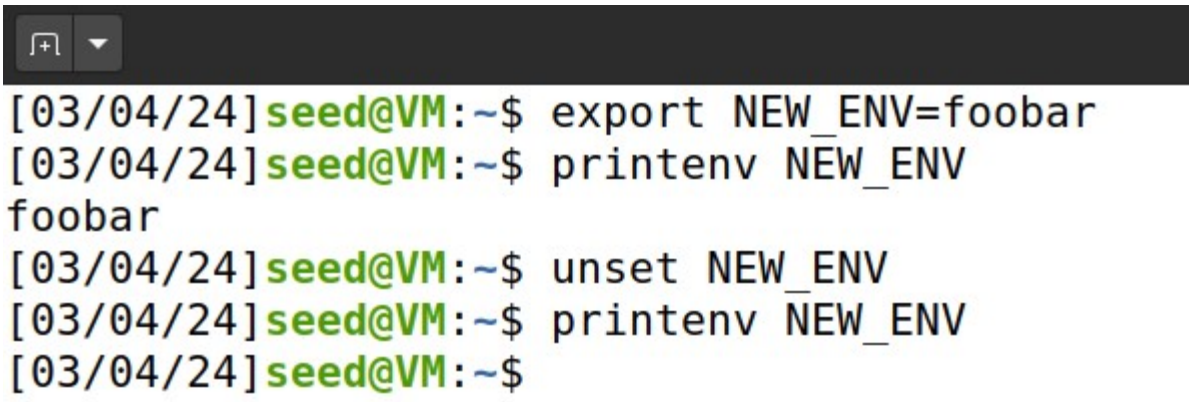
Ryan Dotzlaw - 7881954

Do tasks 1, 2, 5, and 6

Task 1: Manipulating Environment Variables

Listing the environment variables can be done with the `env` or `printenv` bash commands, like so:

Setting or modifying an environment variable can be done with the `export` command, and removing one can be done with the `unset` command, like so:



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

Using the code given in `myprintenv.c`, which is as follows:

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

extern char **environ;

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

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

Then, compiling the code and running it gives the following output:

```
[03/04/24]seed@VM:~/.../A3$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1854,unix/VM:/tmp/.ICE-unix/1854
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=1811
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/A3
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
```

We can see that this even includes environment variable that I've set:

```
[03/04/24]seed@VM:~/.../A3$ export NEW_ENV=foobar
[03/04/24]seed@VM:~/.../A3$ ./a.out | grep NEW_ENV
NEW_ENV=foobar
```

From there, we modify the code by commenting out the `printenv()` command in `case 0`, and uncommenting the `printenv()` in `case 1`, then compiling and running the code again.

The output is as follows:

```
[03/04/24]seed@VM:~/.../A3$ gcc myprintenv.c
[03/04/24]seed@VM:~/.../A3$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1854,unix/VM:/tmp/.ICE-unix/1854
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=1811
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/A3
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
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
```

```
OLDPWD=/home/seed/Desktop
_=./a.out
[03/04/24]seed@VM:~/.../A3$ ./a.out | grep NEW_ENV
NEW_ENV=foobar
[03/04/24]seed@VM:~/.../A3$
```

After running again and piping the results into `out1` and `out2` respectivly, we can use the diff command, with the `-s` flag to tell us if these files are identical.

```
[03/04/24]seed@VM:~/.../A3$ diff out1 out2 -s
Files out1 and out2 are identical
[03/04/24]seed@VM:~/.../A3$
```

From the output we can see that the files are identical.

This means that both the parent process and the child process that forked off share the same environment variables.

If there was a vunerable parent program where an attacker could change the `PATH` environment variable (perhaps by exploiting a buffer overflow or similar attack), and a child process that ran some bash command like `dir`, then an attacker could change the `PATH` variable in the parent process to point to a directory with a malicious program called `dir`.

This would result in the child process inheriting the `PATH` env and executing the malicious program unintentionally.

This would be far less challenging than trying to cram an entire program into a buffer overflow attack.

Task 5: Environment Variable and Set-UID Programs

We create the program `env.c` with the following code:

```
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
int main()
{
        int i = 0;
        while (environ[i] != NULL) {
                printf("%s\n", environ[i]);
                i++;
        }
}
```

This program prints out all environment variables.

From there we compile the program with this command:

```
gcc env.c -o t5
```

and modify it with `chown` and `chmod` to make it a `Set-UID` program with `root` as it's owner.

```
[03/04/24]seed@VM:~/.../A3$ sudo chown root t5
[03/04/24]seed@VM:~/.../A3$ sudo chmod 4755 t5
[03/04/24]seed@VM:~/.../A3$
```

Then we change several environment variables as follows:

```
[03/04/24]seed@VM:~/.../A3$ export PATH=real/path:more/path:/bin:/usr/bin:.
[03/04/24]seed@VM:~/.../A3$ export LD_LIBRARY_PATH=/ld/library/path:.
[03/04/24]seed@VM:~/.../A3$ export NEW_ENV=foobar
[03/04/24]seed@VM:~/.../A3$
```

Now we run the program, getting the output:

```
[03/04/24]seed@VM:~/.../A3$ ./t5
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1854,unix/VM:/tmp/.ICE-unix/1854
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=1811
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/A3
```

Clearly an output of environment variables, however, notice the following:

```
[03/04/24]seed@VM:~/.../A3$ printenv | grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=/ld/library/path:.
[03/04/24]seed@VM:~/.../A3$ ./t5 | grep PATH
WINDOWPATH=2
PATH=real/path:more/path:/bin:/usr/bin:.
[03/04/24]seed@VM:~/.../A3$ ./t5 | grep NEW
NEW_ENV=foobar
[03/04/24]seed@VM:~/.../A3$ ./t5 | grep LD_LIBRARY_PATH
[03/04/24]seed@VM:~/.../A3$
```

From this we can see that some environment variables are passed from the user's envs to a Set-UID process, but not all of them.

Specifically `LD_LIBRARY_PATH` was not passed to the program.

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

We create a program called `just_ls.c` with the following code:

```
int main()
{
        system("ls");
        return 0;
}
```

From there we compile it, change it's owner to root, and make it a Set-UID program:

```
[03/04/24]seed@VM:~/.../A3$ sudo chown root just_ls
[03/04/24]seed@VM:~/.../A3$ sudo chmod 4755 just_ls
[03/04/24]seed@VM:~/.../A3$
```

Running it gives the result:

```
[03/04/24]seed@VM:~/.../A3$ ./just_ls
a.out  cap_leak.c  catall.c  env.c  just_ls  just_ls.c  myenv.c  myprintenv.c  out1  out2  t5
[03/04/24]seed@VM:~/.../A3$
```

Then, by modifying the `PATH` env, we can change where the `system(ls)` looks for the `ls` program.

We modify the path so it looks in the `home/seed` directory first; our user home directory.

```
[03/04/24]seed@VM:~/.../A3$ export PATH=/home/seed:$PATH
[03/04/24]seed@VM:~/.../A3$
```

Then we create a 'malicious' program and compile it into an executable called `ls` and drop it in the `home/seed` directory.

The malicious program has the following code:

```
#include <stdio.h>
#include <unistd.h>
int main()
{
        if(geteuid()){
                printf("Nothing to see here, just a normal user program\n");
        } else {
                printf("You got hacked!!!\n");
        }
        return 0;
}
```

The program will display "You got hacked!!!" if the effective user id is 0, AKA root.

```
[03/04/24]seed@VM:~/.../A3$ gcc hack.c -o ls
[03/04/24]seed@VM:~/.../A3$ ./ls
Nothing to see here, just a normal user program
[03/04/24]seed@VM:~/.../A3$
```

Then when we execute our `just_ls` program and the output is now as follows:

```
[03/04/24]seed@VM:~/.../A3$ ./just_ls
You got hacked!!![03/04/24]seed@VM:~/.../A3$
```

This means that we were able to execute a custom program by changing the `PATH` env.

Additionally, since `just_ls` was a root owned Set-UID program, the effective permissions for our malicious program were high enough to 'hack' the user.