Environment Variable and Set-UID Program Lab
William Wadsworth, Rafik Tarbari

9.12.22

Introduction

In this lab, we experiment with environment variables and see how they affect programs. Specifically, how environment variables are shared (or not) between parent and child processes, with privileged (`Set-UID`) programs and non-privileged programs alike. All programs were compiled using `gcc` and output was redirected into their respective files using i/o redirection.

2.1 Task 1: Manipulating Environment Variables

We used `printenv` (or `env`, they produce the same list) to compile a list of all environment variables. Alternatively, to find a specific environment variable, you can do "`printenv [environment variable]`" or "`env | grep [environment variable]`". The commands *`export`* and *`unset`* can be used to set or unset environment variables.

```
[09/12/22]seed@VM:~/.../lab1$ env
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/4342,unix/VM:/tmp/.ICE-unix/4342
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=4048
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/systemsec/lab1
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
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:bd=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=01;31:*
.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01
;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;3
1:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=
01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.
```

```
35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.m
p4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:
*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;
35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a
=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*
.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/6f76005e_4028_4bb1_8675_062ada991a6b
INVOCATION_ID=96f9b7df992c4ff086ce2780e07263ac
MANAGERPID=3661
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.99
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:38769
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:
.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
_=/usr/bin/env
OLDPWD=/home/seed
[09/12/22]seed@VM:~/.../lab1$
```

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

In this task, we determine how a child process inherits its environment variables from its parent.

We start by compiling `myprint.c` (shown to the right). In `myprint.c`, in the switch/case in the main program function, case 0 is labeled the child process, and the default case is the parent process. We compile and run the program first with case 0, then compile and run the program again using the default case by uncommenting the default case and commenting out case 0.

Lastly, we used the `diff` command to see the difference between the two output files. When doing so, `diff` returns nothing, signifying that the child process inherits its environment variables from its parent.

```c
#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);
  }
}
```

```
[09/12/22]seed@VM:~/Labsetup$ diff file file2
[09/12/22]seed@VM:~/Labsetup$
```

2.3 Task 3: Environment Variables and `execve()`

In this task, we use the `execve()` command to see how environment variables change, specifically if they are inherited automatically.

To start, we compile and run `myenv.c`, which is shown below:

```
 1 #include <unistd.h>
 2
 3 extern char **environ;
 4
 5 int main()
 6 {
 7   char *argv[2];
 8
 9   argv[0] = "/usr/bin/env";
10   argv[1] = NULL;
11
12   execve("/usr/bin/env", argv, NULL);
13
14   return 0 ;
15 }
16
~
```

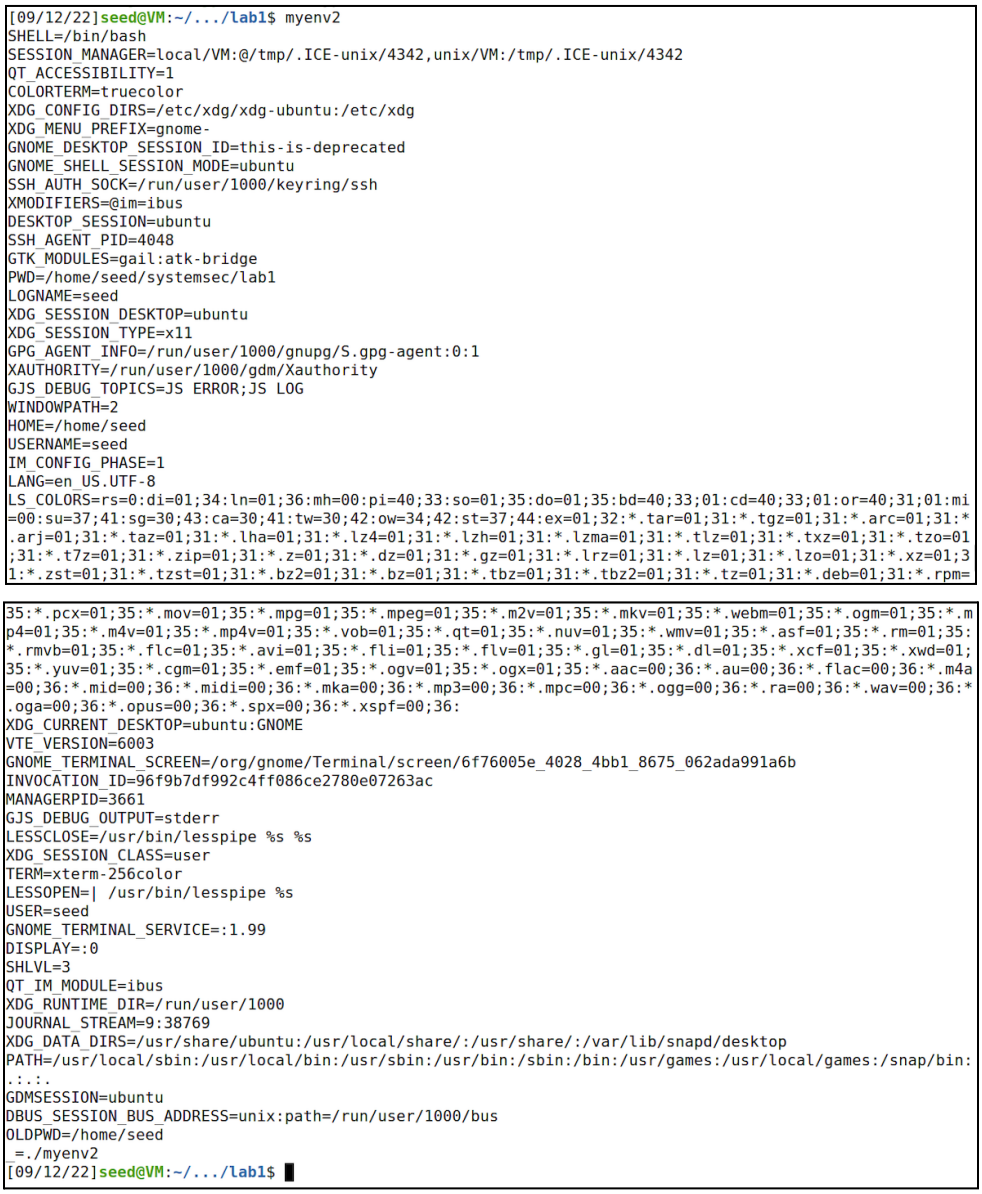
which gives us the following output:

```
[09/12/22]seed@VM:~/.../lab1$ myenv1
[09/12/22]seed@VM:~/.../lab1$
```

Next, in the `execve()` function in line 12 we replace `NULL` with `environ`:

```
 1 #include <unistd.h>
 2
 3 extern char **environ;
 4
 5 int main()
 6 {
 7   char *argv[2];
 8
 9   argv[0] = "/usr/bin/env";
10   argv[1] = NULL;
11
12   execve("/usr/bin/env", argv, environ);
13
14   return 0 ;
15 }
16
~
```

which gives us the following output:



```
[09/12/22]seed@VM:~/.../lab1$ myenv2
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/4342,unix/VM:/tmp/.ICE-unix/4342
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=4048
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/systemsec/lab1
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
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:bd=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=01;31:*
.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01
;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;3
1:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=
35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.m
p4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:
*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;
35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a
=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*
.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/6f76005e_4028_4bb1_8675_062ada991a6b
INVOCATION_ID=96f9b7df992c4ff086ce2780e07263ac
MANAGERPID=3661
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.99
DISPLAY=:0
SHLVL=3
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:38769
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:
.:.:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/seed
_=./myenv2
[09/12/22]seed@VM:~/.../lab1$
```

As we passed `environ` to `execve()` which is a pointer pointing to the environment, the child process has inherited the environment variables of the parent process.

2.4 Task 4: Environment Variables and `system()`

In this task, we verify that the `system()` function passes the environment variables of the calling process to the new program `/bin/sh`.

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

The output indicates that we indeed get the environment variables.

2.5 Task 5: Environment Variables and `Set-UID` Programs

In this task, we explore how `Set-UID` programs affect environment variables.

```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++;
    }
}
```

```
SHLVL=1
LD_LIBRARY_PATH=bar1
HOME=/home/seed
```

```
ANY_NAME=bar2
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
WINDOWPATH=2
PATH=/home/seed/.local/bin:/usr/local/sbin:/usr/local/bin:/
usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/s
nap/bin:.
```

All the environment variables `PATH`, `LD_LIBRARY_PATH`, and `ANY_NAME` have been passed to the child process.

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

To start this task, we set `/home/seed` to the beginning of the `PATH` environment variable by entering: `export PATH=/home/seed:$PATH`. The program below is an example of the system running the command in the `system()` function (in this case, `cat /etc/shadow`), rather than the default function located in `/bin/cat`:

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

```
[09/12/22]seed@VM:~/Labsetup$ sudo chown root foo2
[09/12/22]seed@VM:~/Labsetup$ sudo chmod 4755 foo2
[09/12/22]seed@VM:~/Labsetup$ cat /etc/shadow
cat: /etc/shadow: Permission denied
[09/12/22]seed@VM:~/Labsetup$ ./foo2
cat: /etc/shadow: Permission denied
```

In the first figure, we change the command to output the content of the shadow file which can only be read as root. We get an error message “Permission Denied” which means our program is not running with root privilege

After linking `/bin/zsh` to `/bin/sh`, we are able to run our malicious program:

```
[09/12/22]seed@VM:~/Labsetup$ sudo chown root foo2
[09/12/22]seed@VM:~/Labsetup$ sudo chmod 4755 foo2
[09/12/22]seed@VM:~/Labsetup$ ./foo2
root:!:18590:0:99999:7:::
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
sync:*:18474:0:99999:7:::
games:*:18474:0:99999:7:::
man:*:18474:0:99999:7:::
lp:*:18474:0:99999:7:::
mail:*:18474:0:99999:7:::
news:*:18474:0:99999:7:::
uucp:*:18474:0:99999:7:::
proxy:*:18474:0:99999:7:::
www-data:*:18474:0:99999:7:::
backup:*:18474:0:99999:7:::
list:*:18474:0:99999:7:::
irc:*:18474:0:99999:7:::
gnats:*:18474:0:99999:7:::
nobody:*:18474:0:99999:7:::
systemd-network:*:18474:0:99999:7:::
systemd-resolve:*:18474:0:99999:7:::
systemd-timesync:*:18474:0:99999:7:::
messagebus:*:18474:0:99999:7:::
syslog:*:18474:0:99999:7:::
_apt:*:18474:0:99999:7:::
tss:*:18474:0:99999:7:::
uuidd:*:18474:0:99999:7:::
tcpdump:*:18474:0:99999:7:::
avahi-autoipd:*:18474:0:99999:7:::
usbmux:*:18474:0:99999:7:::
rtkit:*:18474:0:99999:7:::
```

2.7 Task 7: The LD_PRELOAD Environment Variable and Set-UID Programs

In this task, we create a dynamic link library called `mylib.c`:

```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");
}
```

After compiling the above program in a specific way (`gcc -fPIC -g -c mylib.c` then `gcc -shared -o libmylib.so.1.0.1 mylib.o -lc`), we set `LD_PRELOAD=./libmylib.so.1.0.1`. Next, we compile the program below in the same directory as our dynamic link library and run it under the conditions shown below the code:

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

- Regular Program, run as normal user

```
[09/12/22]seed@VM:~/Labsetup$ gcc myprog.c
[09/12/22]seed@VM:~/Labsetup$ ./a.out
I am not sleeping!
[09/12/22]seed@VM:~/Labsetup$
```

- `Set-UID` root Program, run as normal user:

```
[09/12/22]seed@VM:~/Labsetup$ sudo chown root a.out
[09/12/22]seed@VM:~/Labsetup$ sudo chmod 4755 a.out
[09/12/22]seed@VM:~/Labsetup$ ./a.out
[09/12/22]seed@VM:~/Labsetup$
```

- `Set-UID` root Program, `export LD_PRELOAD` in root

```
[09/12/22]seed@VM:~/Labsetup$ sudo chown root a.out
[09/12/22]seed@VM:~/Labsetup$ sudo chmod 4755 a.out
[09/12/22]seed@VM:~/Labsetup$ $LD_PRELOAD
[09/12/22]seed@VM:~/Labsetup$ echo $LD_PRELOAD

[09/12/22]seed@VM:~/Labsetup$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/12/22]seed@VM:~/Labsetup$ ./a.out
[09/12/22]seed@VM:~/Labsetup$
```

- `Set-UID` user1 Program, `export LD_RELOAD` in root

```
[09/12/22]seed@VM:~/Labsetup$ sudo chown cyberraf a.out
[09/12/22]seed@VM:~/Labsetup$ sudo chmod 4755 a.out
[09/12/22]seed@VM:~/Labsetup$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/12/22]seed@VM:~/Labsetup$ ./a.out
```

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

In this task, we set the following program as a `Set-UID`, root-owned program.

If a user without root privileges tries to do anything with a file that they do not have access to (for example, any normal user trying to access or manipulate anything in the `/etc/shadow` directory), they would be blocked due to a lack of permissions. However, when we comment out `system()` on line 22 and uncomment line 23, the program outputs the file specified when running the program.

```
1  #include <unistd.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  int main(int argc, char *argv[])
7  {
8    char *v[3];
9    char *command;
10
11   if(argc < 2) {
12     printf("Please type a file name.\n");
13     return 1;
14   }
15
16   v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;
17
18   command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
19   sprintf(command, "%s %s", v[0], v[1]);
20
21   // Use only one of the followings.
22   system(command);
23   // execve(v[0], v, NULL);
24
25   return 0 ;
26 }
```

2.9 Task 9: Capability Leaking

```
[09/12/22]seed@VM:~/Labsetup$ gcc cap_leak.c
[09/12/22]seed@VM:~/Labsetup$ ./a.out
Cannot open /etc/zzz
[09/12/22]seed@VM:~/Labsetup$ sudo chown root a.out
[09/12/22]seed@VM:~/Labsetup$ sudo chmod 4755 a.out
[09/12/22]seed@VM:~/Labsetup$ ./a.out
fd is 3
```

Once we compile the program into a.out, we change its `Set-UID` to root and the permission to 4755.

When run, it opens a shell that helps to exploit the vulnerability. Once more, we execute the program `a.out echo` into `/etc/zzz`.

```
$ ./a.out
fd is 4
$ echo mmmmmm-hhhhh >& 4
$ cat /etc/zzz
mmmmmm-hhhhh
$
```

Summary

In a nutshell, we explored the vulnerabilities of environment variables. We've learned how to set and unset/export environment variables from parent to child processes. We saw the difference between the use of `execve()` and `system()` to pass environment variables and changing the `Set-UID` and the permission of the program to get root access to the system.