SECURE CODING LAB
ENVIRONMENT VARIABLES

Accessing Environment Variables:

```
#include<stdio.h>

extern char** environ;
void main(int argc,char* argv[],char*envp[])
{
        int i=0;
        while(environ[i]!=NULL)
        {
                printf("%s\n",environ[i++]);
        }
}
```

```
tousif@TousifVM:~/scenv$ gcc environ.c
tousif@TousifVM:~/scenv$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/TousifVM:@/tmp/.ICE-unix/1190,unix/TousifVM:/
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LANGUAGE=en_IN:en
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
GTK_MODULES=gail:atk-bridge
PWD=/home/tousif/scenv
LOGNAME=tousif
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1207
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.III0V1
HOME=/home/tousif
USERNAME=tousif
```

execve() and environment variables :

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

extern char** environ;
void main(int argc,char* argv[],char* envp[])
{
        int i=0;
        char* v[2];
        char* newenv[3];
        if(argc<2) return;
        v[0]="/usr/bin/env";
        v[1]=NULL;
        newenv[0]="AAA=aaa";
        newenv[1]="BBB=bbb";
        newenv[2]=NULL;
        switch(argv[1][0])
        {
                case '1': execve(v[0],v,NULL);
                case '2': execve(v[0],v,newenv);
                case '3': execve(v[0],v,environ);
                default: execve(v[0],v,NULL);
        }
}
```

```
tousif@TousifVM:~/scenv$ gcc execve.c
tousif@TousifVM:~/scenv$ ./a.out 1
tousif@TousifVM:~/scenv$ ./a.out 2
AAA=aaa
BBB=bbb
tousif@TousifVM:~/scenv$ ./a.out 3
SHELL=/bin/bash
SESSION_MANAGER=local/TousifVM:@/tmp/.ICE-unix/1190,unix/TousifVM:/
x/1190
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
SSH_AGENT_LAUNCHER=gnome-keyring
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LANGUAGE=en_IN:en
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
GTK_MODULES=gail:atk-bridge
PWD=/home/tousif/scenv
LOGNAME=tousif
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1207
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.III0V1
HOME=/home/tousif
USERNAME=tousif
```

Shell Variables :

Internal variables used by shell.

```
tousif@TousifVM:~/scenv$ name=tousif
tousif@TousifVM:~/scenv$ echo $name
tousif
tousif@TousifVM:~/scenv$ unset name
tousif@TousifVM:~/scenv$ echo $name

tousif@TousifVM:~/scenv$
```

Shell variables & environment variables :

```
tousif@TousifVM:~/scenv$ strings /proc/$$/environ | grep LOGNAME
LOGNAME=tousif
tousif@TousifVM:~/scenv$ LOGNAME2=alice
tousif@TousifVM:~/scenv$ export LOGNAME3=bob
tousif@TousifVM:~/scenv$ env | grep LOGNAME
LOGNAME=tousif
LOGNAME3=bob
tousif@TousifVM:~/scenv$
```

Only LOGNAME and LOGNAME3 get into child process. Because LOGNAME is already a environment variable and LOGNAME3 is exported as User-defined shell variable to the child process. LOGNAME2 is not shown because it didn't got exported to child process.

```
tousif@TousifVM:~/scenv$ strings /proc/$$/environ | grep LOGNAME
LOGNAME=tousif
tousif@TousifVM:~/scenv$ echo $LOGNAME
tousif
tousif@TousifVM:~/scenv$ LOGNAME=bob
tousif@TousifVM:~/scenv$ echo $LOGNAME
bob
tousif@TousifVM:~/scenv$ strings /proc/$$/environ | grep LOGNAME
LOGNAME=tousif
tousif@TousifVM:~/scenv$ unset LOGNAME
tousif@TousifVM:~/scenv$ echo $LOGNAME

tousif@TousifVM:~/scenv$ strings /proc/$$/environ | grep LOGNAME
LOGNAME=tousif
tousif@TousifVM:~/scenv$
```

When shell program starts, it copies environment variables into its own shell variables. changes made to the shell variable will not reflect on the environment variables as shown above.

```
#include <stdio.h>
int main()
{
	printf("hello world");
	return 0;
}
```

Static Linking:

Static compiled program is 100 times larger than a dynamic program.

```
tousif@TousifVM:~/scenv$ vi hello.c
tousif@TousifVM:~/scenv$ gcc -o dynamic_hello hello.c
tousif@TousifVM:~/scenv$ gcc -static -o static_hello hello.c
tousif@TousifVM:~/scenv$ ls -l hello.c dynamic_hello static_hello
-rwxrwxr-x 1 tousif tousif  15960 Nov 16 21:59 dynamic_hello
-rw-rw-r-- 1 tousif tousif     69 Nov 16 21:58 hello.c
-rwxrwxr-x 1 tousif tousif 900272 Nov 16 21:59 static_hello
tousif@TousifVM:~/scenv$
```

Dynamic Linking :

we can use “ldd” command to see what shared libraries a program depends on.

```
tousif@TousifVM:~/scenv$ ldd static_hello
        not a dynamic executable
tousif@TousifVM:~/scenv$ ldd dynamic_hello
        linux-vdso.so.1 (0x00007ffd25585000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f5fbfab0000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f5fbfcee000)
tousif@TousifVM:~/scenv$
```

Attacks Via Dynamic Linker :

mytest.c

```
#include<stdlib.h>
int main()
{
        sleep(1);
        return 0;
}
```

Implementing our own sleep() function.

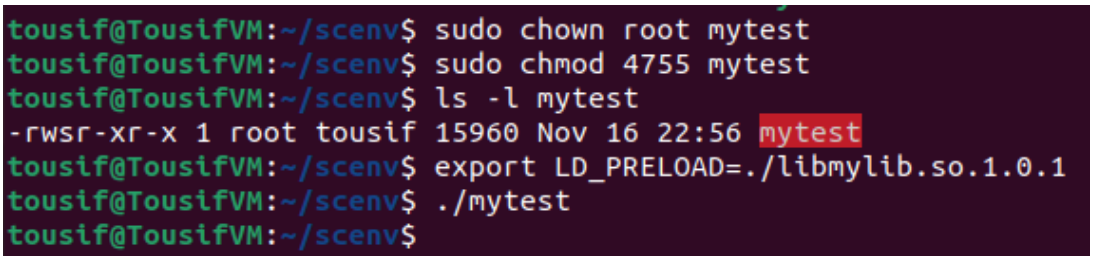
```
#include<stdio.h>
void sleep(int s)
{
        printf("Iam not sleeping\n");

}
```

Ex 1: Normal Programs

compiling above code by creating a shared library and adding the shared library to the LD_PRELOAD environment variable.

```
tousif@TousifVM:~/scenv$ vi mytest.c
tousif@TousifVM:~/scenv$ gcc mytest.c -o mytest
tousif@TousifVM:~/scenv$ ./mytest
tousif@TousifVM:~/scenv$ vi sleep.c
tousif@TousifVM:~/scenv$ gcc -c sleep.c
tousif@TousifVM:~/scenv$ gcc -shared -o libmylib.so.1.0.1 sleep.o
tousif@TousifVM:~/scenv$ ls -l libmylib.so.1.0.1 mytest mytest.c sleep.c sleep.o
-rwxrwxr-x 1 tousif tousif 15568 Nov 16 22:57 libmylib.so.1.0.1
-rwxrwxr-x 1 tousif tousif 15960 Nov 16 22:56 mytest
-rw-rw-r-- 1 tousif tousif    74 Nov 16 22:56 mytest.c
-rw-rw-r-- 1 tousif tousif    72 Nov 16 22:56 sleep.c
-rw-rw-r-- 1 tousif tousif  1504 Nov 16 22:56 sleep.o
tousif@TousifVM:~/scenv$ export LD_PRELOAD=./libmylib.so.1.0.1
tousif@TousifVM:~/scenv$ ./mytest
Iam not sleeping
tousif@TousifVM:~/scenv$ unset LD_PRELOAD
tousif@TousifVM:~/scenv$ ./mytest
tousif@TousifVM:~/scenv$
```

Ex 2: Set-UID Programs :



Here our sleep() function was not invoked.
This is due to a countermeasure implemented by the dynamic linker. It ignores the LD_PRELOAD and LD_LIBRARY_PATH environment variables when the EUID and RUID differ.

Attacks Via External Program :

vul.c :

```
#include<stdlib.h>
int main()
{
      system("cal");
}
```

cal.c :

```
#include<stdlib.h>
int main()
{
      system("/bin/dash");
}
```

Compared to system(), execve()'s attack surface is smaller.
execve() does not invoke shell , is not affected by environment variables.

```
tousif@TousifVM:~/scenv$ gcc -o vul vul.c
tousif@TousifVM:~/scenv$ sudo chown root vul
tousif@TousifVM:~/scenv$ sudo chown 4755 vul
tousif@TousifVM:~/scenv$ rm cal
tousif@TousifVM:~/scenv$ gcc -o vul vul.c
tousif@TousifVM:~/scenv$ sudo chown root vul
tousif@TousifVM:~/scenv$ sudo chown 4755 vul
tousif@TousifVM:~/scenv$ vul
   November 2022
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30

tousif@TousifVM:~/scenv$ gcc -o cal cal.c
tousif@TousifVM:~/scenv$ export PATH=.:$PATH
tousif@TousifVM:~/scenv$ echo $PATH
.:.:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
r/local/games:/snap/bin:/snap/bin:/home/tousif/:/home/tousif/:/home/tousif/:/ho
me/tousif/
tousif@TousifVM:~/scenv$ vul
$ echo $0
/bin/dash
$ exit
tousif@TousifVM:~/scenv$
```

Thus , when invoking external programs in privileged programs, we should use execve().

Attacks Via Application Code :

prog.c :

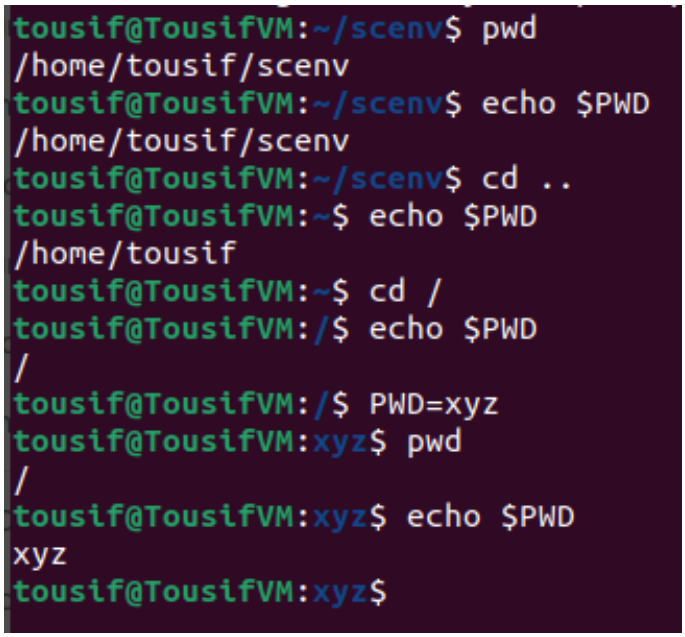
```
#include<stdio.h>
#include<stdlib.h>

int main(void)
{
        char arr[64];
        char *ptr;

        ptr=getenv("PWD");
        if(ptr!=NULL)
        {
                sprintf(arr, "Present working directory is: %s",ptr);
                printf("%s\n",arr);
```

```
    }
    return 0;
}
```

Programs may directly use env variables. If these are privileged programs, it may result in untrusted inputs.

Program uses getenv() to know its current directory from PWD env variable.

Developers may choose to use a secure version of getenv(), such as secure_getenv().



Value of PWD comes from the shell program, so every time we change our folder the shell program updates its shell variable.

We can change the shell variable ourselves as shown above.