Name : Mohammed Tousif
Roll : CB.EN.P2CYS22008

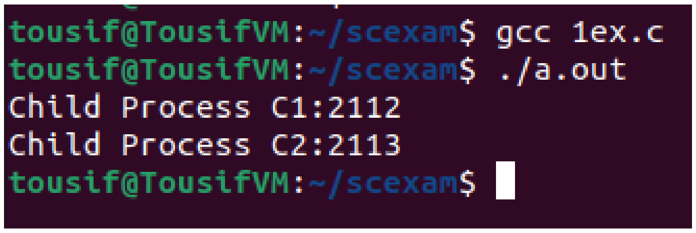
SECURE CODING LAB
FIRST ASSESSMENT

1. Write a C program P1 using appropriate system call(s) to create 2 child processes C1 and C2. Print the process IDs of all the processes

Source Code :

```
#include <syscall.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>

void main()
{
pid_t c1, c2;

c1 = fork();

if (c1 == 0) {
   printf("Child Process C1:%d\n",getpid());
} else {
   c2 = fork();

   if (c2 == 0) {
      printf("Child Process C2:%d\n",getpid());
   }
}
}
```

Output :



```
tousif@TousifVM:~/scexam$ gcc 1ex.c
tousif@TousifVM:~/scexam$ ./a.out
Child Process C1:2112
Child Process C2:2113
tousif@TousifVM:~/scexam$ 
```

2. Create another C program P2 which should obtain a filename as input via command line and display its contents using appropriate system calls. In Q1, modify the child process C1 to execute the program P1 and C2 to execute program P2

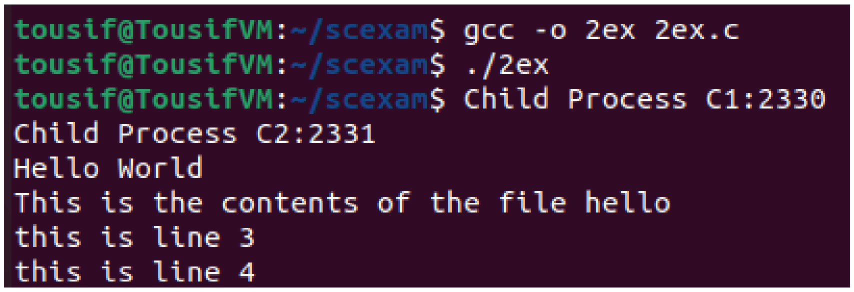
<u>Source code</u> :

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

int main(int argc, char* argv[]) {
pid_t c1, c2;
c1 = fork();
if (c1 == 0)
{
        system("./1ex");
}
else{
c2=fork();
if(c2==0)
{
   system("./file2 hello.txt");
}
}
}
```

file2 :

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

int main(int argc, char* argv[]) {

    char cmd[50]="cat ";
    strcat(cmd,argv[1]);
    system(cmd);
}
```

Output :



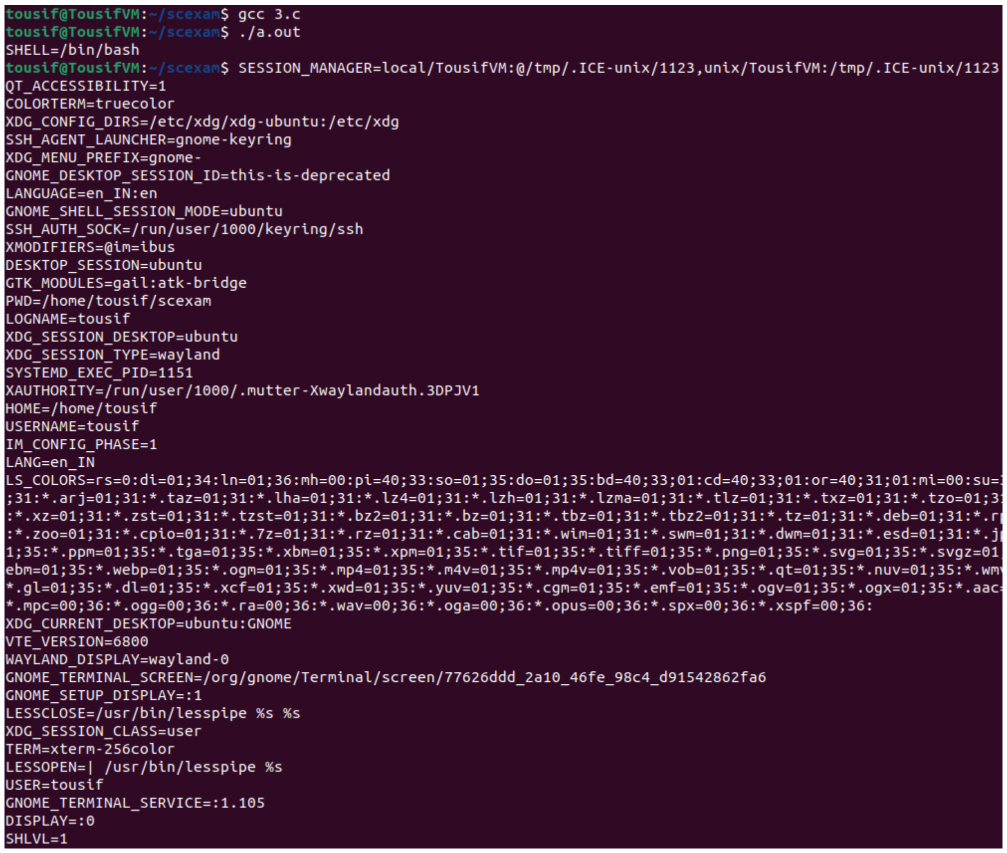
```
tousif@TousifVM:~/scexam$ gcc -o 2ex 2ex.c
tousif@TousifVM:~/scexam$ ./2ex
tousif@TousifVM:~/scexam$ Child Process C1:2330
Child Process C2:2331
Hello World
This is the contents of the file hello
this is line 3
this is line 4
```

3. Make the Child process C1 to print the environment variables obtained. Invoke C2 with 2 new set of environmental variables.

Source Code :

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

extern char** environ;
void main(int argc,char* argv[],char* envp[])
{
        pid_t c1,c2;
        int i=0;
        char* v[2];
        char* newenv[2];
        c1=fork();
        if(c1==0)
        {

        while(environ[i]!=NULL)
        {
                printf("%s\n",environ[i++]);
        }
        }
        else{
        c2=fork();
        if(c2==0)
        {
        v[0]="/usr/bin/env";
        v[1]=NULL;
        newenv[0]="AAA=aaa";
        newenv[1]="BBB=bbb";
        execve(v[0],v,newenv);
        }
        }
}
```

Output :



```
tousif@TousifVM:~/scexam$ gcc 3.c
tousif@TousifVM:~/scexam$ ./a.out
SHELL=/bin/bash
tousif@TousifVM:~/scexam$ SESSION_MANAGER=local/TousifVM:@/tmp/.ICE-unix/1123,unix/TousifVM:/tmp/.ICE-unix/1123
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/scexam
LOGNAME=tousif
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=1151
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.3DPJV1
HOME=/home/tousif
USERNAME=tousif
IM_CONFIG_PHASE=1
LANG=en_IN
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=
;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;3
:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.r
:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.j
1;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01
ebm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wm
*.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
*.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=6800
WAYLAND_DISPLAY=wayland-0
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/77626ddd_2a10_46fe_98c4_d91542862fa6
GNOME_SETUP_DISPLAY=:1
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=tousif
GNOME_TERMINAL_SERVICE=:1.105
DISPLAY=:0
SHLVL=1
```

4. Analyze the code for any vulnerabilities and demonstrate / explain them.

At first, we are creating two child processes using fork(). In second program , we obtain a filename through command line argument and displaying its contents using system() system call. At third program, we are printing the environment variables obtained by the first child process and two new environment variables are set by the second child process.

If a file has been opened with parent process with setuid. when the parent process has ended, still the child process is present with all inherited privileges and env variables causing a vulnerability.

Shell programs behaviour is affected by many environment variables. the most common is PATH variable. when shell program runs a command without its absolute path provided, it uses PATH env variable. system() can manipulate the program. Compared to system() , execve() attack surface is smaller as it does not invoke shell which we used in third program to obtain the new set of environment variables.