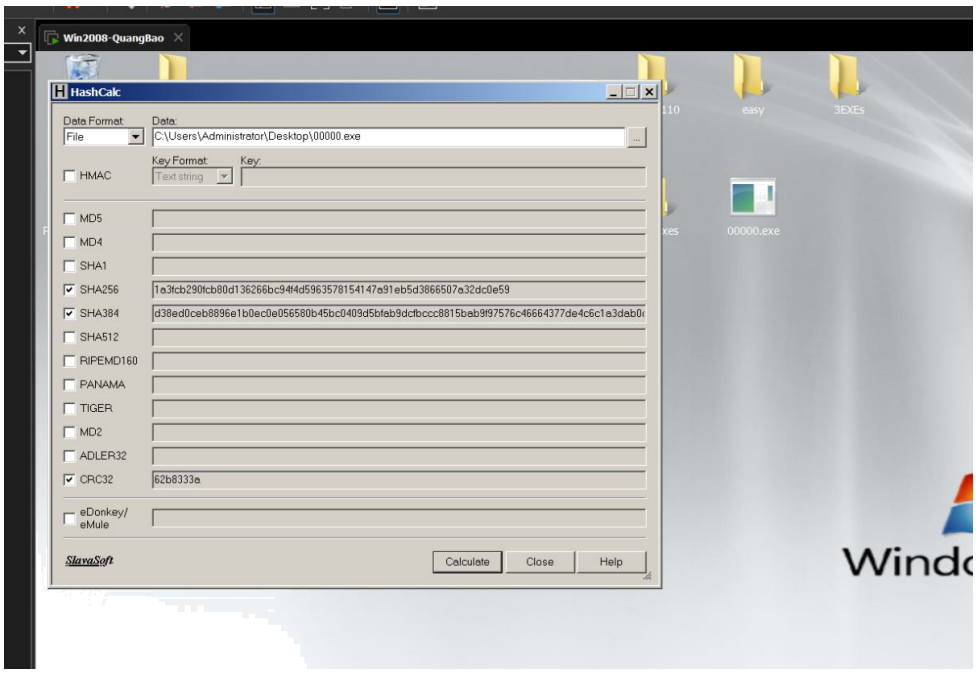
Lab 18.2: Patching EXEs with Ollydbg

18.2.1: Patching an EXE

- **What you need:**
  - A Windows machine, real or virtual. I used a Windows Server 2008 virtual machine.
  - You need several files to examine. They are all in the Documents folder of the VM your instructor handed out. If you don't have
  - that, download them with these links:
    00000.exe
    3EXEs.zip
    easy.zip
    256exes.zi
- **Purpose:**
  - To practice disassembling and modifying binarie.
- **Getting the EXE:**
  - In the Documents folder of the VM handed out by your instructor, find the 00000.exe file
- **Checking the Hash:**
  - Click Start. Type HASH and click HashCalc. In HashCalc, make sure the

  - Click Start, Documents. Drag the 00000.exe file from the Documents folder and drop it onto the HashCalc box.
  - HashCalc calculates the SHA256 hash of the file. It should match the value shown below.

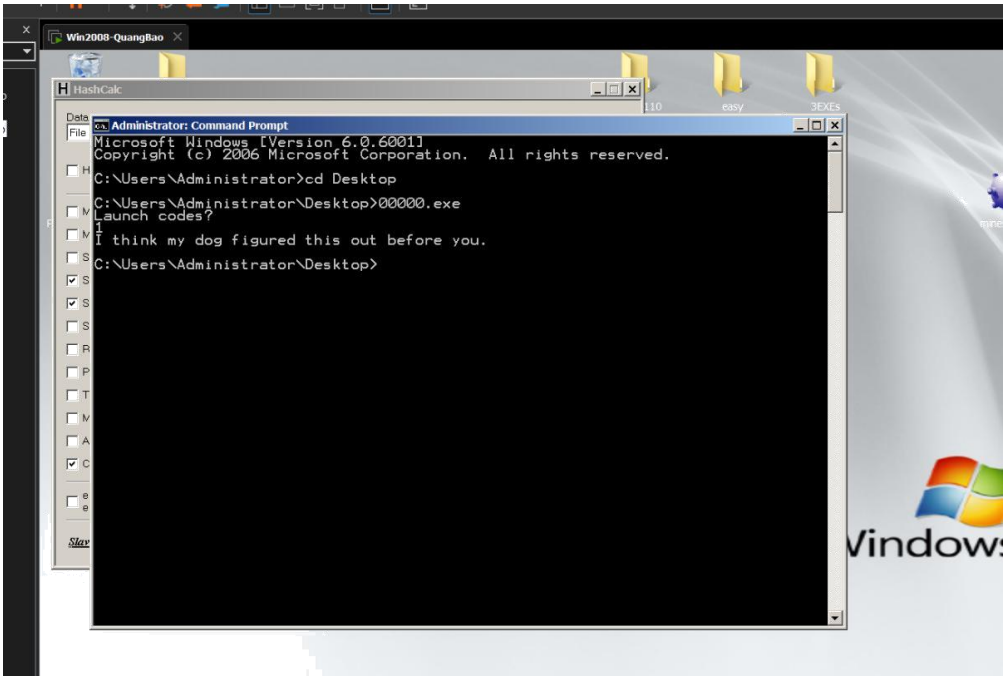


- **Running the EXE:**
  - Click the black square icon at the lower left of your desktop to open a Command Prompt.

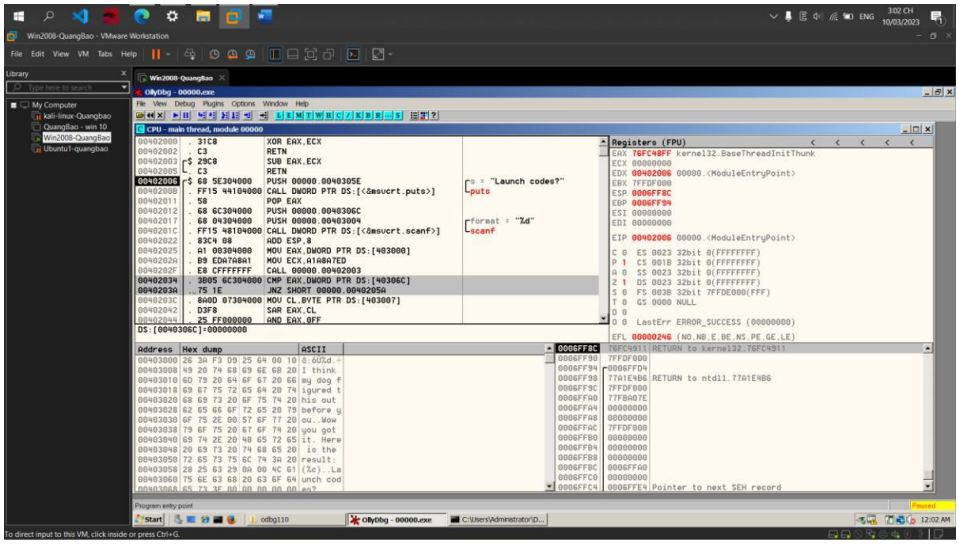
    Execute these commands:

    cd \users\administrator\Desktop

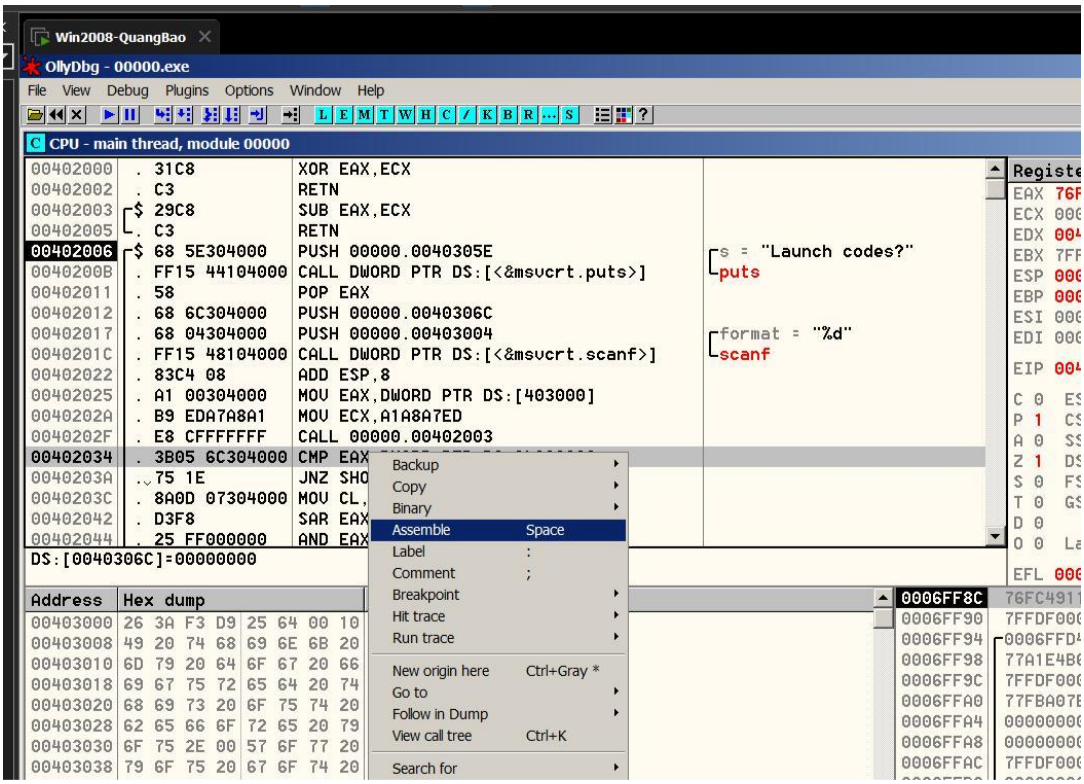
    00000.exe
  - It asks for a "Launch code". Enter 1. Your code is wrong, and it insults you, as shown below



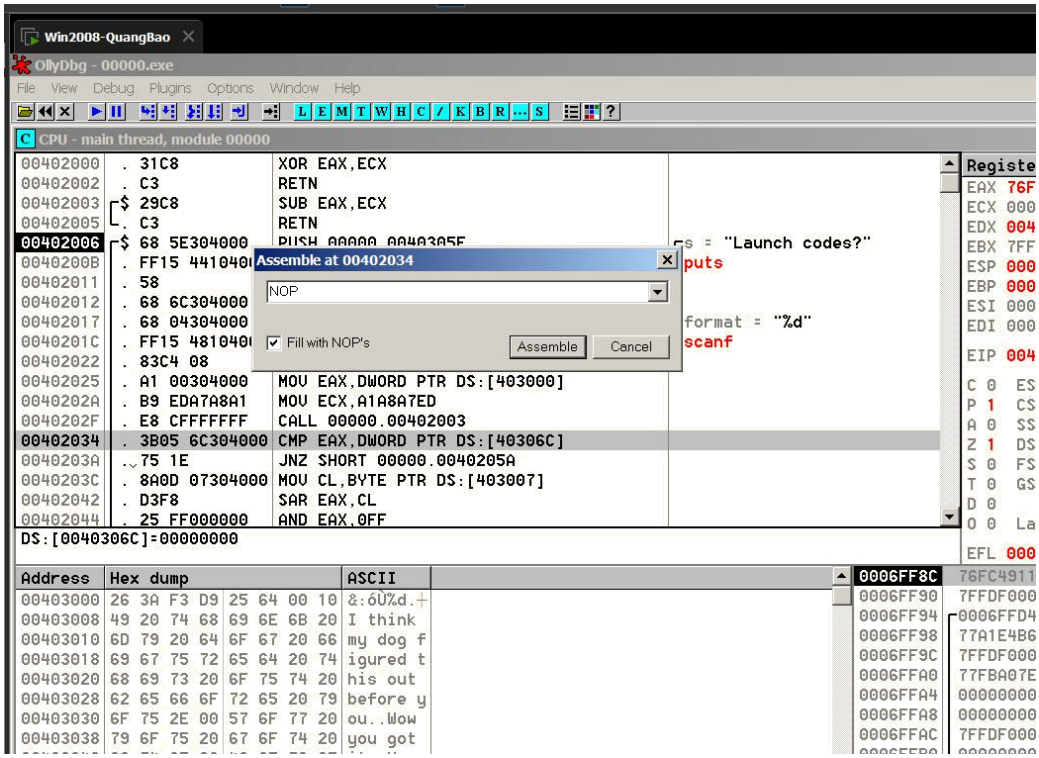
- **Examining the EXE with Ollydbg:**
  - Open the file in OllyDbg, as shown below.
  - Look at the rightmost section, and you can easily see what the program does; it prints out "Launch codes?", reads in a decimal
  - number (%d), and then chooses to print either a winning message with a result, or an insult.
  - The choice is performed by two instructions: CMP (Compare) and JNZ (Jump if Not Zero), outlined in green in the image below



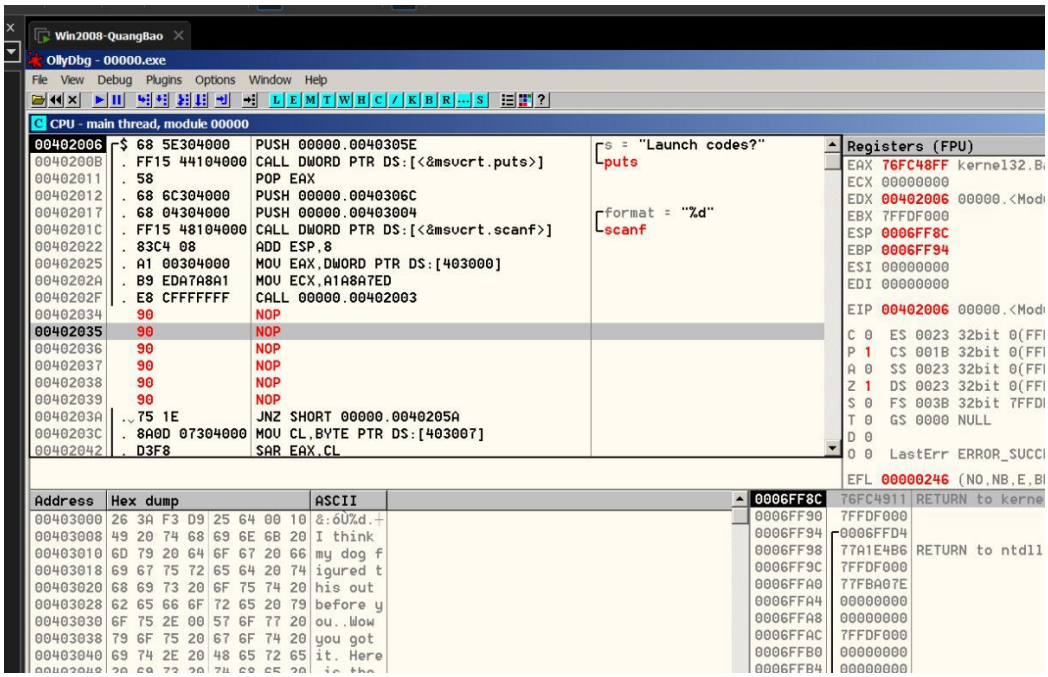
- **Modifying the EXE:**
  - Right-click the CMP instruction and click Assemble, as shown below.



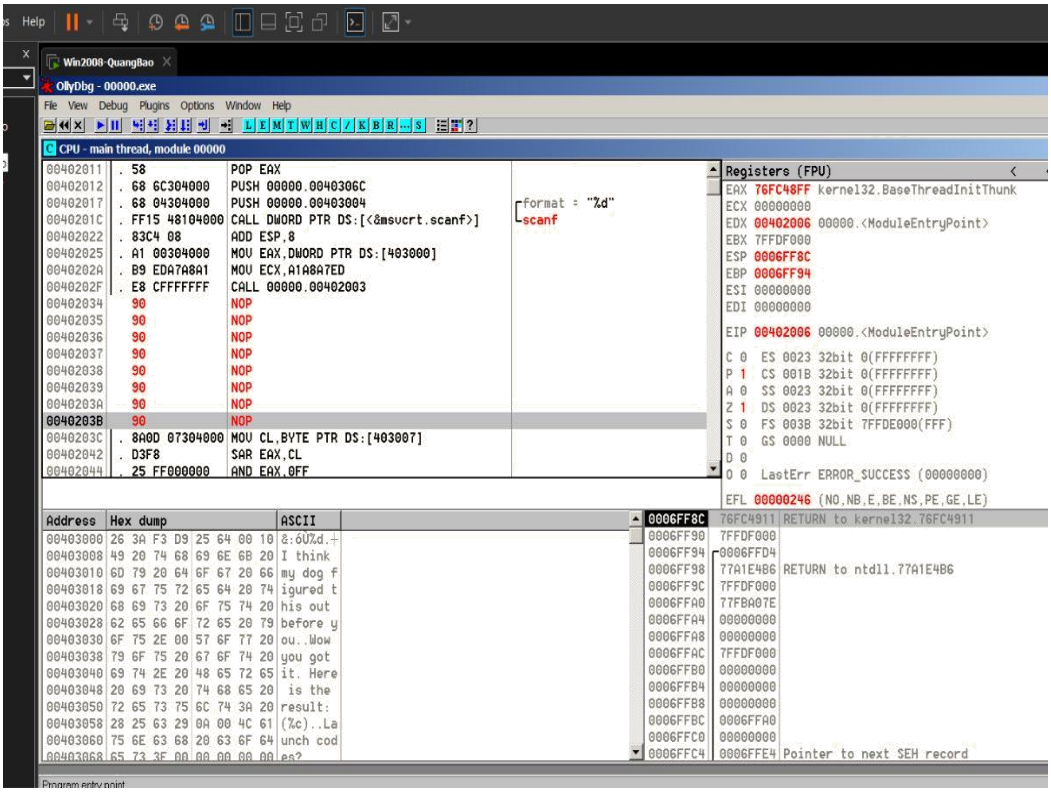
  - In the Assemble box, enter NOP, as shown below.



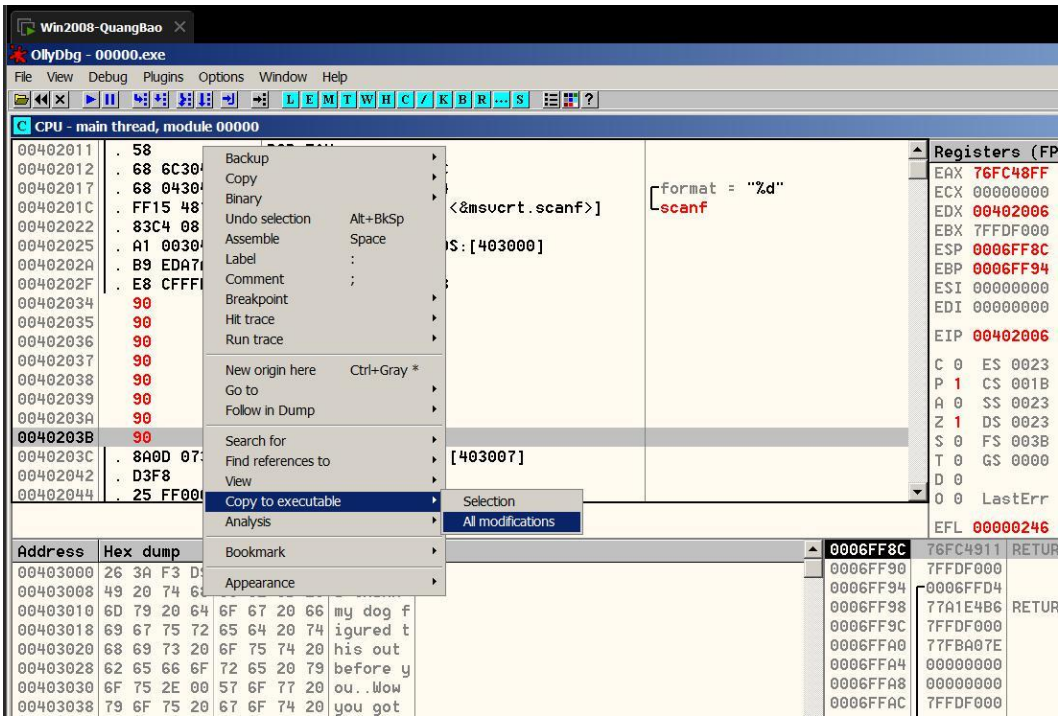
- Click the **Assemble** button. Click the **Cancel** button.
- The CMP instruction is replaced by a series of NOPs, as shown below.



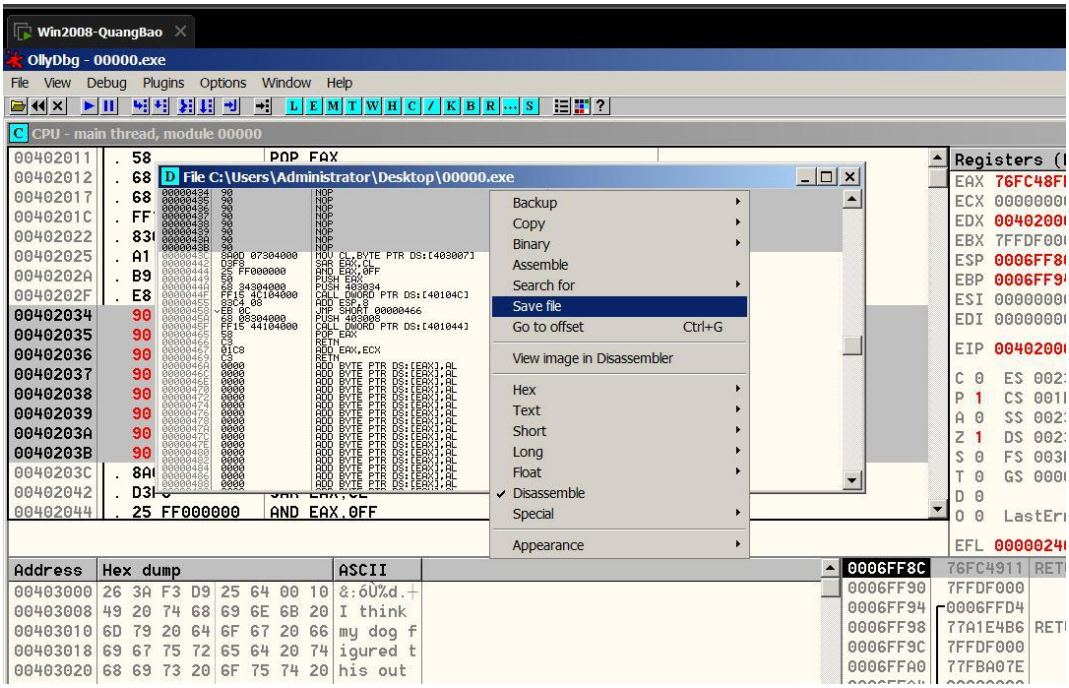
- Repeat the process to replace the **JNZ** instruction with NOPs also, as shown below



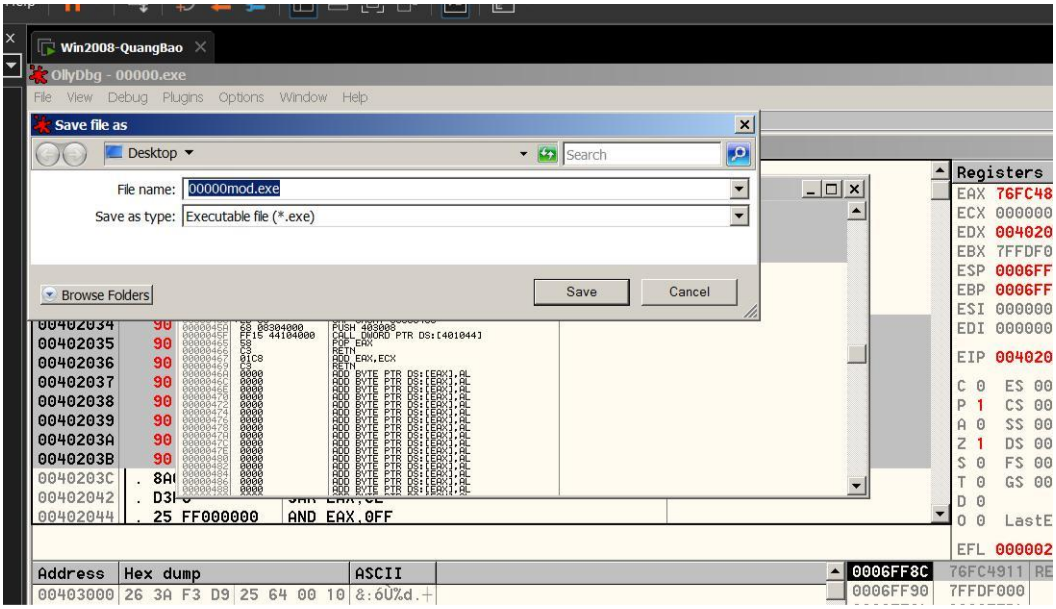
- **Saving the Modified File:**
  - In OllyDbg, in the top left pane, right-click and click **"Copy to executable", "All modifications"**, as shown below



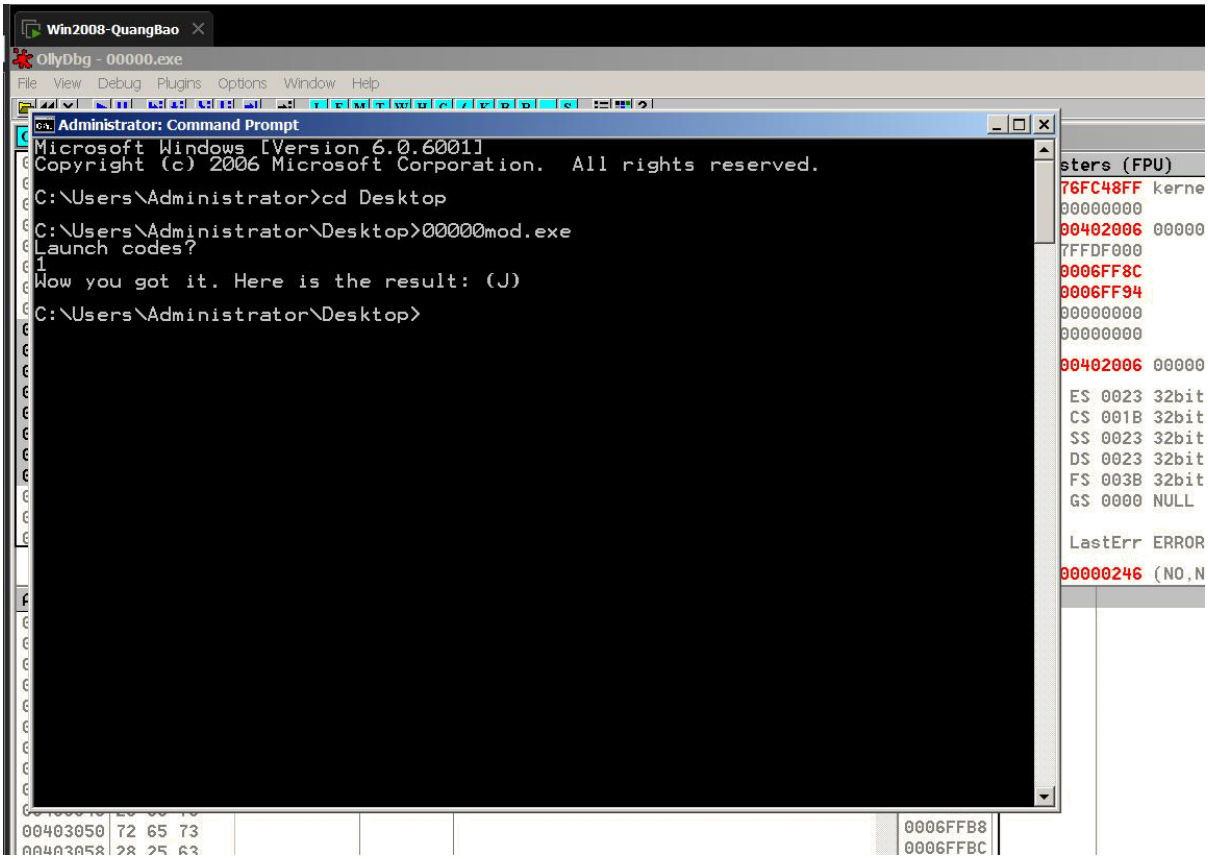
  - A "Copy selection to executable file?" box pops up. Click the "**Copy all**" button.
  - A "**File**" box appears, as shown below.
  - Right-click in it and click "Save file".



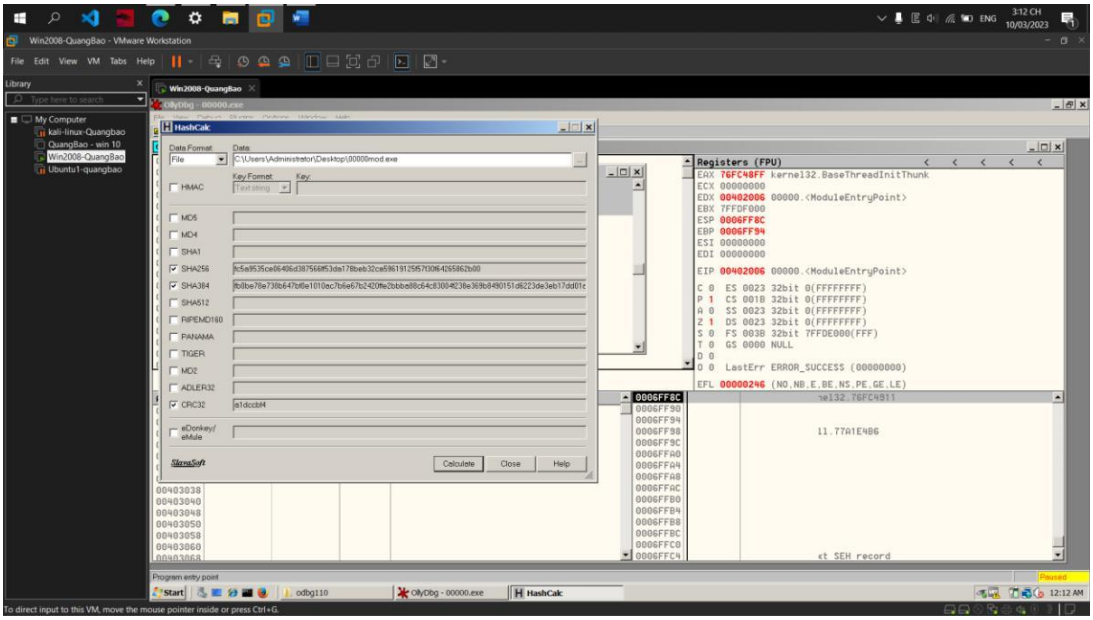
- A "Save file as" box appears. Change the filename to **00000mod.exe**, as shown below, and click **Save**



- **Running the Modified File**
  - In a Command Prompt window, execute these commands:
    cd \users\administrator\Desktop
    00000mod.exe
  - It asks for a "Launch code". Enter 1. It accepts the code now, as shown below.

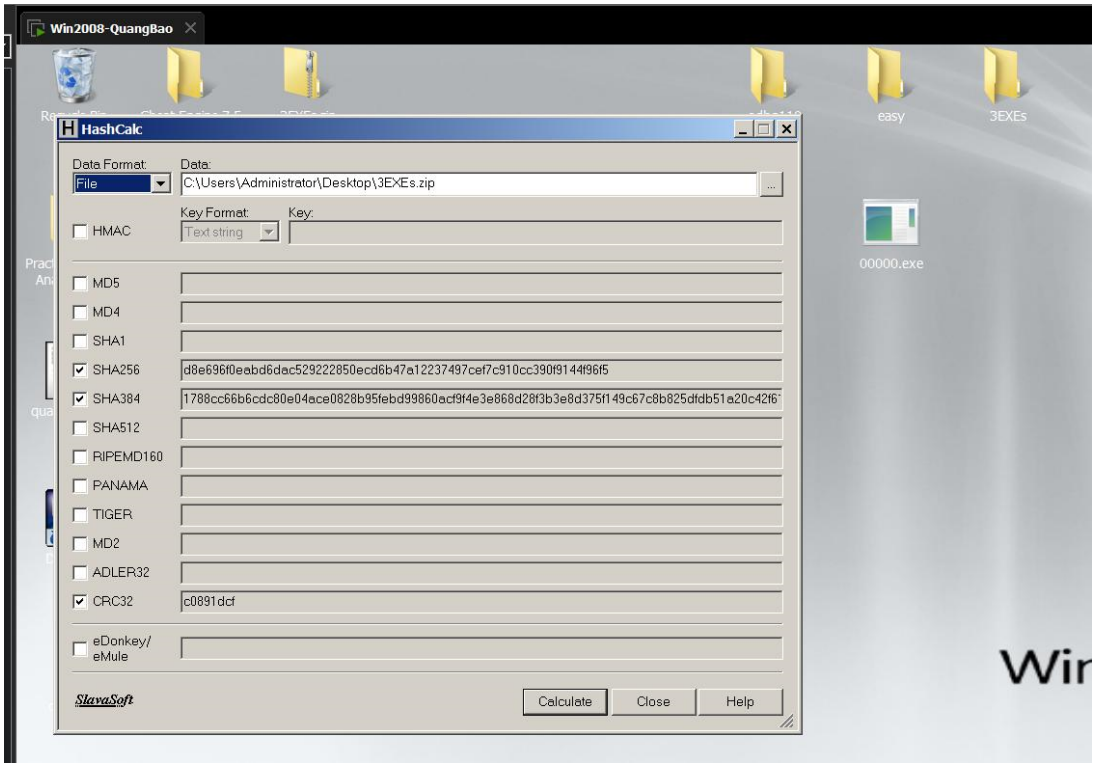


- **Checking the Hash**
  - Calculate the SHA256 hash of the patched file. It should match the value shown below.
  - Find the CRC32 hash, which is covered in a green box in the image below. Enter it into the form below



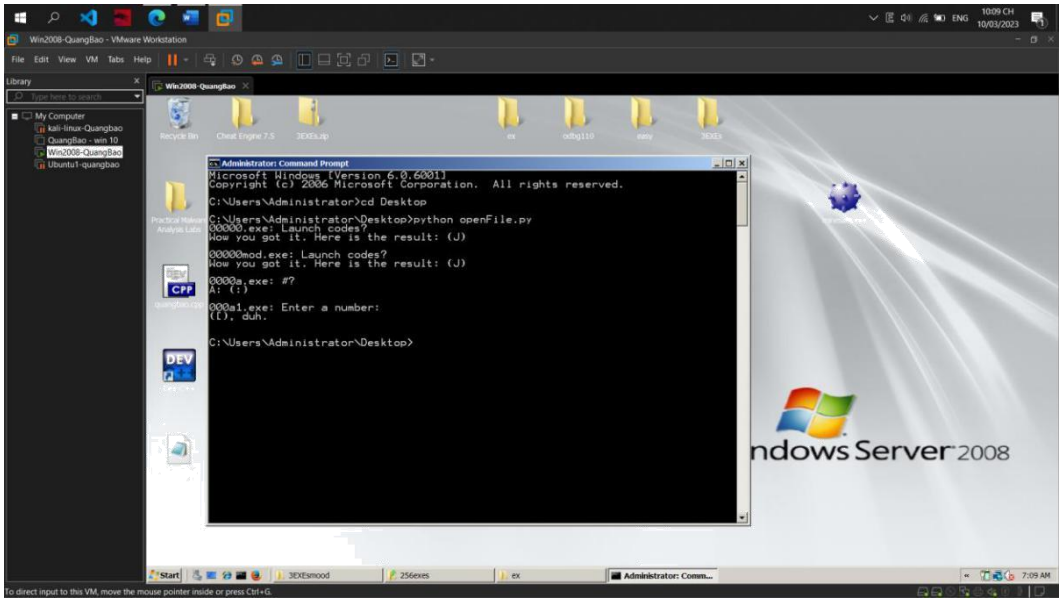
18.2.2: Patching three EXEs:

- **Getting the EXEs:**
  - In the Documents folder of the VM handed out by your instructor, find the **3EXEs.zip** file.
- **Checking the Hash:**
  - Calculate the SHA256 hash of the file. It should match the value shown below



- **Patch the Files:**
  - Patch all 3 files so they will accept any input
- **Gather the Results:**
  - Run the three patched files. Each one returns a single character as a result. Keep the files in alphabetical order, by filename, like this:
    File **00000.exe** Result **C**
    File **0000a.exe** Result **A**

  - If those were the results, the answer would be **CAT** ○ The actual results are different, of course.

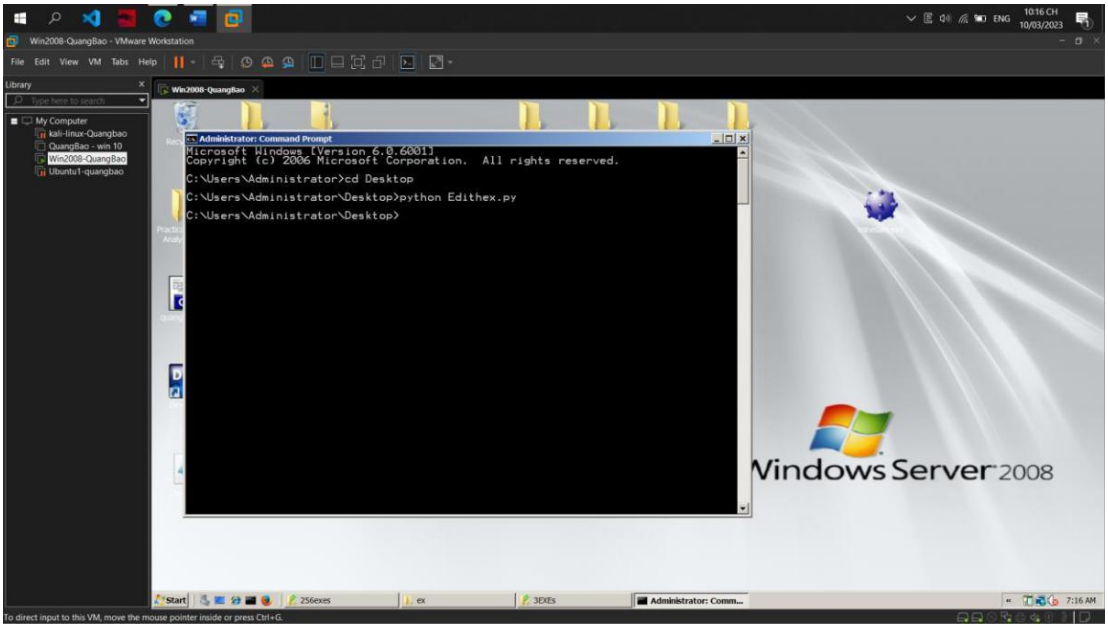


- Use python script to path all file in diretory:

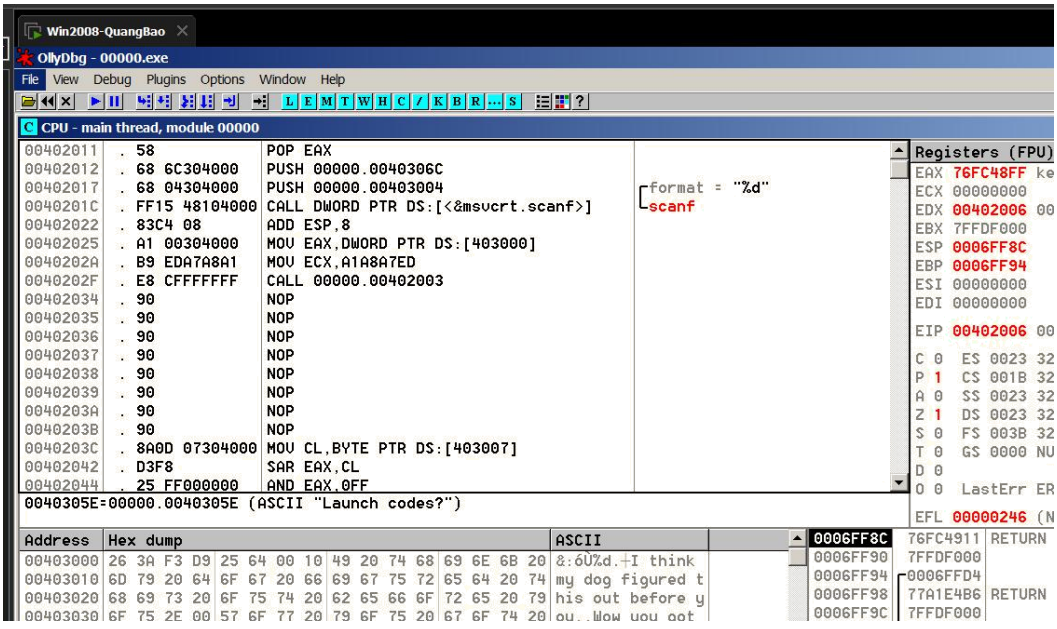
```
import binascii
import os

search_start = b'\x3B\x05'
search_end = b'\x75\x1E'
replace_value = b'\x90\x90\x90\x90\x90\x90\x90\x90'

# change directory to the path where your files are located
os.chdir('C:/Users/Administrator/Desktop/3EXEsmood')

# loop through all files in the directory
for filename in os.listdir('.'):
    if filename.endswith('.exe') or filename.endswith('.dll'):
        # open the file in binary mode
        with open(filename, 'rb') as f:
            # read the file contents into a bytes
            object file_contents = f.read()

        # loop through the file contents searching for the hex
        pattern i = 0
        while i < len(file_contents):
            if file_contents[i:i+2] == search_start:
                j = i+2
                while j < len(file_contents):
                    if file_contents[j:j+2] == search_end:
```

```
                    # replace the hex pattern with the replacement value
                    file_contents = file_contents[:i] + replace_value +
file_contents[j+2:]
                    break
                j += 1
            i += 1

        # write the modified file contents back to the
        file with open(filename, 'wb') as f:
            f.write(file_contents)
```

- Now we run python script in cmd:



- Check file with Ollydbg



```
```

- Now we use another script to run mutilple file and store it in results.txt **import subprocess**
  import os

  # change directory to the path where your executable files are located os.chdir('C:/Users/Administrator/Desktop/3EXEsmood')

  # create a list to hold the results from all executable files
  results = []

  # loop through all files in the
  directory for filename in os.listdir('.'):
  if filename.endswith('.exe'):
  # run the executable file with input "1" using subprocess proc = subprocess.Popen([filename], stdin=subprocess.PIPE, stdout=subprocess.PIPE)

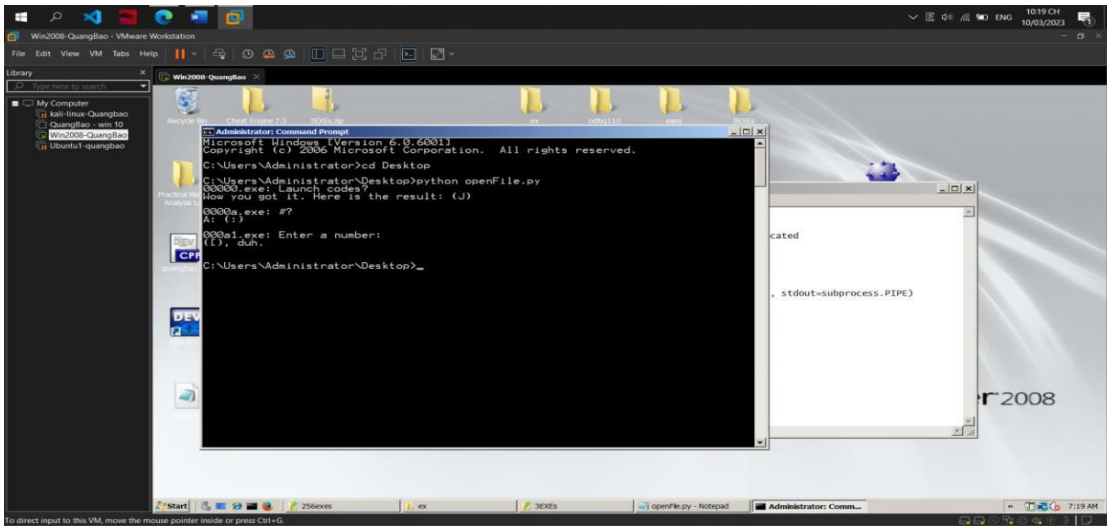
  # send input to the subprocess
  proc.stdin.write(b'18\n')
  proc.stdin.close()

  # wait for the subprocess to finish and get its
  output result = proc.stdout.read()

  # extract the string inside the parentheses and add it to the results result_str = result.decode('utf-8')
  start_index = result_str.find('(')
  end_index = result_str.find(')')
  if start_index != -1 and end_index != -1:
  results.append(result_str[start_index+1:end_index])

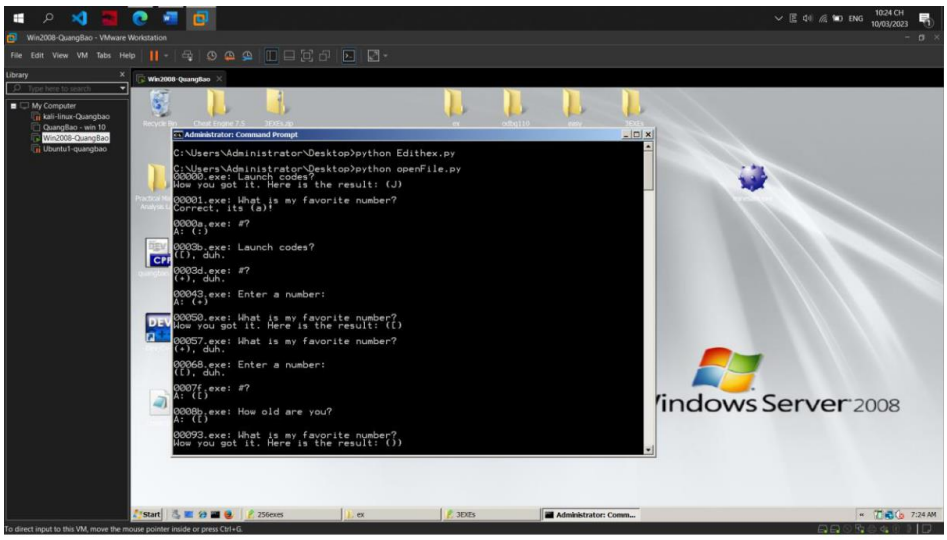
  # write the concatenated results to a text
  file with open('results.txt', 'w') as f:
  f.write(''.join(results))

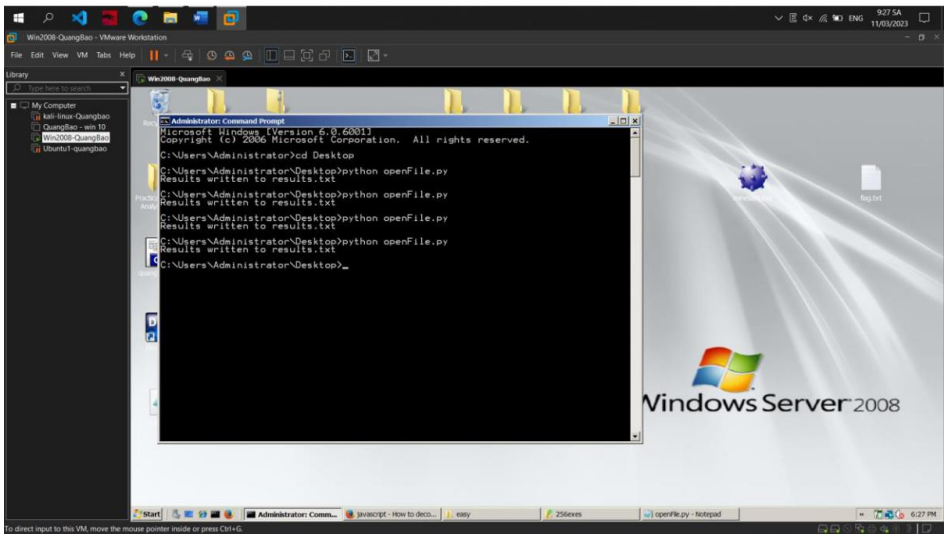
  print('Results written to results.txt')

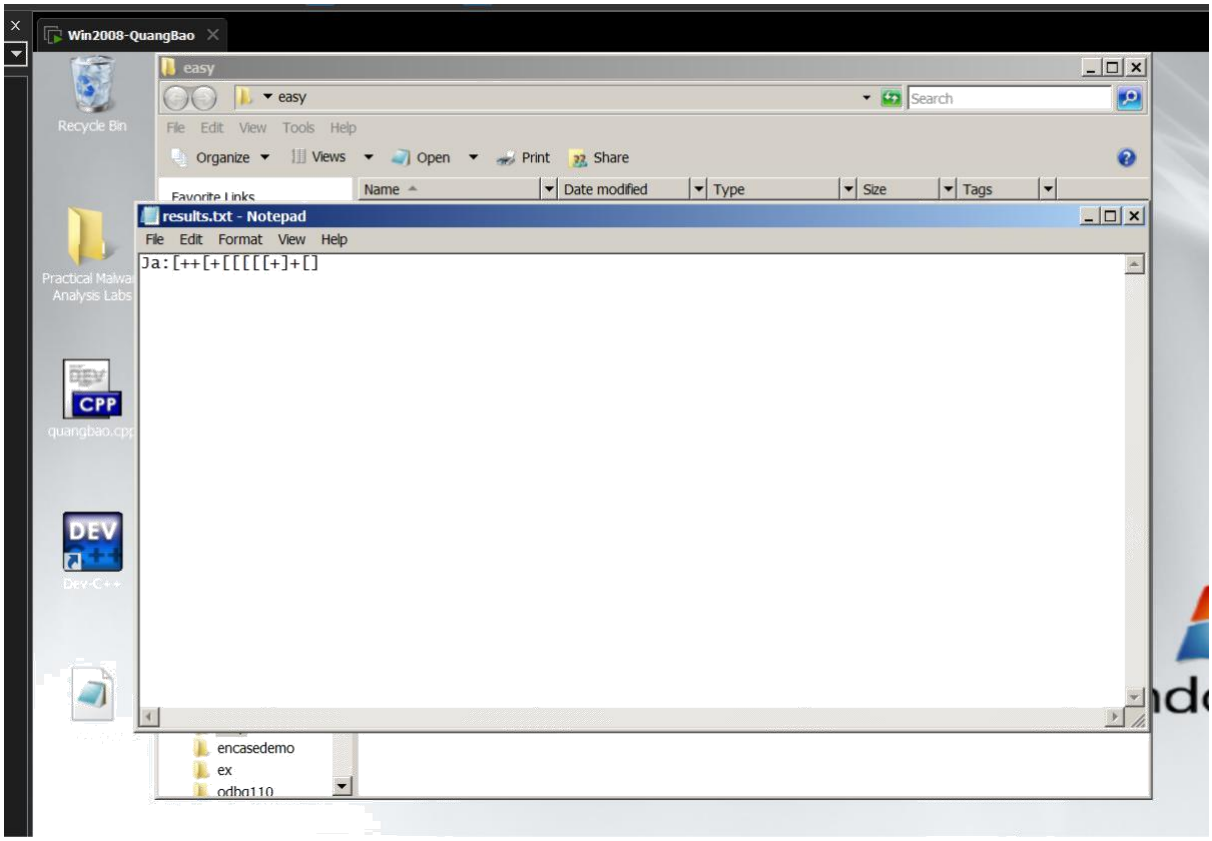


18.2.3: Patching 19 EXEs:

- **Getting the EXEs**
  - In the Documents folder of the VM handed out by your instructor, find the easy.zip file. Unzip it. There are 19 EXEs in it**.**
  - Patch all 19 files, run them, and combine the Results to get a 19-character flag

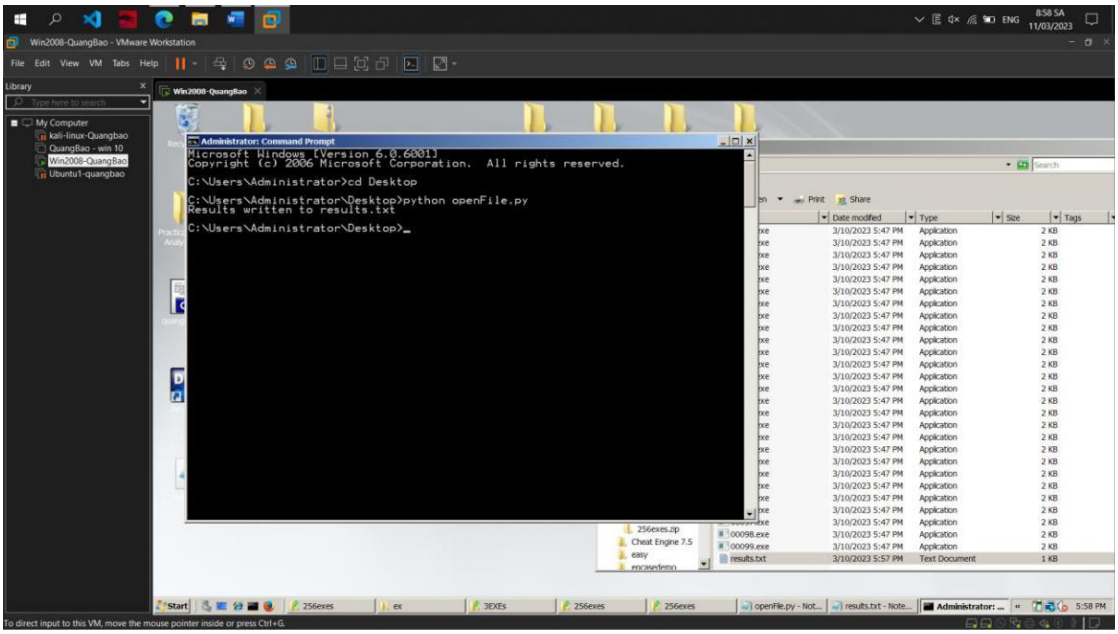




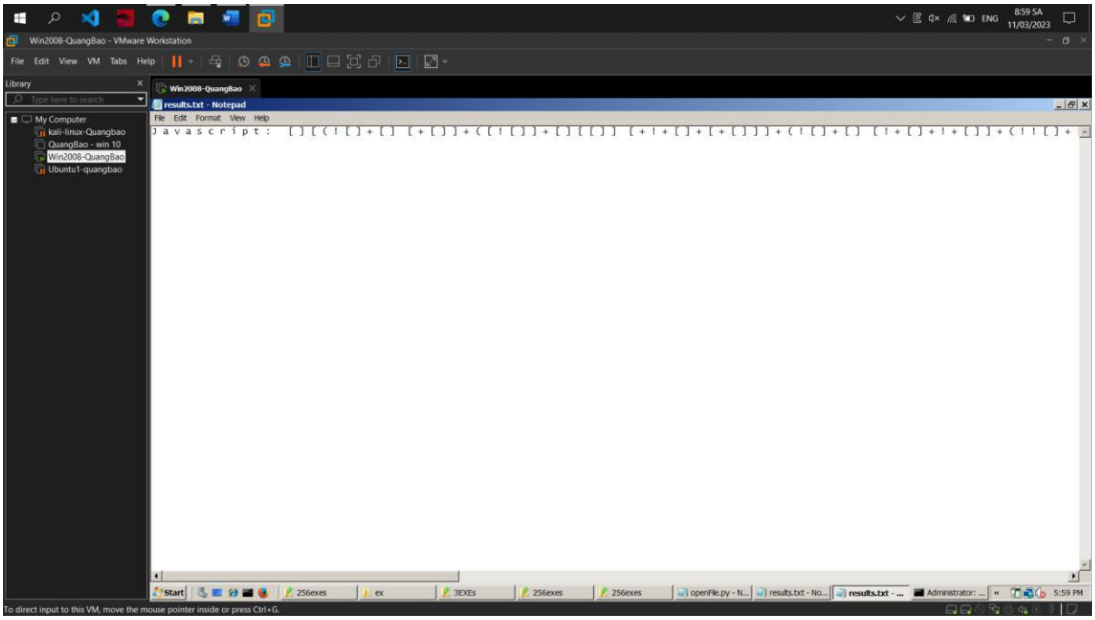


18.2.4: Patching 256 EXEs:

- **Getting the EXEs**
  - In the Documents folder of the VM handed out by your instructor, find the 256exes.zip file. Unzip it. There are 256 EXEs in it.
  - Patch all 256 files, run them, and combine the Results to get a 256-character flag



- Now we got the string as below



- This is Jsfuck programming Language. Results as below

Javascript:[][(![]+[][+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[][!+[]+!+[]]+(!![]+[][+[]]+(!![]+[][!+[]+!+[]+!+[]]+(!![]+[][+!+[]]][([][(![]+[][+[]]+([![]]+[][[])[+!+[]+[+[]]]+(![]+[][!+[]+!+[]]+(!![]+[][+[]]+(!![]+[][!+[]+!+[]+!+[]]+(!![]+[][+[]]+(!![]+[][!+[]+!+[]+!+[]]+(!![]+[][+!+

- Calculate the SHA256 hash of that file.