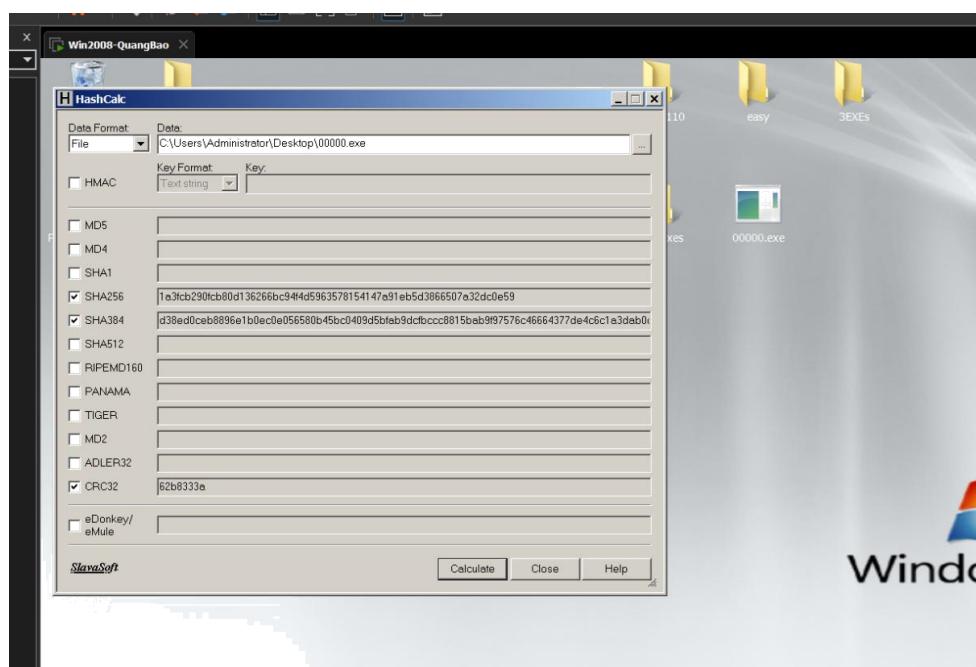


Lab 18.2: Patching EXEs with Ollydbg

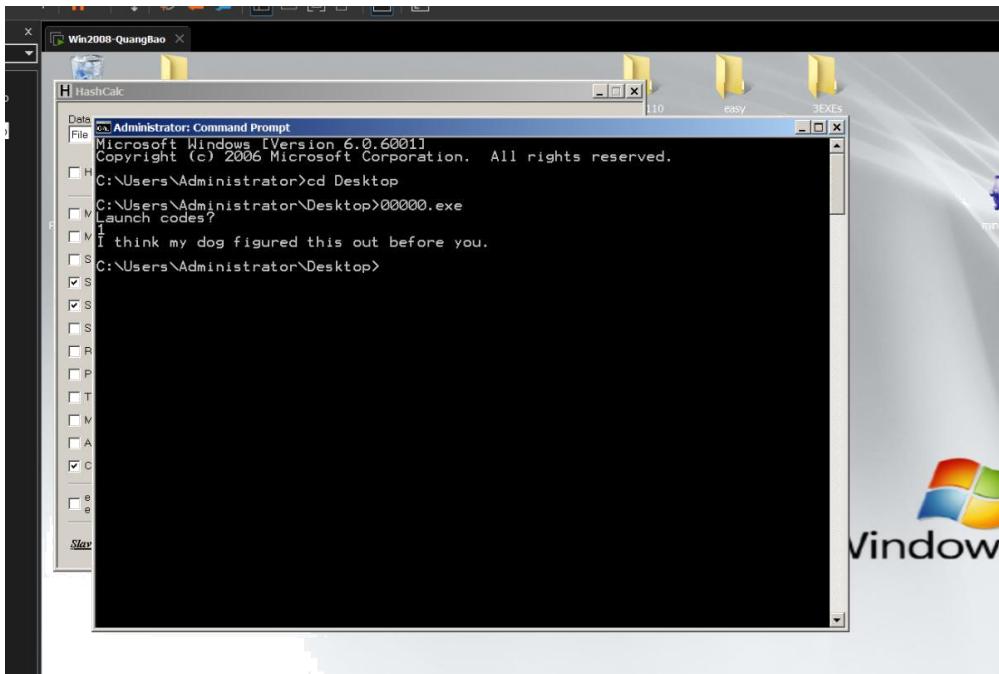
18.2.1: Patching an EXE

- **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.zip](#)
- **Purpose:**
 - To practice disassembling and modifying binary.
- **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.

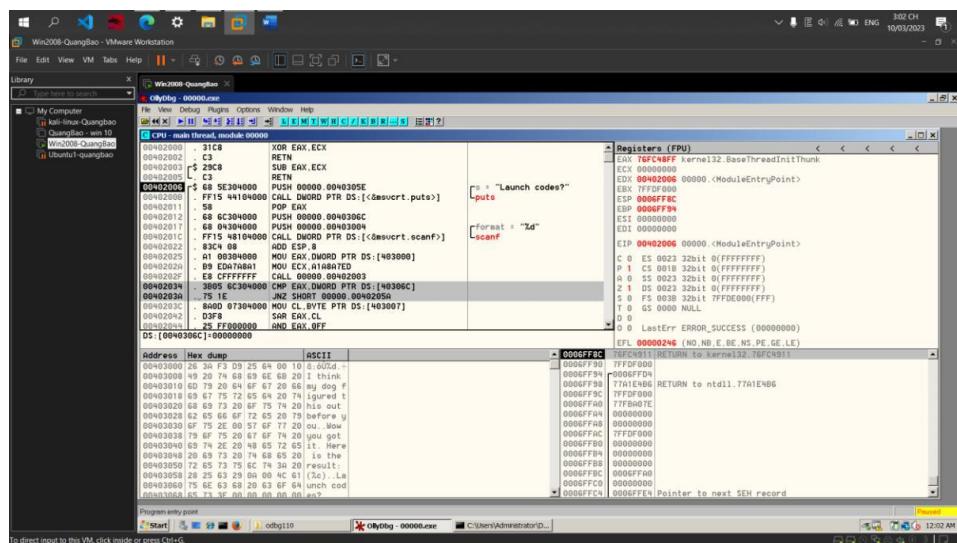


- **Running the EXE:**
 - Click the black square icon at the lower left of your desktop to open a Command Prompt.
 - Execute these commands:

```
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

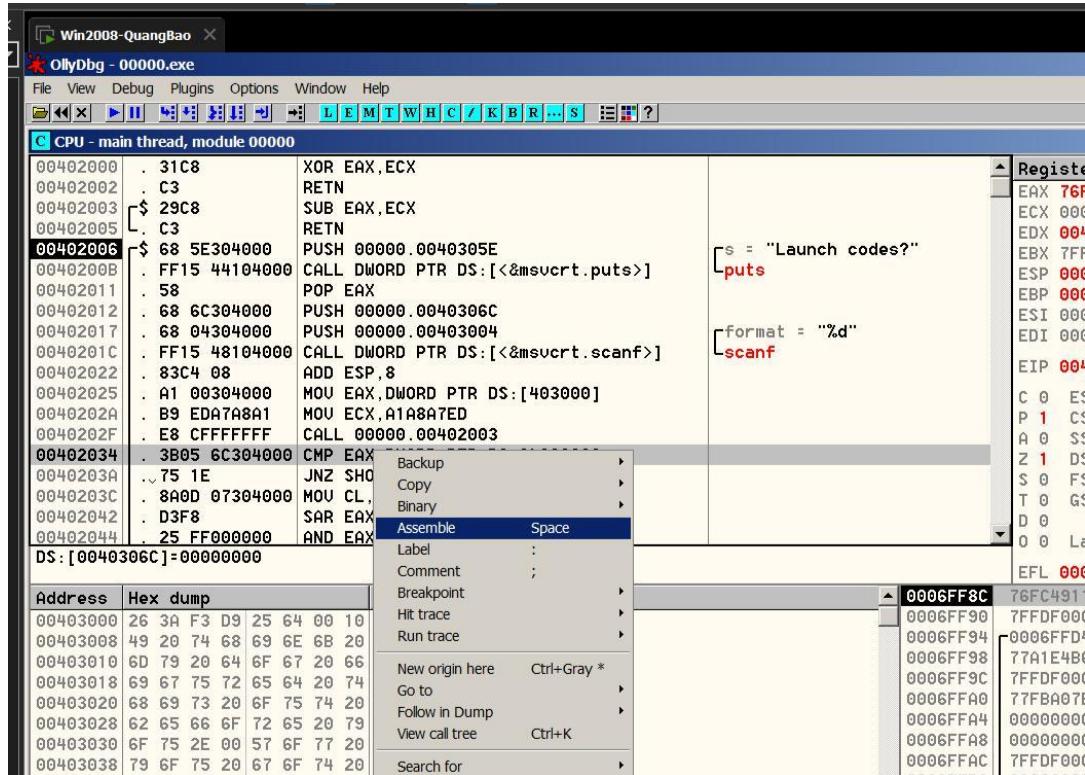


- **Examining the EXE with OllyDbg:**
 - Open the file in OllyDbg, as shown below.
 - Look at the rightmost section, and you can easily see what the program does; it prints out "Launch codes?", reads in a decimal number (%d), and then chooses to print either a winning message with a result, or an insult.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

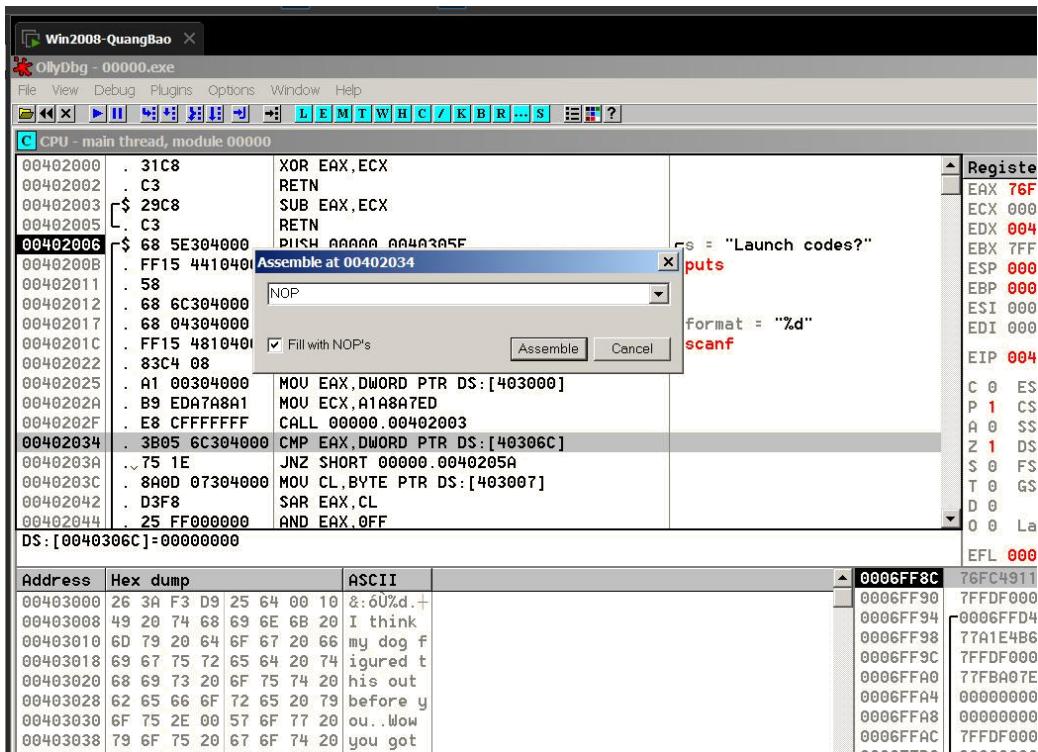


- **Modifying the EXE:**

- 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.

Assembly window (CPU - main thread, module 00000):

```

00402006  . $ 68 5E304000 PUSH 00000.0040305E
00402008  . FF15 44104000 CALL DWORD PTR DS:[<&msvcrt.puts>]
00402011  . 58          POP  EAX
00402012  . 68 6C304000 PUSH 00000.0040306C
00402017  . 68 04304000 PUSH 00000.00403004
0040201C  . FF15 48104000 CALL DWORD PTR DS:[<&msvcrt.scprintf>]
00402022  . 83C4 08      ADD   ESP,8
00402025  . A1 00304000 MOU  EAX,DWORD PTR DS:[403000]
0040202A  . B9 EDA7A8A1 MOU  ECX,A1A8A7ED
0040202F  . E8 CFFFFFFF CALL  00000.00402003
00402034  . 90          NOP
00402035  . 90          NOP
00402036  . 90          NOP
00402037  . 90          NOP
00402038  . 90          NOP
00402039  . 90          NOP
0040203A  . 75 1E       JNZ SHORT 00000.0040205A
0040203C  . 8A0D 07304000 MOU  CL,BYTE PTR DS:[403007]
00402042  . D3F8         SAR   EAX,CL

```

Registers (FPU) window:

EAX	76FC48FF	kernel32.Bi...
ECX	00000000	
EDX	00402006	00000.<Modi...
EBX	7FFDF000	
ESP	0006FF8C	
EBP	0006FF94	
ESI	00000000	
EDI	00000000	
EIP	00402006	00000.<Modi...
C 0	ES	0023 32bit 0(FF)
P 1	CS	001B 32bit 0(FF)
A 0	SS	0023 32bit 0(FF)
Z 1	DS	0023 32bit 0(FF)
S 0	FS	003B 32bit 7FFD
T 0	GS	0000 NULL
D 0		
O 0	LastErr	ERROR_SUCC
EFL	00000246	(NO,NB,E,BI...

Dump window (Address, Hex dump, ASCII):

Address	Hex dump	ASCII	0006FF8C
00403900	26 3A F3 D9 25 64 00 10	&:0%zd.+	0006FF90
00403908	49 20 74 68 69 6E 6B 20	I think	0006FF94
00403910	6D 79 20 64 6F 67 20 66	my dog f	0006FF98
00403918	69 67 75 72 65 64 20 74	igured t	0006FF9C
00403920	68 69 73 20 6F 75 74 20	his out	0006FFA0
00403928	62 65 66 6F 72 65 20 79	before y	0006FFA4
00403930	6F 75 2E 00 57 6F 77 20	ou..Wow	0006FFA8
00403938	79 6F 75 20 67 6F 74 20	you got	0006FFAC
00403940	69 74 2E 20 48 65 72 65	it. Here	0006FFB0
00403948	69 74 2E 20 48 65 72 65	it. Here	0006FFB4

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

Assembly window (CPU - main thread, module 00000):

```

00402011  . 58          POP  EAX
00402012  . 68 6C304000 PUSH 00000.0040306C
00402017  . 68 04304000 PUSH 00000.00403004
0040201C  . FF15 48104000 CALL DWORD PTR DS:[<&msvcrt.scprintf>]
00402022  . 83C4 08      ADD   ESP,8
00402025  . A1 00304000 MOU  EAX,DWORD PTR DS:[403000]
0040202A  . B9 EDA7A8A1 MOU  ECX,A1A8A7ED
0040202F  . E8 CFFFFFFF CALL  00000.00402003
00402034  . 90          NOP
00402035  . 90          NOP
00402036  . 90          NOP
00402037  . 90          NOP
00402038  . 90          NOP
00402039  . 90          NOP
0040203A  . 90          NOP
0040203B  . 90          NOP
0040203C  . 8A0D 07304000 MOU  CL,BYTE PTR DS:[403007]
00402042  . D3F8         SAR   EAX,CL
00402044  . 25 FF000000 AND  EAX,0FF

```

Registers (FPU) window:

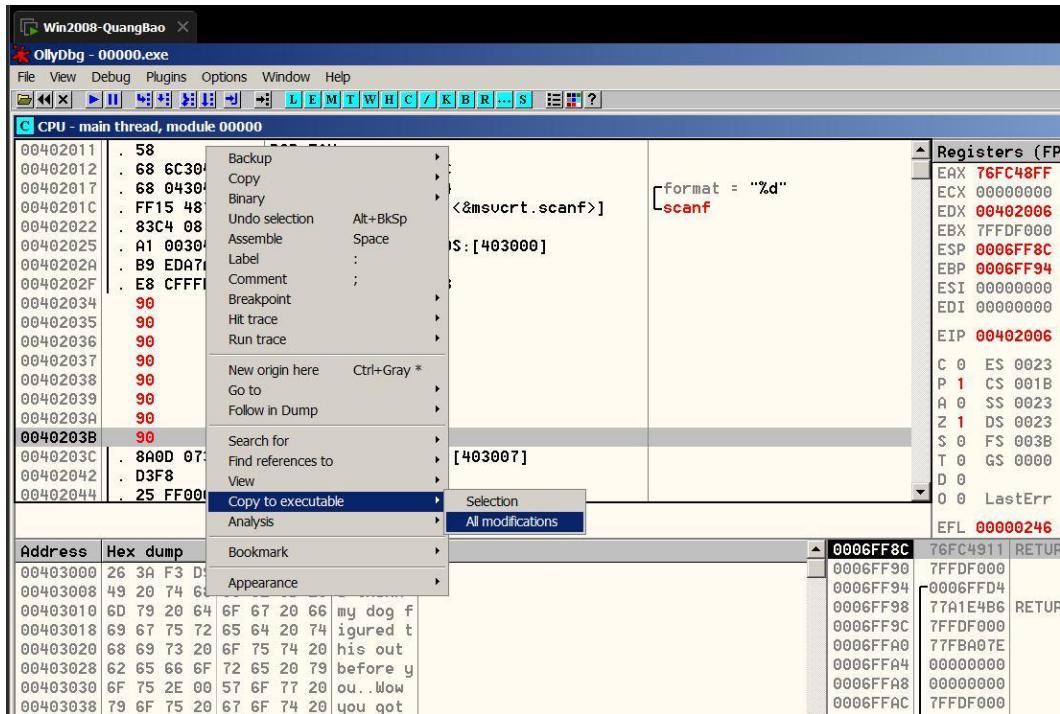
EAX	76FC48FF	kernel32.BaseThreadInitThunk
ECX	00000000	
EDX	00402006	00000.<ModuleEntryPoint>
EBX	7FFDF000	
ESP	0006FF8C	
EBP	0006FF94	
ESI	00000000	
EDI	00000000	
EIP	00402006	00000.<ModuleEntryPoint>
C 0	ES	0023 32bit 0(FFFFFFFF)
P 1	CS	001B 32bit 0(FFFFFFFF)
A 0	SS	0023 32bit 0(FFFFFFFF)
Z 1	DS	0023 32bit 0(FFFFFFFF)
S 0	FS	003B 32bit 7FFDE000(FFF)
T 0	GS	0000 NULL
D 0		
O 0	LastErr	ERROR_SUCCESS (00000000)
EFL	00000246	(NO,NB,E,BE,NS,PE,GE,LE)

Dump window (Address, Hex dump, ASCII):

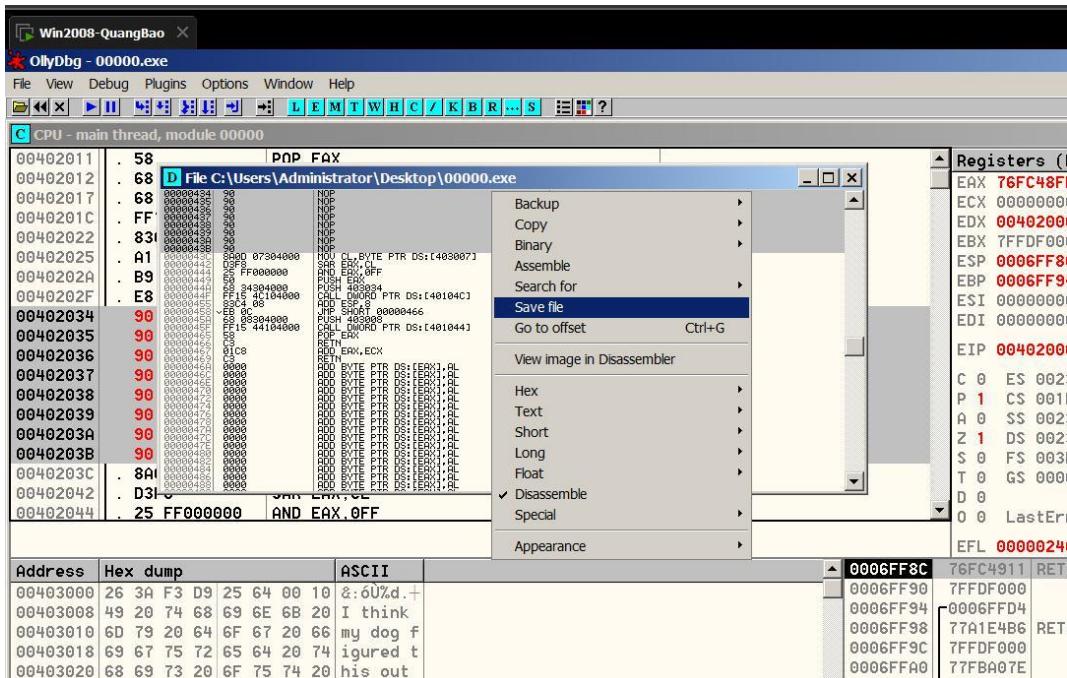
Address	Hex dump	ASCII	0006FF8C
00403900	26 3A F3 D9 25 64 00 10	&:0%zd.+	76FC4911 RETURN to kernel32.76FC4911
00403908	49 20 74 68 69 6E 6B 20	I think	0006FF90
00403910	6D 79 20 64 6F 67 20 66	my dog f	0006FF94
00403918	69 67 75 72 65 64 20 74	igured t	0006FF98
00403920	68 69 73 20 6F 75 74 20	his out	0006FF9C
00403928	62 65 66 6F 72 65 20 79	before y	0006FFA0
00403930	6F 75 2E 00 57 6F 77 20	ou..Wow	0006FFA4
00403938	79 6F 75 20 67 6F 74 20	you got	0006FFAC
00403940	69 74 2E 20 48 65 72 65	it. Here	0006FFB0
00403948	69 74 2E 20 48 65 72 65	it. Here	0006FFB4
00403950	72 65 73 75 6C 74 3A 20	result:	0006FFB8
00403958	28 25 63 29 0A 00 4C 61	(%c)..La	0006FFBC
00403960	75 6E 63 68 20 63 6F 64	nch cod	0006FFC0
00403968	65 73 3F 00 00 00 00 00	les?	0006FFC4

- **Saving the Modified File:**

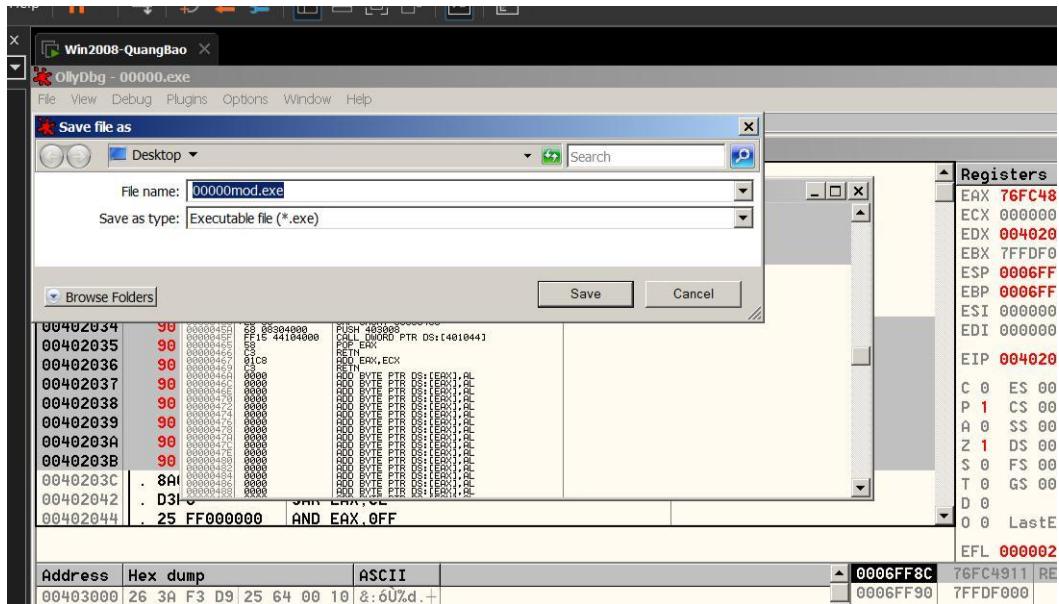
- In OllyDbg, in the top left pane, right-click and click "Copy to executable", "All modifications", as shown below



- 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.

Win2008-QuangBao

OllyDbg - 00000.exe

Administrator: Command Prompt

```

Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd Desktop
C:\Users\Administrator\Desktop>00000mod.exe
Launch codes?
1
Wow you got it. Here is the result: (J)
C:\Users\Administrator\Desktop>

```

Registers (FPU)

76FC48FF	kernel32
00000000	
00402006	00000
7FFDF000	
0006FF8C	
0006FF94	
00000000	
00000000	
00402006	00000
ES 0023 32bit	
CS 001B 32bit	
SS 0023 32bit	
DS 0023 32bit	
FS 003B 32bit	
GS 0000 NULL	

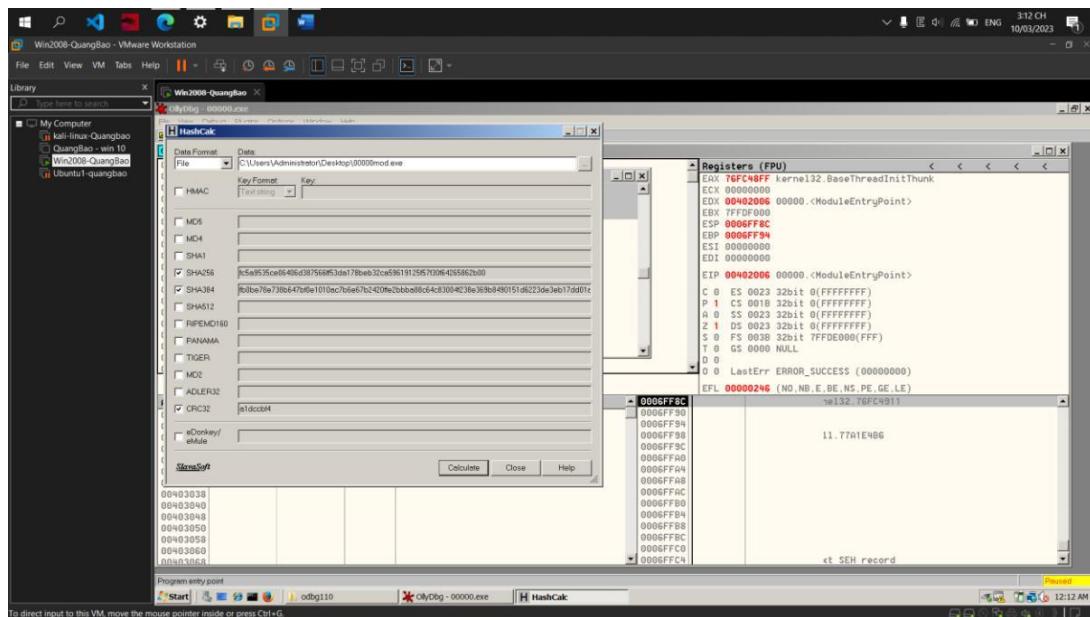
LastErr ERROR

00000246 (NO_N

0006FFB8
0006FFBC

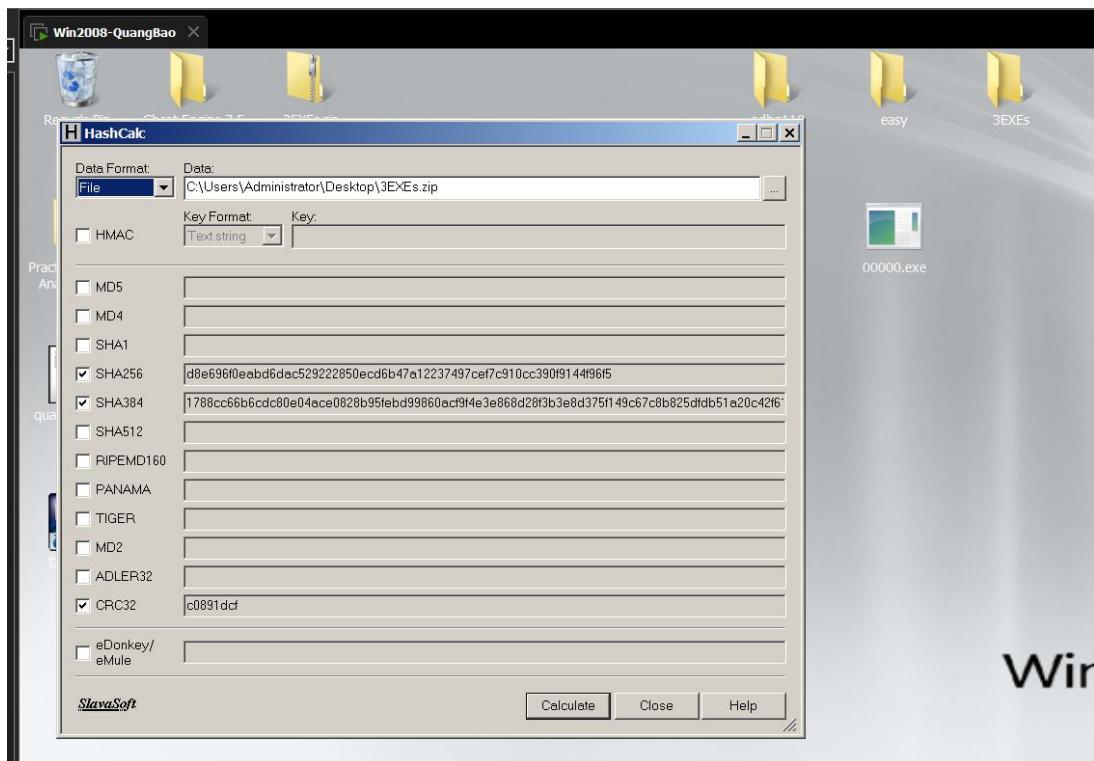
- **Checking the Hash**

- 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

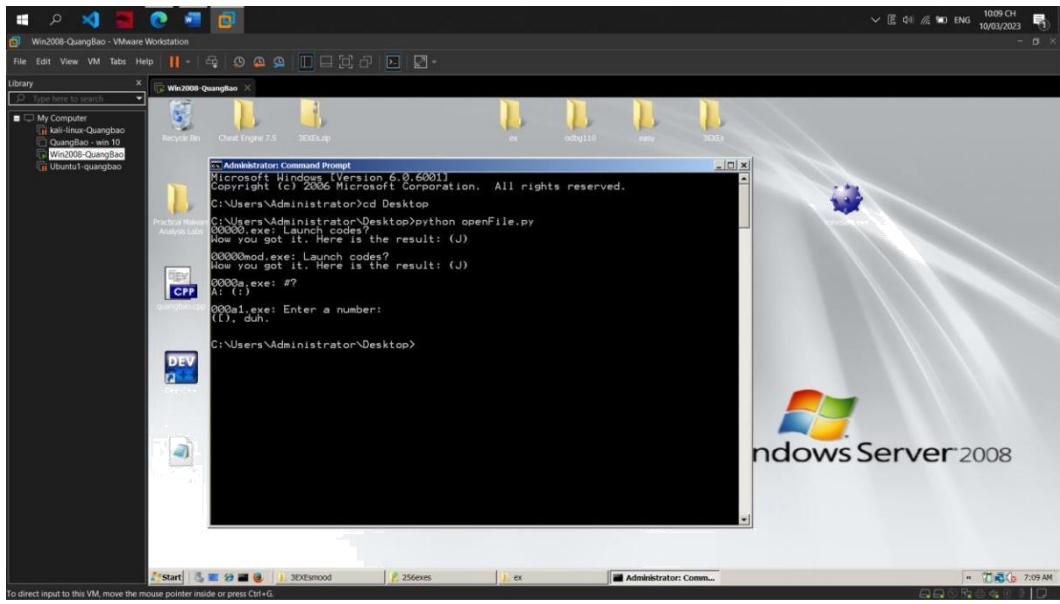


18.2.2: Patching three EXEs:

- **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 **0000.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 direotory:

```

p 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\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:

```

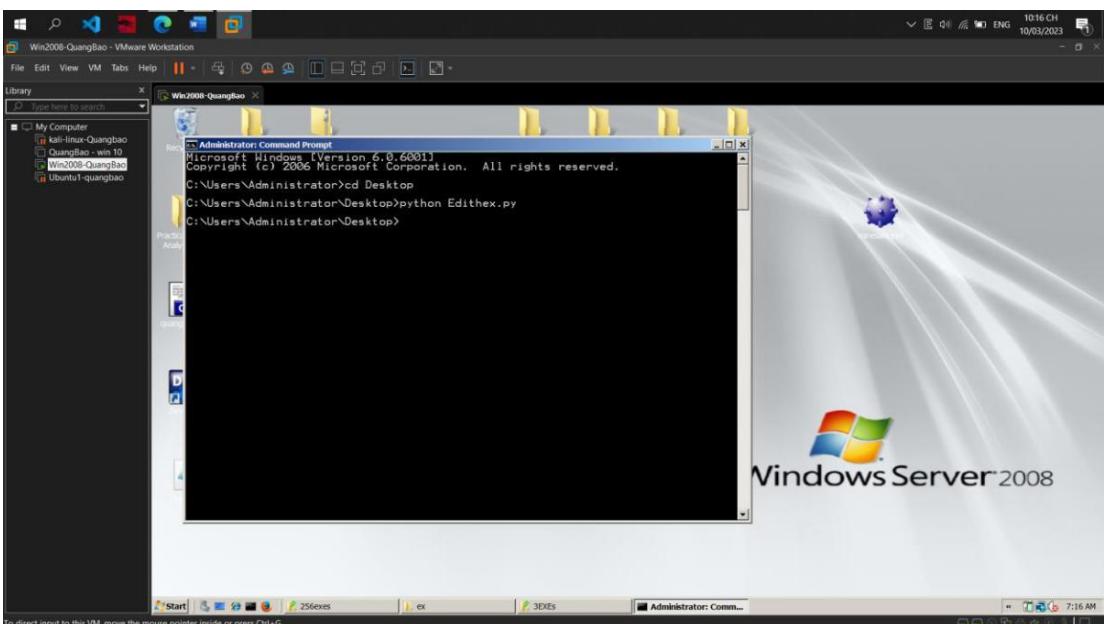
```

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

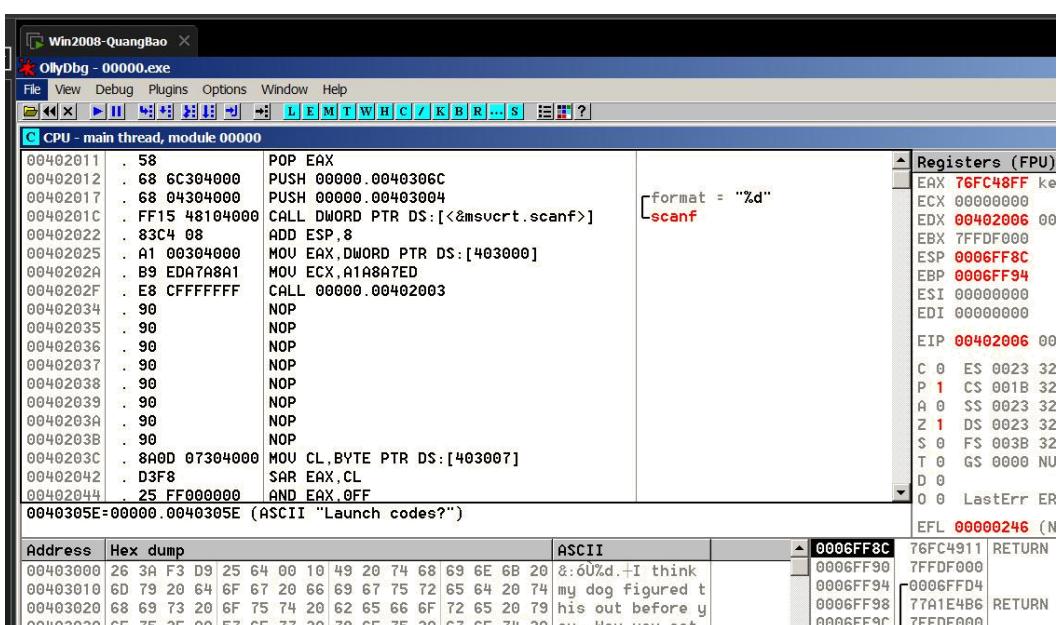
# write the modified file contents back to the
# file with open(filename, 'wb') as f:
f.write(file_contents)

```

- Now we run python script in cmd:



- Check file with Ollydbg



- Now we use another script to run multiple 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)
```

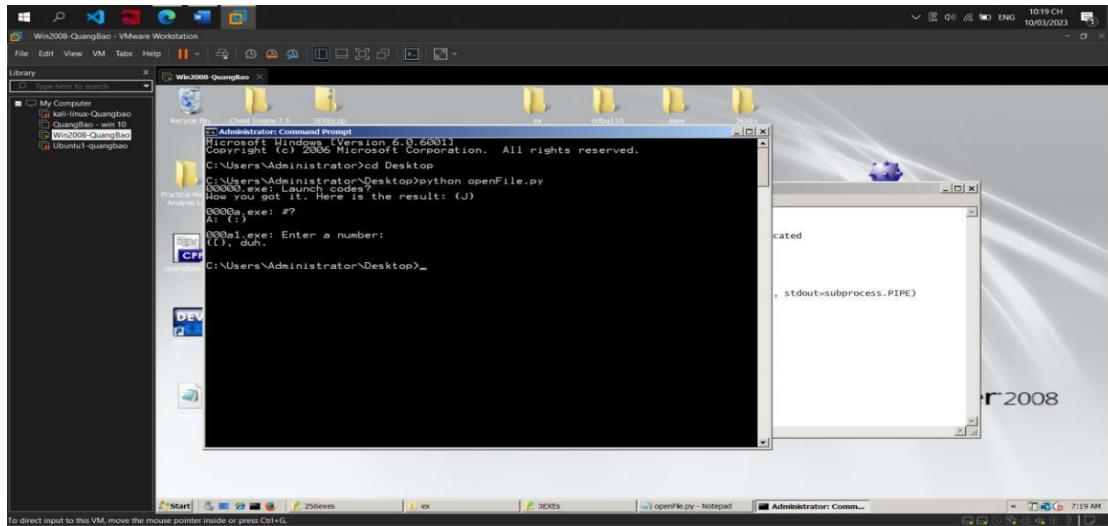
```
# send input to the subprocess  
proc.stdin.write(b'18\n')  
proc.stdin.close()
```

```
# 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])
```

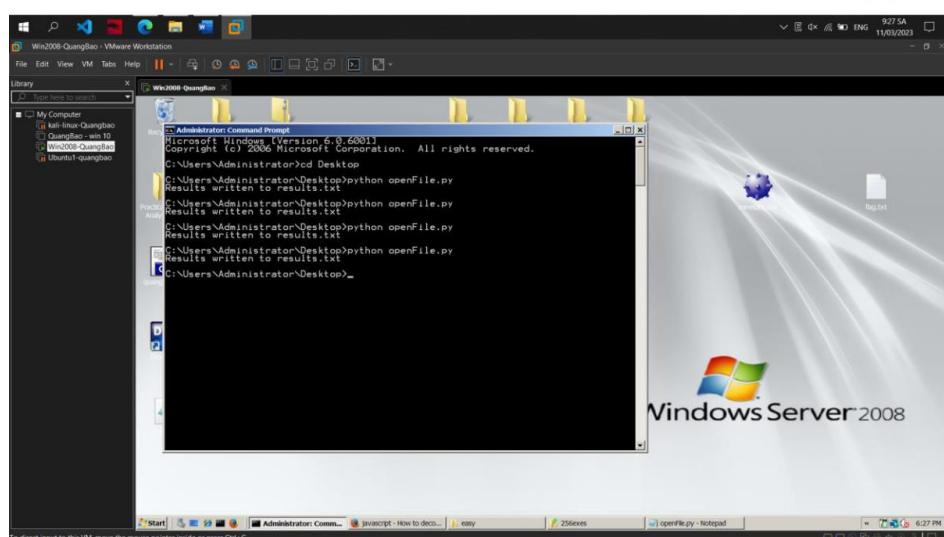
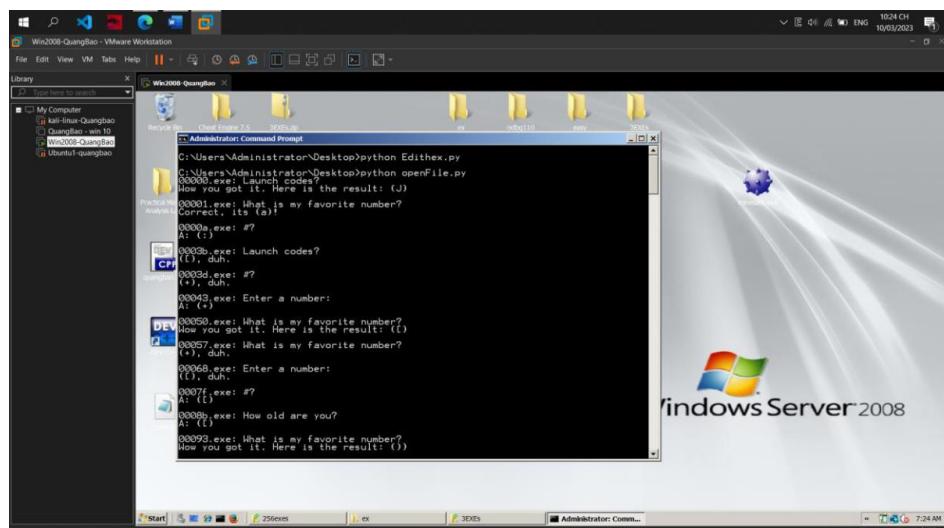
```
# write the concatenated results to a text  
file with open('results.txt', 'w') as f:  
    f.write("\n".join(results))
```

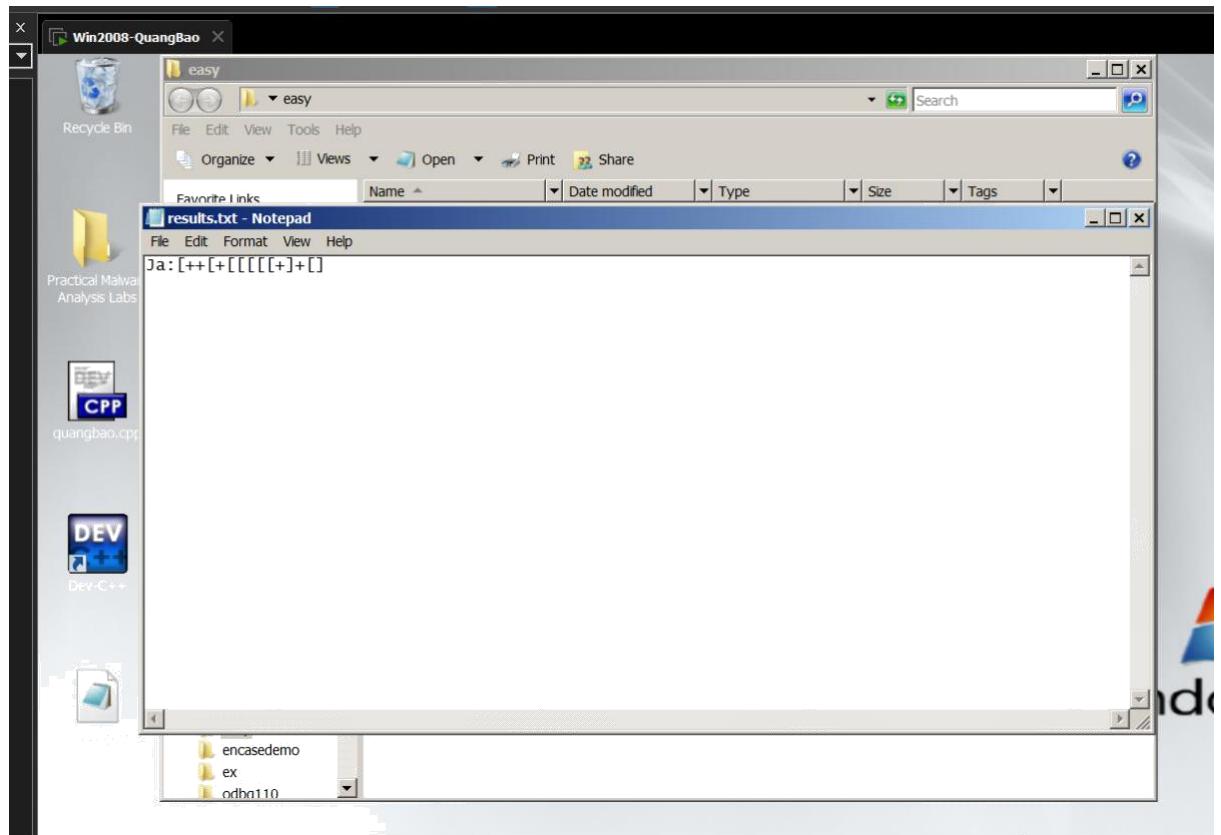
```
print('Results written to results.txt')
```



18.2.3: Patching 19 EXEs:

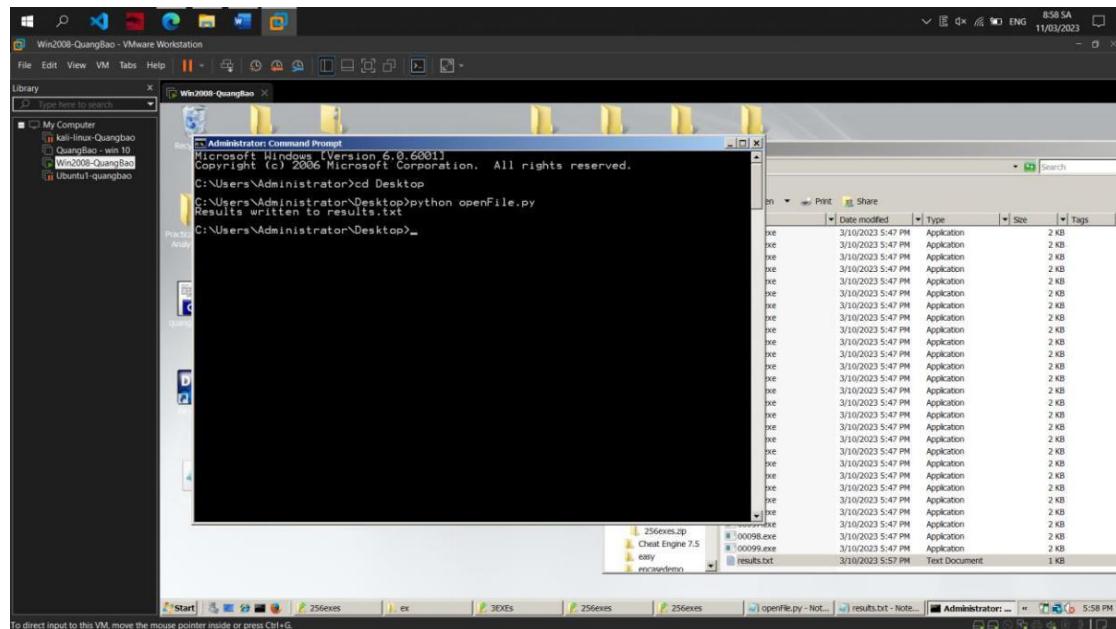
- **Getting the EXEs**
 - In the Documents folder of the VM handed out by your instructor, find the easy.zip file. Unzip it. There are 19 EXEs in it.
 - Patch all 19 files, run them, and combine the Results to get a 19-character flag





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

