Proj 9: Patching EXEs with Ollydbg (10 pts + 70 pts extra)

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 binaries.

9.1: Patching an EXE (15 pts)

Getting the EXE

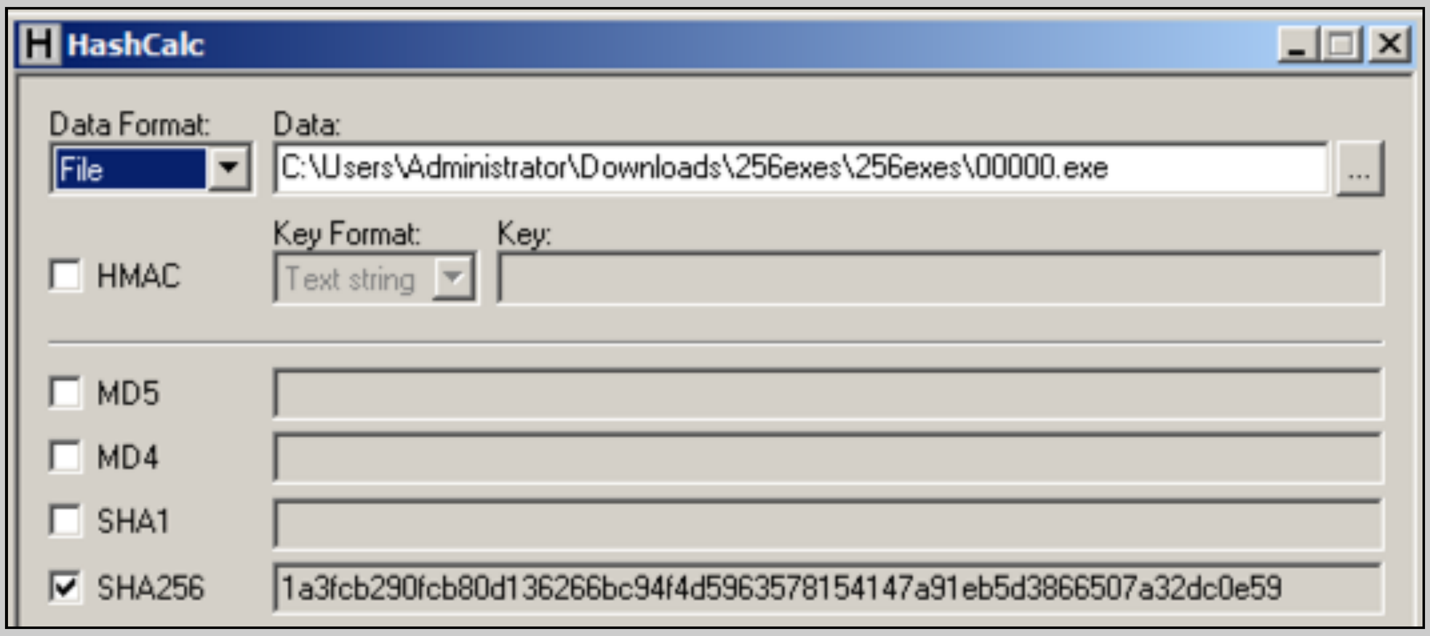
In the Documents folder of the VM handed out by your instructor, find the **00000.exe** file.

Checking the Hash

Click **Start**. Type **HASH** and click **HashCalc**. In HashCalc, make sure the **SHA256** box is checked, as shown below.

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.



Data Format: File

Data: C:\Users\Administrator\Downloads\256exes\256exes\00000.exe

☐ HMAC Key Format: Text string Key:

☐ MD5

☐ MD4

☐ SHA1

☑ SHA256 1a3fcb290fcb80d136266bc94f4d596357815414 7a91eb5d3866507a32dc0e59

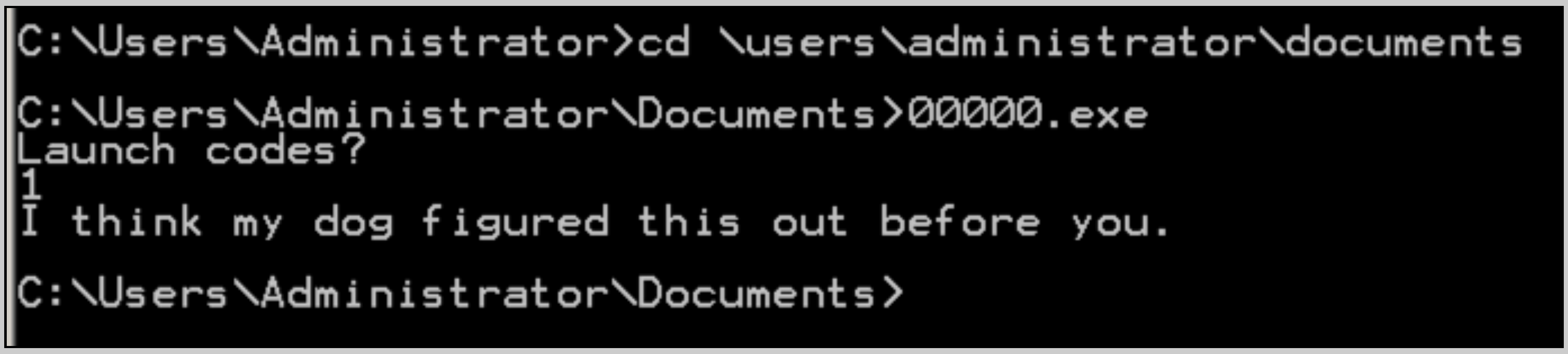
Running the EXE

Click the black square icon at the lower left of your desktop to open a Command Prompt.

Execute these commands:

```
cd \users\administrator\documents
00000.exe
```

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



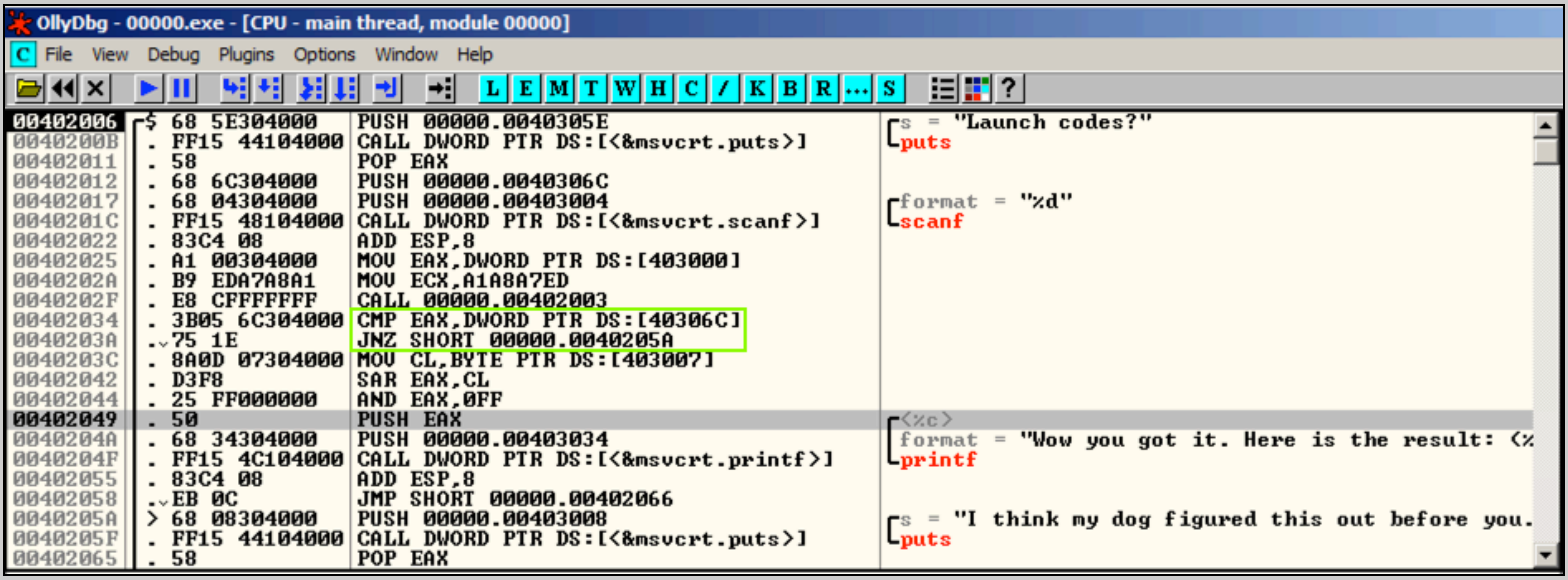
```
C:\Users\Administrator>cd \users\administrator\documents

C:\Users\Administrator\Documents>00000.exe
Launch codes?
1
I think my dog figured this out before you.

C:\Users\Administrator\Documents>
```

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.



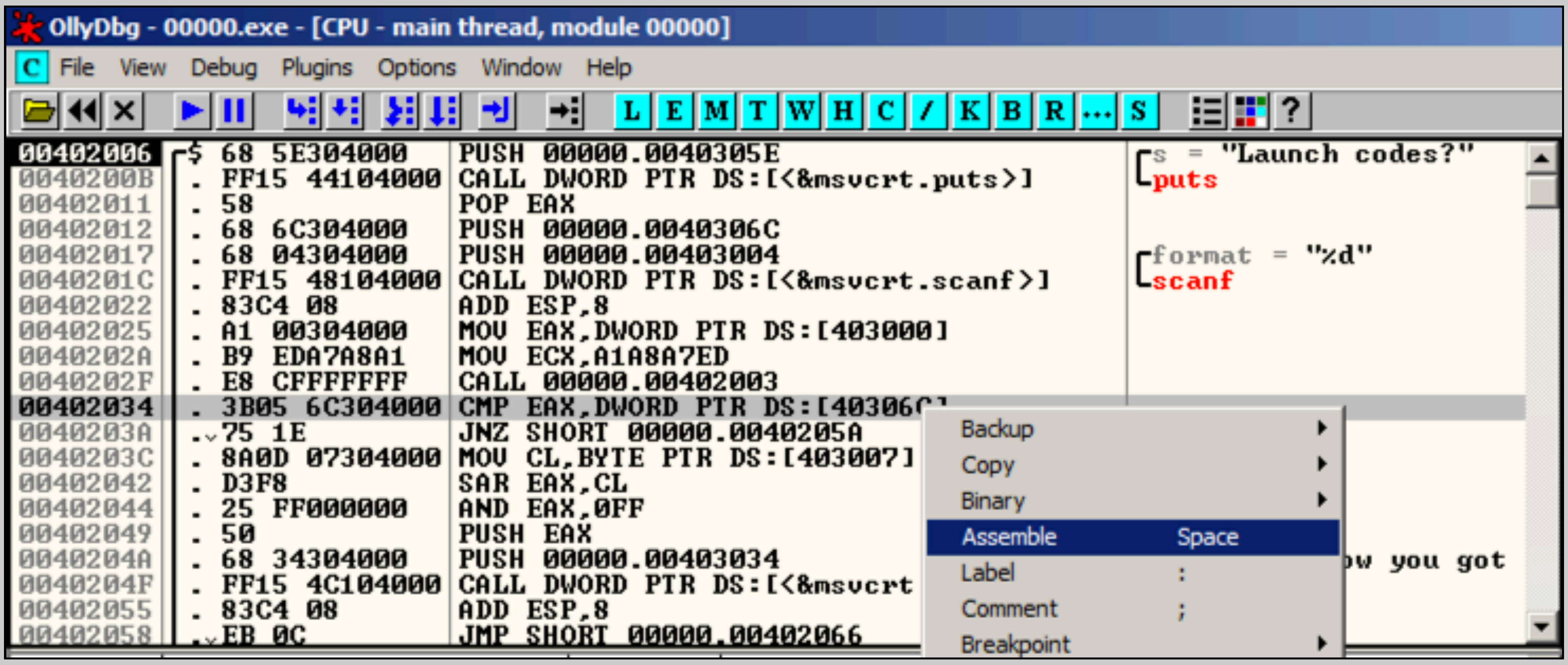
OllyDbg - 00000.exe - [CPU - main thread, module 00000]

```
00402006  ┌$ 68 5E304000    PUSH 00000.0040305E                     ┌s = "Launch codes?"
0040200B  . FF15 44104000   CALL DWORD PTR DS:[<&msvcrt.puts>]      └puts
00402011  . 58              POP EAX
00402012  . 68 6C304000     PUSH 00000.0040306C
00402017  . 68 04304000     PUSH 00000.00403004                     ┌format = "%d"
0040201C  . FF15 48104000   CALL DWORD PTR DS:[<&msvcrt.scanf>]     └scanf
00402022  . 83C4 08         ADD ESP,8
00402025  . A1 00304000     MOV EAX,DWORD PTR DS:[403000]
0040202A  . B9 EDA7A8A1     MOV ECX,A1A8A7ED
0040202F  . E8 CFFFFFFF     CALL 00000.00402003
00402034  . 3B05 6C304000   CMP EAX,DWORD PTR DS:[40306C]
0040203A  .v75 1E           JNZ SHORT 00000.0040205A
0040203C  . 8A0D 07304000   MOV CL,BYTE PTR DS:[403007]
00402042  . D3F8            SAR EAX,CL
00402044  . 25 FF000000     AND EAX,0FF
00402049  . 50              PUSH EAX                                ┌<%c>
0040204A  . 68 34304000     PUSH 00000.00403034                     │format = "Wow you got it. Here is the result: <%
0040204F  . FF15 4C104000   CALL DWORD PTR DS:[<&msvcrt.printf>]    └printf
00402055  . 83C4 08         ADD ESP,8
00402058  .vEB 0C           JMP SHORT 00000.00402066
0040205A  > 68 08304000     PUSH 00000.00403008                     ┌s = "I think my dog figured this out before you.
0040205F  . FF15 44104000   CALL DWORD PTR DS:[<&msvcrt.puts>]      └puts
00402065  . 58              POP EAX
```

C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S

Modifying the EXE

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

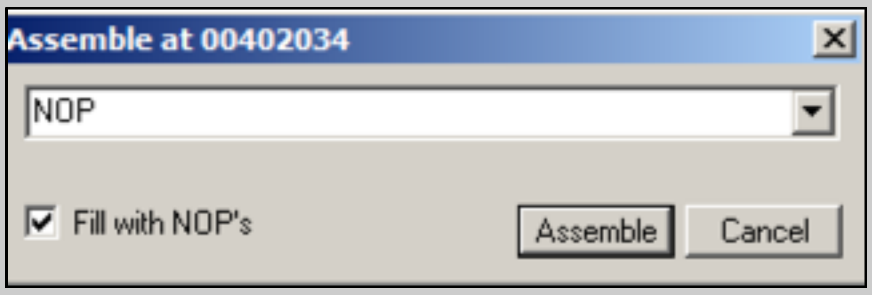


OllyDbg - 00000.exe - [CPU - main thread, module 00000]

```
00402006  ┌$ 68 5E304000    PUSH 00000.0040305E                     ┌s = "Launch codes?"
0040200B  . FF15 44104000   CALL DWORD PTR DS:[<&msvcrt.puts>]      └puts
00402011  . 58              POP EAX
00402012  . 68 6C304000     PUSH 00000.0040306C
00402017  . 68 04304000     PUSH 00000.00403004                     ┌format = "%d"
0040201C  . FF15 48104000   CALL DWORD PTR DS:[<&msvcrt.scanf>]     └scanf
00402022  . 83C4 08         ADD ESP,8
00402025  . A1 00304000     MOV EAX,DWORD PTR DS:[403000]
0040202A  . B9 EDA7A8A1     MOV ECX,A1A8A7ED
0040202F  . E8 CFFFFFFF     CALL 00000.00402003
00402034  . 3B05 6C304000   CMP EAX,DWORD PTR DS:[40306C
0040203A  .v75 1E           JNZ SHORT 00000.0040205A
0040203C  . 8A0D 07304000   MOV CL,BYTE PTR DS:[403007]
00402042  . D3F8            SAR EAX,CL
00402044  . 25 FF000000     AND EAX,0FF
00402049  . 50              PUSH EAX
0040204A  . 68 34304000     PUSH 00000.00403034                                 ow you got
0040204F  . FF15 4C104000   CALL DWORD PTR DS:[<&msvcrt
00402055  . 83C4 08         ADD ESP,8
00402058  .vEB 0C           JMP SHORT 00000.00402066
```

C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Backup
Copy
Binary
Assemble Space
Label :
Comment ;
Breakpoint

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



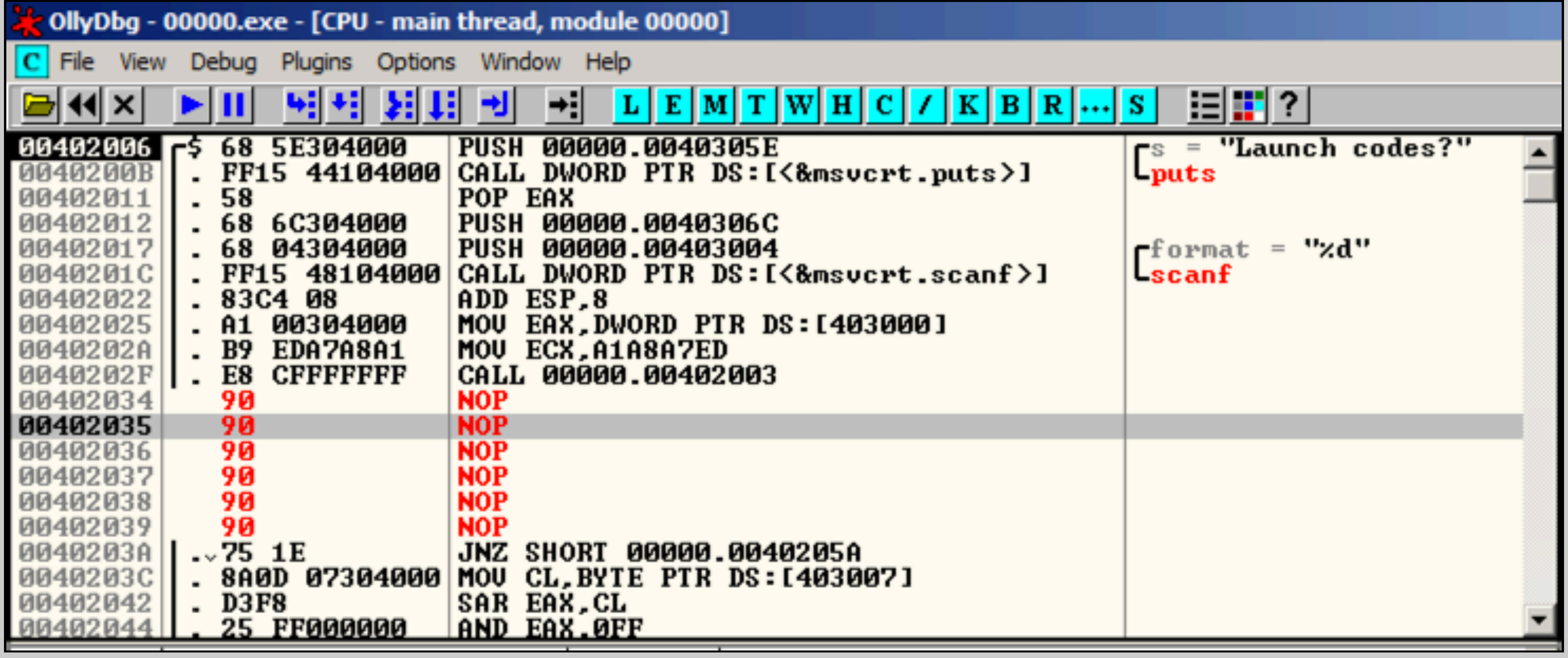
Assemble at 00402034

NOP

☑ Fill with NOP's

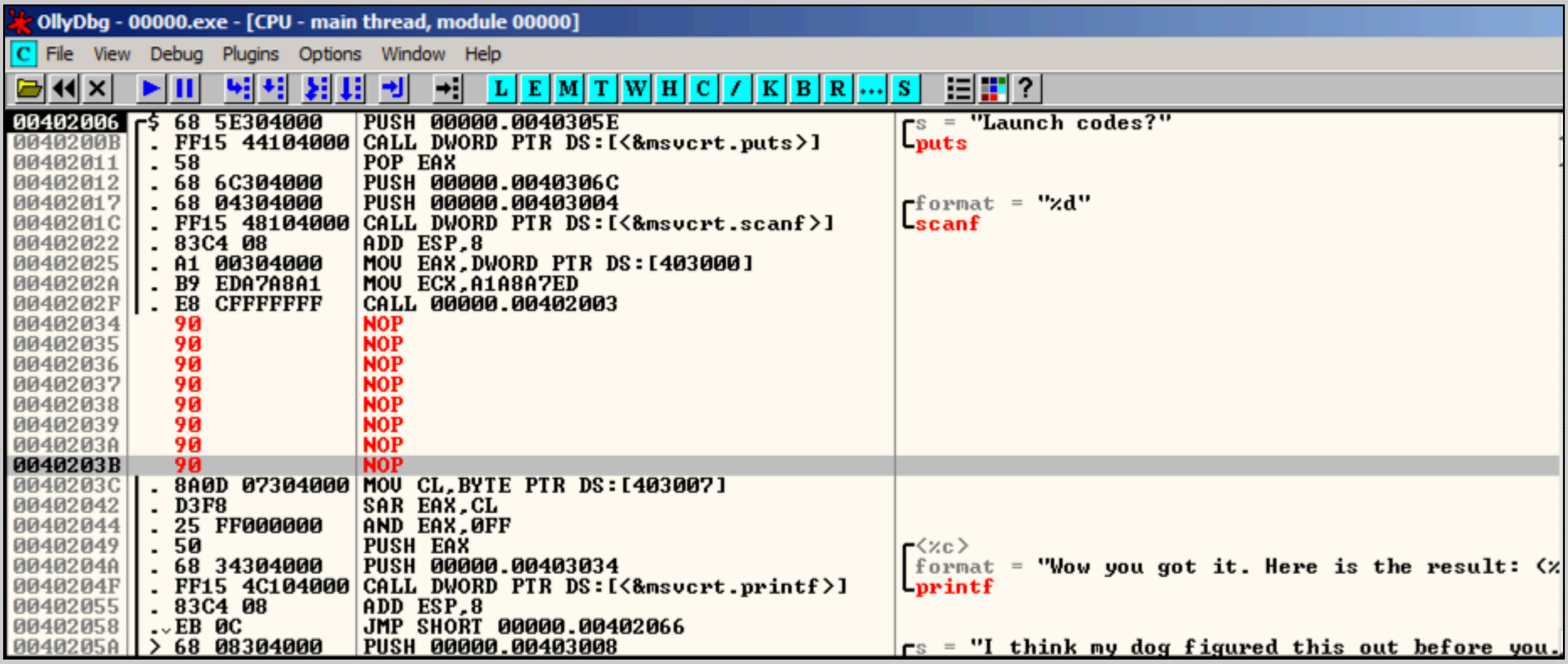
Click the **Assemble** button. Click the **Cancel** button.

The CMP instruction is replaced by a series of NOPs, as shown below.


OllyDbg - 00000.exe - [CPU - main thread, module 00000]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S

```
00402006  $ 68 5E304000   PUSH 00000.0040305E                         s = "Launch codes?"
0040200B  . FF15 44104000 CALL DWORD PTR DS:[<&msvcrt.puts>]          puts
00402011  . 58            POP EAX
00402012  . 68 6C304000   PUSH 00000.0040306C
00402017  . 68 04304000   PUSH 00000.00403004                         format = "%d"
0040201C  . FF15 48104000 CALL DWORD PTR DS:[<&msvcrt.scanf>]         scanf
00402022  . 83C4 08       ADD ESP,8
00402025  . A1 00304000   MOV EAX,DWORD PTR DS:[403000]
0040202A  . B9 EDA7A8A1   MOV 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  .v75 1E         JNZ SHORT 00000.0040205A
0040203C  . 8A0D 07304000 MOV CL,BYTE PTR DS:[403007]
00402042  . D3F8          SAR EAX,CL
00402044  . 25 FF000000   AND EAX,0FF
```

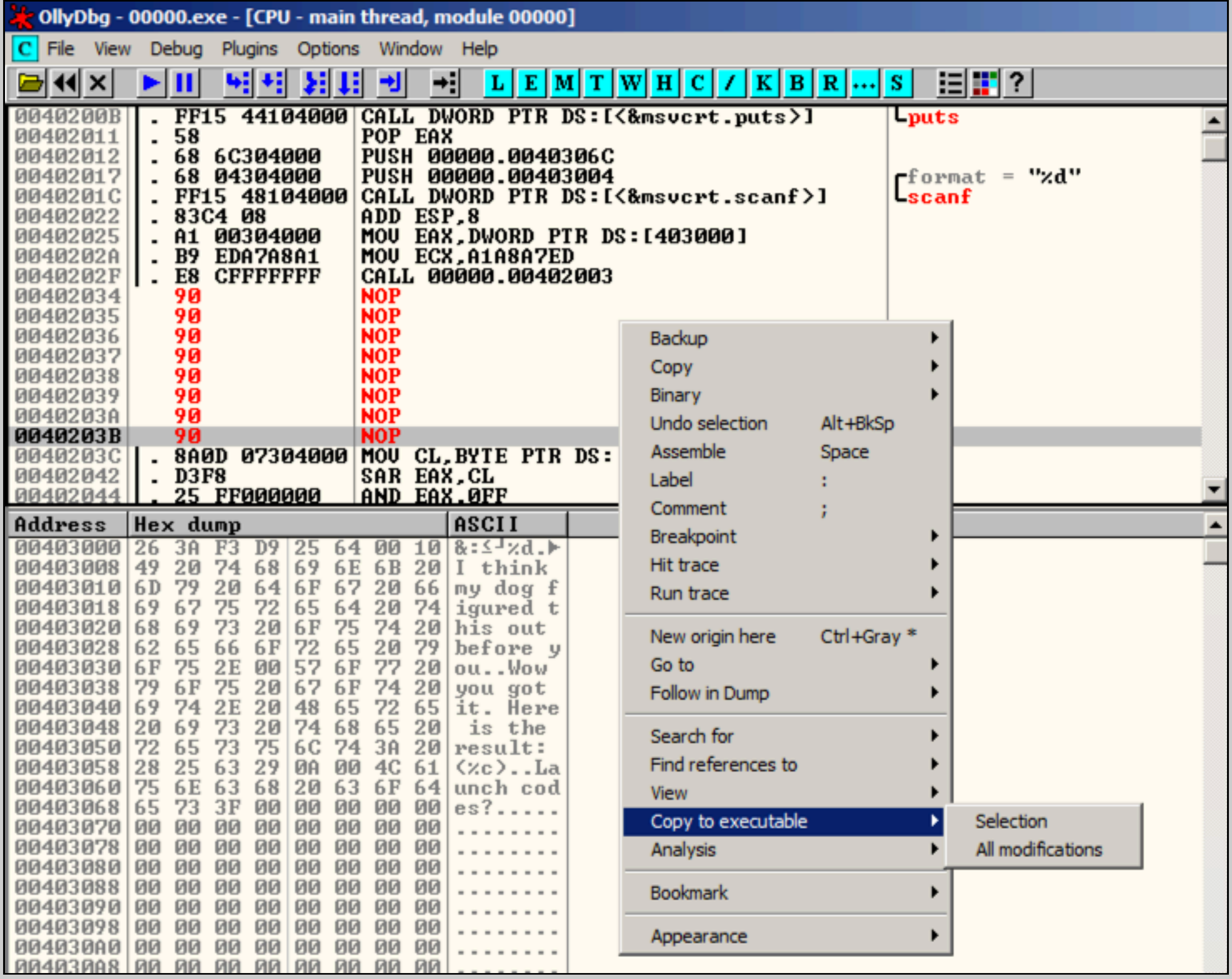
Repeat the process to replace the JNZ instruction with NOPs also, as shown below.


OllyDbg - 00000.exe - [CPU - main thread, module 00000]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S

```
00402006  $ 68 5E304000   PUSH 00000.0040305E                         s = "Launch codes?"
0040200B  . FF15 44104000 CALL DWORD PTR DS:[<&msvcrt.puts>]          puts
00402011  . 58            POP EAX
00402012  . 68 6C304000   PUSH 00000.0040306C
00402017  . 68 04304000   PUSH 00000.00403004                         format = "%d"
0040201C  . FF15 48104000 CALL DWORD PTR DS:[<&msvcrt.scanf>]         scanf
00402022  . 83C4 08       ADD ESP,8
00402025  . A1 00304000   MOV EAX,DWORD PTR DS:[403000]
0040202A  . B9 EDA7A8A1   MOV 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 MOV CL,BYTE PTR DS:[403007]
00402042  . D3F8          SAR EAX,CL
00402044  . 25 FF000000   AND EAX,0FF
00402049  . 50            PUSH EAX                                    <%c>
0040204A  . 68 34304000   PUSH 00000.00403034                         format = "Wow you got it. Here is the result: <%
0040204F  . FF15 4C104000 CALL DWORD PTR DS:[<&msvcrt.printf>]        printf
00402055  . 83C4 08       ADD ESP,8
00402058  .vEB 0C         JMP SHORT 00000.00402066
0040205A  > 68 08304000   PUSH 00000.00403008                         s = "I think my dog figured this out before you.
```

Saving the Modified File

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



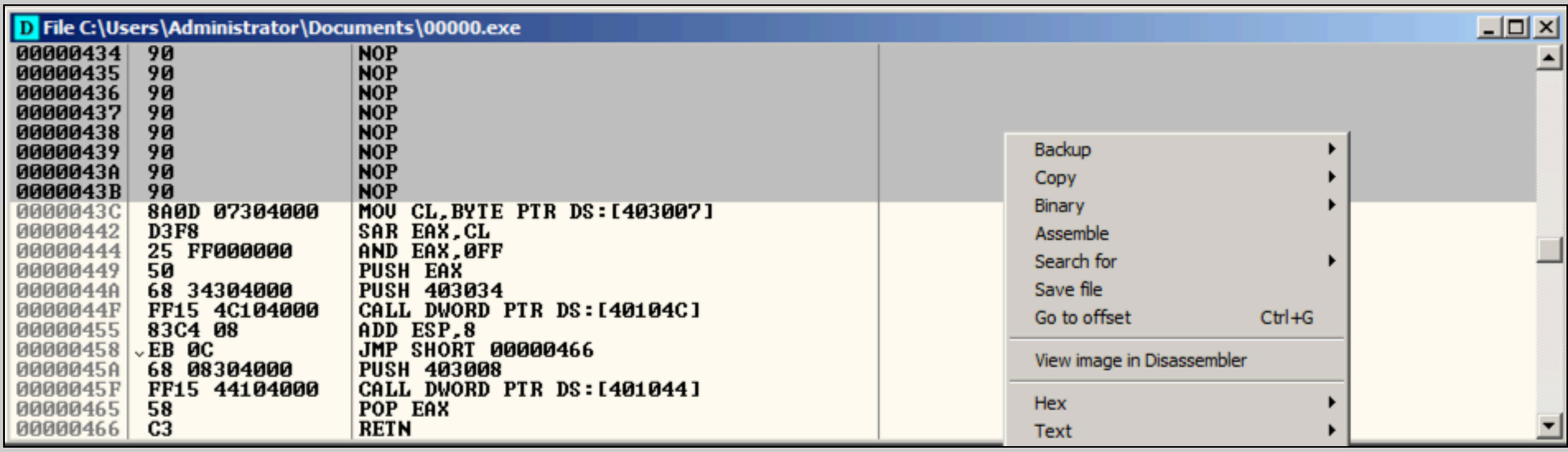
```
0040200B  . FF15 44104000  CALL DWORD PTR DS:[<&msvcrt.puts>]   └puts
00402011  . 58             POP EAX
00402012  . 68 6C304000    PUSH 00000.0040306C
00402017  . 68 04304000    PUSH 00000.00403004                  ┌format = "%d"
0040201C  . FF15 48104000  CALL DWORD PTR DS:[<&msvcrt.scanf>]  └scanf
00402022  . 83C4 08        ADD ESP,8
00402025  . A1 00304000    MOV EAX,DWORD PTR DS:[403000]
0040202A  . B9 EDA7A8A1    MOV 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  MOV CL,BYTE PTR DS:
00402042  . D3F8           SAR EAX,CL
00402044  . 25 FF000000    AND EAX,0FF
```

```
Address  Hex dump                 ASCII
00403000 26 3A F3 D9 25 64 00 10  &:≤┘%d.►
00403008 49 20 74 68 69 6E 6B 20  I think
00403010 6D 79 20 64 6F 67 20 66  my dog f
00403018 69 67 75 72 65 64 20 74  igured t
00403020 68 69 73 20 6F 75 74 20  his out
00403028 62 65 66 6F 72 65 20 79  before y
00403030 6F 75 2E 00 57 6F 77 20  ou..Wow
00403038 79 6F 75 20 67 6F 74 20  you got
00403040 69 74 2E 20 48 65 72 65  it. Here
00403048 20 69 73 20 74 68 65 20   is the
00403050 72 65 73 75 6C 74 3A 20  result:
00403058 28 25 63 29 0A 00 4C 61  (%c)..La
00403060 75 6E 63 68 20 63 6F 64  unch cod
00403068 65 73 3F 00 00 00 00 00  es?.....
00403070 00 00 00 00 00 00 00 00  ........
00403078 00 00 00 00 00 00 00 00  ........
00403080 00 00 00 00 00 00 00 00  ........
00403088 00 00 00 00 00 00 00 00  ........
00403090 00 00 00 00 00 00 00 00  ........
00403098 00 00 00 00 00 00 00 00  ........
004030A0 00 00 00 00 00 00 00 00  ........
004030A8 00 00 00 00 00 00 00 00  ........
```

A "Copy selection to executable file?" box pops up. Click the "**Copy all**" button.

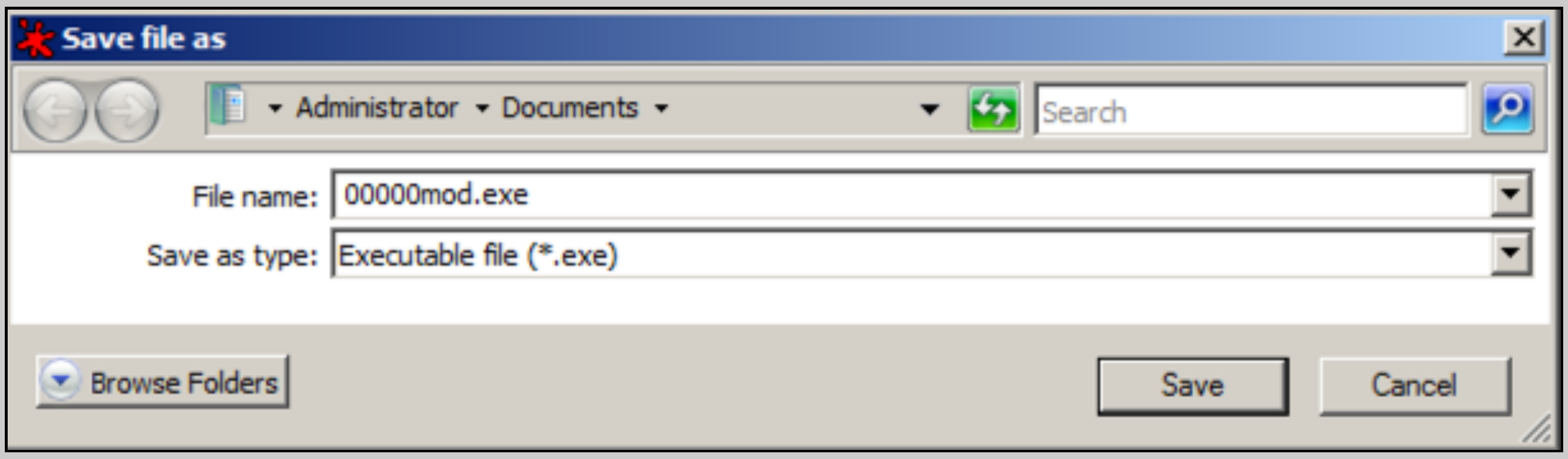
A "File" box appears, as shown below.

Right-click in it and click "**Save file**".


D File C:\Users\Administrator\Documents\00000.exe
Backup
Copy
Binary
Assemble
Search for
Save file
Go to offset Ctrl+G
View image in Disassembler
Hex
Text

```
00000434   90             NOP
00000435   90             NOP
00000436   90             NOP
00000437   90             NOP
00000438   90             NOP
00000439   90             NOP
0000043A   90             NOP
0000043B   90             NOP
0000043C   8A0D 07304000  MOV CL,BYTE PTR DS:[403007]
00000442   D3F8           SAR EAX,CL
00000444   25 FF000000    AND EAX,0FF
00000449   50             PUSH EAX
0000044A   68 34304000    PUSH 403034
0000044F   FF15 4C104000  CALL DWORD PTR DS:[40104C]
00000455   83C4 08        ADD ESP,8
00000458  ˅EB 0C          JMP SHORT 00000466
0000045A   68 08304000    PUSH 403008
0000045F   FF15 44104000  CALL DWORD PTR DS:[401044]
00000465   58             POP EAX
00000466   C3             RETN
```

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



File name: 00000mod.exe

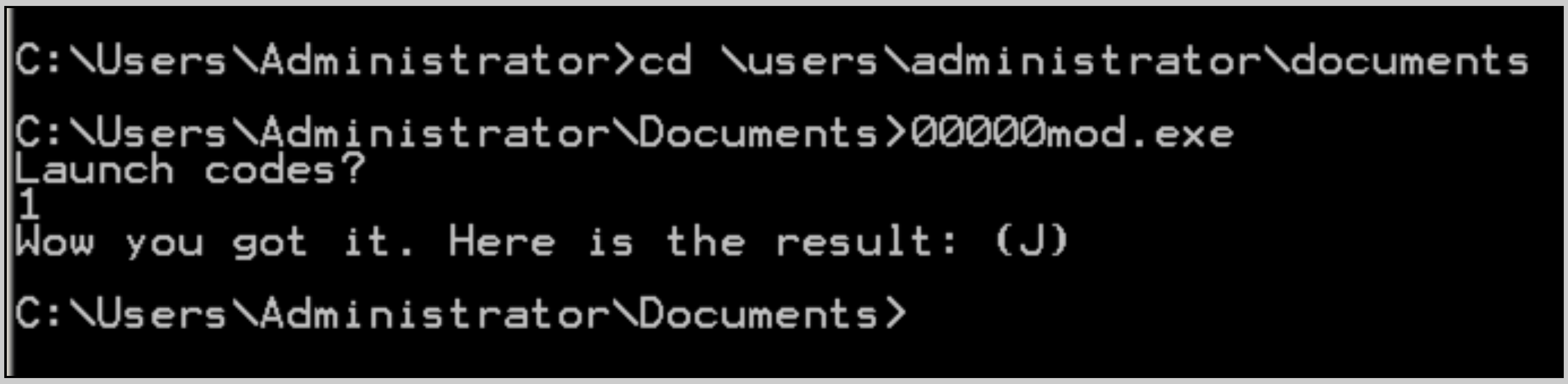
Save as type: Executable file (*.exe)

Running the Modified File

In a Command Prompt window, execute these commands:

```
cd \users\administrator\documents
00000mod.exe
```

It asks for a "Launch code". Enter **1**. It accepts the code now, as shown below.

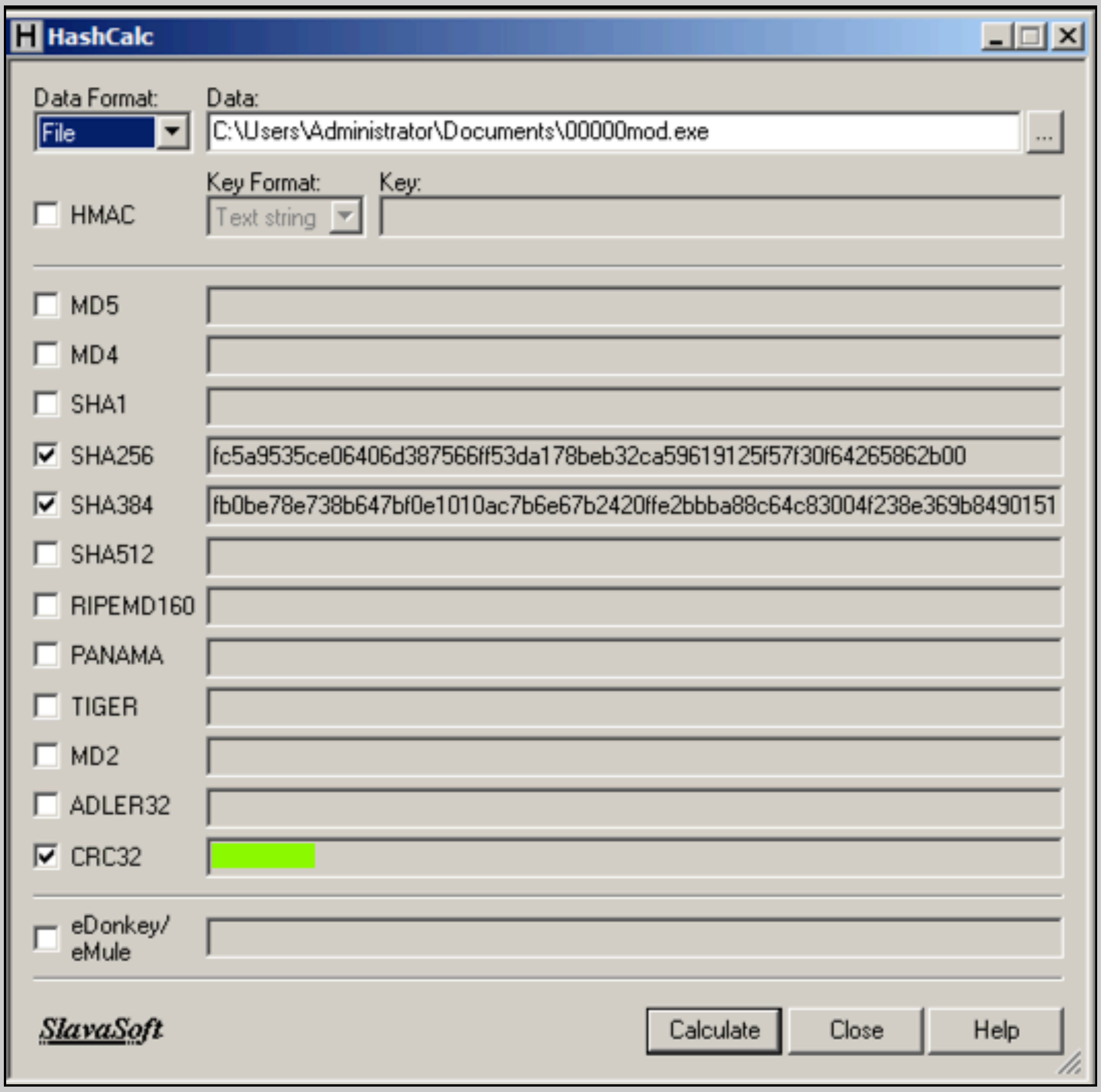


```
C:\Users\Administrator>cd \users\administrator\documents

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

C:\Users\Administrator\Documents>
```

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.



Data Format: File

Data: C:\Users\Administrator\Documents\00000mod.exe

☐ HMAC Key Format: Text string Key:

☐ MD5

☐ MD4

☐ SHA1

☑ SHA256 fc5a9535ce06406d387566ff53da178beb32ca59619125f57f30f64265862b00

☑ SHA384 fb0be78e738b647bf0e1010ac7b6e67b2420ffe2bbba88c64c83004f238e369b8490151

☐ SHA512

☐ RIPEMD160

☐ PANAMA

☐ TIGER

☐ MD2

☐ ADLER32

☑ CRC32 [redacted]

☐ eDonkey/ eMule

9.1: Recording Your Success (10 pts)

Use the form below to record your score in Canvas.

Name or Email:

CRC32:

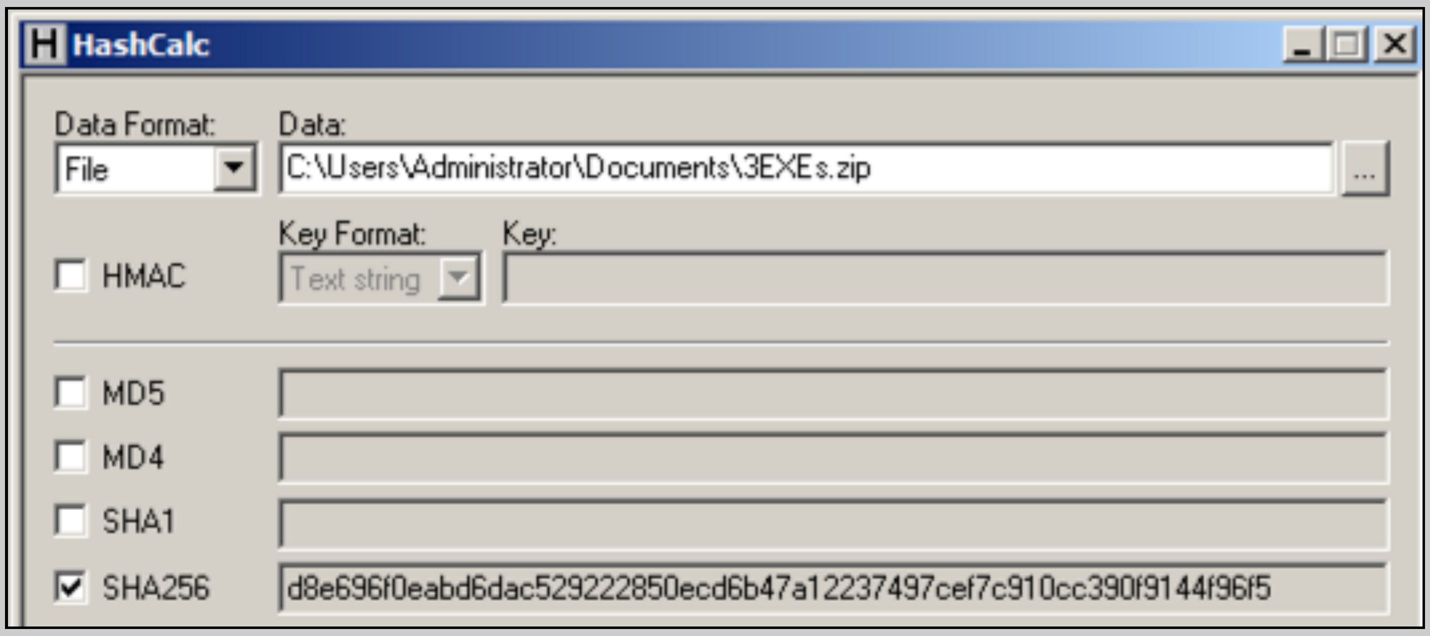
9.2: Patching Three EXEs (10 pts extra)

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.



Data Format: File

Data: C:\Users\Administrator\Documents\3EXEs.zip

□ HMAC Key Format: Text string Key:

□ MD5

□ MD4

□ SHA1

☑ SHA256 d8e696f0eabd6dac529222850ecd6b47a12237497cef7c910cc390f9144f96f5

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**
- File **000a1.exe** Result **T**

If those were the results, the answer would be **CAT**

The actual results are different, of course.

9.2: Recording Your Success (10 pts extra)

Use the form below to record your score in Canvas.

Name or Email:

Results: 3 Characters like this: `CAT`

SUBMIT

9.3: Patching 19 EXEs (30 pts extra)

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.

Goal

Patch all 19 files, run them, and combine the Results to get a 19-character flag.

Hints

There are hints here.

9.3: Recording Your Success (30 pts extra)

Use the form below to record your score in Canvas.

Name or Email:

Results: 19 Characters like this:

Impenetrable!Cyber!

SUBMIT

9.4: Patching 256 EXEs (30 pts extra)

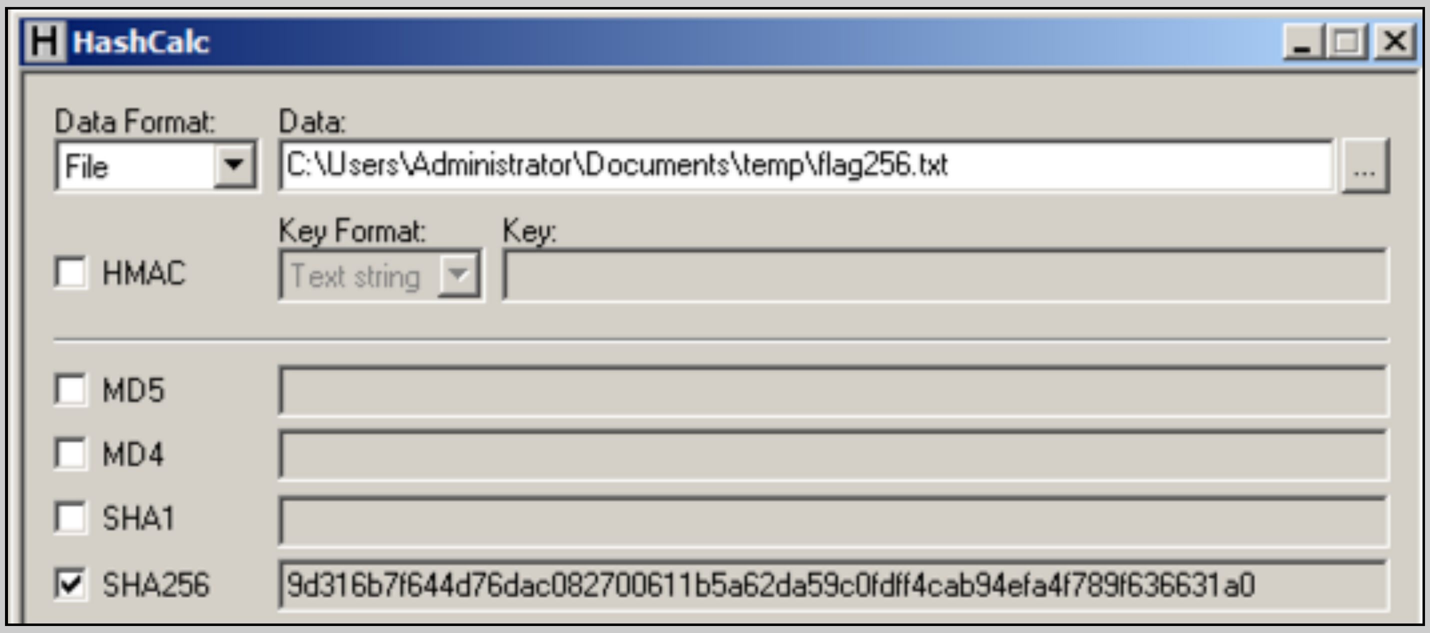
Getting the EXEs

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.

Goal: Gather the Results

Patch all 256 files and run them. Each file will give you one "Result" character. Gather all those characters into a file 256 bytes long.

Calculate the SHA256 hash of that file. It should match the value shown below.



Data Format: File

Data: C:\Users\Administrator\Documents\temp\flag256.txt

□ HMAC

Key Format: Text string

Key:

□ MD5

□ MD4

□ SHA1

☑ SHA256 9d316b7f644d76dac082700611b5a62da59c0fdff4cab94efa4f789f636631a0

Calculate the CRC32 of that file to win.

9.4: Recording Your Success (30 pts extra)

Use the form below to record your score in Canvas.

Name or Email:

CRC32 hash like this: `07b01710`

SUBMIT

Credit

This is based on the 67k Challenge from EasyCTF 2017.

Modified 7-12-17 11:35 am
Integrated with Canvas 9-11-18