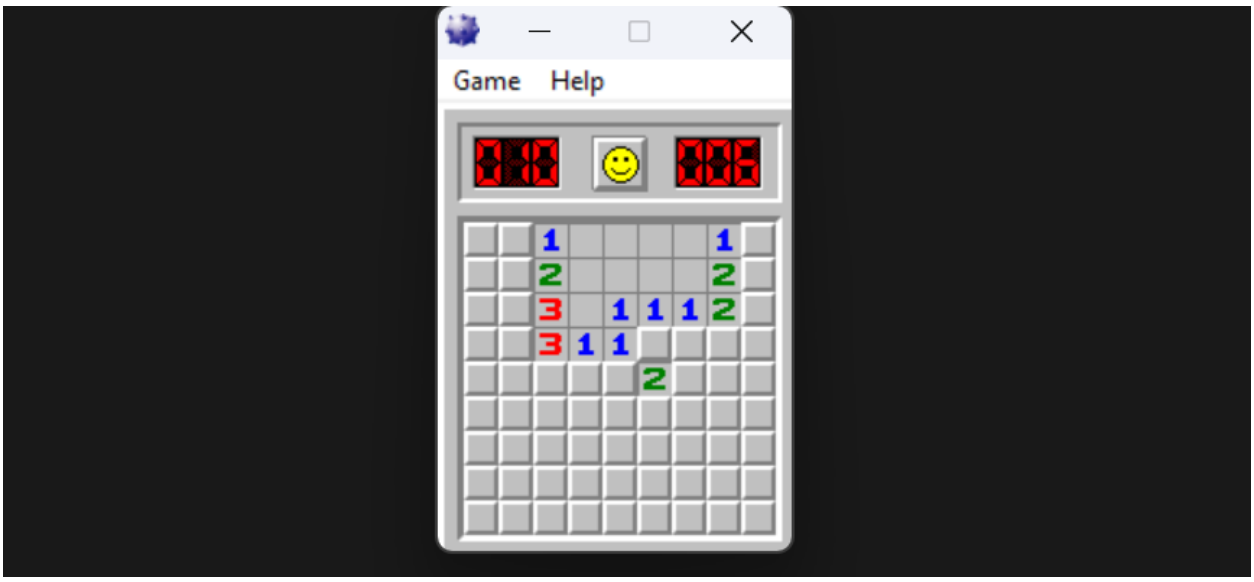
Lab 18.1: Hacking Minesweeper with Ollydbg

python --version

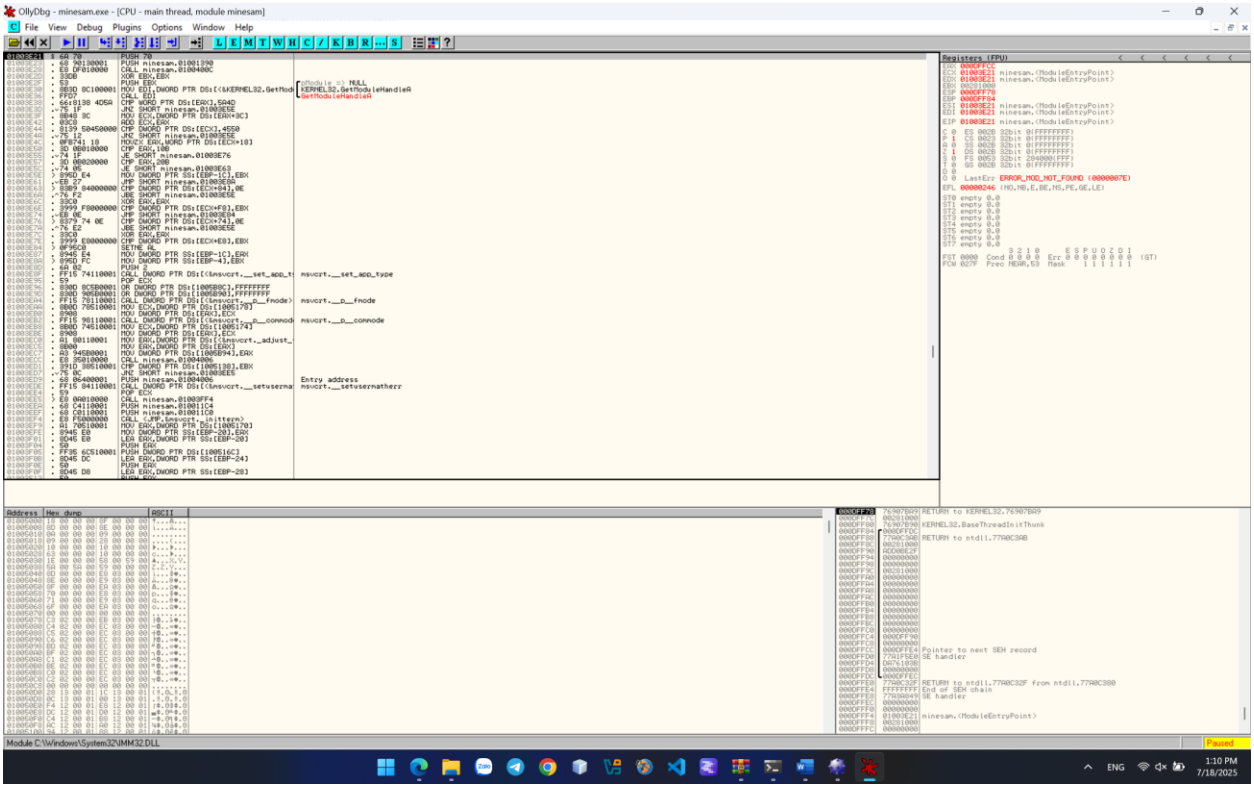
```
PS C:\Users\Lenovo legion 5>  python --version
Python 2.7.18
PS C:\Users\Lenovo legion 5>
```

Getting Minesweeper

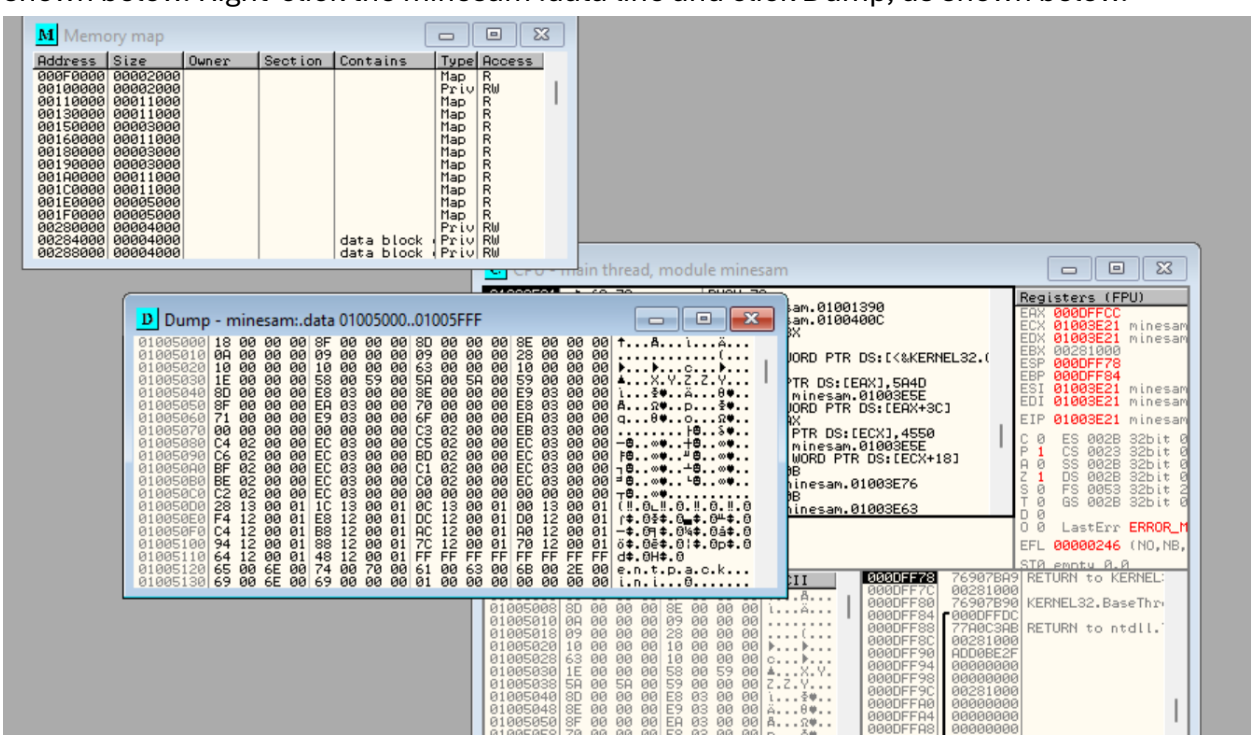


NGUYEN NAM KHANH – HE191159 – IA1902 – IAM302

Viewing the Game in OllyDbg



From the OllyDbg menu bar, click View, Memory. The memory segments are shown, as shown below. Right-click the minesam .data line and click Dump, as shown below.



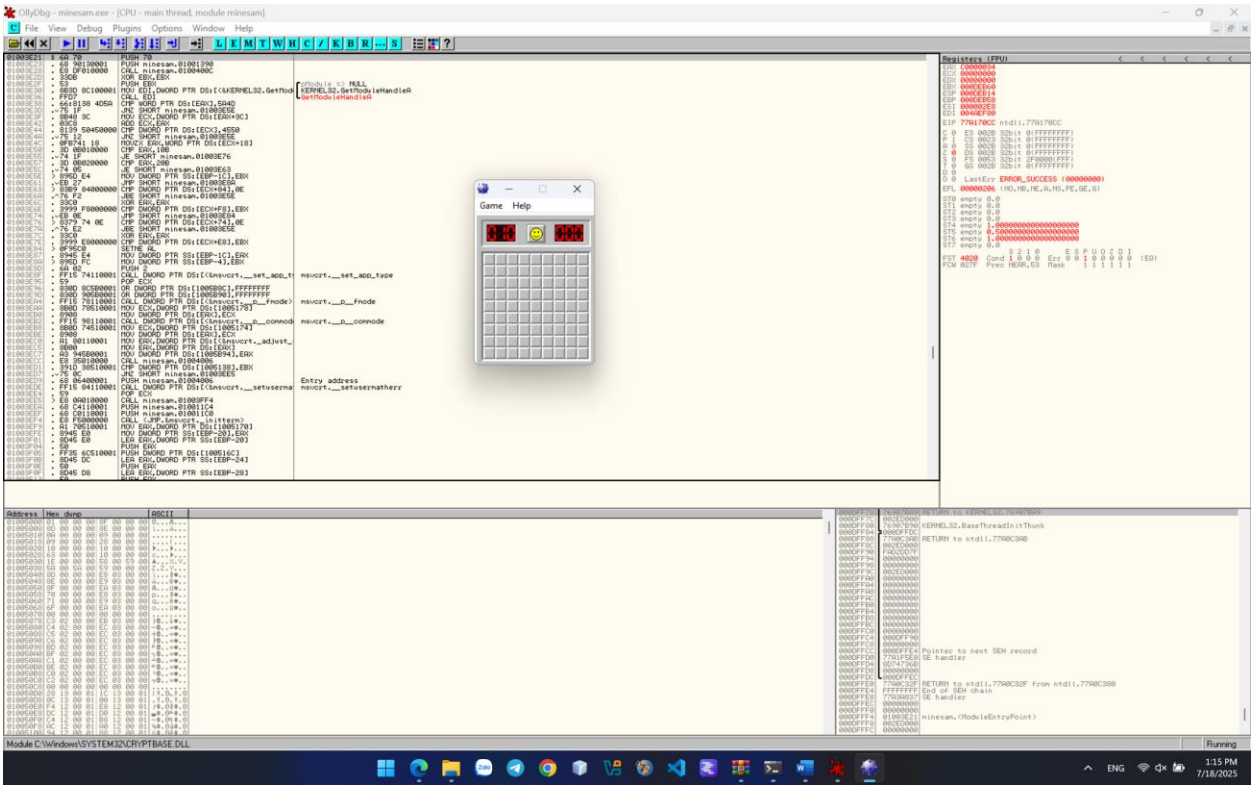
In the Dump window, scroll down to show memory near 01005340. This area contains only zeroes, as shown below

```
01005230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005290 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010052A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010052B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010052C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010052D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010052E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010052F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005310 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005320 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005330 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005340 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005350 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005360 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005370 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005380 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
01005390 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010053A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010053B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010053C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010053D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
010053E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

o From the OllyDbg menu bar, click View, CPU.

o From the OllyDbg menu bar, click Debug, Run.

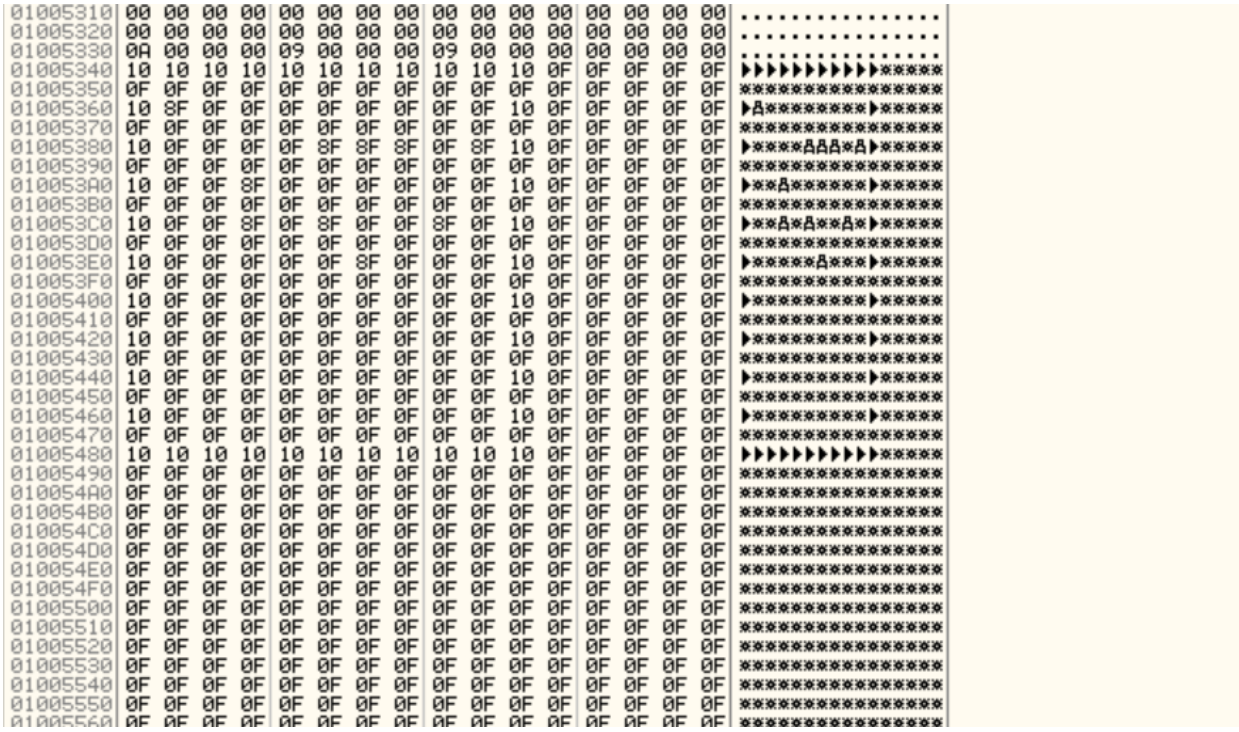
o A Minesweeper window opens, but does not come to the front. Click its button on the taskbar to bring it to the front, as shown below



- **Viewing the Stored Gameboard**

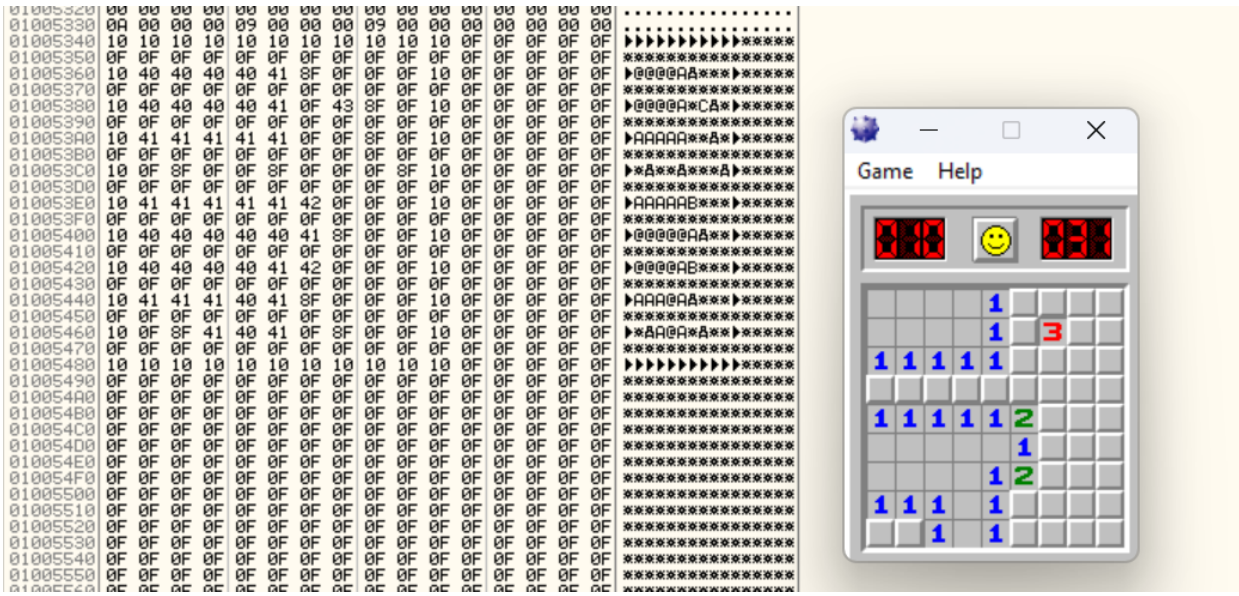
  o From the OllyDbg menu bar, click Window, Dump.

  o The memory after 01005340 now contains data, as shown below



o Click the Minesweeper button on the taskbar to bring it to the front.

o Compare the Minesweeper gameboard with the Dump window. You can see that the gameboard is stored in RAM, using an "A" for "1", and a "B" for "2", as shown below.



o If we can read the RAM, we can cheat at the game.

o Notice the green-highlighted region in the image above. If we can find this sequence of bytes in RAM, we can find the gameboard in a memory dump

- **Getting Procdump**

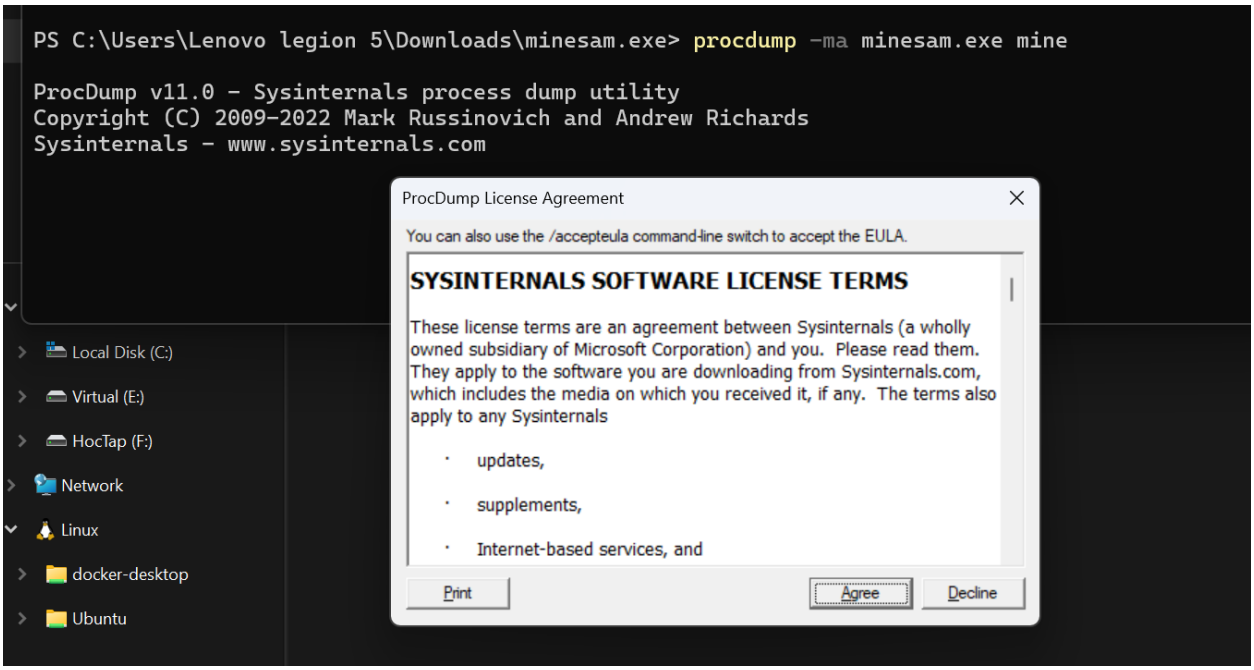
- **Capturing Process Memory**

o Close Minesweeper. Close OllyDbg. Double-click minesam.exe to run

o Open a Command Prompt and execute these commands: cd C:\Users\Administrator\Deskstop\ procdump -ma minesam.exe mine

o A box pops up, titled ProcDump License Agreement. Click Agree.

o Procdump makes a dump file, as shown below.



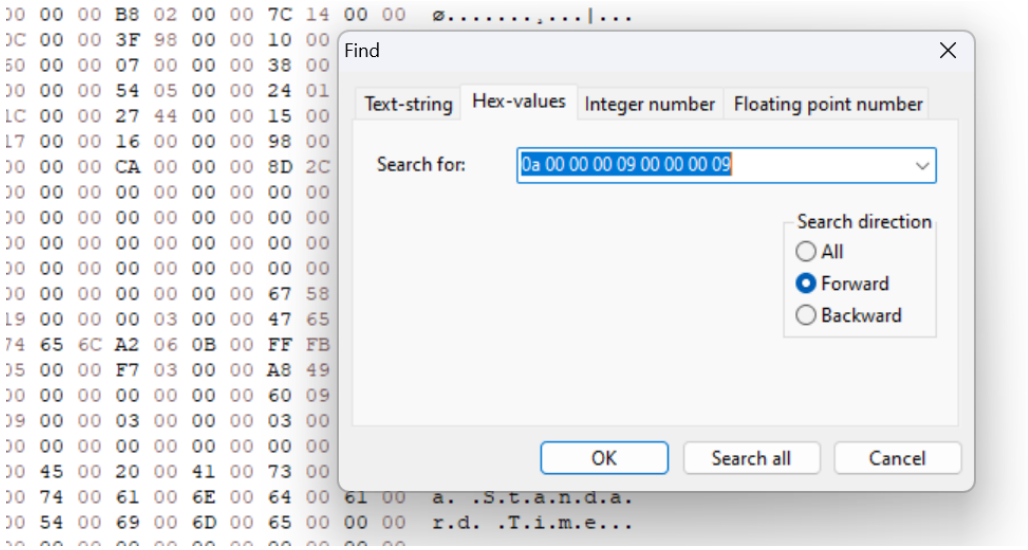

```
PS C:\Users\Lenovo legion 5\Downloads\minesam.exe> procdump -ma minesam.exe mine

ProcDump v11.0 - Sysinternals process dump utility
Copyright (C) 2009-2022 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[13:32:06] Dump 1 initiated: C:\Users\Lenovo legion 5\Downloads\minesam.exe\mine.dmp
[13:32:06] Dump 1 writing: Estimated dump file size is 61 MB.
[13:32:06] Dump 1 complete: 61 MB written in 0.2 seconds
[13:32:06] Dump count reached.

PS C:\Users\Lenovo legion 5\Downloads\minesam.exe>
```

- **Viewing the Memory with HxD**

o From the HxD menu bar, click Search, Find.

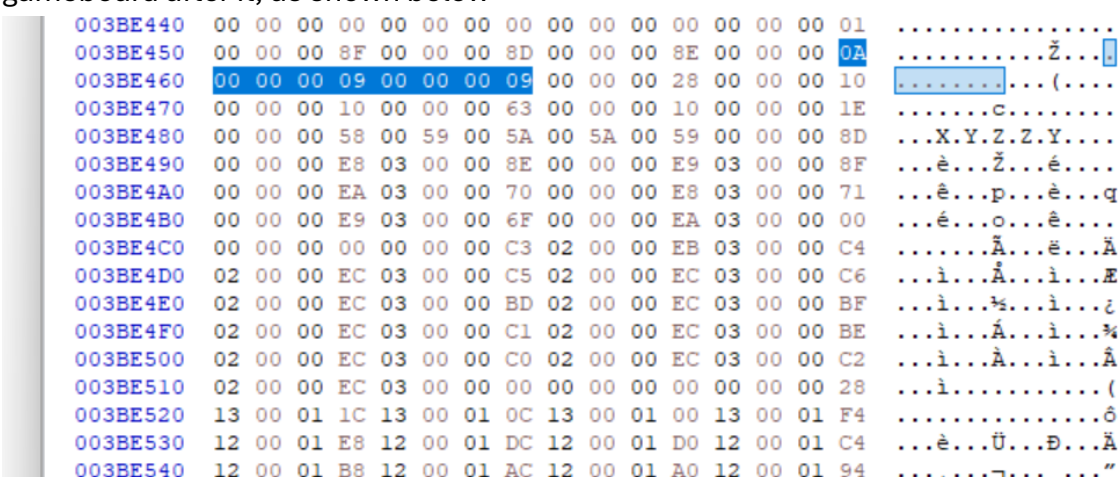
o In the "Find" field, select a Datatype of Hex-values.

o In the "Search for" field, enter this text, as shown below **0a 00 00 00 09 00 00 00 09**



o In the "Find" box, click OK.

o The string is found, but it may not be the correct hit. The first one doesn't have the gameboard after it, as shown below



o From the HxD menu bar, click Search, "Find again".

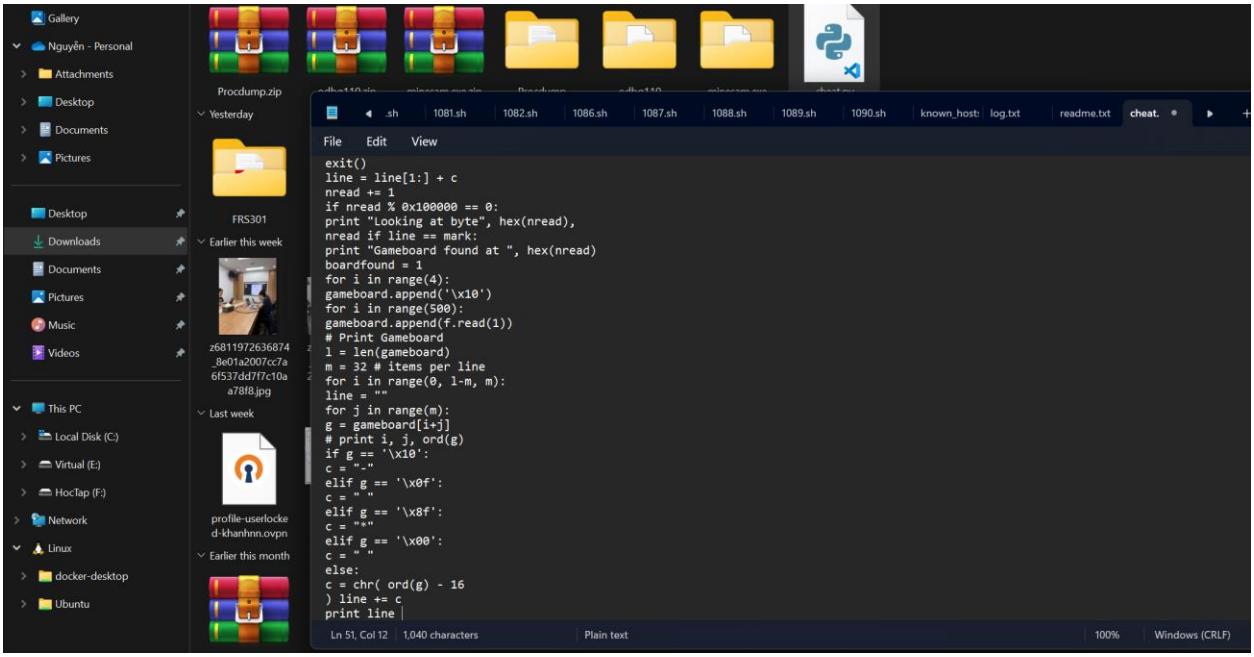
o This time it finds the gameboard data, as shown below

```
003BE740  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003BE750  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003BE760  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003BE770  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A  ................
003BE780  00 00 00 09 00 00 00 09 00 00 00 00 00 00 00 10  ................
003BE790  10 10 10 10 10 10 10 10 10 10 0F 0F 0F 0F 0F 0F  ................
003BE7A0  0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 10  ................
003BE7B0  8F 0F 0F 0F 0F 0F 0F 0F 8F 10 0F 0F 0F 0F 0F 0F  ................
003BE7C0  0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 10  ................
003BE7D0  0F 0F 0F 0F 8F 8F 0F 0F 0F 10 0F 0F 0F 0F 0F 0F  ................
003BE7E0  0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 10  ................
003BE7F0  0F 0F 8F 0F 0F 0F 8F 0F 0F 10 0F 0F 0F 0F 0F 0F  ................
003BE800  0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 10  ................
003BE810  0F 0F 0F 0F 0F 0F 0F 0F 8F 10 0F 0F 0F 0F 0F 0F  ................
003BE820  0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 10  ................
003BE830  0F 0F 0F 0F 0F 0F 0F 0F 0F 10 0F 0F 0F 0F 0F 0F  ................
003BE840  0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 10  ................
003BE850  0F 8F 0F 0F 0F 0F 0F 0F 8F 10 0F 0F 0F 0F 0F 0F  ................
```

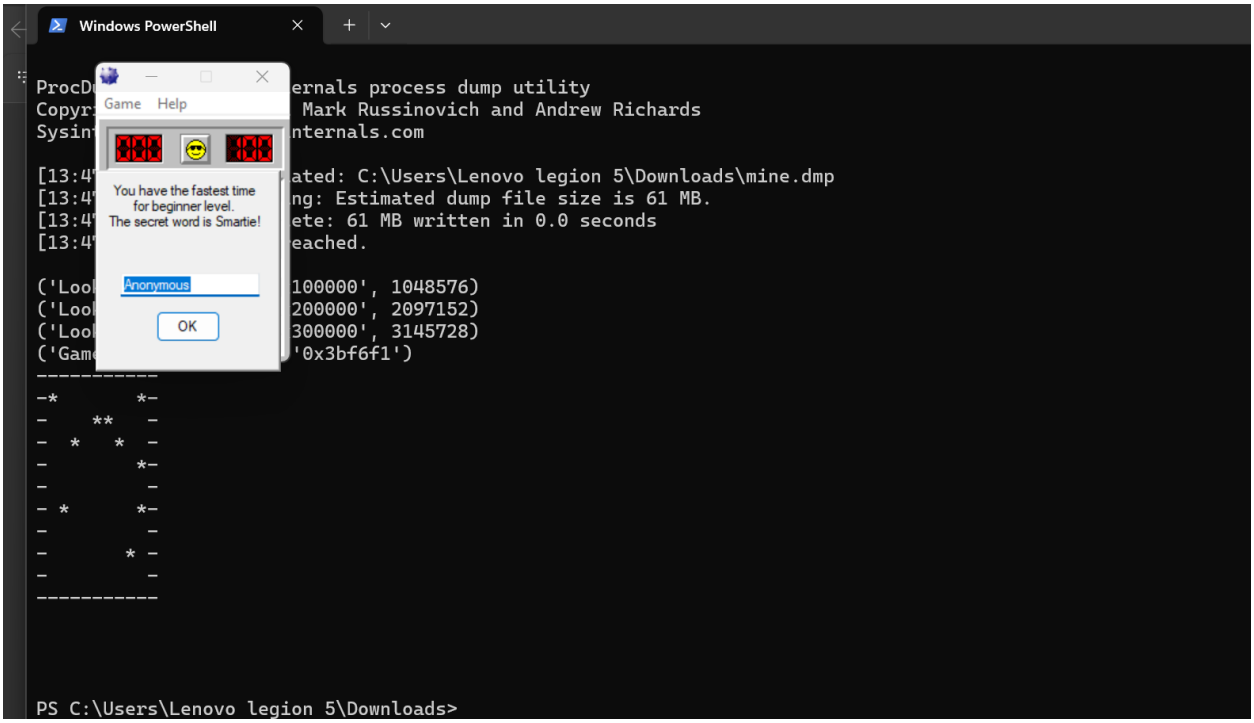
- **Creating a Python Script**

o In the Command Prompt window, execute this command: python cheat.py

o The program shows the location of the mines. With this information, you should easily be able to click all the squares without mines, as shown below.



o When you win the game, a secret word will appear, which is covered by a green box in the image below.



• **Intermediate Level**

o In Minesweeper, click Game, Intermediate.

o Create a cheating tool that works for this level and win the game, as shown below.

o Hint: Search for 10 10 10 10 to find the gameboard

o Open a Command Prompt and execute these commands: procdump -ma minesam.exe mine

o Replace “\x28\x00\x00\x00\x10\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x10\x10\x10\x10” in

mark on script

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

cheat.py

C: > Users > Lenovo legion 5 > Downloads > cheat.py

```python
import os

# Dump memory
os.system("del mine.dmp")
os.system("procdump -ma minesam.exe mine")

# Find gameboard
mark = '\x28\x00\x00\x00\x10\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x10\x10\x10\x10'
nread = 20
boardfound = 0
gameboard = []

with open("mine.dmp", "rb") as f:
    line = f.read(20)
    while boardfound == 0:
        c = f.read(1)
        if c == b"":  # Python 3: bytes so prefix with b
            print("File ended, but gameboard not found!")
            exit()
        line = line[1:] + c
        nread += 1
```

o Run the script

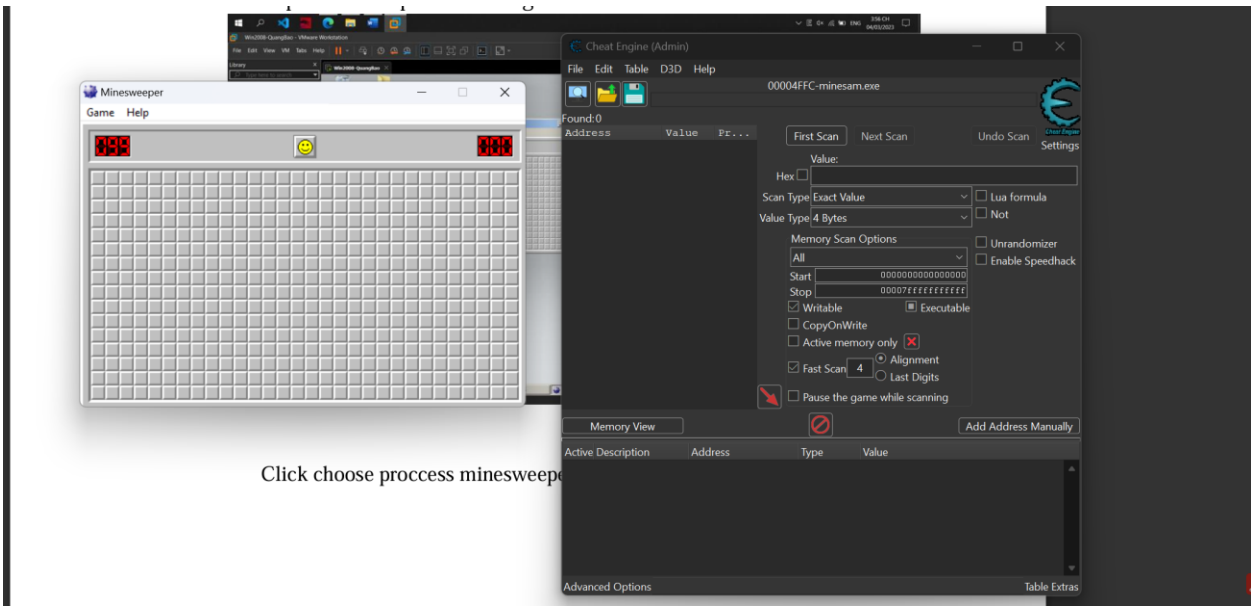
```
PS C:\Users\Lenovo legion 5\Downloads> python .\cheat.py

ProcDump v11.0 - Sysinternals process dump utility
Copyright (C) 2009-2022 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[15:09:28] Dump 1 initiated: C:\Users\Lenovo legion 5\Downloads\mine.dmp
[15:09:29] Dump 1 writing: Estimated dump file size is 61 MB.
[15:09:29] Dump 1 complete: 61 MB written in 0.1 seconds
[15:09:29] Dump count reached.

('Looking at byte', '0x100000', 1048576)
('Looking at byte', '0x200000', 2097152)
('Looking at byte', '0x300000', 3145728)
('Gameboard found at', '0x3c29f8')
------------------
-          *   **-
-             *  -
-***   *   *    *-
-   *           *-
-                -
-      **        -
-    * *      *  -
-                -
-* **         *  -
-  *          ** -
-     *  **   *  -
-    *   *       -
-         *  *   -
-      * *      *-
PS C:\Users\Lenovo legion 5\Downloads>
```

Expert Level

o Using Cheat Engine\

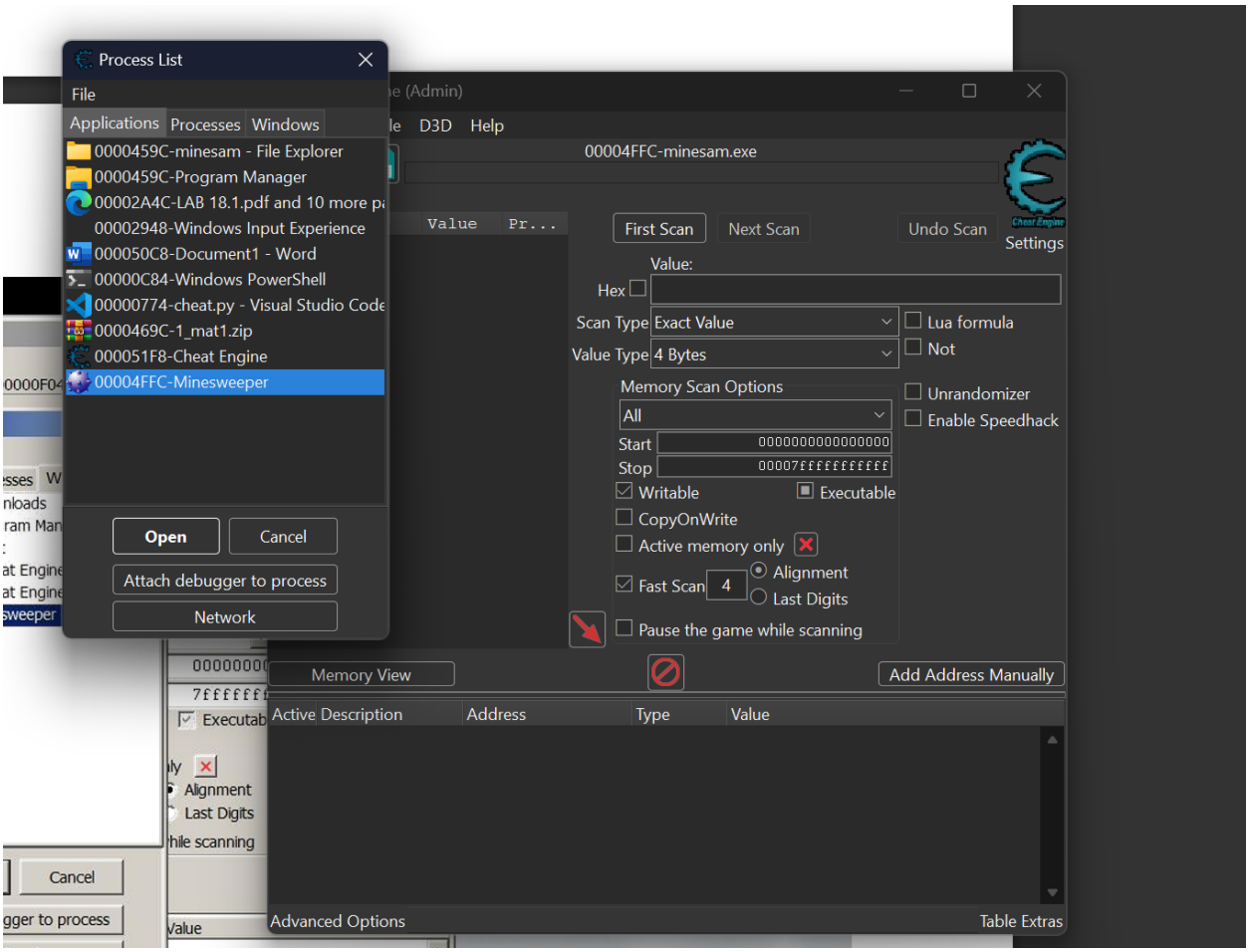
o Change the number of boom in minesweeper

First open minesweeper in Expert Level Open Cheat Engine

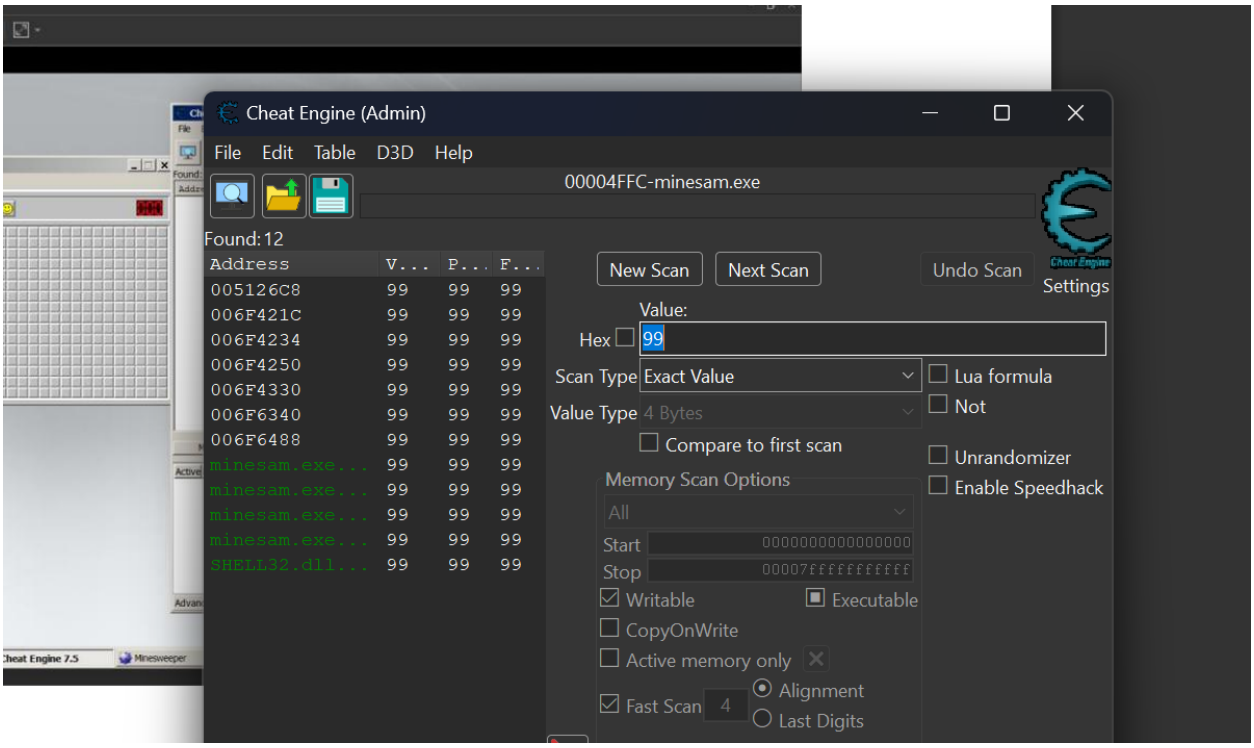


NGUYEN NAM KHANH – HE191159 – IA1902 – IAM302

Click choose proccess minesweeper



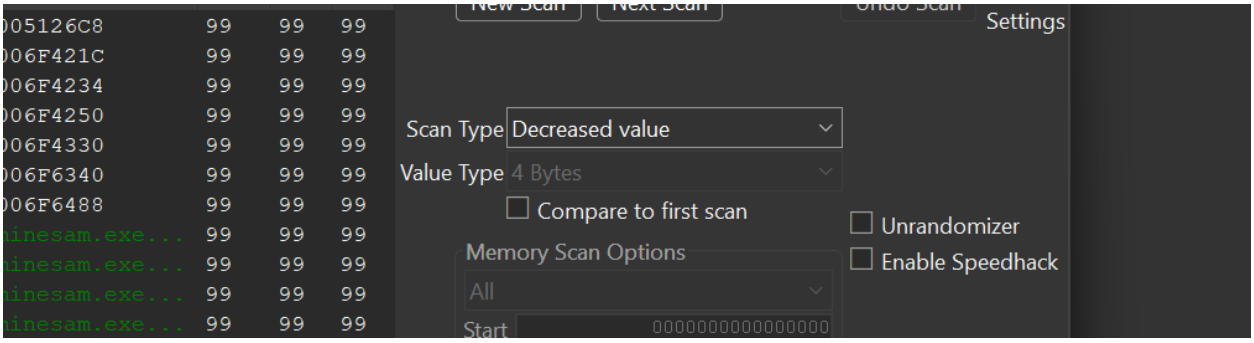
Type 99 values and press Enter - 99 is the number of the booms



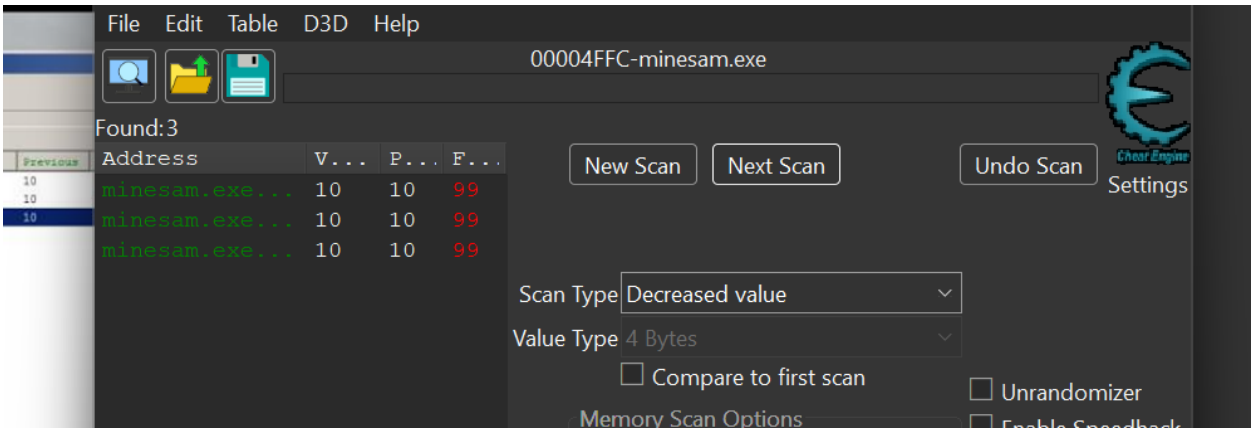
Click first scan

After first scan, click to Decreased Value in Scan type

Change game mode into Beginer



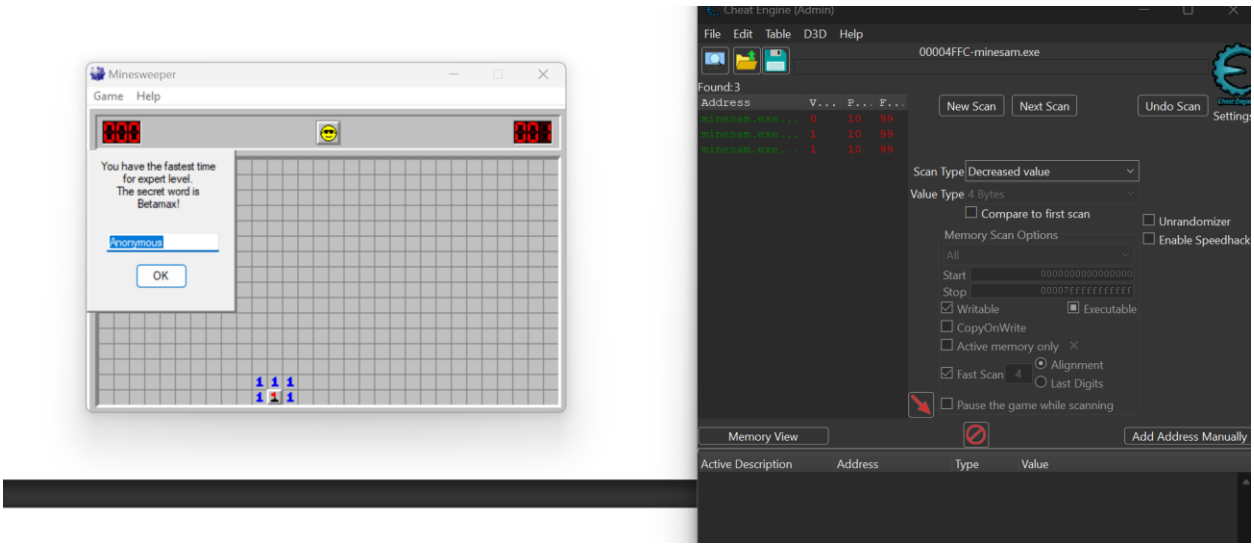
Click next scan



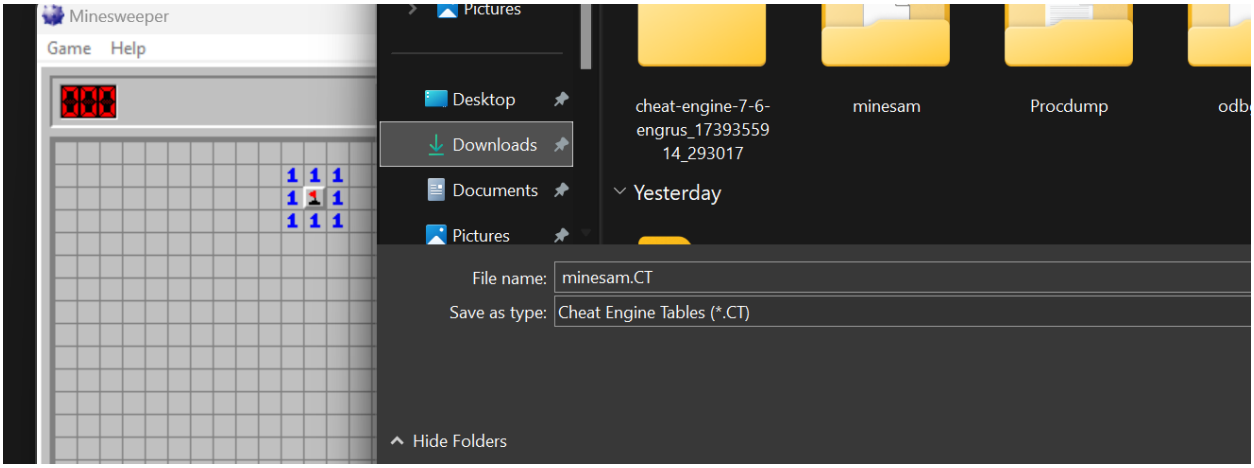
Choose all results and hit “Ctrl + E” to edit value

Change all to “1”, Click Enter

Now change the game mode to Expert level and click random box



You won the game! Right Click and chose Add to new group Change name to Minesweeper And pess “Ctrl + s” to save file.

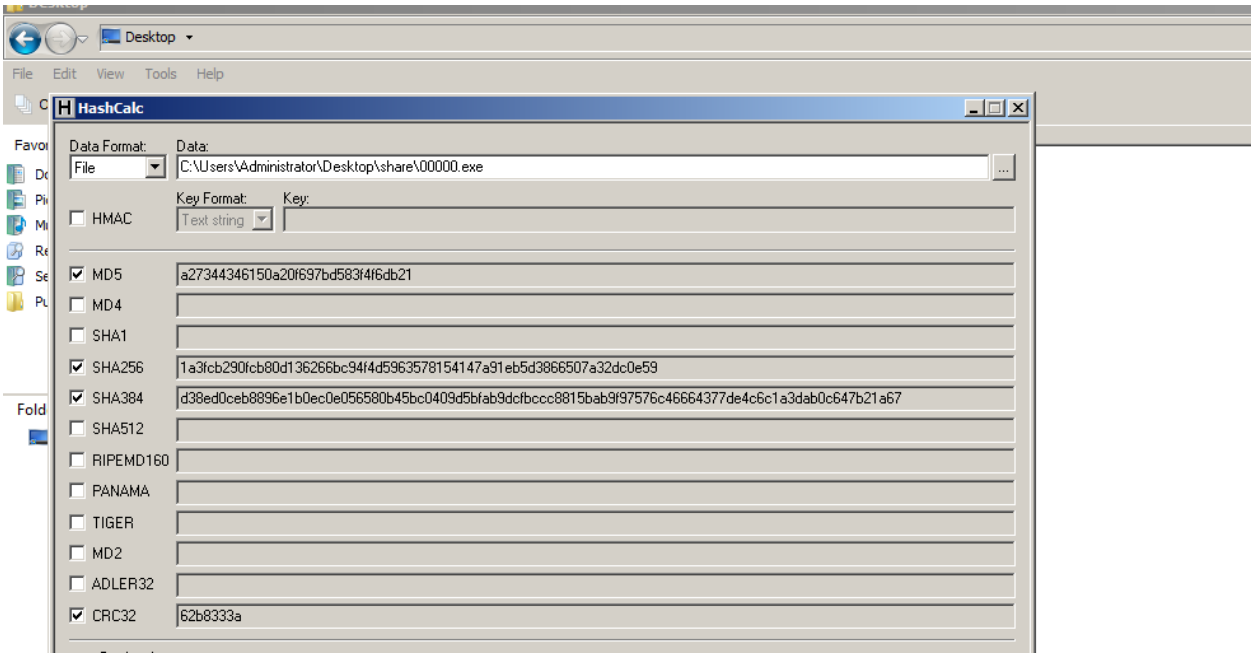


Use can use this file whenever you want.

Lab 18.2: Patching EXEs with Ollydbg

18.2.1: Patching an EXE

• Checking the Hash:

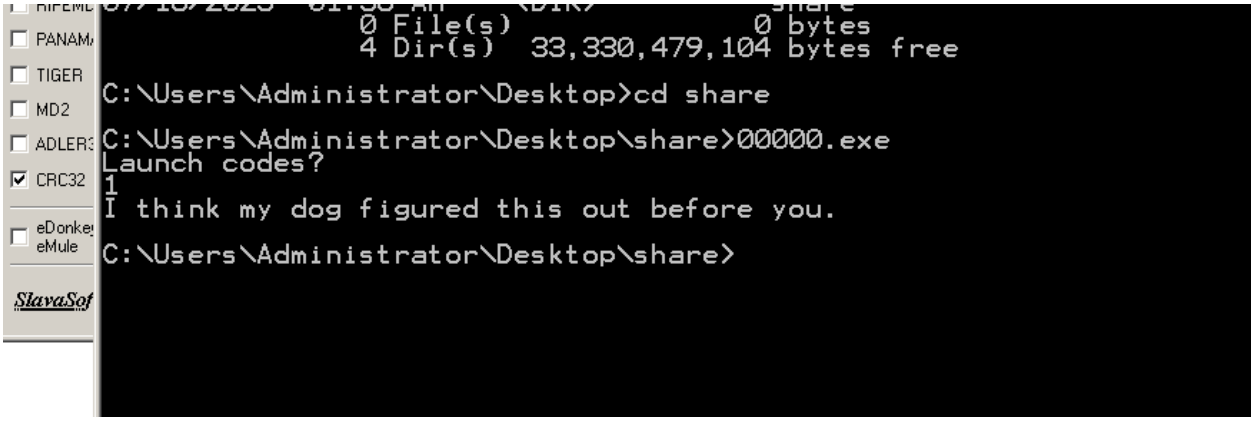


• Running the EXE:

Execute these commands: cd \users\administrator\Desktop

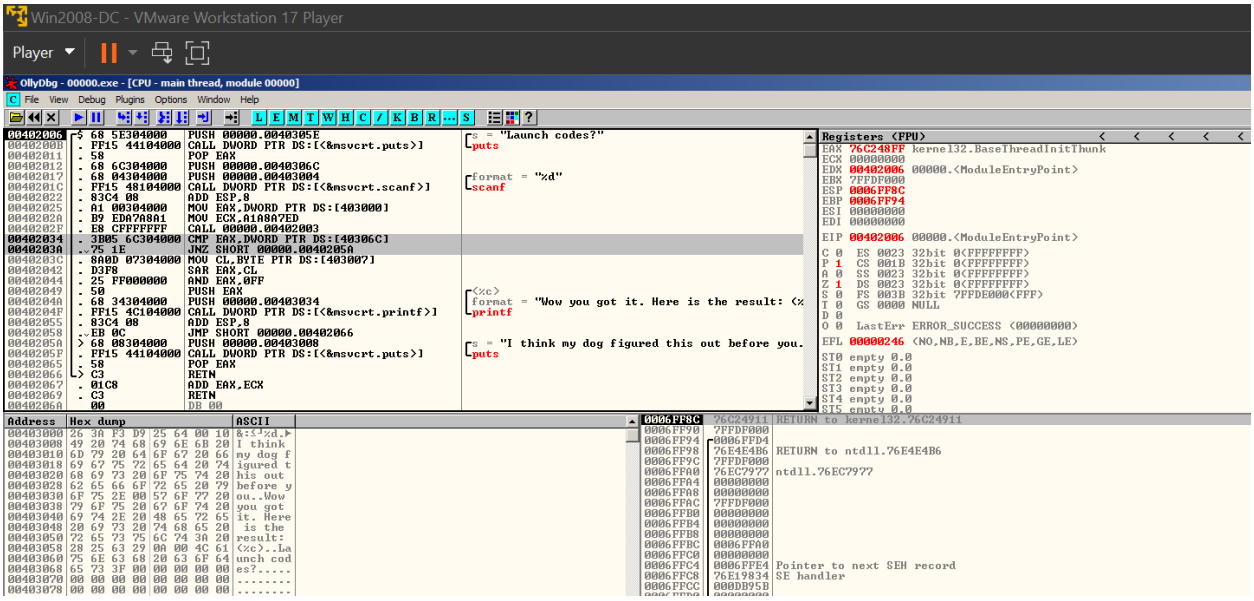
00000.exe

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



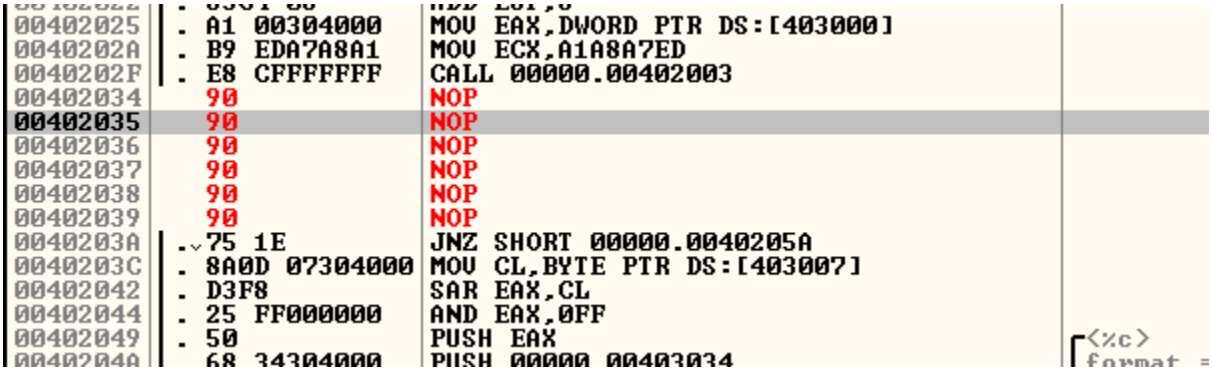
NGUYEN NAM KHANH – HE191159 – IA1902 – IAM302

• Examining the EXE with Ollydbg: The choice is performed by two instructions: CMP (Compare) and JNZ (Jump if Not Zero), outlined in green in the image below

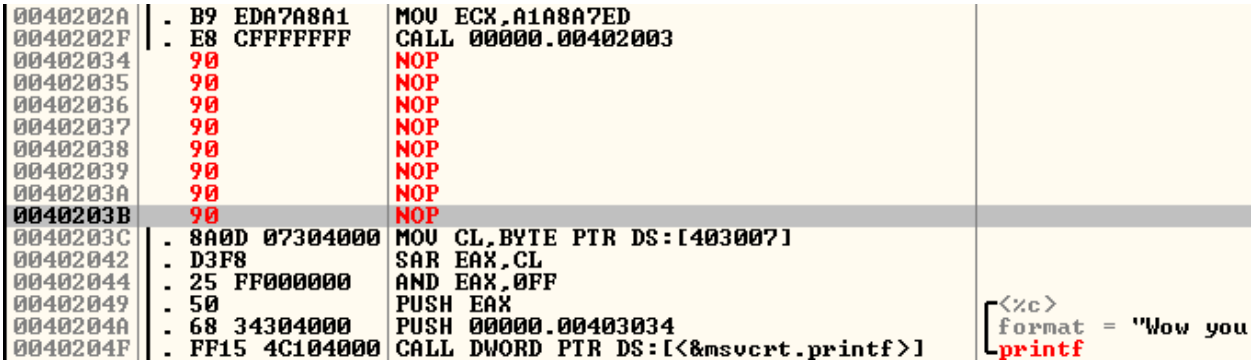


• Modifying the EXE:

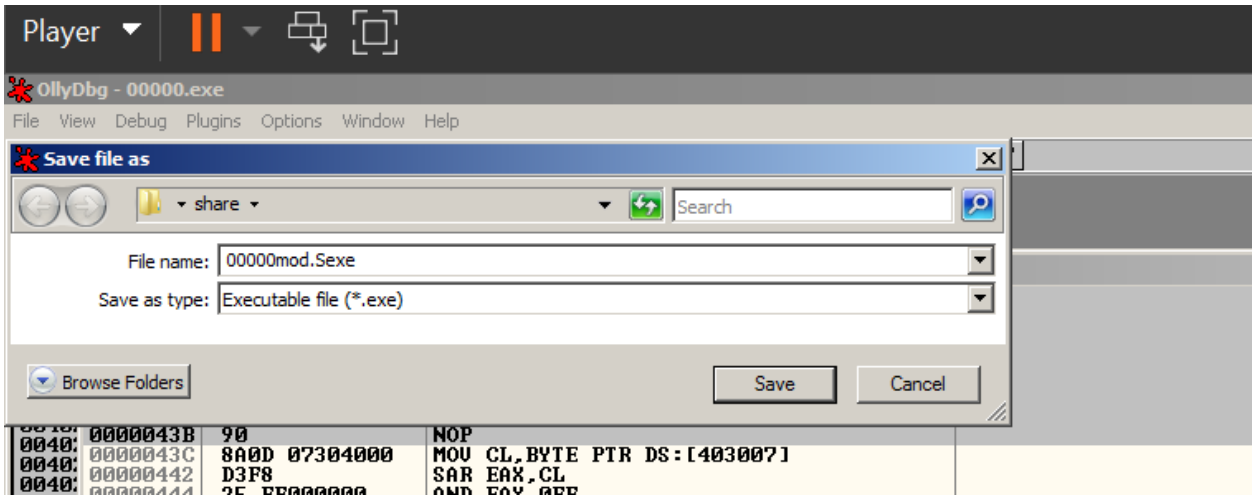
Right-click the CMP instruction and click Assemble, as shown below.



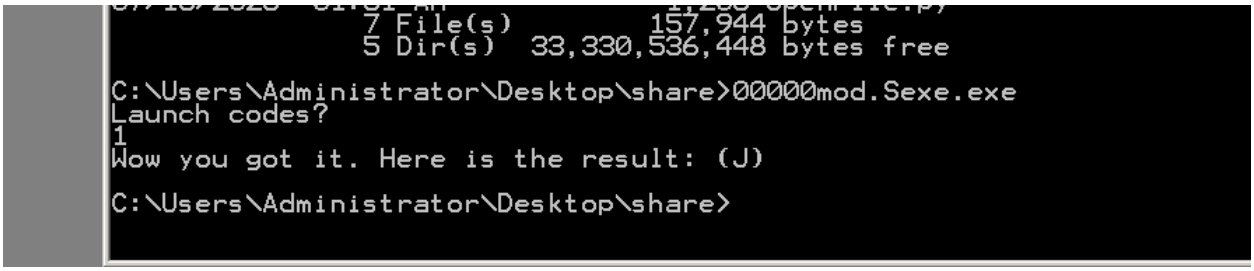
Repeat the process to replace the JNZ instruction with NOPs also, as shown below



- Saving the Modified File:



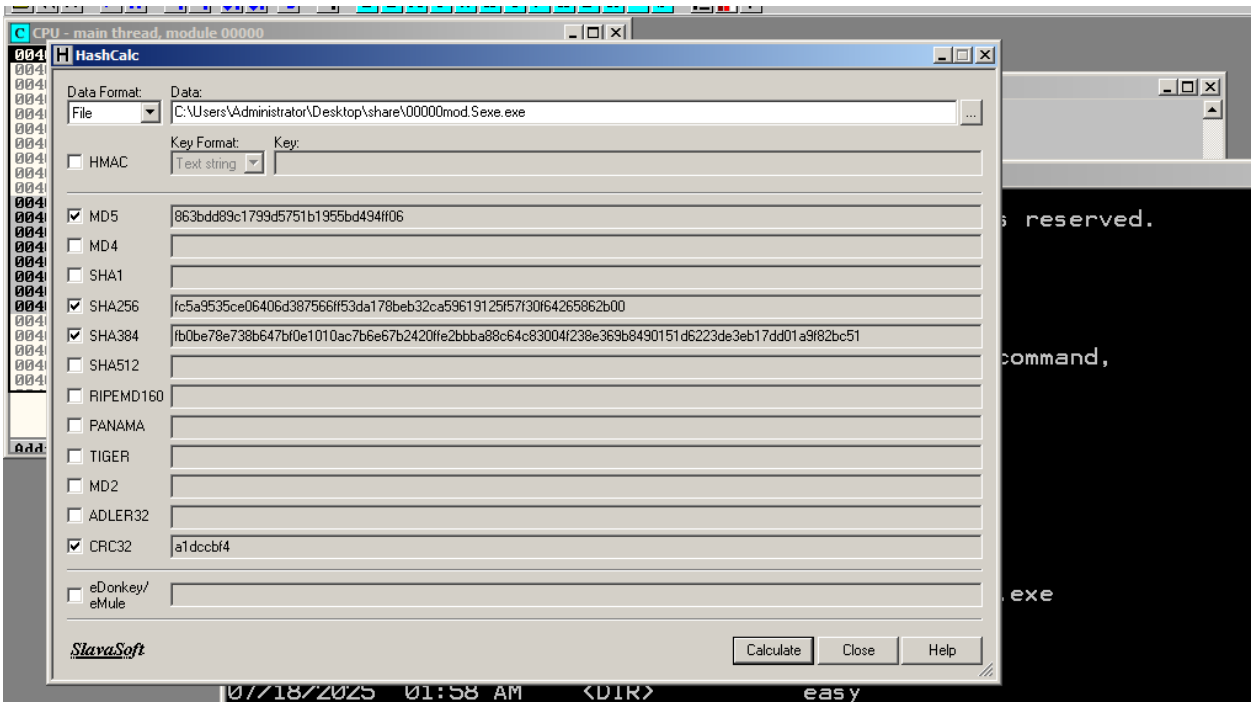
• Running the Modified File:



• Checking the Hash:

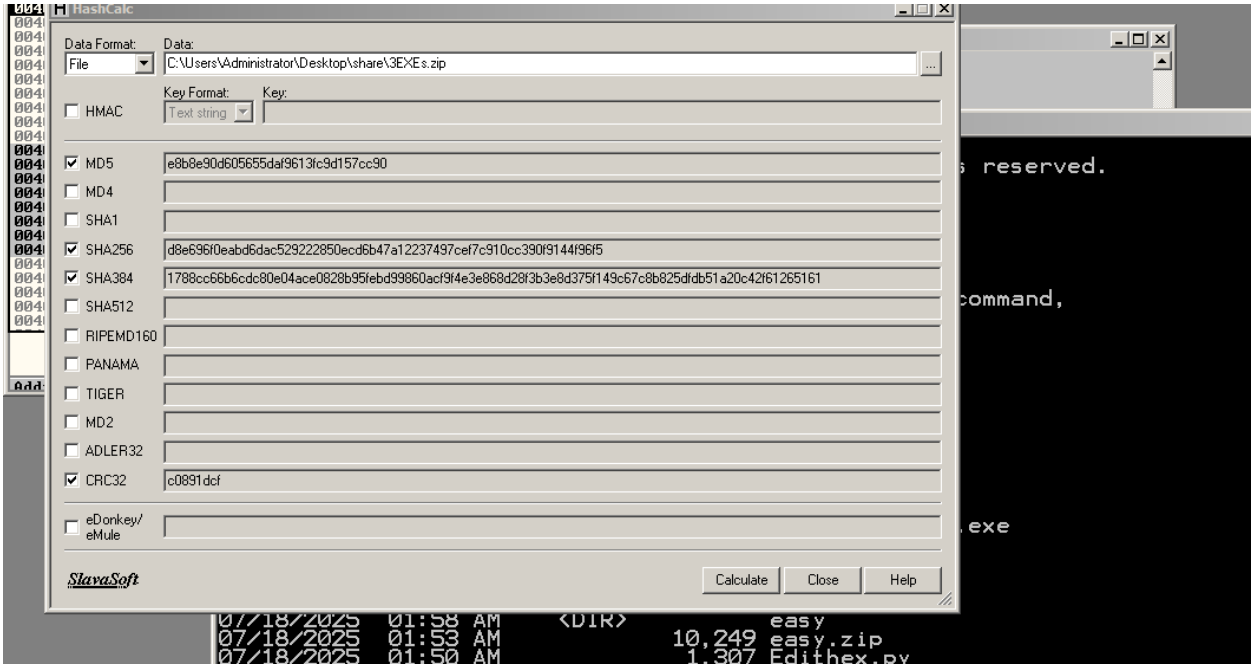
o Calculate the SHA256 hash of the patched file. It should match the value shown below.

o 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:

• Checking the Hash:



• Patch the Files:

o Patch all 3 files so they will accept any input

• Gather the Results:

o 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

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

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

C:\Users\Administrator>cd Desktop

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

00000mod.exe: Launch codes?
Wow you got it. Here is the result: (J)

0000a.exe: #?
A: (:)

000a1.exe: Enter a number:
([), duh.

C:\Users\Administrator\Desktop>
```

o Now we run python script in cmd:

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

C:\Users\Administrator>cd Desktop

C:\Users\Administrator\Desktop>python Edithex.py

C:\Users\Administrator\Desktop>
```

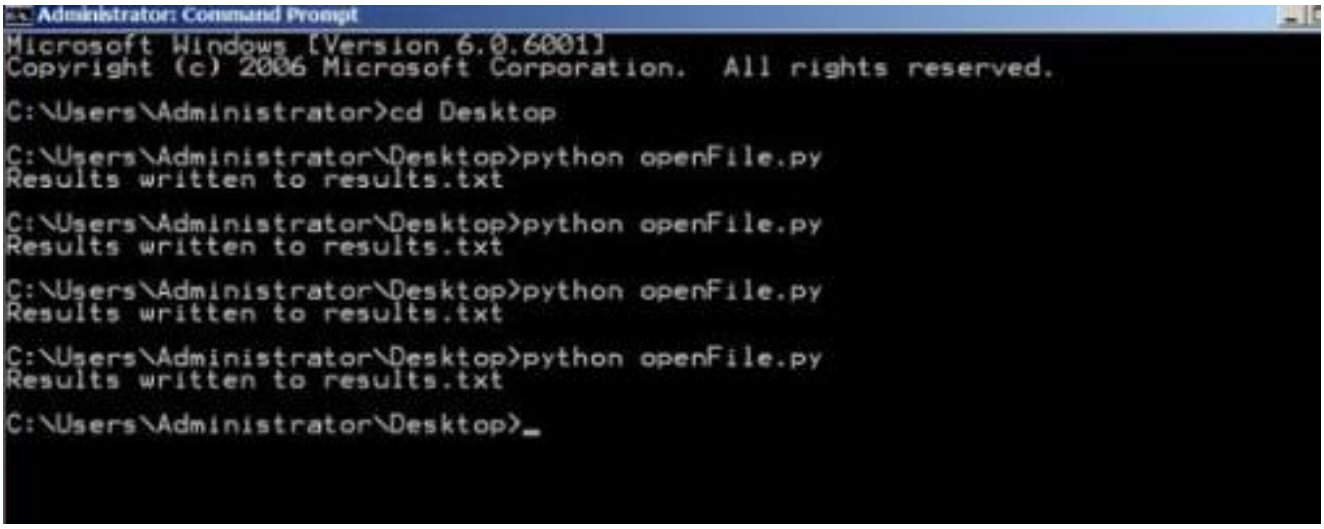
o Check file with Ollydbg

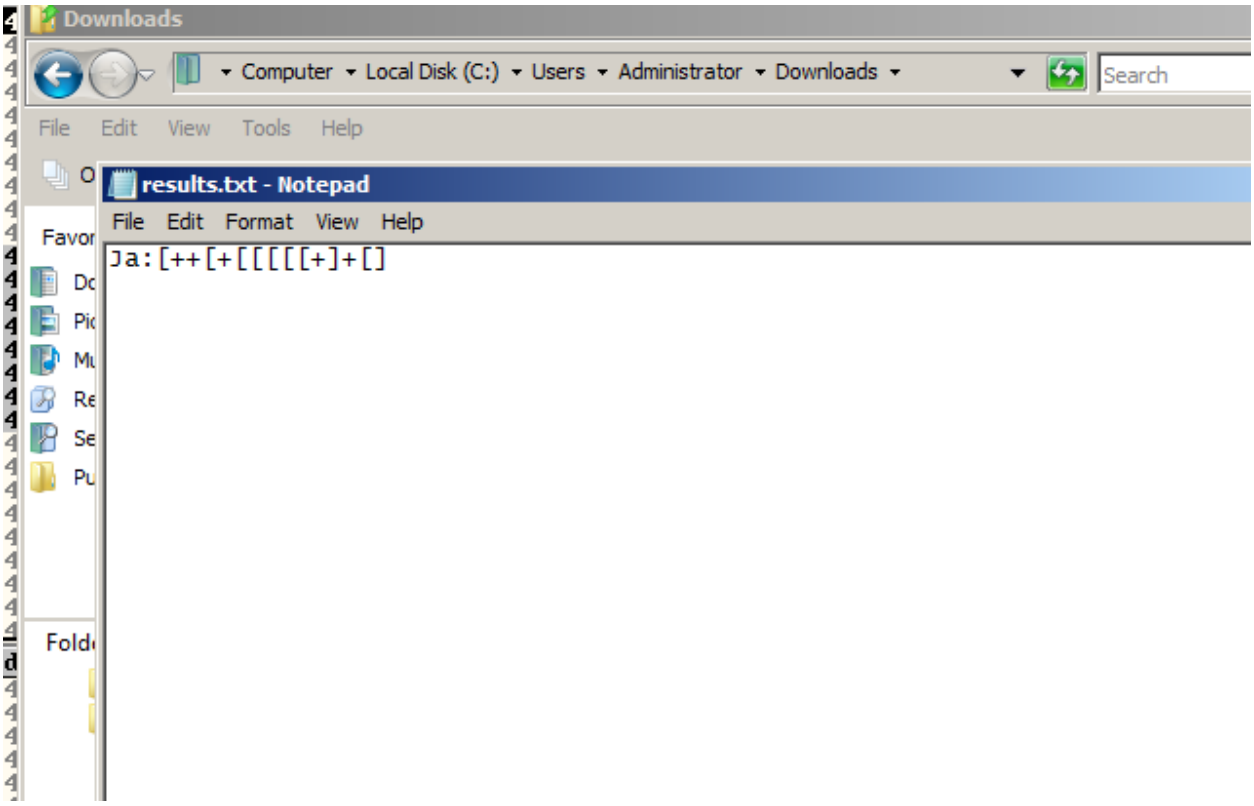
```
00402006  $ 68 5E304000   PUSH 00000mod.0040305E                    s = "Launch codes?"
0040200B  . FF15 44104000 CALL DWORD PTR DS:[<&msvcrt.puts>]        puts
00402011  . 58            POP EAX
00402012  . 68 6C304000   PUSH 00000mod.0040306C
00402017  . 68 04304000   PUSH 00000mod.00403004                     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 00000mod.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 00000mod.00403034                    format = "Wow you got it. Here is the resul
0040204F  . FF15 4C104000 CALL DWORD PTR DS:[<&msvcrt.printf>]      printf
```

o Now we use another script to run mutilple file and store it in results.txt

```
C:\Users\Administrator>cd Desktop

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

0000a.exe: #?
A: (:)

000a1.exe: Enter a number:
([), duh.

C:\Users\Administrator\Desktop>_
```

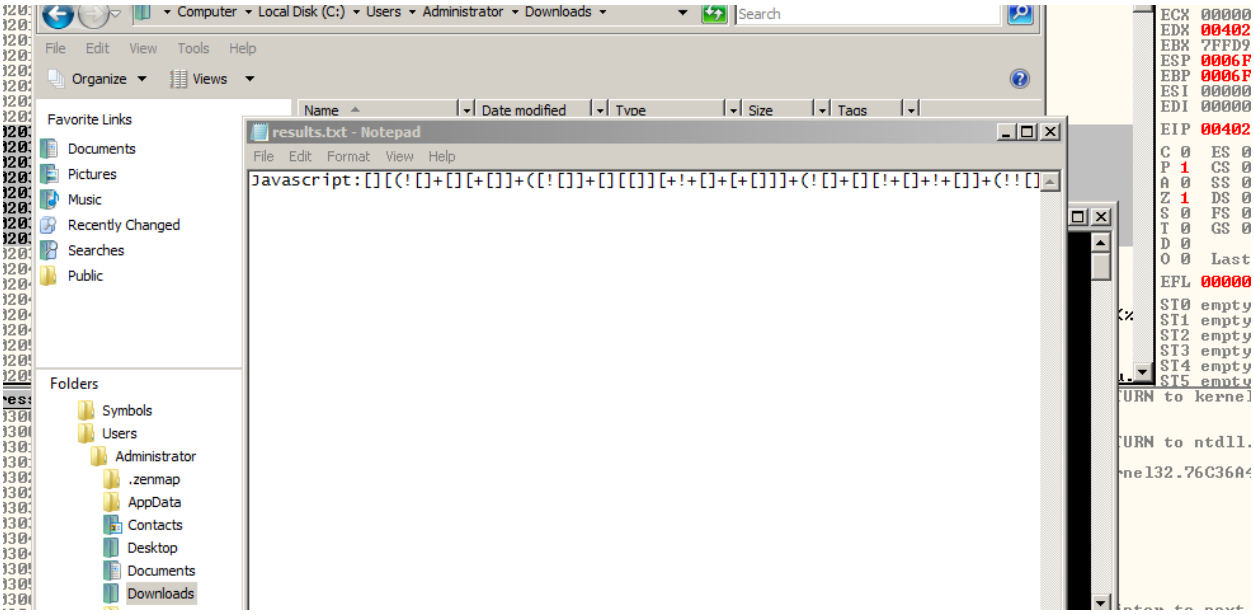
18.2.3: Patching 19 EXEs:



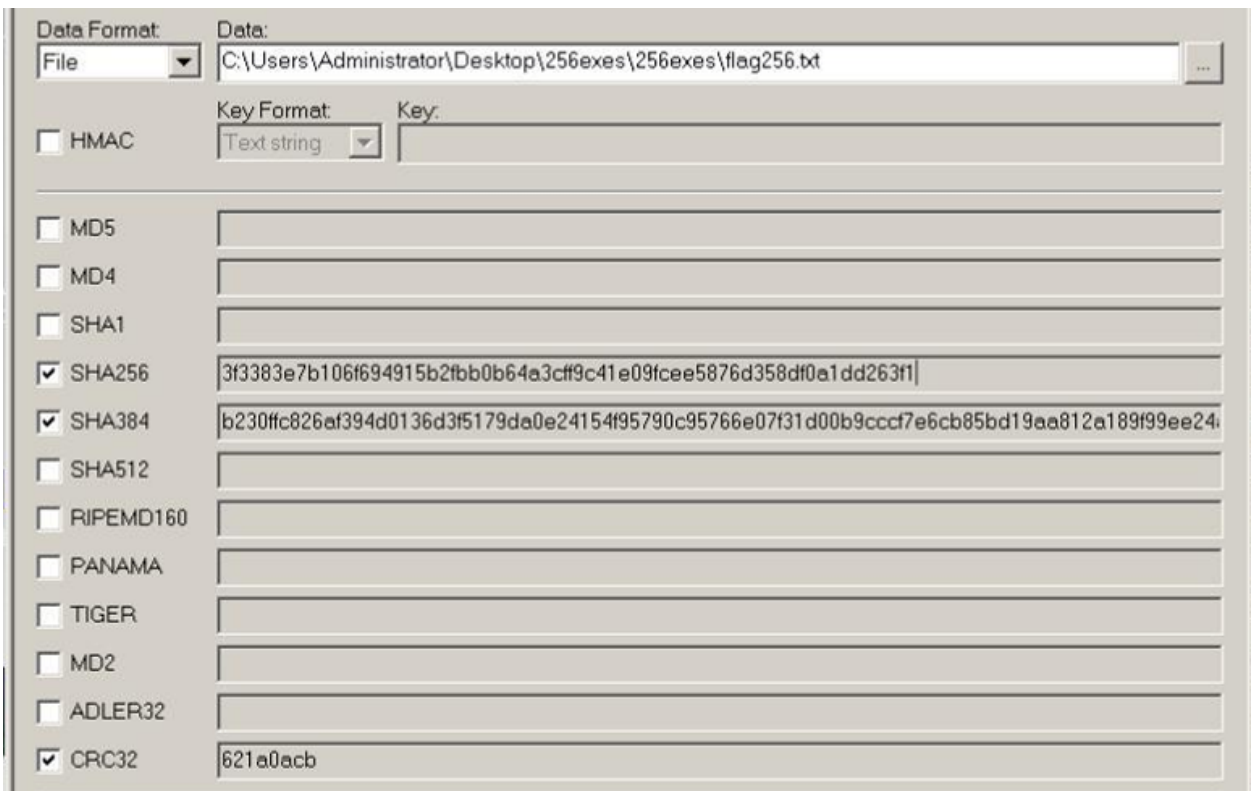


18.2.4: Patching 256 EXEs:

o Now we got the string as below

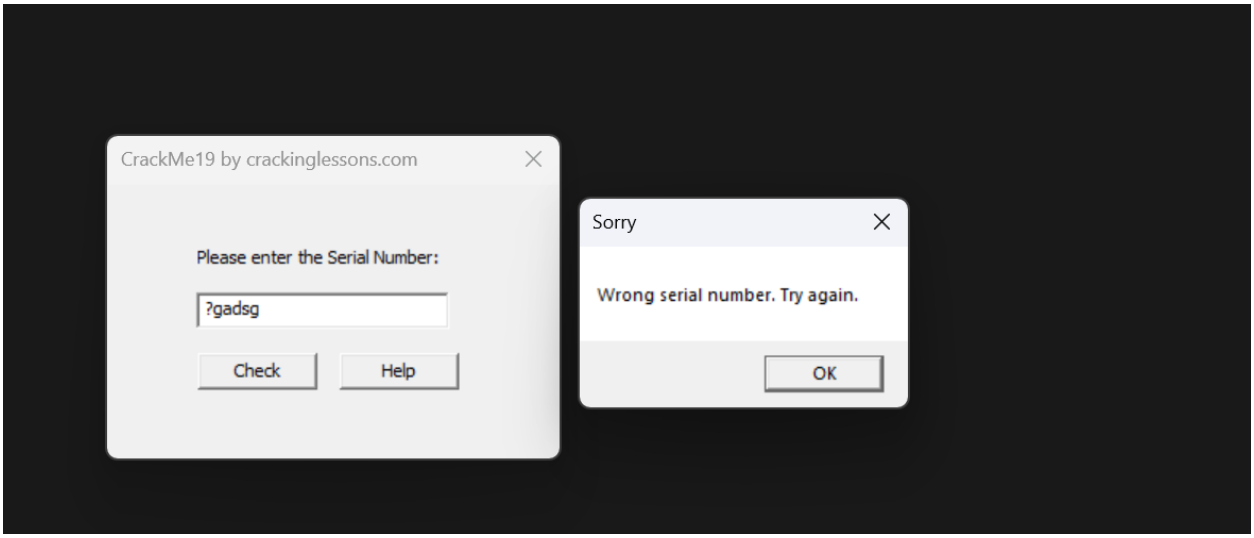


o Calculate the SHA256 hash of that file.



CRACKME 19

This crackme comes in 2 files. The crackme19.exe is the main file and there is also a DLL called CrackmeLibrary.dll. The objective of this crackme is to practise patching the DLL instead of the crackme19.exe file.



Becasue this challenge provides .dll file, first we need to find string references their code of success part and fail part, and notice this JE command. Above these command is the TEST command for AL variable. If AL equals 0, the JE will be executed. So we need to change the value of AL instead of 0 value.

```
00401105   E8 06250000      call crackme19.403610
0040110A   50               push eax
0040110B   FF15 00D04000    call dword ptr ds:[<checkSerialNumber>]
00401111   83C4 08          add esp,8
00401114   6A 00            push 0
00401116   84C0             test al,al
00401118   74 1A            je crackme19.401134
0040111A   68 E81A4100      push crackme19.411AE8      411AE8:"Congrats!"
0040111F   68 F41A4100      push crackme19.411AF4      411AF4:"Well done!"
00401124   6A 00            push 0
00401126   FF15 1CD14000    call dword ptr ds:[<MessageBoxA>]
0040112C   33C0             xor eax,eax
0040112E   8BE5             mov esp,ebp
00401130   5D               pop ebp
00401131   C2 1000          ret 10
00401134   68 001B4100      push crackme19.411B00      411B00:"Sorry"
00401139   68 081B4100      push crackme19.411B08      411B08:"Wrong serial number. Try again."
0040113E   6A 00            push 0
00401140   FF15 1CD14000    call dword ptr ds:[<MessageBoxA>]
00401146   33C0             xor eax,eax
00401148   8BE5             mov esp,ebp
0040114A   5D               pop ebp
```

Notice the function checkSerialNumber, this function is block assigning AL with 0 value.

```
00401104   50               push eax
00401105   E8 06250000      call crackme19.403610
0040110A   50               push eax
0040110B   FF15 00D04000    call dword ptr ds:[<checkSerialNumber>]
00401111   83C4 08          add esp,8
00401114   6A 00            push 0
00401116   84C0             test al,al
00401118   74 1A            je crackme19.401134
0040111A   68 E81A4100      push crackme19.411AE8
0040111F   68 F41A4100      push crackme19.411AF4
00401124   6A 00            push 0
00401126   FF15 1CD14000    call dword ptr ds:[<MessageBoxA>]
0040112C   33C0             xor eax,eax
0040112E   8BE5             mov esp,ebp
00401130   5D               pop ebp

10001000 <crackmelibrary.checkSerialNumber>
push ebp
mov ebp,esp
cmp dword ptr ss:[ebp+8],12BD64A
sete al
pop ebp
ret
```

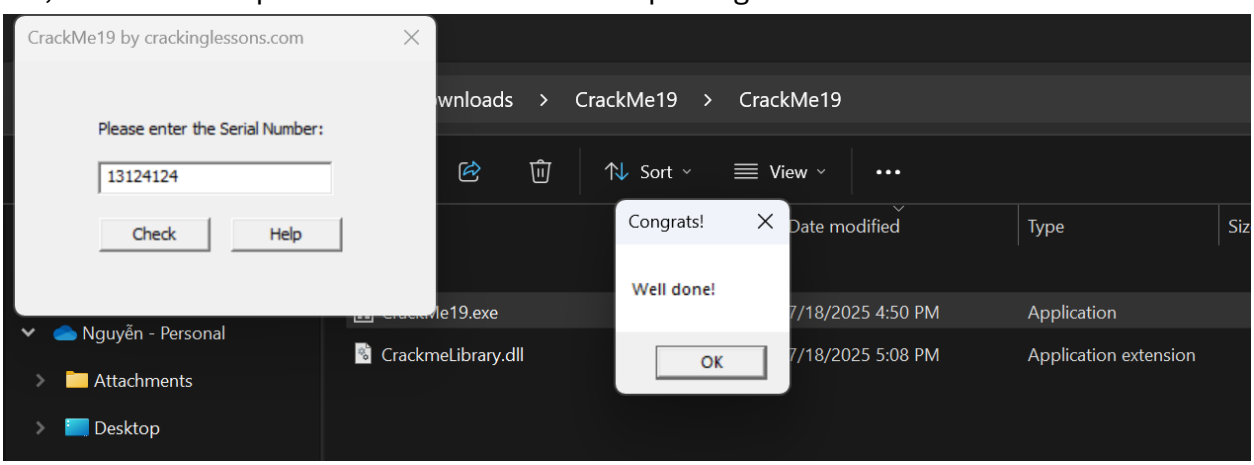
Inspect into this function, we found the code SETE AL, this assigning 0 into AL, so we need to fill NOP in this place.

```
10001001   8BEC          mov ebp,esp
10001003   817D 08 4AD62B01   cmp dword ptr ss:[ebp+8],12BD64A
1000100A   0F94C0        sete al
1000100D   5D            pop ebp
1000100E   C3            ret
1000100F   55            push ebp
10001010   8BEC          mov ebp,esp
10001012   8B45 0C       mov eax,dword ptr ss:[ebp+C]      [ebp+0C]:NlsAnsiCodePage+22E4
10001015   83E8 00       sub eax,0
10001018   74 33         je crackmelibrary.1000104D
```

Like below:

```
10001001   8BEC          mov ebp,esp
10001003   817D 08 4AD62B01   cmp dword ptr ss:[ebp+8],12BD64A
1000100A   90            nop
1000100B   90            nop
1000100C   90            nop
1000100D   5D            pop ebp
1000100E   C3            ret
1000100F   55            push ebp
10001010   8BEC          mov ebp,esp
10001012   8B45 0C       mov eax,dword ptr ss:[ebp+C]      [ebp+0C]:NlsAnsiCodePage+22E4
10001015   83E8 00       sub eax,0
10001018   74 33         je crackmelibrary.1000104D
1000101A   83E8 01       sub eax,1
1000101D   74 20         je crackmelibrary.1000103F
1000101F   83E8 01       sub eax,1
```

After that we patch the file, but notice that, function checkSerialNumber is executed in .dll file, so we need to patch this file into new .dll replacing the old one.



Done!!!