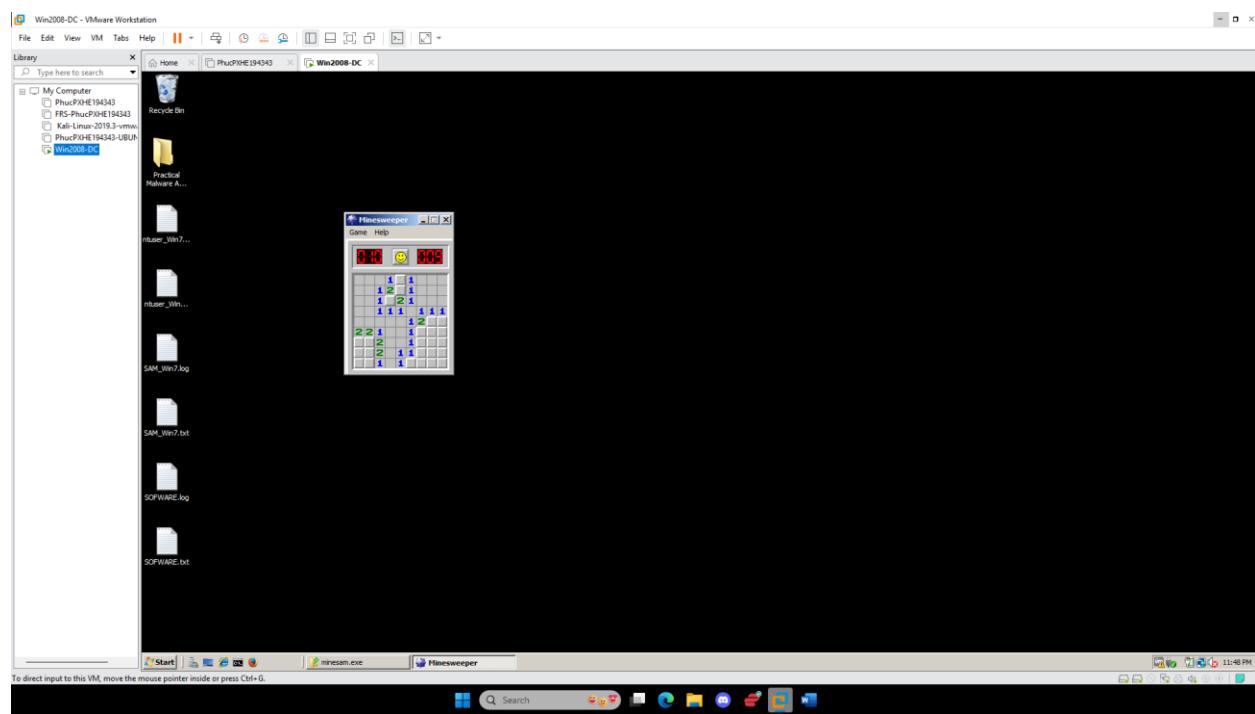
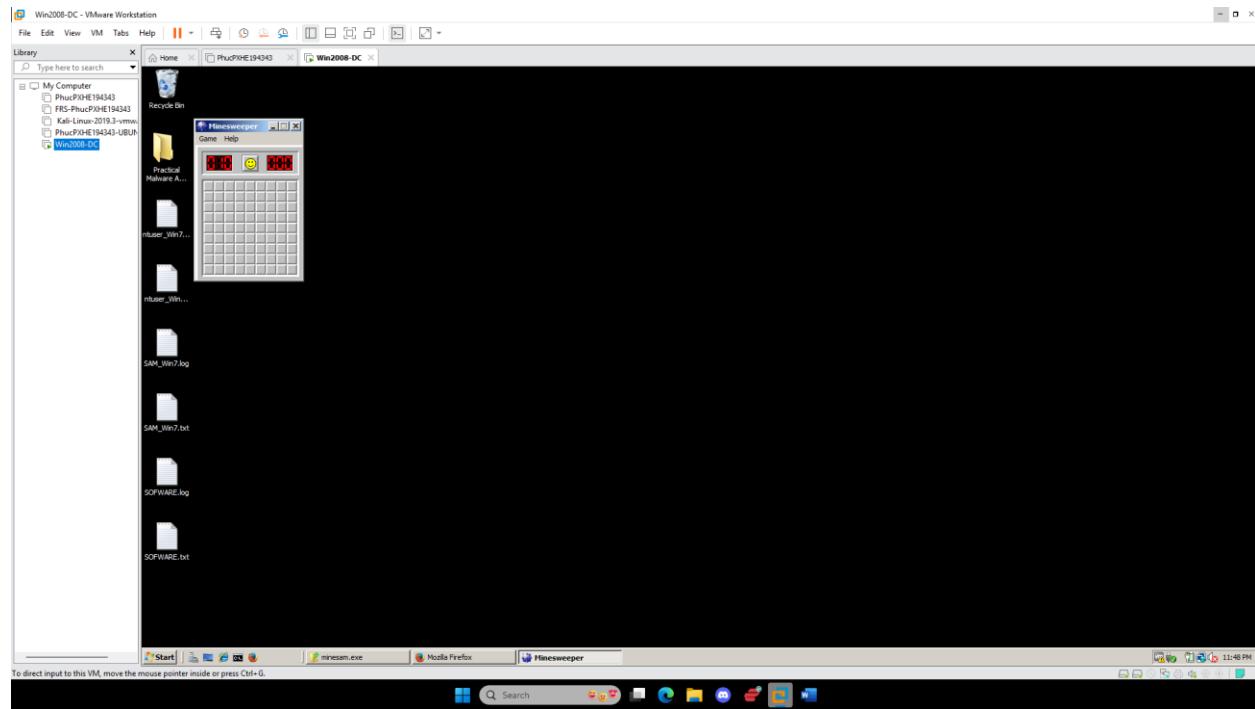


Lab 18.1



Win2008-DC - VMware Workstation

File Edit View VM Tabs Help ||

Library Type here to search

My Computer

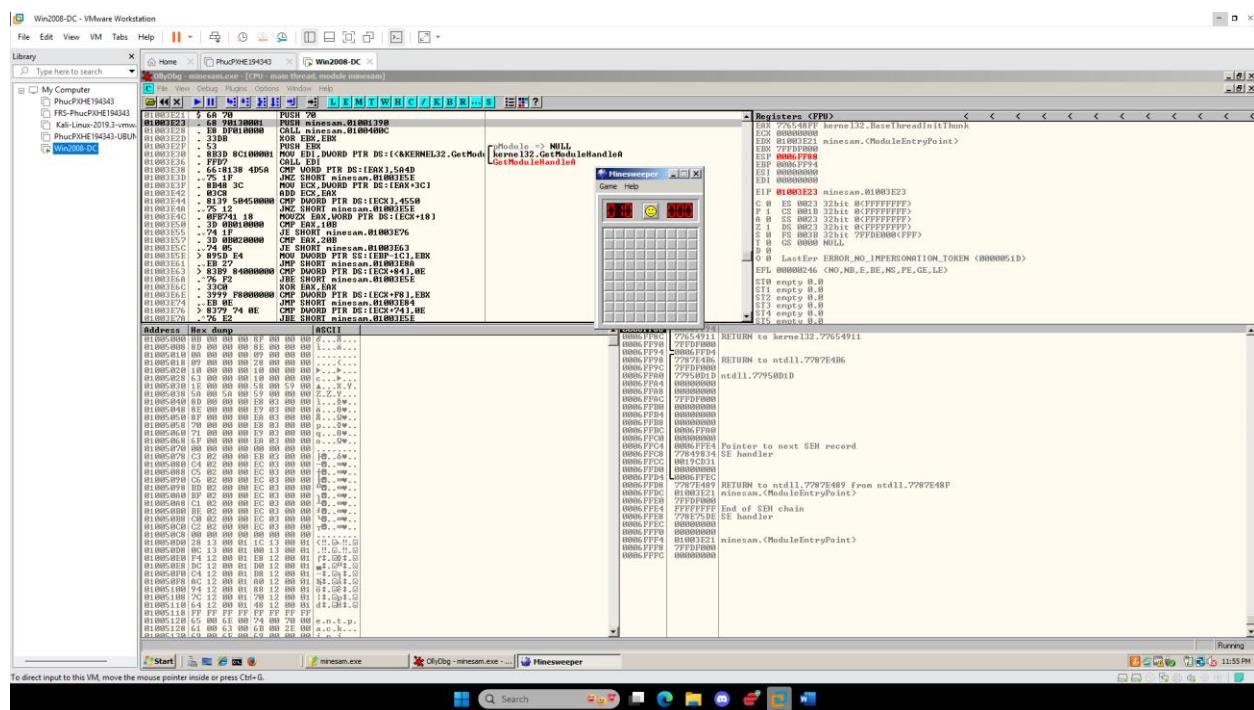
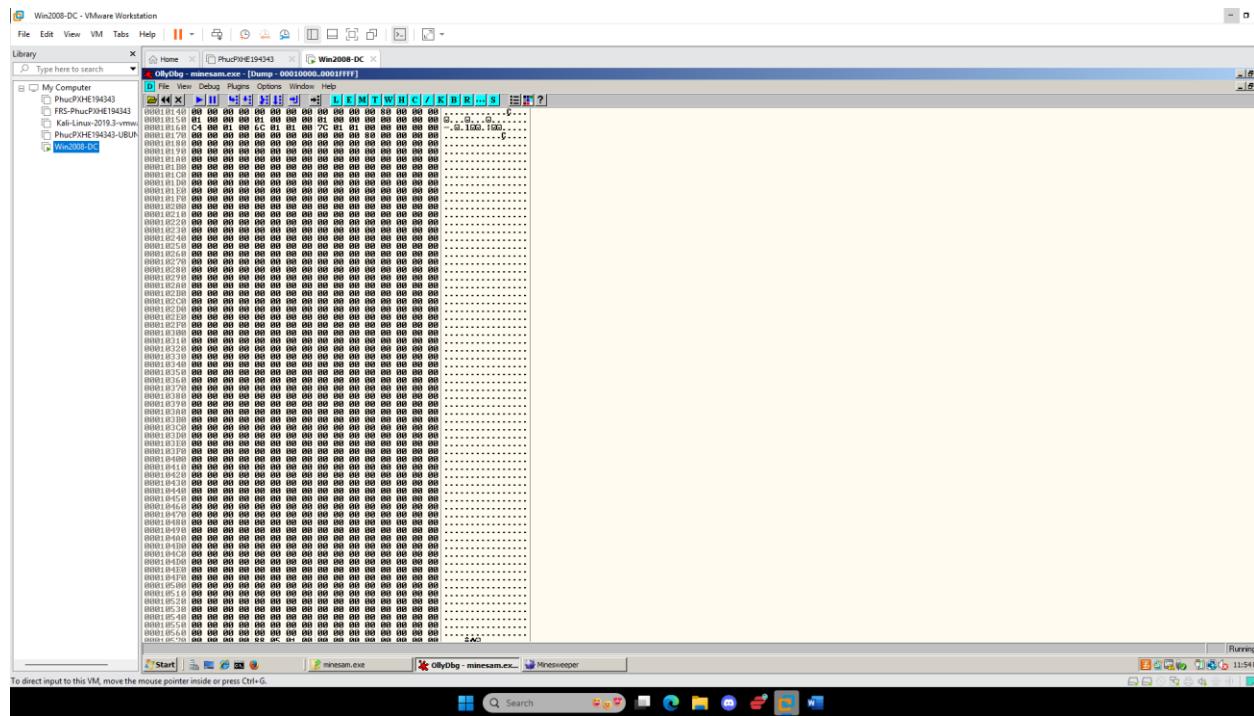
- PhucPXHE19434
- PhucPXHE19434
- Kali-Linux-2019.4-vmlinuz
- PhucPXHE19434-UBU
- Win2008-DC

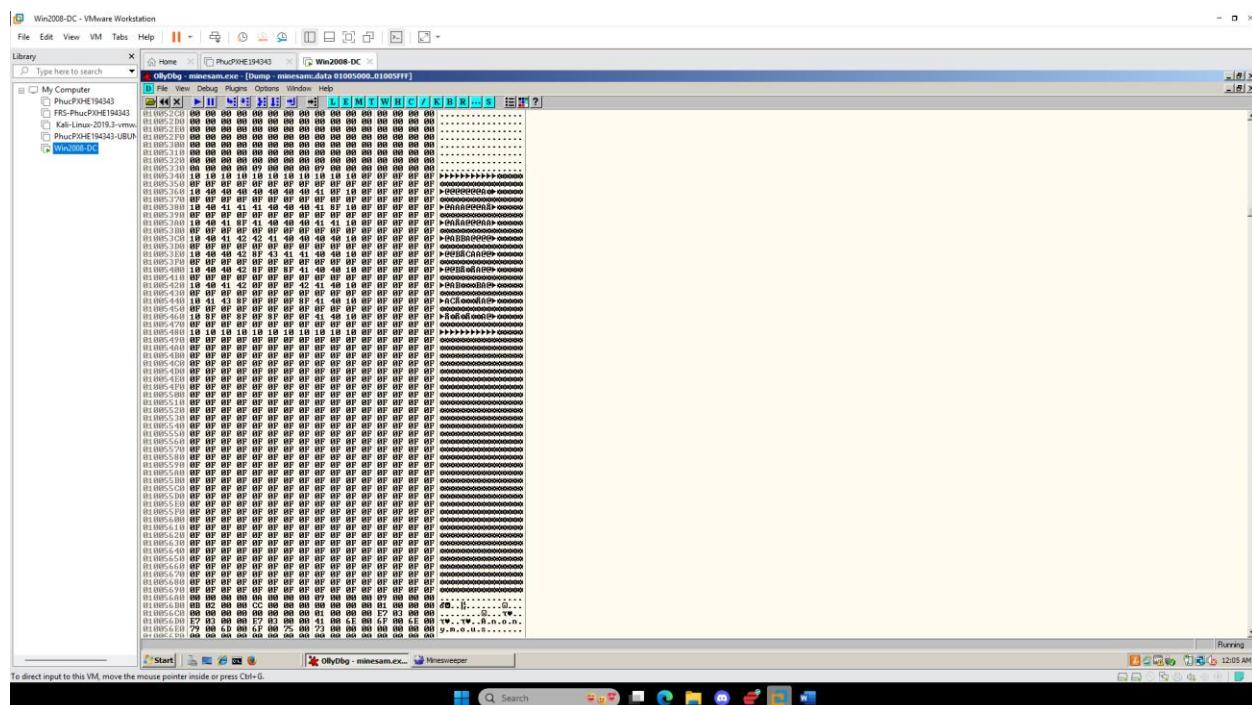
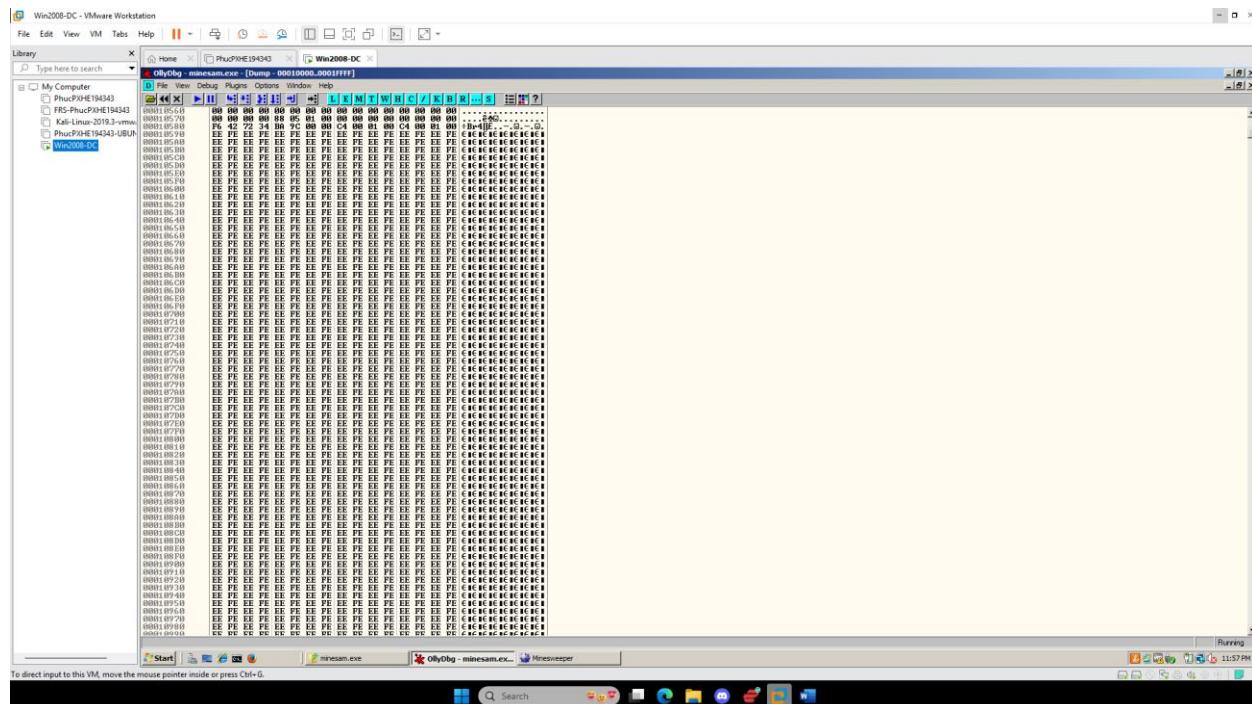
Home PhucPXHE19434 [Memory map] Win2008-DC

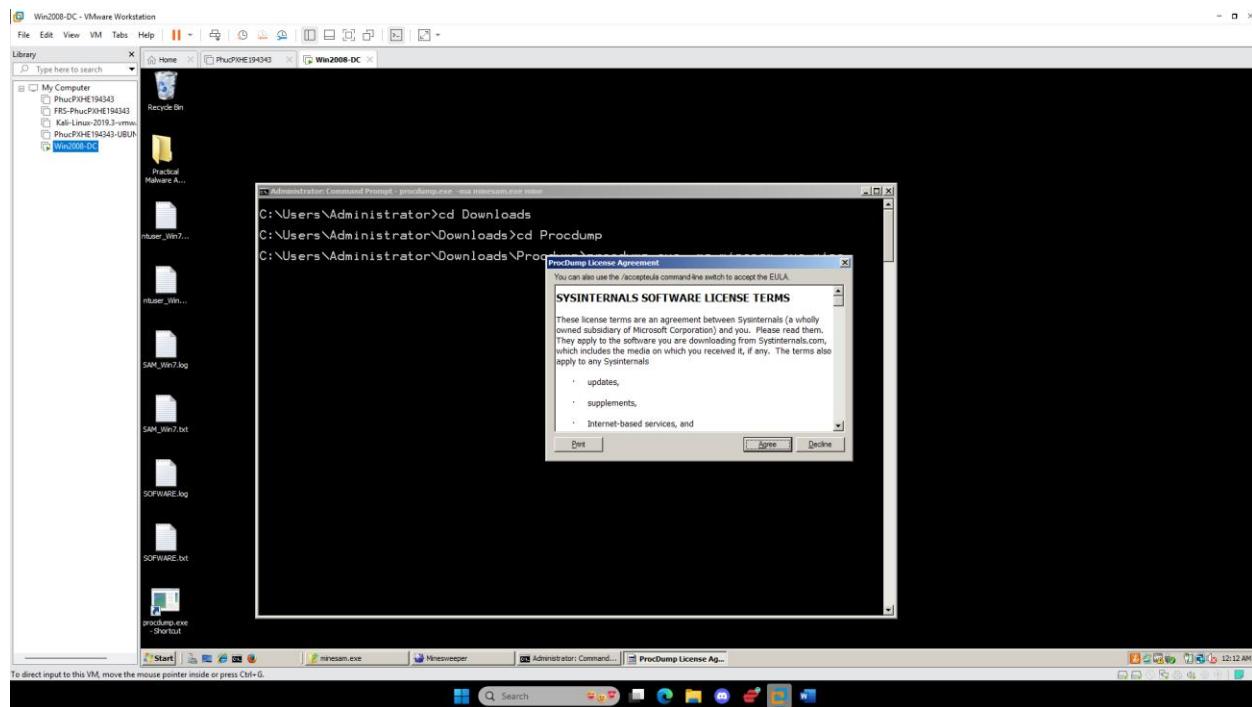
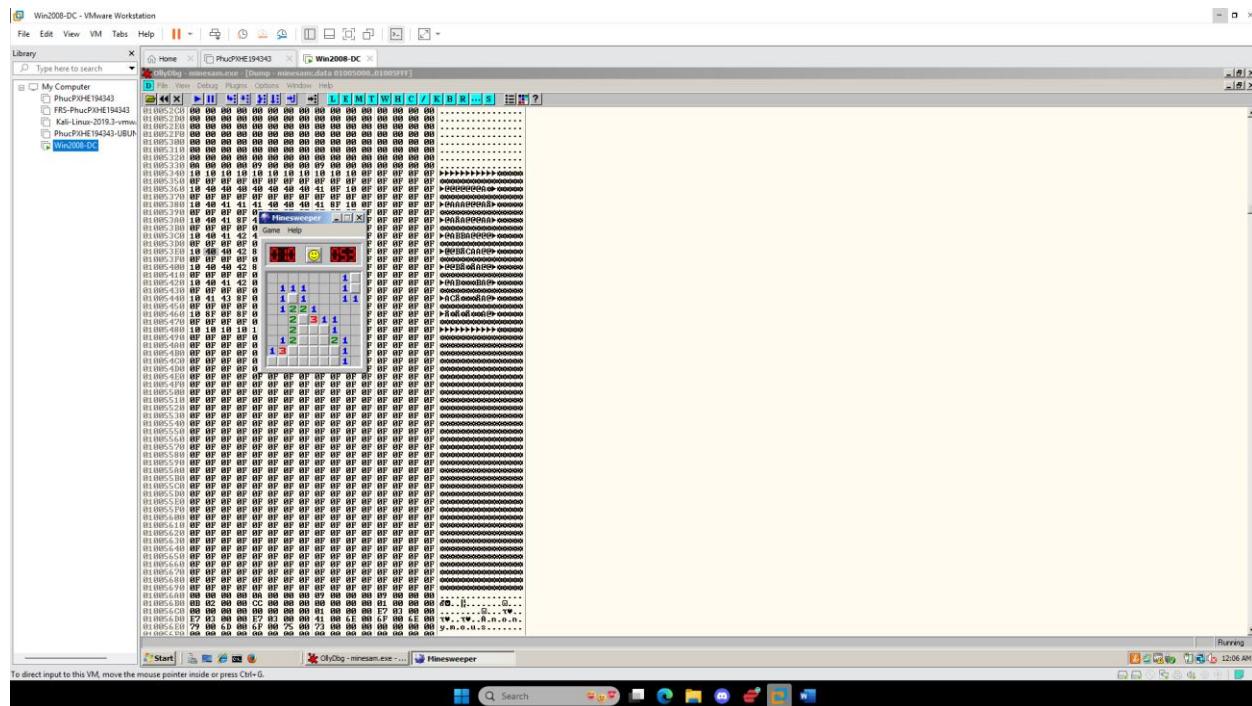
File View Debug Plugins Options Window Help

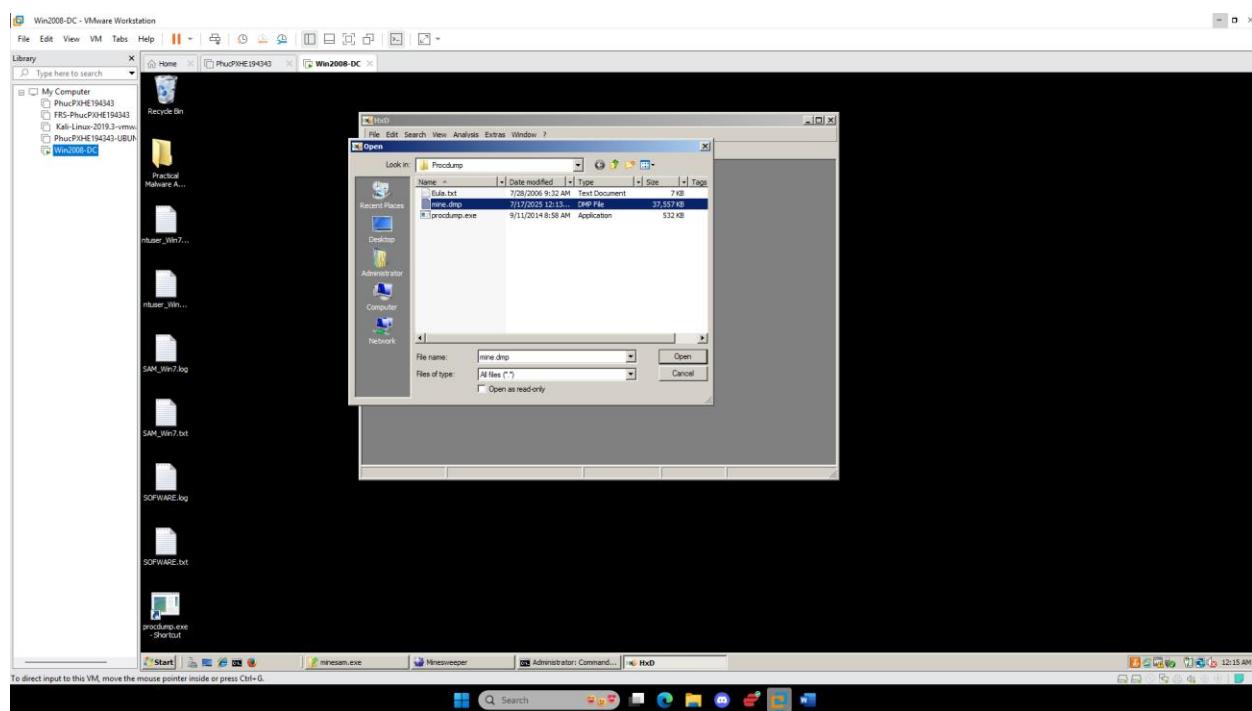
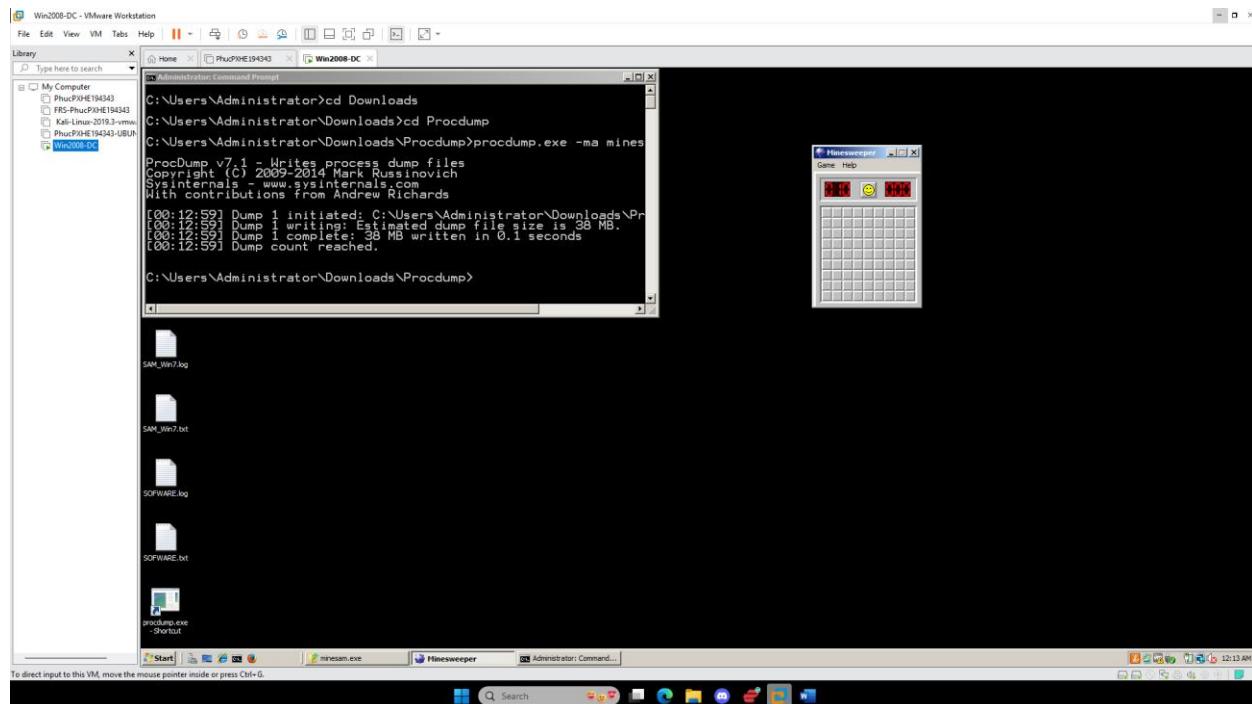
Address Size Owner Section Contains Type Rwx Initial Mapped as

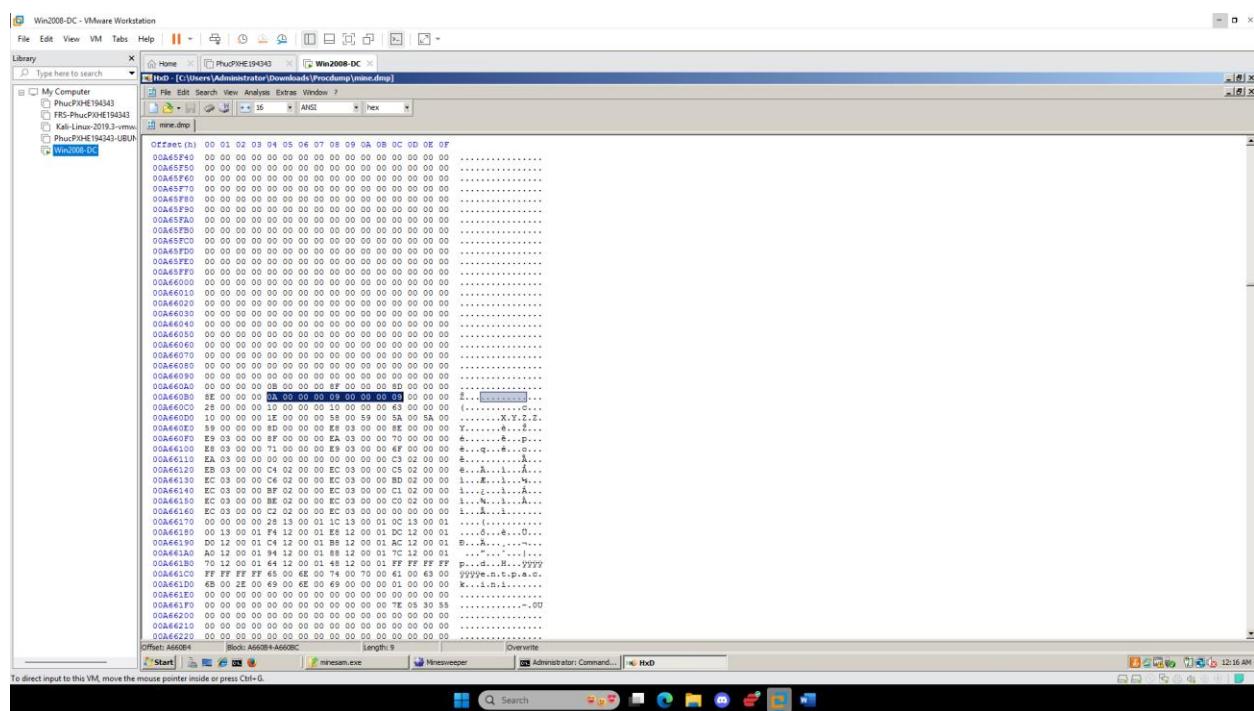
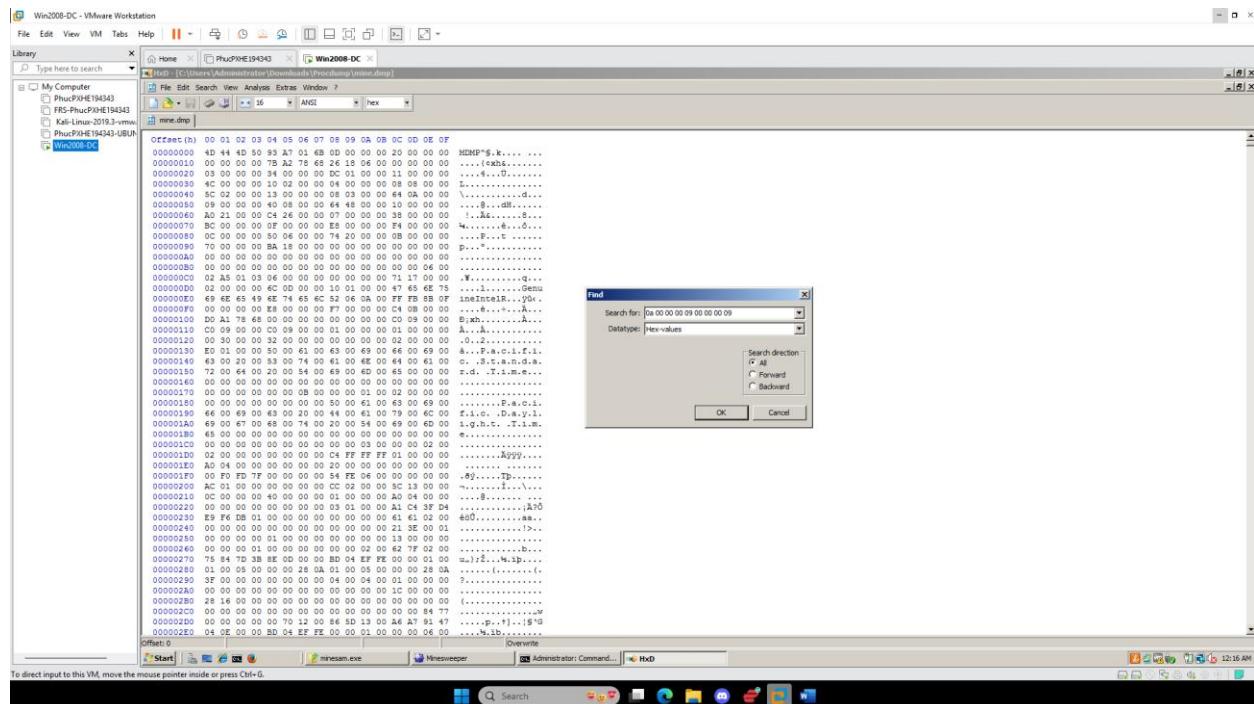
Address	Size	Owner	Section	Contains	Type	Rwx	Initial	Mapped as
00420000	00010000			Pri	RU			
00421000	00010000			Pri	RU			
00422000	00010000			Pri	RU			
00423000	00010000			Pri	RU			
00424000	00010000			Pri	RU			
00425000	00010000			Pri	RU			
00426000	00010000			Pri	RU			
00427000	00010000			Pri	RU			
00428000	00010000			Pri	RU			
00429000	00010000			Pri	RU			
0042A000	00010000			Pri	RU			
0042B000	00010000			Pri	RU			
0042C000	00010000			Pri	RU			
0042D000	00010000			Pri	RU			
0042E000	00010000			Pri	RU			
0042F000	00010000			Pri	RU			
00430000	00010000			Pri	RU			
00431000	00010000			Pri	RU			
00432000	00010000			Pri	RU			
00433000	00010000			Pri	RU			
00434000	00010000			Pri	RU			
00435000	00010000			Pri	RU			
00436000	00010000			Pri	RU			
00437000	00010000			Pri	RU			
00438000	00010000			Pri	RU			
00439000	00010000			Pri	RU			
0043A000	00010000			Pri	RU			
0043B000	00010000			Pri	RU			
0043C000	00010000			Pri	RU			
0043D000	00010000			Pri	RU			
0043E000	00010000			Pri	RU			
0043F000	00010000			Pri	RU			
00440000	00010000			Pri	RU			
00441000	00010000			Pri	RU			
00442000	00010000			Pri	RU			
00443000	00010000			Pri	RU			
00444000	00010000			Pri	RU			
00445000	00010000			Pri	RU			
00446000	00010000			Pri	RU			
00447000	00010000			Pri	RU			
00448000	00010000			Pri	RU			
00449000	00010000			Pri	RU			
0044A000	00010000			Pri	RU			
0044B000	00010000			Pri	RU			
0044C000	00010000			Pri	RU			
0044D000	00010000			Pri	RU			
0044E000	00010000			Pri	RU			
0044F000	00010000			Pri	RU			
00450000	00010000			Pri	RU			
00451000	00010000			Pri	RU			
00452000	00010000			Pri	RU			
00453000	00010000			Pri	RU			
00454000	00010000			Pri	RU			
00455000	00010000			Pri	RU			
00456000	00010000			Pri	RU			
00457000	00010000			Pri	RU			
00458000	00010000			Pri	RU			
00459000	00010000			Pri	RU			
0045A000	00010000			Pri	RU			
0045B000	00010000			Pri	RU			
0045C000	00010000			Pri	RU			
0045D000	00010000			Pri	RU			
0045E000	00010000			Pri	RU			
0045F000	00010000			Pri	RU			
00460000	00010000			Pri	RU			
00461000	00010000			Pri	RU			
00462000	00010000			Pri	RU			
00463000	00010000			Pri	RU			
00464000	00010000			Pri	RU			
00465000	00010000			Pri	RU			
00466000	00010000			Pri	RU			
00467000	00010000			Pri	RU			
00468000	00010000			Pri	RU			
00469000	00010000			Pri	RU			
0046A000	00010000			Pri	RU			
0046B000	00010000			Pri	RU			
0046C000	00010000			Pri	RU			
0046D000	00010000			Pri	RU			
0046E000	00010000			Pri	RU			
0046F000	00010000			Pri	RU			
00470000	00010000			Pri	RU			
00471000	00010000			Pri	RU			
00472000	00010000			Pri	RU			
00473000	00010000			Pri	RU			
00474000	00010000			Pri	RU			
00475000	00010000			Pri	RU			
00476000	00010000			Pri	RU			
00477000	00010000			Pri	RU			
00478000	00010000			Pri	RU			
00479000	00010000			Pri	RU			
0047A000	00010000			Pri	RU			
0047B000	00010000			Pri	RU			
0047C000	00010000			Pri	RU			
0047D000	00010000			Pri	RU			
0047E000	00010000			Pri	RU			
0047F000	00010000			Pri	RU			
00480000	00010000			Pri	RU			
00481000	00010000			Pri	RU			
00482000	00010000			Pri	RU			
00483000	00010000			Pri	RU			
00484000	00010000			Pri	RU			
00485000	00010000			Pri	RU			
00486000	00010000			Pri	RU			
00487000	00010000			Pri	RU			
00488000	00010000			Pri	RU			
00489000	00010000			Pri	RU			
0048A000	00010000			Pri	RU			
0048B000	00010000			Pri	RU			
0048C000	00010000			Pri	RU			
0048D000	00010000			Pri	RU			
0048E000	00010000			Pri	RU			
0048F000	00010000			Pri	RU			
00490000	00010000			Pri	RU			
00491000	00010000			Pri	RU			
00492000	00010000			Pri	RU			
00493000	00010000			Pri	RU			
00494000	00010000			Pri	RU			
00495000	00010000			Pri	RU			
00496000	00010000			Pri	RU			
00497000	00010000			Pri	RU			
00498000	00010000			Pri	RU			
00499000	00010000			Pri	RU			
0049A000	00010000			Pri	RU			
0049B000	00010000			Pri	RU			
0049C000	00010000			Pri	RU			
0049D000	00010000			Pri	RU			
0049E000	00010000			Pri	RU			
0049F000	00010000			Pri	RU			
004A0000	00010000			Pri	RU			
004A1000	00010000			Pri	RU			
004A2000	00010000			Pri	RU			
004A3000	00010000			Pri	RU			
004A4000	00010000			Pri	RU			
004A5000	00010000			Pri	RU			
004A6000	00010000			Pri	RU			
004A7000	00010000			Pri	RU			
004A8000	00010000			Pri	RU			
004A9000	00010000			Pri	RU			
004AA000	00010000			Pri	RU			
004AB000	00010000			Pri	RU			
004AC000	00010000			Pri	RU			
004AD000	00010000			Pri	RU			
004AE000	00010000			Pri	RU			
004AF000	00010000			Pri	RU			
004B0000	00010000			Pri	RU			
004B1000	00010000			Pri	RU			
004B2000	00010000			Pri	RU			
004B3000	00010000			Pri	RU			
004B4000	00010000			Pri	RU			
004B5000	00010000			Pri	RU			
004B6000	00010000			Pri	RU			
004B7000	00010000			Pri	RU			
004B8000	00010000			Pri	RU			
004B9000	00010000			Pri	RU			
004BA000	00010000			Pri	RU			
004BB000	00010000			Pri	RU			
004BC000	00010000			Pri	RU			
004BD000	00010000			Pri	RU			
004BE000	00010000			Pri	RU			
004BF000	00010000			Pri	RU			
004C0000	00010000			Pri	RU			
004C1000	00010000			Pri	RU			
004C2000	00010000			Pri	RU			
004C3000	00010000			Pri	RU			
004C4000	00010000			Pri	RU			
004C5000	00010000			Pri	RU			
004C6000	00010000			Pri	RU			
004C7000	00010000			Pri	RU			
004C8000	00010000			Pri	RU			
004C9000	00010000			Pri	RU			
004CA000	00010000			Pri	RU			
004CB000	00010000			Pri	RU			
004CC000	00010000			Pri	RU			
004CD000	00010000			Pri	RU			
004CE000	00010000			Pri	RU			
004CF000	00010000			Pri	RU			
004D0000	00010000			Pri	RU			
004D1000	00010000			Pri	RU			
004D2000	00010000			Pri	RU			
004D3000	00010000			Pri	RU			
004D4000	00010000			Pri	RU			
004D5000	00010000			Pri	RU			
004D6000	00010000			Pri	RU			
004D7000	00010000			Pri	RU			
004D8000	00010000			Pri	RU			
004D9000	00010000			Pri	RU			
004DA000	00010000			Pri	RU			
004DB000	00010000			Pri	RU			
004DC000	00010000			Pri	RU			
004DD000	00010000			Pri	RU			
004DE000	00010000			Pri	RU			
004DF000	00010000			Pri	RU			
004E0000	00010000			Pri	RU			
004E1000	00010000			Pri	RU			

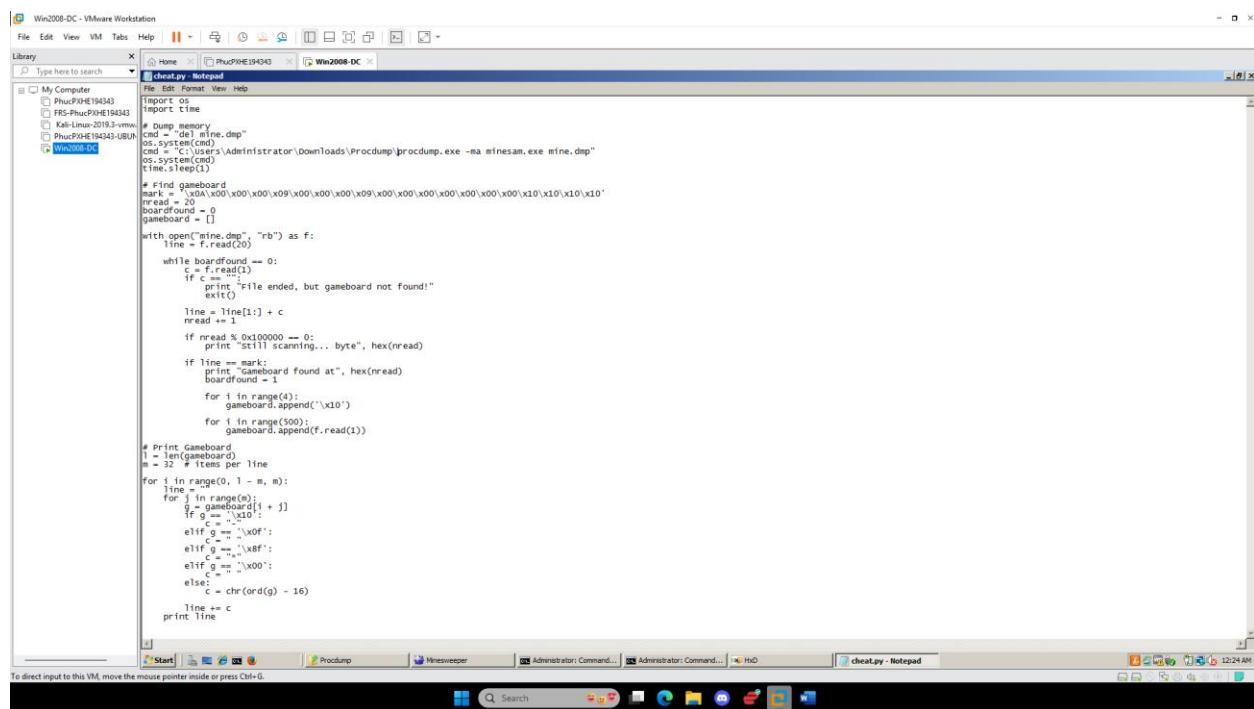
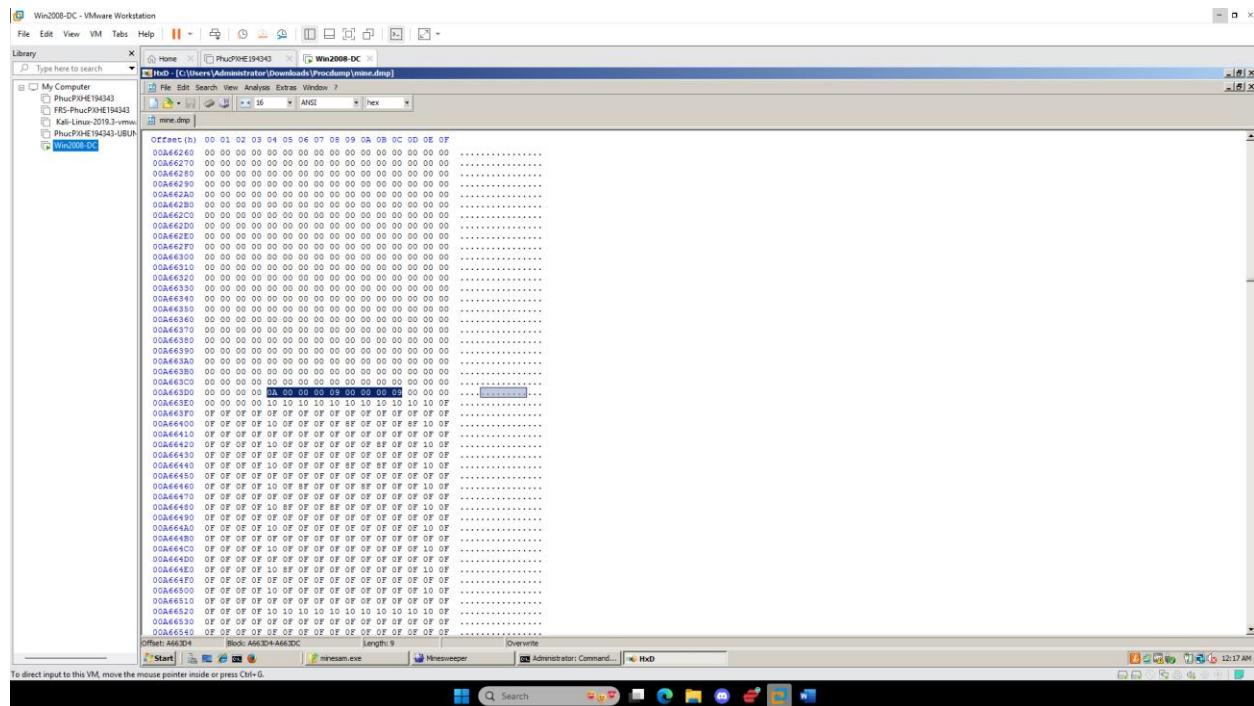












Win2008-DC - VMware Workstation

File Edit View VM Tabs Help

Type here to search

Library

cheat.py - Notepad

```
File Edit Format View Help
Import os
Import time
# dump memory
cmd = "(de)mine.dmp"
cmd = "cmd /c %s" % cmd
cmd = "%s > %s" % (cmd, "C:\Users\Administrator\Downloads\ProcDump\procdump.exe -ma minesam.exe mine.dmp")
os.system(cmd)
time.sleep(1)

# find gameboard
mark = "\x28\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x10\x10\x10"
nread = 20
boardfound = 0
gameboard = []
gameboard.append("")

with open("mine.dmp", "rb") as f:
    line = f.read(20)
    while boardfound == 0:
        c = ord(line[0])
        if c == mark:
            print "file ended, but gameboard not found!"
            exit(0)
        line = line[1:] + c
        nread += 1
        if nread % 0x100000 == 0:
            print "still scanning... byte", hex(nread)
        if line == mark:
            print "Gameboard found at", hex(nread)
            boardfound = 1
            for i in range(4):
                gameboard.append("\x10")
            for i in range(500):
                gameboard.append(f.read(1))

# Print Gameboard
m = len(gameboard)
for i in range(m):
    line = gameboard[i]
    for j in range(8):
        if g == "\x10":
            c = " "
        elif g == "\x0F":
            c = "#"
        elif g == "\x8F":
            c = "="
        elif g == "\x00":
            c = "."
        else:
            c = chr(ord(g) - 16)
        line += c
    print line

Start | Programs | Help | ProcDump | Minesweeper | Administrator: Command... | Administrator: Command... | HxD | cheat.py - Notepad
```

The screenshot shows a Windows 2008 DC - VMware Workstation interface. The left pane displays the file explorer with various virtual machine snapshots. The right pane shows a Command Prompt window titled "Administrator: Command Prompt".

In the Command Prompt, the user runs the command:

```
C:\Users\Administrator\Desktop>python cheat.py
```

The output indicates that Python could not find the specified file:

```
Could Not Find C:\Users\Administrator\Desktop\mine.dmp
```

Then, the user runs the ProcDump command to create a dump file:

```
ProcDump v7.1 - Writes process dump files  
Copyright (C) 2009-2014 Mark Russinovich  
Sysinternals - www.sysinternals.com  
With contributions from Andrew Richards
```

The ProcDump output shows the creation of a dump file named "mine.dmp":

```
[00:22:23] Dump 1 initiated: C:\Users\Administrator\Desktop\mine.dmp  
[00:22:23] Dump 1 writing: Estimated dump file size is 38 MB.  
[00:22:24] Dump 1 complete: 38 MB written in 0.0 seconds  
[00:22:24] Dump count reached.
```

Below the dump command, the user runs the "cheat.py" script again:

```
Still scanning... byte 0x100000  
Still scanning... byte 0x200000  
Still scanning... byte 0x300000  
Still scanning... byte 0x400000  
Still scanning... byte 0x500000  
Still scanning... byte 0x600000  
Still scanning... byte 0x700000  
Still scanning... byte 0x800000  
Still scanning... byte 0x900000  
Still scanning... byte 0xa00000  
Gameboard found at 0xa66444  
- x x -  
- - x x -  
- x x x -  
- x x -  
- - -  
- - -
```

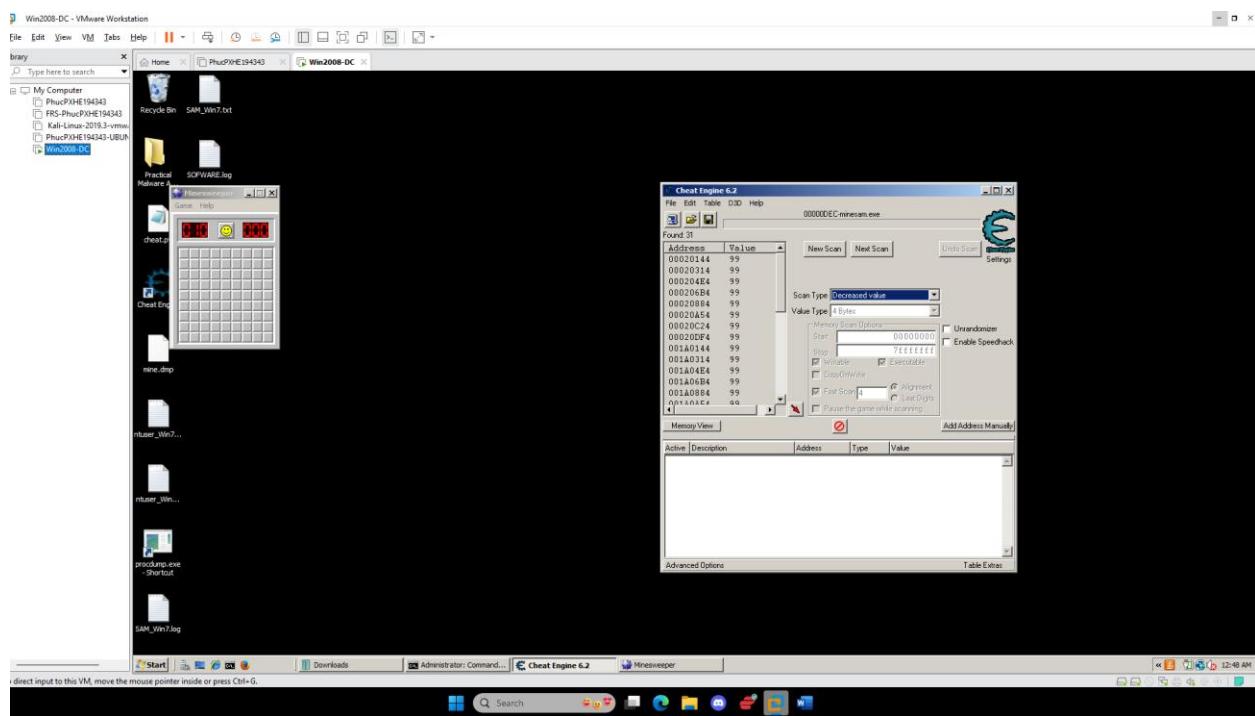
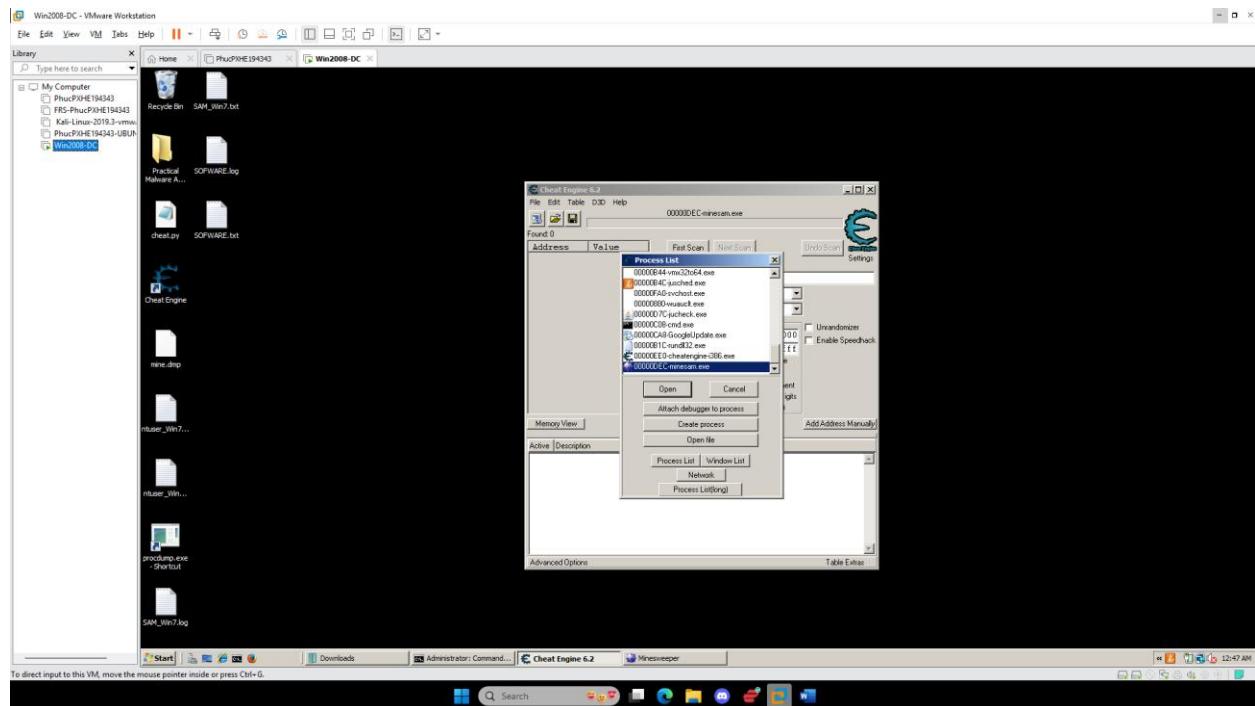
Finally, a "Hivesweeper" window is displayed, showing a solved 4x4 grid with the word "Smash!" as the secret word:

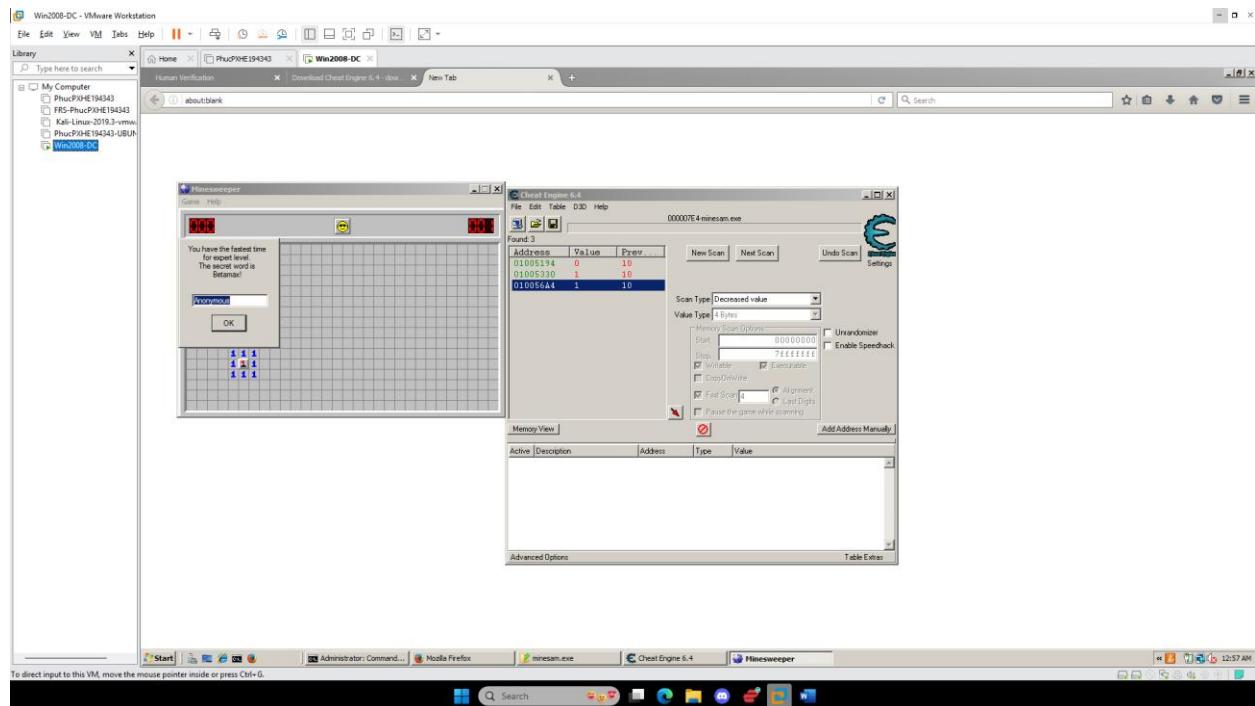
100	100	100	100
100	100	100	100
100	100	100	100
100	100	100	100

You have the fastest time
for beginner level!
The secret word is Smash!

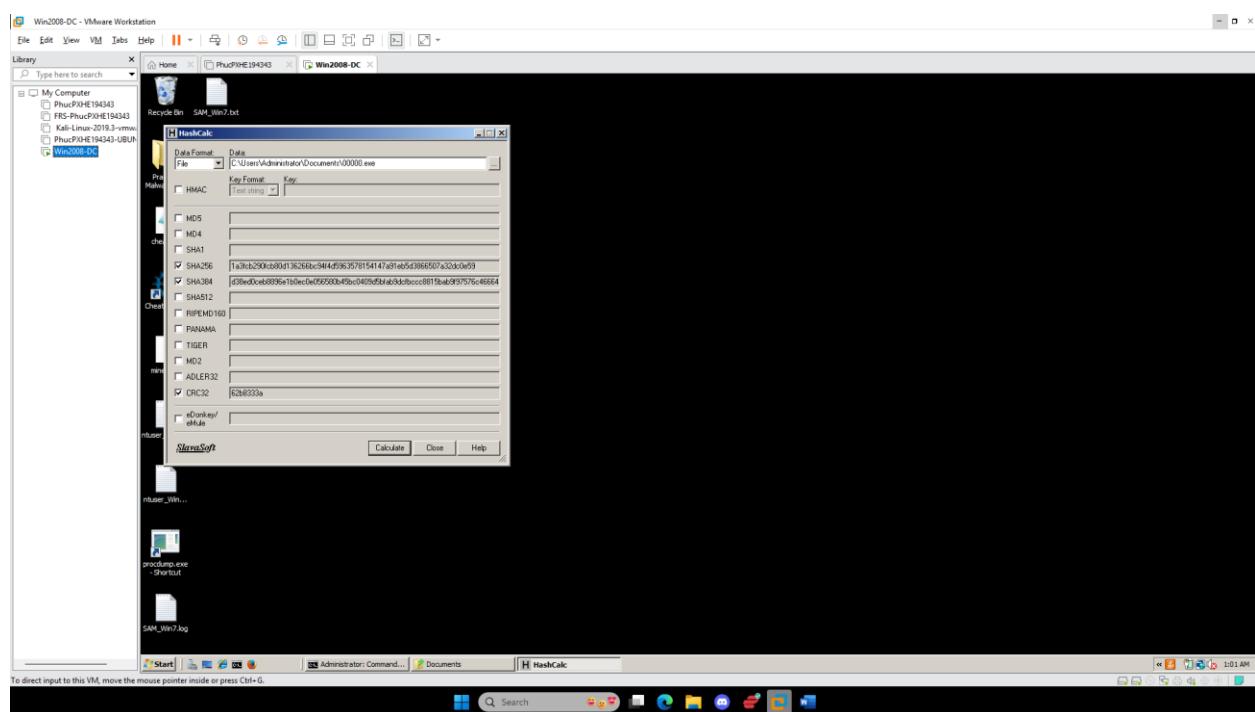
PhucPX-E19434

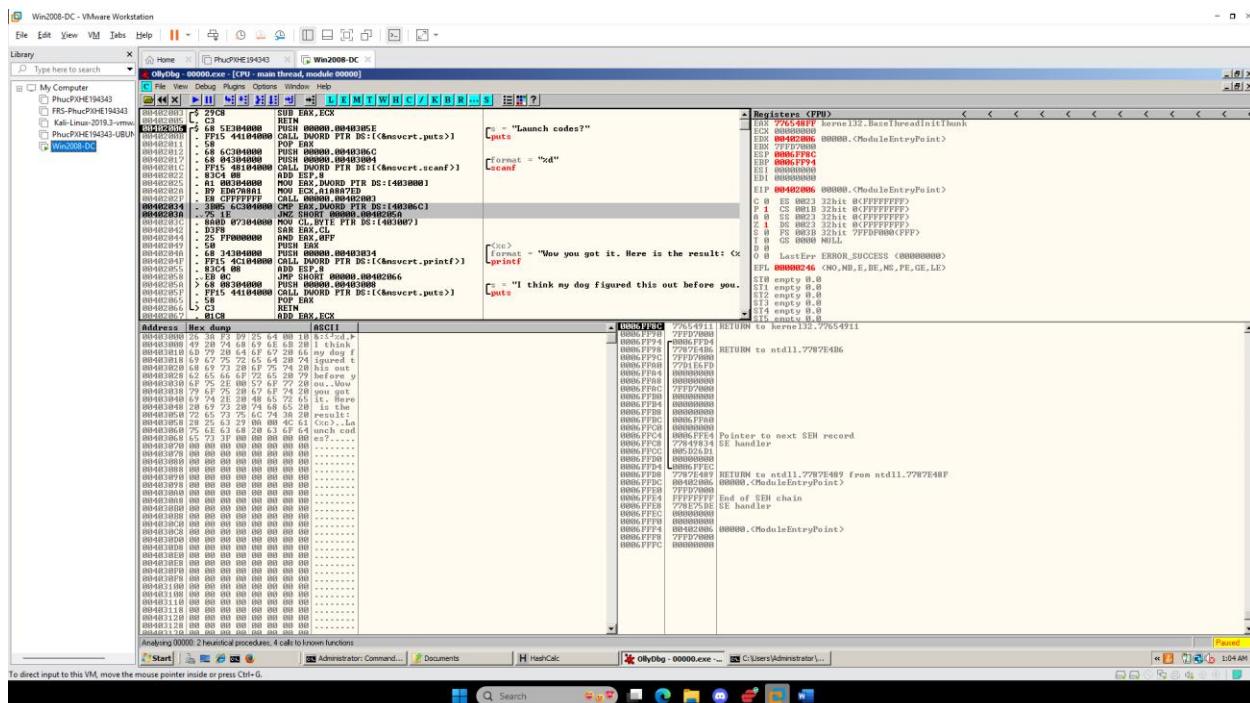
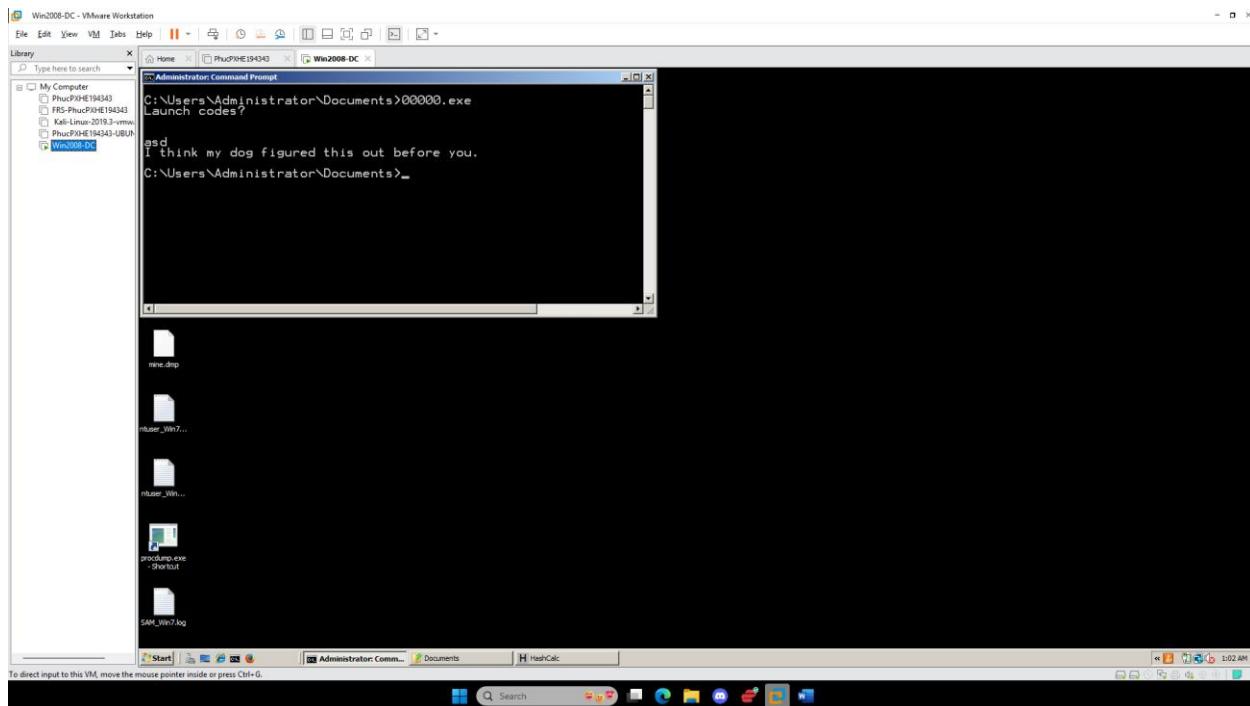
OK





Lab 18.2

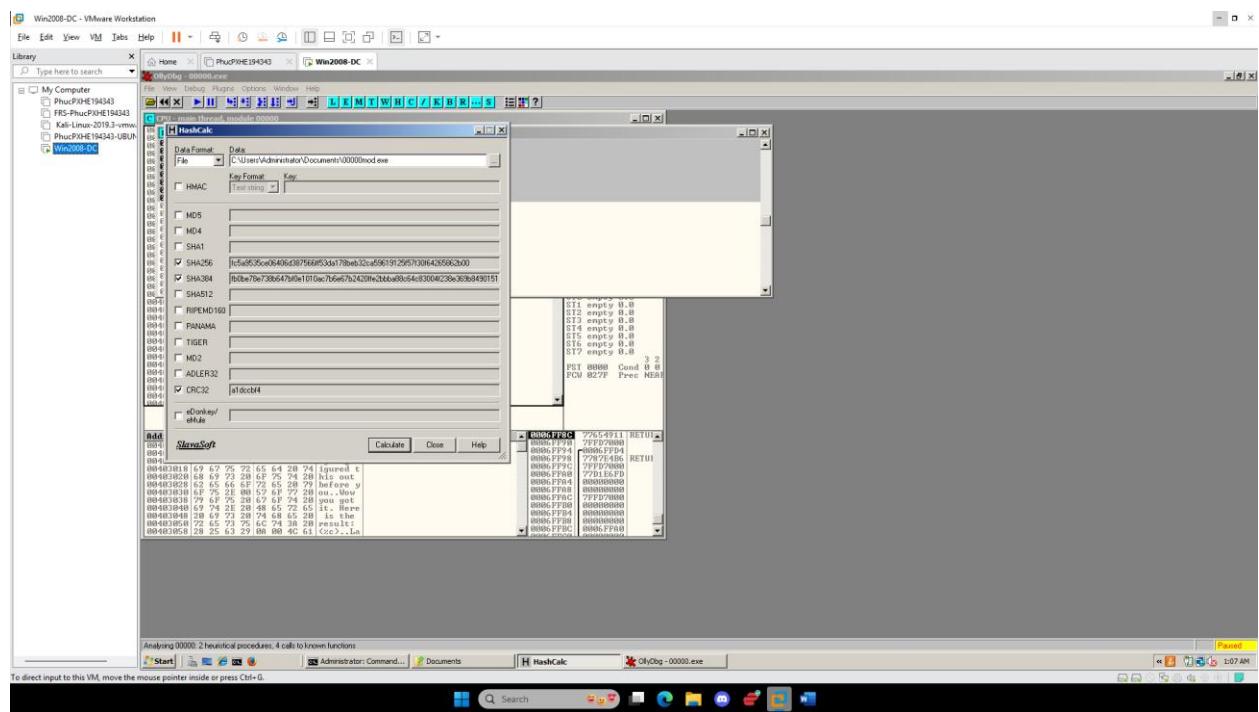
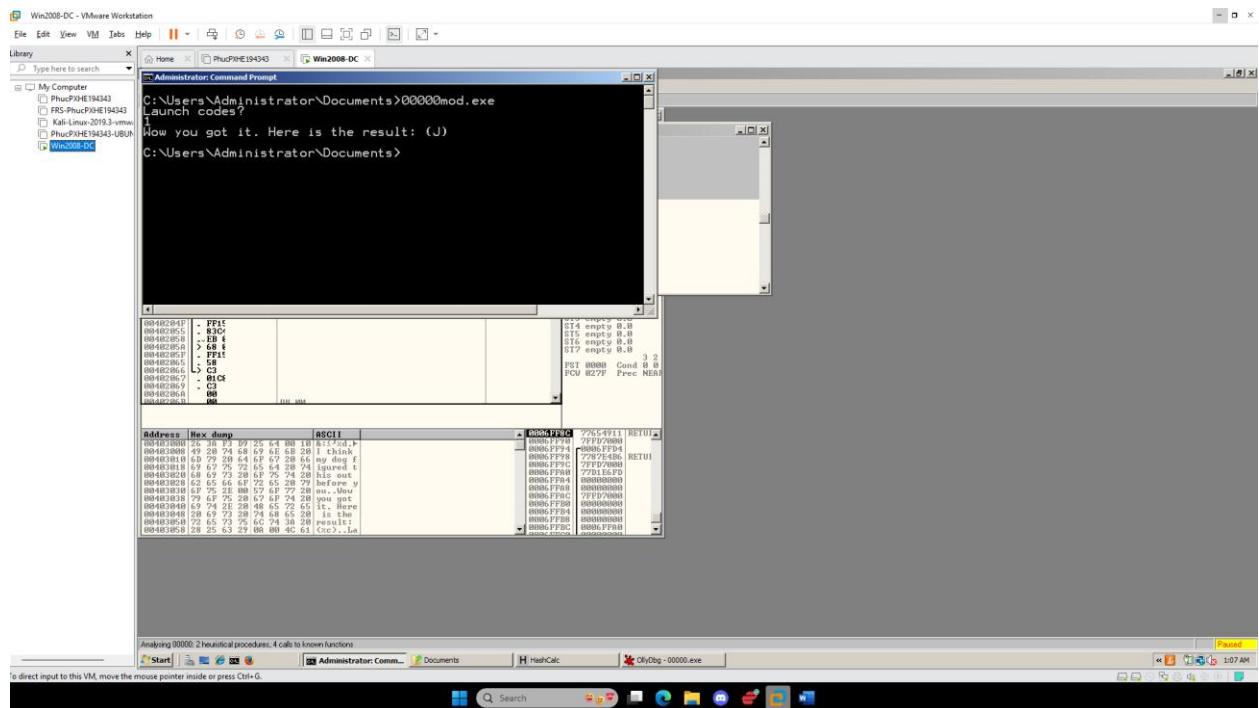


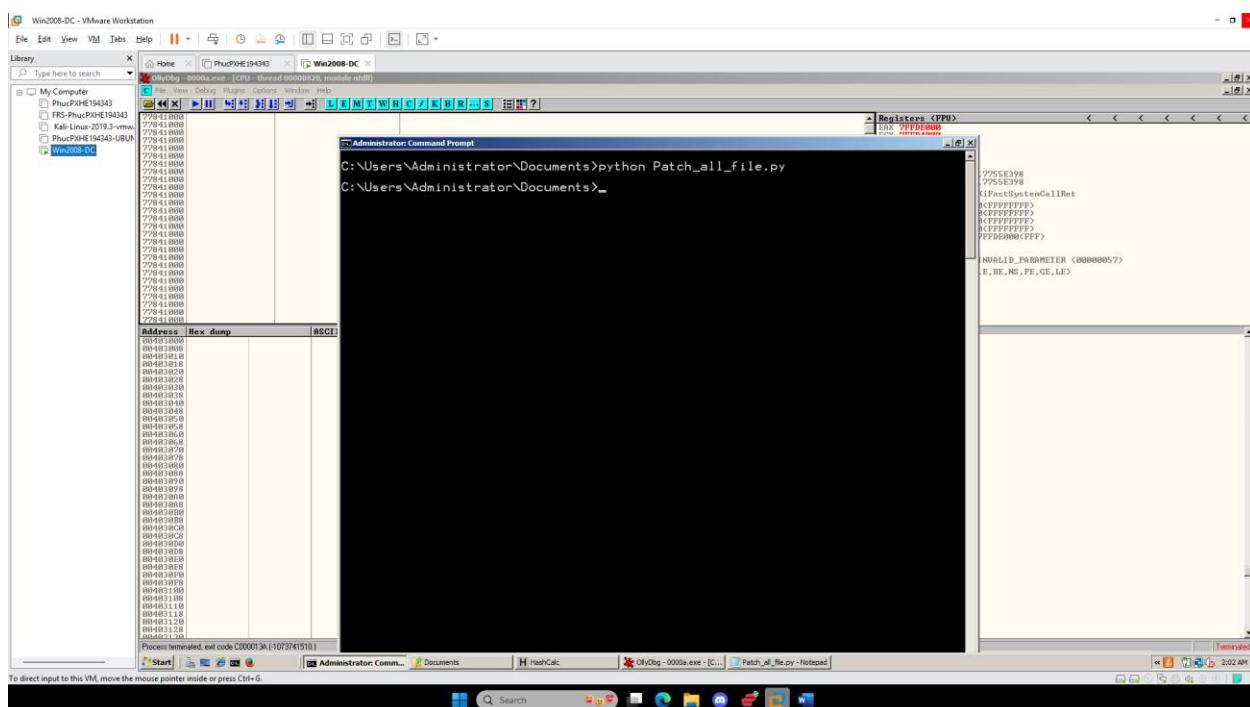
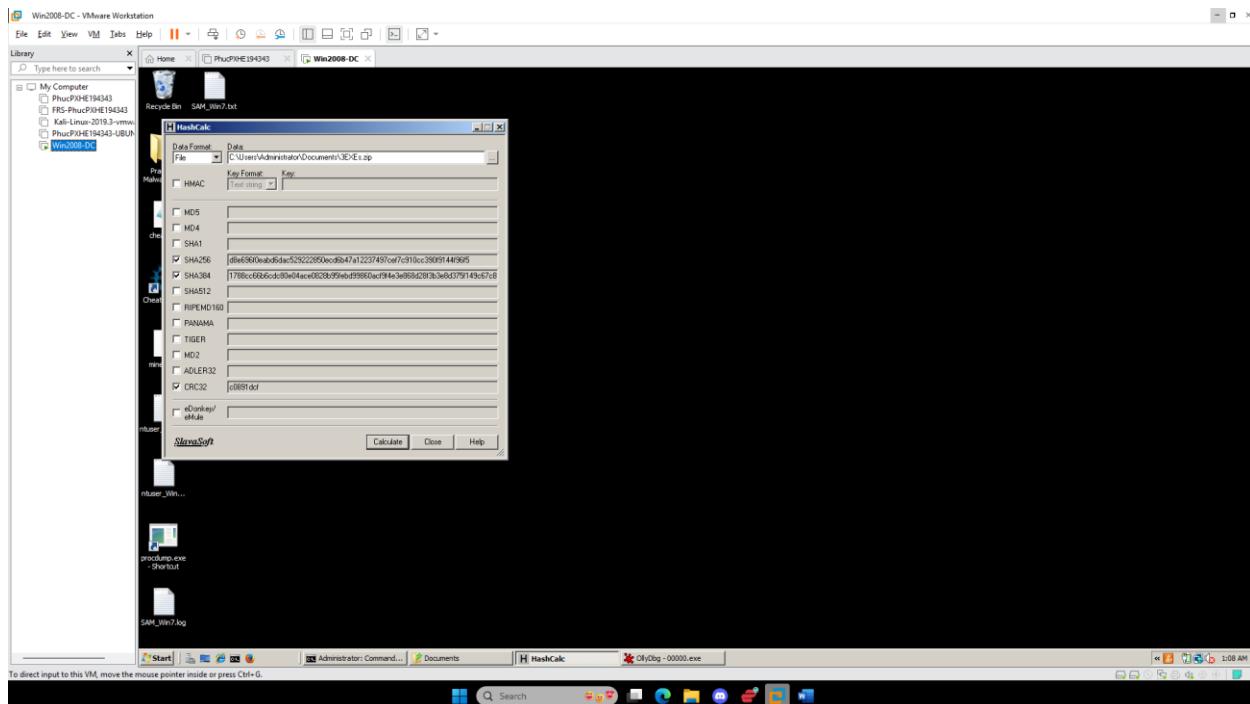


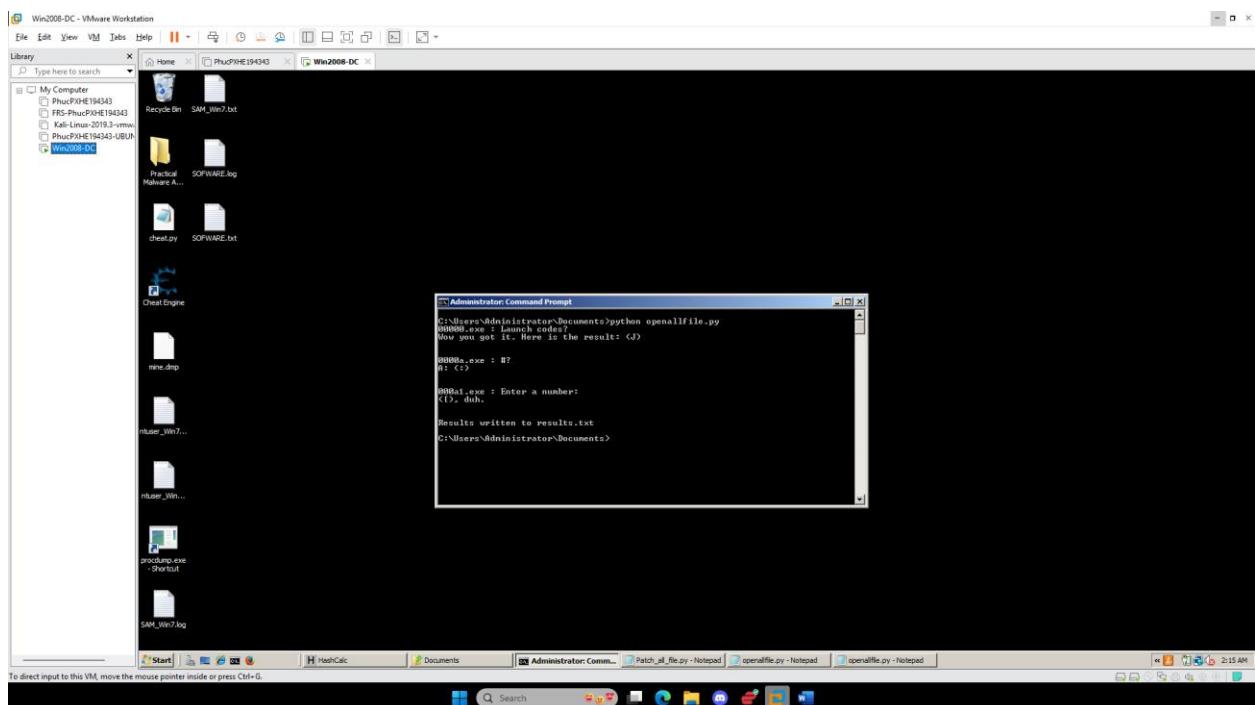
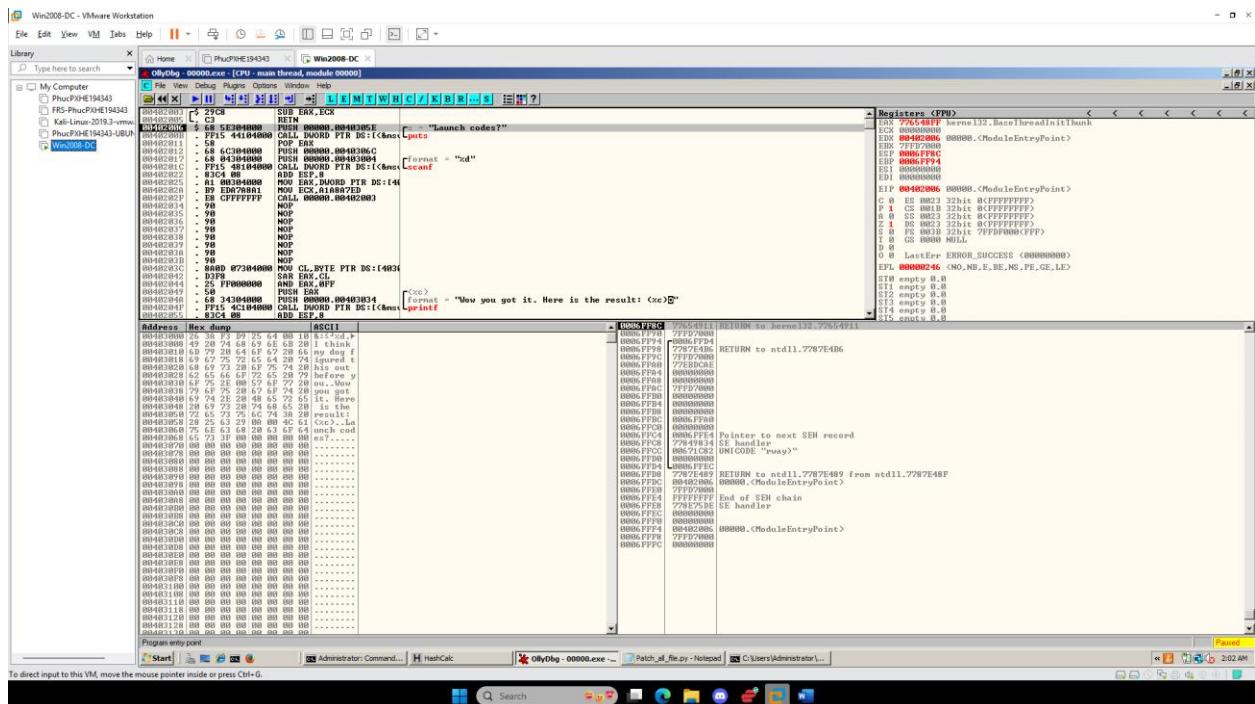
The screenshot shows the OllyDbg debugger interface. The assembly window displays assembly code, including a call to `_printf` and a printf-like string. The registers window shows CPU register values. The Registers window title bar says "Registers (CPU)" and lists registers ECX, EDX, EBP, EIP, etc. The assembly window title bar says "0x0040107F - CPU - main thread (mod=00000000)".

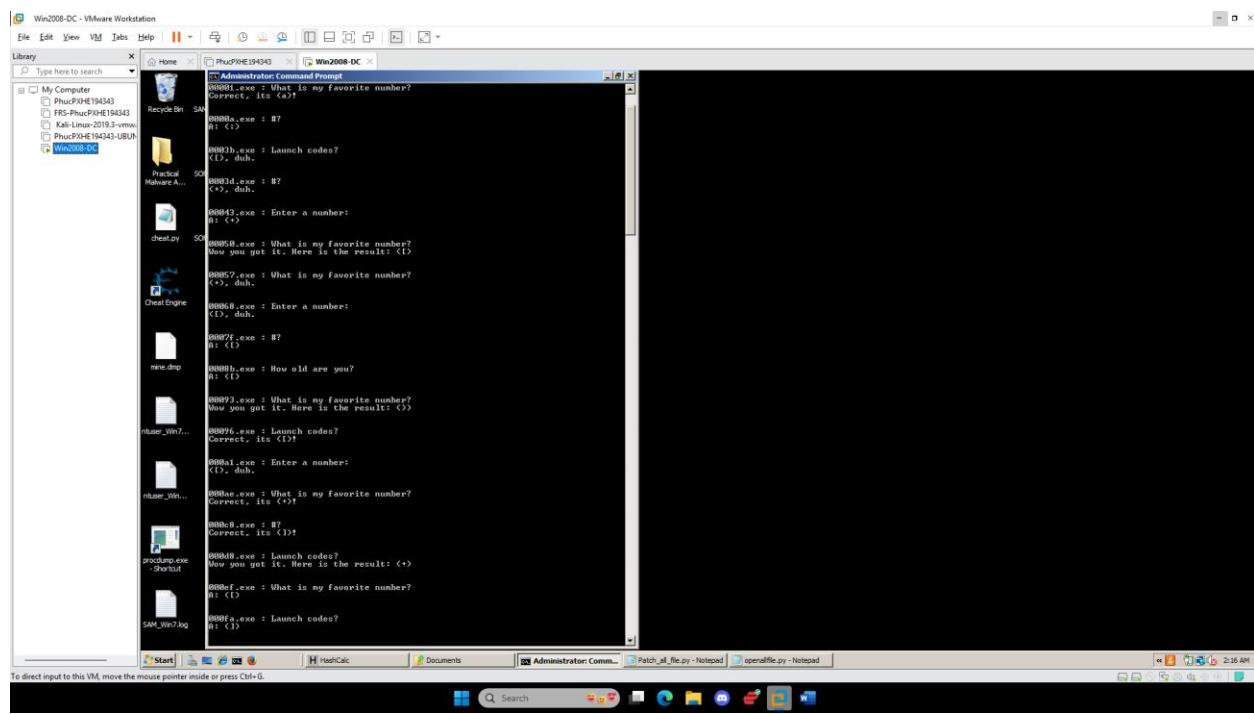
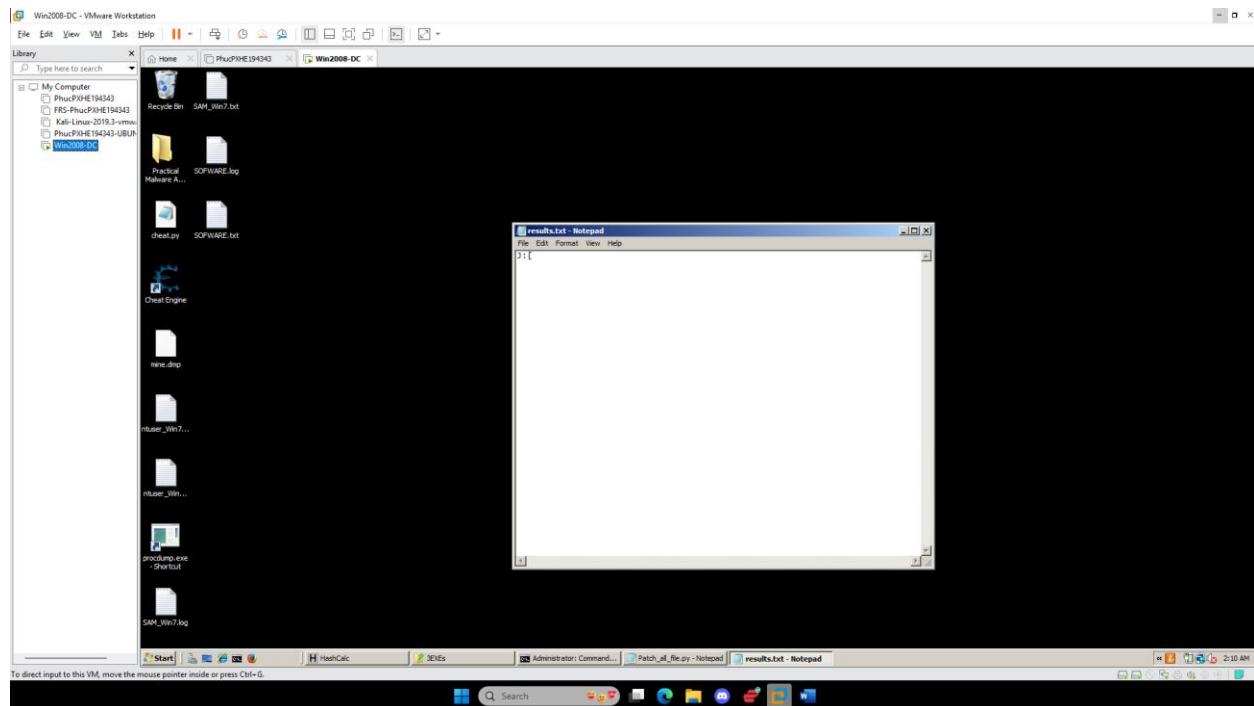
The screenshot shows the OllyDbg debugger interface running on a Windows 2008 DC host. The assembly window displays assembly code, and the memory dump windows show the state of memory at address 00000000. The status bar indicates the current assembly instruction is 00000000.

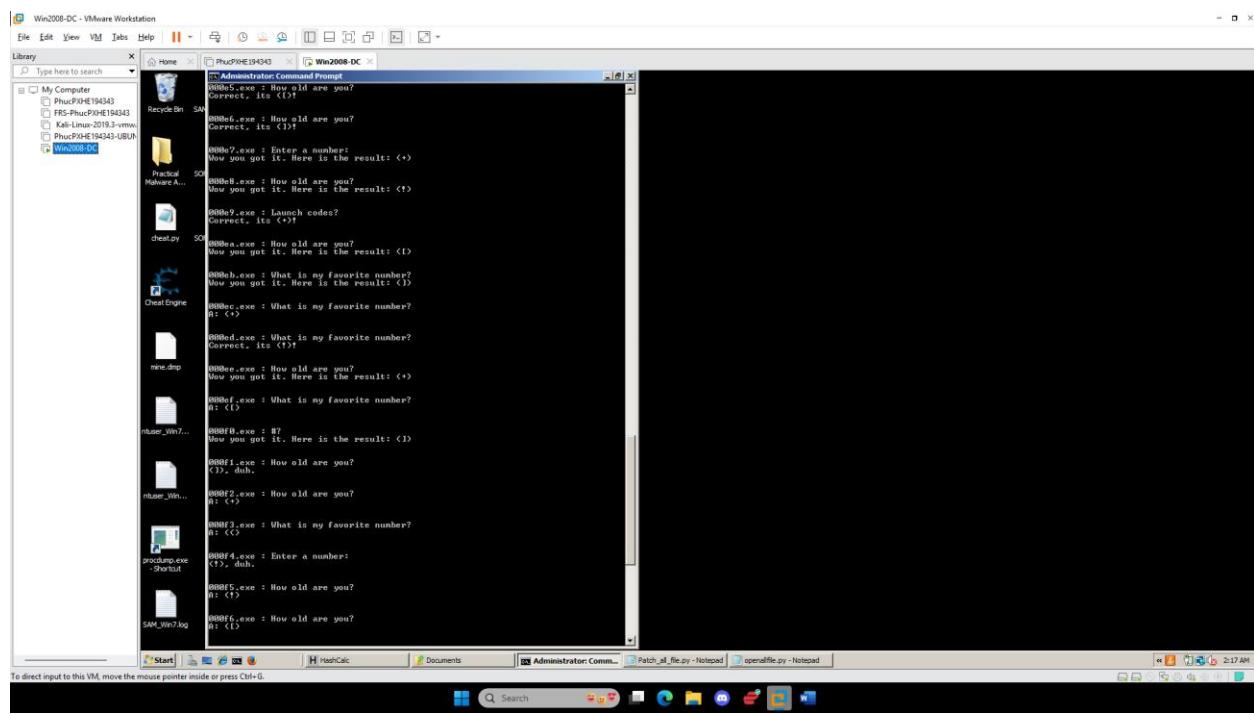
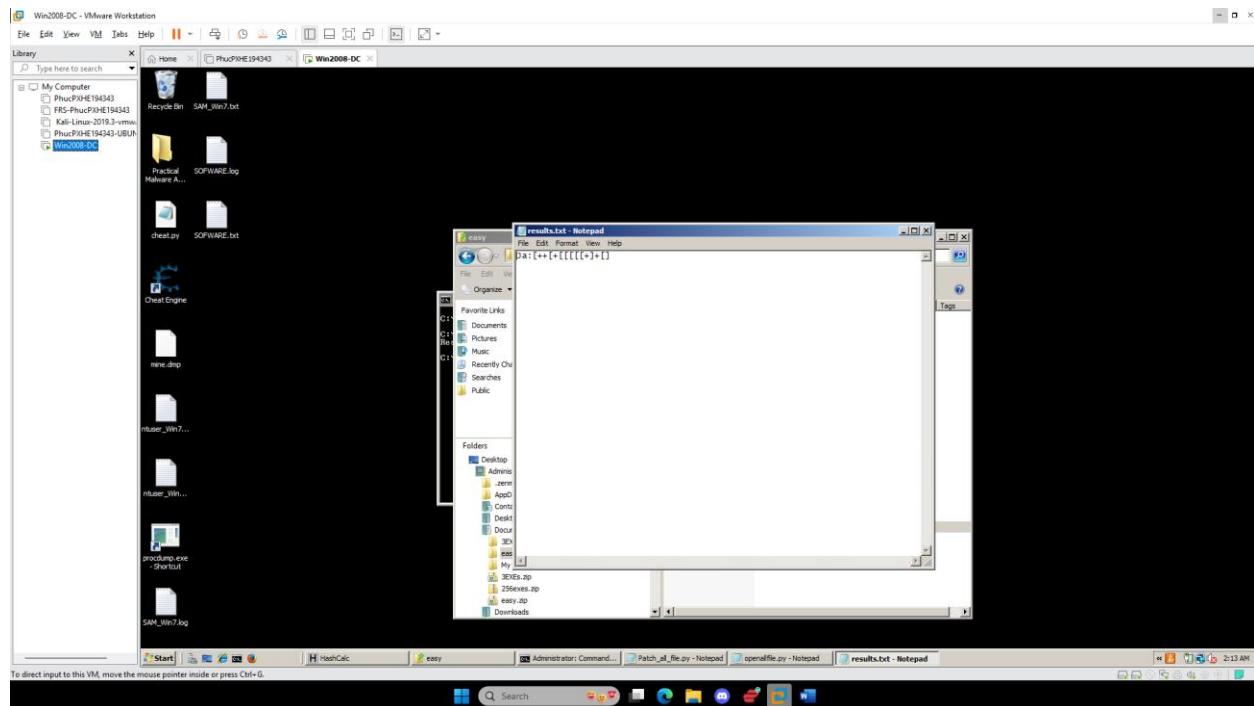
The screenshot shows the OllyDbg debugger interface running on a Win2008-DC VM. The assembly pane displays assembly code for the module 00000000, specifically the main thread's entry point at address 00000040. The code includes instructions like MOVS, PUSH, CALL, and RETN. The memory dump pane below shows the memory starting at address 0006FFFF, containing ASCII text and binary data. A context menu is open over the assembly code, with options like Backup, Copy, Insert, Assemble, Search for, Save file, Go to offset, Ctrl+G, View image in Disassembler, Hex, Text, Short, Long, ST6, Float, Disassemble, FST, FCW, Special, and Appearance. The status bar at the bottom indicates "Analyzing 00000: 2 heuristic procedures, 4 calls to known functions".

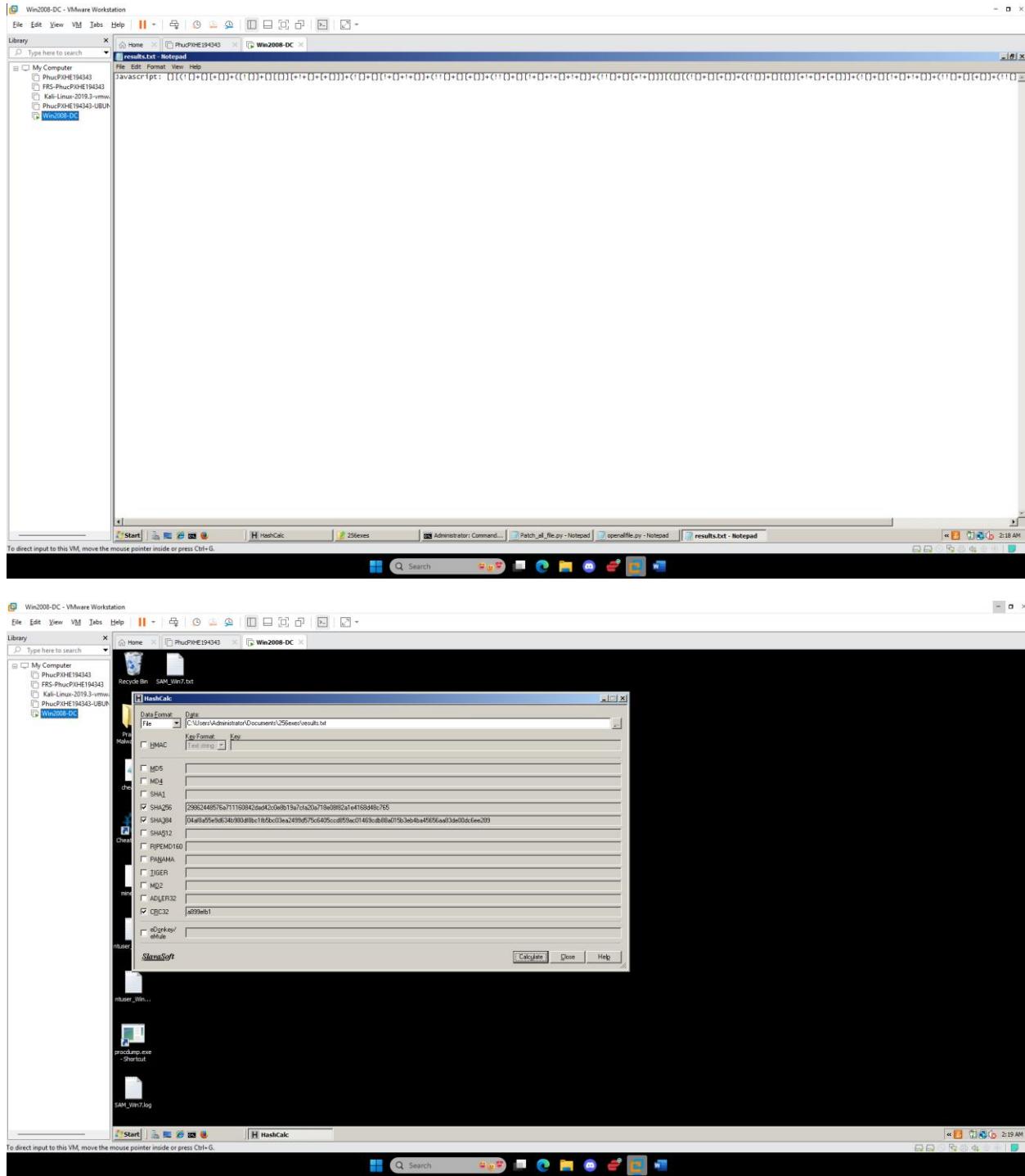








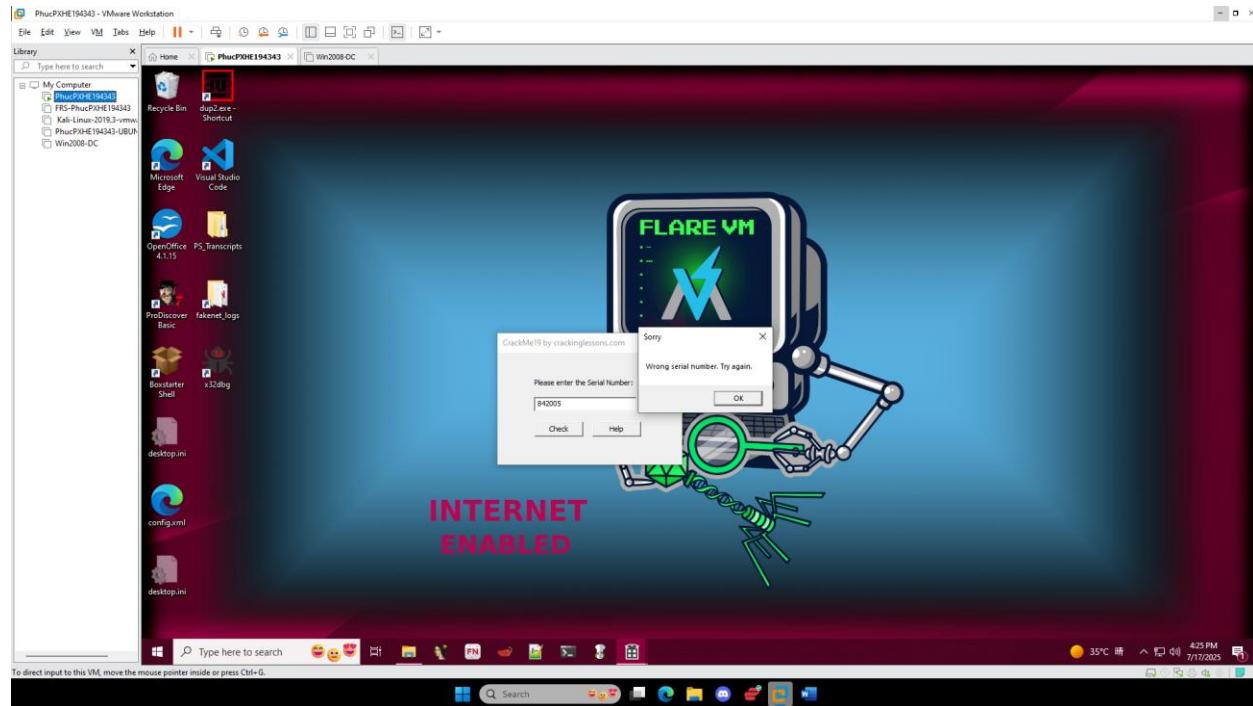




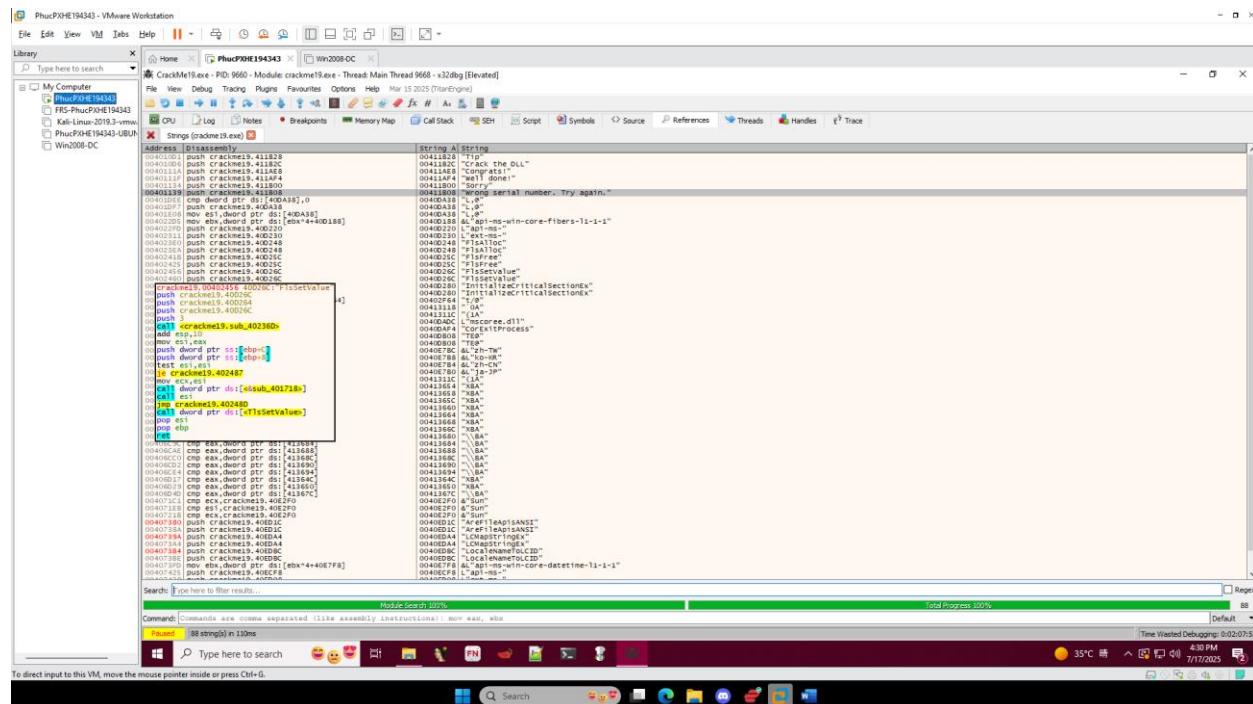
Crack Me 19

Step 1: Khi chạy thử tệp thực thi, có thể quan sát thấy chương trình yêu cầu người dùng nhập một số Serial. Nếu giá trị nhập vào không chính xác, chương trình sẽ hiển thị thông

báo lỗi. Điều này cho thấy có cơ chế kiểm tra hợp lệ của Serial Number được tích hợp bên trong chương trình.

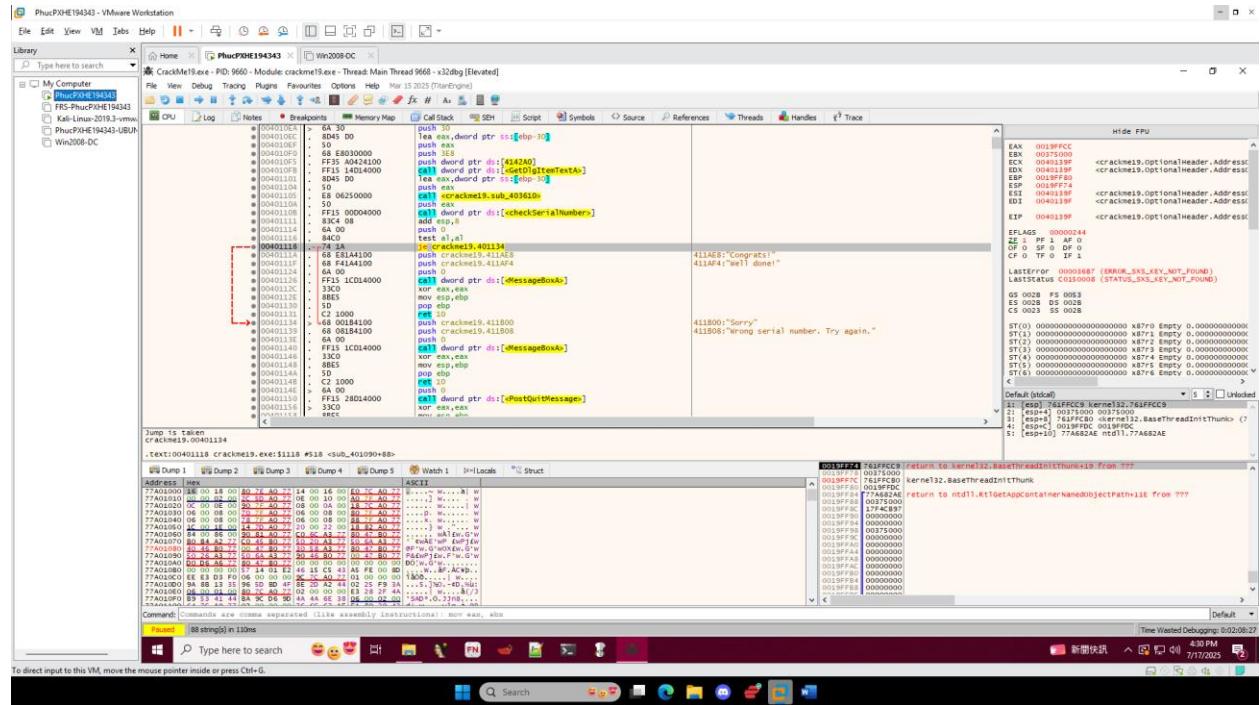


Step 2: Vào String preferences để tìm string cần tìm.

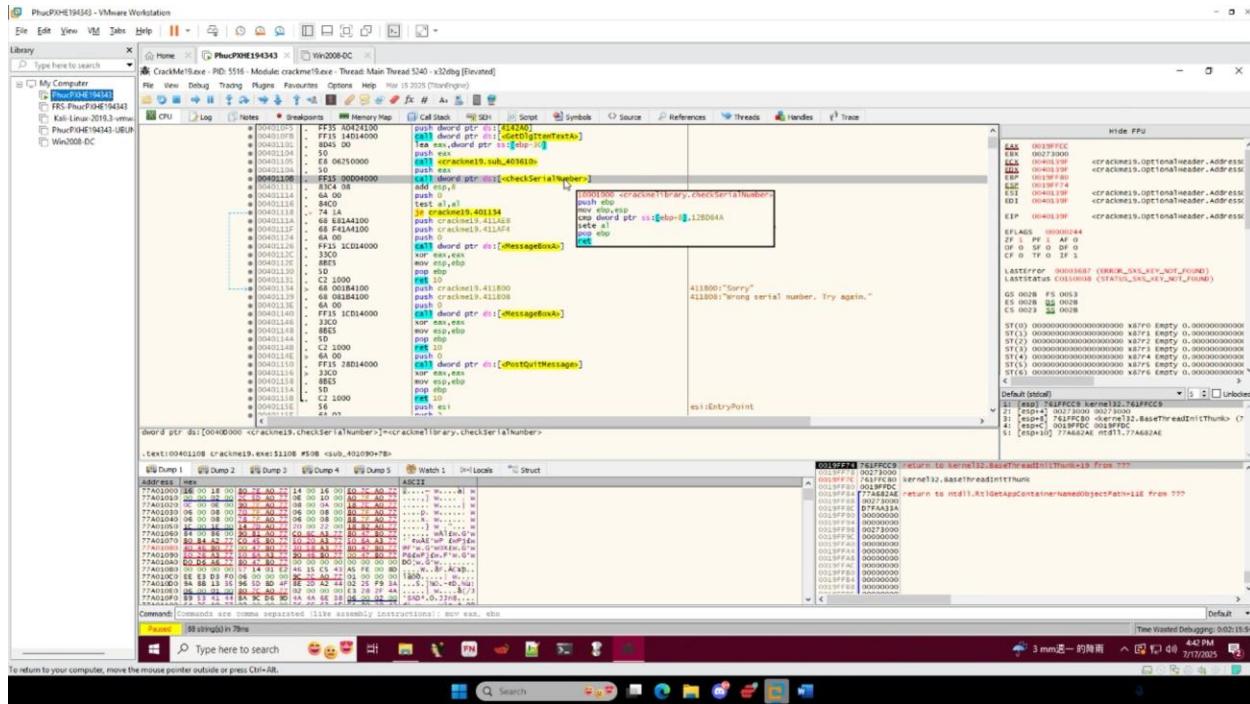


Step 3: Qua quá trình phân tích, có thể thấy chương trình sử dụng lệnh nhảy có điều kiện JE (Jump if Equal). Câu lệnh này kiểm tra điều kiện so sánh và, nếu điều kiện đúng (tức là hai

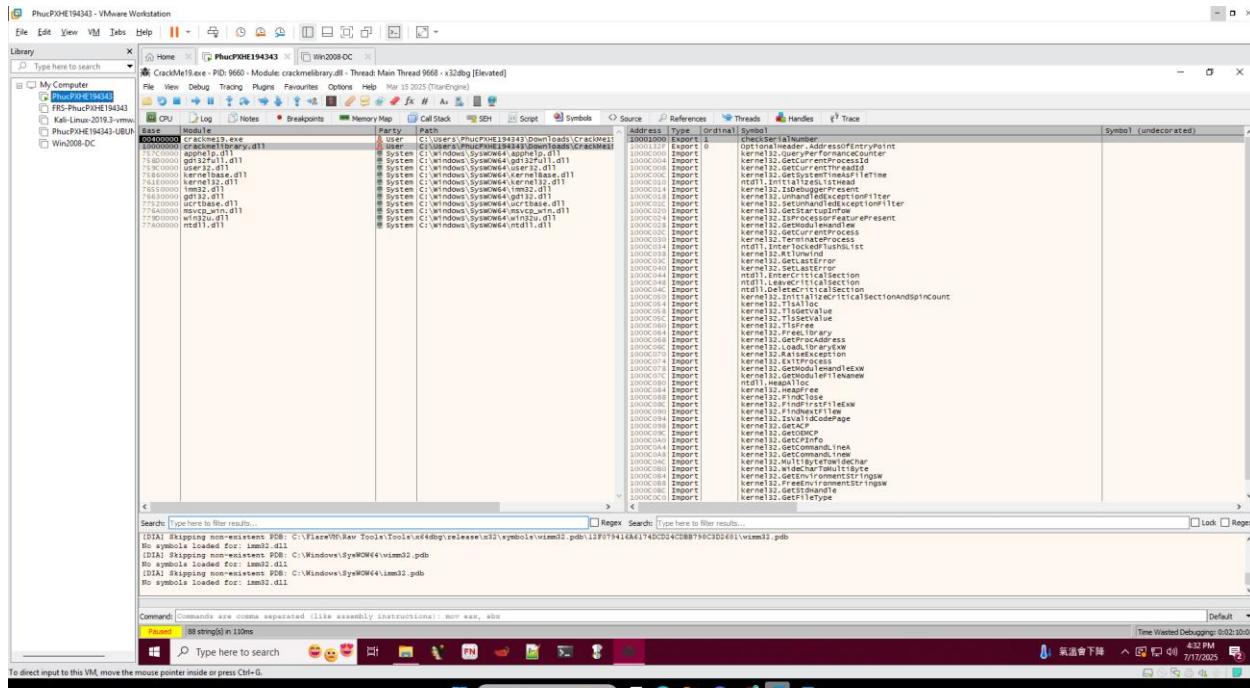
giá trị bằng nhau), chương trình sẽ thực hiện nhảy qua đoạn mã hiển thị thông báo thành công và chuyển trực tiếp đến thông báo lỗi. Điều này cho thấy kết quả kiểm tra Serial không khớp với giá trị hợp lệ."



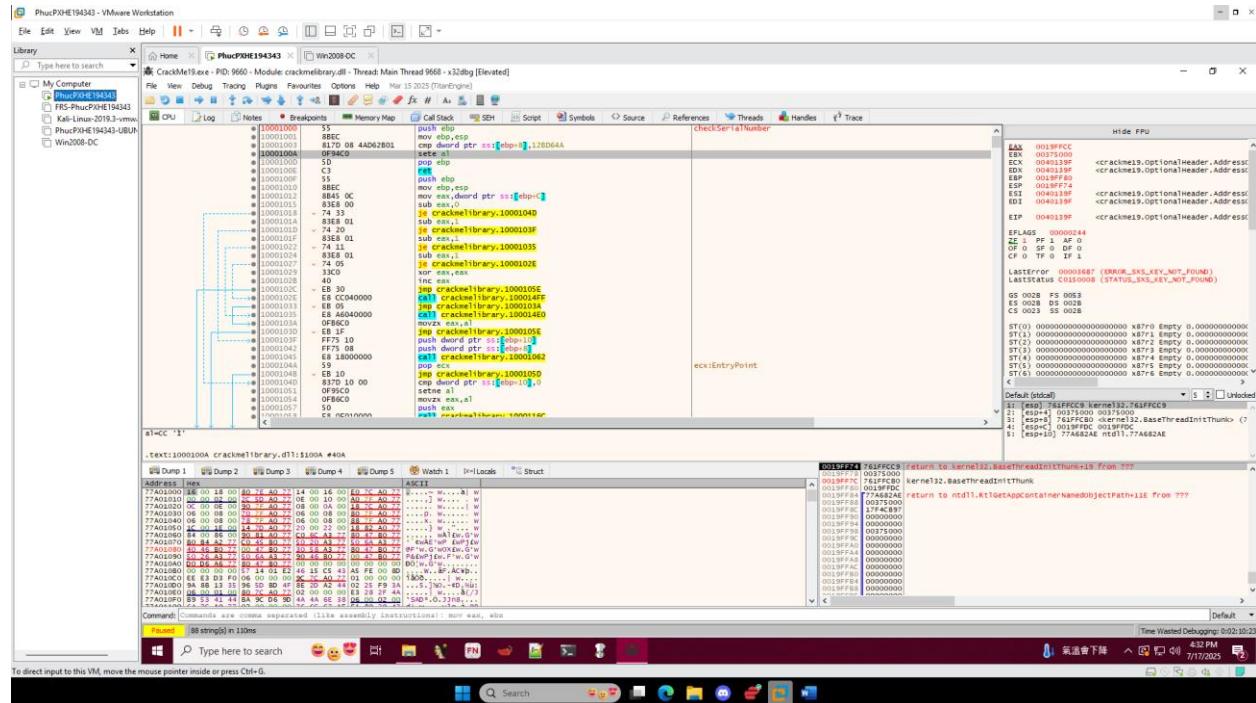
Step 4: Khi phân tích ngược đoạn mã, ta có thể thấy chương trình thực hiện lệnh test al, al, đây là thao tác logic dùng để kiểm tra giá trị của thanh ghi AL, tức là byte thấp nhất của thanh ghi EAX. Nếu AL bằng 0, cờ ZF (Zero Flag) sẽ được thiết lập, dẫn đến việc lệnh nhảy có điều kiện JE (Jump if Equal) được thực thi — nghĩa là chương trình sẽ bỏ qua phần xử lý thành công và chuyển đến thông báo lỗi. Khi truy vết ngược lên, ta nhận thấy rằng giá trị AL được thiết lập trong hàm CheckSerialNumber, cho thấy hàm này chịu trách nhiệm xác thực mã Serial mà người dùng nhập vào.



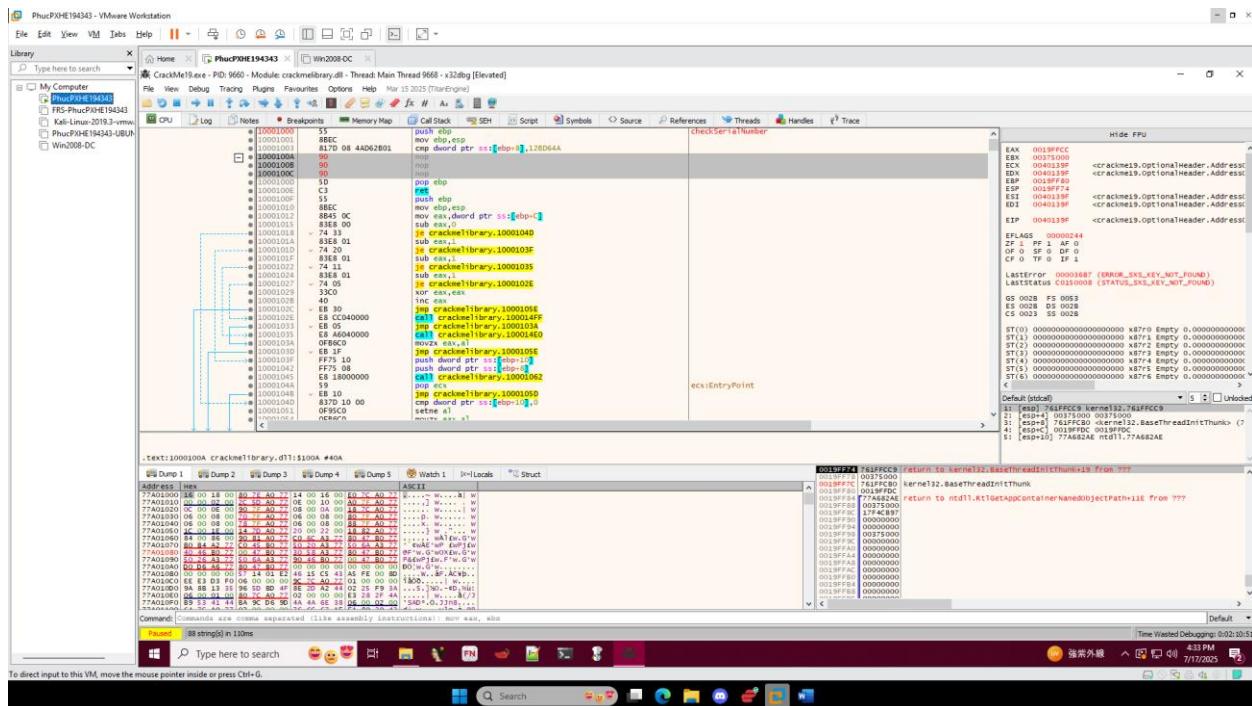
Step 5: Tiếp theo, khi kiểm tra phần Symbols — nơi cung cấp thông tin về tên hàm, biến và cấu trúc dữ liệu được chương trình sử dụng — ta có thể thấy rằng hàm CheckSerialNumber thực chất nằm trong một tệp thư viện động (DLL). Điều này cho thấy quá trình kiểm tra serial được thực hiện thông qua một hàm được import từ DLL, chứ không được định nghĩa trực tiếp trong file thực thi chính. Việc sử dụng DLL như vậy thường nhằm mục đích tái sử dụng mã hoặc để tách biệt logic xử lý nhạy cảm.



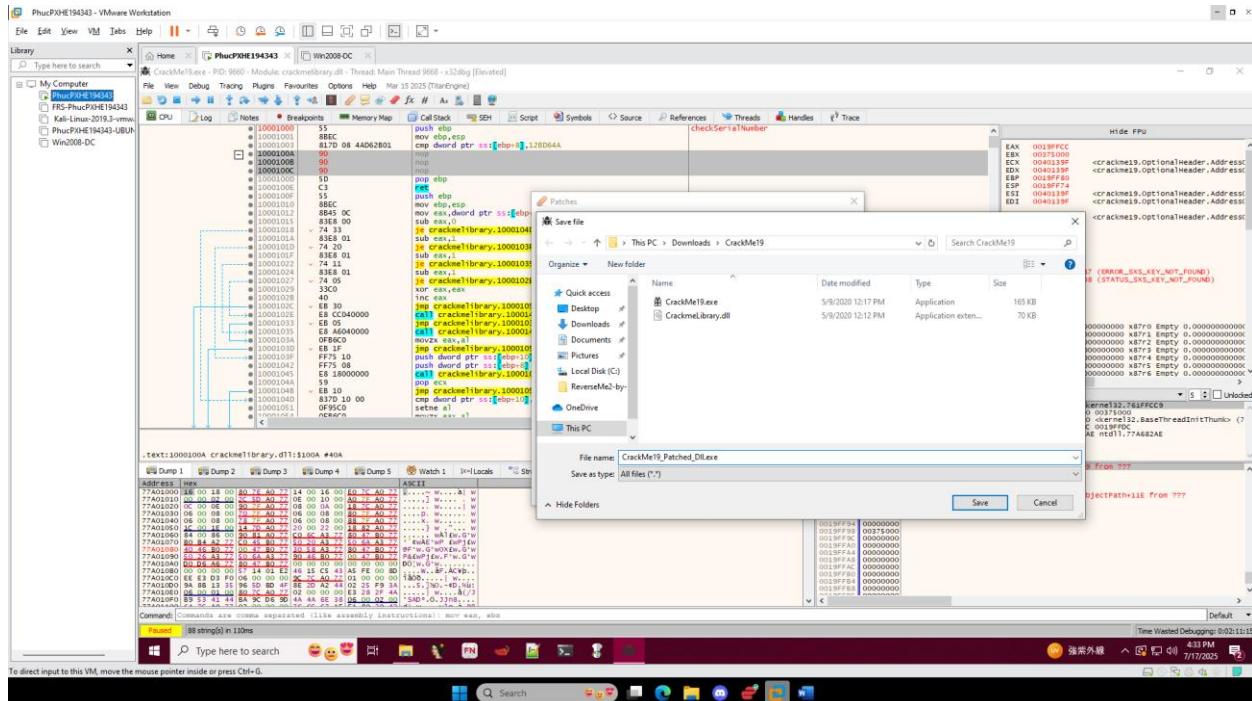
Step 6: Khi đi sâu vào phân tích hàm CheckSerialNumber từ danh sách Symbols, có thể quan sát thấy hàm này sử dụng lệnh so sánh cmp để đối chiếu hai giá trị. Sau đó, nó sử dụng lệnh sete al, đây là một lệnh đặc biệt trong tập lệnh x86, có chức năng gán giá trị cho thanh ghi AL dựa trên trạng thái của cờ ZF (Zero Flag). Cụ thể, nếu hai giá trị được so sánh là bằng nhau (ZF = 1), sete al sẽ gán AL = 1; ngược lại, nếu không bằng nhau (ZF = 0), AL sẽ được gán bằng 0. Do đó, trong trường hợp người dùng nhập sai Serial, cmp cho kết quả khác nhau, dẫn đến AL = 0, và kết quả này sẽ ảnh hưởng trực tiếp đến luồng điều khiển của chương trình.



Step 7: Do đó, để can thiệp vào quá trình xác thực và ngăn việc giá trị AL bị gán bằng 0 khi serial không đúng, ta có thể thực hiện kỹ thuật patching bằng cách thay thế lệnh sete al bằng một chuỗi lệnh NOP (No Operation). Việc chèn NOP sẽ loại bỏ chức năng gán giá trị cho thanh ghi AL, từ đó giữ nguyên trạng thái của AL và khiến chương trình có thể bỏ qua bước xác thực không hợp lệ. Đây là một phương pháp thường được sử dụng trong dịch ngược nhằm thay đổi hành vi kiểm tra điều kiện trong mã máy



Step 8: Tiến hành patch file



Step 9: Thử file đã patched, và đã thành công

