IMPORTANT: Please disable your windows defender or Anti-Malware software before conducting this exercise as they will determine the files as malware. I recommend you to create a Windows VM and do this exercise in it like I did in class.

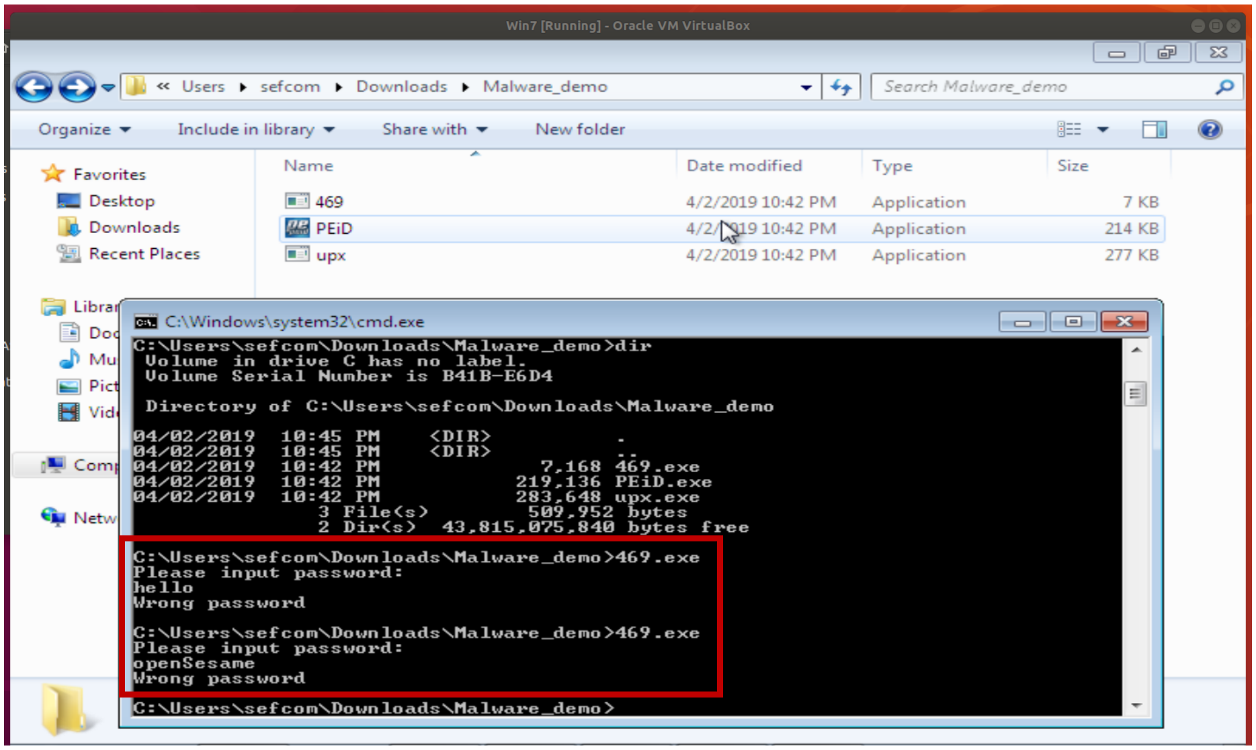
1. This exercise will walk you through the IDA pro demo. First, download the zip file named '469_malware_demo.zip'

2. The file contains 4 binaries --- **469, PEiD, upx**, and installer for **IDA pro 5**.

As I demonstrated in class, PEiD is used to read signatures in PE files and to determine whether the file is obfuscated using a packer.

3. '469.exe' is a program with encrypted criminal's bank account information. It is your job to retrieve them.

Let's first run it and see what it does.

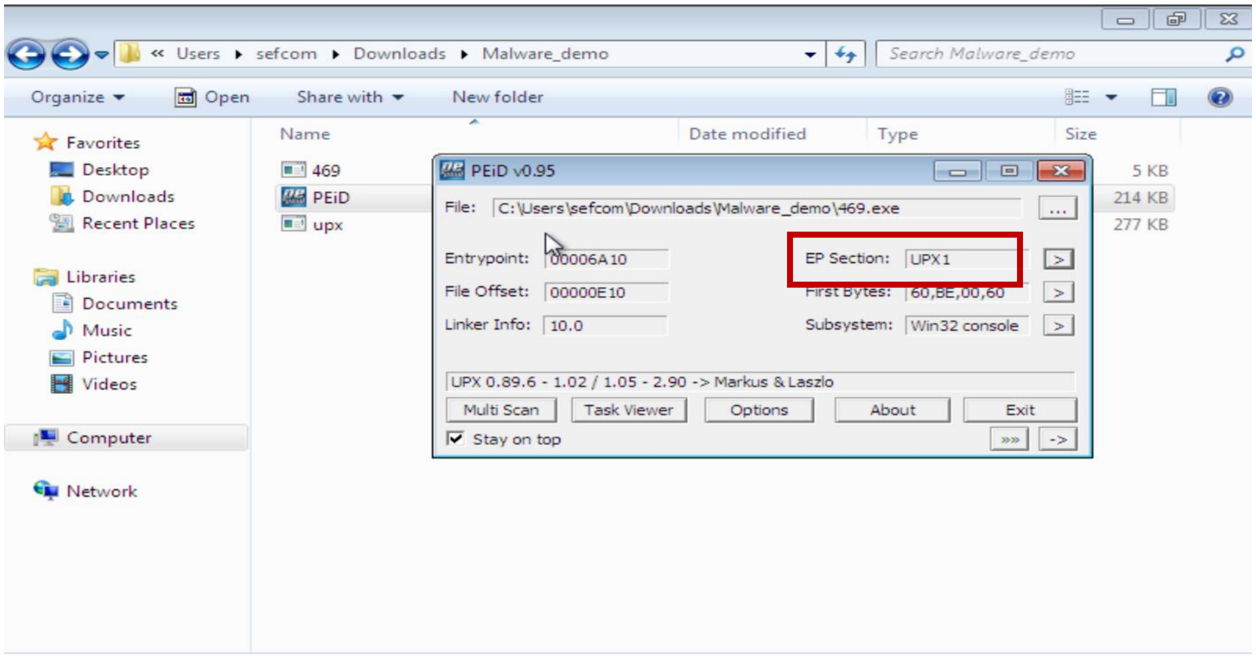

Win7 [Running] - Oracle VM VirtualBox
Users ▸ sefcom ▸ Downloads ▸ Malware_demo
Search Malware_demo
Organize
Include in library
Share with
New folder
Favorites
Desktop
Downloads
Recent Places
C:\Windows\system32\cmd.exe

Name	Date modified	Type	Size
469	4/2/2019 10:42 PM	Application	7 KB
PEiD	4/2/2019 10:42 PM	Application	214 KB
upx	4/2/2019 10:42 PM	Application	277 KB

```
C:\Users\sefcom\Downloads\Malware_demo>dir
 Volume in drive C has no label.
 Volume Serial Number is B41B-E6D4

 Directory of C:\Users\sefcom\Downloads\Malware_demo

04/02/2019  10:45 PM    <DIR>          .
04/02/2019  10:45 PM    <DIR>          ..
04/02/2019  10:42 PM             7,168 469.exe
04/02/2019  10:42 PM           219,136 PEiD.exe
04/02/2019  10:42 PM           283,648 upx.exe
               3 File(s)        509,952 bytes
               2 Dir(s)  43,815,075,840 bytes free

C:\Users\sefcom\Downloads\Malware_demo>469.exe
Please input password:
hello
Wrong password

C:\Users\sefcom\Downloads\Malware_demo>469.exe
Please input password:
openSesame
Wrong password

C:\Users\sefcom\Downloads\Malware_demo>
```

I tried two different passwords, **'hello'** and **'openSesame'**, but they are both wrong passwords.

4. Before we disassemble the executable and reveal the hidden information, let's use PEiD to check the file.



File: C:\Users\sefcom\Downloads\Malware_demo\469.exe

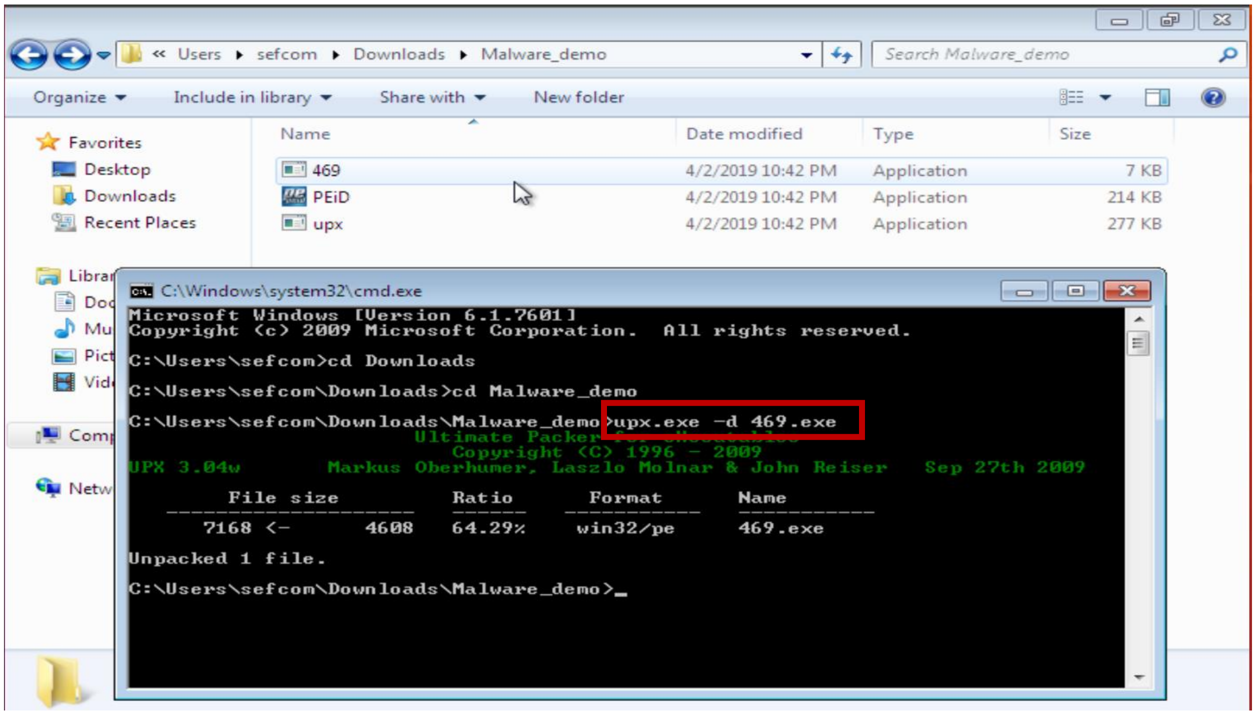
Entrypoint: 00006A10
File Offset: 00000E10
Linker Info: 10.0
EP Section: UPX1
First Bytes: 60,BE,00,60
Subsystem: Win32 console

UPX 0.89.6 - 1.02 / 1.05 - 2.90 -> Markus & Laszlo

5. Unlike the unpacked binary (i.e., ‘.text’ or ‘.rdata’), the EP Section indicates that it is packed by a program called **UPX**. It is a well-known packer for PE files that uses compression.

Let’s unpack it using the ‘upx.exe’ included in the folder.

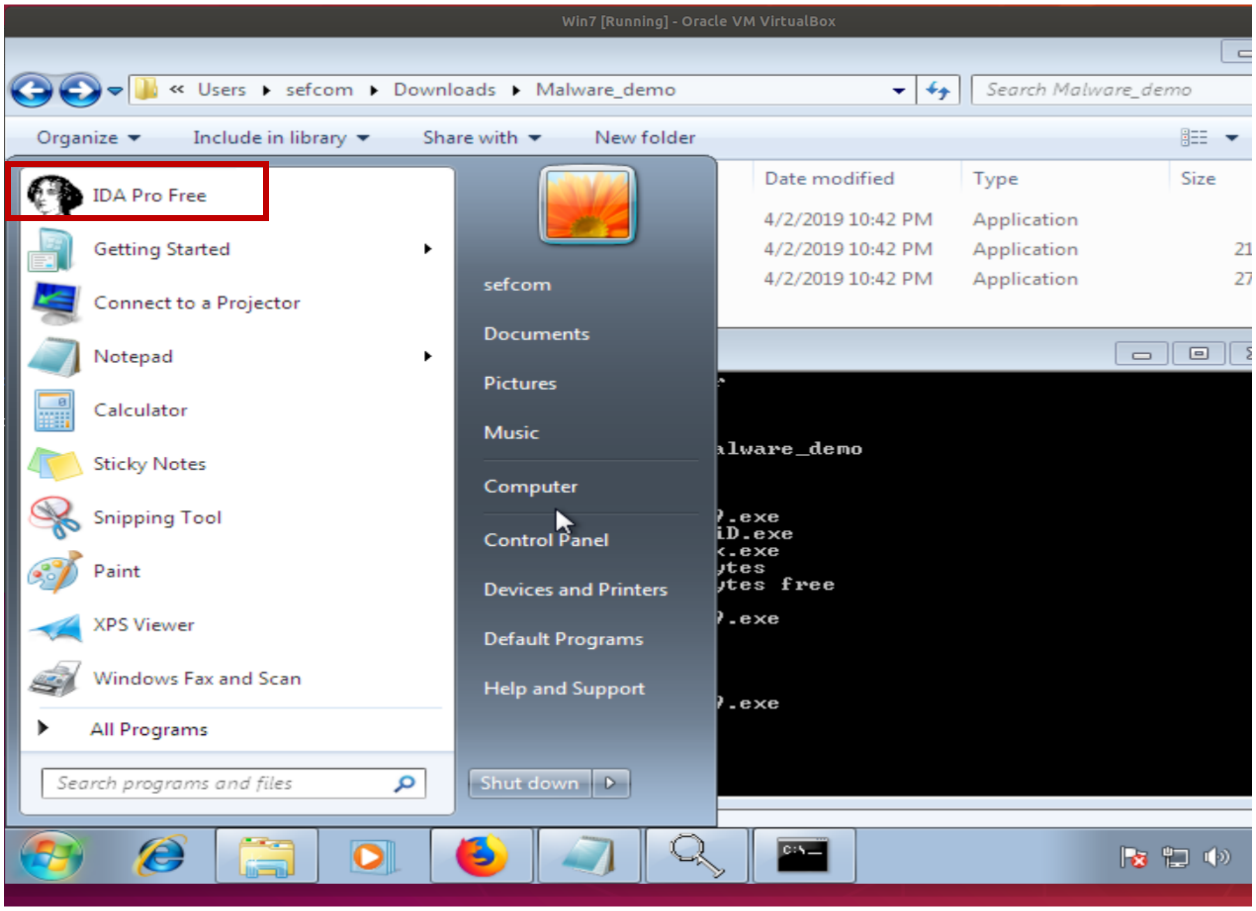
>> upx.exe -d 469.exe



Name	Date modified	Type	Size
469	4/2/2019 10:42 PM	Application	7 KB
PEiD	4/2/2019 10:42 PM	Application	214 KB
upx	4/2/2019 10:42 PM	Application	277 KB

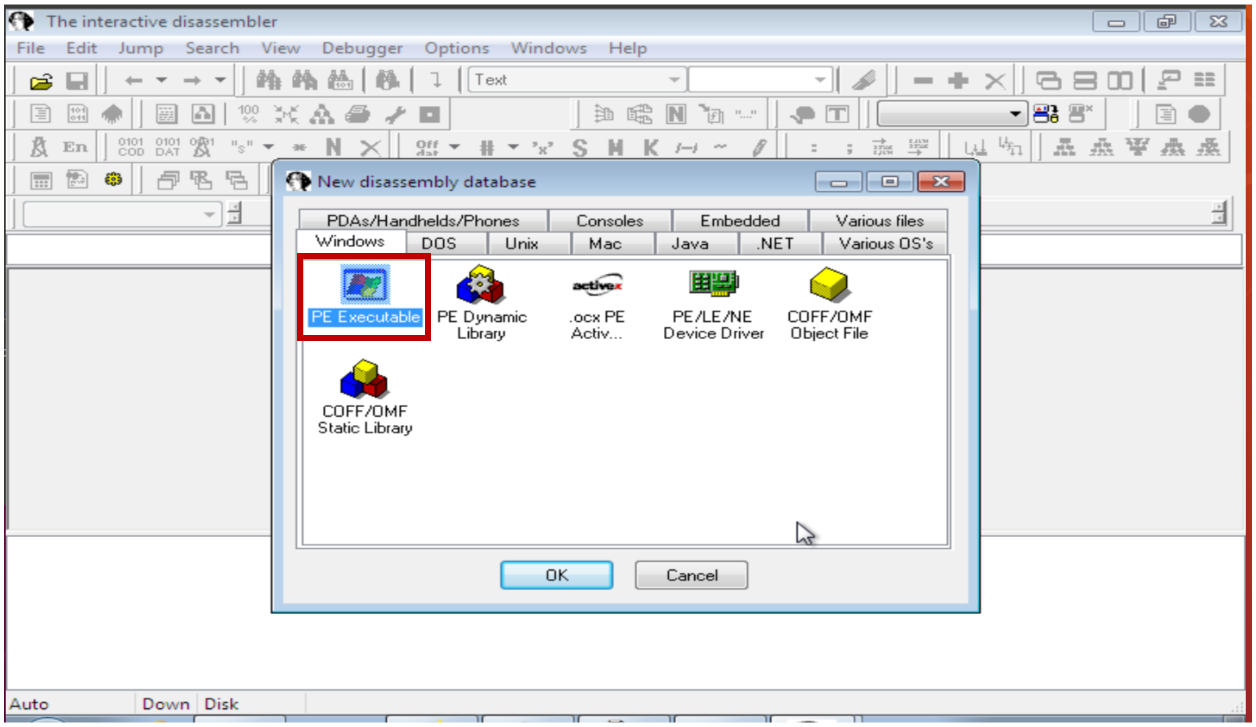
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\sefcom>cd Downloads

C:\Users\sefcom\Downloads>cd Malware_demo

C:\Users\sefcom\Downloads\Malware_demo>upx.exe -d 469.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2009
UPX 3.04w       Markus Oberhumer, Laszlo Molnar & John Reiser   Sep 27th 2009

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
      7168 <-      4608   64.29%   win32/pe      469.exe

Unpacked 1 file.

C:\Users\sefcom\Downloads\Malware_demo>_
```

6. Finally, disassemble the binary using IDA pro. Install IDA Pro using the installer included, or you can download the latest free version of IDA Pro from https://www.hex-rays.com/products/ida/



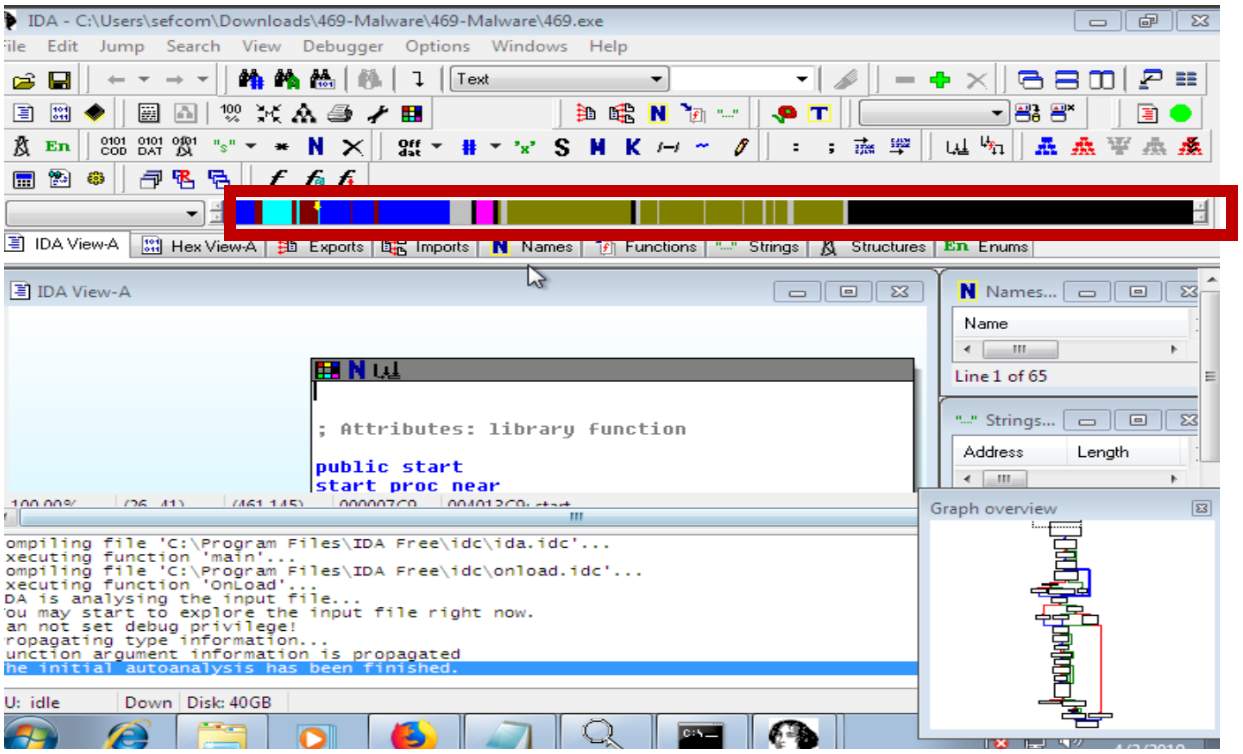
Date modified	Type	Size
4/2/2019 10:42 PM	Application	
4/2/2019 10:42 PM	Application	21
4/2/2019 10:42 PM	Application	27

```
alware_demo

?.exe
iD.exe
x.exe
ytes
ytes free
?.exe


?.exe
```



7. When it starts, select 'PE Executables' from the options under the 'Windows' tab as shown in the picture above.

Unless you want certain options only, run the analysis by clicking 'Next' buttons.



```
; Attributes: library function

public start
start proc near
```

```
ompiling file 'C:\Program Files\IDA Free\idc\ida.idc'...
xecuting function 'main'...
ompiling file 'C:\Program Files\IDA Free\idc\onload.idc'...
xecuting function 'OnLoad'...
DA is analysing the input file...
ou may start to explore the input file right now.
an not set debug privilege!
ropagating type information...
unction argument information is propagated
he initial autoanalysis has been finished.
```

8. You will see a screen like the picture above.

You can see various items from each window. For example:

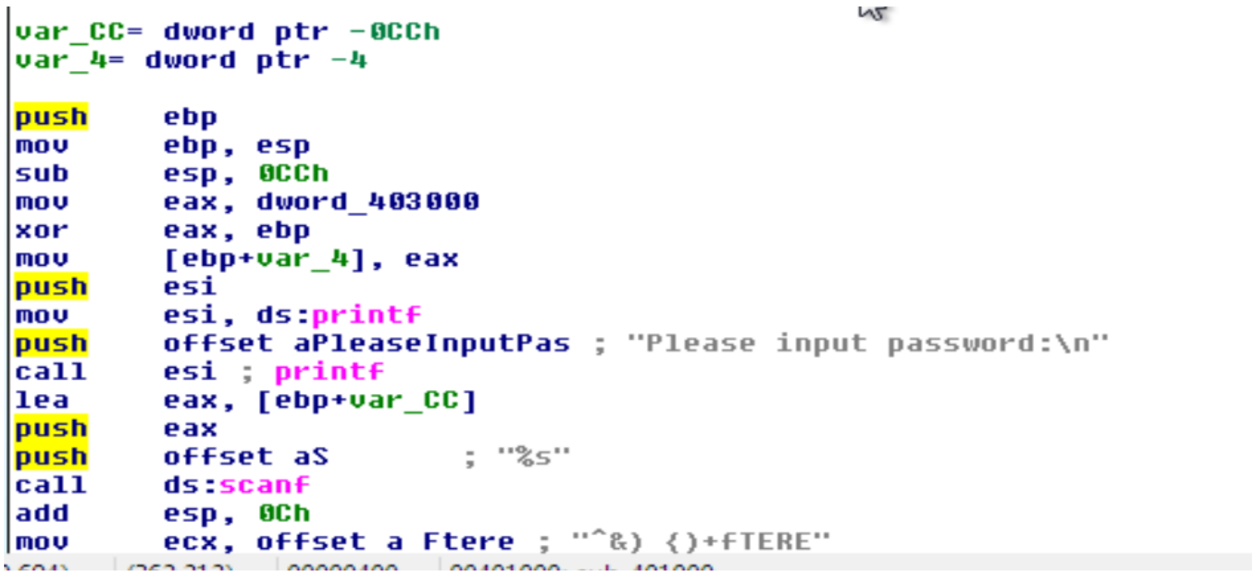
Names window: lists all known named locations in the program. F is a function, C is code/instruction, A is a string, D is defined data, etc.

Strings window: lists a list of embedded strings in the program.

IDA View Window: is a disassembly window that shows the program in assembly. It supports graphical view and text view modes. You can toggle between the two modes using the Space key.

Hex View Window: dumps the binary into hex.

Move to the beginning of the program by clicking the leftmost blue portion of the bar indicated in the picture above.



```
var_CC= dword ptr -0CCh
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 0CCh
mov     eax, dword_403000
xor     eax, ebp
mov     [ebp+var_4], eax
push    esi
mov     esi, ds:printf
push    offset aPleaseInputPas ; "Please input password:\n"
call    esi ; printf
lea     eax, [ebp+var_CC]
push    eax
push    offset aS       ; "%s"
call    ds:scanf
add     esp, 0Ch
mov     ecx, offset a_Ftere ; "^&) {}+FTERE"
```

9. You can see the first few instructions of the program.

As you can see the plain text “Please input password:\n” stored in the variable “aPleaseInputPas” from the picture, this part of the program is the portion simply loads the program to the stack and asks the user for a password.

Assembly instruction shown by IDA pro follows **“intel” syntax**:

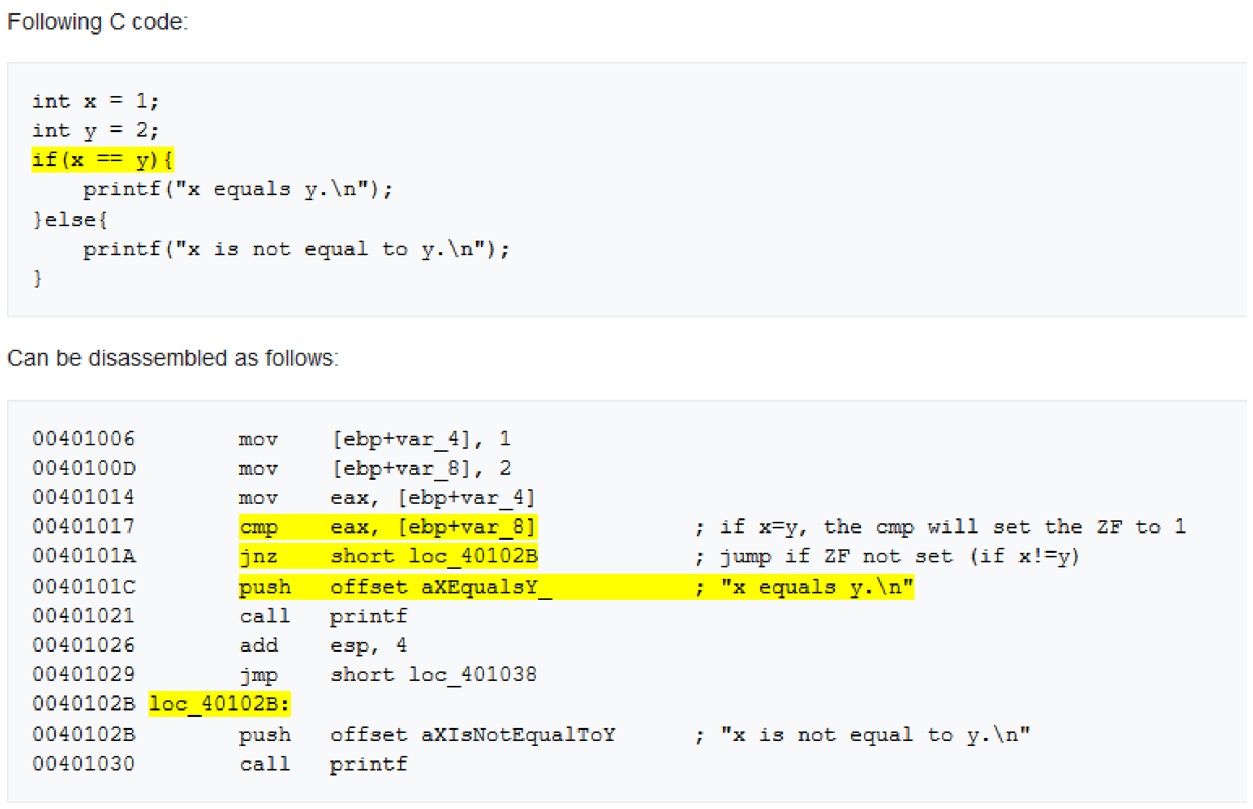
<instruction> <dst>, <src>: <instruction> is an operator, while <dst> and <src> are operands.

(e.g.) **mov ebp, esp**: move the value (or address) stored in current stack pointer (esp) to base pointer (ebp).

Both esp and ebp are ***registers***. Registers are tiny, internal memory of CPU. CPU utilizes registers to speed up the operations, instead of obtaining data from external memory through the control bus.

(e.g.) **cmp eax, [ebp + var]**: compare the value of [ebp + var] (the value of 'var' added to that of 'ebp) and the value of eax (accumulator register, which is used in arithmetic operation).

Another example is:



Following C code:

```
int x = 1;
int y = 2;
if(x == y){
    printf("x equals y.\n");
}else{
    printf("x is not equal to y.\n");
}
```

Can be disassembled as follows:

```
00401006        mov     [ebp+var_4], 1
0040100D        mov     [ebp+var_8], 2
00401014        mov     eax, [ebp+var_4]
00401017        cmp     eax, [ebp+var_8]            ; if x=y, the cmp will set the ZF to 1
0040101A        jnz     short loc_40102B            ; jump if ZF not set (if x!=y)
0040101C        push    offset aXEqualsY_           ; "x equals y.\n"
00401021        call    printf
00401026        add     esp, 4
00401029        jmp     short loc_401038
0040102B loc_40102B:
0040102B        push    offset aXIsNotEqualToY      ; "x is not equal to y.\n"
00401030        call    printf
```

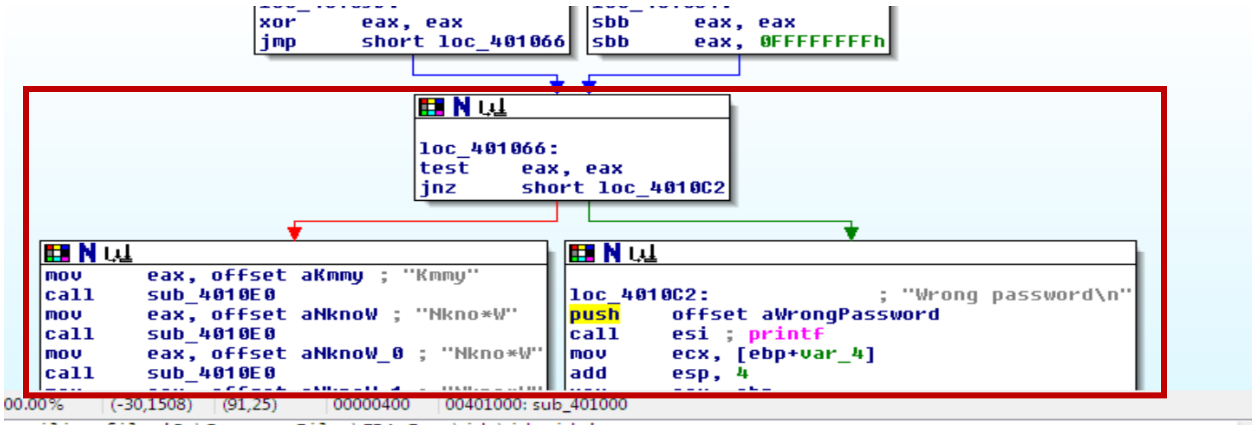
For more tutorials on assembly and IDA pro, please refer to the following links:

http://www-verimag.imag.fr/~mounier/Enseignement/Software_Security/BH_Eagle_ida_pro.pdf

https://www.felixcloutier.com/x86/

https://www.tutorialspoint.com/assembly_programming/index.htm

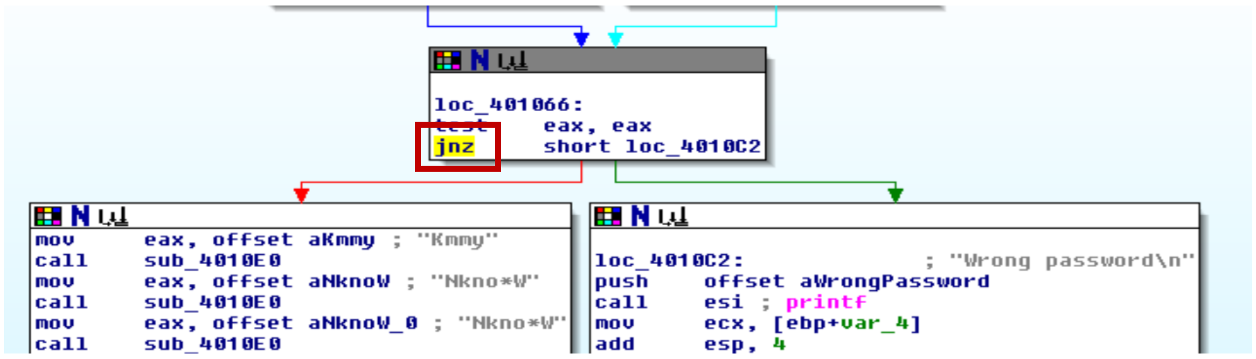
Please stop following the instructions of this document and play around with IDA pro and try to understand the assembly and have fun☺



```
xor     eax, eax                    sbb     eax, eax
jmp     short loc_401066            sbb     eax, 0FFFFFFFFh

loc_401066:
test    eax, eax
jnz     short loc_4010C2

mov     eax, offset aKmmy ; "Kmmy"          loc_4010C2:                 ; "Wrong password\n"
call    sub_4010E0                          push    offset aWrongPassword
mov     eax, offset aNknoW ; "Nkno*W"       call    esi ; printf
call    sub_4010E0                          mov     ecx, [ebp+var_4]
mov     eax, offset aNknoW_0 ; "Nkno*W"     add     esp, 4
call    sub_4010E0
```

```
00.00%   (-30,1508)   (91,25)   00000400   00401000: sub_401000
```

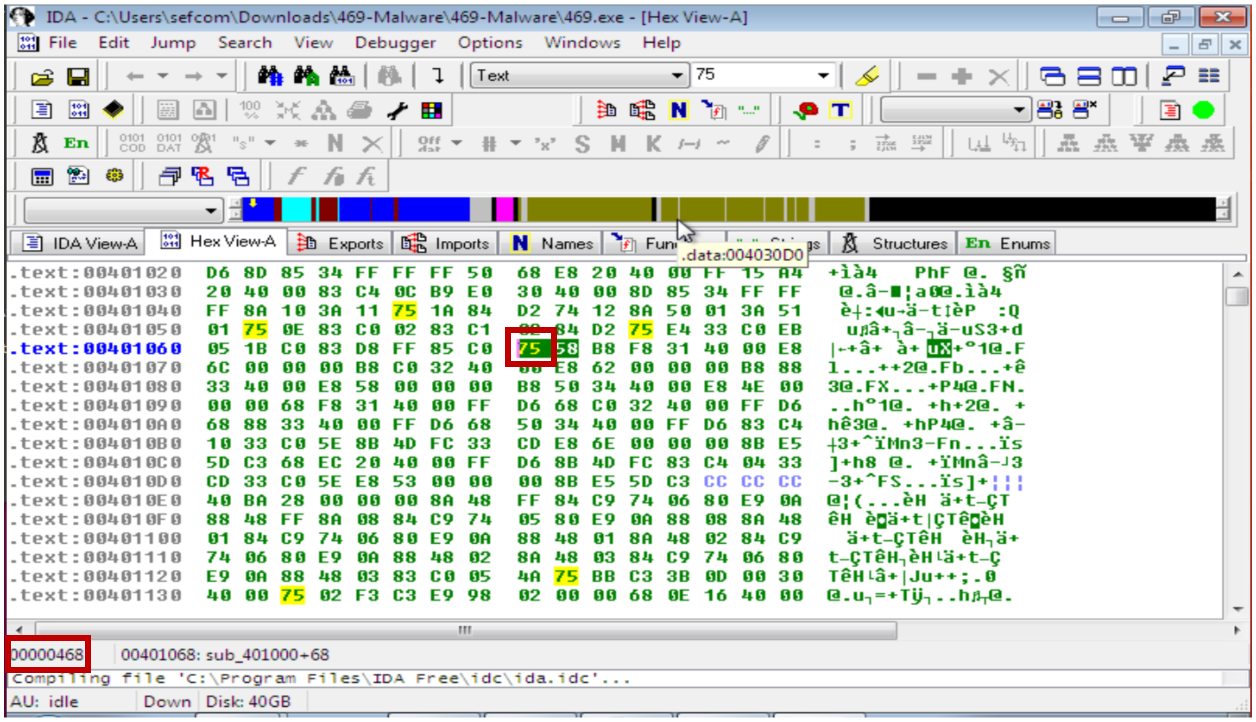
9. If you scroll down the assembly view window, you will find the branch that determines which location of the program the control should jump to. It is determined by the two lines in the top box --- "**test** eax, eax" and '**jnz** short loc_4010C2".



```
loc_401066:
test    eax, eax
jnz     short loc_4010C2

mov     eax, offset aKmmy ; "Kmmy"          loc_4010C2:                 ; "Wrong password\n"
call    sub_4010E0                          push    offset aWrongPassword
mov     eax, offset aNknoW ; "Nkno*W"       call    esi ; printf
call    sub_4010E0                          mov     ecx, [ebp+var_4]
mov     eax, offset aNknoW_0 ; "Nkno*W"     add     esp, 4
call    sub_4010E0
```

10. jnz instruction means "*jump if not zero*". If the *Zero Flag (ZF)* is set to zero, it jumps to the left block, which prints the hidden information, while the block on the right prints out "Wrong password" and exits the program as we saw earlier.

Click or select the instruction "jnz" and it will show the byte offset and the hex code of the instruction in the "Hex View" window.


IDA - C:\Users\sefcom\Downloads\469-Malware\469-Malware\469.exe - [Hex View-A]

```
.text:00401020  D6 8D 85 34 FF FF FF 50  68 E8 20 40 00 FF 15 A4  +ìà4   PhF @. §ñ
.text:00401030  20 40 00 83 C4 0C B9 E0  30 40 00 8D 85 34 FF FF   @.â-■¦a0@.ìà4
.text:00401040  FF 8A 10 3A 11 75 1A 84  D2 74 12 8A 50 01 3A 51   è+:◄u→ä-t↕èP  :Q
.text:00401050  01 75 0E 83 C0 02 83 C1  02 84 D2 75 E4 33 C0 EB   u♫â+┐â-┐ä-uS3+d
.text:00401060  05 1B C0 83 D8 FF 85 C0  75 58 B8 F8 31 40 00 E8  |←+â+ à+ uX+°1@.F
.text:00401070  6C 00 00 00 B8 C0 32 40  00 E8 62 00 00 00 B8 88  l...++2@.Fb...+ê
.text:00401080  33 40 00 E8 58 00 00 00  B8 50 34 40 00 E8 4E 00  3@.FX...+P4@.FN.
.text:00401090  00 00 68 F8 31 40 00 FF  D6 68 C0 32 40 00 FF D6  ..h°1@. +h+2@. +
.text:004010A0  68 88 33 40 00 FF D6 68  50 34 40 00 FF D6 83 C4  hê3@. +hP4@. +â-
.text:004010B0  10 33 C0 5E 8B 4D FC 33  CD E8 6E 00 00 00 8B E5  ┼3+^ïMn3-Fn...ïs
.text:004010C0  5D C3 68 EC 20 40 00 FF  D6 8B 4D FC 83 C4 04 33  ]+h8 @. +ïMnâ-┘3
.text:004010D0  CD 33 C0 5E E8 53 00 00  00 8B E5 5D C3 CC CC CC  -3+^FS...ïs]+¦¦¦
.text:004010E0  40 BA 28 00 00 00 8A 48  FF 84 C9 74 06 80 E9 0A  @¦(...èH ä+t_ÇT
.text:004010F0  88 48 FF 8A 08 84 C9 74  05 80 E9 0A 88 08 8A 48  êH è◘ä+t|ÇTê◘èH
.text:00401100  01 84 C9 74 06 80 E9 0A  88 48 01 8A 48 02 84 C9   ä+t_ÇTêH  èH┐ä+
.text:00401110  74 06 80 E9 0A 88 48 02  8A 48 03 84 C9 74 06 80  t_ÇTêH┐èH└ä+t_Ç
.text:00401120  E9 0A 88 48 03 83 C0 05  4A 75 BB C3 3B 0D 00 30  TêH└â+|Ju++;.0
.text:00401130  40 00 75 02 F3 C3 E9 98  02 00 00 68 0E 16 40 00  @.u┐=+Tÿ┐..h♫┐@.
```

.data:004030D0

00000468 | 00401068: sub_401000+68

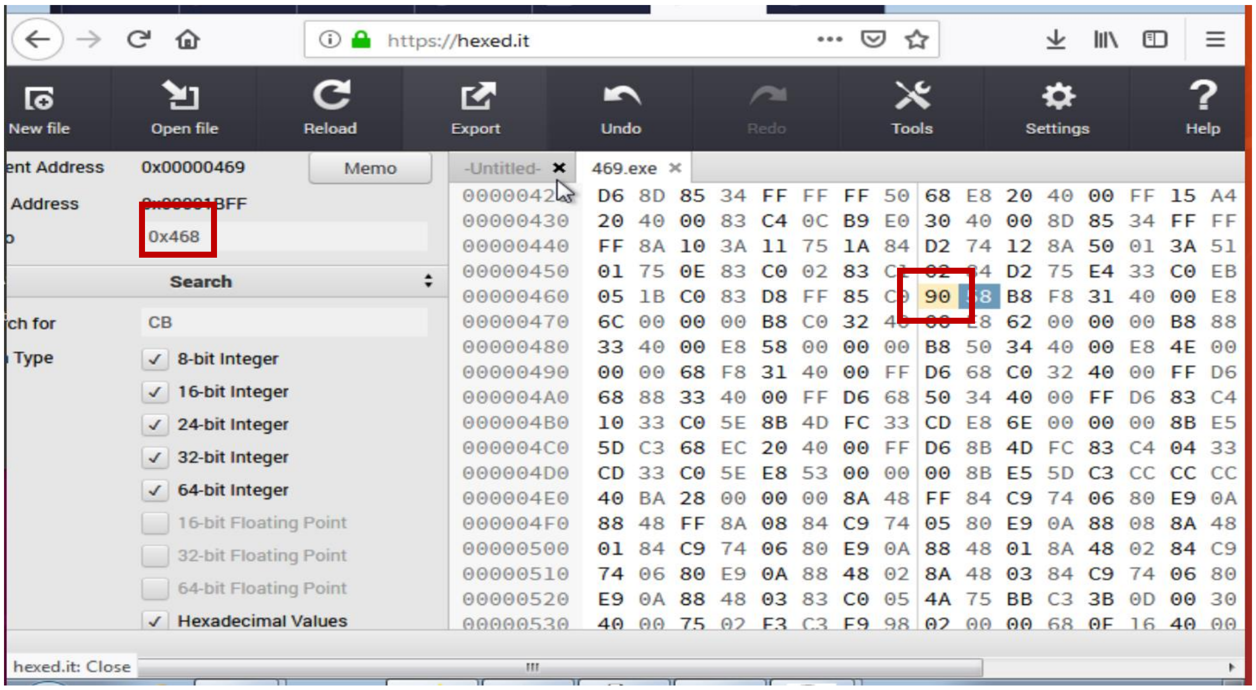
```
Compiling file 'C:\Program Files\IDA Free\idc\ida.idc'...
```

AU: idle | Down | Disk: 40GB

11. Assembly Window and Hex View Window are by default synced, so the instruction you selected in the Assembly Window should be highlighted in the Hex View Window, along with the byte offset in hex of the selected instruction at the bottom left corner (see the picture above).

The hex code for assembly 'jnz' is **75**.

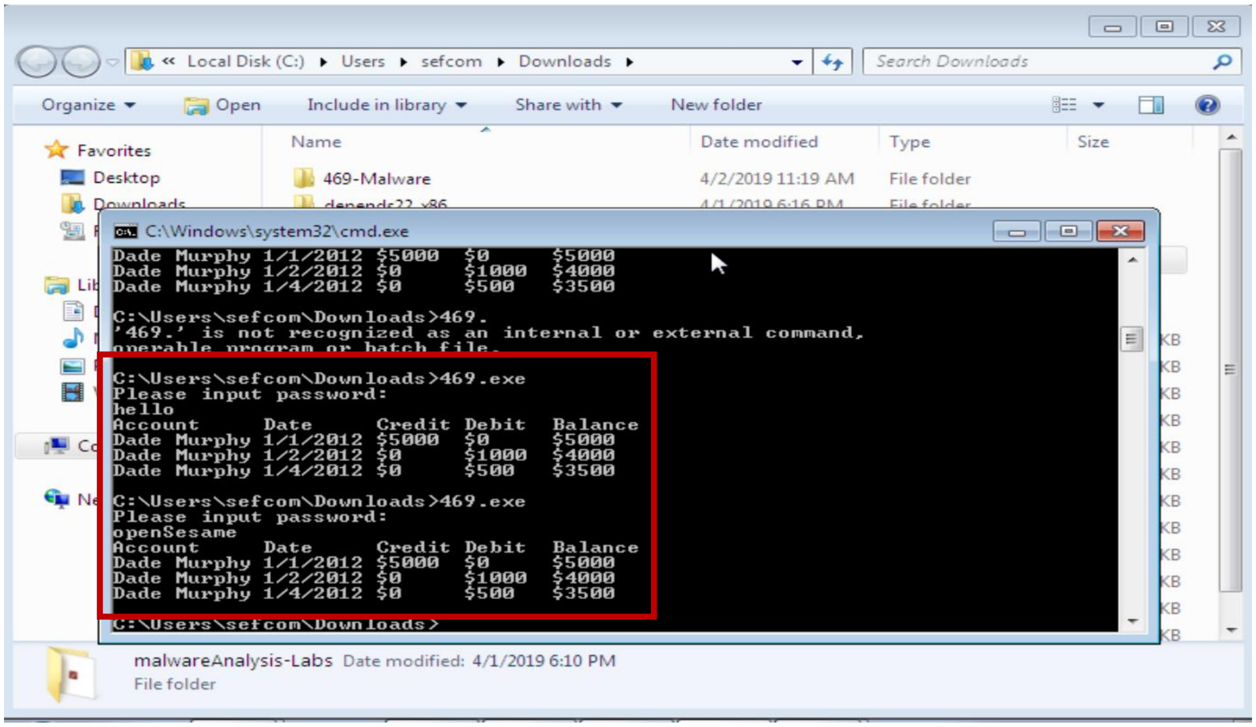
Let's change it to "**nop**" instruction (**No Operation**) so it never jump to the part printing out "Wrong password".



```
ent Address   0x00000469
Address       0x00001BFF
              0x468

ch for        CB
Type          ✓ 8-bit Integer
              ✓ 16-bit Integer
              ✓ 24-bit Integer
              ✓ 32-bit Integer
              ✓ 64-bit Integer
                16-bit Floating Point
                32-bit Floating Point
                64-bit Floating Point
              ✓ Hexadecimal Values
```

```
0000042   D6 8D 85 34 FF FF FF 50 68 E8 20 40 00 FF 15 A4
00000430  20 40 00 83 C4 0C B9 E0 30 40 00 8D 85 34 FF FF
00000440  FF 8A 10 3A 11 75 1A 84 D2 74 12 8A 50 01 3A 51
00000450  01 75 0E 83 C0 02 83 C1 02 84 D2 75 E4 33 C0 EB
00000460  05 1B C0 83 D8 FF 85 C0 90 58 B8 F8 31 40 00 E8
00000470  6C 00 00 00 B8 C0 32 40 00 E8 62 00 00 00 B8 88
00000480  33 40 00 E8 58 00 00 00 B8 50 34 40 00 E8 4E 00
00000490  00 00 68 F8 31 40 00 FF D6 68 C0 32 40 00 FF D6
000004A0  68 88 33 40 00 FF D6 68 50 34 40 00 FF D6 83 C4
000004B0  10 33 C0 5E 8B 4D FC 33 CD E8 6E 00 00 00 8B E5
000004C0  5D C3 68 EC 20 40 00 FF D6 8B 4D FC 83 C4 04 33
000004D0  CD 33 C0 5E E8 53 00 00 00 8B E5 5D C3 CC CC CC
000004E0  40 BA 28 00 00 00 8A 48 FF 84 C9 74 06 80 E9 0A
000004F0  88 48 FF 8A 08 84 C9 74 05 80 E9 0A 88 08 8A 48
00000500  01 84 C9 74 06 80 E9 0A 88 48 01 8A 48 02 84 C9
00000510  74 06 80 E9 0A 88 48 02 8A 48 03 84 C9 74 06 80
00000520  E9 0A 88 48 03 83 C0 05 4A 75 BB C3 3B 0D 00 30
00000530  40 00 75 02 F3 C3 E9 98 02 00 00 68 0F 16 40 00
```

12. Open the "469.exe" in https://hexed.it to modify. Change the instruction code of "jnz" (75) at byte offset 0x468 to "nop" (**90**).



```
C:\Windows\system32\cmd.exe
Dade Murphy 1/1/2012 $5000  $0     $5000
Dade Murphy 1/2/2012 $0     $1000  $4000
Dade Murphy 1/4/2012 $0     $500   $3500

C:\Users\sefcom\Downloads>469.
'469.' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\sefcom\Downloads>469.exe
Please input password:
hello
Account     Date     Credit Debit  Balance
Dade Murphy 1/1/2012 $5000  $0     $5000
Dade Murphy 1/2/2012 $0     $1000  $4000
Dade Murphy 1/4/2012 $0     $500   $3500

C:\Users\sefcom\Downloads>469.exe
Please input password:
openSesame
Account     Date     Credit Debit  Balance
Dade Murphy 1/1/2012 $5000  $0     $5000
Dade Murphy 1/2/2012 $0     $1000  $4000
Dade Murphy 1/4/2012 $0     $500   $3500

C:\Users\sefcom\Downloads>
```

13. Download the modified program and run it.

I used the same passwords (“hello” and “openSesame”) again and the program shows me the account information!

There are at least two more ways to get the hidden information by modifying the binary:

1. You can change the very first instruction of the program to jump to the location of the hidden information so the program does not even ask you for the password.
2. Change the “cmp” instructions that checks the password to “nop” so the program does not check the password.

Please try various methods to achieve the same goal.