

TASK 3

Cracking a Password

Rev A

Cracking a Password

- 1 Comparing Binaries Using Relyze Error! Bookmark not defined.
- 2 Crack and Decrypt the Password..... Error! Bookmark not defined.
- 3 Converting From HEX to ASCII Error! Bookmark not defined.
- 4 Run the Program..... Error! Bookmark not defined.
- 5 Connect to the IRC Channel..... Error! Bookmark not defined.
 - 5.1 Open PuTTY..... Error! Bookmark not defined.
 - 5.1.1 Input Host Name..... Error! Bookmark not defined.
 - 5.1.2 Input Username..... Error! Bookmark not defined.
 - 5.1.3 Input Password Error! Bookmark not defined.
 - 5.2 Input irssi..... Error! Bookmark not defined.
 - 5.2.1 Set Nickname Error! Bookmark not defined.
 - 5.2.2 Request Network List..... Error! Bookmark not defined.
 - 5.2.3 Connect to network (somber) Error! Bookmark not defined.
 - 5.2.4 Join #nymeria..... Error! Bookmark not defined.
 - 5.3 Observe conversation Error! Bookmark not defined.
 - 5.3.1 Identify Other Players Error! Bookmark not defined.
 - 5.3.2 See Who the Players Are Error! Bookmark not defined.

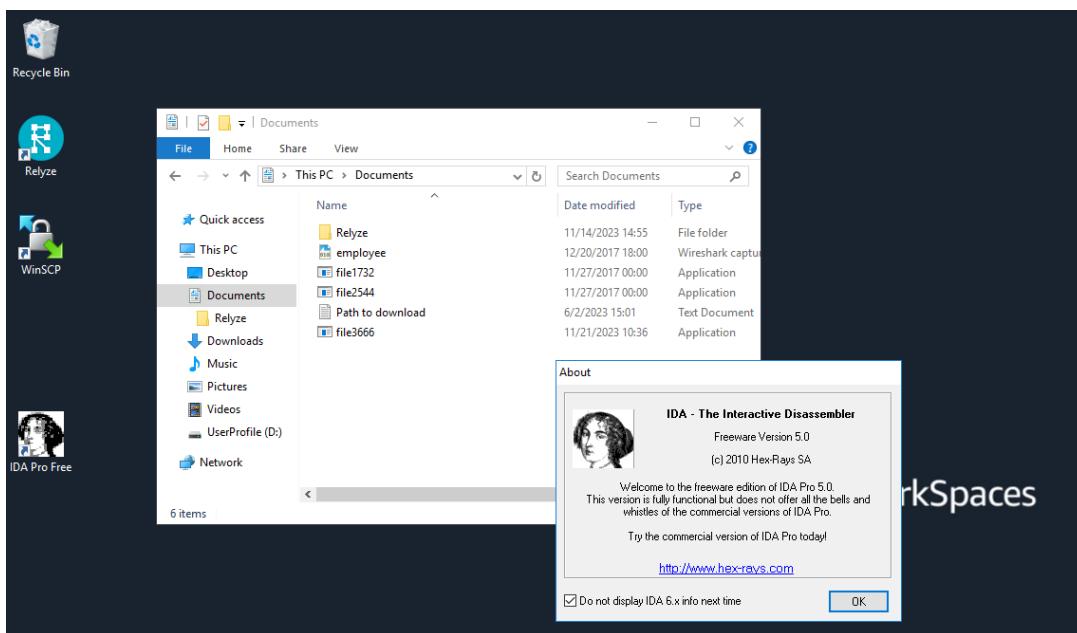
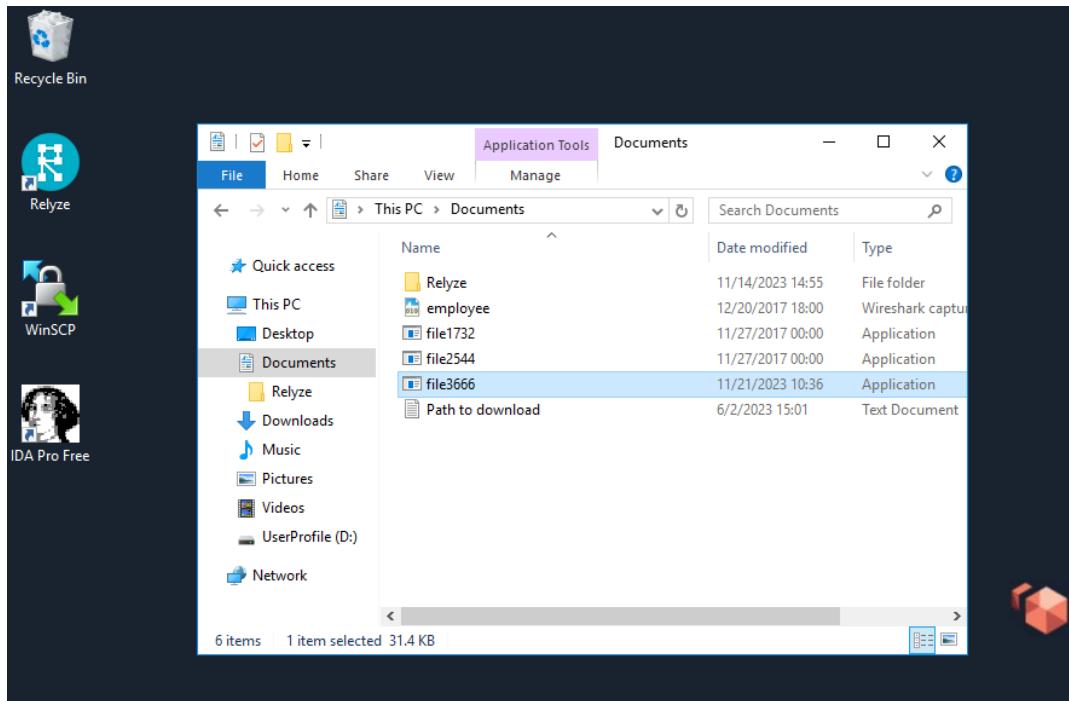
Cracking a Password

1 Opening a File in IDA Pro

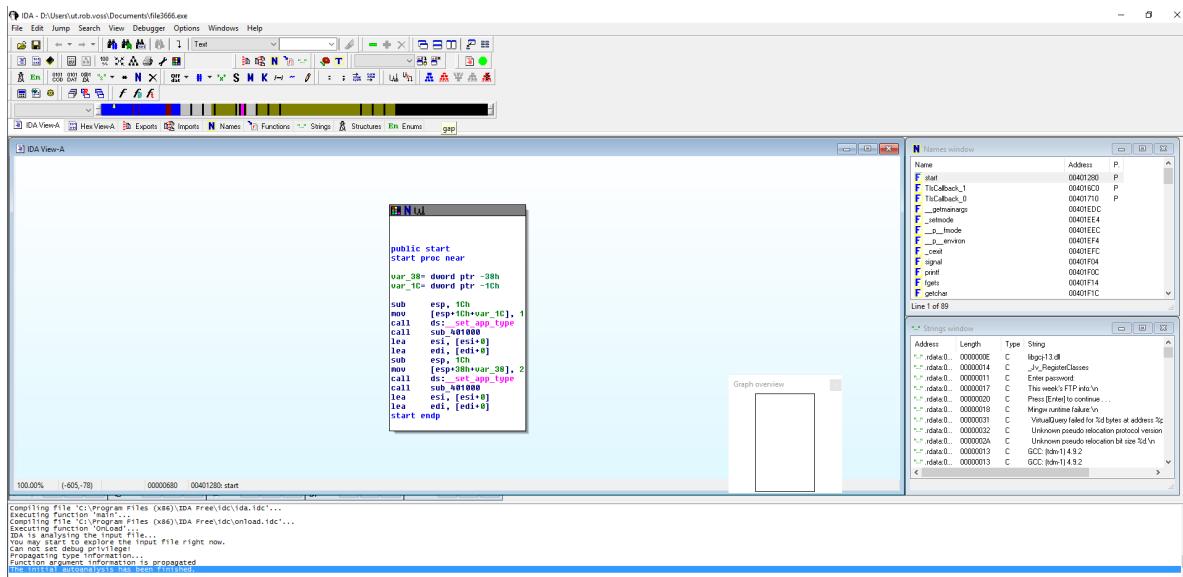
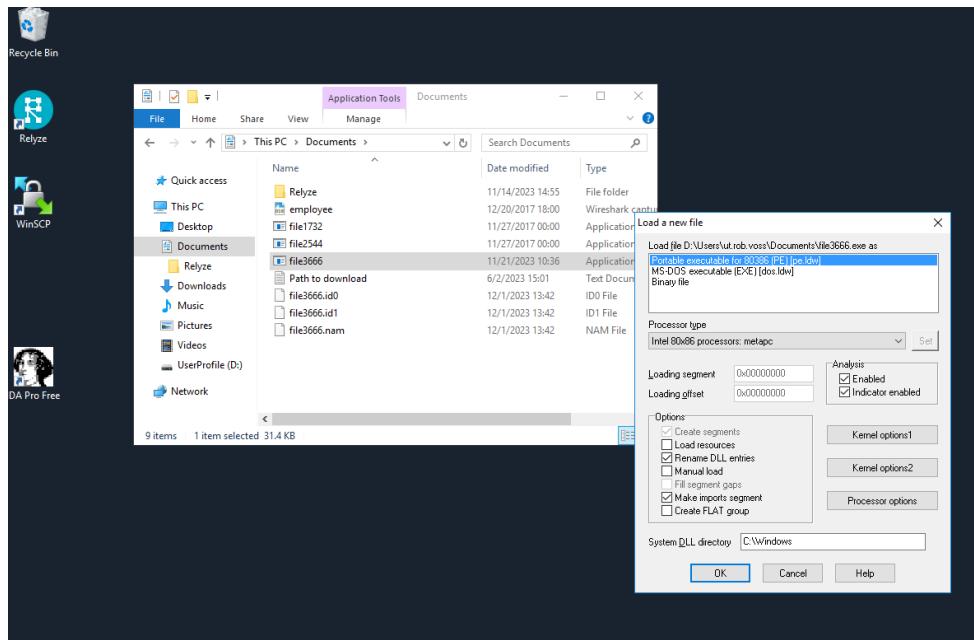
Open file 3666 in IDA Pro

Select file and drag and drop on top of IDA Pro

Click ok



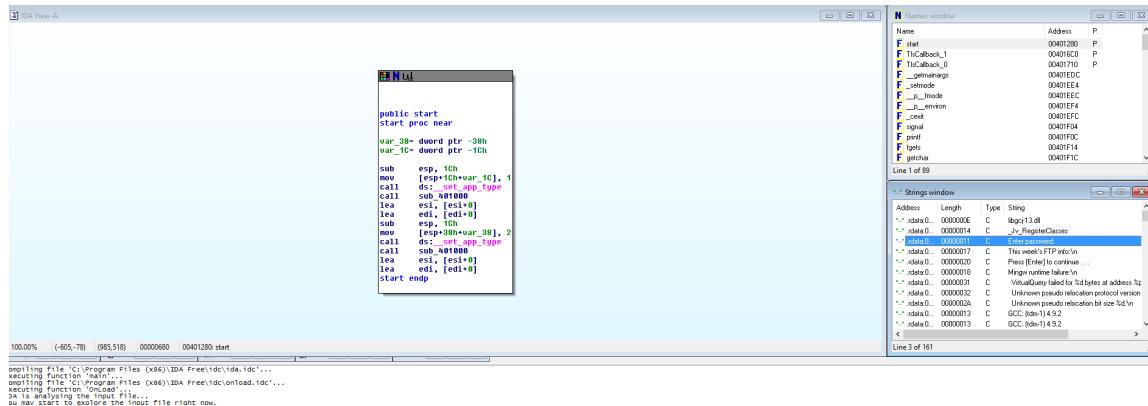
Cracking a Password



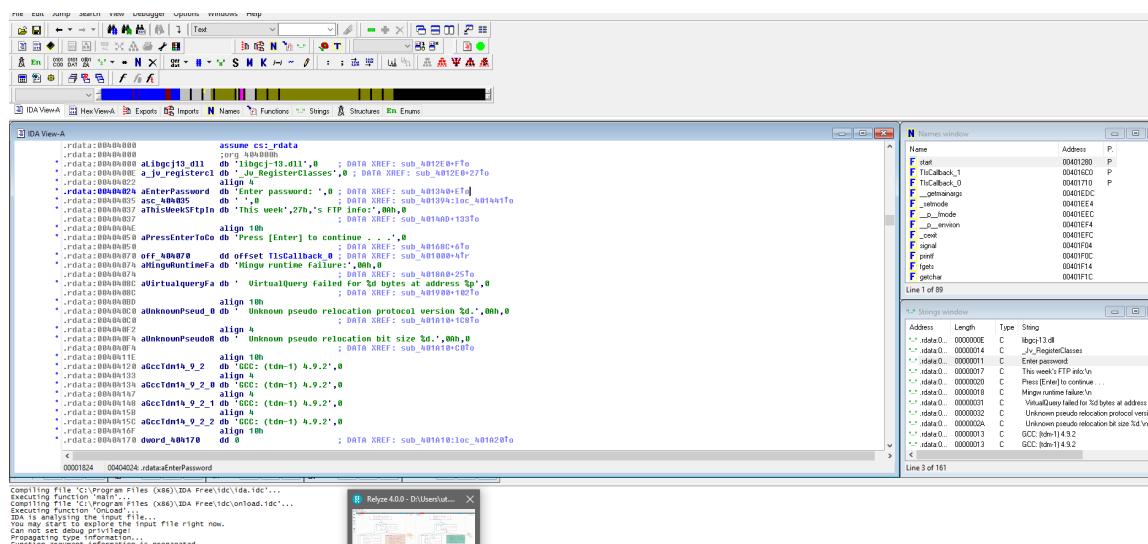
Cracking a Password

2 Renaming Subroutines

Starting from the Strings window, locate the subroutine that asks for a password.
(This is where the flow of control for this program starts.)



Double click on sub routine to have it load into the View a window



Cracking a Password

Find the sub routine and double click on it...

The screenshot shows the IDA Pro interface with several windows open:

- IDA View-A:** Displays assembly code for a function. The code includes instructions like `assume cs:rdata`, `sub_401340`, and `sub_401340+10h`. It contains comments such as "Unknown pseudo relocation protocol version 3d", "Unknown pseudo relocation bit size 3d", and "Unknown pseudo relocation bit size 3d".
- Names window:** Shows a list of symbols with their addresses and types.
- Strings window:** Shows a list of strings with their addresses and lengths.
- Registers window:** Shows CPU registers and memory dump options.
- Stack window:** Shows the stack contents.
- Call graph window:** Shows the call graph for the current function.
- File menu:** Contains options like File, Edit, Jump, Search, View, Debugger, Options, Windows, Help.

Compiler output at the bottom:

```
Compiling file 'C:\Program Files (x86)\IDA Free\idc\ida.idc'...
Compiling file 'C:\Program Files (x86)\IDA Free\idc\onload.idc'...
Executing function _onLoad_411...
```

The screenshot shows the IDA Pro interface with several windows open:

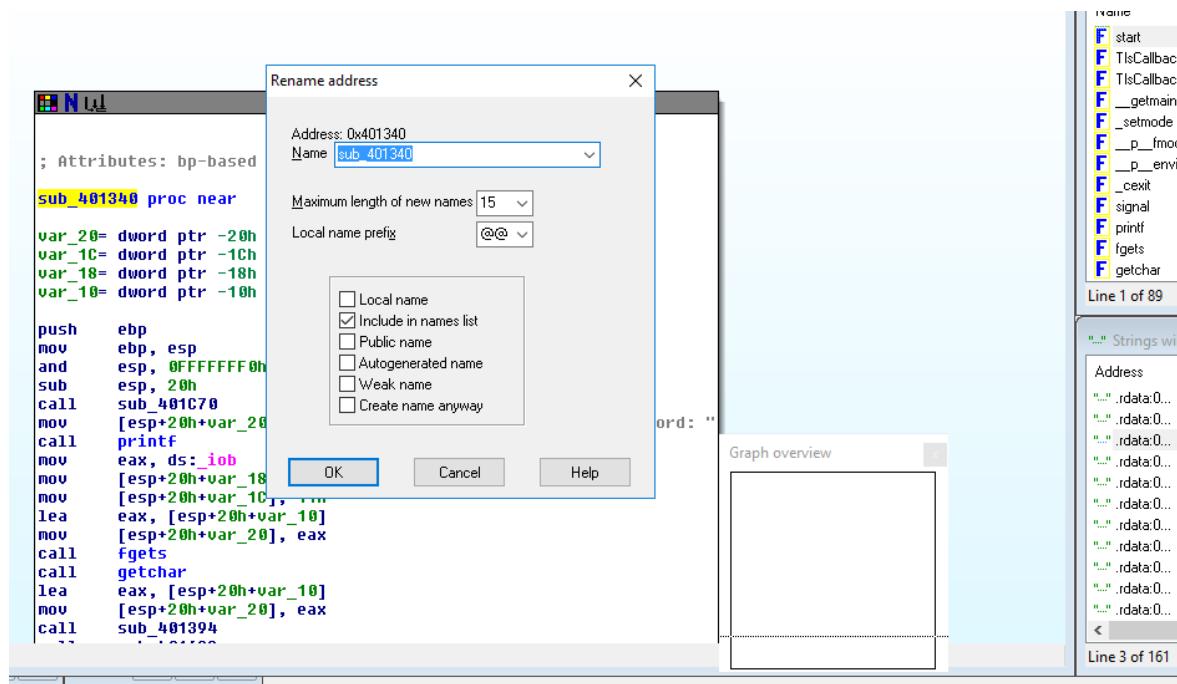
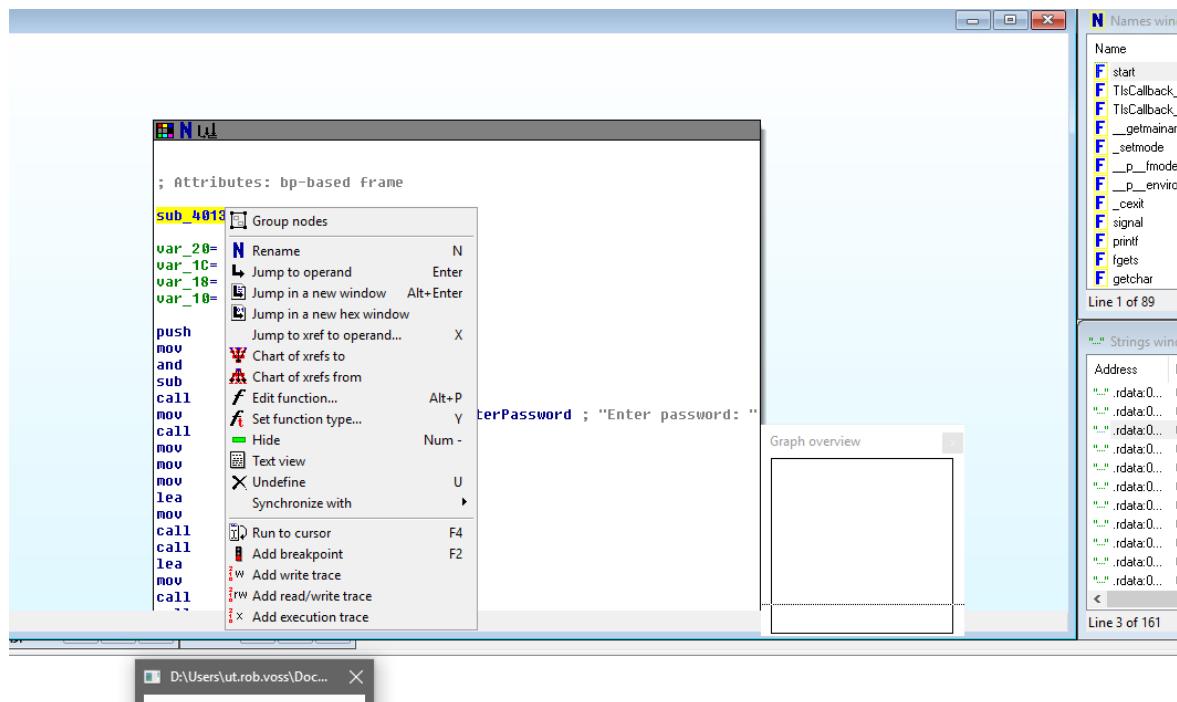
- IDA View-A:** Displays assembly code for a function. The code includes instructions like `push ebp`, `mov esp,ebp`, and `and esp, 0FFFFFFFh`. It contains comments such as "Attributes: bp-based Frame", "Unknown pseudo relocation bit size 3d", and "Unknown pseudo relocation bit size 3d".
- Names window:** Shows a list of symbols with their addresses and types.
- Strings window:** Shows a list of strings with their addresses and lengths.
- Registers window:** Shows CPU registers and memory dump options.
- Stack window:** Shows the stack contents.
- Call graph window:** Shows the call graph for the current function.
- File menu:** Contains options like File, Edit, Jump, Search, View, Debugger, Options, Windows, Help.

Compiler output at the bottom:

```
Compiling file 'C:\Program Files (x86)\IDA Free\idc\ida.idc'...
Compiling file 'C:\Program Files (x86)\IDA Free\idc\onload.idc'...
Executing function _onLoad_411...
IDA is analyzing the input file...
```

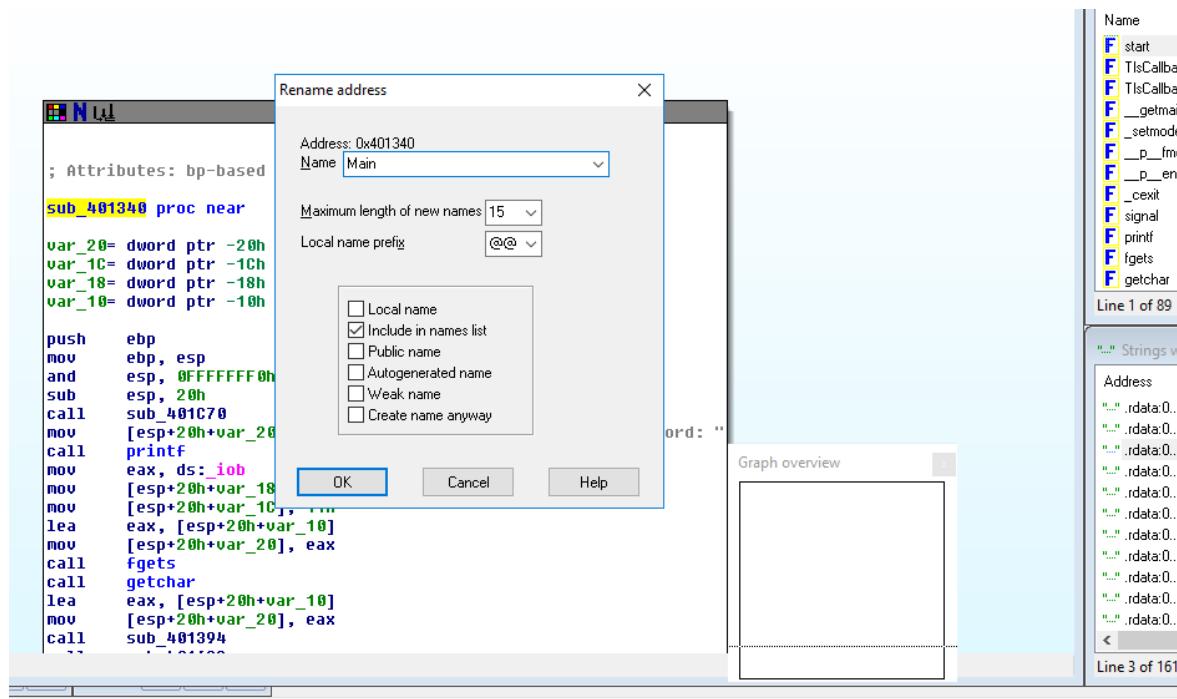
Cracking a Password

Right click on the sub routine name and select rename...

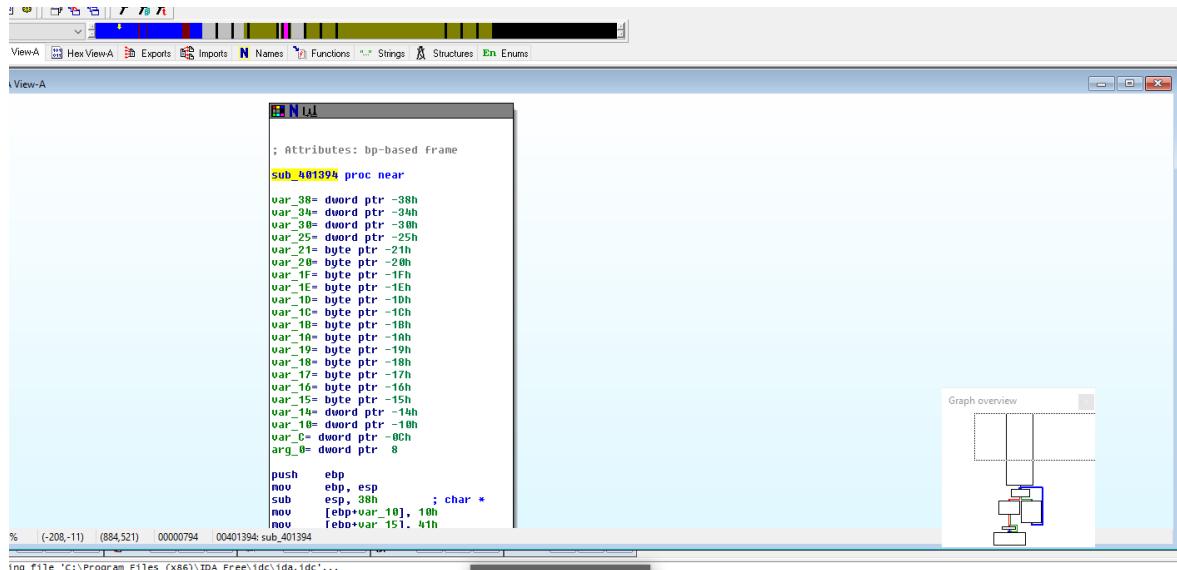


Cracking a Password

Rename this sub routine **Main**

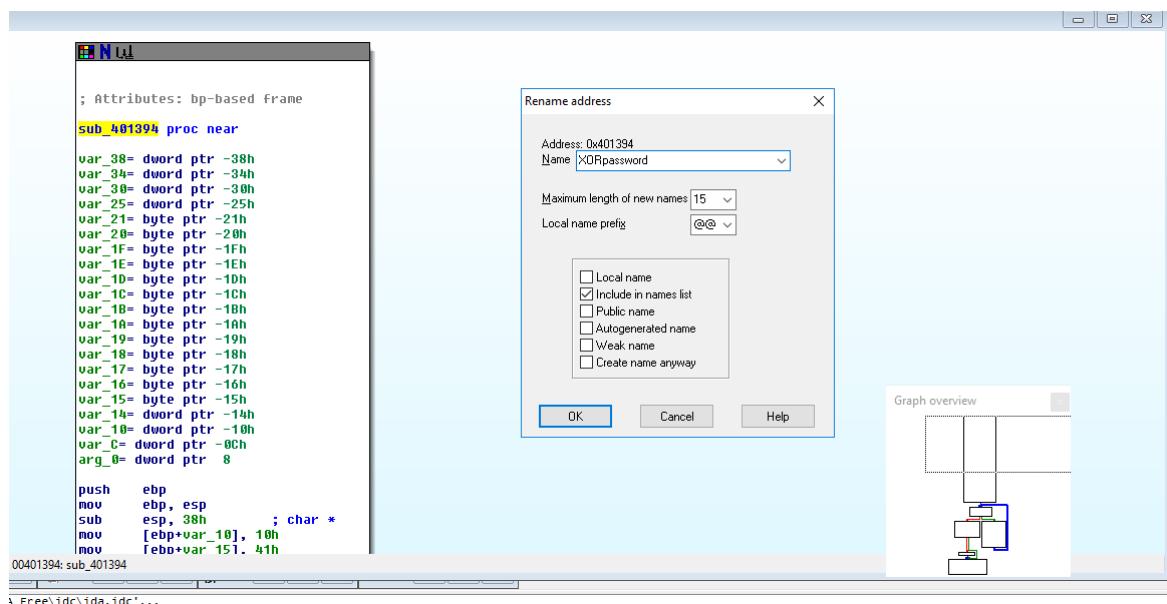
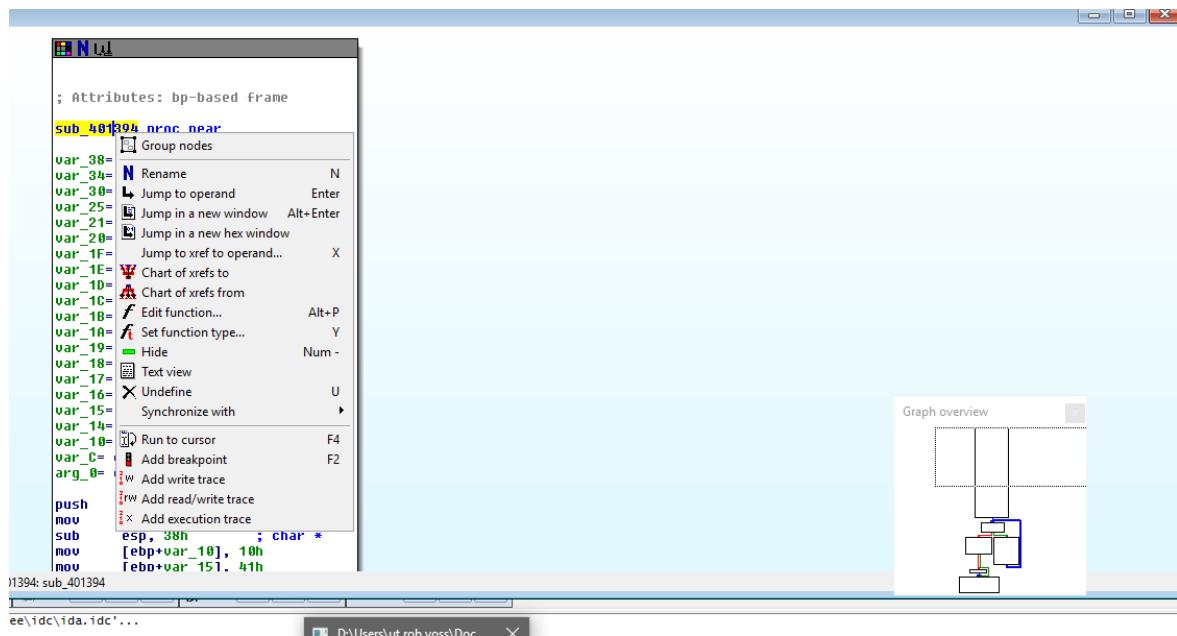


Select first call sub routine sub_401394



Cracking a Password

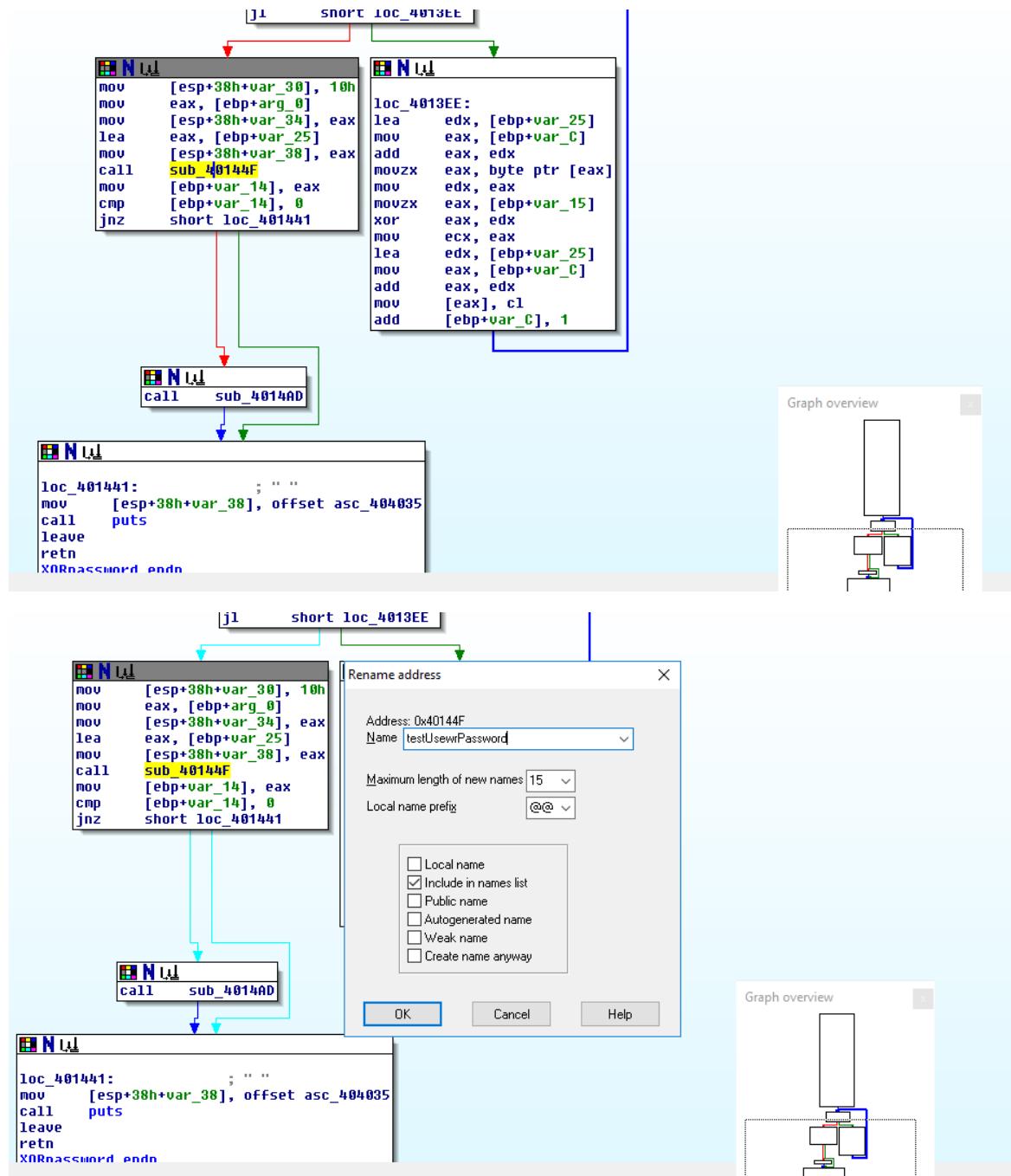
Rename it the XORpassword



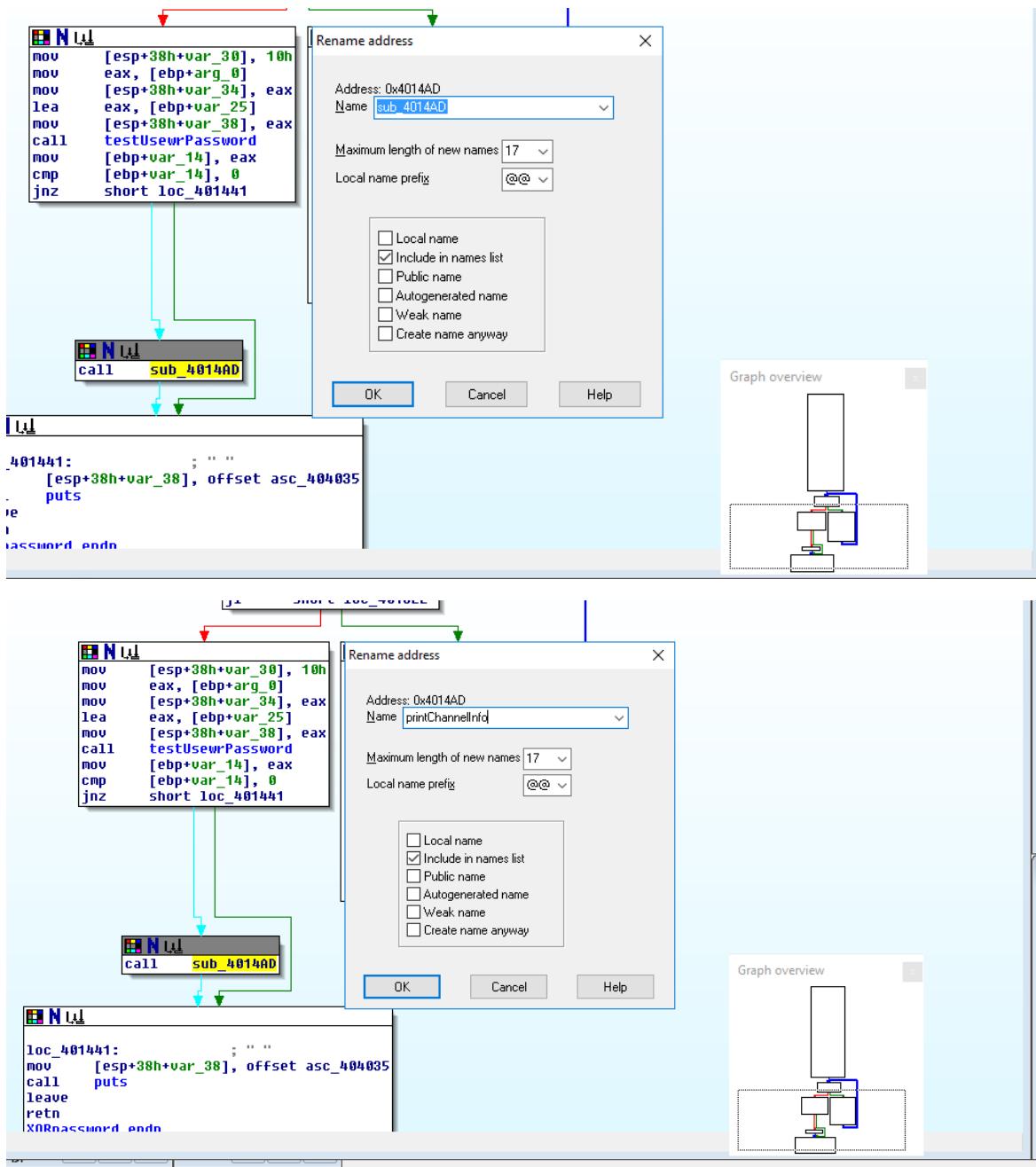
Cracking a Password

Scroll down and locate

- 1) Password comparison sub routine
- 2) Print routine



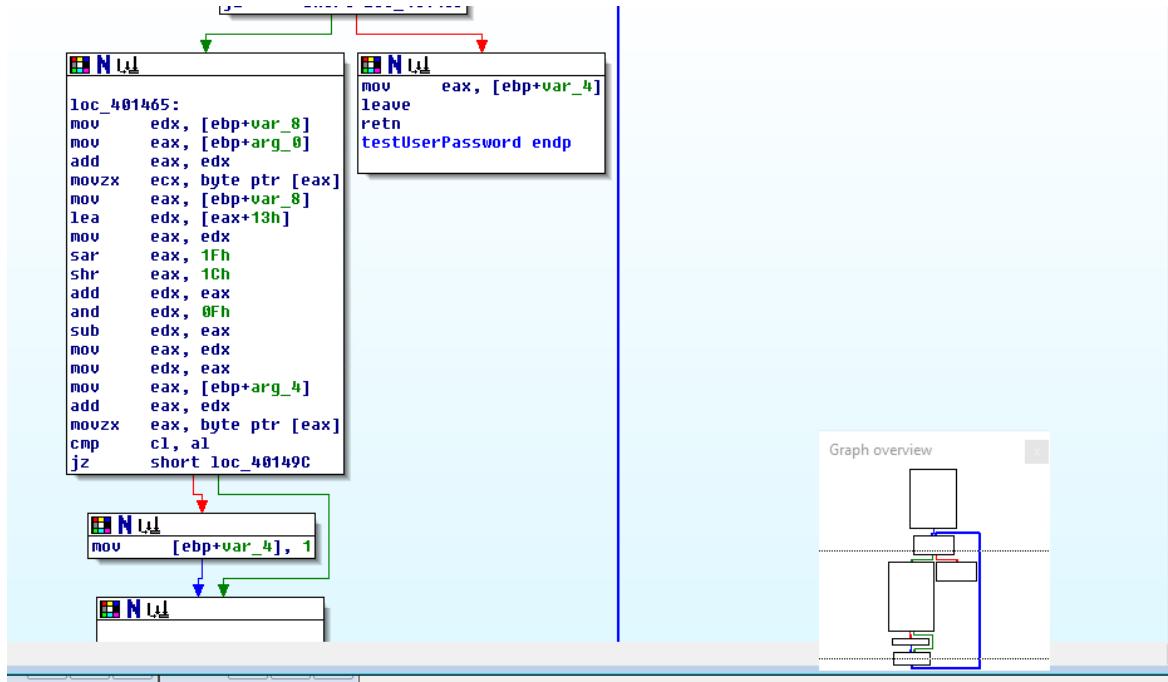
Cracking a Password



Cracking a Password

Return to and open testUserPassword

Investigate this sub routine

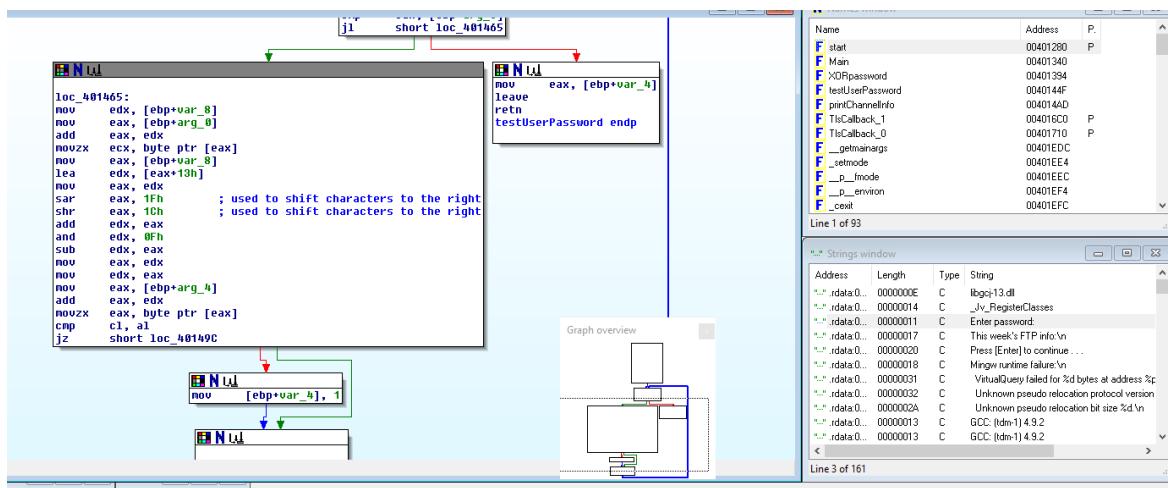


SAR and SHR are two new concepts...

They are both used to shift characters to the right...(SAL SHL will shift to the left)

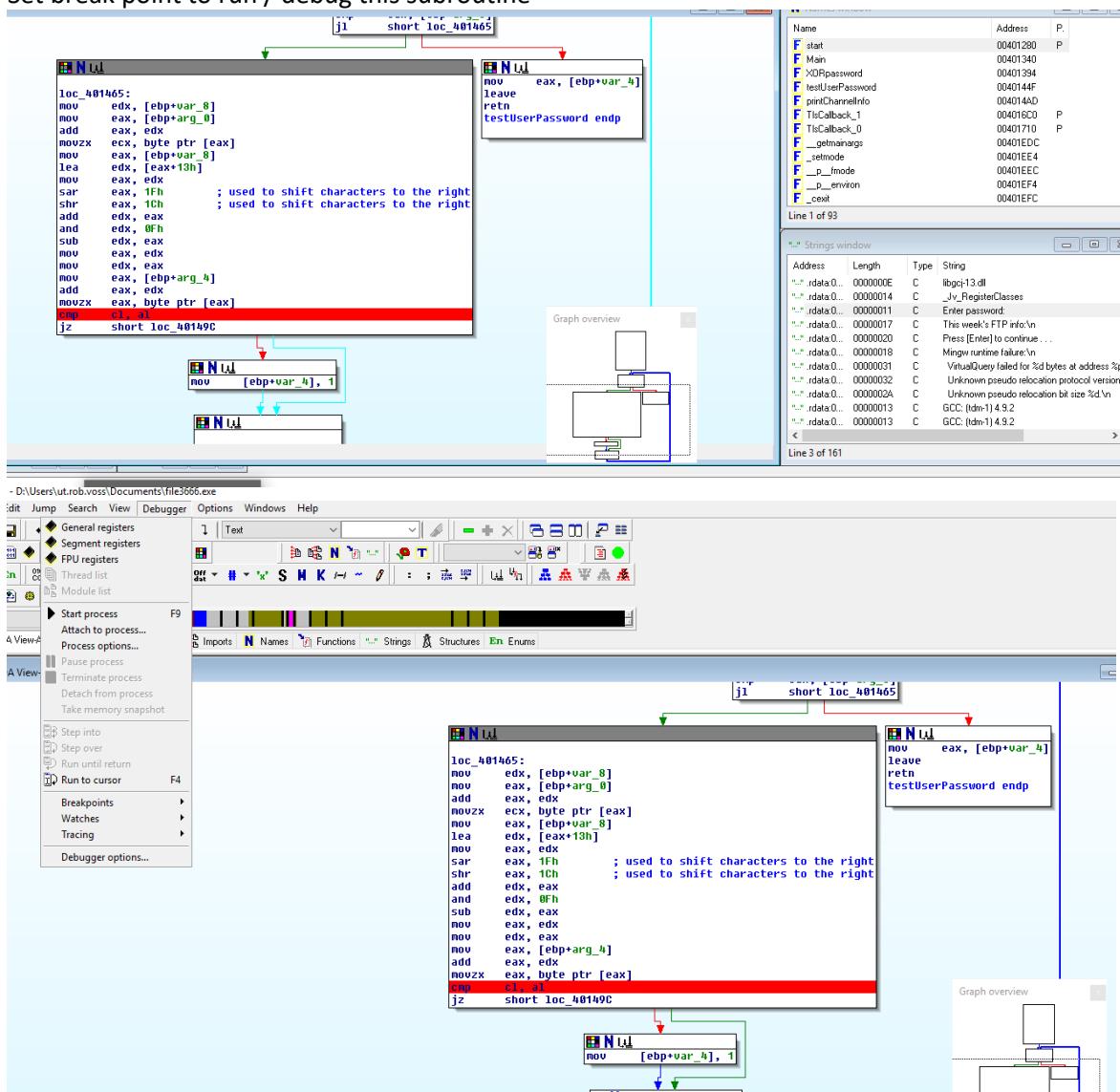
Add a comment here

: used to shift characters to the right



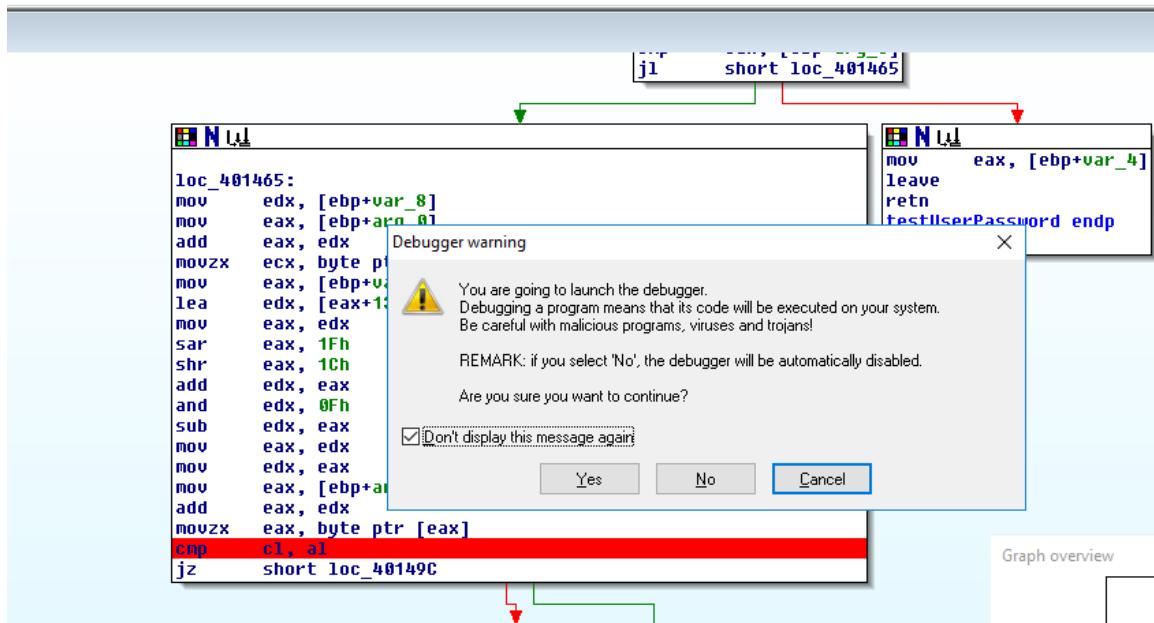
Cracking a Password

Set break point to run / debug this subroutine

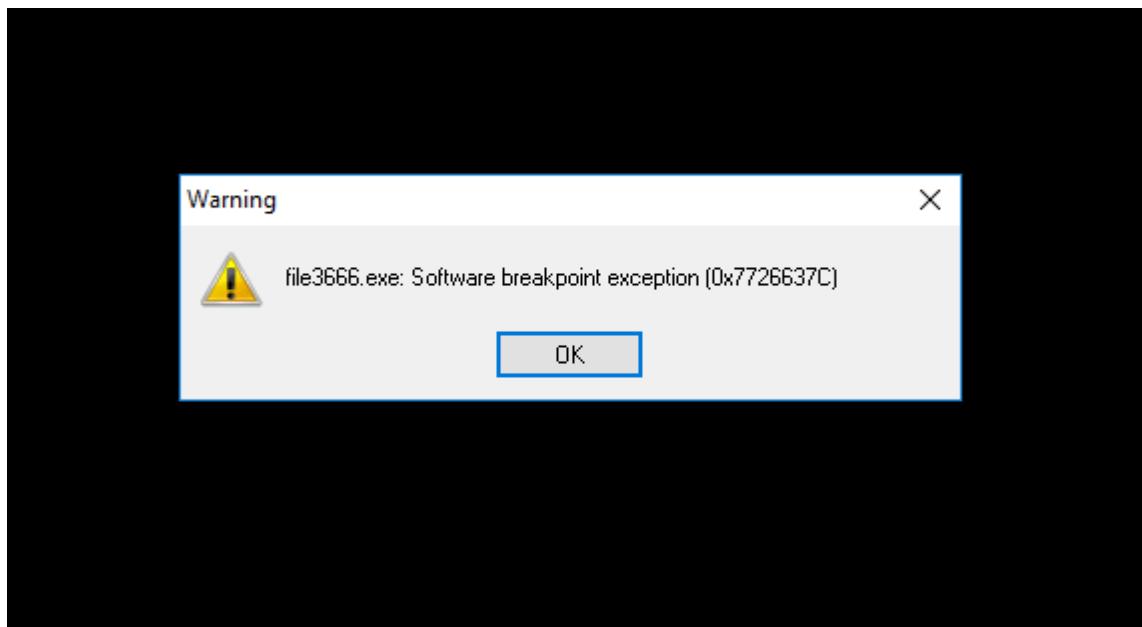


Start Process or F9

Cracking a Password

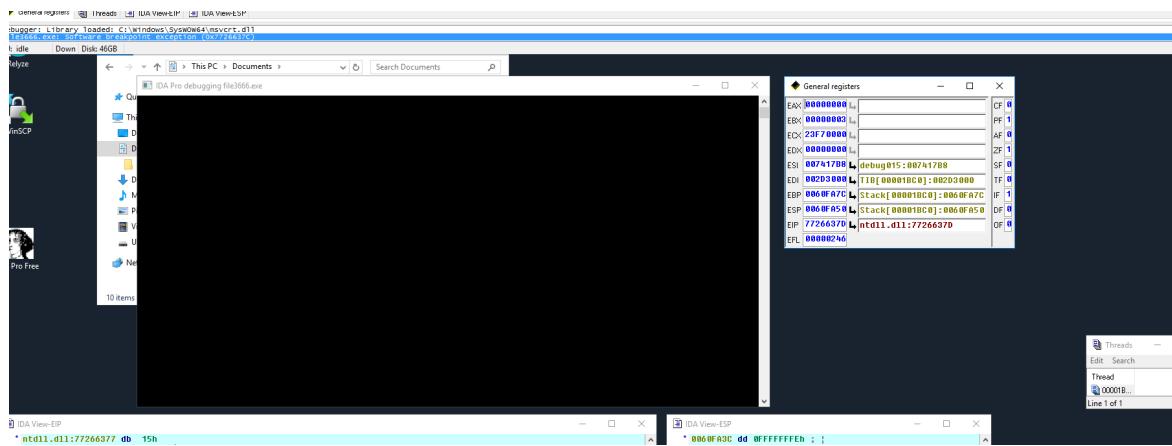


Knowing that this file is safe you can select don't display and hit yes



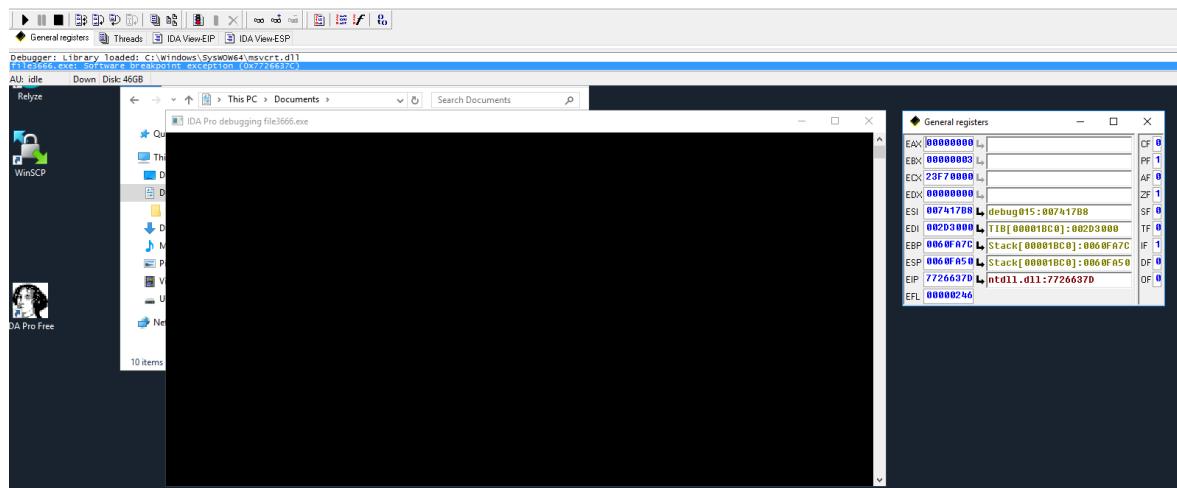
Hit OK

Cracking a Password



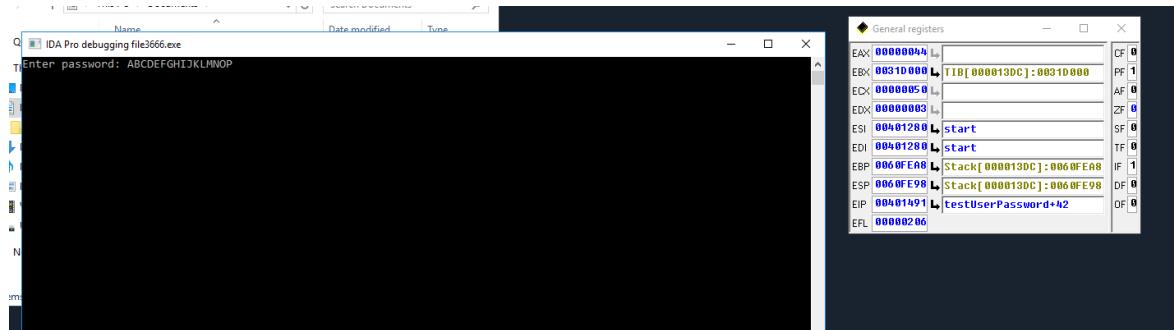
Start Process

Hit the right arrow button on the upper left task bar



Cracking a Password

Enter a password for testing...in this case something that will be easily identifiable in hex
ABCDEFGHIJKLMNP



Cycling through the routine you can see that the registers are at different hex and will cycle up by one each time...

Through multiple processes it has become apparent that the password is similar to the original password PasswordStillSux...but this sun routine adds another element to it by shifting the characters 3 times to the right...as seen in register EDX...

Knowing this you can do two things now...

Rename the sub routine shiftPasswordCharacters

And you now know the password is SuxPasswordStill

When you push to the right on PasswordStillSux the last letter jumps to become the first letter...if you do 3 shifts the password now becomes SuxPasswordStill...

Cracking a Password

The screenshot shows a debugger interface with two windows. The top window displays the assembly code for the `shiftPasswordCharacters` procedure:

```
; Attributes: bp-based frame
shiftPasswordCharacters proc near

var_8= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr 8
arg_4= dword ptr 0Ch
arg_8= dword ptr 10h

push    ebp
mov     ebp, esp
sub    esp, 10h
mov     [ebp+var_4], 0
mov     [ebp+var_8], 0
jmp    short loc_4014A0
```

The bottom window shows the assembly code for the `loc_4014A0` label:

```
loc_4014A0:
mov     eax, [ebp+var_8]
cmp     eax, [ebp+arg_8]
```

To the right, a "Graph overview" window shows a control flow graph with nodes and edges representing the program's execution flow.

The screenshot shows a debugger interface with two windows. The left window displays the assembly code for the `loc_401465` label:

```
loc_401465:
mov     edx, [ebp+var_8]
mov     eax, [ebp+arg_0]
add    eax, edx
movzx  ecx, byte ptr [eax]
mov     eax, [ebp+var_8]
lea     edx, [eax+13h]
mov     eax, edx
sar     eax, 1Fh      ; used to shift characters to the right
shr     eax, 1Ch      ; used to shift characters to the right
add    edx, eax
and    edx, 0Fh
sub    edx, eax
mov     eax, edx
mov     edx, eax
mov     eax, [ebp+arg_4]
add    eax, edx
movzx  eax, byte ptr [eax]
cmp     cl, al
jz     short loc_40149C
```

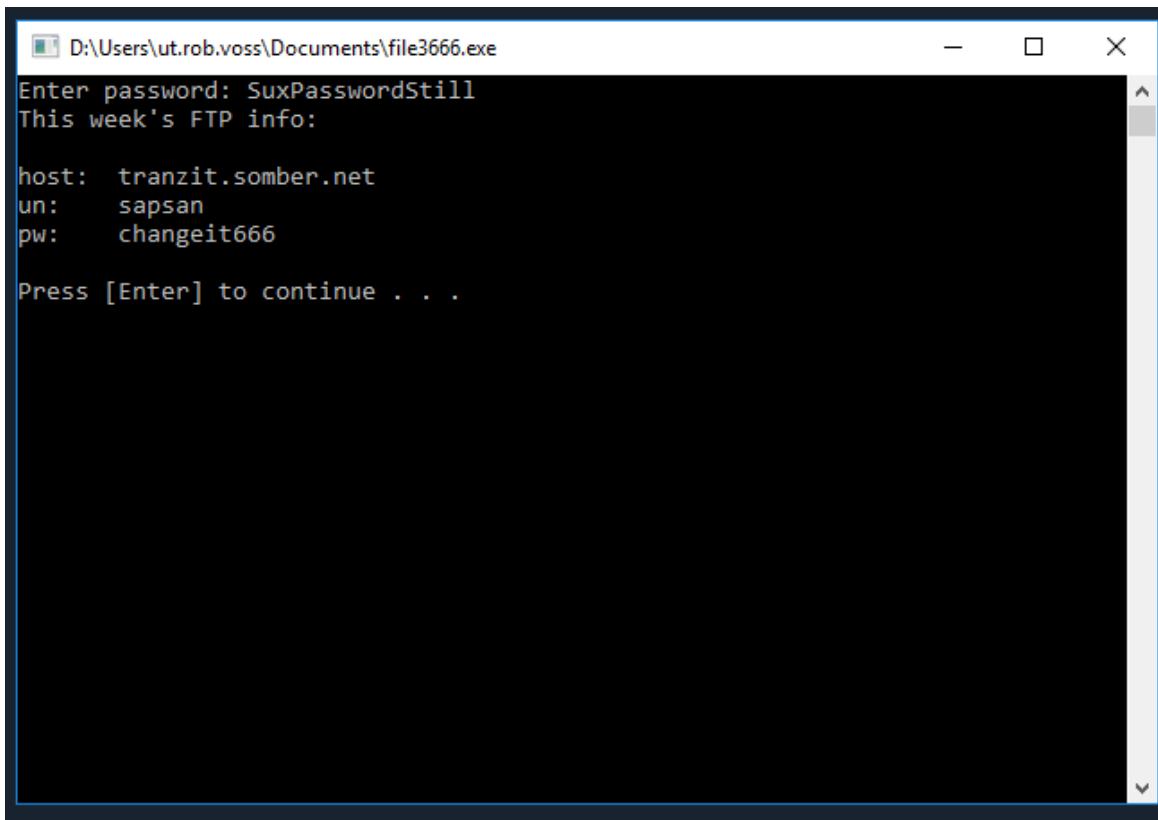
The right window shows the assembly code for the end of the `shiftPasswordCharacters` procedure:

```
mov     eax, [ebp+var_4]
leave
ret
shiftPasswordCharacters endp
```

A red box highlights the `cmp cl, al` instruction. Below it, a green box highlights the `mov [ebp+var_4], 1` instruction. To the right, a "Graph overview" window shows a control flow graph with nodes and edges representing the program's execution flow.

Cracking a Password

Run the password



D:\Users\ut.rob.voss\Documents\file3666.exe

```
Enter password: SuxPasswordStill
This week's FTP info:

host: tranzit.somber.net
un: sapsan
pw: changeit666

Press [Enter] to continue . . .
```