Fall 2015

# CODEBREAKER CHALLENGE 3.0

## Challenge Scenario

NSA has discovered that the leadership of a terrorist organization is using a new method of communicating secret messages to its operatives in the field and has provided each individual with a unique program for decoding messages. Your mission is to reverse-engineer this software and develop capabilities to exploit the secret messaging component.

## The Challenge

- There are 4 different levels or "tasks" to this challenge problem
  - Task 1: Determine how to execute the hidden functionality
  - Task 2: Bypass an authentication check
  - Task 3: Create an encoder program
  - Task 4: Spoof a message to a high-value target
- Each task gets progressively harder and builds off previous ones

## The Challenge (cont.)

- Challenge materials and instructions can be found at https://codebreaker.ltsnet.net
- Register for an account with your .edu email address

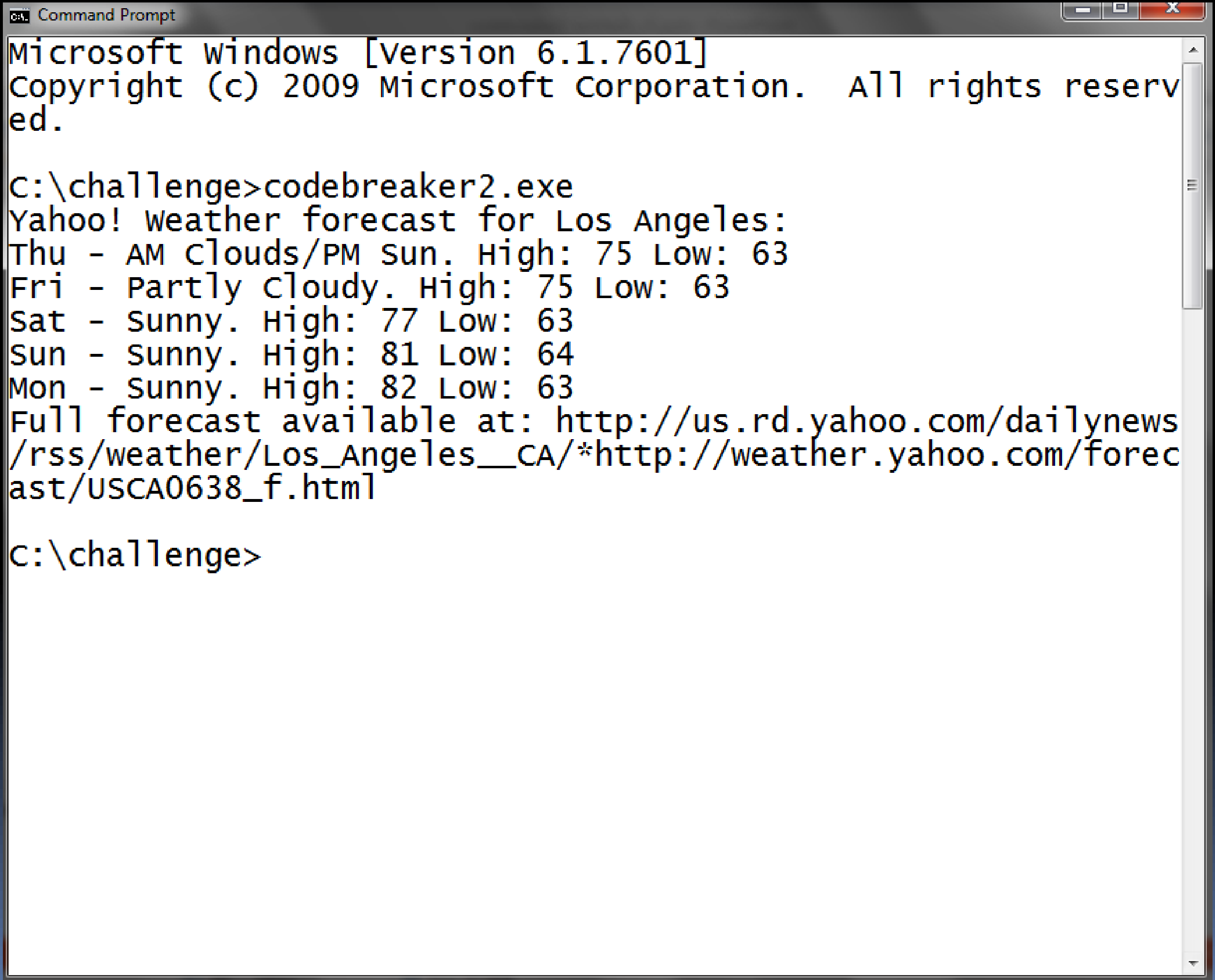
## Reverse Engineering Tips

- Examine strings in the binary using IDA
  - Look for clues that relate to the functionality you are trying to find / reverse
  - Utilize IDA xrefs to find code that references the string(s) of interest
  - Utilize symbols (e.g., function names) to help determine what a section of code does
- Try setting debugger breakpoints to help RE code
  - Single-step after hitting a breakpoint and see how the values in registers/memory change
  - Look for the result of interesting computations. You can sometimes get the data you need from memory
- Leverage online resources, e.g.,Intel manuals, RE lectures, etc. for help on reverse-engineering

## Technical Walkthrough

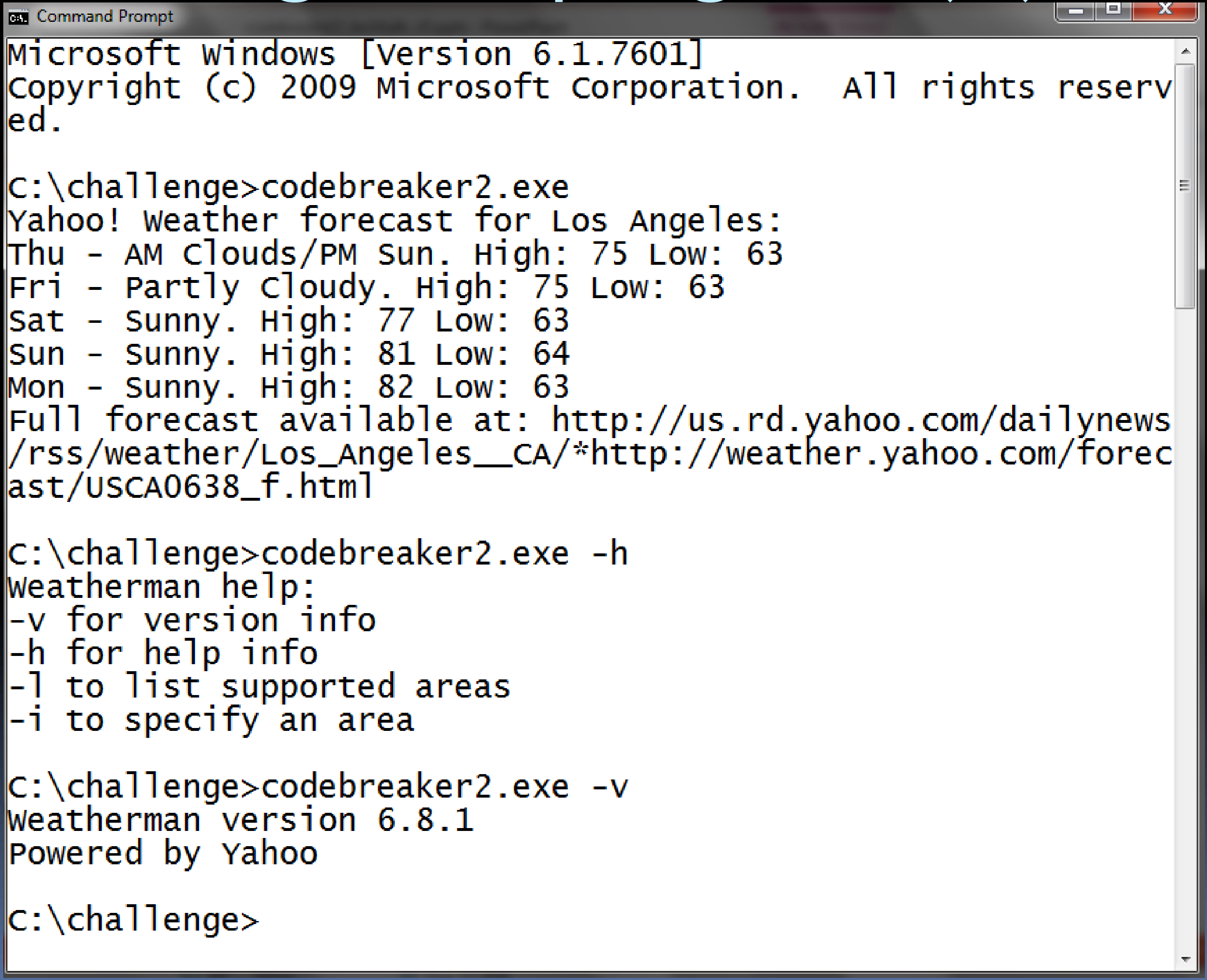
- 2014 Codebreaker Challenge on Windows using IDA Pro Demo
- This binary can be downloaded from https://codebreaker.ltsnet.net/resources

## Running the program



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

C:\challenge>codebreaker2.exe
Yahoo! Weather forecast for Los Angeles:
Thu - AM Clouds/PM Sun. High: 75 Low: 63
Fri - Partly Cloudy. High: 75 Low: 63
Sat - Sunny. High: 77 Low: 63
Sun - Sunny. High: 81 Low: 64
Mon - Sunny. High: 82 Low: 63
Full forecast available at: http://us.rd.yahoo.com/dailynews
/rss/weather/Los_Angeles__CA/*http://weather.yahoo.com/forec
ast/USCA0638_f.html

C:\challenge>
```

## Running the program (2)

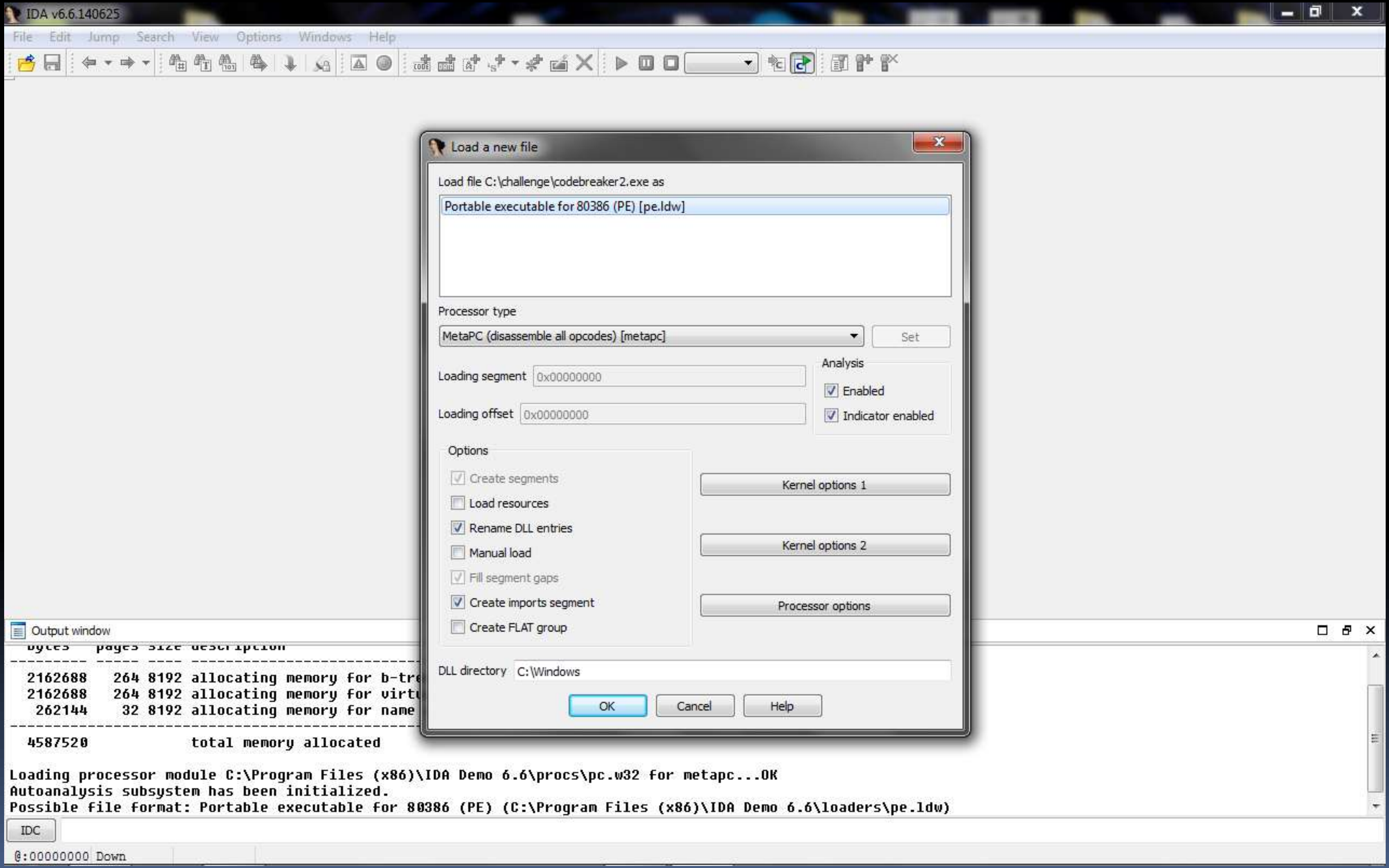

Command Prompt

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

C:\challenge>codebreaker2.exe
Yahoo! Weather forecast for Los Angeles:
Thu - AM Clouds/PM Sun. High: 75 Low: 63
Fri - Partly Cloudy. High: 75 Low: 63
Sat - Sunny. High: 77 Low: 63
Sun - Sunny. High: 81 Low: 64
Mon - Sunny. High: 82 Low: 63
Full forecast available at: http://us.rd.yahoo.com/dailynews
/rss/weather/Los_Angeles__CA/*http://weather.yahoo.com/forec
ast/USCA0638_f.html

C:\challenge>codebreaker2.exe -h
Weatherman help:
-v for version info
-h for help info
-l to list supported areas
-i to specify an area

C:\challenge>codebreaker2.exe -v
Weatherman version 6.8.1
Powered by Yahoo

C:\challenge>
```

## Disassemble

- Disassemble the Codebreaker2 binary
  - If asked whether you want to use Proximity View
    - Click no
    - Use graph view

# Disassemble (2)



Load file C:\challenge\codebreaker2.exe as

Portable executable for 80386 (PE) [pe.ldw]

Processor type: MetaPC (disassemble all opcodes) [metapc]

Loading segment: 0x00000000

Loading offset: 0x00000000

Analysis
- [✓] Enabled
- [✓] Indicator enabled

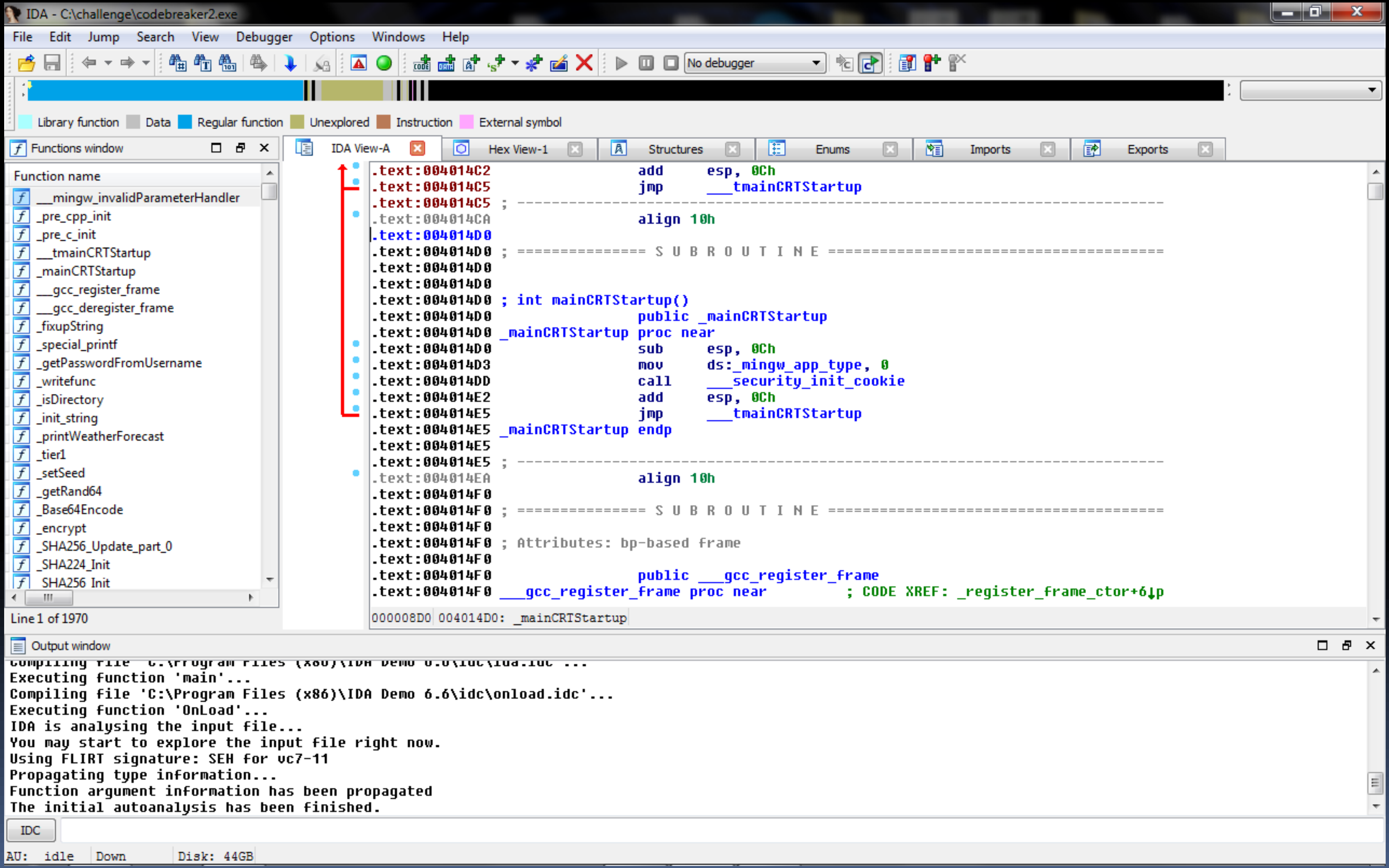
Options
- [✓] Create segments
- [ ] Load resources
- [✓] Rename DLL entries
- [ ] Manual load
- [✓] Fill segment gaps
- [✓] Create imports segment
- [ ] Create FLAT group

DLL directory: C:\Windows

```
 bytes   pages size description
--------- ----- ---- ----------------------------
  2162688   264 8192 allocating memory for b-tre
  2162688   264 8192 allocating memory for virtu
   262144    32 8192 allocating memory for name
---------------------------------------------
  4587520           total memory allocated

Loading processor module C:\Program Files (x86)\IDA Demo 6.6\procs\pc.w32 for metapc...OK
Autoanalysis subsystem has been initialized.
Possible file format: Portable executable for 80386 (PE) (C:\Program Files (x86)\IDA Demo 6.6\loaders\pe.ldw)
```

@:00000000 Down

# Disassemble (3)


IDA - C:\challenge\codebreaker2.exe
File Edit Jump Search View Debugger Options Windows Help
No debugger
Library function Data Regular function Unexplored Instruction External symbol
Functions window
IDA View-A
Hex View-1
Structures
Enums
Imports
Exports
Output window
IDC

Function name
- __mingw_invalidParameterHandler
- _pre_cpp_init
- _pre_c_init
- ___tmainCRTStartup
- _mainCRTStartup
- ___gcc_register_frame
- ___gcc_deregister_frame
- _fixupString
- _special_printf
- _getPasswordFromUsername
- _writefunc
- _isDirectory
- _init_string
- _printWeatherForecast
- _tier1
- _setSeed
- _getRand64
- _Base64Encode
- _encrypt
- _SHA256_Update_part_0
- _SHA224_Init
- _SHA256_Init

Line 1 of 1970

```
.text:004014C2                 add     esp, 0Ch
.text:004014C5                 jmp     ___tmainCRTStartup
.text:004014C5 ; ---------------------------------------------------------------------------
.text:004014CA                 align 10h
.text:004014D0
.text:004014D0 ; =============== S U B R O U T I N E =======================================
.text:004014D0
.text:004014D0
.text:004014D0 ; int mainCRTStartup()
.text:004014D0                 public _mainCRTStartup
.text:004014D0 _mainCRTStartup proc near
.text:004014D0                 sub     esp, 0Ch
.text:004014D3                 mov     ds:_mingw_app_type, 0
.text:004014DD                 call    ___security_init_cookie
.text:004014E2                 add     esp, 0Ch
.text:004014E5                 jmp     ___tmainCRTStartup
.text:004014E5 _mainCRTStartup endp
.text:004014E5
.text:004014E5 ; ---------------------------------------------------------------------------
.text:004014EA                 align 10h
.text:004014F0
.text:004014F0 ; =============== S U B R O U T I N E =======================================
.text:004014F0
.text:004014F0 ; Attributes: bp-based frame
.text:004014F0
.text:004014F0                 public ___gcc_register_frame
.text:004014F0 ___gcc_register_frame proc near          ; CODE XREF: _register_frame_ctor+6↓p
000008D0 004014D0: _mainCRTStartup
```

```
Compiling file 'C:\Program Files (x86)\IDA Demo 6.6\idc\ida.idc'...
Executing function 'main'...
Compiling file 'C:\Program Files (x86)\IDA Demo 6.6\idc\onload.idc'...
Executing function 'OnLoad'...
IDA is analysing the input file...
You may start to explore the input file right now.
Using FLIRT signature: SEH for vc7-11
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
```

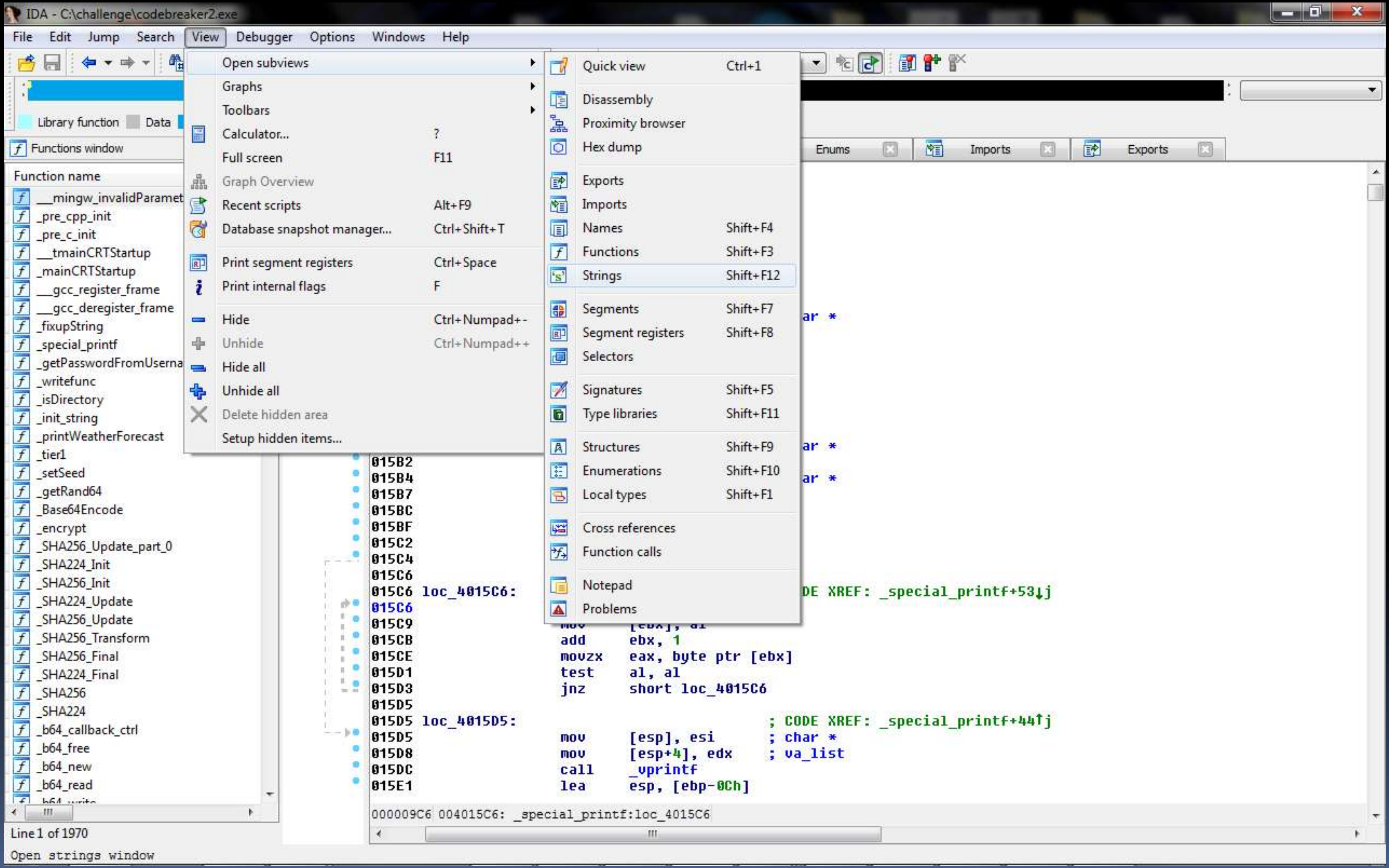
AU: idle Down Disk: 44GB

## Observe Strings

- Observe the strings that show up in IDA
  - Click Views->Open Subviews->Strings
  - You should see the strings that are displayed when you run the program

```
Yahoo! Weather forecast for

Full forecast available at:

Weatherman version 6.8.1
```

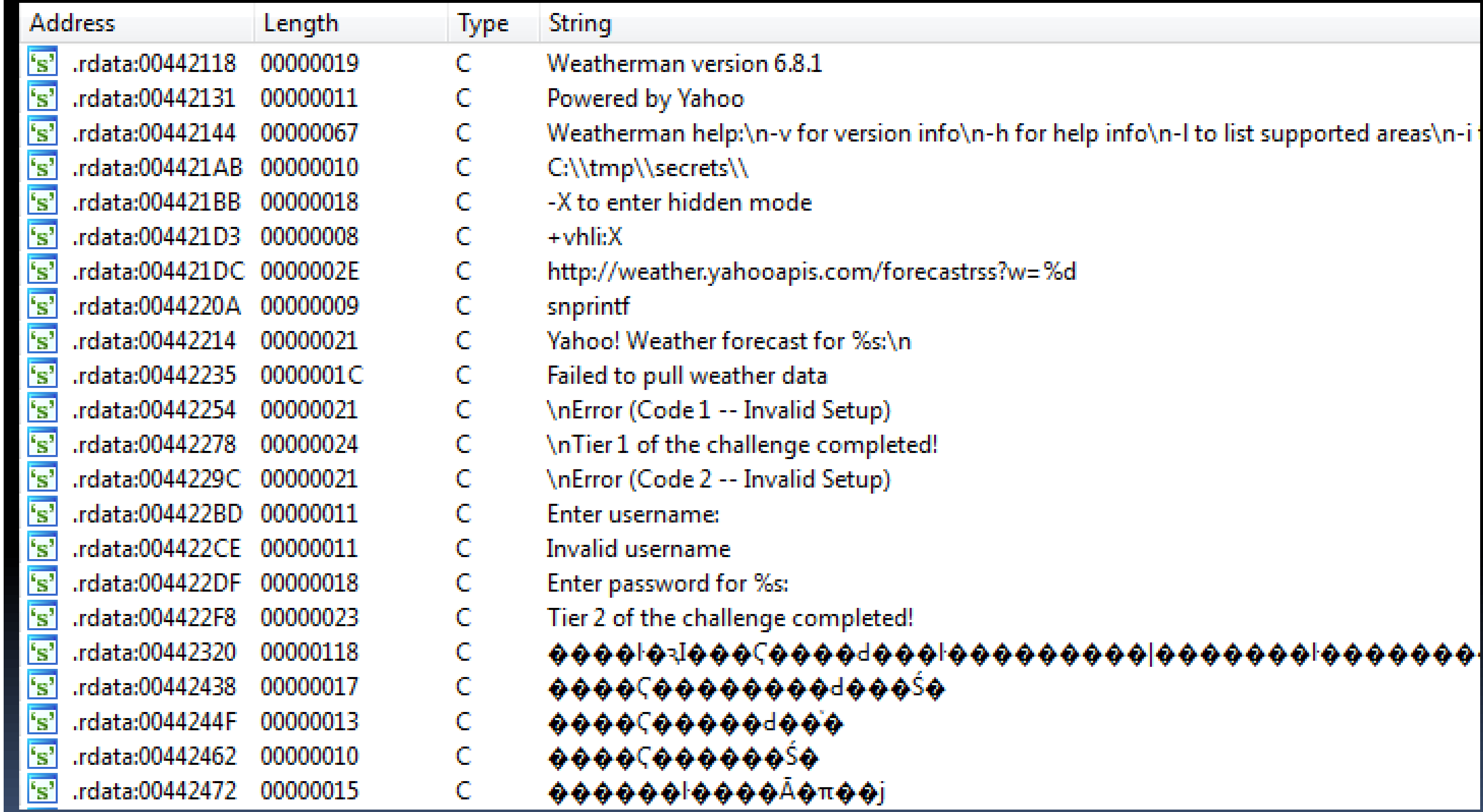
# Observe Strings (2)



```
015B2
015B4
015B7
015BC
015BF
015C2
015C4
015C6
015C6 loc_4015C6:                             DE XREF: _special_printf+53↓j
015C6
015C9
015CB                   add     ebx, 1
015CE                   movzx   eax, byte ptr [ebx]
015D1                   test    al, al
015D3                   jnz     short loc_4015C6
015D5
015D5 loc_4015D5:                             ; CODE XREF: _special_printf+44↑j
015D5                   mov     [esp], esi      ; char *
015D8                   mov     [esp+4], edx    ; va_list
015DC                   call    _vprintf
015E1                   lea     esp, [ebp-0Ch]
000009C6 004015C6: _special_printf:loc_4015C6
```

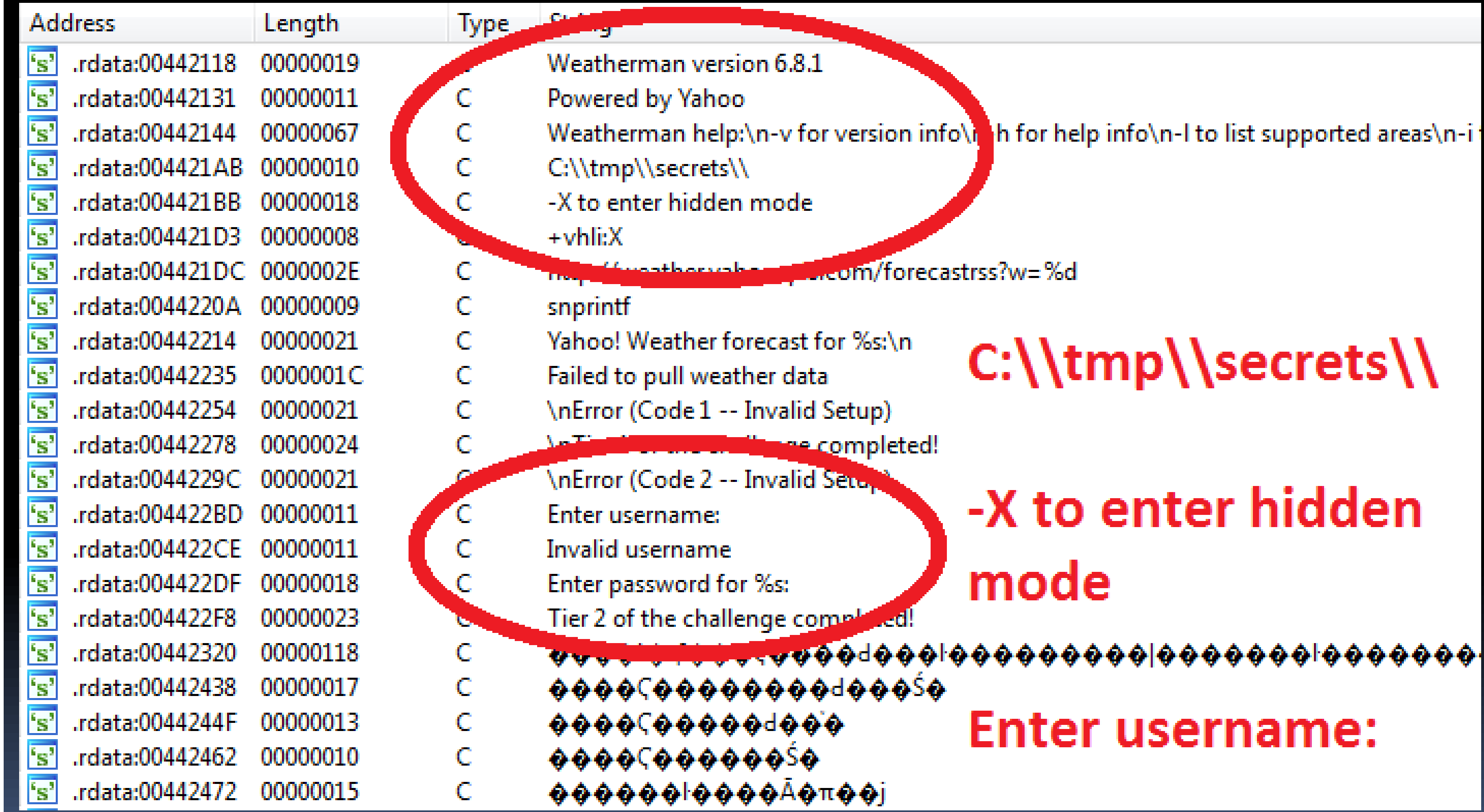
```
Open strings window
```

# Observe Strings (3)



| Address | Length | Type | String |
|---|---|---|---|
| .rdata:00442118 | 00000019 | C | Weatherman version 6.8.1 |
| .rdata:00442131 | 00000011 | C | Powered by Yahoo |
| .rdata:00442144 | 00000067 | C | Weatherman help:\n-v for version info\n-h for help info\n-l to list supported areas\n-i |
| .rdata:004421AB | 00000010 | C | C:\\tmp\\secrets\\ |
| .rdata:004421BB | 00000018 | C | -X to enter hidden mode |
| .rdata:004421D3 | 00000008 | C | +vhli:X |
| .rdata:004421DC | 0000002E | C | http://weather.yahooapis.com/forecastrss?w=%d |
| .rdata:0044220A | 00000009 | C | snprintf |
| .rdata:00442214 | 00000021 | C | Yahoo! Weather forecast for %s:\n |
| .rdata:00442235 | 0000001C | C | Failed to pull weather data |
| .rdata:00442254 | 00000021 | C | \nError (Code 1 -- Invalid Setup) |
| .rdata:00442278 | 00000024 | C | \nTier 1 of the challenge completed! |
| .rdata:0044229C | 00000021 | C | \nError (Code 2 -- Invalid Setup) |
| .rdata:004422BD | 00000011 | C | Enter username: |
| .rdata:004422CE | 00000011 | C | Invalid username |
| .rdata:004422DF | 00000018 | C | Enter password for %s: |
| .rdata:004422F8 | 00000023 | C | Tier 2 of the challenge completed! |
| .rdata:00442320 | 00000118 | C | �����l�ʇI����Ç�����d����l������������\|��������l��������� |
| .rdata:00442438 | 00000017 | C | �����Ç�����������d����Ś� |
| .rdata:0044244F | 00000013 | C | �����Ç��������d���̊� |
| .rdata:00442462 | 00000010 | C | �����Ç���������Ś� |
| .rdata:00442472 | 00000015 | C | �������l������Ā�π��j |

# Observe Strings (4)



| Address | Length | Type | String |
|---|---|---|---|
| .rdata:00442118 | 00000019 | [illegible] | Weatherman version 6.8.1 |
| .rdata:00442131 | 00000011 | C | Powered by Yahoo |
| .rdata:00442144 | 00000067 | C | Weatherman help:\n-v for version info\n-h for help info\n-l to list supported areas\n-i |
| .rdata:004421AB | 00000010 | C | C:\\tmp\\secrets\\ |
| .rdata:004421BB | 00000018 | C | -X to enter hidden mode |
| .rdata:004421D3 | 00000008 | [illegible] | +vhli:X |
| .rdata:004421DC | 0000002E | C | [illegible].com/forecastrss?w=%d |
| .rdata:0044220A | 00000009 | C | snprintf |
| .rdata:00442214 | 00000021 | C | Yahoo! Weather forecast for %s:\n |
| .rdata:00442235 | 0000001C | C | Failed to pull weather data |
| .rdata:00442254 | 00000021 | C | \nError (Code 1 -- Invalid Setup) |
| .rdata:00442278 | 00000024 | C | [illegible] completed! |
| .rdata:0044229C | 00000021 | [illegible] | \nError (Code 2 -- Invalid Setup) |
| .rdata:004422BD | 00000011 | C | Enter username: |
| .rdata:004422CE | 00000011 | C | Invalid username |
| .rdata:004422DF | 00000018 | C | Enter password for %s: |
| .rdata:004422F8 | 00000023 | [illegible] | Tier 2 of the challenge completed! |
| .rdata:00442320 | 00000118 | C | [illegible]����d����l����������\|��������l�������� |
| .rdata:00442438 | 00000017 | C | ����Ç����������d����Ś� |
| .rdata:0044244F | 00000013 | C | ����Ç�������d���� |
| .rdata:00442462 | 00000010 | C | ����Ç��������Ś� |
| .rdata:00442472 | 00000015 | C | �������l�����Ā�π��j |

**C:\\tmp\\secrets\**

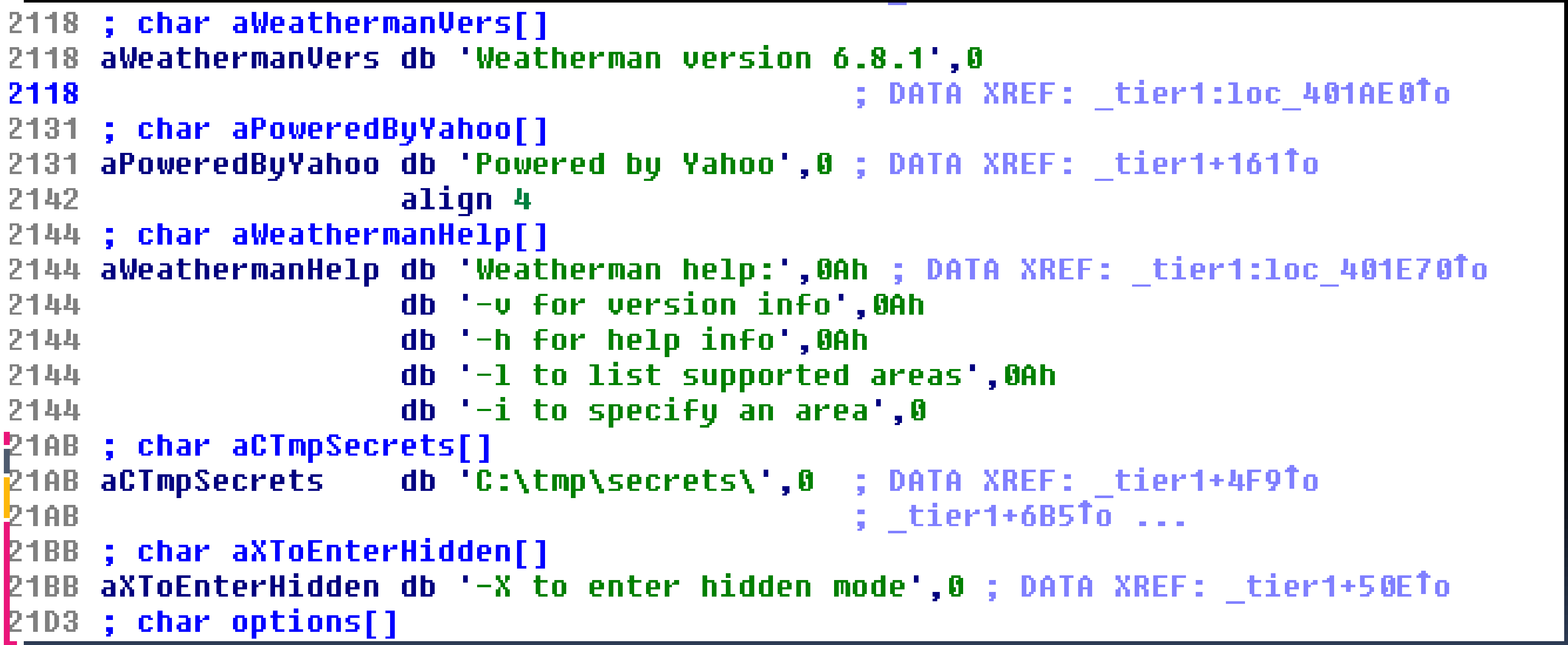
**-X to enter hidden mode**

**Enter username:**

## C:\\tmp\\secrets

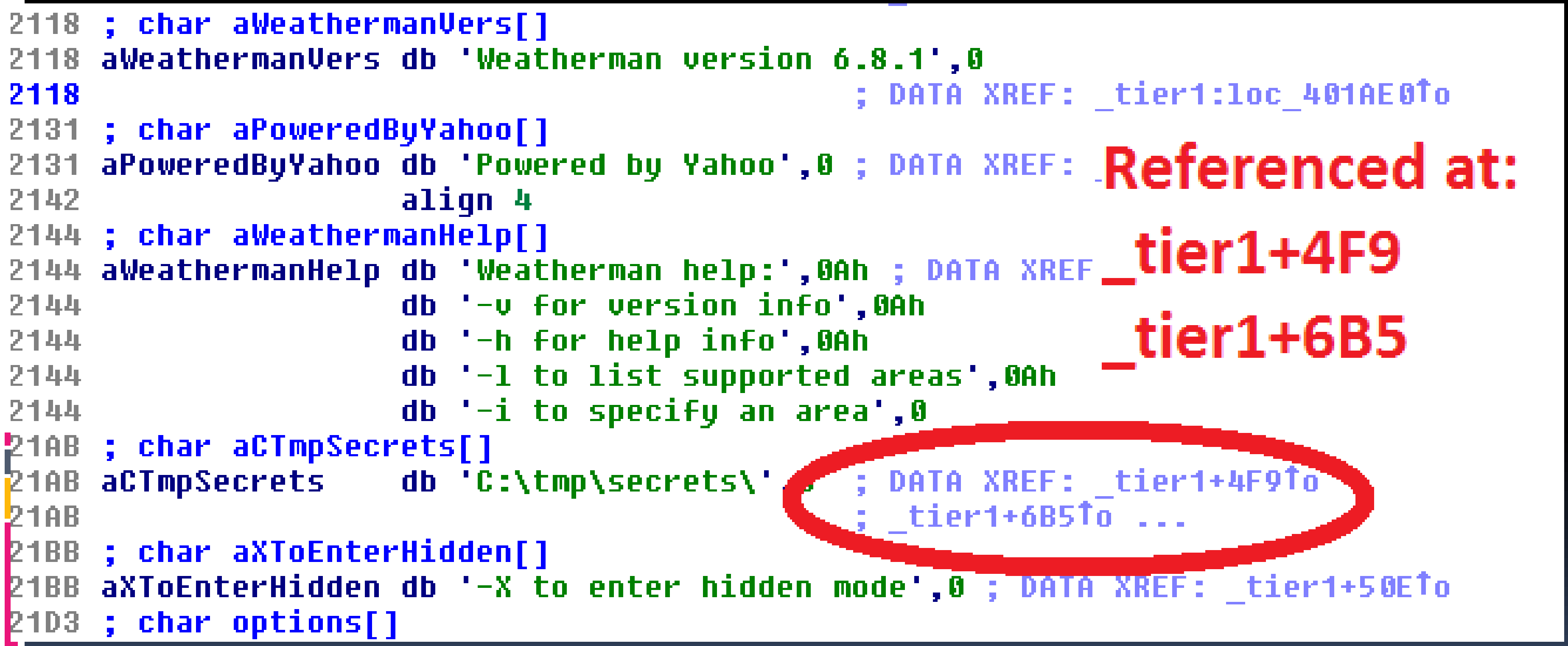
- Double click on the "C:\\tmp\\secrets" string
    - This takes you to the data section of the binary where the string is stored
- To the right of the string are cross references to this address (show up as DATA XREF in IDA)
- Press ctrl-x to pull up a cross-references window; you will see two different references

# C:\\tmp\\secrets (2)



```
2118 ; char aWeathermanVers[]
2118 aWeathermanVers db 'Weatherman version 6.8.1',0
2118                                               ; DATA XREF: _tier1:loc_401AE0↑o
2131 ; char aPoweredByYahoo[]
2131 aPoweredByYahoo db 'Powered by Yahoo',0 ; DATA XREF: _tier1+161↑o
2142                 align 4
2144 ; char aWeathermanHelp[]
2144 aWeathermanHelp db 'Weatherman help:',0Ah ; DATA XREF: _tier1:loc_401E70↑o
2144                 db '-v for version info',0Ah
2144                 db '-h for help info',0Ah
2144                 db '-l to list supported areas',0Ah
2144                 db '-i to specify an area',0
21AB ; char aCTmpSecrets[]
21AB aCTmpSecrets    db 'C:\tmp\secrets\',0  ; DATA XREF: _tier1+4F9↑o
21AB                                               ; _tier1+6B5↑o ...
21BB ; char aXToEnterHidden[]
21BB aXToEnterHidden db '-X to enter hidden mode',0 ; DATA XREF: _tier1+50E↑o
21D3 ; char options[]
```

# C:\\tmp\\secrets (3)

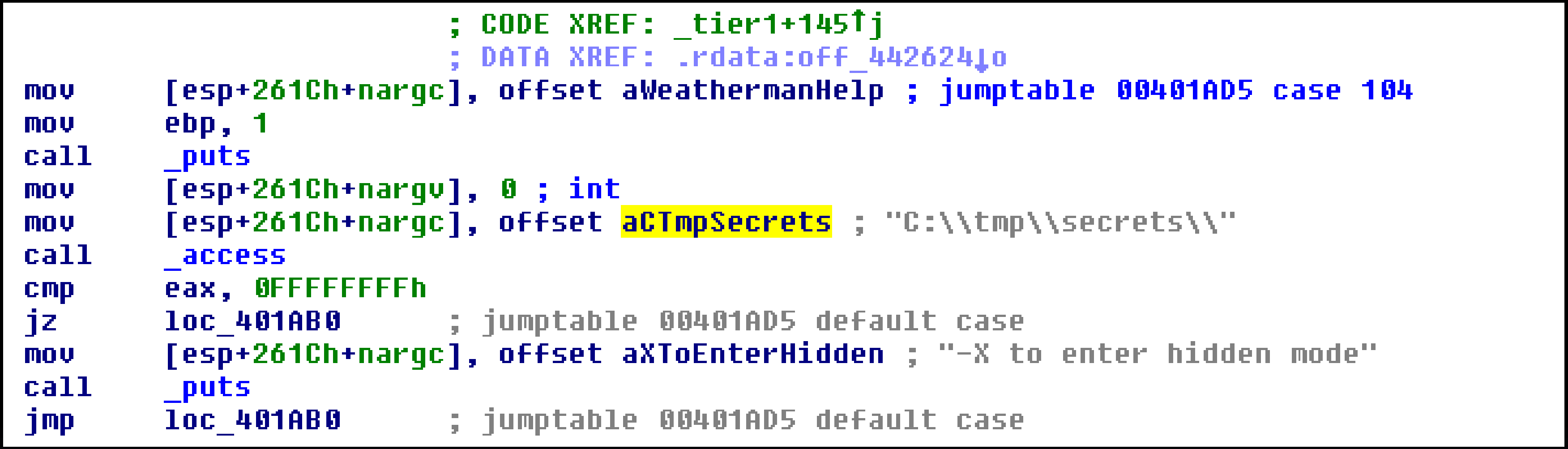


```
2118 ; char aWeathermanVers[]
2118 aWeathermanVers db 'Weatherman version 6.8.1',0
2118                                              ; DATA XREF: _tier1:loc_401AE0↑o
2131 ; char aPoweredByYahoo[]
2131 aPoweredByYahoo db 'Powered by Yahoo',0 ; DATA XREF: _
2142                 align 4
2144 ; char aWeathermanHelp[]
2144 aWeathermanHelp db 'Weatherman help:',0Ah ; DATA XREF _
2144                 db '-v for version info',0Ah
2144                 db '-h for help info',0Ah
2144                 db '-l to list supported areas',0Ah
2144                 db '-i to specify an area',0
21AB ; char aCTmpSecrets[]
21AB aCTmpSecrets    db 'C:\tmp\secrets\',0     ; DATA XREF: _tier1+4F9↑o
21AB                                              ; _tier1+6B5↑o ...
21BB ; char aXToEnterHidden[]
21BB aXToEnterHidden db '-X to enter hidden mode',0 ; DATA XREF: _tier1+50E↑o
21D3 ; char options[]
```

## Double-click Reference

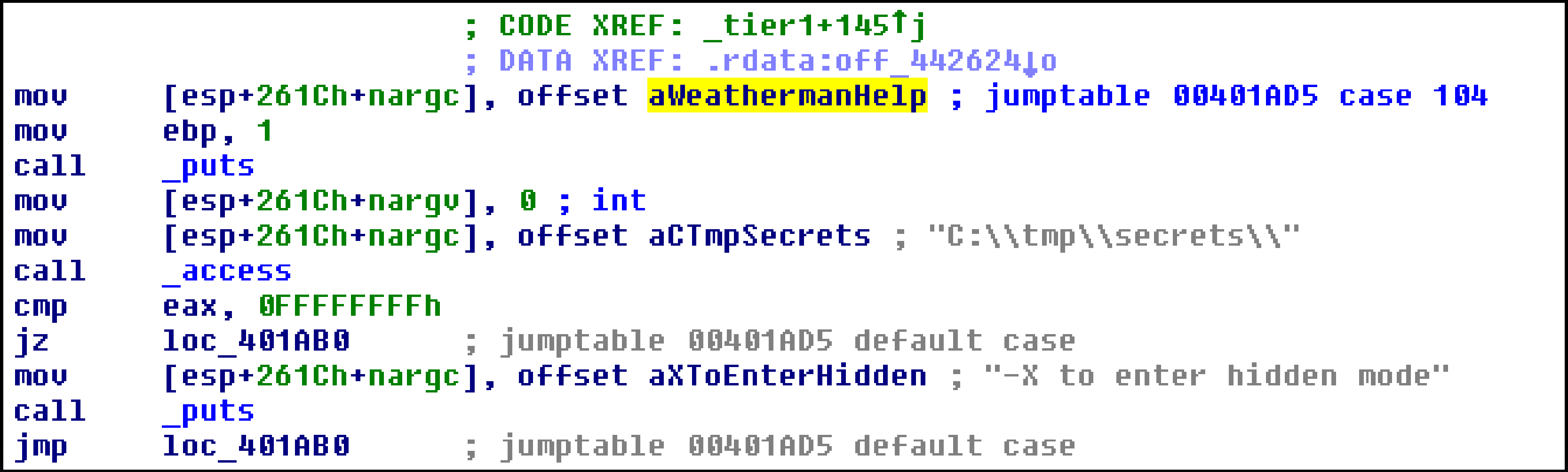
- You should now be looking at disassembled x86 code
  - We just leveraged the fact that in order to use “C:\\tmp\\secrets” in the program, the code had to reference the address in the data section of the program where the string was stored.
- Using xrefs in IDA is a quick and easy way to find interesting code sections

# Double-click Reference (2)

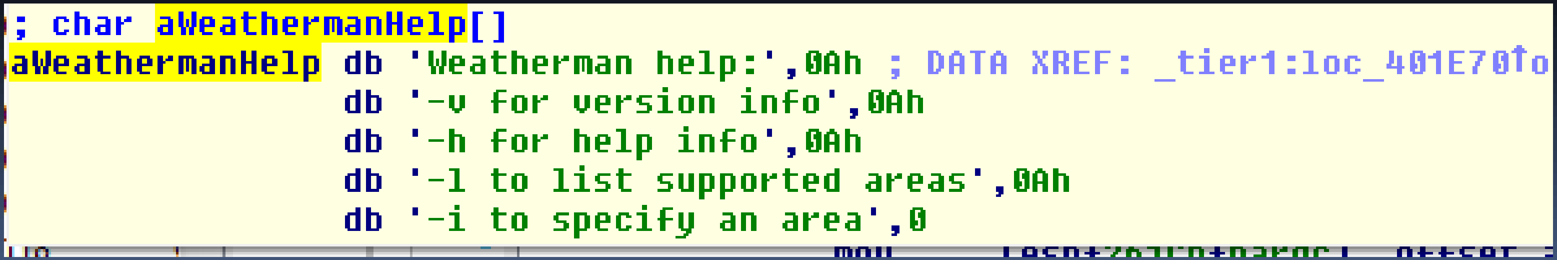


```
                         ; CODE XREF: _tier1+145↑j
                         ; DATA XREF: .rdata:off_442624↓o
mov     [esp+261Ch+nargc], offset aWeathermanHelp ; jumptable 00401AD5 case 104
mov     ebp, 1
call    _puts
mov     [esp+261Ch+nargv], 0 ; int
mov     [esp+261Ch+nargc], offset aCTmpSecrets ; "C:\\tmp\\secrets\\"
call    _access
cmp     eax, 0FFFFFFFFh
jz      loc_401AB0      ; jumptable 00401AD5 default case
mov     [esp+261Ch+nargc], offset aXToEnterHidden ; "-X to enter hidden mode"
call    _puts
jmp     loc_401AB0      ; jumptable 00401AD5 default case
```

# Explore Code Block

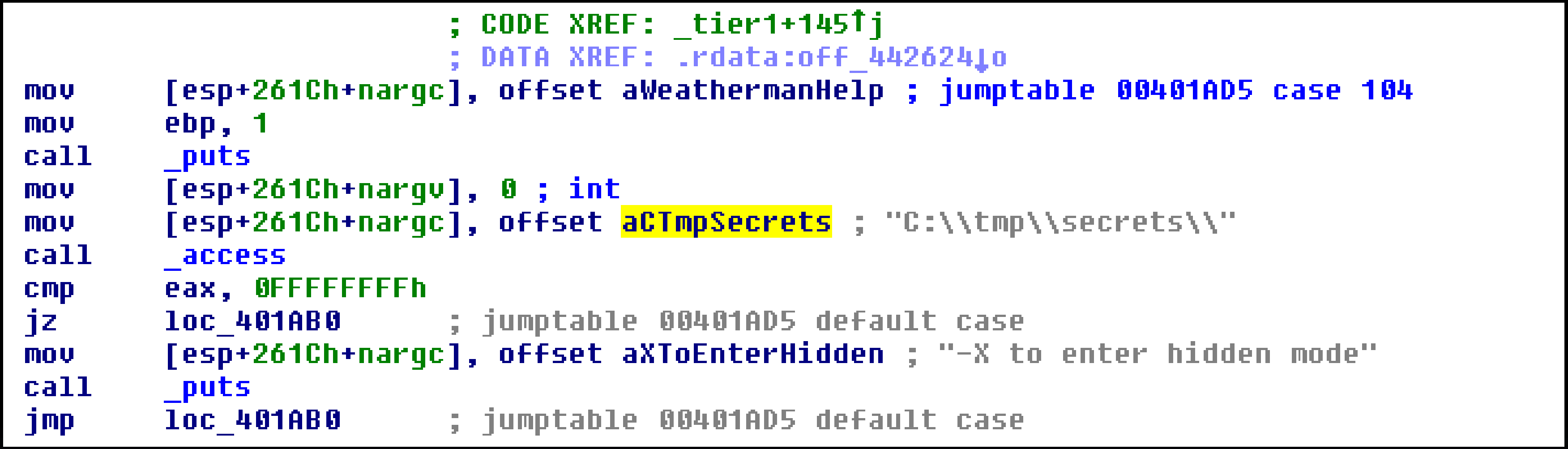


```
                              ; CODE XREF: _tier1+145↑j
                              ; DATA XREF: .rdata:off_442624↓o
mov     [esp+261Ch+nargc], offset aWeathermanHelp ; jumptable 00401AD5 case 104
mov     ebp, 1
call    _puts
mov     [esp+261Ch+nargv], 0 ; int
mov     [esp+261Ch+nargc], offset aCTmpSecrets ; "C:\\tmp\\secrets\\"
call    _access
cmp     eax, 0FFFFFFFFh
jz      loc_401AB0      ; jumptable 00401AD5 default case
mov     [esp+261Ch+nargc], offset aXToEnterHidden ; "-X to enter hidden mode"
call    _puts
jmp     loc_401AB0      ; jumptable 00401AD5 default case
```



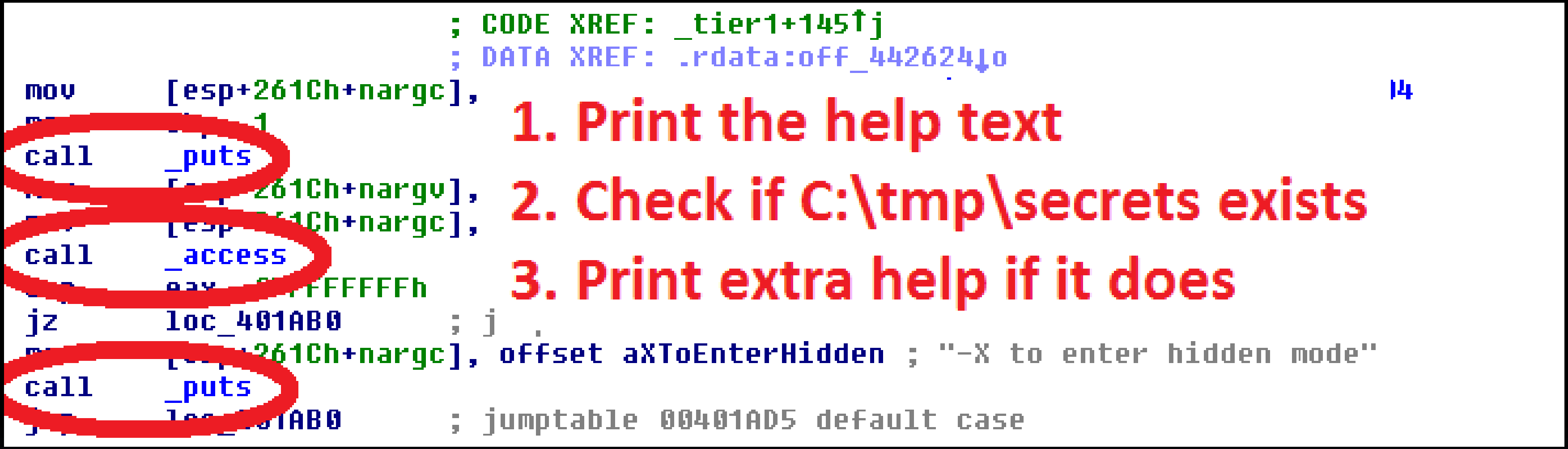
```
; char aWeathermanHelp[]
aWeathermanHelp db 'Weatherman help:',0Ah ; DATA XREF: _tier1:loc_401E70↑o
                db '-v for version info',0Ah
                db '-h for help info',0Ah
                db '-l to list supported areas',0Ah
                db '-i to specify an area',0
```

# Explore Code Block (2)



```
                          ; CODE XREF: _tier1+145↑j
                          ; DATA XREF: .rdata:off_442624↓o
mov     [esp+261Ch+nargc], offset aWeathermanHelp ; jumptable 00401AD5 case 104
mov     ebp, 1
call    _puts
mov     [esp+261Ch+nargv], 0 ; int
mov     [esp+261Ch+nargc], offset aCTmpSecrets ; "C:\\tmp\\secrets\\"
call    _access
cmp     eax, 0FFFFFFFFh
jz      loc_401AB0      ; jumptable 00401AD5 default case
mov     [esp+261Ch+nargc], offset aXToEnterHidden ; "-X to enter hidden mode"
call    _puts
jmp     loc_401AB0      ; jumptable 00401AD5 default case
```

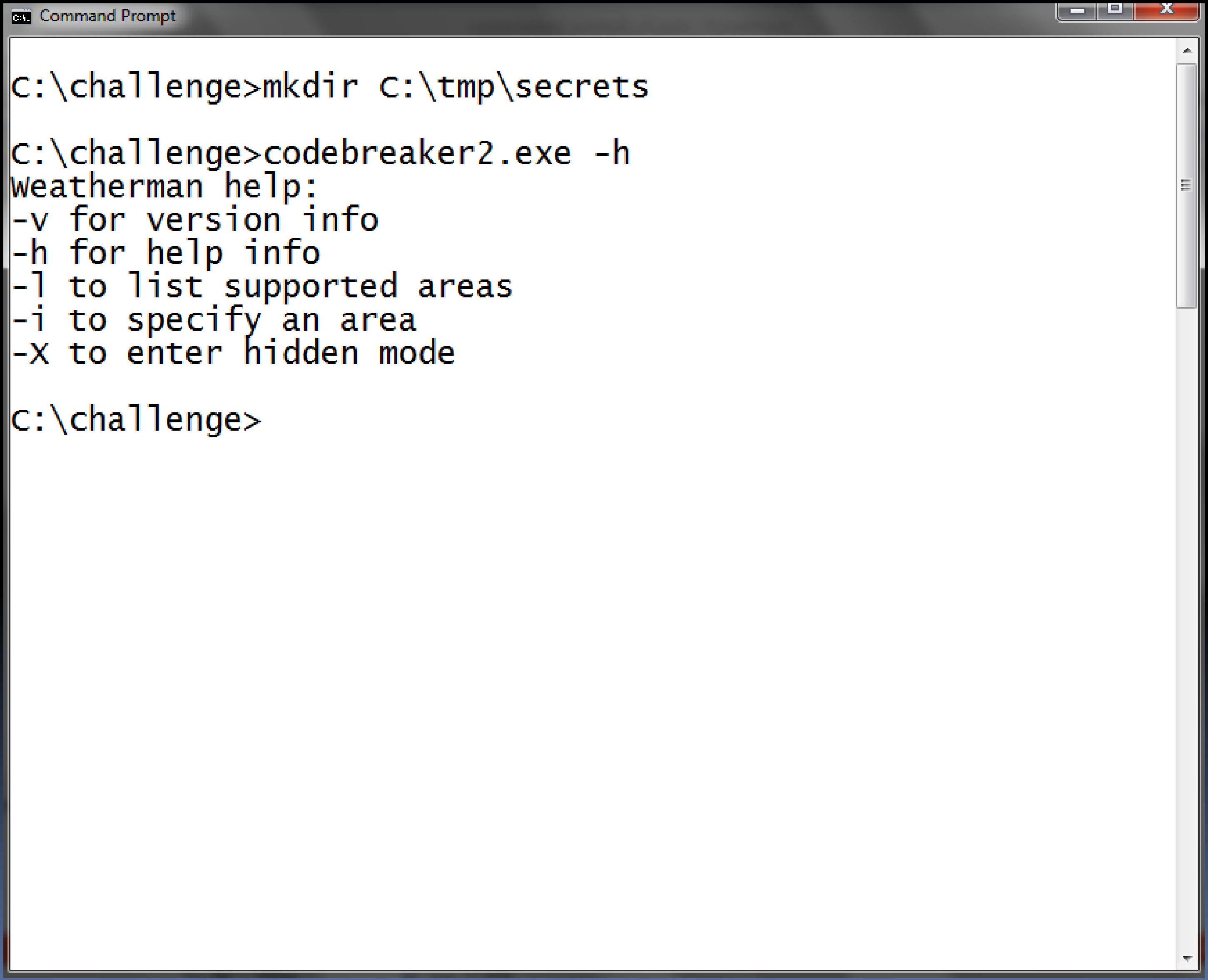
## Explore Code Block (3)



```
                         ; CODE XREF: _tier1+145↑j
                         ; DATA XREF: .rdata:off_442624↓o
mov     [esp+261Ch+nargc],                                                    14
m[illegible]    [illegible] 1
call    _puts
m[illegible]    [[illegible] 261Ch+nargv],
m[illegible]    [esp [illegible]1Ch+nargc],
call    _access
c[illegible]    eax, [illegible]FFFFFFh
jz      loc_401AB0       ; j
m[illegible]    [[illegible]261Ch+nargc], offset aXToEnterHidden ; "-X to enter hidden mode"
call    _puts
j[illegible]    loc_[illegible]1AB0       ; jumptable 00401AD5 default case
```

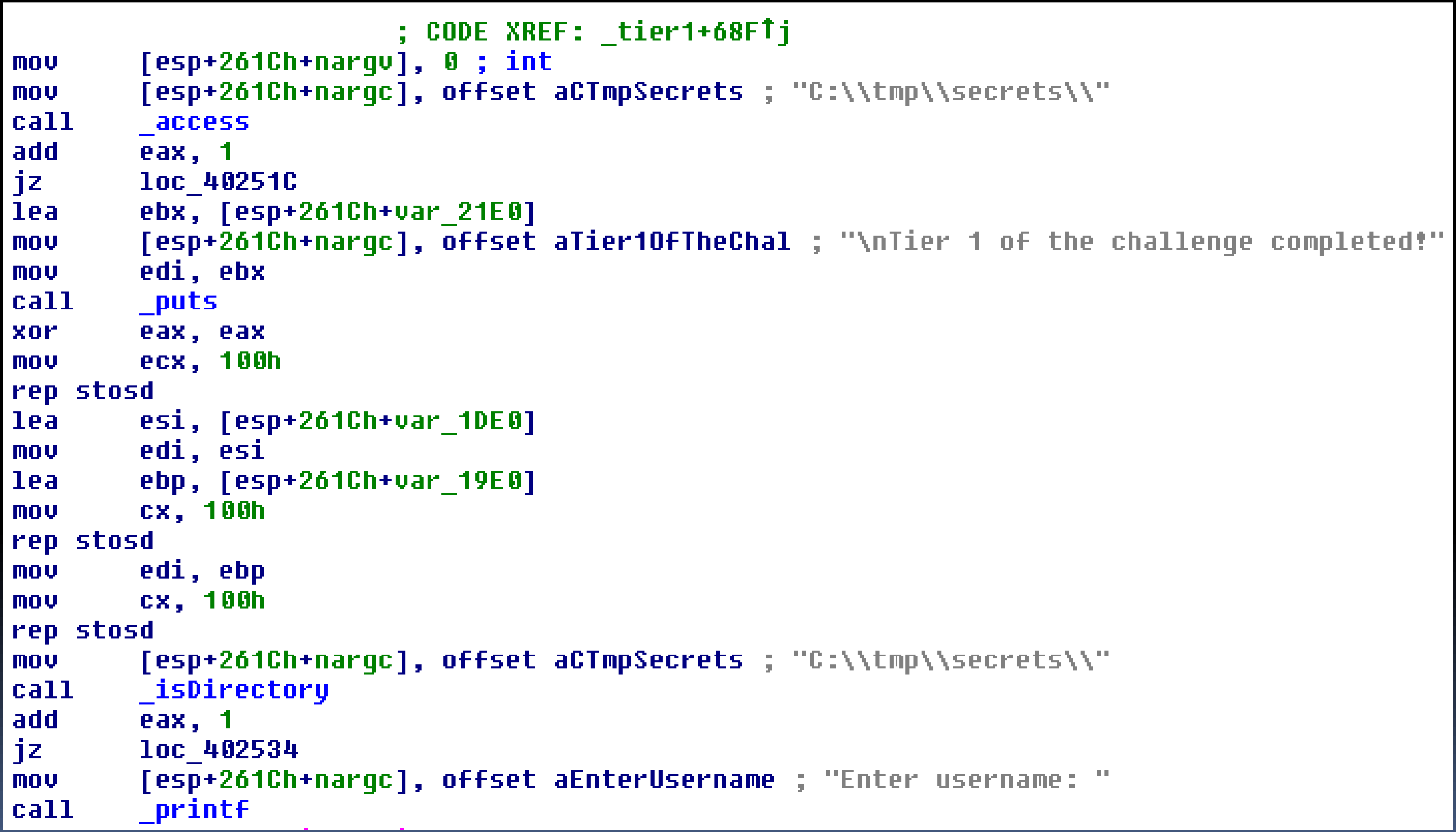
1. Print the help text
2. Check if C:\tmp\secrets exists
3. Print extra help if it does

## Running the program (for real)



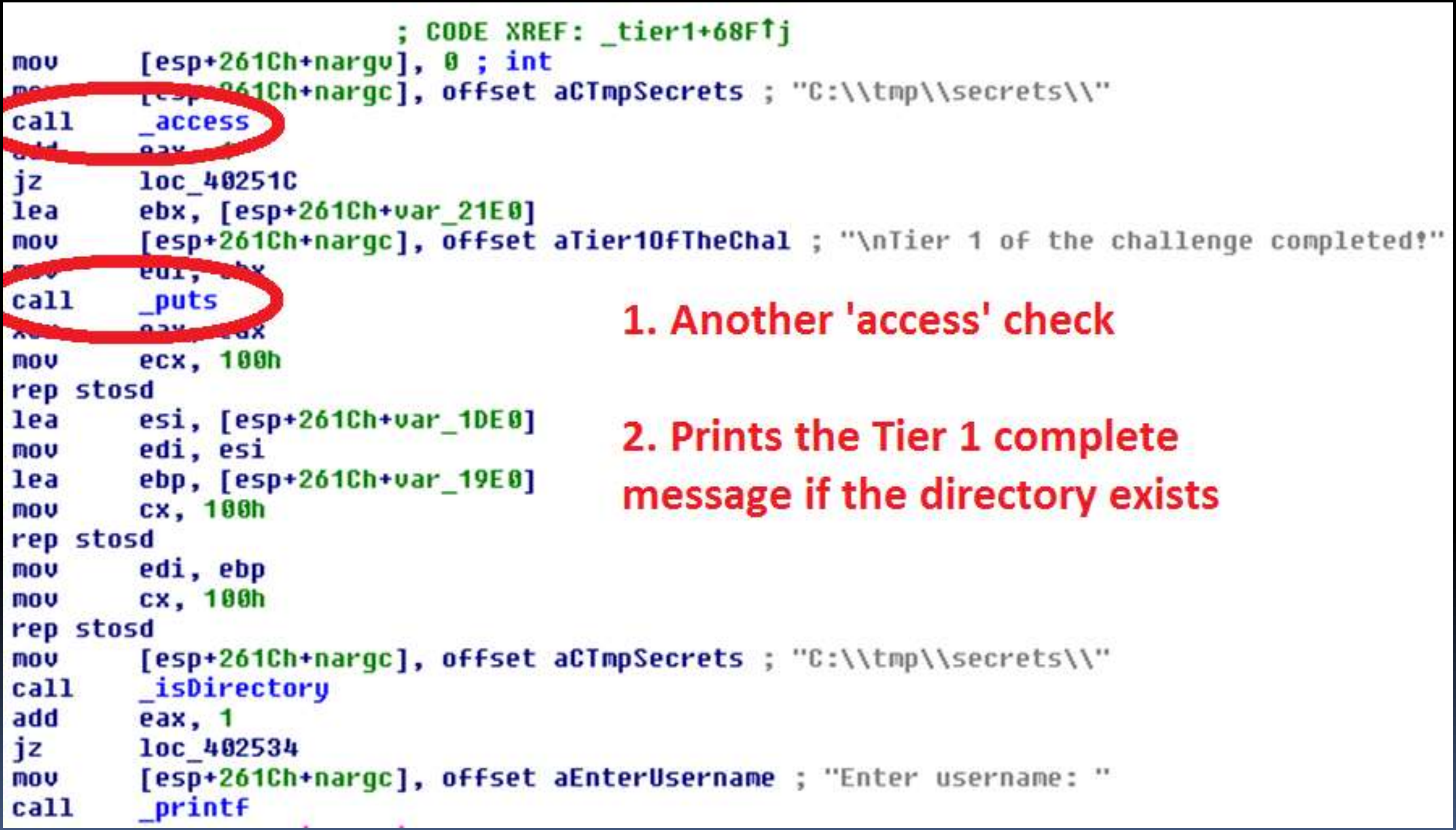
```
C:\challenge>mkdir C:\tmp\secrets

C:\challenge>codebreaker2.exe -h
Weatherman help:
-v for version info
-h for help info
-l to list supported areas
-i to specify an area
-X to enter hidden mode

C:\challenge>
```

# Explore Code Block (4)



```
                        ; CODE XREF: _tier1+68F↑j
mov     [esp+261Ch+nargv], 0 ; int
mov     [esp+261Ch+nargc], offset aCTmpSecrets ; "C:\\tmp\\secrets\\"
call    _access
add     eax, 1
jz      loc_40251C
lea     ebx, [esp+261Ch+var_21E0]
mov     [esp+261Ch+nargc], offset aTier1OfTheChal ; "\nTier 1 of the challenge completed!"
mov     edi, ebx
call    _puts
xor     eax, eax
mov     ecx, 100h
rep stosd
lea     esi, [esp+261Ch+var_1DE0]
mov     edi, esi
lea     ebp, [esp+261Ch+var_19E0]
mov     cx, 100h
rep stosd
mov     edi, ebp
mov     cx, 100h
rep stosd
mov     [esp+261Ch+nargc], offset aCTmpSecrets ; "C:\\tmp\\secrets\\"
call    _isDirectory
add     eax, 1
jz      loc_402534
mov     [esp+261Ch+nargc], offset aEnterUsername ; "Enter username: "
call    _printf
```

# Explore Code Block (5)

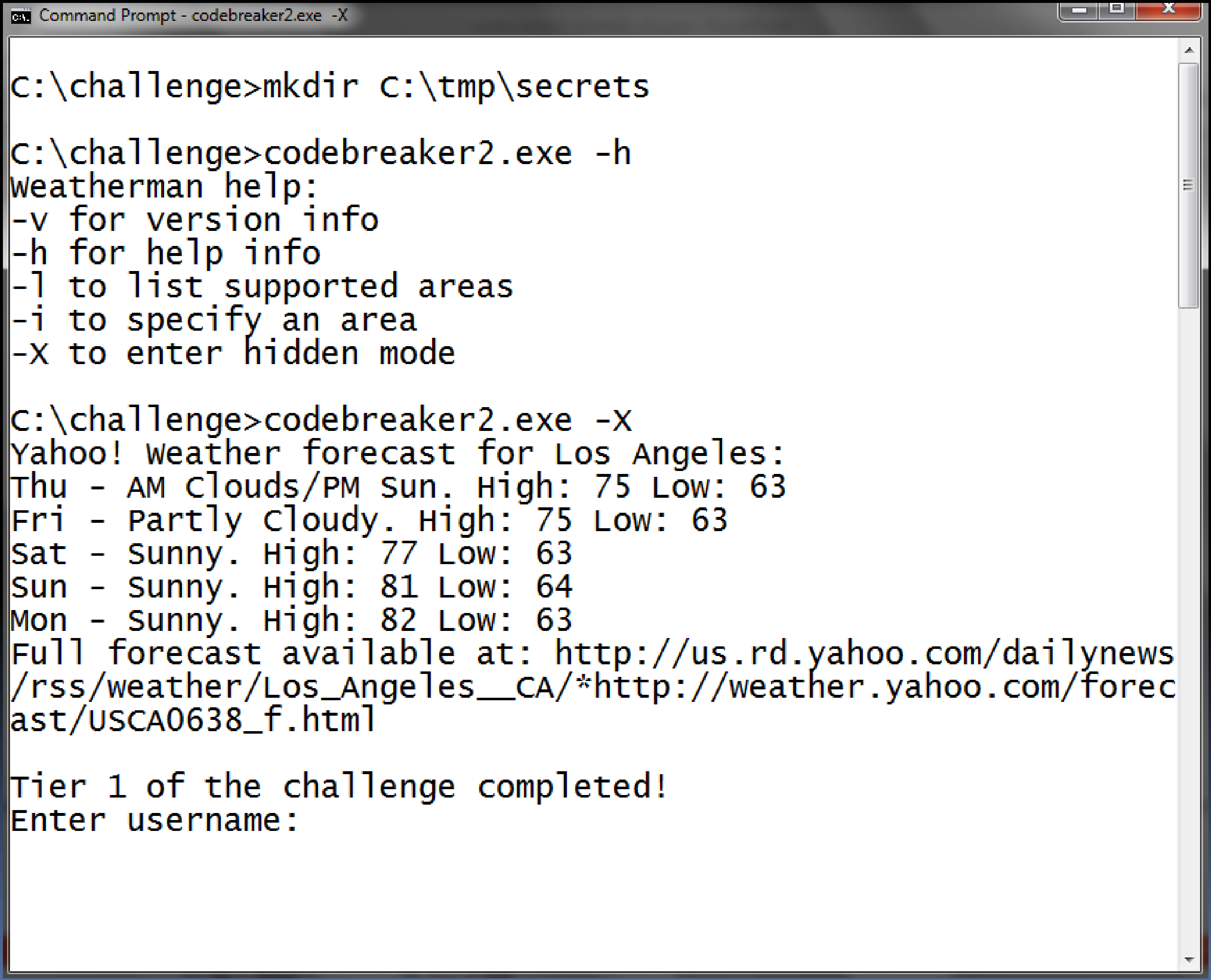


```
                         ; CODE XREF: _tier1+68F↑j
mov     [esp+261Ch+nargv], 0 ; int
mov     [esp+261Ch+nargc], offset aCTmpSecrets ; "C:\\tmp\\secrets\\"
call    _access
add     eax, 1
jz      loc_40251C
lea     ebx, [esp+261Ch+var_21E0]
mov     [esp+261Ch+nargc], offset aTier1OfTheChal ; "\nTier 1 of the challenge completed!"
mov     edi, ebx
call    _puts
xor     eax, eax
mov     ecx, 100h
rep stosd
lea     esi, [esp+261Ch+var_1DE0]
mov     edi, esi
lea     ebp, [esp+261Ch+var_19E0]
mov     cx, 100h
rep stosd
mov     edi, ebp
mov     cx, 100h
rep stosd
mov     [esp+261Ch+nargc], offset aCTmpSecrets ; "C:\\tmp\\secrets\\"
call    _isDirectory
add     eax, 1
jz      loc_402534
mov     [esp+261Ch+nargc], offset aEnterUsername ; "Enter username: "
call    _printf
```

1. Another 'access' check

2. Prints the Tier 1 complete message if the directory exists

## Running the program (for real)(2)



Command Prompt - codebreaker2.exe -X

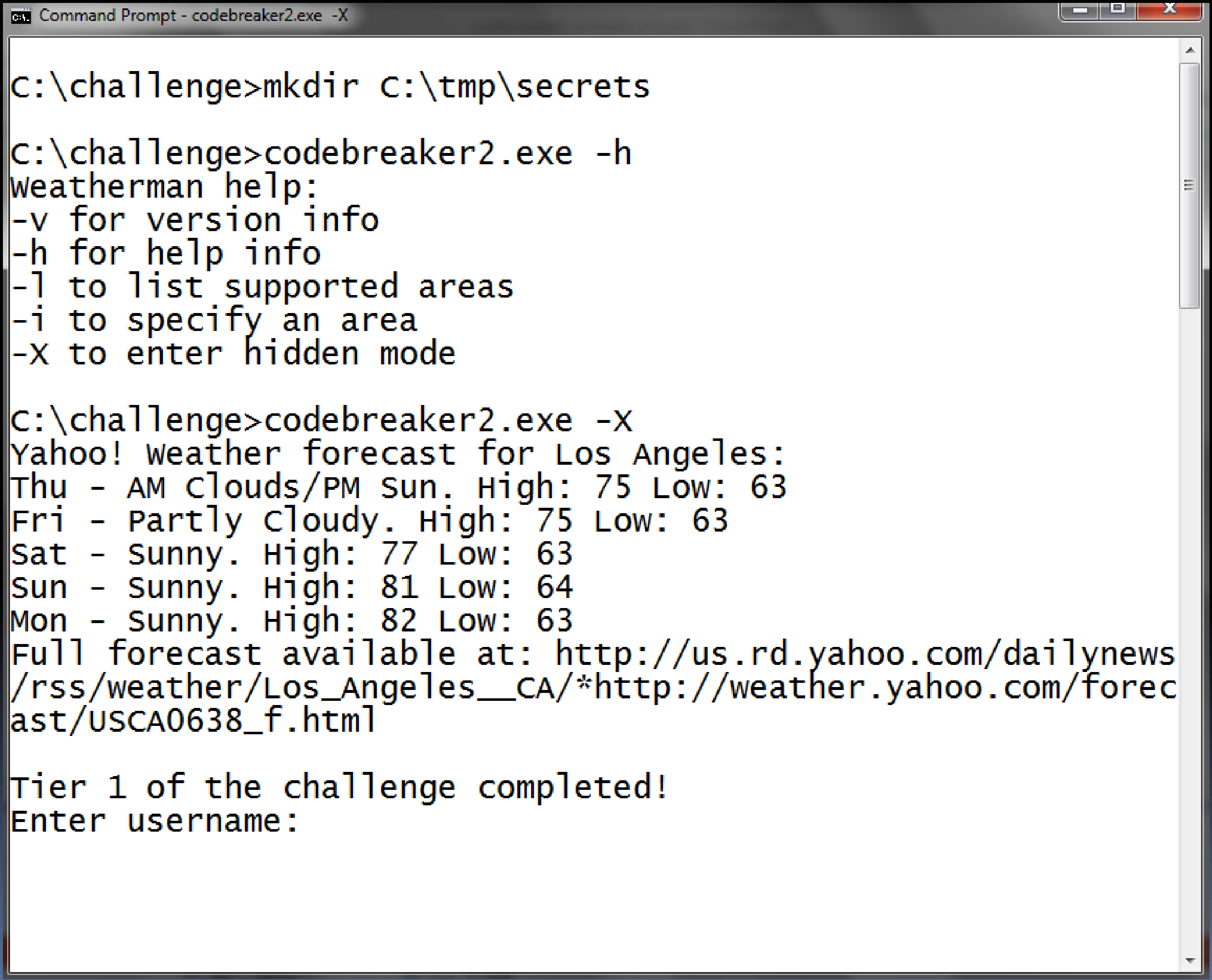
```
C:\challenge>mkdir C:\tmp\secrets

C:\challenge>codebreaker2.exe -h
Weatherman help:
-v for version info
-h for help info
-l to list supported areas
-i to specify an area
-X to enter hidden mode

C:\challenge>codebreaker2.exe -X
Yahoo! Weather forecast for Los Angeles:
Thu - AM Clouds/PM Sun. High: 75 Low: 63
Fri - Partly Cloudy. High: 75 Low: 63
Sat - Sunny. High: 77 Low: 63
Sun - Sunny. High: 81 Low: 64
Mon - Sunny. High: 82 Low: 63
Full forecast available at: http://us.rd.yahoo.com/dailynews
/rss/weather/Los_Angeles__CA/*http://weather.yahoo.com/forec
ast/USCA0638_f.html

Tier 1 of the challenge completed!
Enter username:
```

## Tier 1 Complete!

- Pretty straight forward
- Just looking at the strings may have been enough to get you through this

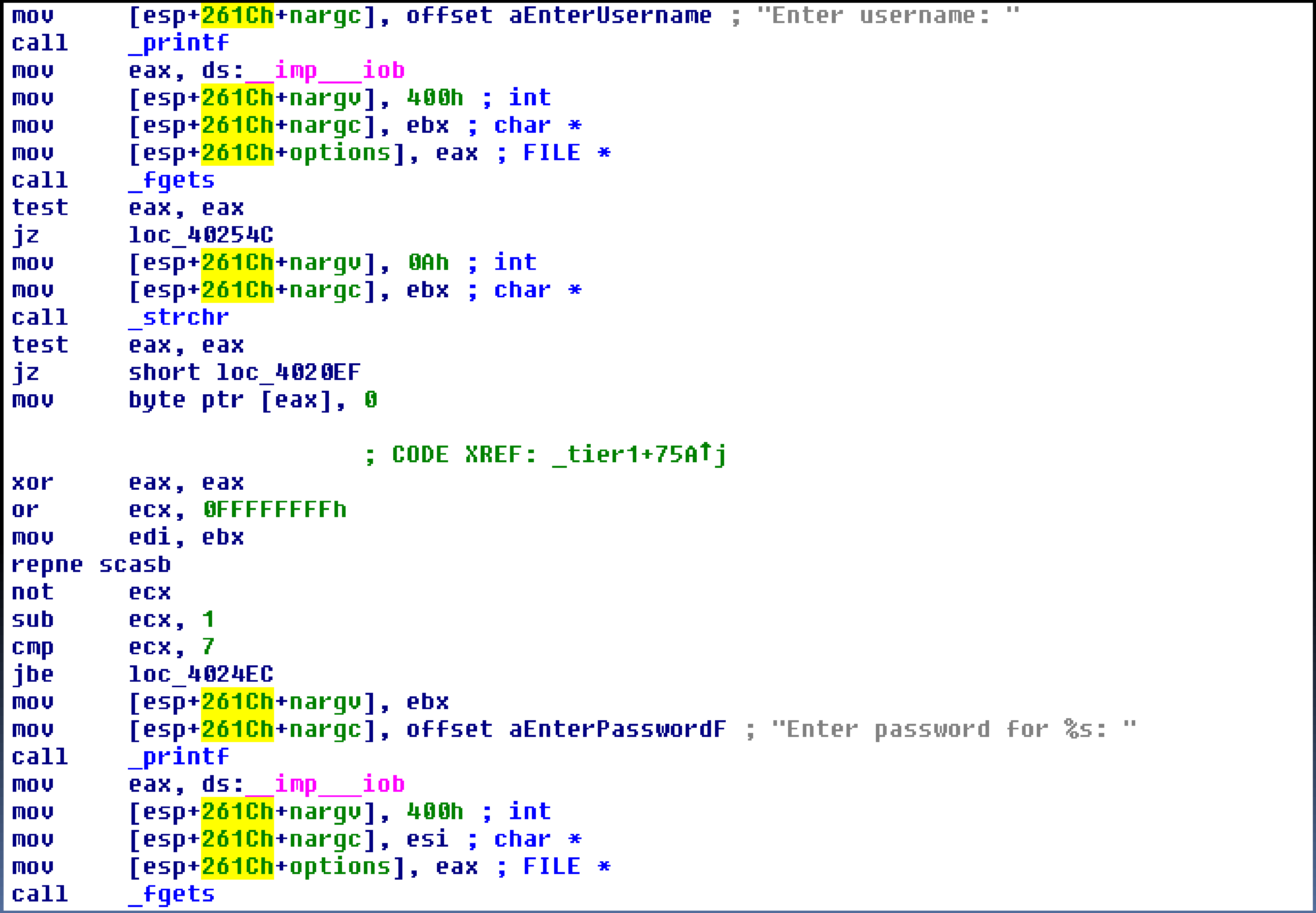
- … on to Tier 2!

## Running the program (for real)(2)



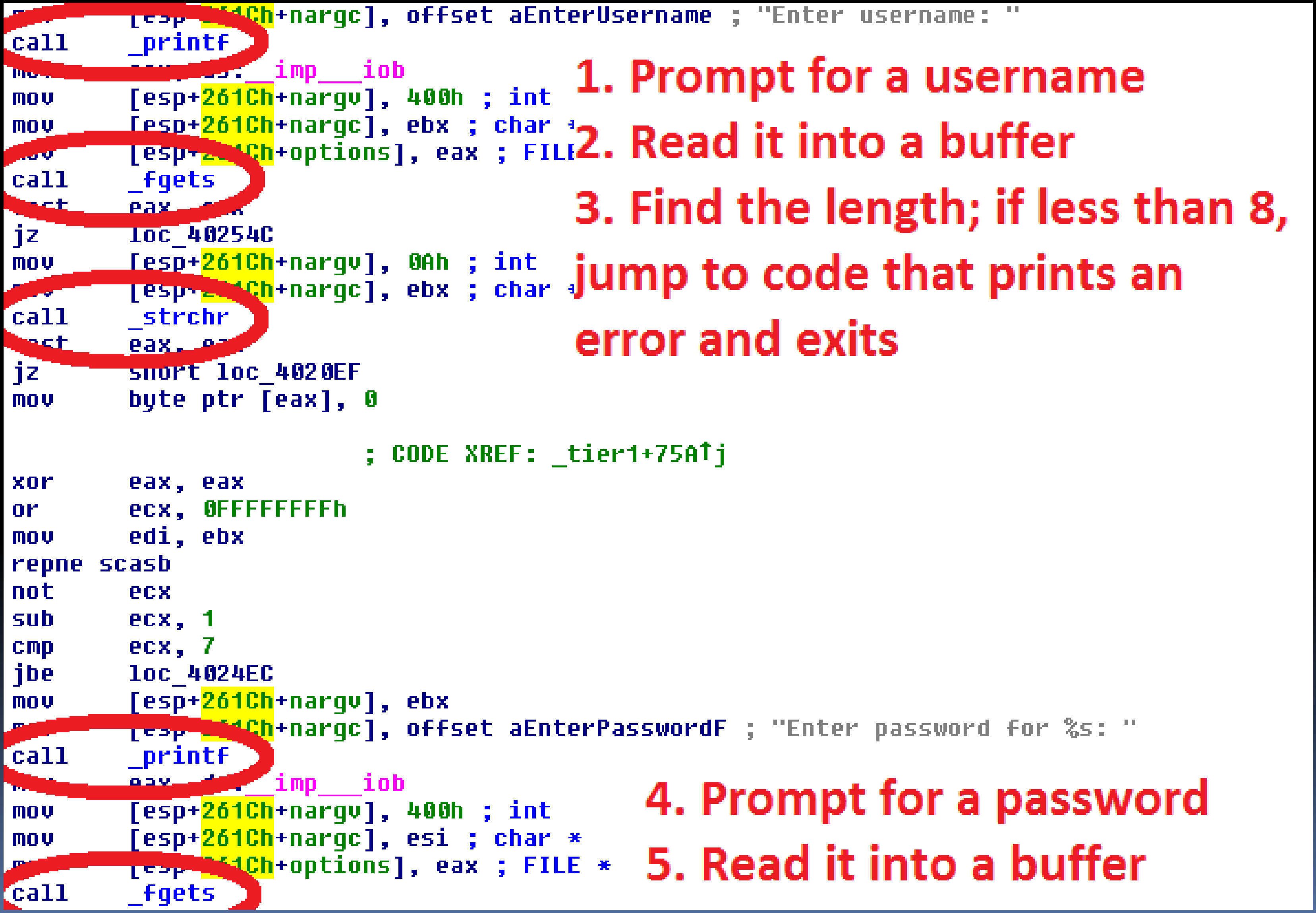
```
C:\challenge>mkdir C:\tmp\secrets

C:\challenge>codebreaker2.exe -h
Weatherman help:
-v for version info
-h for help info
-l to list supported areas
-i to specify an area
-X to enter hidden mode

C:\challenge>codebreaker2.exe -X
Yahoo! Weather forecast for Los Angeles:
Thu - AM Clouds/PM Sun. High: 75 Low: 63
Fri - Partly Cloudy. High: 75 Low: 63
Sat - Sunny. High: 77 Low: 63
Sun - Sunny. High: 81 Low: 64
Mon - Sunny. High: 82 Low: 63
Full forecast available at: http://us.rd.yahoo.com/dailynews
/rss/weather/Los_Angeles__CA/*http://weather.yahoo.com/forec
ast/USCA0638_f.html

Tier 1 of the challenge completed!
Enter username:
```

# Explore Code Block (6)



```
mov     [esp+261Ch+nargc], offset aEnterUsername ; "Enter username: "
call    _printf
mov     eax, ds:__imp___iob
mov     [esp+261Ch+nargv], 400h ; int
mov     [esp+261Ch+nargc], ebx ; char *
mov     [esp+261Ch+options], eax ; FILE *
call    _fgets
test    eax, eax
jz      loc_40254C
mov     [esp+261Ch+nargv], 0Ah ; int
mov     [esp+261Ch+nargc], ebx ; char *
call    _strchr
test    eax, eax
jz      short loc_4020EF
mov     byte ptr [eax], 0

                         ; CODE XREF: _tier1+75A↑j
xor     eax, eax
or      ecx, 0FFFFFFFFh
mov     edi, ebx
repne scasb
not     ecx
sub     ecx, 1
cmp     ecx, 7
jbe     loc_4024EC
mov     [esp+261Ch+nargv], ebx
mov     [esp+261Ch+nargc], offset aEnterPasswordF ; "Enter password for %s: "
call    _printf
mov     eax, ds:__imp___iob
mov     [esp+261Ch+nargv], 400h ; int
mov     [esp+261Ch+nargc], esi ; char *
mov     [esp+261Ch+options], eax ; FILE *
call    _fgets
```

# Explore Code Block (7)

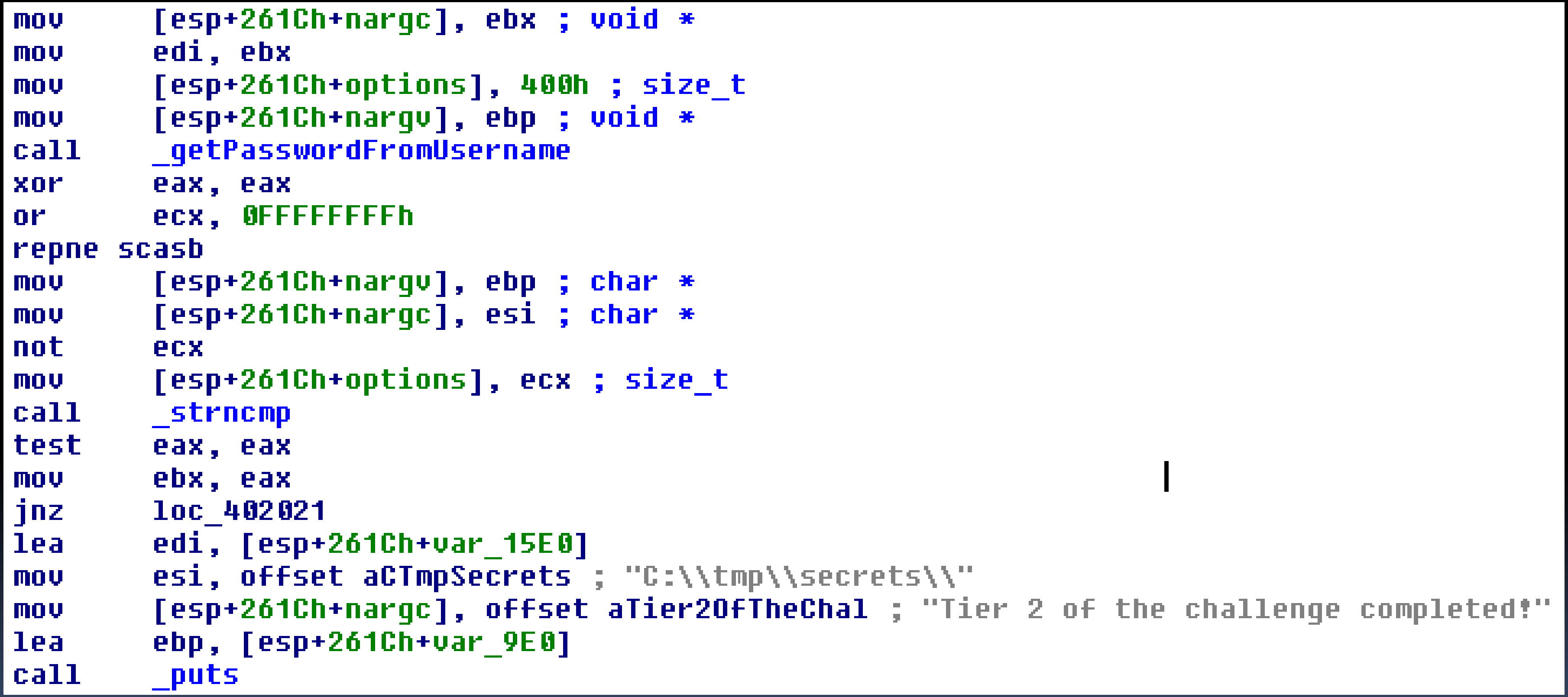


```
mov     [esp+261Ch+nargc], offset aEnterUsername ; "Enter username: "
call    _printf
mov     eax, ds:__imp___iob
mov     [esp+261Ch+nargv], 400h ; int
mov     [esp+261Ch+nargc], ebx ; char *
mov     [esp+261Ch+options], eax ; FILE *
call    _fgets
test    eax, eax
jz      loc_40254C
mov     [esp+261Ch+nargv], 0Ah ; int
mov     [esp+261Ch+nargc], ebx ; char *
call    _strchr
test    eax, eax
jz      short loc_4020EF
mov     byte ptr [eax], 0

                        ; CODE XREF: _tier1+75A↑j
xor     eax, eax
or      ecx, 0FFFFFFFFh
mov     edi, ebx
repne scasb
not     ecx
sub     ecx, 1
cmp     ecx, 7
jbe     loc_4024EC
mov     [esp+261Ch+nargv], ebx
mov     [esp+261Ch+nargc], offset aEnterPasswordF ; "Enter password for %s: "
call    _printf
mov     eax, ds:__imp___iob
mov     [esp+261Ch+nargv], 400h ; int
mov     [esp+261Ch+nargc], esi ; char *
mov     [esp+261Ch+options], eax ; FILE *
call    _fgets
```

1. Prompt for a username
2. Read it into a buffer
3. Find the length; if less than 8, jump to code that prints an error and exits

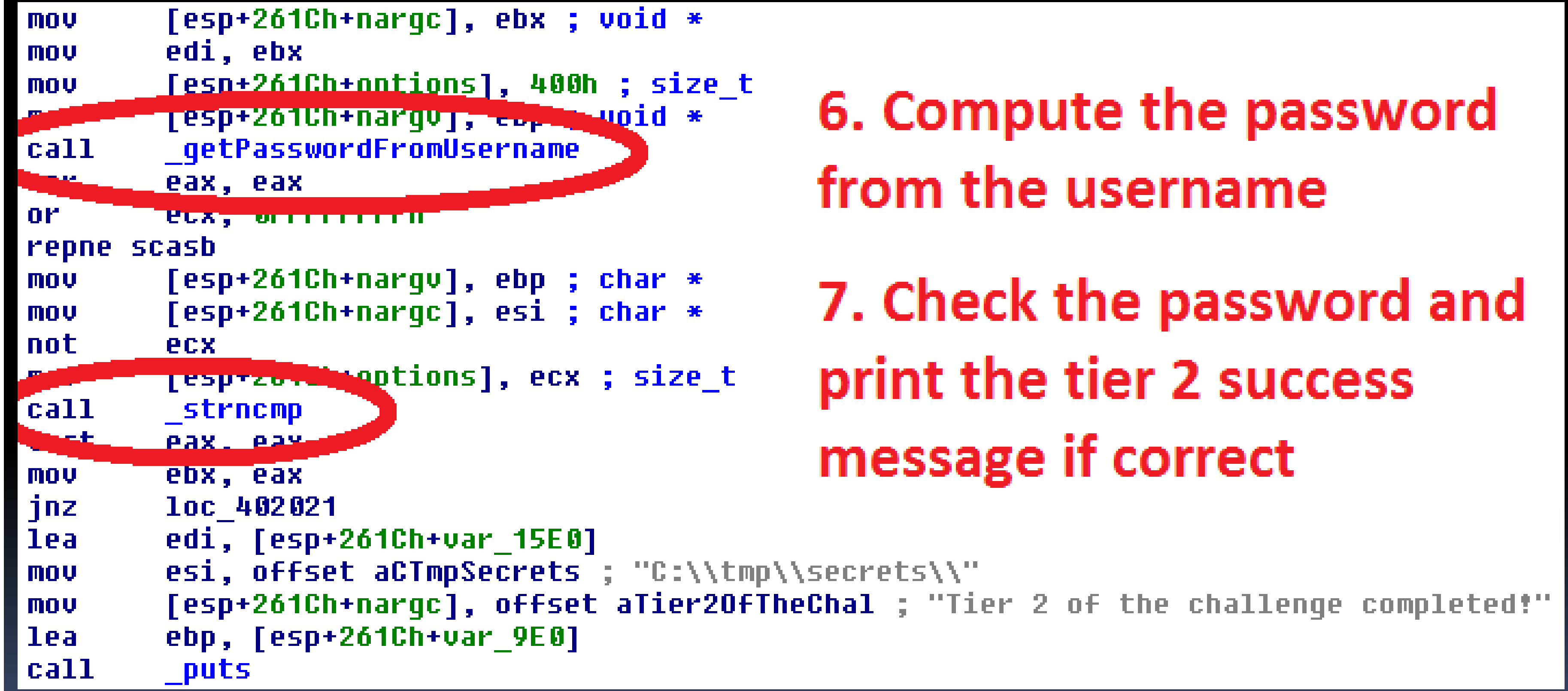
4. Prompt for a password
5. Read it into a buffer

# Explore Code Block (8)



```
mov     [esp+261Ch+nargc], ebx ; void *
mov     edi, ebx
mov     [esp+261Ch+options], 400h ; size_t
mov     [esp+261Ch+nargv], ebp ; void *
call    _getPasswordFromUsername
xor     eax, eax
or      ecx, 0FFFFFFFFh
repne scasb
mov     [esp+261Ch+nargv], ebp ; char *
mov     [esp+261Ch+nargc], esi ; char *
not     ecx
mov     [esp+261Ch+options], ecx ; size_t
call    _strncmp
test    eax, eax
mov     ebx, eax
jnz     loc_402021
lea     edi, [esp+261Ch+var_15E0]
mov     esi, offset aCTmpSecrets ; "C:\\tmp\\secrets\\"
mov     [esp+261Ch+nargc], offset aTier2OfTheChal ; "Tier 2 of the challenge completed!"
lea     ebp, [esp+261Ch+var_9E0]
call    _puts
```

# Explore Code Block (9)

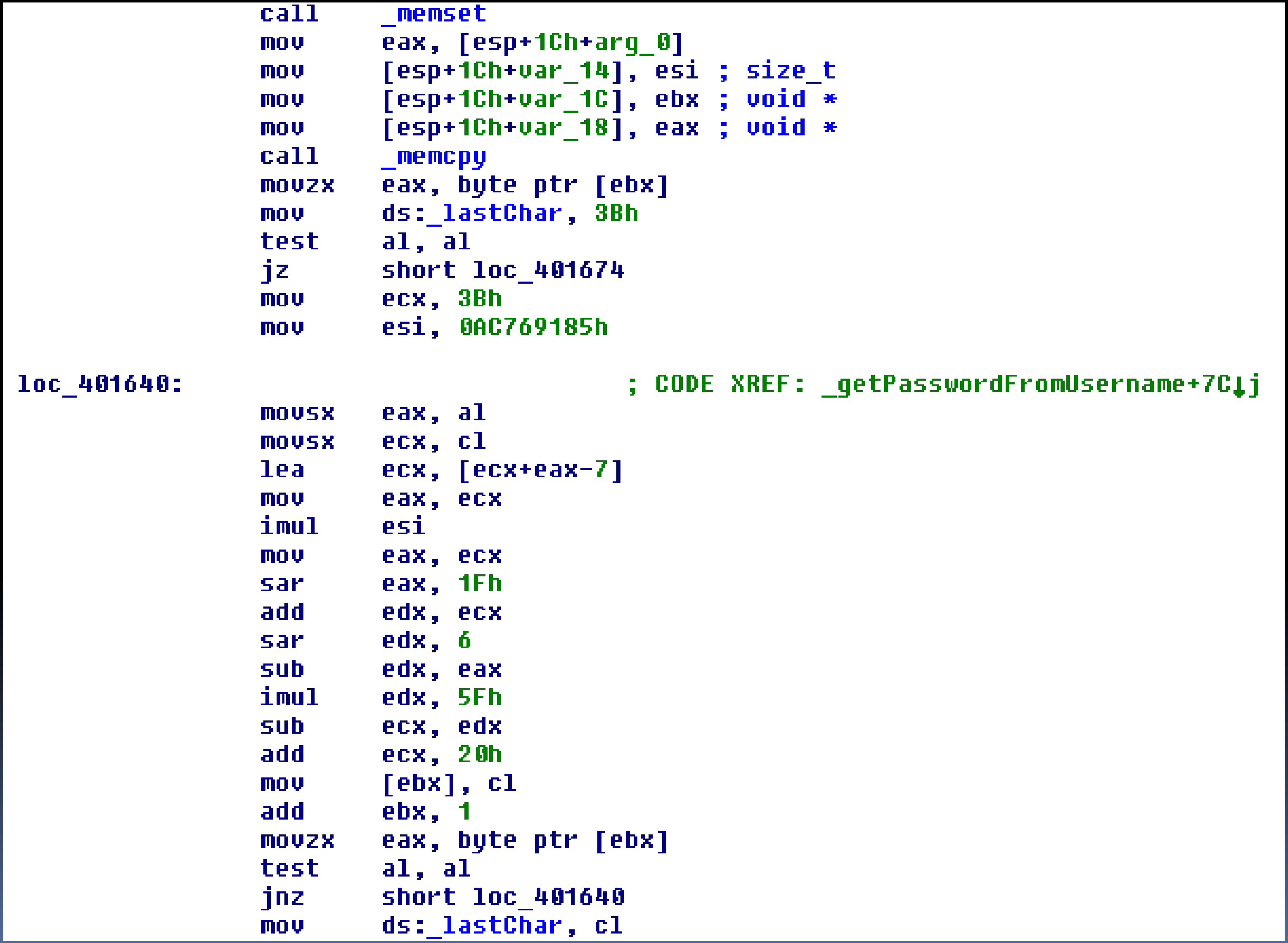


```
mov     [esp+261Ch+nargc], ebx ; void *
mov     edi, ebx
mov     [esp+261Ch+options], 400h ; size_t
mov     [esp+261Ch+nargv], ebp ; void *
call    _getPasswordFromUsername
xor     eax, eax
or      ecx, 0FFFFFFFFh
repne scasb
mov     [esp+261Ch+nargv], ebp ; char *
mov     [esp+261Ch+nargc], esi ; char *
not     ecx
mov     [esp+261Ch+options], ecx ; size_t
call    _strncmp
test    eax, eax
mov     ebx, eax
jnz     loc_402021
lea     edi, [esp+261Ch+var_15E0]
mov     esi, offset aCTmpSecrets ; "C:\\tmp\\secrets\\"
mov     [esp+261Ch+nargc], offset aTier2OfTheChal ; "Tier 2 of the challenge completed!"
lea     ebp, [esp+261Ch+var_9E0]
call    _puts
```

6. Compute the password from the username

7. Check the password and print the tier 2 success message if correct

# getPasswordFromUsername



```
                call    _memset
                mov     eax, [esp+1Ch+arg_0]
                mov     [esp+1Ch+var_14], esi ; size_t
                mov     [esp+1Ch+var_1C], ebx ; void *
                mov     [esp+1Ch+var_18], eax ; void *
                call    _memcpy
                movzx   eax, byte ptr [ebx]
                mov     ds:_lastChar, 3Bh
                test    al, al
                jz      short loc_401674
                mov     ecx, 3Bh
                mov     esi, 0AC769185h

loc_401640:                             ; CODE XREF: _getPasswordFromUsername+7C↓j
                movsx   eax, al
                movsx   ecx, cl
                lea     ecx, [ecx+eax-7]
                mov     eax, ecx
                imul    esi
                mov     eax, ecx
                sar     eax, 1Fh
                add     edx, ecx
                sar     edx, 6
                sub     edx, eax
                imul    edx, 5Fh
                sub     ecx, edx
                add     ecx, 20h
                mov     [ebx], cl
                add     ebx, 1
                movzx   eax, byte ptr [ebx]
                test    al, al
                jnz     short loc_401640
                mov     ds:_lastChar, cl
```

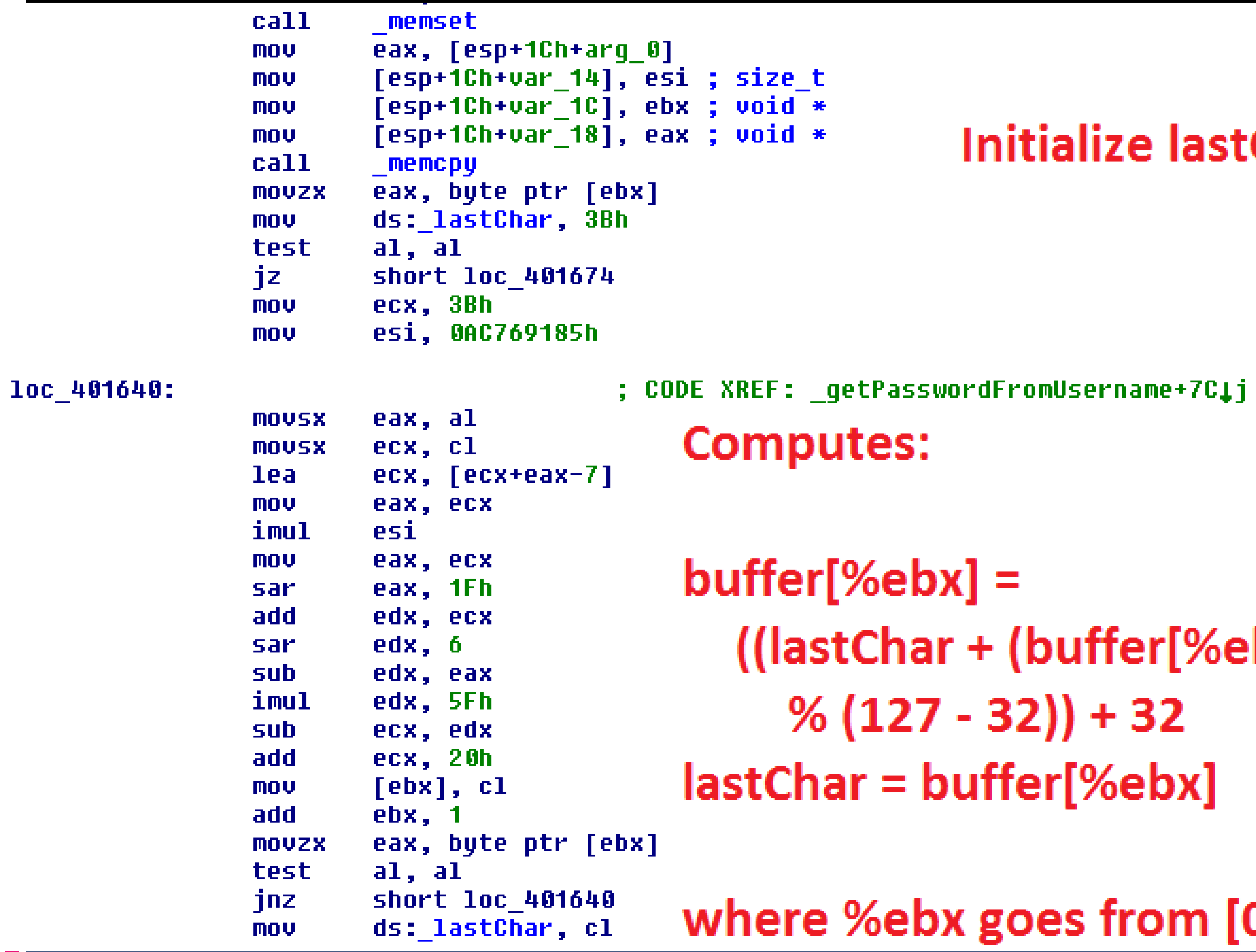
## What does this code do?

```
mov edx, 0xAC769185          // edx = 0xAC769185
mov eax, ecx                 // ecx = input value
imul edx                     // edx:eax = eax * edx
lea eax, [edx + ecx*0x1]     // eax = edx + ecx
mov edx, eax                 // edx = eax
sar edx, 0x6                 // arith right shift; edx = edx >> 0x6
mov eax, ecx                 // eax = ecx
sar eax, 0x1f                // eax = eax >> 0x1f (31)
mov ebx, edx                 // ebx = edx
sub ebx, eax                 // ebx = ebx - eax
mov eax, ebx                 // eax = ebx
imul eax, eax, 0x5f          // edx:eax = eax * 0x5f (95)
mov edx, ecx                 // edx = ecx
sub edx, eax                 // edx = edx – eax
                             // edx is the final result
```

## Signed Division and Remainder

- The code computes: edx = ecx % 95
- Why multiply by 0xAC769185 and where did that number come from?
  - Division is a time consuming operation
  - When the divisor is a constant, the compiler can optimize the computation
- The basic trick is to multiply by a "magic value" (~ $2^{32}/d$) and extract the leftmost 32 bits of the product
- The following site computes these numbers for you: http://www.hackersdelight.org/magic.htm

# getPasswordFromUsername (2)



```
                call    _memset
                mov     eax, [esp+1Ch+arg_0]
                mov     [esp+1Ch+var_14], esi ; size_t
                mov     [esp+1Ch+var_1C], ebx ; void *
                mov     [esp+1Ch+var_18], eax ; void *
                call    _memcpy
                movzx   eax, byte ptr [ebx]
                mov     ds:_lastChar, 3Bh
                test    al, al
                jz      short loc_401674
                mov     ecx, 3Bh
                mov     esi, 0AC769185h

loc_401640:                             ; CODE XREF: _getPasswordFromUsername+7C↓j
                movsx   eax, al
                movsx   ecx, cl
                lea     ecx, [ecx+eax-7]
                mov     eax, ecx
                imul    esi
                mov     eax, ecx
                sar     eax, 1Fh
                add     edx, ecx
                sar     edx, 6
                sub     edx, eax
                imul    edx, 5Fh
                sub     ecx, edx
                add     ecx, 20h
                mov     [ebx], cl
                add     ebx, 1
                movzx   eax, byte ptr [ebx]
                test    al, al
                jnz     short loc_401640
                mov     ds:_lastChar, cl
```

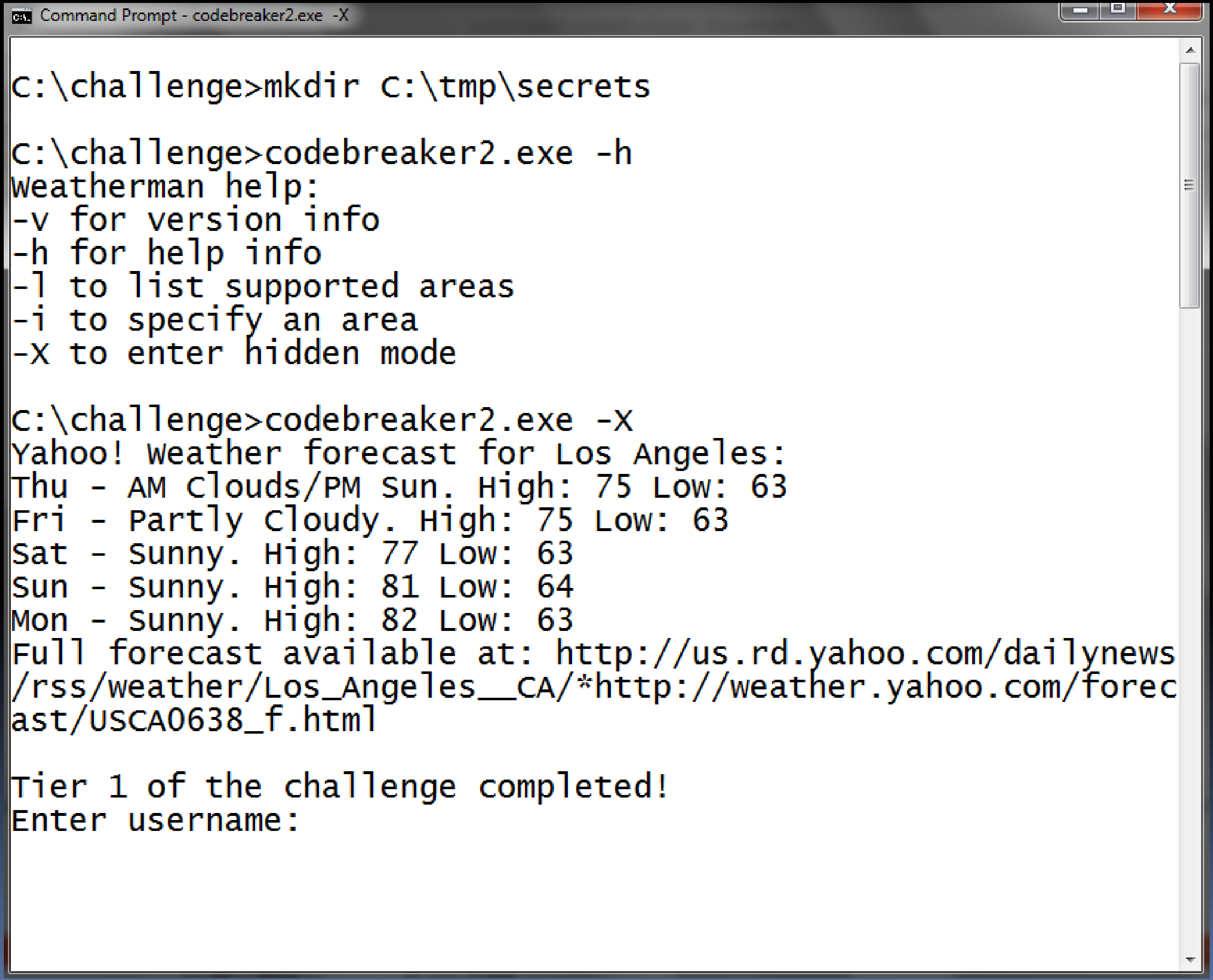
Initialize lastChar to 0x3B

Computes:

buffer[%ebx] =
((lastChar + (buffer[%ebx] - 7))
% (127 - 32)) + 32
lastChar = buffer[%ebx]

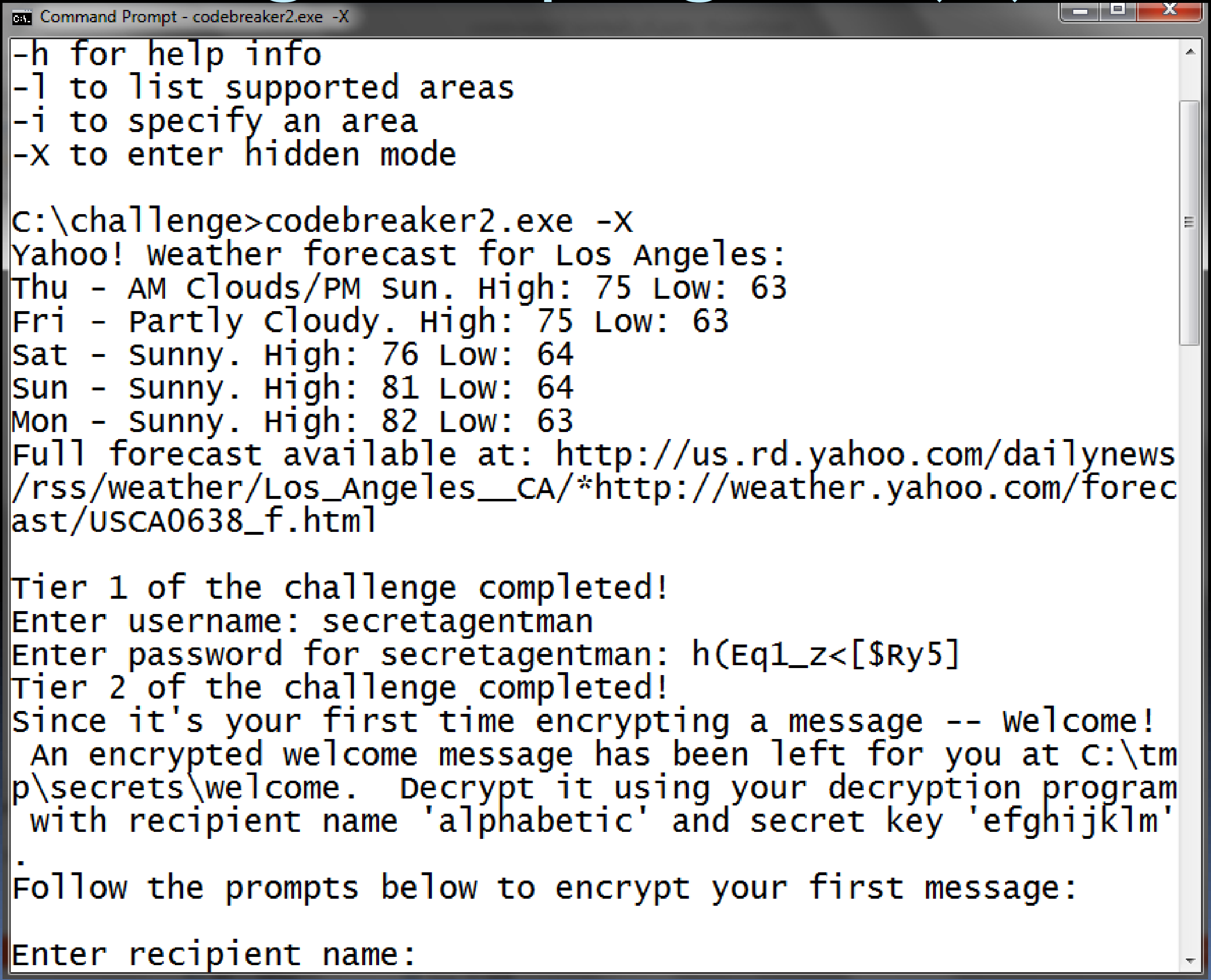
where %ebx goes from [0:passwordLength)

## Running the program (3)



```
C:\challenge>mkdir C:\tmp\secrets

C:\challenge>codebreaker2.exe -h
Weatherman help:
-v for version info
-h for help info
-l to list supported areas
-i to specify an area
-X to enter hidden mode

C:\challenge>codebreaker2.exe -X
Yahoo! Weather forecast for Los Angeles:
Thu - AM Clouds/PM Sun. High: 75 Low: 63
Fri - Partly Cloudy. High: 75 Low: 63
Sat - Sunny. High: 77 Low: 63
Sun - Sunny. High: 81 Low: 64
Mon - Sunny. High: 82 Low: 63
Full forecast available at: http://us.rd.yahoo.com/dailynews
/rss/weather/Los_Angeles__CA/*http://weather.yahoo.com/forec
ast/USCA0638_f.html

Tier 1 of the challenge completed!
Enter username:
```

## Running the program (4)

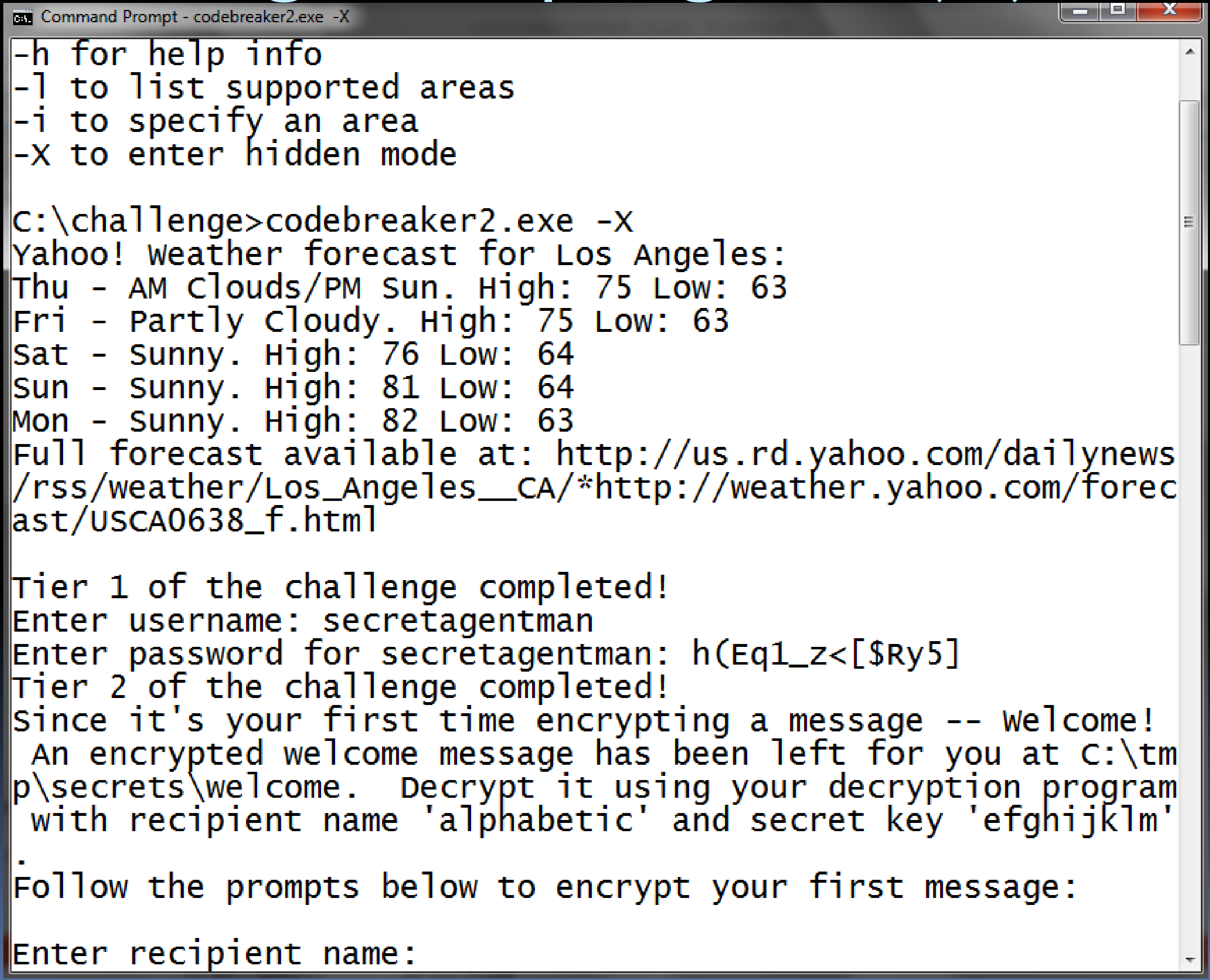


```
-h for help info
-l to list supported areas
-i to specify an area
-X to enter hidden mode

C:\challenge>codebreaker2.exe -X
Yahoo! Weather forecast for Los Angeles:
Thu - AM Clouds/PM Sun. High: 75 Low: 63
Fri - Partly Cloudy. High: 75 Low: 63
Sat - Sunny. High: 76 Low: 64
Sun - Sunny. High: 81 Low: 64
Mon - Sunny. High: 82 Low: 63
Full forecast available at: http://us.rd.yahoo.com/dailynews
/rss/weather/Los_Angeles__CA/*http://weather.yahoo.com/forec
ast/USCA0638_f.html

Tier 1 of the challenge completed!
Enter username: secretagentman
Enter password for secretagentman: h(Eq1_z<[$Ry5]
Tier 2 of the challenge completed!
Since it's your first time encrypting a message -- Welcome!
 An encrypted welcome message has been left for you at C:\tm
p\secrets\welcome.  Decrypt it using your decryption program
 with recipient name 'alphabetic' and secret key 'efghijklm'
.
Follow the prompts below to encrypt your first message:

Enter recipient name:
```

## Tier 2 Complete!

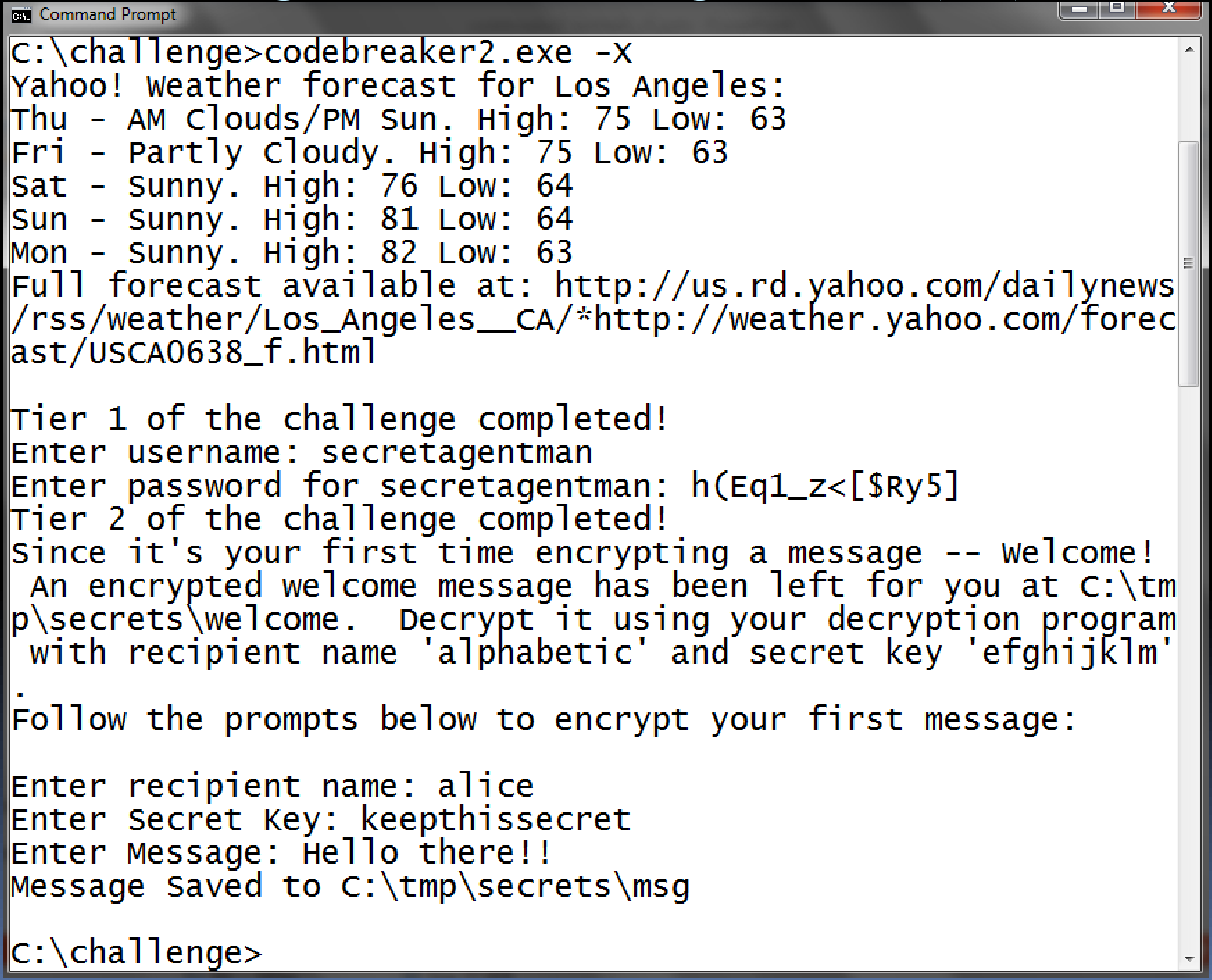
- Required either reverse engineering the password derivation function or just using a debugger to see the computed value
- … on to Tier 3!

## Running the program (4)



```
-h for help info
-l to list supported areas
-i to specify an area
-X to enter hidden mode

C:\challenge>codebreaker2.exe -X
Yahoo! Weather forecast for Los Angeles:
Thu - AM Clouds/PM Sun. High: 75 Low: 63
Fri - Partly Cloudy. High: 75 Low: 63
Sat - Sunny. High: 76 Low: 64
Sun - Sunny. High: 81 Low: 64
Mon - Sunny. High: 82 Low: 63
Full forecast available at: http://us.rd.yahoo.com/dailynews
/rss/weather/Los_Angeles__CA/*http://weather.yahoo.com/forec
ast/USCA0638_f.html

Tier 1 of the challenge completed!
Enter username: secretagentman
Enter password for secretagentman: h(Eq1_z<[$Ry5]
Tier 2 of the challenge completed!
Since it's your first time encrypting a message -- Welcome!
 An encrypted welcome message has been left for you at C:\tm
p\secrets\welcome.  Decrypt it using your decryption program
 with recipient name 'alphabetic' and secret key 'efghijklm'
.
Follow the prompts below to encrypt your first message:

Enter recipient name:
```

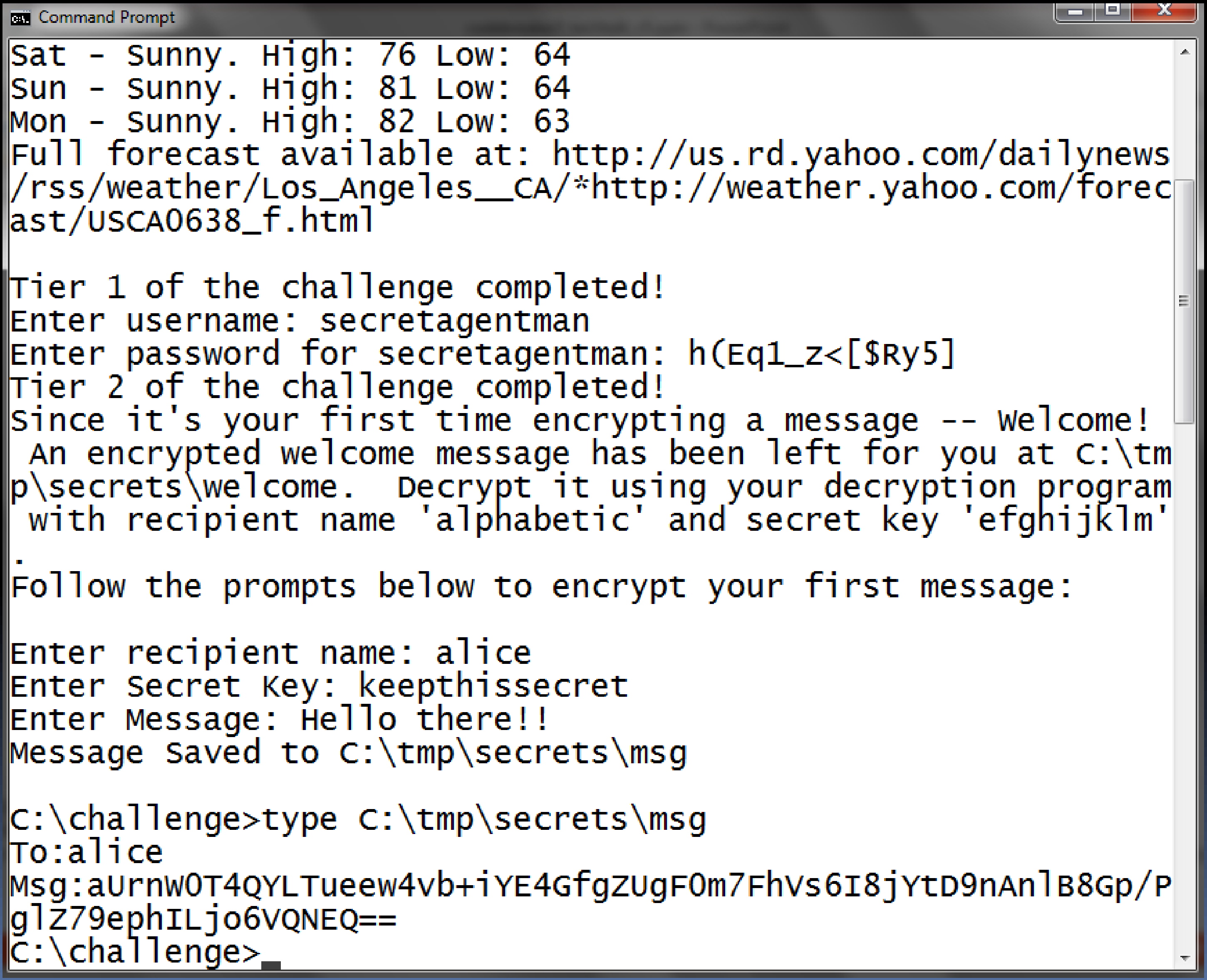
# Running the program (5)



Command Prompt

```
C:\challenge>codebreaker2.exe -X
Yahoo! Weather forecast for Los Angeles:
Thu - AM Clouds/PM Sun. High: 75 Low: 63
Fri - Partly Cloudy. High: 75 Low: 63
Sat - Sunny. High: 76 Low: 64
Sun - Sunny. High: 81 Low: 64
Mon - Sunny. High: 82 Low: 63
Full forecast available at: http://us.rd.yahoo.com/dailynews
/rss/weather/Los_Angeles__CA/*http://weather.yahoo.com/forec
ast/USCA0638_f.html

Tier 1 of the challenge completed!
Enter username: secretagentman
Enter password for secretagentman: h(Eq1_z<[$Ry5]
Tier 2 of the challenge completed!
Since it's your first time encrypting a message -- Welcome!
 An encrypted welcome message has been left for you at C:\tm
p\secrets\welcome.  Decrypt it using your decryption program
 with recipient name 'alphabetic' and secret key 'efghijklm'
.
Follow the prompts below to encrypt your first message:

Enter recipient name: alice
Enter Secret Key: keepthissecret
Enter Message: Hello there!!
Message Saved to C:\tmp\secrets\msg

C:\challenge>
```

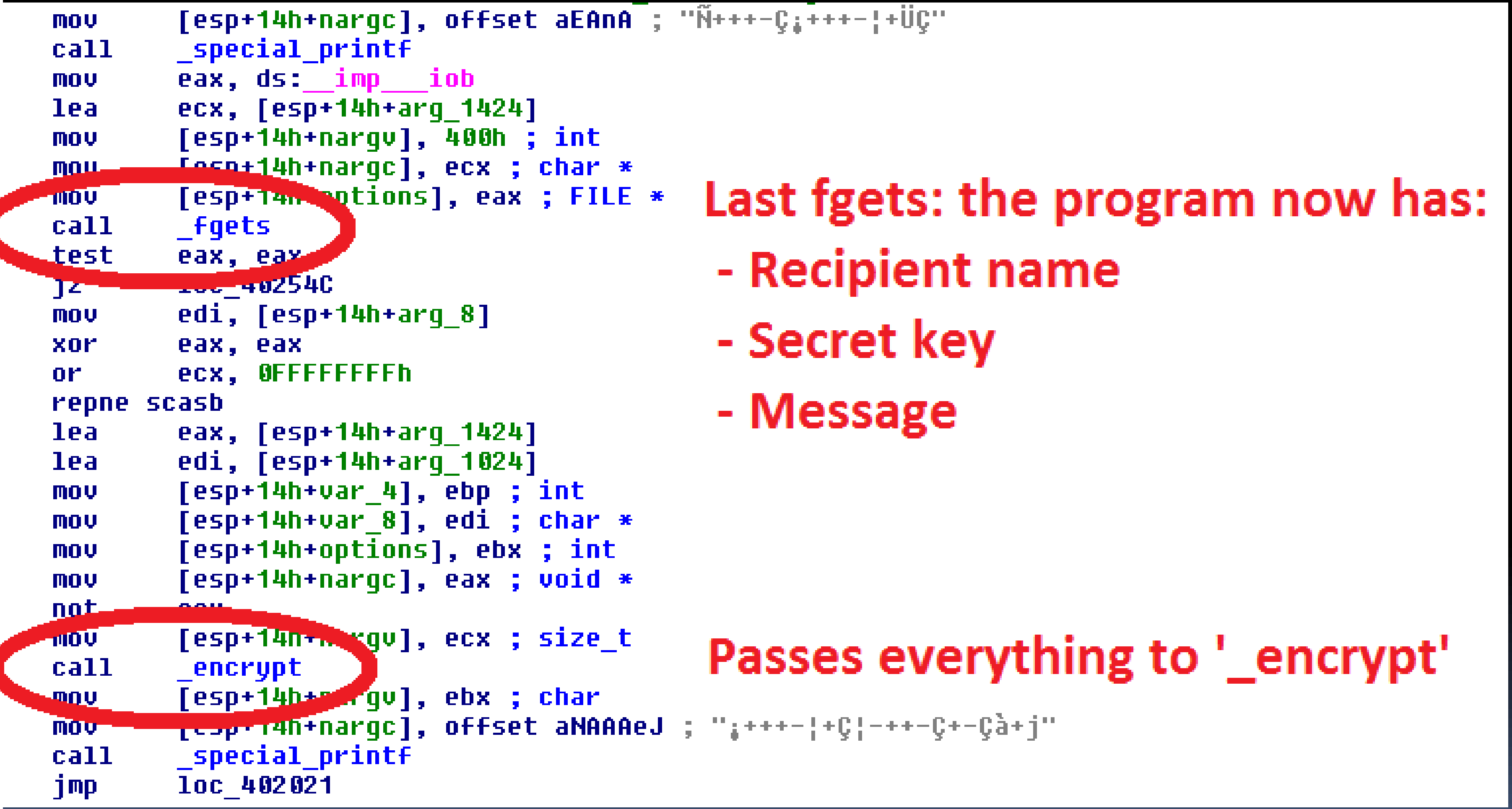
## Running the program (6)



Command Prompt

```
Sat - Sunny. High: 76 Low: 64
Sun - Sunny. High: 81 Low: 64
Mon - Sunny. High: 82 Low: 63
Full forecast available at: http://us.rd.yahoo.com/dailynews
/rss/weather/Los_Angeles__CA/*http://weather.yahoo.com/forec
ast/USCA0638_f.html

Tier 1 of the challenge completed!
Enter username: secretagentman
Enter password for secretagentman: h(Eq1_z<[$Ry5]
Tier 2 of the challenge completed!
Since it's your first time encrypting a message -- Welcome!
 An encrypted welcome message has been left for you at C:\tm
p\secrets\welcome.  Decrypt it using your decryption program
 with recipient name 'alphabetic' and secret key 'efghijklm'
.
Follow the prompts below to encrypt your first message:

Enter recipient name: alice
Enter Secret Key: keepthissecret
Enter Message: Hello there!!
Message Saved to C:\tmp\secrets\msg

C:\challenge>type C:\tmp\secrets\msg
To:alice
Msg:aUrnW0T4QYLTueew4vb+iYE4GfgZUgF0m7FhVs6I8jYtD9nAnlB8Gp/P
glZ79ephILjo6VQNEQ==
C:\challenge>_
```

# Explore Code Block (10)



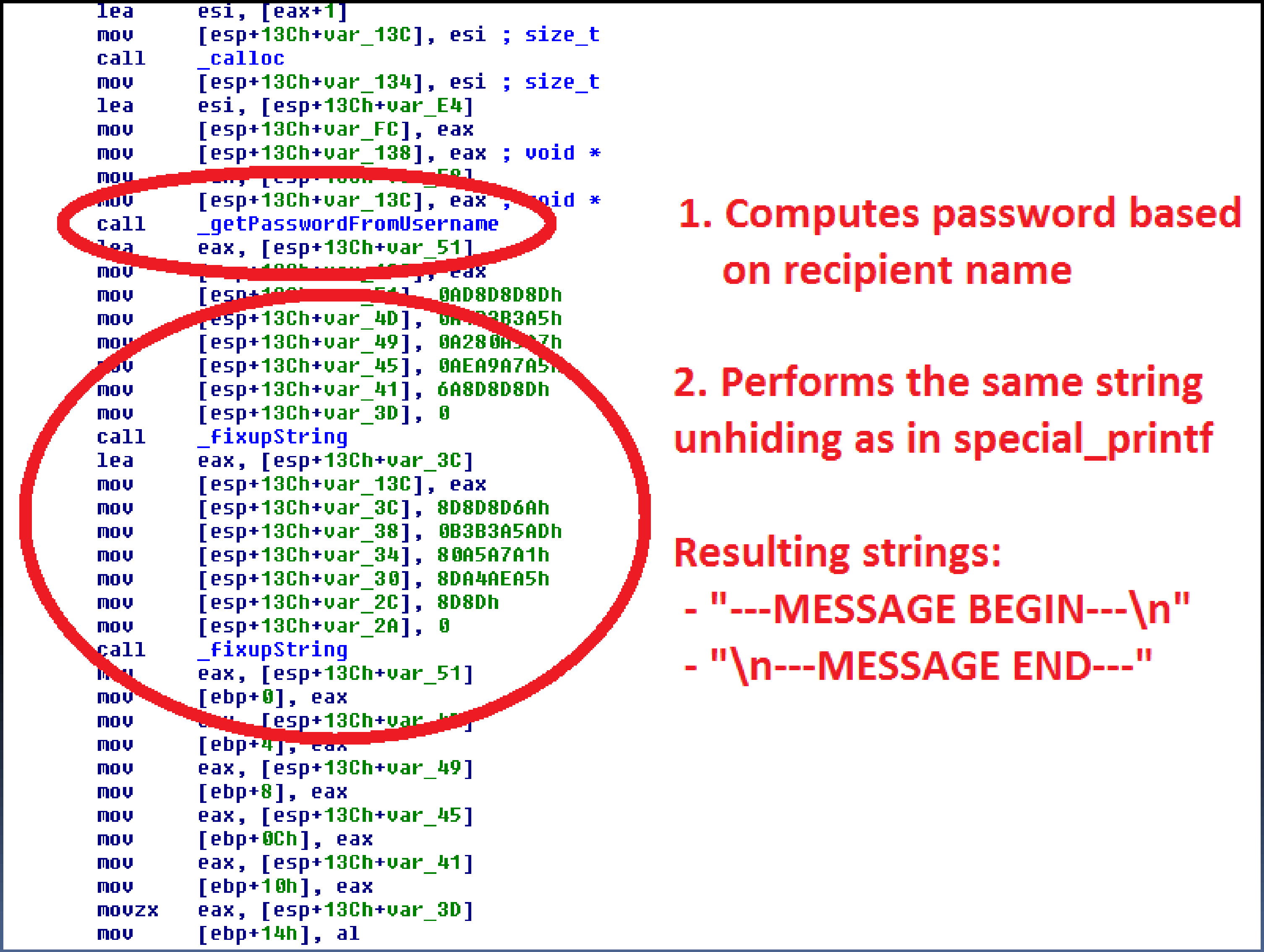
```
mov     [esp+14h+nargc], offset aEAnA ; "Ñ+++-Ç¡+++-¦+ÜÇ"
call    _special_printf
mov     eax, ds:__imp___iob
lea     ecx, [esp+14h+arg_1424]
mov     [esp+14h+nargv], 400h ; int
mov     [esp+14h+nargc], ecx ; char *
mov     [esp+14h+options], eax ; FILE *
call    _fgets
test    eax, eax
jz      loc_40254C
mov     edi, [esp+14h+arg_8]
xor     eax, eax
or      ecx, 0FFFFFFFFh
repne scasb
lea     eax, [esp+14h+arg_1424]
lea     edi, [esp+14h+arg_1024]
mov     [esp+14h+var_4], ebp ; int
mov     [esp+14h+var_8], edi ; char *
mov     [esp+14h+options], ebx ; int
mov     [esp+14h+nargc], eax ; void *
not     ecx
mov     [esp+14h+nargv], ecx ; size_t
call    _encrypt
mov     [esp+14h+nargv], ebx ; char
mov     [esp+14h+nargc], offset aNAAAeJ ; "¡+++-¦+Ç¦-++-Ç+-Çà+j"
call    _special_printf
jmp     loc_402021
```

Last fgets: the program now has:

- Recipient name
- Secret key
- Message

Passes everything to '_encrypt'

# Inside encrypt (1)



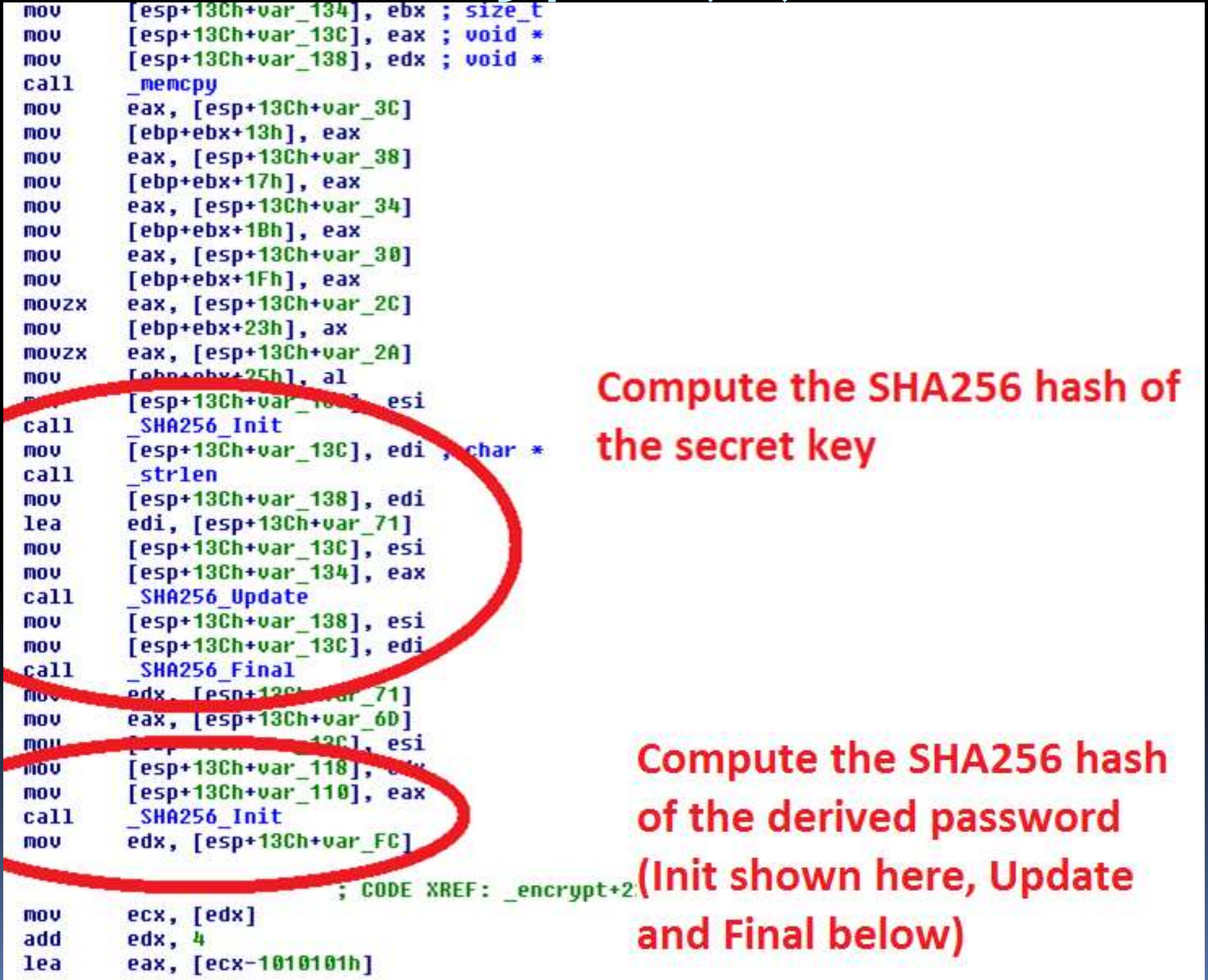
```
lea     esi, [eax+1]
mov     [esp+13Ch+var_13C], esi ; size_t
call    _calloc
mov     [esp+13Ch+var_134], esi ; size_t
lea     esi, [esp+13Ch+var_E4]
mov     [esp+13Ch+var_FC], eax
mov     [esp+13Ch+var_138], eax ; void *
mov     [illegible], [esp+13Ch+var_E8]
mov     [esp+13Ch+var_13C], eax ; void *
call    _getPasswordFromUsername
lea     eax, [esp+13Ch+var_51]
mov     [esp+13Ch+var_[illegible]], eax
mov     [esp+13Ch+var_51], 0AD8D8D8Dh
mov     [esp+13Ch+var_4D], 0A[illegible]3B3A5h
mov     [esp+13Ch+var_49], 0A280A[illegible]7h
mov     [esp+13Ch+var_45], 0AEA9A7A5h
mov     [esp+13Ch+var_41], 6A8D8D8Dh
mov     [esp+13Ch+var_3D], 0
call    _fixupString
lea     eax, [esp+13Ch+var_3C]
mov     [esp+13Ch+var_13C], eax
mov     [esp+13Ch+var_3C], 8D8D8D6Ah
mov     [esp+13Ch+var_38], 0B3B3A5ADh
mov     [esp+13Ch+var_34], 80A5A7A1h
mov     [esp+13Ch+var_30], 8DA4AEA5h
mov     [esp+13Ch+var_2C], 8D8Dh
mov     [esp+13Ch+var_2A], 0
call    _fixupString
mov     eax, [esp+13Ch+var_51]
mov     [ebp+0], eax
mov     eax, [esp+13Ch+var_4D]
mov     [ebp+4], eax
mov     eax, [esp+13Ch+var_49]
mov     [ebp+8], eax
mov     eax, [esp+13Ch+var_45]
mov     [ebp+0Ch], eax
mov     eax, [esp+13Ch+var_41]
mov     [ebp+10h], eax
movzx   eax, [esp+13Ch+var_3D]
mov     [ebp+14h], al
```

1. Computes password based on recipient name

2. Performs the same string unhiding as in special_printf

Resulting strings:
- "---MESSAGE BEGIN---\n"
- "\n---MESSAGE END---"

# Inside encrypt (2)

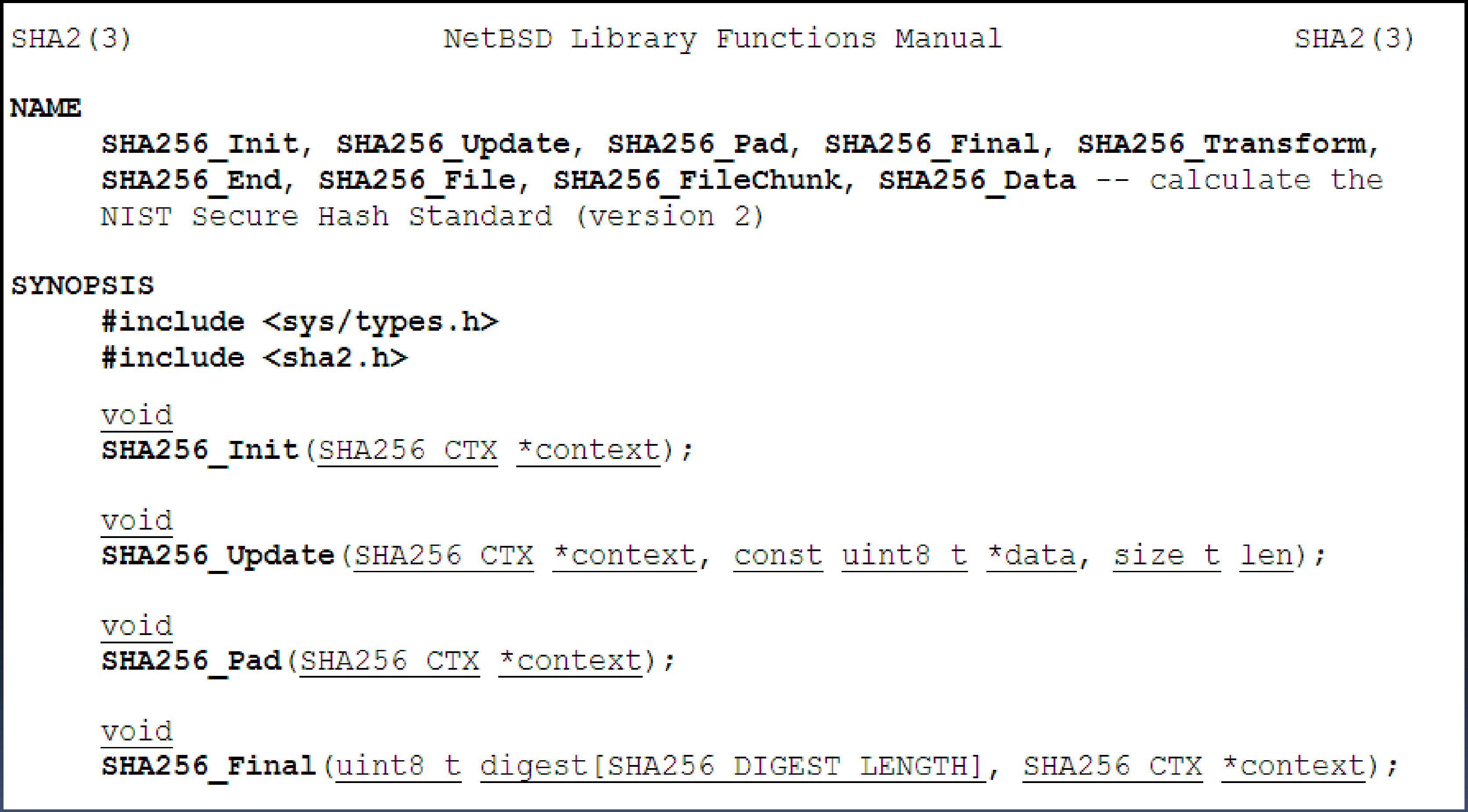


```
mov     [esp+13Ch+var_134], ebx ; size_t
mov     [esp+13Ch+var_13C], eax ; void *
mov     [esp+13Ch+var_138], edx ; void *
call    _memcpy
mov     eax, [esp+13Ch+var_3C]
mov     [ebp+ebx+13h], eax
mov     eax, [esp+13Ch+var_38]
mov     [ebp+ebx+17h], eax
mov     eax, [esp+13Ch+var_34]
mov     [ebp+ebx+1Bh], eax
mov     eax, [esp+13Ch+var_30]
mov     [ebp+ebx+1Fh], eax
movzx   eax, [esp+13Ch+var_2C]
mov     [ebp+ebx+23h], ax
movzx   eax, [esp+13Ch+var_2A]
mov     [ebp+ebx+25h], al
mov     [esp+13Ch+var_[illegible]], esi
call    _SHA256_Init
mov     [esp+13Ch+var_13C], edi ; char *
call    _strlen
mov     [esp+13Ch+var_138], edi
lea     edi, [esp+13Ch+var_71]
mov     [esp+13Ch+var_13C], esi
mov     [esp+13Ch+var_134], eax
call    _SHA256_Update
mov     [esp+13Ch+var_138], esi
mov     [esp+13Ch+var_13C], edi
call    _SHA256_Final
mov     edx, [esp+13Ch+var_71]
mov     eax, [esp+13Ch+var_6D]
mov     [esp+13Ch+var_13C], esi
mov     [esp+13Ch+var_118], edx
mov     [esp+13Ch+var_110], eax
call    _SHA256_Init
mov     edx, [esp+13Ch+var_FC]

                                  ; CODE XREF: _encrypt+2

mov     ecx, [edx]
add     edx, 4
lea     eax, [ecx-1010101h]
```

Compute the SHA256 hash of the secret key

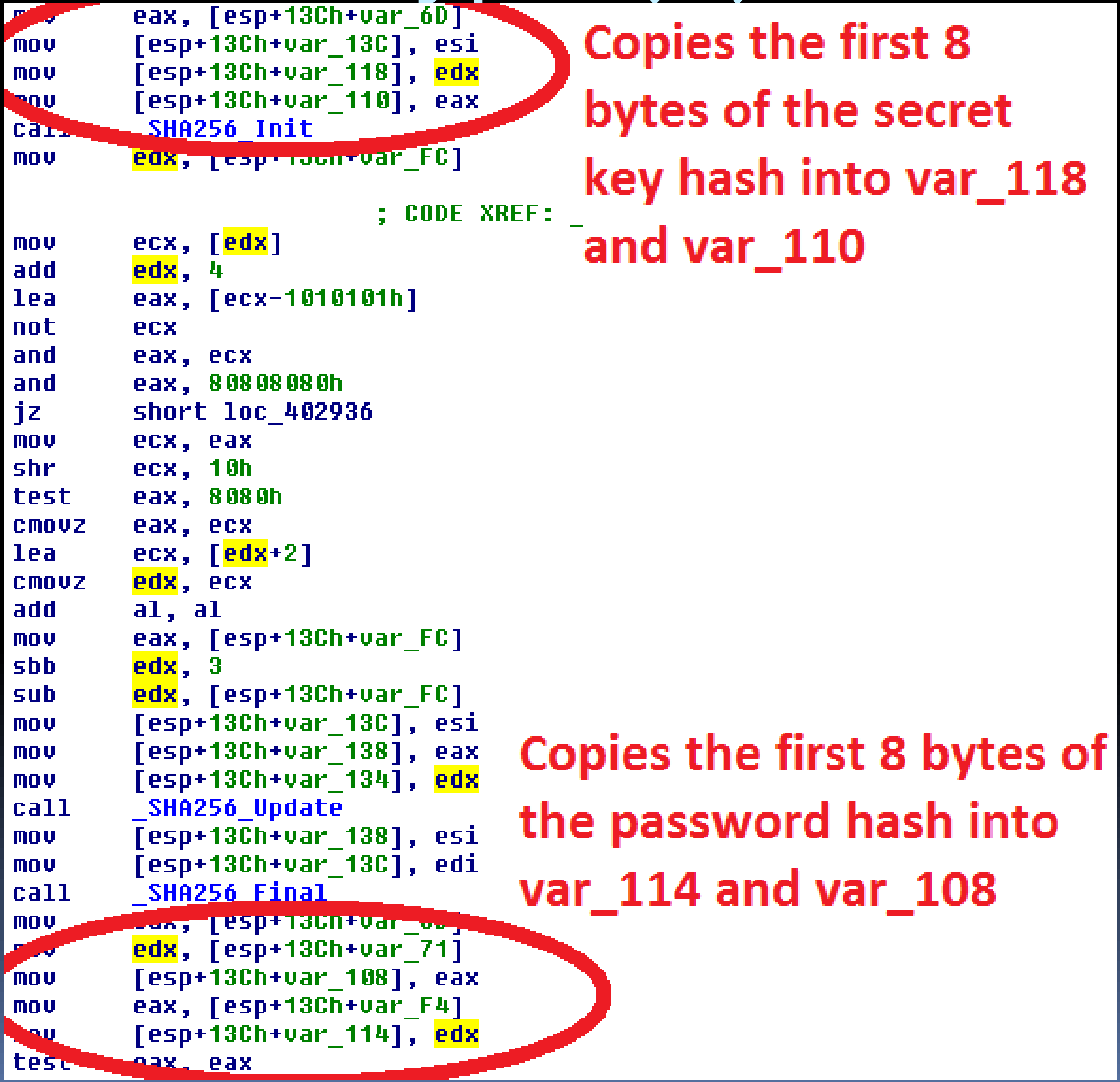
Compute the SHA256 hash of the derived password (Init shown here, Update and Final below)

# SHA256 Functions



```
SHA2(3)                NetBSD Library Functions Manual                SHA2(3)

NAME
     SHA256_Init, SHA256_Update, SHA256_Pad, SHA256_Final, SHA256_Transform,
     SHA256_End, SHA256_File, SHA256_FileChunk, SHA256_Data -- calculate the
     NIST Secure Hash Standard (version 2)

SYNOPSIS
     #include <sys/types.h>
     #include <sha2.h>

     void
     SHA256_Init(SHA256_CTX *context);

     void
     SHA256_Update(SHA256_CTX *context, const uint8_t *data, size_t len);

     void
     SHA256_Pad(SHA256_CTX *context);

     void
     SHA256_Final(uint8_t digest[SHA256_DIGEST_LENGTH], SHA256_CTX *context);
```

# Inside encrypt (3)

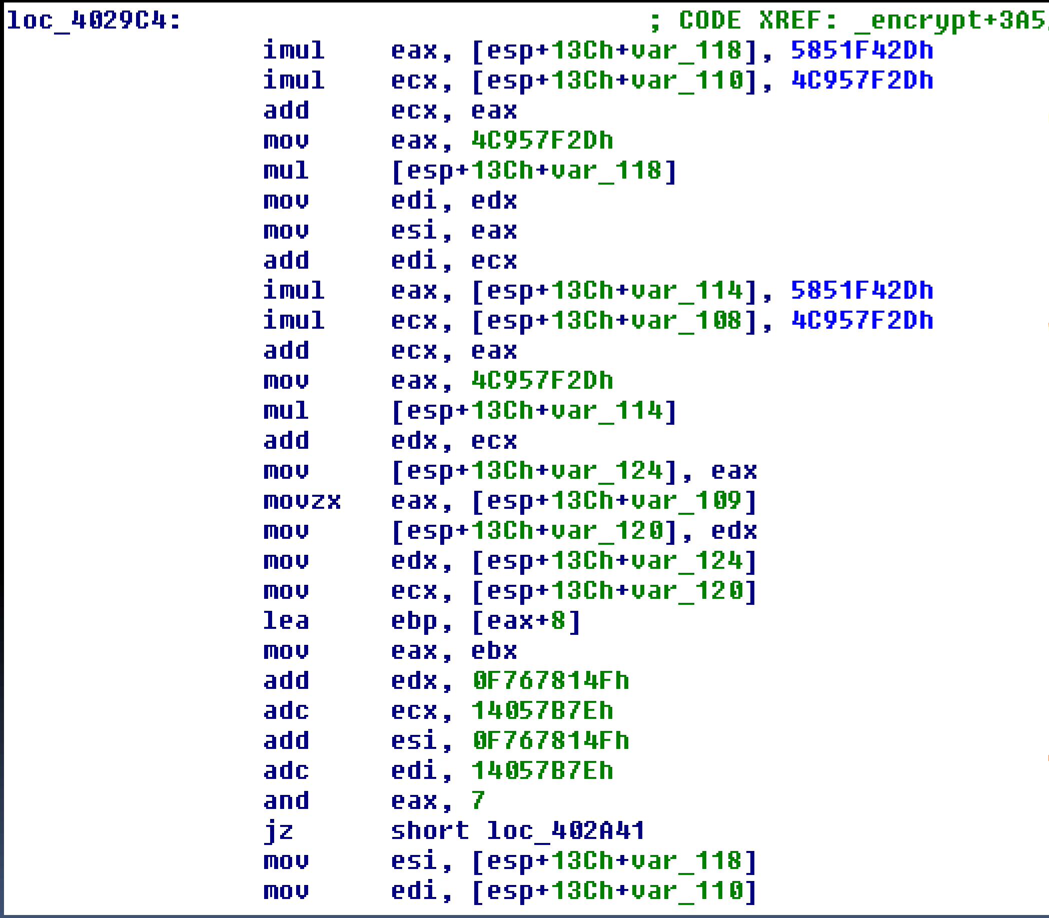


```
mov     eax, [esp+13Ch+var_6D]
mov     [esp+13Ch+var_13C], esi
mov     [esp+13Ch+var_118], edx
mov     [esp+13Ch+var_110], eax
call    _SHA256_Init
mov     edx, [esp+13Ch+var_FC]

                             ; CODE XREF: _
mov     ecx, [edx]
add     edx, 4
lea     eax, [ecx-1010101h]
not     ecx
and     eax, ecx
and     eax, 80808080h
jz      short loc_402936
mov     ecx, eax
shr     ecx, 10h
test    eax, 8080h
cmovz   eax, ecx
lea     ecx, [edx+2]
cmovz   edx, ecx
add     al, al
mov     eax, [esp+13Ch+var_FC]
sbb     edx, 3
sub     edx, [esp+13Ch+var_FC]
mov     [esp+13Ch+var_13C], esi
mov     [esp+13Ch+var_138], eax
mov     [esp+13Ch+var_134], edx
call    _SHA256_Update
mov     [esp+13Ch+var_138], esi
mov     [esp+13Ch+var_13C], edi
call    _SHA256_Final
mov     eax, [esp+13Ch+var_[illegible]]
mov     edx, [esp+13Ch+var_71]
mov     [esp+13Ch+var_108], eax
mov     eax, [esp+13Ch+var_F4]
mov     [esp+13Ch+var_114], edx
test    eax, eax
```

Copies the first 8 bytes of the secret key hash into var_118 and var_110

Copies the first 8 bytes of the password hash into var_114 and var_108

# Initial computation



```
loc_4029C4:                             ; CODE XREF: _encrypt+3A5↓j
                imul    eax, [esp+13Ch+var_118], 5851F42Dh
                imul    ecx, [esp+13Ch+var_110], 4C957F2Dh
                add     ecx, eax
                mov     eax, 4C957F2Dh
                mul     [esp+13Ch+var_118]
                mov     edi, edx
                mov     esi, eax
                add     edi, ecx
                imul    eax, [esp+13Ch+var_114], 5851F42Dh
                imul    ecx, [esp+13Ch+var_108], 4C957F2Dh
                add     ecx, eax
                mov     eax, 4C957F2Dh
                mul     [esp+13Ch+var_114]
                add     edx, ecx
                mov     [esp+13Ch+var_124], eax
                movzx   eax, [esp+13Ch+var_109]
                mov     [esp+13Ch+var_120], edx
                mov     edx, [esp+13Ch+var_124]
                mov     ecx, [esp+13Ch+var_120]
                lea     ebp, [eax+8]
                mov     eax, ebx
                add     edx, 0F767814Fh
                adc     ecx, 14057B7Eh
                add     esi, 0F767814Fh
                adc     edi, 14057B7Eh
                and     eax, 7
                jz      short loc_402A41
                mov     esi, [esp+13Ch+var_118]
                mov     edi, [esp+13Ch+var_110]
```

Computes

result = (A * X) + C

where

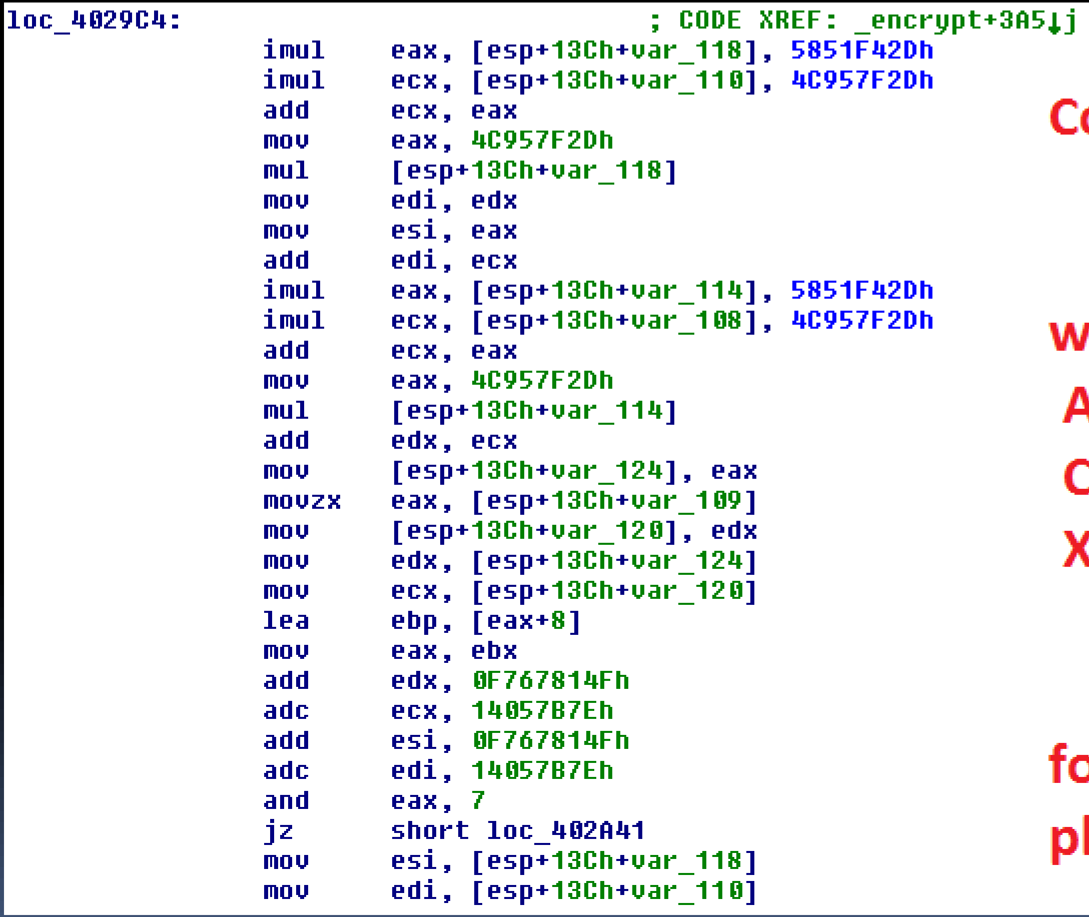
A = 6364136223846793005

C = 1442695040888963407

X = the 64-bits from the hash

for both values. Results are placed in EDX:ECX and EDI:ESI

# Subsequent computations



```
loc_4029C4:                             ; CODE XREF: _encrypt+3A5↓j
                imul    eax, [esp+13Ch+var_118], 5851F42Dh
                imul    ecx, [esp+13Ch+var_110], 4C957F2Dh
                add     ecx, eax
                mov     eax, 4C957F2Dh
                mul     [esp+13Ch+var_118]
                mov     edi, edx
                mov     esi, eax
                add     edi, ecx
                imul    eax, [esp+13Ch+var_114], 5851F42Dh
                imul    ecx, [esp+13Ch+var_108], 4C957F2Dh
                add     ecx, eax
                mov     eax, 4C957F2Dh
                mul     [esp+13Ch+var_114]
                add     edx, ecx
                mov     [esp+13Ch+var_124], eax
                movzx   eax, [esp+13Ch+var_109]
                mov     [esp+13Ch+var_120], edx
                mov     edx, [esp+13Ch+var_124]
                mov     ecx, [esp+13Ch+var_120]
                lea     ebp, [eax+8]
                mov     eax, ebx
                add     edx, 0F767814Fh
                adc     ecx, 14057B7Eh
                add     esi, 0F767814Fh
                adc     edi, 14057B7Eh
                and     eax, 7
                jz      short loc_402A41
                mov     esi, [esp+13Ch+var_118]
                mov     edi, [esp+13Ch+var_110]
```

Computes

result = (A * X) + C

where

A = 6364136223846793005

C = 1442695040888963407

X = 64-bit result computed previous

for both values. Results are placed in EDX:ECX and EDI:ESI

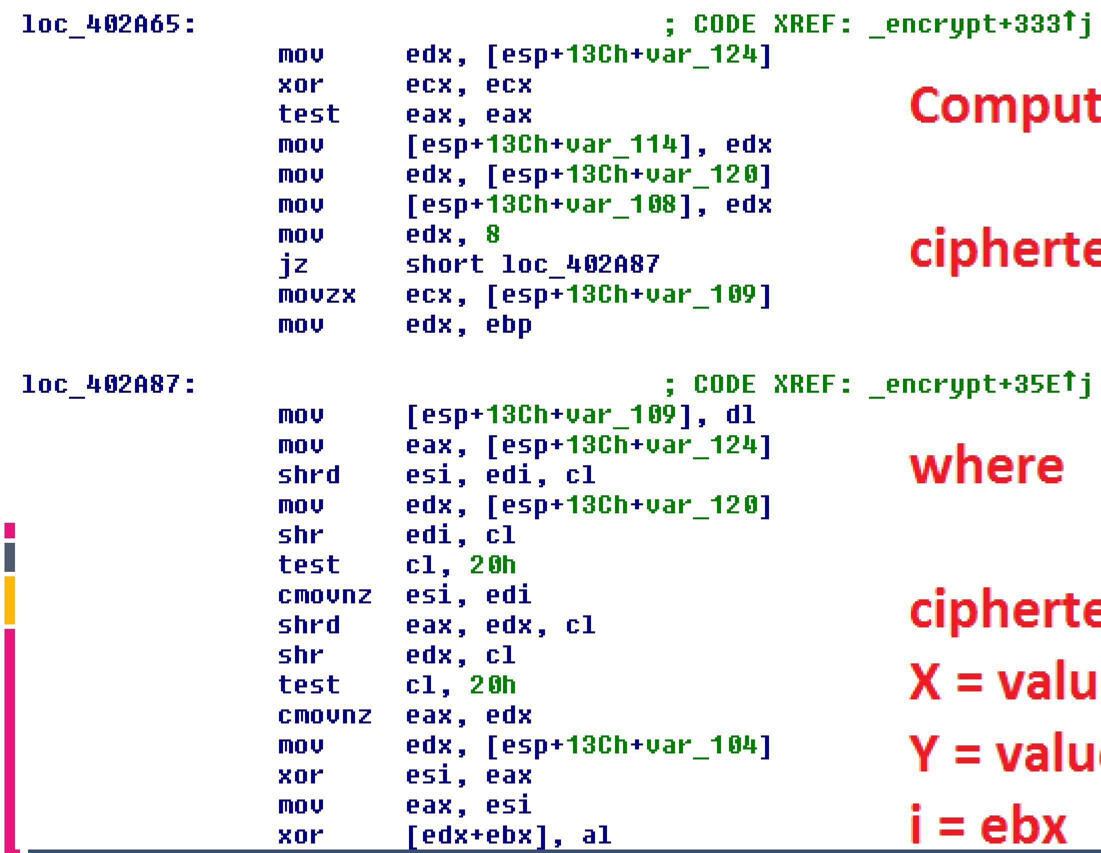
## So…

- byte buffer[] = SHA256(secret key)
- byte result[0-7] = (A * buffer[0-7]) + C
- byte result[8-15] = (A * result[0-7]) + C
- byte result[16-23] = (A * result[8-15]) + C

- … and same for the derived password

- From now on we will refer to:
  - 'result' from secret key as X
  - 'result' from derived password as Y

# Inside encrypt (4)



```
loc_402A65:                             ; CODE XREF: _encrypt+333↑j
                mov     edx, [esp+13Ch+var_124]
                xor     ecx, ecx
                test    eax, eax
                mov     [esp+13Ch+var_114], edx
                mov     edx, [esp+13Ch+var_120]
                mov     [esp+13Ch+var_108], edx
                mov     edx, 8
                jz      short loc_402A87
                movzx   ecx, [esp+13Ch+var_109]
                mov     edx, ebp

loc_402A87:                             ; CODE XREF: _encrypt+35E↑j
                mov     [esp+13Ch+var_109], dl
                mov     eax, [esp+13Ch+var_124]
                shrd    esi, edi, cl
                mov     edx, [esp+13Ch+var_120]
                shr     edi, cl
                test    cl, 20h
                cmovnz  esi, edi
                shrd    eax, edx, cl
                shr     edx, cl
                test    cl, 20h
                cmovnz  eax, edx
                mov     edx, [esp+13Ch+var_104]
                xor     esi, eax
                mov     eax, esi
                xor     [edx+ebx], al
```

Compute

ciphertext[i] = plaintext[i] ^ (X[i] ^ Y[i])

where

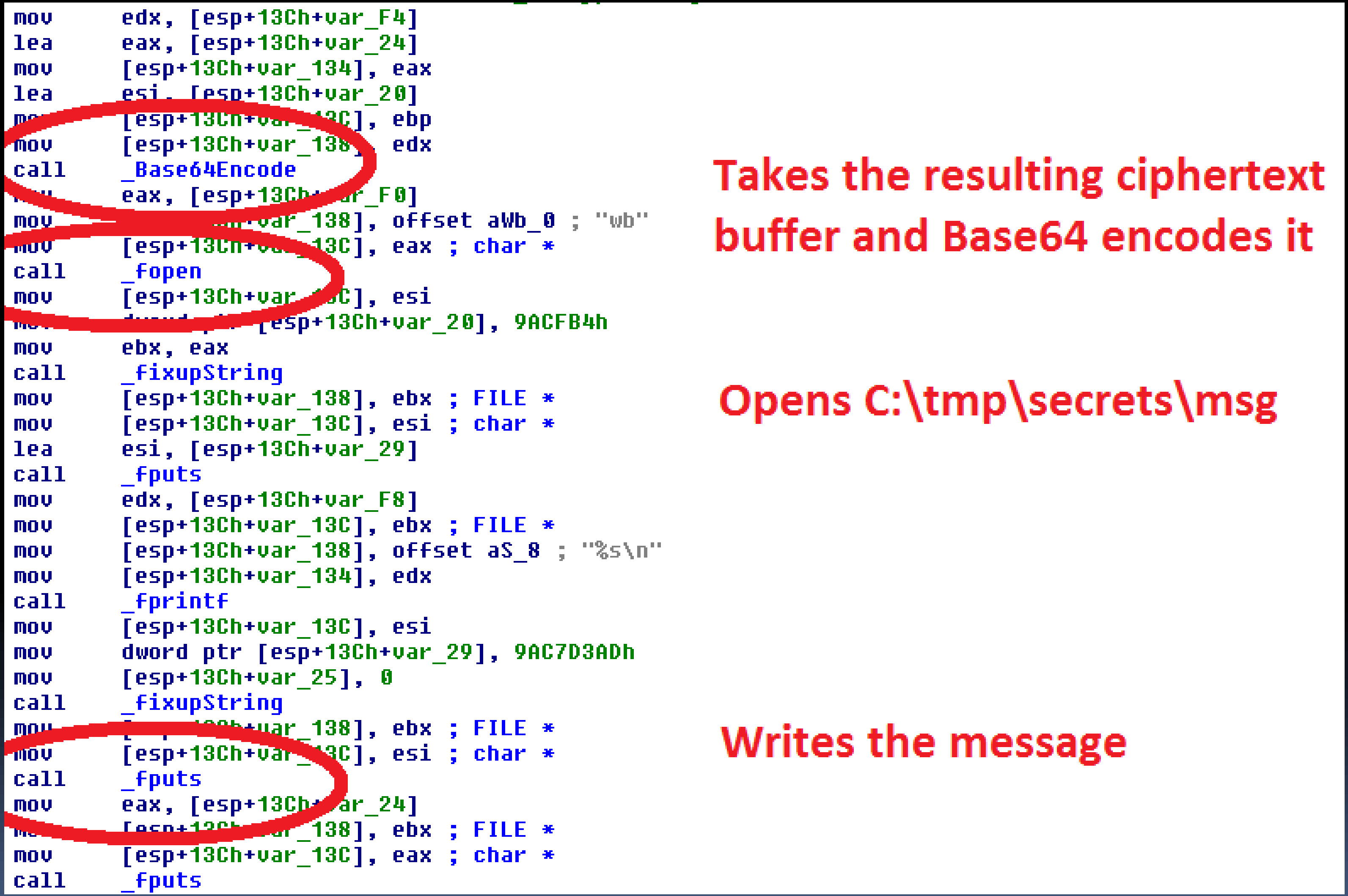
ciphertext = plaintext = var_104

X = value computed above (secret key)

Y = value computed above (password)

i = ebx

## Inside encrypt (5)



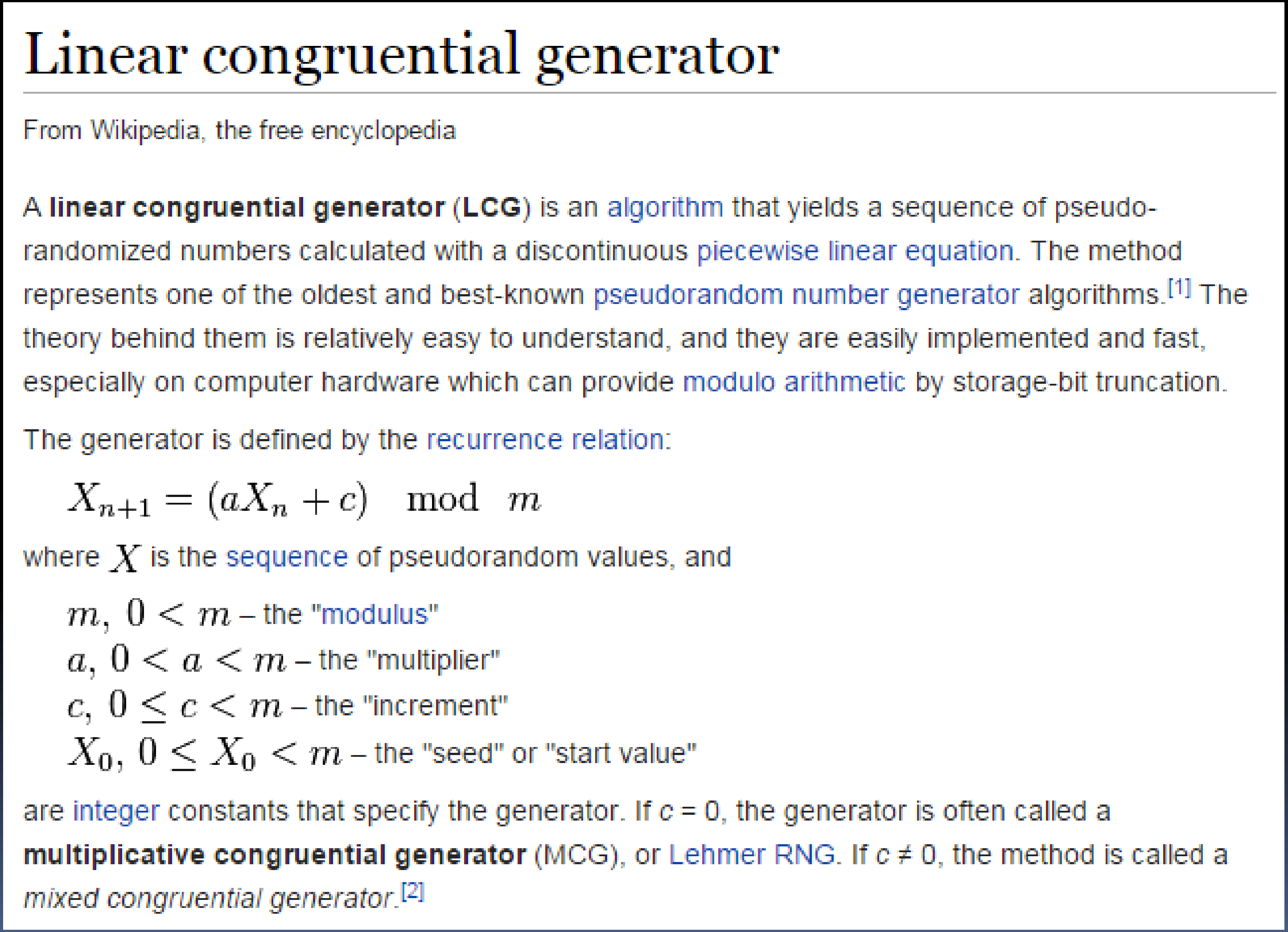
```
mov     edx, [esp+13Ch+var_F4]
lea     eax, [esp+13Ch+var_24]
mov     [esp+13Ch+var_134], eax
lea     esi, [esp+13Ch+var_20]
mov     [esp+13Ch+var_13C], ebp
mov     [esp+13Ch+var_138], edx
call    _Base64Encode
mov     eax, [esp+13Ch+var_F0]
mov     [esp+13Ch+var_138], offset aWb_0 ; "wb"
mov     [esp+13Ch+var_13C], eax ; char *
call    _fopen
mov     [esp+13Ch+var_13C], esi
mov     dword ptr [esp+13Ch+var_20], 9ACFB4h
mov     ebx, eax
call    _fixupString
mov     [esp+13Ch+var_138], ebx ; FILE *
mov     [esp+13Ch+var_13C], esi ; char *
lea     esi, [esp+13Ch+var_29]
call    _fputs
mov     edx, [esp+13Ch+var_F8]
mov     [esp+13Ch+var_13C], ebx ; FILE *
mov     [esp+13Ch+var_138], offset aS_8 ; "%s\n"
mov     [esp+13Ch+var_134], edx
call    _fprintf
mov     [esp+13Ch+var_13C], esi
mov     dword ptr [esp+13Ch+var_29], 9AC7D3ADh
mov     [esp+13Ch+var_25], 0
call    _fixupString
mov     [esp+13Ch+var_138], ebx ; FILE *
mov     [esp+13Ch+var_13C], esi ; char *
call    _fputs
mov     eax, [esp+13Ch+var_24]
mov     [esp+13Ch+var_138], ebx ; FILE *
mov     [esp+13Ch+var_13C], eax ; char *
call    _fputs
```

Takes the resulting ciphertext buffer and Base64 encodes it

Opens C:\tmp\secrets\msg

Writes the message

## What’s happening



# Linear congruential generator

From Wikipedia, the free encyclopedia

A **linear congruential generator** (**LCG**) is an algorithm that yields a sequence of pseudo-randomized numbers calculated with a discontinuous piecewise linear equation. The method represents one of the oldest and best-known pseudorandom number generator algorithms.[1] The theory behind them is relatively easy to understand, and they are easily implemented and fast, especially on computer hardware which can provide modulo arithmetic by storage-bit truncation.

The generator is defined by the recurrence relation:

$$X_{n+1} = (aX_n + c) \mod m$$

where $X$ is the sequence of pseudorandom values, and

$m,\ 0 < m$ – the "modulus"
$a,\ 0 < a < m$ – the "multiplier"
$c,\ 0 \le c < m$ – the "increment"
$X_0,\ 0 \le X_0 < m$ – the "seed" or "start value"

are integer constants that specify the generator. If $c = 0$, the generator is often called a **multiplicative congruential generator** (MCG), or Lehmer RNG. If $c \neq 0$, the method is called a *mixed congruential generator*.[2]

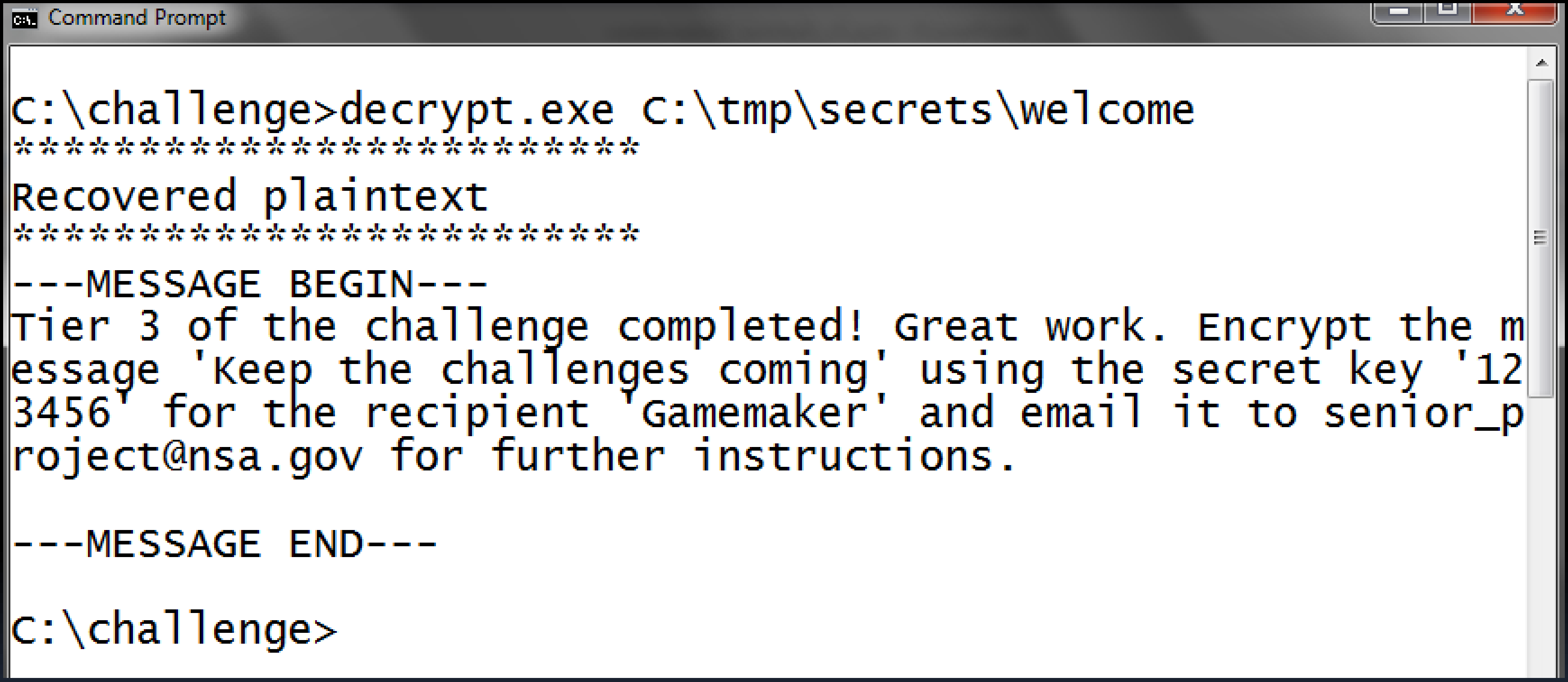
## Tier 3 Solution

- Write a program to:
  - Compute the password given a username*
  - Base64 decode the string**
  - Run the encryption algorithm in reverse to decrypt a given input
    - plaintext[i] = ciphertext[i] ^ (X[i] ^ Y[i])
    - where you compute X and Y from the given secret key and derived password

*Or just get it by running the Codebreaker binary in a debugger

** Or do it online, use existing programs, etc.

# Tier 3 Solution (2)



```
C:\challenge>decrypt.exe C:\tmp\secrets\welcome
*************************
Recovered plaintext
*************************
---MESSAGE BEGIN---
Tier 3 of the challenge completed! Great work. Encrypt the m
essage 'Keep the challenges coming' using the secret key '12
3456' for the recipient 'Gamemaker' and email it to senior_p
roject@nsa.gov for further instructions.

---MESSAGE END---

C:\challenge>
```

… on to Tier 4!

# Tier 4 Solution

- You’d get the following message via email:

To:Tier3_Codebreakers

Msg:z/W4uhaRU+8N7/qKSzuwXfNPZ8Tf867ajNJ33tU85wTtgXywSTefsB86
3g26B5rR2Q9/oqFztnrT6nTUq8JMuJbWTUD5YIsN7uTbw6F9/GzsgdBG567
A303kSOTEM+Fsp7QialTheU9/W/o2jiGZUeW6yYdhaMrDP6vDJlq+MNRMX
Zg8ereNKyBQDvGPR4iHUNBHoCP20Sb+/9WkeupRs2mkkoBAo8rdirZuoJ
NOwnugF9T/KwoR9EHVxNneIdDiGom8O2UilAUaR6pKHTu1xS6MfkVh5C
KArmVTY6MAC6Vi8CnZJvM/WZT6cg6dLesgFrtXX8uwhzcTYwLe+t2m5Mv
vDtiZyot9pLdBNAr6N3+znHCDInAIGlJe3shipbBQoqKxbb8VNY9DR4fJMG9
YIhnMyYn1g+mLGC41niWUqTbbBrnwSJgZ+u5AwLcpHXkA649O4IoHEyV+
bgWL/bKFVWL7KDAzEx4FdhwnYfe25SHirjFxVTrNiyR/FPPa/MgfixkrlVrZkY
GsZNlvDZjG8sxrH9tQokkOO7yaplHsBaYiwqCGVKum55iRyKgG1q2RuDAY
yzs1uvA2JnHnBZW1gEOpyy6RPiPuV7/z5DyQiMYhEzDA1Y9Dne92BagYoa
FTsCNMRX+W+L1XepcN49BEUDEMUKuUnLT6G+QuLw==

## Tier 4 Solution (2)

- Maybe there’s a problem with the encryption scheme…

## Tier 4 Solution (3)

- plaintext[i] = ciphertext[i] ^ (X[i] ^ Y[i])

- byte buffer[] = SHA256(secret key)
- byte X[0-7] = (A * buffer[0-7]) + C
- byte X[8-15] = (A * X[0-7]) + C
- …

- byte buffer[] = SHA256(derived password)
- byte Y[0-7] = (A * buffer[0-7]) + C
- …

- We have the ciphertext, and the constants

## Tier 4 Solution (4)

- plaintext[0-7] = ciphertext[0-7] ^ (X[0-7] ^ Y[0-7])

- byte buffer[] = SHA256(secret key)
- byte X[0-7] = (A * buffer[0-7]) + C
- byte X[8-15] = (A * X[0-7]) + C
- …

- byte buffer[] = SHA256(derived password)
- byte Y[0-7] = (A * buffer[0-7]) + C
- …

- All messages include the username too, from which we can derive the password, and then the SHA256 hash

## Tier 4 Solution (4)

- plaintext[0-7] = ciphertext[0-7] ^ (X[0-7] ^ Y[0-7])

- byte buffer[] = SHA256(secret key)
- byte X[0-7] = (A * buffer[0-7]) + C
- byte X[8-15] = (A * X[0-7]) + C
- …

- byte buffer[] = SHA256(derived password)
- byte Y[0-7] = (A * buffer[0-7]) + C
- …

- We know the plaintext at the beginning too, since the program always adds '---MESSAGE BEGIN---'

## Tier 4 Solution (5)

- plaintext[0-7] = ciphertext[0-7] ^ (X[0-7] ^ Y[0-7])

- byte buffer[] = SHA256(secret key)
- byte X[0-7] = (A * buffer[0-7]) + C
- byte X[8-15] = (A * X[0-7]) + C
- …

- byte buffer[] = SHA256(derived password)
- byte Y[0-7] = (A * buffer[0-7]) + C
- …

- Reversing the encryption equation, we get:

- X[0-7] = plaintext[0-7] ^ ciphertext[0-7] ^ Y[0-7]

## Tier 4 Solution (6)

- plaintext[0-7] = ciphertext[0-7] ^ (X[0-7] ^ Y[0-7])

- byte buffer[] = SHA256(secret key)
- byte X[0-7] = (A * buffer[0-7]) + C
- byte X[8-15] = (A * X[0-7]) + C
- …

- byte buffer[] = SHA256(derived password)
- byte Y[0-7] = (A * buffer[0-7]) + C
- …

- X[8-15] and Y[8-15] (and on) are computed from the SHA256 bytes, so we have the rest of those too. ☺

## Tier 4 Solution (7)

- Decrypted message:
- Congratulations!! You have solved the final tier of the Codebreaker Challenge! Please send us an email at senior_project@nsa.gov and let us know how you solved it. We hope you have enjoyed working on this problem. If you are interested in solving even more challenging and exciting problems on a daily basis that directly impact our national security posture and military forces around the world, consider applying for a career at NSA -- https://www.nsa.gov/careers.

## Questions

?

… if this work interests you, consider applying for an internship or full-time position at https://www.nsa.gov/careers

Use event code 483-1 to associate yourself with the Codebreaker Challenge

## Extra Slides

## 64-bit Data Types

Consider the following program:

```
int main(){
    char one              = 0x11;                  // sizeof(char) == 1
    char two              = 0x22;
    int three             = 0x33333333;            // sizeof(int) == 4
    int four              = 0x44444444;
    long long five   = 0x5555555555555555;         // sizeof(long long) == 8
    long long six    = 0x6666666666666666;
    printf("8b: %hu 32b: %u 64b: %llu\n", one + two, three + four, five + six);
    return 0;
}
```

# 64-bit Data Types – x86_64

Part 1: Move values onto the stack

```
mov   BYTE PTR [rbp-0x2],0x11
mov   BYTE PTR [rbp-0x1],0x22
mov   DWORD PTR [rbp-0xc],0x33333333
mov   DWORD PTR [rbp-0x8],0x44444444
mov   DWORD PTR [rbp-0x20],0x55555555
mov   DWORD PTR [rbp-0x1c],0x55555555
mov   DWORD PTR [rbp-0x18],0x66666666
mov   DWORD PTR [rbp-0x14],0x66666666
```

# 64-bit Data Types – x86_64

Part 2: Load into registers and compute

```
mov    rax,QWORD PTR [rbp-0x18]    // 0x6666666666666666 in rax
mov    rdx,QWORD PTR [rbp-0x20]    // 0x7777777777777777 in rdx
lea    rcx,[rdx+rax*1]             // rcx = rax + rdx*1
mov    eax,DWORD PTR [rbp-0x8]     // 0x44444444 in eax
mov    edx,DWORD PTR [rbp-0xc]     // 0x33333333 in edx
add    edx,eax                     // edx = edx + eax
movsx  esi,BYTE PTR [rbp-0x2]      // 0x11 in esi
movsx  eax,BYTE PTR [rbp-0x1]      // 0x22 in eax
add    esi,eax                     // esi = esi + eax
```

# 64-bit Data Types – x86

No 64-bit registers ☹

```
long long five  = 0x5555555555555555; // sizeof(long long) == 8
long long six   = 0x6666666666666666;
```

Let’s make it work with 32-bit ones!

## 64-bit Data Types – x86

Part 1: Move values onto the stack (same as x86_64)

```
mov    BYTE PTR [ebp-1],0x11
mov    BYTE PTR [ebp-2],0x22
mov    DWORD PTR [ebp-8],0x33333333
mov    DWORD PTR [ebp-12],0x44444444
mov    DWORD PTR [ebp-24],0x55555555
mov    DWORD PTR [ebp-20],0x55555555
mov    DWORD PTR [ebp-32],0x66666666
mov    DWORD PTR [ebp-28],0x66666666
```

## 64-bit Data Types – x86

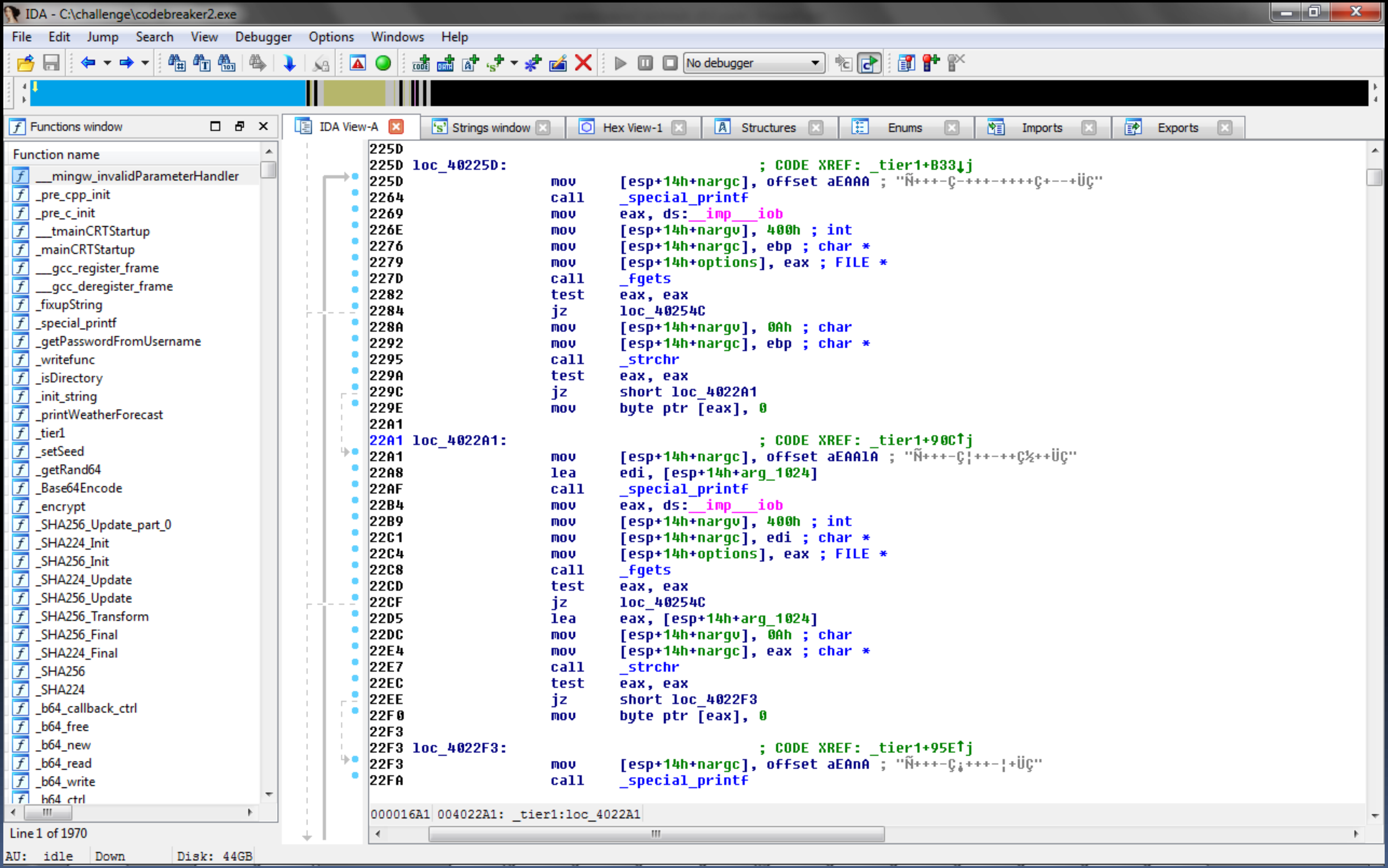
Part 2: Load into registers and compute

```
mov     eax,DWORD PTR [ebp-32]      // 0x66666666 in eax
mov     edx,DWORD PTR [ebp-28]      // 0x66666666 in edx
add     eax,DWORD PTR [ebp-24]      // eax = eax + 0x55555555
adc     edx,DWORD PTR [ebp-20]      // edx = edx + 0x55555555  + CF
...
mov     eax,DWORD PTR [ebp-12]      // 0x444444 in eax
add     eax,DWORD PTR [ebp-8]       // eax = eax + 0x33333333
...
movsx   edx,BYTE PTR [ebp-1]        // 0x11 in edx
movsx  eax,BYTE PTR [ebp-2]         // 0x22 in eax
lea    eax,[edx+eax]                // eax = edx + eax*
```

## Strings?

- The strings that are used here don’t appear in the list of strings
- … or do they?  Scroll down in the code to try and spot them.

# Explore Code Block (10)



```
__mingw_invalidParameterHandler
_pre_cpp_init
_pre_c_init
__tmainCRTStartup
_mainCRTStartup
__gcc_register_frame
__gcc_deregister_frame
_fixupString
_special_printf
_getPasswordFromUsername
_writefunc
_isDirectory
_init_string
_printWeatherForecast
_tier1
_setSeed
_getRand64
_Base64Encode
_encrypt
_SHA256_Update_part_0
_SHA224_Init
_SHA256_Init
_SHA224_Update
_SHA256_Update
_SHA256_Transform
_SHA256_Final
_SHA224_Final
_SHA256
_SHA224
_b64_callback_ctrl
_b64_free
_b64_new
_b64_read
_b64_write
_b64_ctrl
```

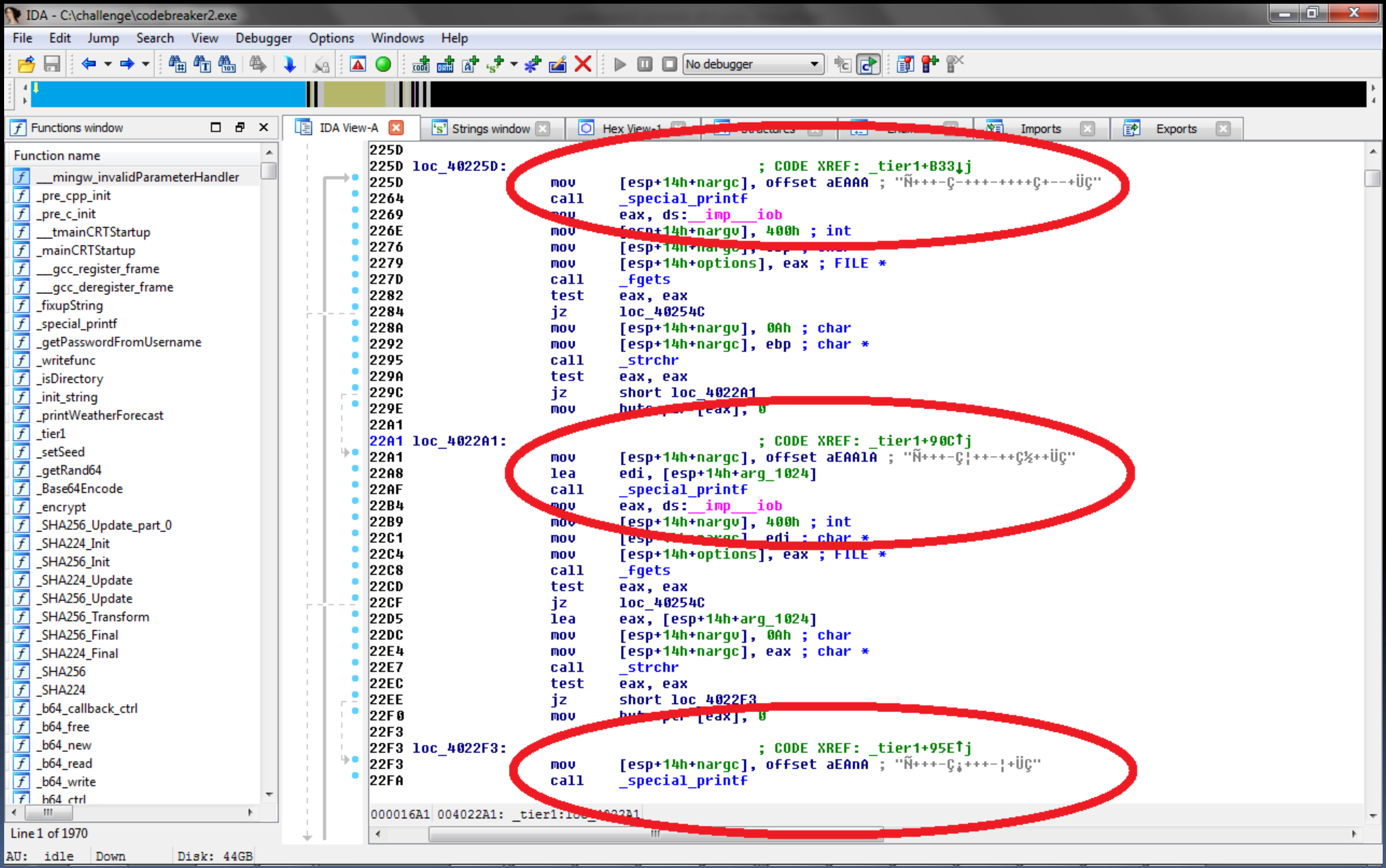
Line 1 of 1970

AU: idle Down Disk: 44GB

```
225D
225D loc_40225D:                            ; CODE XREF: _tier1+B33↓j
225D                 mov     [esp+14h+nargc], offset aEAAA ; "Ñ+++-Ç-+++-++++Ç+--+ÜÇ"
2264                 call    _special_printf
2269                 mov     eax, ds:__imp___iob
226E                 mov     [esp+14h+nargv], 400h ; int
2276                 mov     [esp+14h+nargc], ebp ; char *
2279                 mov     [esp+14h+options], eax ; FILE *
227D                 call    _fgets
2282                 test    eax, eax
2284                 jz      loc_40254C
228A                 mov     [esp+14h+nargv], 0Ah ; char
2292                 mov     [esp+14h+nargc], ebp ; char *
2295                 call    _strchr
229A                 test    eax, eax
229C                 jz      short loc_4022A1
229E                 mov     byte ptr [eax], 0
22A1
22A1 loc_4022A1:                            ; CODE XREF: _tier1+90C↑j
22A1                 mov     [esp+14h+nargc], offset aEAA1A ; "Ñ+++-Ç¦++-++Ç½++ÜÇ"
22A8                 lea     edi, [esp+14h+arg_1024]
22AF                 call    _special_printf
22B4                 mov     eax, ds:__imp___iob
22B9                 mov     [esp+14h+nargv], 400h ; int
22C1                 mov     [esp+14h+nargc], edi ; char *
22C4                 mov     [esp+14h+options], eax ; FILE *
22C8                 call    _fgets
22CD                 test    eax, eax
22CF                 jz      loc_40254C
22D5                 lea     eax, [esp+14h+arg_1024]
22DC                 mov     [esp+14h+nargv], 0Ah ; char
22E4                 mov     [esp+14h+nargc], eax ; char *
22E7                 call    _strchr
22EC                 test    eax, eax
22EE                 jz      short loc_4022F3
22F0                 mov     byte ptr [eax], 0
22F3
22F3 loc_4022F3:                            ; CODE XREF: _tier1+95E↑j
22F3                 mov     [esp+14h+nargc], offset aEAnA ; "Ñ+++-Ç¡+++-¦+ÜÇ"
22FA                 call    _special_printf
```

000016A1 004022A1: _tier1:loc_4022A1

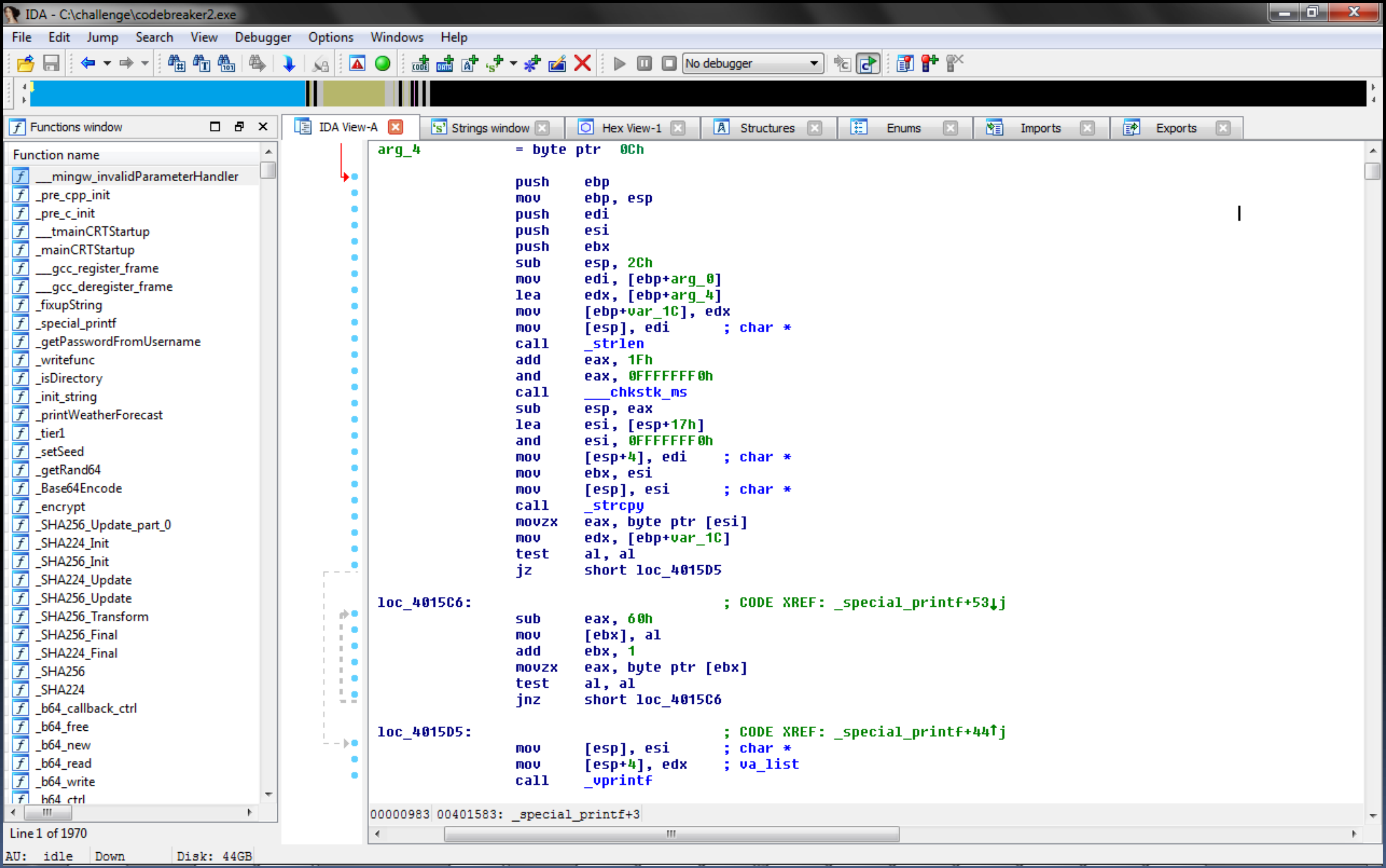
# Explore Code Block (11)


IDA - C:\challenge\codebreaker2.exe

Function name
```
__mingw_invalidParameterHandler
_pre_cpp_init
_pre_c_init
__tmainCRTStartup
_mainCRTStartup
__gcc_register_frame
__gcc_deregister_frame
_fixupString
_special_printf
_getPasswordFromUsername
_writefunc
_isDirectory
_init_string
_printWeatherForecast
_tier1
_setSeed
_getRand64
_Base64Encode
_encrypt
_SHA256_Update_part_0
_SHA224_Init
_SHA256_Init
_SHA224_Update
_SHA256_Update
_SHA256_Transform
_SHA256_Final
_SHA224_Final
_SHA256
_SHA224
_b64_callback_ctrl
_b64_free
_b64_new
_b64_read
_b64_write
_b64_ctrl
```
Line 1 of 1970

```
225D
225D loc_40225D:                             ; CODE XREF: _tier1+B33↓j
225D          mov     [esp+14h+nargc], offset aEAAA ; "Ñ+++-Ç-+++-++++Ç+--+ÜÇ"
2264          call    _special_printf
2269          mov     eax, ds:__imp___iob
226E          mov     [esp+14h+nargv], 400h ; int
2276          mov     [esp+14h+nargc], ebp ; char *
2279          mov     [esp+14h+options], eax ; FILE *
227D          call    _fgets
2282          test    eax, eax
2284          jz      loc_40254C
228A          mov     [esp+14h+nargv], 0Ah ; char
2292          mov     [esp+14h+nargc], ebp ; char *
2295          call    _strchr
229A          test    eax, eax
229C          jz      short loc_4022A1
229E          mov     byte ptr [eax], 0
22A1
22A1 loc_4022A1:                             ; CODE XREF: _tier1+90C↑j
22A1          mov     [esp+14h+nargc], offset aEAA1A ; "Ñ+++-Ç¦++-++Ç½++ÜÇ"
22A8          lea     edi, [esp+14h+arg_1024]
22AF          call    _special_printf
22B4          mov     eax, ds:__imp___iob
22B9          mov     [esp+14h+nargv], 400h ; int
22C1          mov     [esp+14h+nargc], edi ; char *
22C4          mov     [esp+14h+options], eax ; FILE *
22C8          call    _fgets
22CD          test    eax, eax
22CF          jz      loc_40254C
22D5          lea     eax, [esp+14h+arg_1024]
22DC          mov     [esp+14h+nargv], 0Ah ; char
22E4          mov     [esp+14h+nargc], eax ; char *
22E7          call    _strchr
22EC          test    eax, eax
22EE          jz      short loc_4022F3
22F0          mov     byte ptr [eax], 0
22F3
22F3 loc_4022F3:                             ; CODE XREF: _tier1+95E↑j
22F3          mov     [esp+14h+nargc], offset aEAnA ; "Ñ+++-Ç¡+++-¦+ÜÇ"
22FA          call    _special_printf
```
```
000016A1 004022A1: _tier1:loc_4022A1
```
```
AU:  idle   Down      Disk: 44GB
```

# Inside special_printf



```
__mingw_invalidParameterHandler
_pre_cpp_init
_pre_c_init
__tmainCRTStartup
_mainCRTStartup
__gcc_register_frame
__gcc_deregister_frame
_fixupString
_special_printf
_getPasswordFromUsername
_writefunc
_isDirectory
_init_string
_printWeatherForecast
_tier1
_setSeed
_getRand64
_Base64Encode
_encrypt
_SHA256_Update_part_0
_SHA224_Init
_SHA256_Init
_SHA224_Update
_SHA256_Update
_SHA256_Transform
_SHA256_Final
_SHA224_Final
_SHA256
_SHA224
_b64_callback_ctrl
_b64_free
_b64_new
_b64_read
_b64_write
_b64_ctrl
```

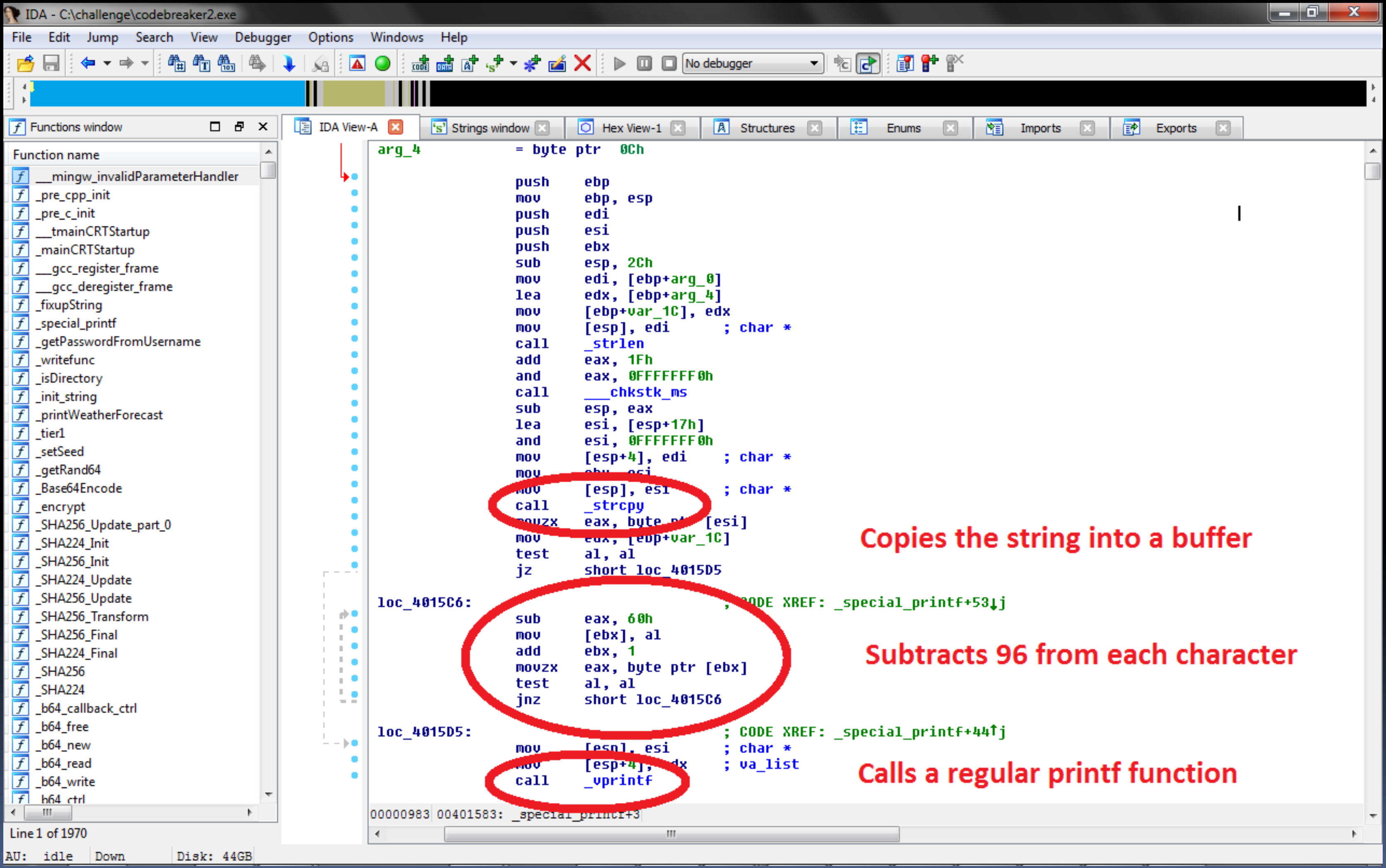
Line 1 of 1970

```
arg_4           = byte ptr  0Ch

                push    ebp
                mov     ebp, esp
                push    edi
                push    esi
                push    ebx
                sub     esp, 2Ch
                mov     edi, [ebp+arg_0]
                lea     edx, [ebp+arg_4]
                mov     [ebp+var_1C], edx
                mov     [esp], edi      ; char *
                call    _strlen
                add     eax, 1Fh
                and     eax, 0FFFFFFF0h
                call    ___chkstk_ms
                sub     esp, eax
                lea     esi, [esp+17h]
                and     esi, 0FFFFFFF0h
                mov     [esp+4], edi    ; char *
                mov     ebx, esi
                mov     [esp], esi      ; char *
                call    _strcpy
                movzx   eax, byte ptr [esi]
                mov     edx, [ebp+var_1C]
                test    al, al
                jz      short loc_4015D5

loc_4015C6:                             ; CODE XREF: _special_printf+53↓j
                sub     eax, 60h
                mov     [ebx], al
                add     ebx, 1
                movzx   eax, byte ptr [ebx]
                test    al, al
                jnz     short loc_4015C6

loc_4015D5:                             ; CODE XREF: _special_printf+44↑j
                mov     [esp], esi      ; char *
                mov     [esp+4], edx    ; va_list
                call    _vprintf
```

```
00000983 00401583: _special_printf+3
```

```
AU:  idle   Down     Disk: 44GB
```

# Inside special_printf (2)



```
__mingw_invalidParameterHandler
_pre_cpp_init
_pre_c_init
__tmainCRTStartup
_mainCRTStartup
__gcc_register_frame
__gcc_deregister_frame
_fixupString
_special_printf
_getPasswordFromUsername
_writefunc
_isDirectory
_init_string
_printWeatherForecast
_tier1
_setSeed
_getRand64
_Base64Encode
_encrypt
_SHA256_Update_part_0
_SHA224_Init
_SHA256_Init
_SHA224_Update
_SHA256_Update
_SHA256_Transform
_SHA256_Final
_SHA224_Final
_SHA256
_SHA224
_b64_callback_ctrl
_b64_free
_b64_new
_b64_read
_b64_write
_b64_ctrl
```

Line 1 of 1970

AU: idle Down Disk: 44GB

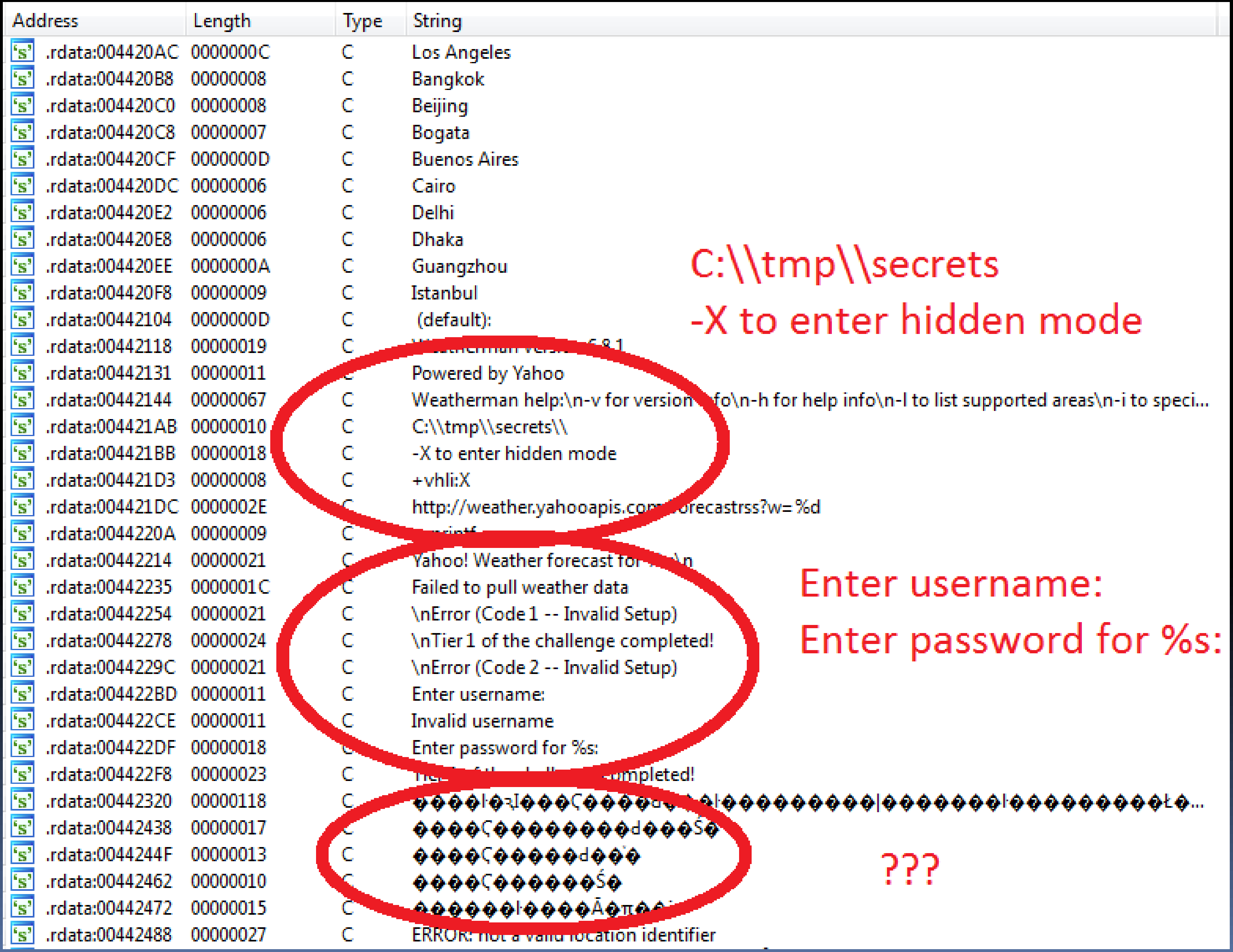
```
arg_4           = byte ptr  0Ch

                push    ebp
                mov     ebp, esp
                push    edi
                push    esi
                push    ebx
                sub     esp, 2Ch
                mov     edi, [ebp+arg_0]
                lea     edx, [ebp+arg_4]
                mov     [ebp+var_1C], edx
                mov     [esp], edi      ; char *
                call    _strlen
                add     eax, 1Fh
                and     eax, 0FFFFFFF0h
                call    ___chkstk_ms
                sub     esp, eax
                lea     esi, [esp+17h]
                and     esi, 0FFFFFFF0h
                mov     [esp+4], edi    ; char *
                mov     ebx, esi
                mov     [esp], esi      ; char *
                call    _strcpy
                movzx   eax, byte ptr [esi]
                mov     edx, [ebp+var_1C]
                test    al, al
                jz      short loc_4015D5

loc_4015C6:                             ; CODE XREF: _special_printf+53↓j
                sub     eax, 60h
                mov     [ebx], al
                add     ebx, 1
                movzx   eax, byte ptr [ebx]
                test    al, al
                jnz     short loc_4015C6

loc_4015D5:                             ; CODE XREF: _special_printf+44↑j
                mov     [esp], esi      ; char *
                mov     [esp+4], edx    ; va_list
                call    _vprintf
00000983 00401583: _special_printf+3
```

Copies the string into a buffer

Subtracts 96 from each character

Calls a regular printf function

# Mystery solved!



| Address | Length | Type | String |
|---|---|---|---|
| .rdata:004420AC | 0000000C | C | Los Angeles |
| .rdata:004420B8 | 00000008 | C | Bangkok |
| .rdata:004420C0 | 00000008 | C | Beijing |
| .rdata:004420C8 | 00000007 | C | Bogata |
| .rdata:004420CF | 0000000D | C | Buenos Aires |
| .rdata:004420DC | 00000006 | C | Cairo |
| .rdata:004420E2 | 00000006 | C | Delhi |
| .rdata:004420E8 | 00000006 | C | Dhaka |
| .rdata:004420EE | 0000000A | C | Guangzhou |
| .rdata:004420F8 | 00000009 | C | Istanbul |
| .rdata:00442104 | 0000000D | C | (default): |
| .rdata:00442118 | 00000019 | C | Weatherman version 6.8.1 |
| .rdata:00442131 | 00000011 | C | Powered by Yahoo |
| .rdata:00442144 | 00000067 | C | Weatherman help:\n-v for version info\n-h for help info\n-l to list supported areas\n-i to speci... |
| .rdata:004421AB | 00000010 | C | C:\\tmp\\secrets\\ |
| .rdata:004421BB | 00000018 | C | -X to enter hidden mode |
| .rdata:004421D3 | 00000008 | C | +vhli:X |
| .rdata:004421DC | 0000002E | C | http://weather.yahooapis.com/forecastrss?w=%d |
| .rdata:0044220A | 00000009 | C | [illegible]printf |
| .rdata:00442214 | 00000021 | C | Yahoo! Weather forecast for [illegible]\n |
| .rdata:00442235 | 0000001C | C | Failed to pull weather data |
| .rdata:00442254 | 00000021 | C | \nError (Code 1 -- Invalid Setup) |
| .rdata:00442278 | 00000024 | C | \nTier 1 of the challenge completed! |
| .rdata:0044229C | 00000021 | C | \nError (Code 2 -- Invalid Setup) |
| .rdata:004422BD | 00000011 | C | Enter username: |
| .rdata:004422CE | 00000011 | C | Invalid username |
| .rdata:004422DF | 00000018 | C | Enter password for %s: |
| .rdata:004422F8 | 00000023 | C | Tier 2 of the challenge completed! |
| .rdata:00442320 | 00000118 | C | ����l�ʇl���Ç�����d��l�����������\|��������l�������������Ł�... |
| .rdata:00442438 | 00000017 | C | ����Ç���������d����Ś� |
| .rdata:0044244F | 00000013 | C | ����Ç�������d��̀� |
| .rdata:00442462 | 00000010 | C | ����Ç��������Ś� |
| .rdata:00442472 | 00000015 | C | ������l�����Ā�π��[illegible] |
| .rdata:00442488 | 00000027 | C | ERROR: not a valid location identifier |

C:\\tmp\\secrets

-X to enter hidden mode

Enter username:

Enter password for %s:

???