Fall 2016

CODEBREAKER CHALLENGE 4

Challenge Scenario

- Terrorists have recently developed a new type of remotely controlled Improvised Explosive Device (IED), making it harder for the U.S. Armed Forces to detect and ultimately prevent roadside bomb attacks against troops deployed overseas.

- Your task is to develop the capability to disarm the IEDs remotely and permanently render them inoperable without the risk of civilian casualties.

The Challenge

- There are six different levels to this challenge:
  - Task 1: Compute hash and identify IED ports
  - Task 2: Refine IED network traffic signature
  - Task 3: Decrypt IED key file
  - Task 4: Disarm the IED with the key
  - Task 5: Disarm an IED without a key
  - Task 6: Permanently disable any IED

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

Network Traffic Analysis

- Great tools available – packet analysis:
  - Wireshark: cross platform, parsers for many protocols
  - Microsoft Message Analyzer: Great features for active capturing on Windows
- Available features/functionality:
  - Display filters to focus in on traffic
  - TCP stream following
  - Extract files from packet payloads
  - Dissecting custom protocols (Lua script interface)
  - Traffic statistics/characterization

Technical Walkthrough

- 2015 Codebreaker Challenge on Windows using IDA Pro Demo
- This binary can be downloaded from https://codebreaker.ltsnet.net/resources

2015 Backstory

- 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

- Intelligence suggests that each member is provided a program that can be used to read the messages, and that a customized cryptographic implementation is used to generate a public/private key pair, which is then used to authenticate messages from leadership

2015 Backstory (2)

- A copy of the program belonging to a high-ranking operative has been recovered …
- Your mission is to reverse-engineer this software and develop capabilities to exploit the secret messaging component

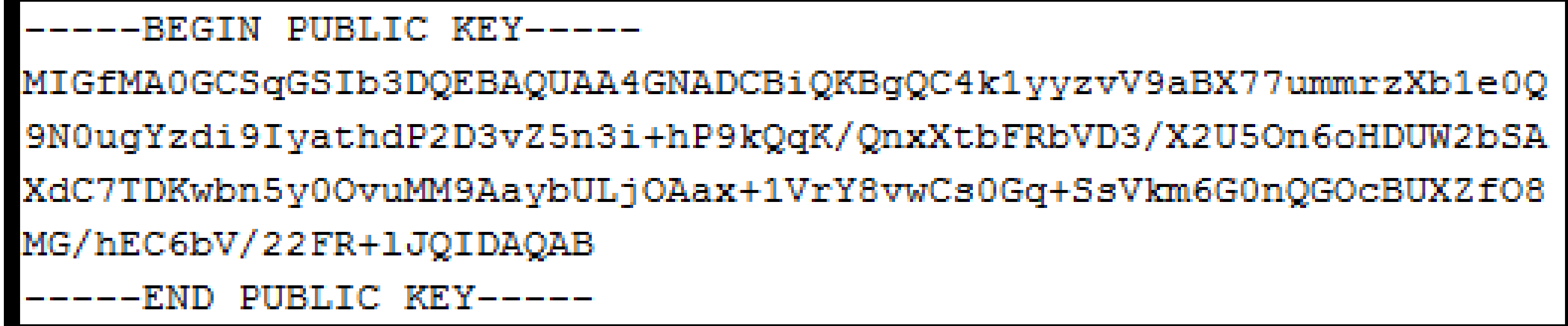
2015 Challenge

- Four different levels or "tasks" to this challenge problem
  - Task 1: Execute program hidden functionality
  - Task 2: Bypass an authentication check
  - Task 3: Create an encoder program
  - Task 4: Spoof a message to a high-value target

2015 Challenge – Task 1

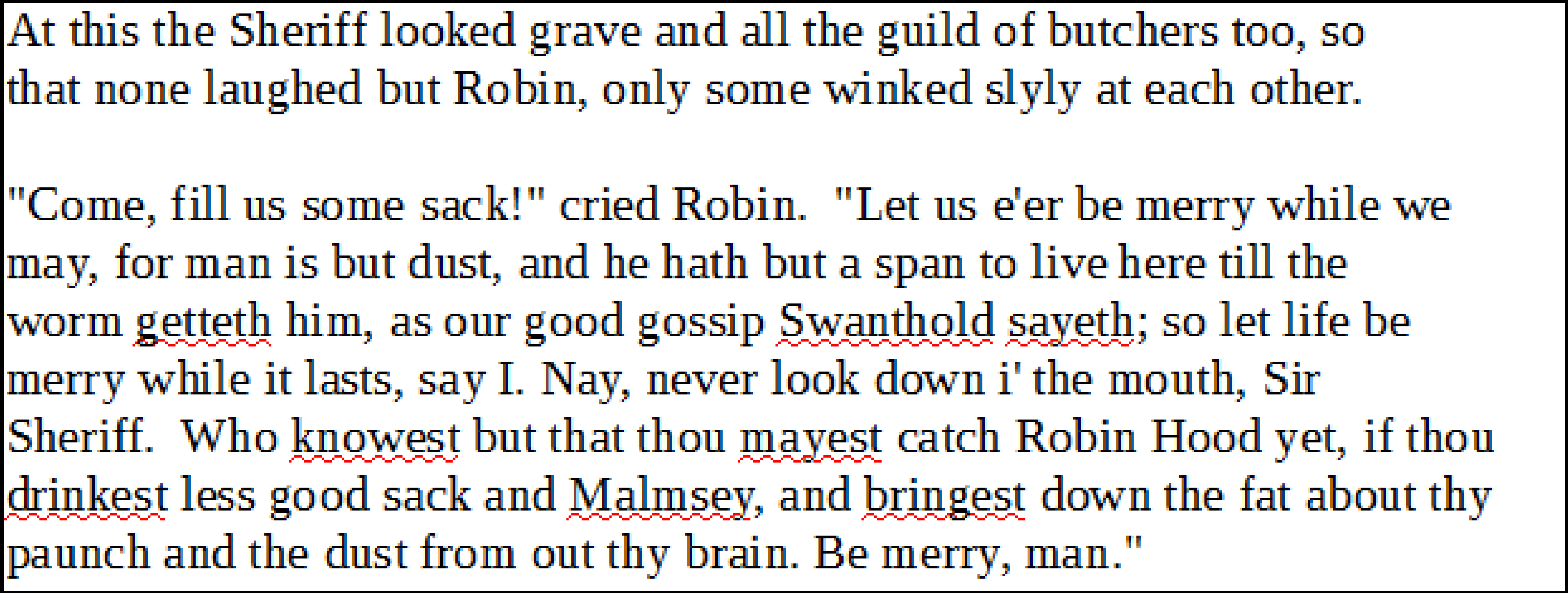
- We need your help with decoding a message that we've captured … trigger the hidden functionality and decode the secret message
- Provided:
  - tier1_key.pem
  - tier1_msg.txt
  - codebreaker3.exe

tier1_key.pem



```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC4k1yyzvV9aBX77ummrzXb1e0Q
9N0ugYzdi9IyathdP2D3vZ5n3i+hP9kQqK/QnxXtbFRbVD3/X2U5On6oHDUW2bSA
XdC7TDKwbn5y0OvuMM9AaybULjOAax+1VrY8vwCs0Gq+SsVkm6G0nQGOcBUXZfO8
MG/hEC6bV/22FR+1JQIDAQAB
-----END PUBLIC KEY-----
```

tier1_msg.txt

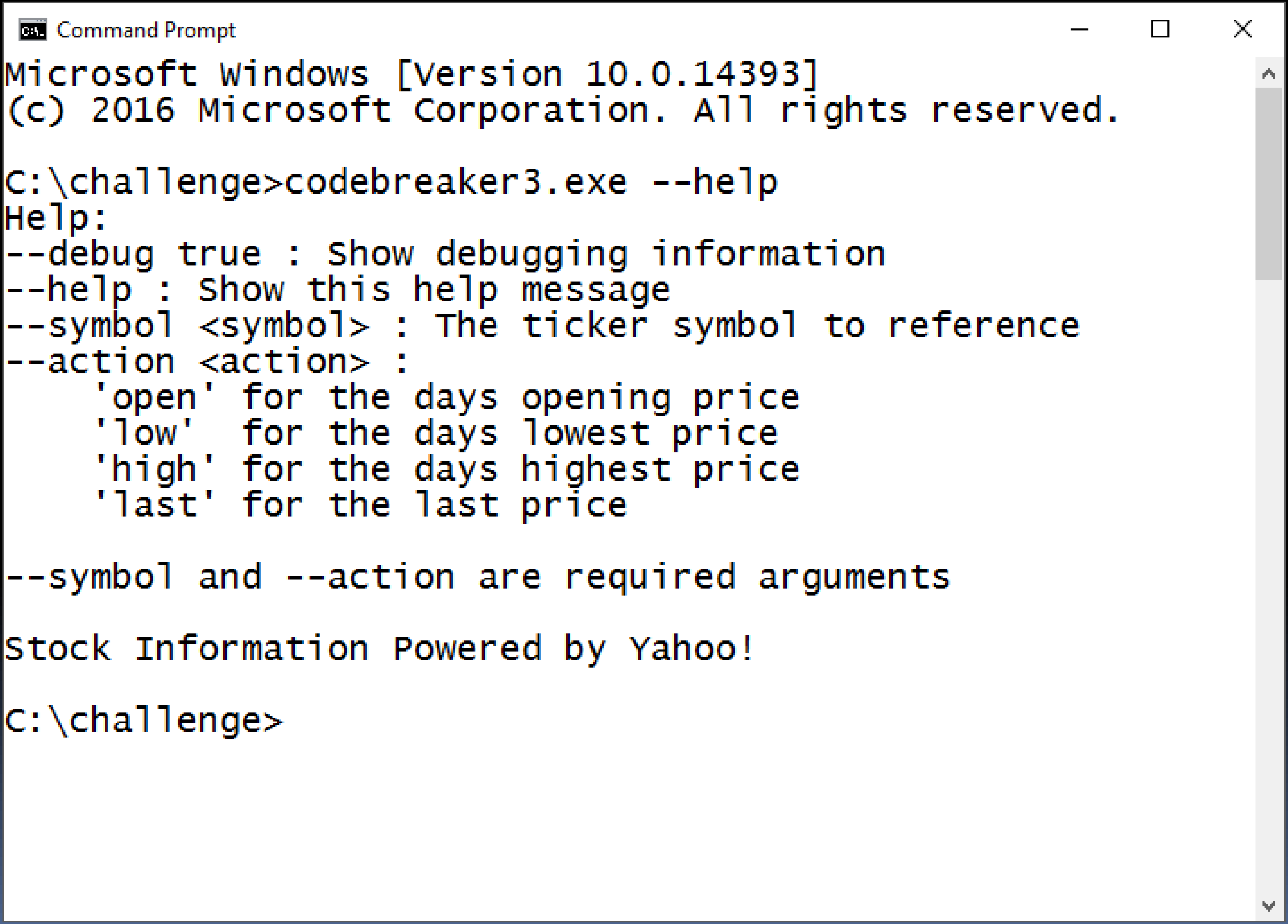


At this the Sheriff looked grave and all the guild of butchers too, so
that none laughed but Robin, only some winked slyly at each other.

"Come, fill us some sack!" cried Robin. "Let us e'er be merry while we
may, for man is but dust, and he hath but a span to live here till the
worm getteth him, as our good gossip Swanthold sayeth; so let life be
merry while it lasts, say I. Nay, never look down i' the mouth, Sir
Sheriff. Who knowest but that thou mayest catch Robin Hood yet, if thou
drinkest less good sack and Malmsey, and bringest down the fat about thy
paunch and the dust from out thy brain. Be merry, man."

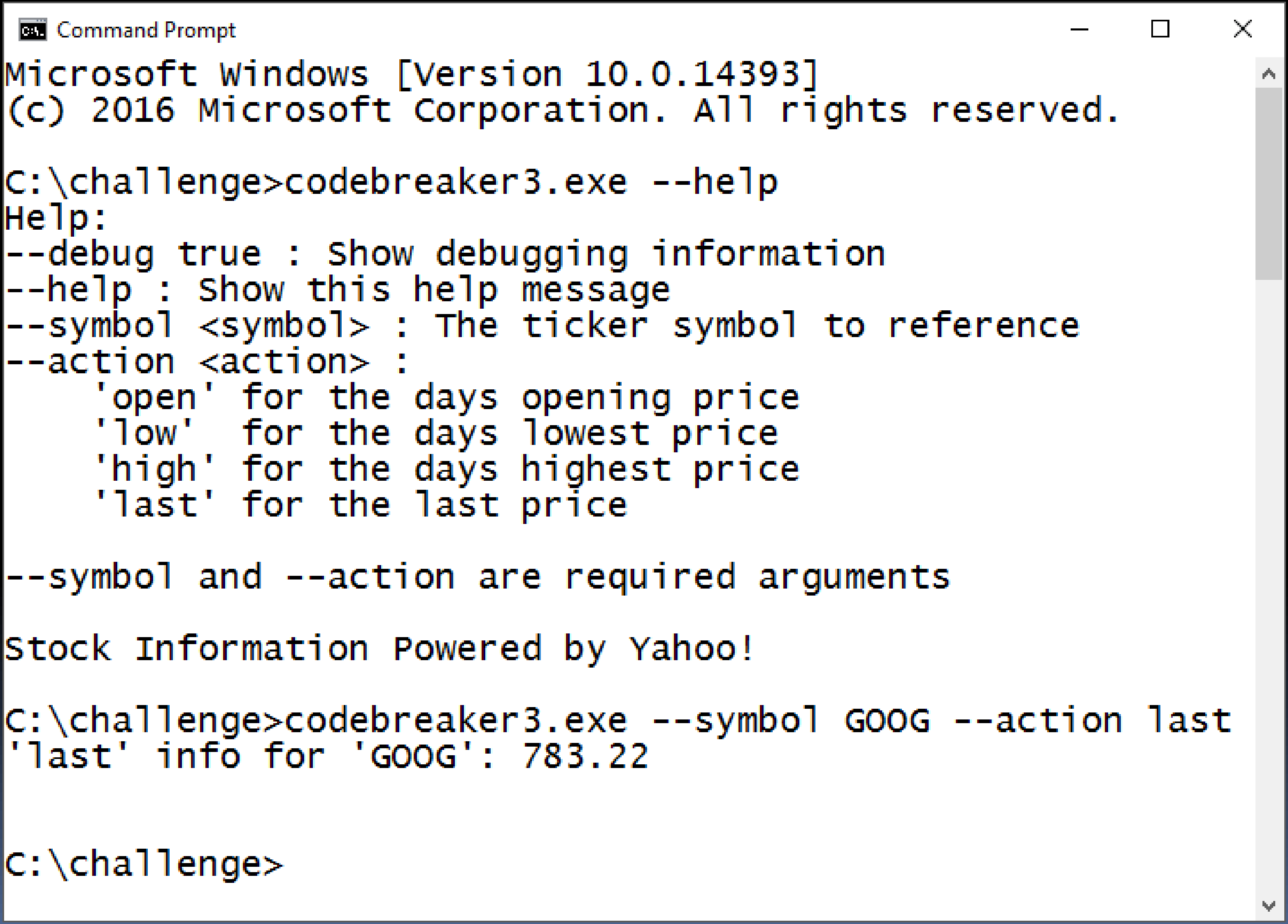
- …

Running the program


Command Prompt

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\challenge>codebreaker3.exe --help
Help:
--debug true : Show debugging information
--help : Show this help message
--symbol <symbol> : The ticker symbol to reference
--action <action> :
    'open' for the days opening price
    'low'  for the days lowest price
    'high' for the days highest price
    'last' for the last price

--symbol and --action are required arguments

Stock Information Powered by Yahoo!

C:\challenge>
```

Running the program (2)

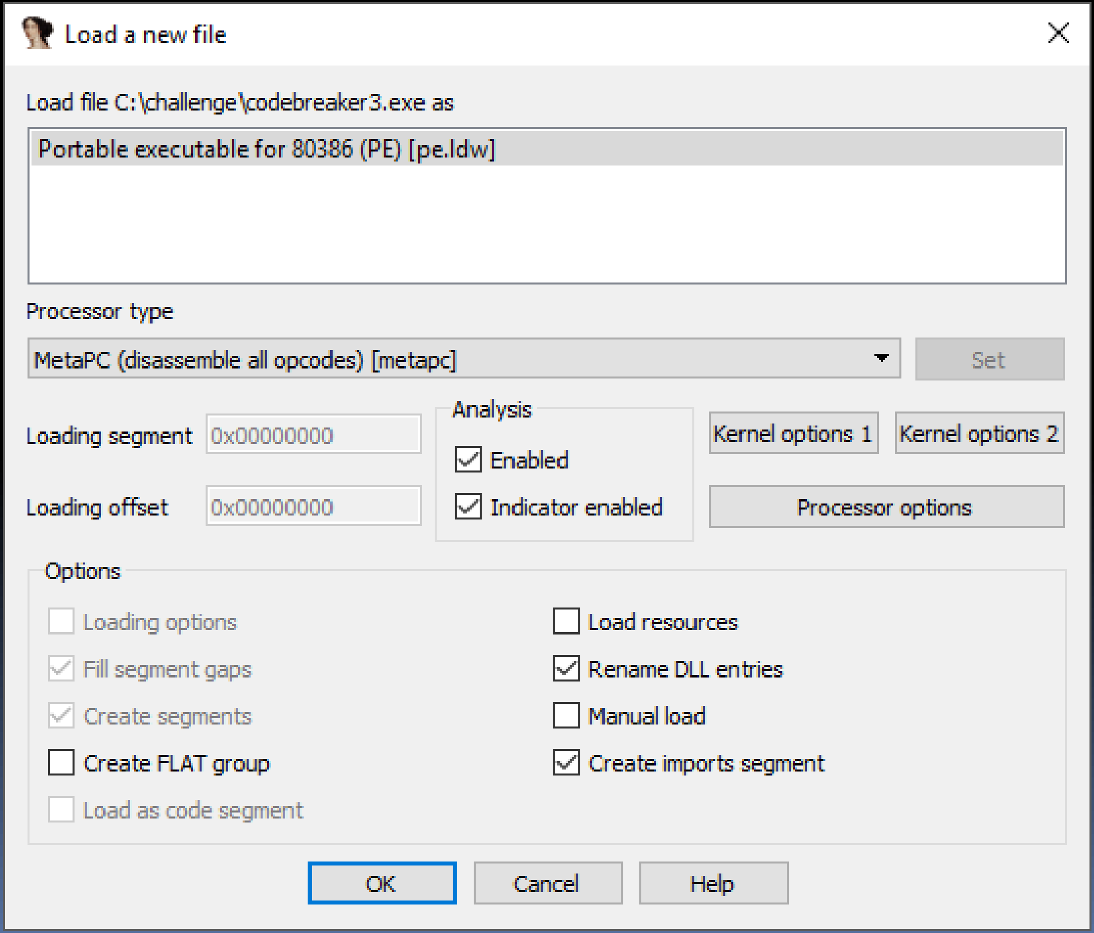


Command Prompt

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\challenge>codebreaker3.exe --help
Help:
--debug true : Show debugging information
--help : Show this help message
--symbol <symbol> : The ticker symbol to reference
--action <action> :
    'open' for the days opening price
    'low'  for the days lowest price
    'high' for the days highest price
    'last' for the last price

--symbol and --action are required arguments

Stock Information Powered by Yahoo!

C:\challenge>codebreaker3.exe --symbol GOOG --action last
'last' info for 'GOOG': 783.22


C:\challenge>
```

Disassemble

- Disassemble the Codebreaker3 binary



Load file C:\challenge\codebreaker3.exe as

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

Processor type

MetaPC (disassemble all opcodes) [metapc]

Loading segment 0x00000000

Loading offset 0x00000000

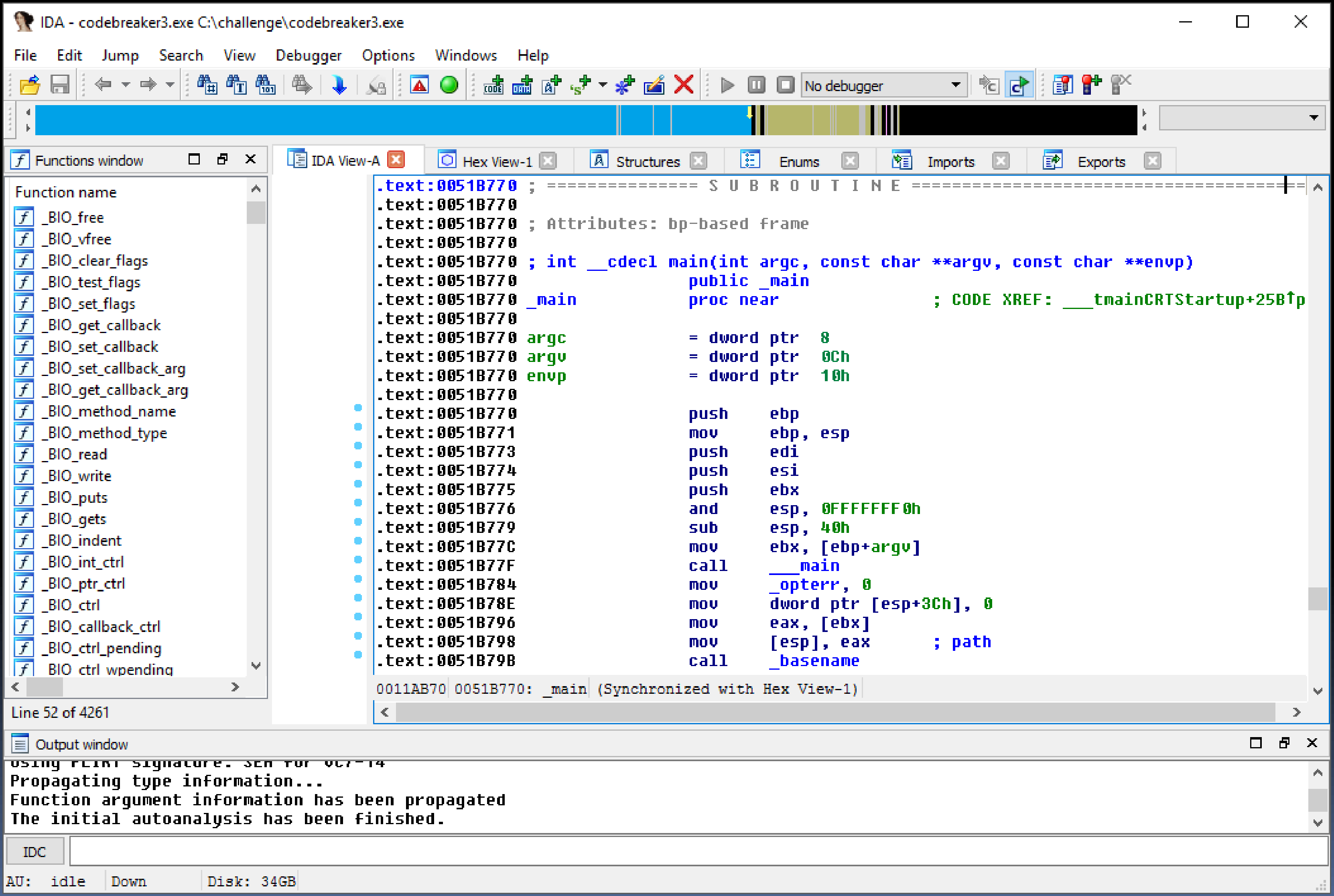
Analysis

- [✓] Enabled
- [✓] Indicator enabled

Options

- [ ] Loading options
- [✓] Fill segment gaps
- [✓] Create segments
- [ ] Create FLAT group
- [ ] Load as code segment
- [ ] Load resources
- [✓] Rename DLL entries
- [ ] Manual load
- [✓] Create imports segment

Disassemble (2)



```
_BIO_free
_BIO_vfree
_BIO_clear_flags
_BIO_test_flags
_BIO_set_flags
_BIO_get_callback
_BIO_set_callback
_BIO_set_callback_arg
_BIO_get_callback_arg
_BIO_method_name
_BIO_method_type
_BIO_read
_BIO_write
_BIO_puts
_BIO_gets
_BIO_indent
_BIO_int_ctrl
_BIO_ptr_ctrl
_BIO_ctrl
_BIO_callback_ctrl
_BIO_ctrl_pending
_BIO_ctrl_wpending
```

```
.text:0051B770 ; =============== S U B R O U T I N E =======================================
.text:0051B770
.text:0051B770 ; Attributes: bp-based frame
.text:0051B770
.text:0051B770 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:0051B770                 public _main
.text:0051B770 _main           proc near               ; CODE XREF: ___tmainCRTStartup+25B↑p
.text:0051B770
.text:0051B770 argc            = dword ptr  8
.text:0051B770 argv            = dword ptr  0Ch
.text:0051B770 envp            = dword ptr  10h
.text:0051B770
.text:0051B770                 push    ebp
.text:0051B771                 mov     ebp, esp
.text:0051B773                 push    edi
.text:0051B774                 push    esi
.text:0051B775                 push    ebx
.text:0051B776                 and     esp, 0FFFFFFF0h
.text:0051B779                 sub     esp, 40h
.text:0051B77C                 mov     ebx, [ebp+argv]
.text:0051B77F                 call    ___main
.text:0051B784                 mov     _opterr, 0
.text:0051B78E                 mov     dword ptr [esp+3Ch], 0
.text:0051B796                 mov     eax, [ebx]
.text:0051B798                 mov     [esp], eax      ; path
.text:0051B79B                 call    _basename
0011AB70 0051B770: _main (Synchronized with Hex View-1)
```

```
Using FLIRT signature: SEH for vc7-14
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
```

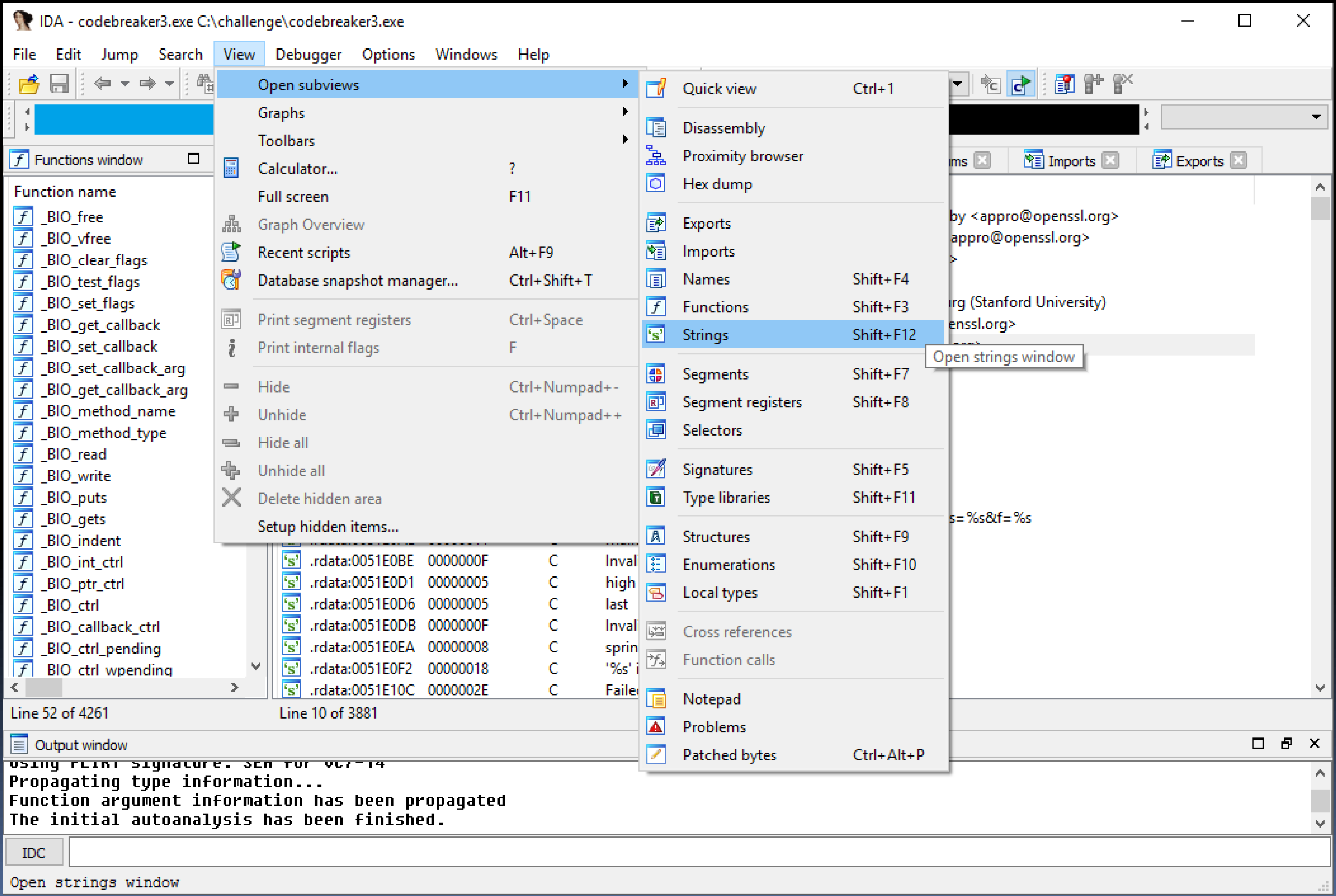
```
AU:  idle   Down      Disk: 34GB
```

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

```
--symbol <symbol> : The ticker symbol to reference

--action <action> :

--symbol and --action are required arguments

Stock information powered by Yahoo!
```

Observe Strings (2)



```
_BIO_free
_BIO_vfree
_BIO_clear_flags
_BIO_test_flags
_BIO_set_flags
_BIO_get_callback
_BIO_set_callback
_BIO_set_callback_arg
_BIO_get_callback_arg
_BIO_method_name
_BIO_method_type
_BIO_read
_BIO_write
_BIO_puts
_BIO_gets
_BIO_indent
_BIO_int_ctrl
_BIO_ptr_ctrl
_BIO_ctrl
_BIO_callback_ctrl
_BIO_ctrl_pending
_BIO_ctrl_wpending
```

```
.rdata:0051E0BE  0000000F  C  Inval
.rdata:0051E0D1  00000005  C  high
.rdata:0051E0D6  00000005  C  last
.rdata:0051E0DB  0000000F  C  Inval
.rdata:0051E0EA  00000008  C  sprin
.rdata:0051E0F2  00000018  C  '%s' i
.rdata:0051E10C  0000002E  C  Faile
```

```
by <appro@openssl.org>
appro@openssl.org>
rg (Stanford University)
enssl.org>
s=%s&f=%s
```

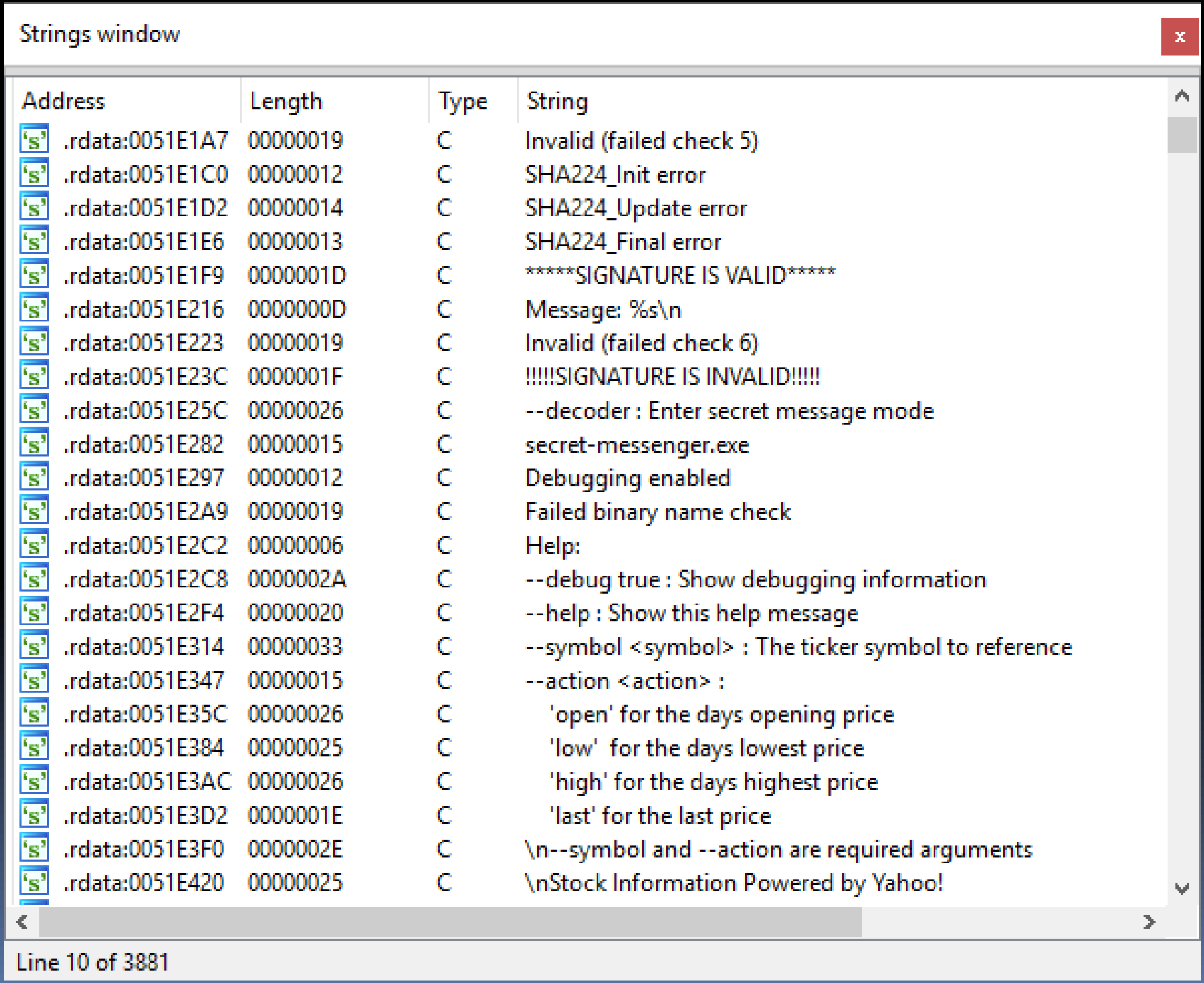
Line 52 of 4261

Line 10 of 3881

```
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
```

```
Open strings window
```

Observe Strings (3)

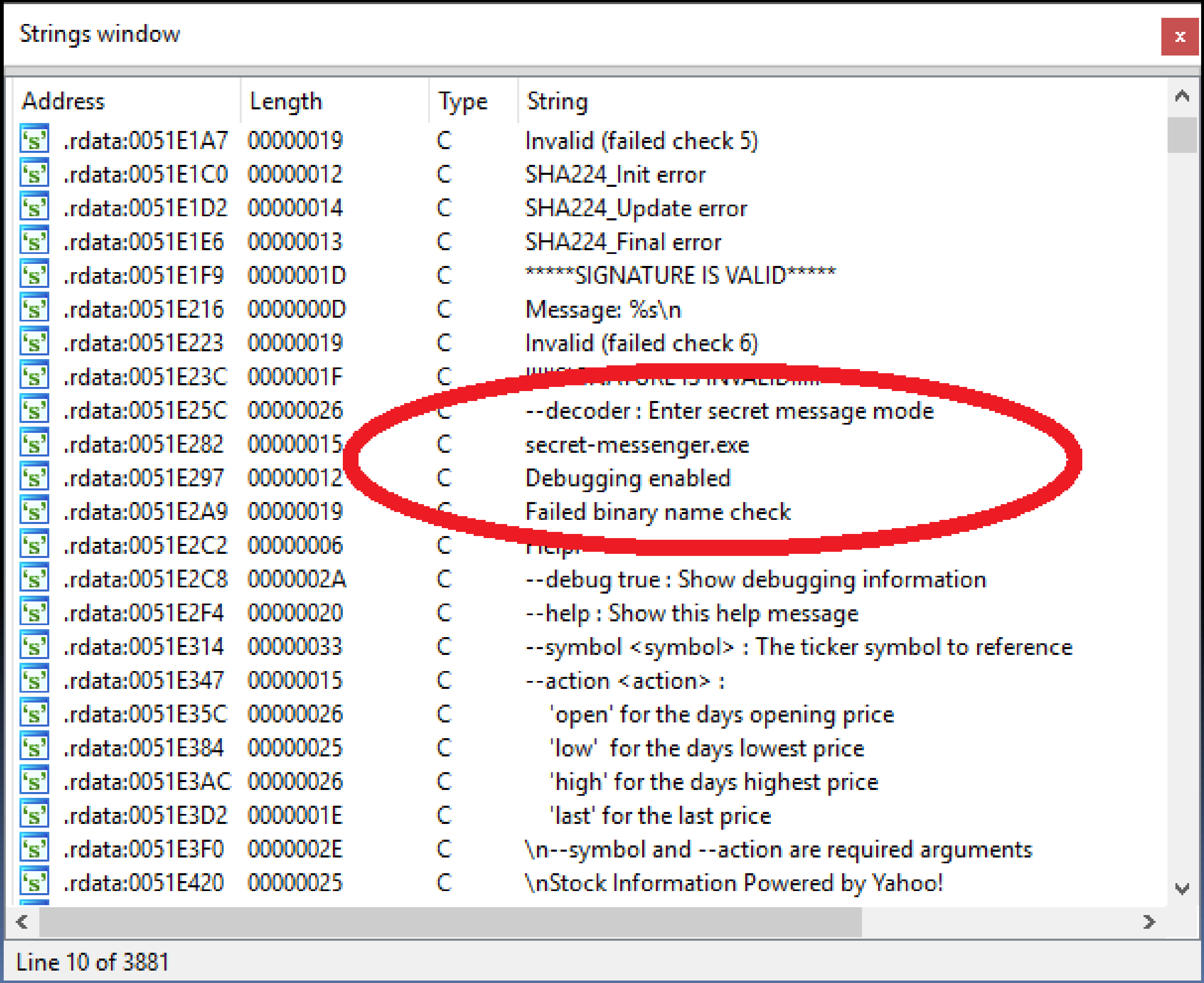


Strings window

Address	Length	Type	String
.rdata:0051E1A7	00000019	C	Invalid (failed check 5)
.rdata:0051E1C0	00000012	C	SHA224_Init error
.rdata:0051E1D2	00000014	C	SHA224_Update error
.rdata:0051E1E6	00000013	C	SHA224_Final error
.rdata:0051E1F9	0000001D	C	*****SIGNATURE IS VALID*****
.rdata:0051E216	0000000D	C	Message: %s\n
.rdata:0051E223	00000019	C	Invalid (failed check 6)
.rdata:0051E23C	0000001F	C	!!!!SIGNATURE IS INVALID!!!!
.rdata:0051E25C	00000026	C	--decoder : Enter secret message mode
.rdata:0051E282	00000015	C	secret-messenger.exe
.rdata:0051E297	00000012	C	Debugging enabled
.rdata:0051E2A9	00000019	C	Failed binary name check
.rdata:0051E2C2	00000006	C	Help:
.rdata:0051E2C8	0000002A	C	--debug true : Show debugging information
.rdata:0051E2F4	00000020	C	--help : Show this help message
.rdata:0051E314	00000033	C	--symbol <symbol> : The ticker symbol to reference
.rdata:0051E347	00000015	C	--action <action> :
.rdata:0051E35C	00000026	C	'open' for the days opening price
.rdata:0051E384	00000025	C	'low' for the days lowest price
.rdata:0051E3AC	00000026	C	'high' for the days highest price
.rdata:0051E3D2	0000001E	C	'last' for the last price
.rdata:0051E3F0	0000002E	C	\n--symbol and --action are required arguments
.rdata:0051E420	00000025	C	\nStock Information Powered by Yahoo!

Line 10 of 3881

Observe Strings (4)

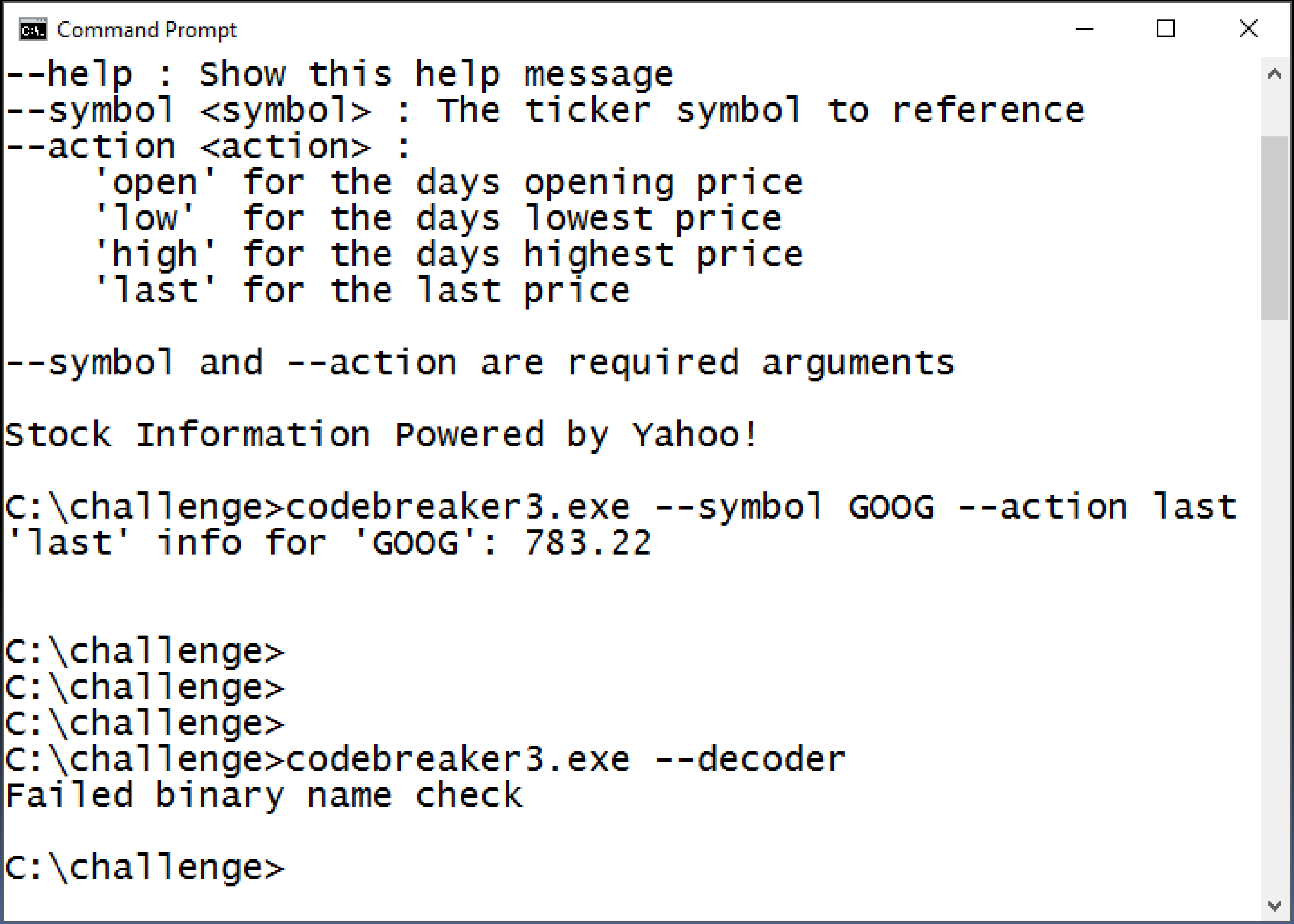


Strings window

Address	Length	Type	String
.rdata:0051E1A7	00000019	C	Invalid (failed check 5)
.rdata:0051E1C0	00000012	C	SHA224_Init error
.rdata:0051E1D2	00000014	C	SHA224_Update error
.rdata:0051E1E6	00000013	C	SHA224_Final error
.rdata:0051E1F9	0000001D	C	*****SIGNATURE IS VALID*****
.rdata:0051E216	0000000D	C	Message: %s\n
.rdata:0051E223	00000019	C	Invalid (failed check 6)
.rdata:0051E23C	0000001F	C	!!!!!SIGNATURE IS INVALID!!!!!
.rdata:0051E25C	00000026	C	--decoder : Enter secret message mode
.rdata:0051E282	00000015	C	secret-messenger.exe
.rdata:0051E297	00000012	C	Debugging enabled
.rdata:0051E2A9	00000019	C	Failed binary name check
.rdata:0051E2C2	00000006	C	Help:
.rdata:0051E2C8	0000002A	C	--debug true : Show debugging information
.rdata:0051E2F4	00000020	C	--help : Show this help message
.rdata:0051E314	00000033	C	--symbol <symbol> : The ticker symbol to reference
.rdata:0051E347	00000015	C	--action <action> :
.rdata:0051E35C	00000026	C	'open' for the days opening price
.rdata:0051E384	00000025	C	'low' for the days lowest price
.rdata:0051E3AC	00000026	C	'high' for the days highest price
.rdata:0051E3D2	0000001E	C	'last' for the last price
.rdata:0051E3F0	0000002E	C	\n--symbol and --action are required arguments
.rdata:0051E420	00000025	C	\nStock Information Powered by Yahoo!

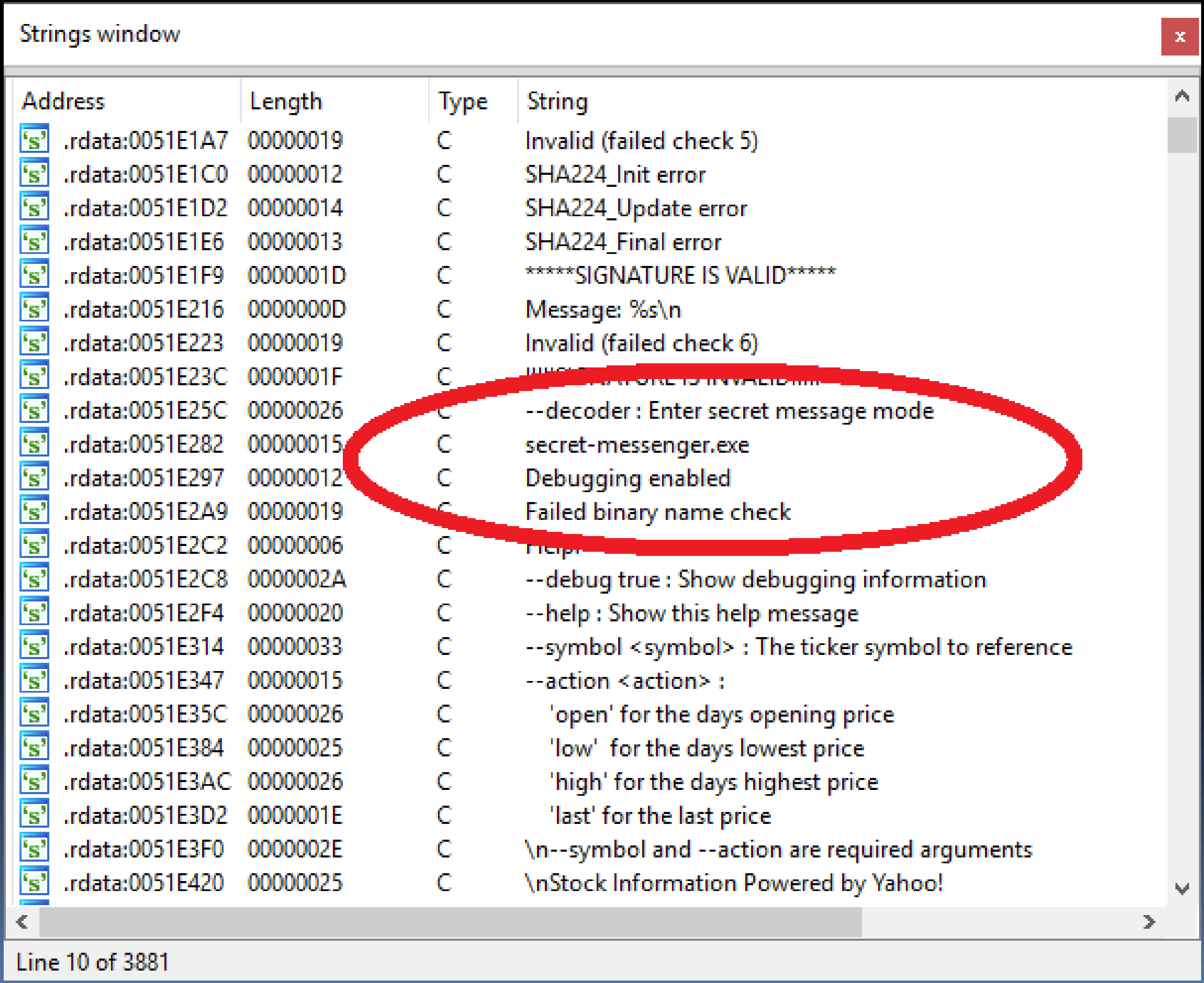
Line 10 of 3881

Running the program (3)



```
--help : Show this help message
--symbol <symbol> : The ticker symbol to reference
--action <action> :
    'open' for the days opening price
    'low'  for the days lowest price
    'high' for the days highest price
    'last' for the last price

--symbol and --action are required arguments

Stock Information Powered by Yahoo!

C:\challenge>codebreaker3.exe --symbol GOOG --action last
'last' info for 'GOOG': 783.22


C:\challenge>
C:\challenge>
C:\challenge>
C:\challenge>codebreaker3.exe --decoder
Failed binary name check

C:\challenge>
```

Observe Strings (4)

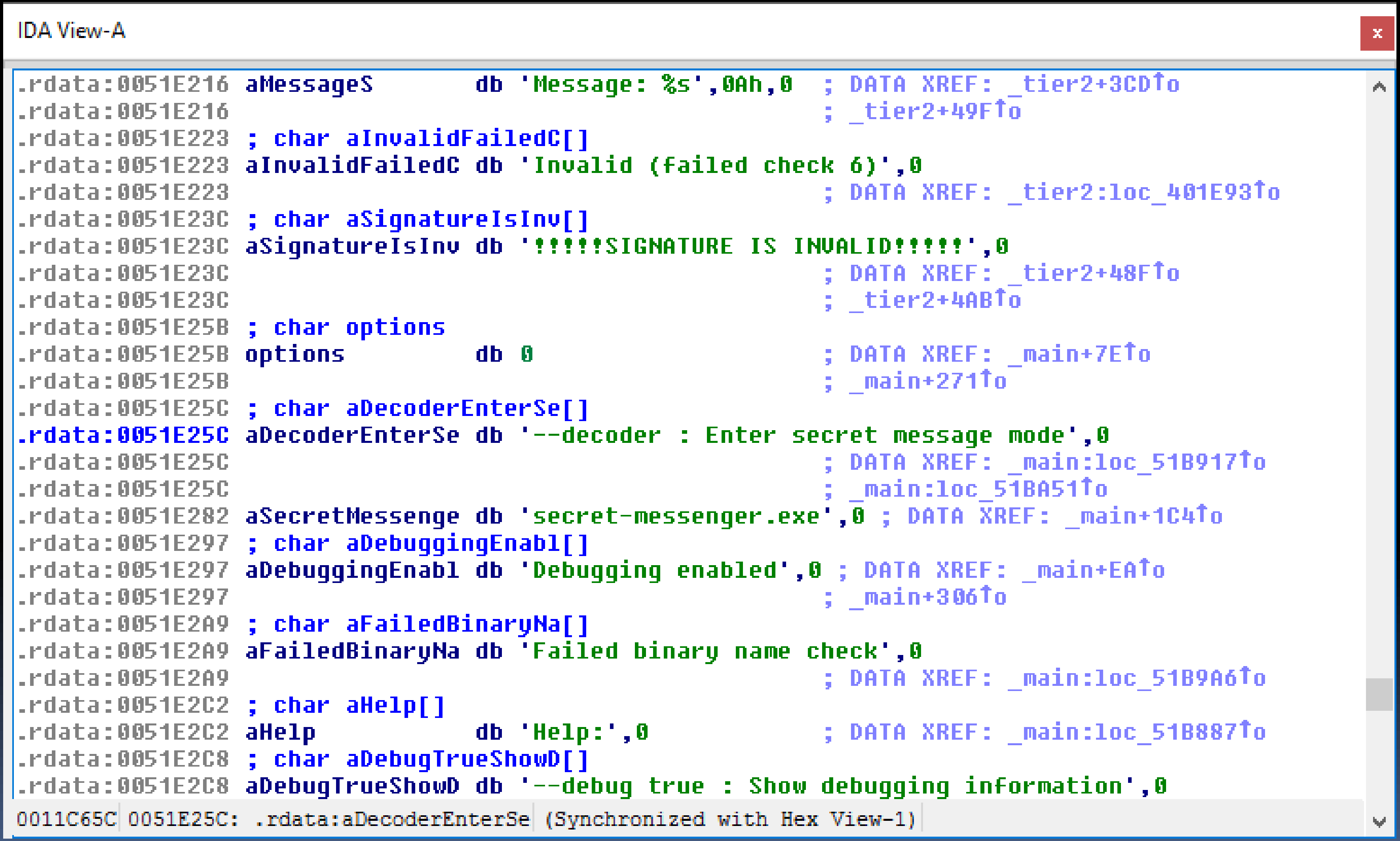


Strings window

Address	Length	Type	String
.rdata:0051E1A7	00000019	C	Invalid (failed check 5)
.rdata:0051E1C0	00000012	C	SHA224_Init error
.rdata:0051E1D2	00000014	C	SHA224_Update error
.rdata:0051E1E6	00000013	C	SHA224_Final error
.rdata:0051E1F9	0000001D	C	*****SIGNATURE IS VALID*****
.rdata:0051E216	0000000D	C	Message: %s\n
.rdata:0051E223	00000019	C	Invalid (failed check 6)
.rdata:0051E23C	0000001F	C	!!!!!SIGNATURE IS INVALID!!!!!
.rdata:0051E25C	00000026	C	--decoder : Enter secret message mode
.rdata:0051E282	00000015	C	secret-messenger.exe
.rdata:0051E297	00000012	C	Debugging enabled
.rdata:0051E2A9	00000019	C	Failed binary name check
.rdata:0051E2C2	00000006	C	Help:
.rdata:0051E2C8	0000002A	C	--debug true : Show debugging information
.rdata:0051E2F4	00000020	C	--help : Show this help message
.rdata:0051E314	00000033	C	--symbol <symbol> : The ticker symbol to reference
.rdata:0051E347	00000015	C	--action <action> :
.rdata:0051E35C	00000026	C	'open' for the days opening price
.rdata:0051E384	00000025	C	'low' for the days lowest price
.rdata:0051E3AC	00000026	C	'high' for the days highest price
.rdata:0051E3D2	0000001E	C	'last' for the last price
.rdata:0051E3F0	0000002E	C	\n--symbol and --action are required arguments
.rdata:0051E420	00000025	C	\nStock Information Powered by Yahoo!

Line 10 of 3881

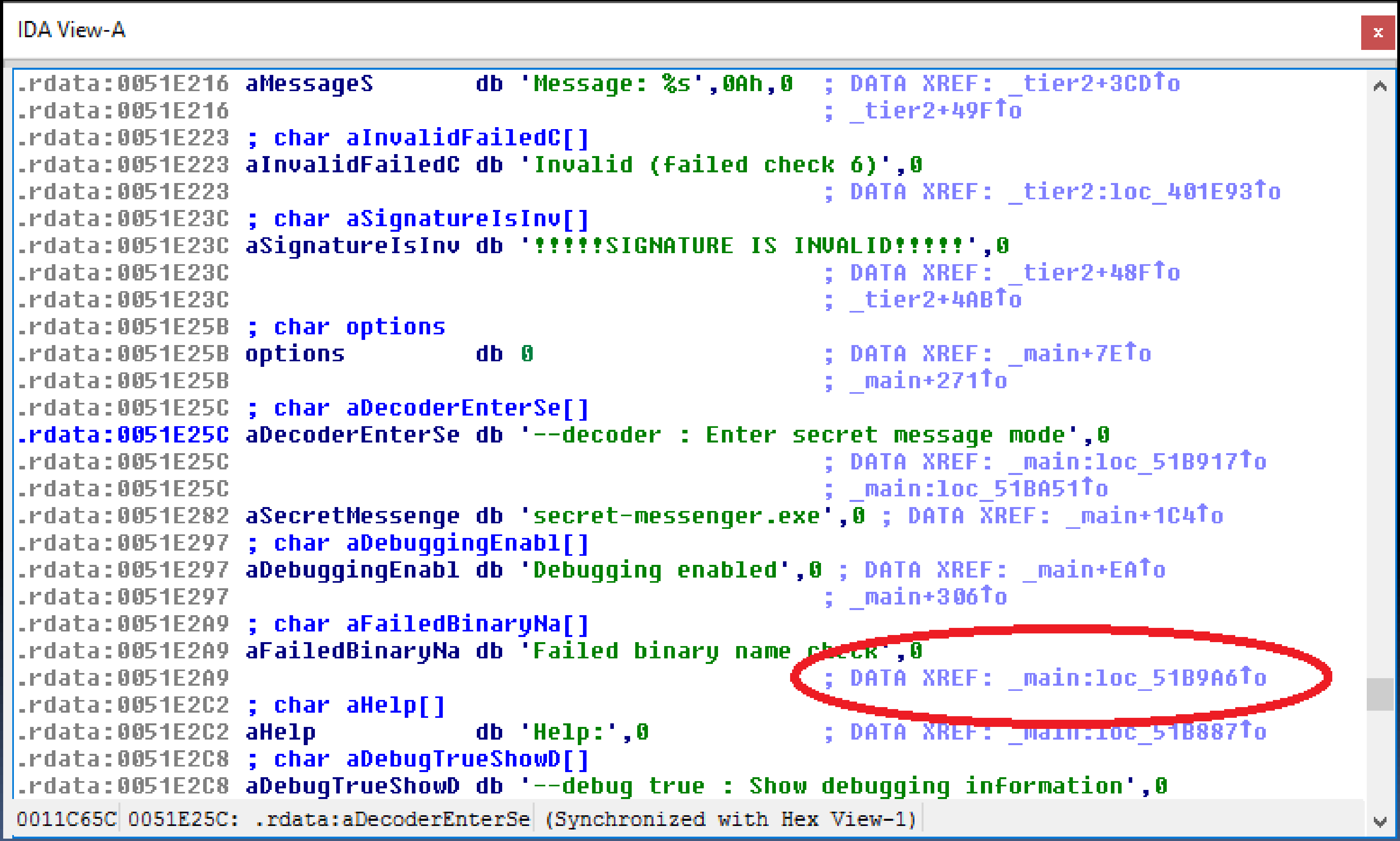
Failed Binary Name Check



IDA View-A

```
.rdata:0051E216 aMessageS       db 'Message: %s',0Ah,0  ; DATA XREF: _tier2+3CD↑o
.rdata:0051E216                                         ; _tier2+49F↑o
.rdata:0051E223 ; char aInvalidFailedC[]
.rdata:0051E223 aInvalidFailedC db 'Invalid (failed check 6)',0
.rdata:0051E223                                         ; DATA XREF: _tier2:loc_401E93↑o
.rdata:0051E23C ; char aSignatureIsInv[]
.rdata:0051E23C aSignatureIsInv db '!!!!!SIGNATURE IS INVALID!!!!!',0
.rdata:0051E23C                                         ; DATA XREF: _tier2+48F↑o
.rdata:0051E23C                                         ; _tier2+4AB↑o
.rdata:0051E25B ; char options
.rdata:0051E25B options         db 0                    ; DATA XREF: _main+7E↑o
.rdata:0051E25B                                         ; _main+271↑o
.rdata:0051E25C ; char aDecoderEnterSe[]
.rdata:0051E25C aDecoderEnterSe db '--decoder : Enter secret message mode',0
.rdata:0051E25C                                         ; DATA XREF: _main:loc_51B917↑o
.rdata:0051E25C                                         ; _main:loc_51BA51↑o
.rdata:0051E282 aSecretMessenge db 'secret-messenger.exe',0 ; DATA XREF: _main+1C4↑o
.rdata:0051E297 ; char aDebuggingEnabl[]
.rdata:0051E297 aDebuggingEnabl db 'Debugging enabled',0 ; DATA XREF: _main+EA↑o
.rdata:0051E297                                         ; _main+306↑o
.rdata:0051E2A9 ; char aFailedBinaryNa[]
.rdata:0051E2A9 aFailedBinaryNa db 'Failed binary name check',0
.rdata:0051E2A9                                         ; DATA XREF: _main:loc_51B9A6↑o
.rdata:0051E2C2 ; char aHelp[]
.rdata:0051E2C2 aHelp           db 'Help:',0            ; DATA XREF: _main:loc_51B887↑o
.rdata:0051E2C8 ; char aDebugTrueShowD[]
.rdata:0051E2C8 aDebugTrueShowD db '--debug true : Show debugging information',0
0011C65C 0051E25C: .rdata:aDecoderEnterSe (Synchronized with Hex View-1)
```

Failed Binary Name Check (2)



```
.rdata:0051E216 aMessageS       db 'Message: %s',0Ah,0  ; DATA XREF: _tier2+3CD↑o
.rdata:0051E216                                         ; _tier2+49F↑o
.rdata:0051E223 ; char aInvalidFailedC[]
.rdata:0051E223 aInvalidFailedC db 'Invalid (failed check 6)',0
.rdata:0051E223                                         ; DATA XREF: _tier2:loc_401E93↑o
.rdata:0051E23C ; char aSignatureIsInv[]
.rdata:0051E23C aSignatureIsInv db '!!!!!SIGNATURE IS INVALID!!!!!',0
.rdata:0051E23C                                         ; DATA XREF: _tier2+48F↑o
.rdata:0051E23C                                         ; _tier2+4AB↑o
.rdata:0051E25B ; char options
.rdata:0051E25B options         db 0                    ; DATA XREF: _main+7E↑o
.rdata:0051E25B                                         ; _main+271↑o
.rdata:0051E25C ; char aDecoderEnterSe[]
.rdata:0051E25C aDecoderEnterSe db '--decoder : Enter secret message mode',0
.rdata:0051E25C                                         ; DATA XREF: _main:loc_51B917↑o
.rdata:0051E25C                                         ; _main:loc_51BA51↑o
.rdata:0051E282 aSecretMessenge db 'secret-messenger.exe',0 ; DATA XREF: _main+1C4↑o
.rdata:0051E297 ; char aDebuggingEnabl[]
.rdata:0051E297 aDebuggingEnabl db 'Debugging enabled',0 ; DATA XREF: _main+EA↑o
.rdata:0051E297                                         ; _main+306↑o
.rdata:0051E2A9 ; char aFailedBinaryNa[]
.rdata:0051E2A9 aFailedBinaryNa db 'Failed binary name check',0
.rdata:0051E2A9                                         ; DATA XREF: _main:loc_51B9A6↑o
.rdata:0051E2C2 ; char aHelp[]
.rdata:0051E2C2 aHelp           db 'Help:',0            ; DATA XREF: _main:loc_51B887↑o
.rdata:0051E2C8 ; char aDebugTrueShowD[]
.rdata:0051E2C8 aDebugTrueShowD db '--debug true : Show debugging information',0
0011C65C 0051E25C: .rdata:aDecoderEnterSe (Synchronized with Hex View-1)
```

Double-click Reference

- You should now be looking at disassembled x86 code
  - We just leveraged the fact that in order to use “Failed binary name check” 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)

```
loc_51B9A6:        ; CODE XREF: _main+1D2j
                   ;             _main+29Aj

mov     dword ptr [esp], offset "Failed binary name check"
call    _puts
mov     dword ptr [esp], 1
call    _exit
```

Double-click Reference (3)

```
mov     dword ptr [esp+8], 15h
mov     [esp+4], edi
mov     dword ptr [esp], offset "secret-messenger.exe"
call    _memcmp
test    eax, eax
jnz     short loc_51B9A6 ; Previous code block
```

So, in C:

```
if(0 != memcmp( <edi>, “secret-messenger.exe”, 21) ) {
    puts(“Failed binary name check”); exit(1); }
```

Double-click Reference (4)

```
; int main(int argc, const char **argv, const char **envp)
...
mov     ebx, [ebp+argv]
...
mov     eax, [ebx]
mov     [esp], eax          ; path
call    _basename
mov     [esp], eax          ; char *
mov     edi, eax
```

argv holds the program arguments. For our invocation, argv will be:

[‘C:\challenge\codebreaker3.exe’, ‘--decoder’]

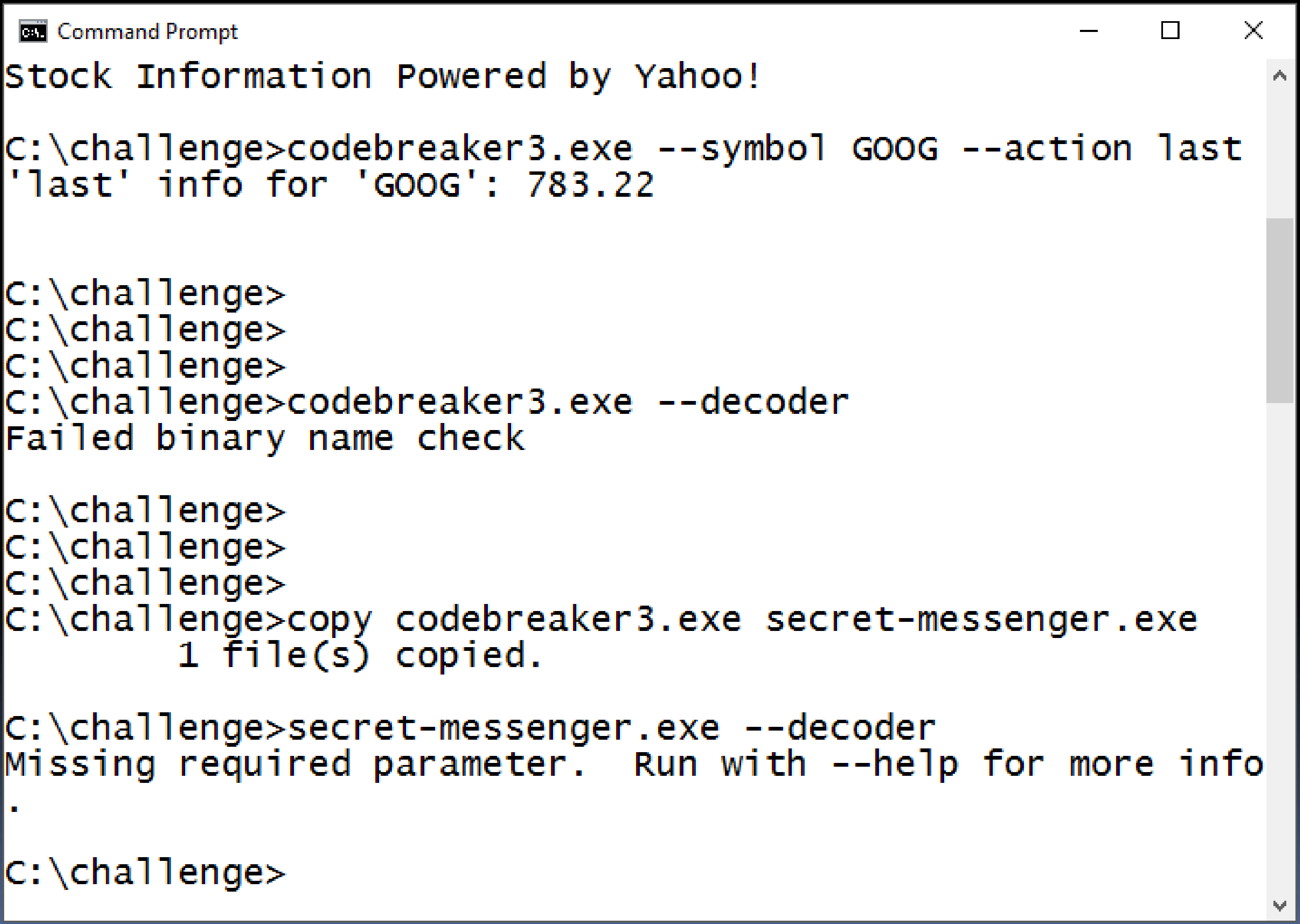
So, here, edi is a pointer to “codebreaker3.exe”

Double-click Reference (5)

So, in C:

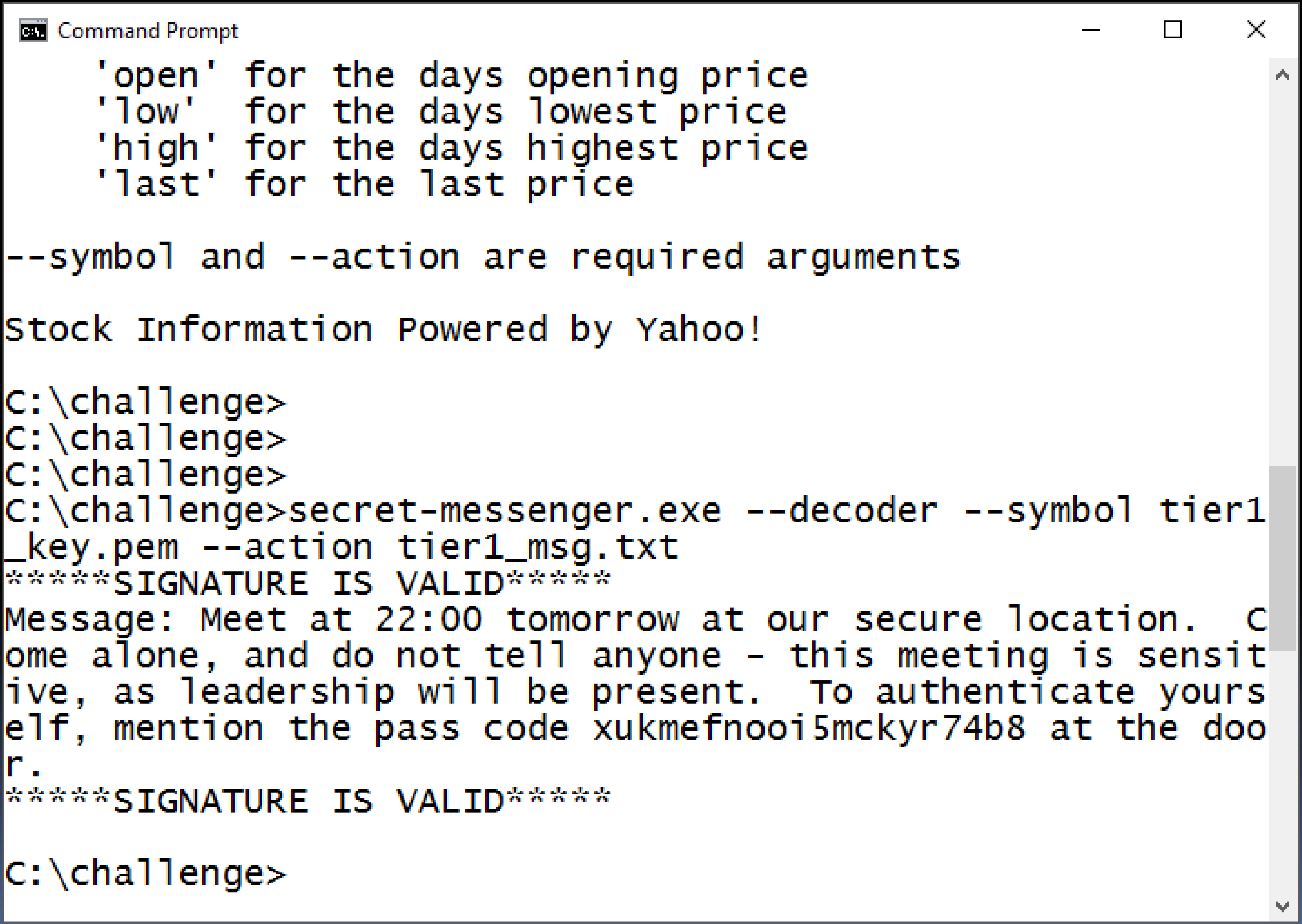
```
if(0 != memcmp( basename(argv[0]), “secret-messenger.exe”, 21) ) {
    puts(“Failed binary name check”); exit(1); }
```

Running the program (4)



```
Stock Information Powered by Yahoo!

C:\challenge>codebreaker3.exe --symbol GOOG --action last
'last' info for 'GOOG': 783.22


C:\challenge>
C:\challenge>
C:\challenge>
C:\challenge>codebreaker3.exe --decoder
Failed binary name check

C:\challenge>
C:\challenge>
C:\challenge>
C:\challenge>copy codebreaker3.exe secret-messenger.exe
        1 file(s) copied.

C:\challenge>secret-messenger.exe --decoder
Missing required parameter.  Run with --help for more info
.

C:\challenge>
```

Running the program (5)



Command Prompt

```
    'open' for the days opening price
    'low'  for the days lowest price
    'high' for the days highest price
    'last' for the last price

--symbol and --action are required arguments

Stock Information Powered by Yahoo!

C:\challenge>
C:\challenge>
C:\challenge>
C:\challenge>secret-messenger.exe --decoder --symbol tier1
_key.pem --action tier1_msg.txt
*****SIGNATURE IS VALID*****
Message: Meet at 22:00 tomorrow at our secure location.  C
ome alone, and do not tell anyone - this meeting is sensit
ive, as leadership will be present.  To authenticate yours
elf, mention the pass code xukmefnooi5mckyr74b8 at the doo
r.
*****SIGNATURE IS VALID*****

C:\challenge>
```

Task 1 Complete!

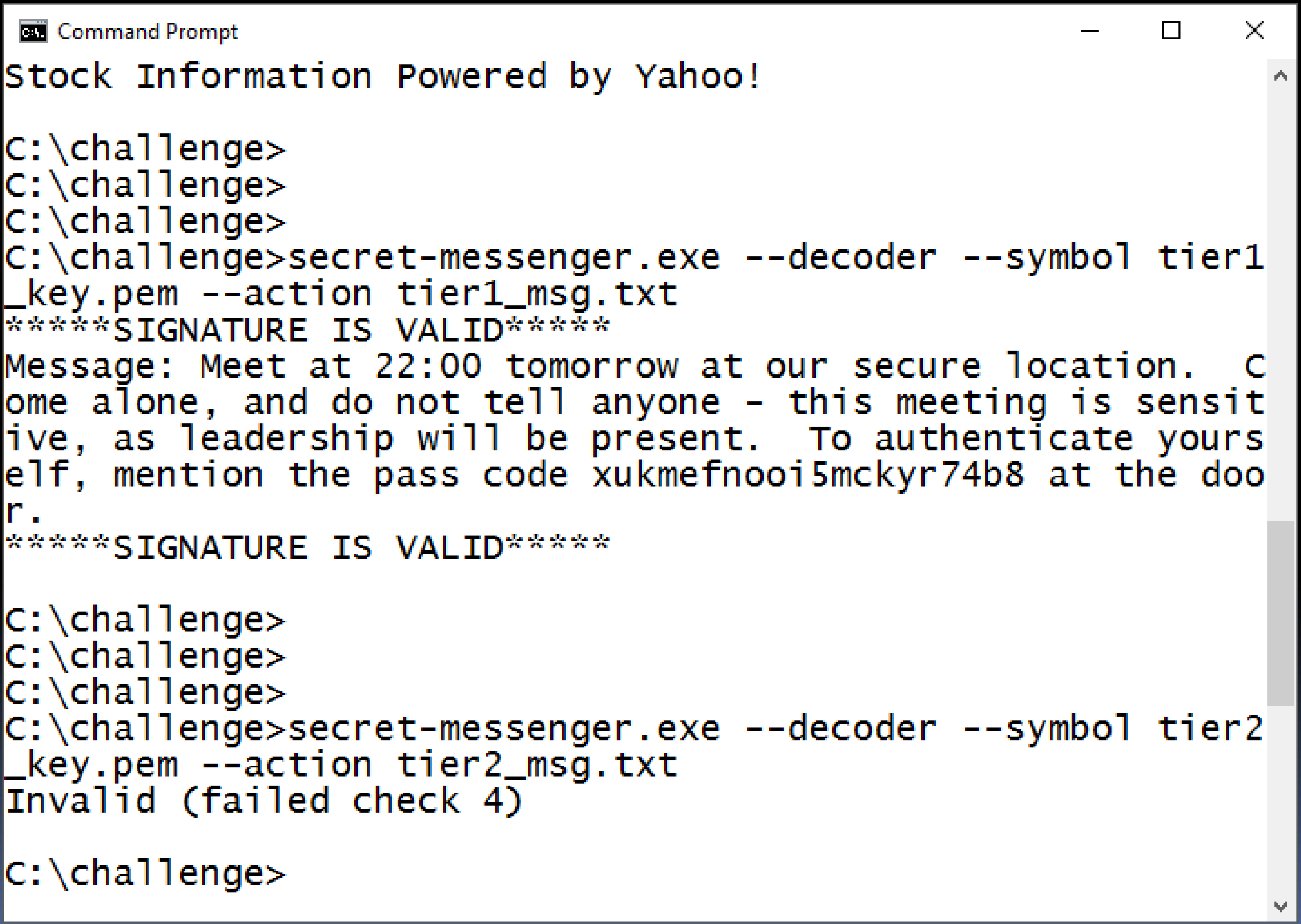
- Fairly straight forward
- Just looking at the strings may have been enough to get you through this
  - --decoder : Enter secret messaging mode
  - secret-messenger.exe

- … on to Task 2!

2015 Challenge – Task 2

- Through SIGINT we have collected a new message file - this one appears to have been sent to a field operative … We believe that this message may contain actionable intelligence, so please report back with the message contents as soon as possible
- Provided:
  - tier2_key.pem
  - tier2_msg.txt

Running the program (6)



```
Stock Information Powered by Yahoo!

C:\challenge>
C:\challenge>
C:\challenge>
C:\challenge>secret-messenger.exe --decoder --symbol tier1
_key.pem --action tier1_msg.txt
*****SIGNATURE IS VALID*****
Message: Meet at 22:00 tomorrow at our secure location.  C
ome alone, and do not tell anyone - this meeting is sensit
ive, as leadership will be present.  To authenticate yours
elf, mention the pass code xukmefnooi5mckyr74b8 at the doo
r.
*****SIGNATURE IS VALID*****

C:\challenge>
C:\challenge>
C:\challenge>
C:\challenge>secret-messenger.exe --decoder --symbol tier2
_key.pem --action tier2_msg.txt
Invalid (failed check 4)

C:\challenge>
```

Invalid (Failed check 4)

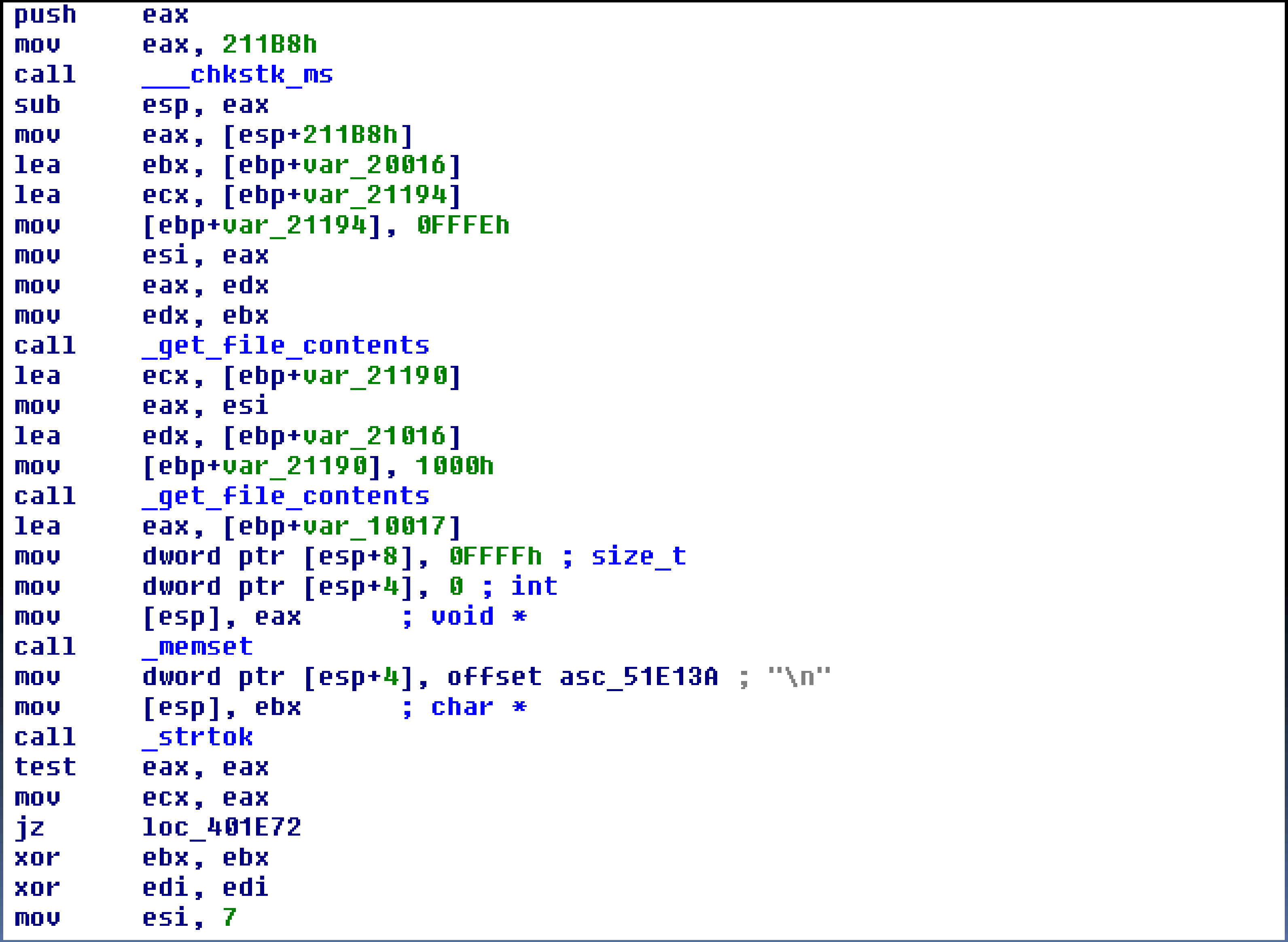
```
loc_401ED3:          ; CODE XREF: _tier2+1E6j

mov     dword ptr [esp], offset "Invalid (failed check 4)"
call    _puts
mov     dword ptr [esp], 1
call    _exit
```

On to _tier2

Starting near where we left off, main calls _tier2:

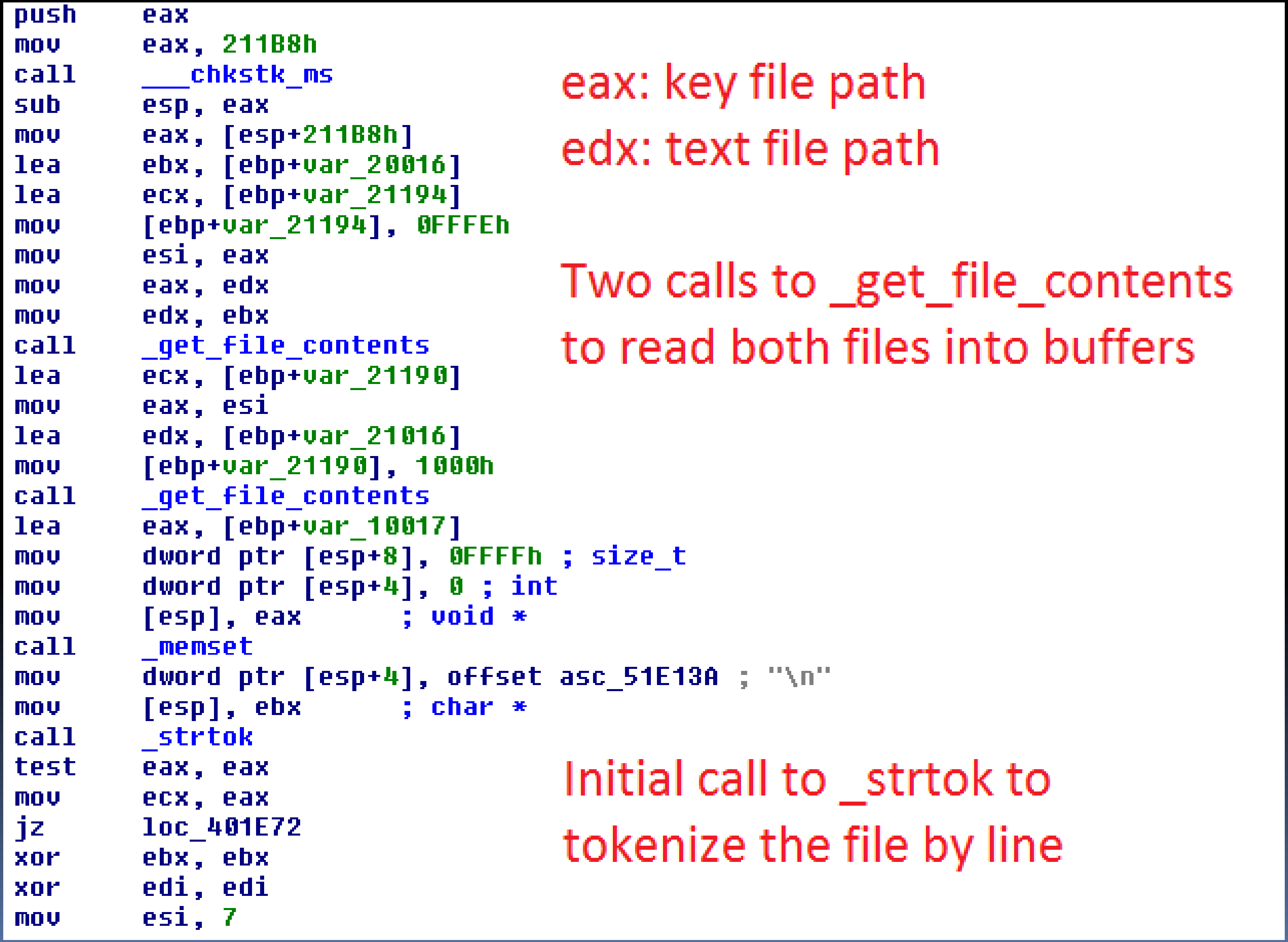
```
mov     edx, [esp+20h] ; key file path
mov     eax, [esp+1Ch] ; text file path
call    _tier2
```

Inside _tier2



```
push    eax
mov     eax, 211B8h
call    ___chkstk_ms
sub     esp, eax
mov     eax, [esp+211B8h]
lea     ebx, [ebp+var_20016]
lea     ecx, [ebp+var_21194]
mov     [ebp+var_21194], 0FFFEh
mov     esi, eax
mov     eax, edx
mov     edx, ebx
call    _get_file_contents
lea     ecx, [ebp+var_21190]
mov     eax, esi
lea     edx, [ebp+var_21016]
mov     [ebp+var_21190], 1000h
call    _get_file_contents
lea     eax, [ebp+var_10017]
mov     dword ptr [esp+8], 0FFFFh ; size_t
mov     dword ptr [esp+4], 0 ; int
mov     [esp], eax      ; void *
call    _memset
mov     dword ptr [esp+4], offset asc_51E13A ; "\n"
mov     [esp], ebx      ; char *
call    _strtok
test    eax, eax
mov     ecx, eax
jz      loc_401E72
xor     ebx, ebx
xor     edi, edi
mov     esi, 7
```

Inside _tier2 (2)



```
push    eax
mov     eax, 211B8h
call    ___chkstk_ms
sub     esp, eax
mov     eax, [esp+211B8h]
lea     ebx, [ebp+var_20016]
lea     ecx, [ebp+var_21194]
mov     [ebp+var_21194], 0FFFEh
mov     esi, eax
mov     eax, edx
mov     edx, ebx
call    _get_file_contents
lea     ecx, [ebp+var_21190]
mov     eax, esi
lea     edx, [ebp+var_21016]
mov     [ebp+var_21190], 1000h
call    _get_file_contents
lea     eax, [ebp+var_10017]
mov     dword ptr [esp+8], 0FFFFh ; size_t
mov     dword ptr [esp+4], 0 ; int
mov     [esp], eax      ; void *
call    _memset
mov     dword ptr [esp+4], offset asc_51E13A ; "\n"
mov     [esp], ebx      ; char *
call    _strtok
test    eax, eax
mov     ecx, eax
jz      loc_401E72
xor     ebx, ebx
xor     edi, edi
mov     esi, 7
```

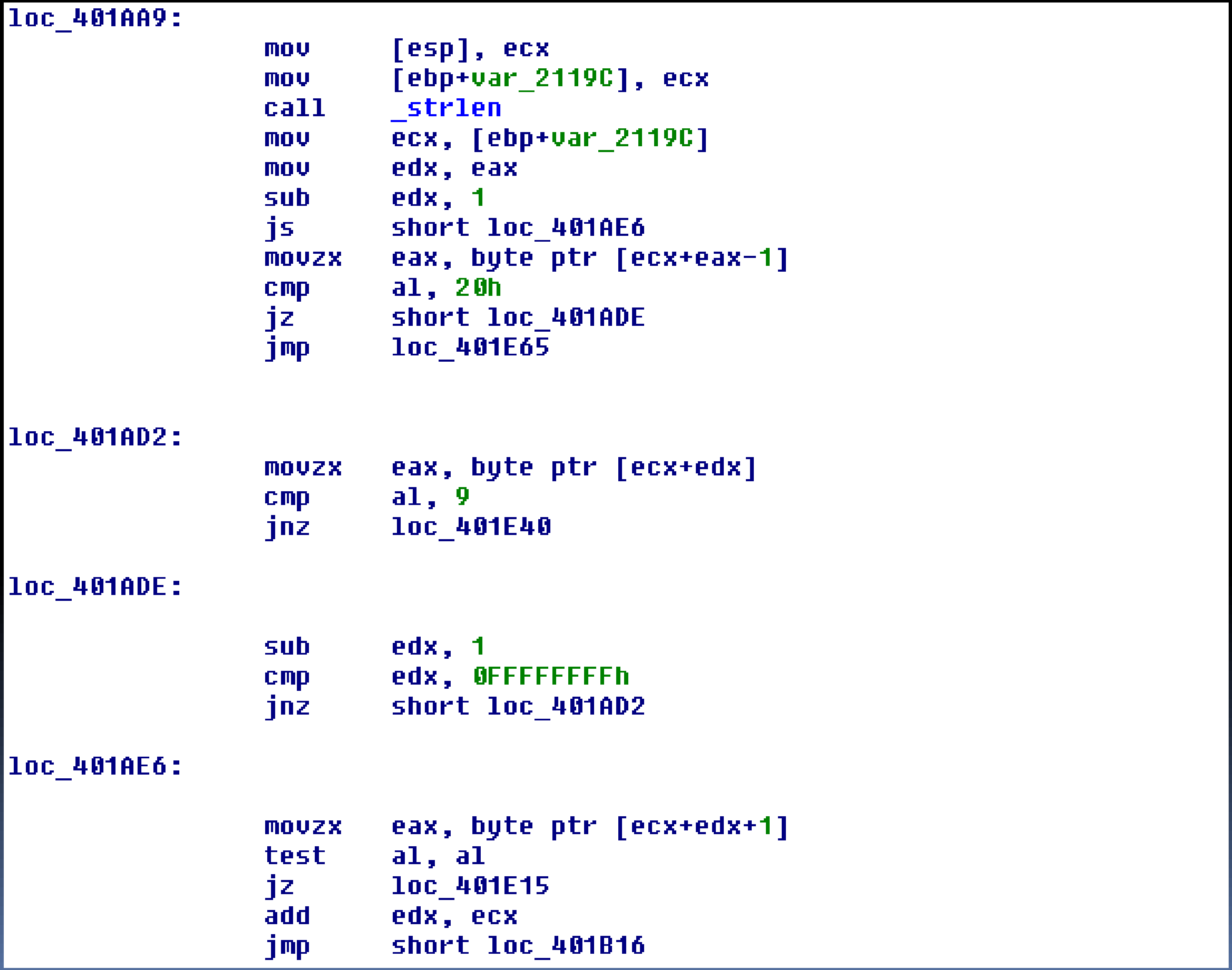
eax: key file path

edx: text file path

Two calls to _get_file_contents to read both files into buffers

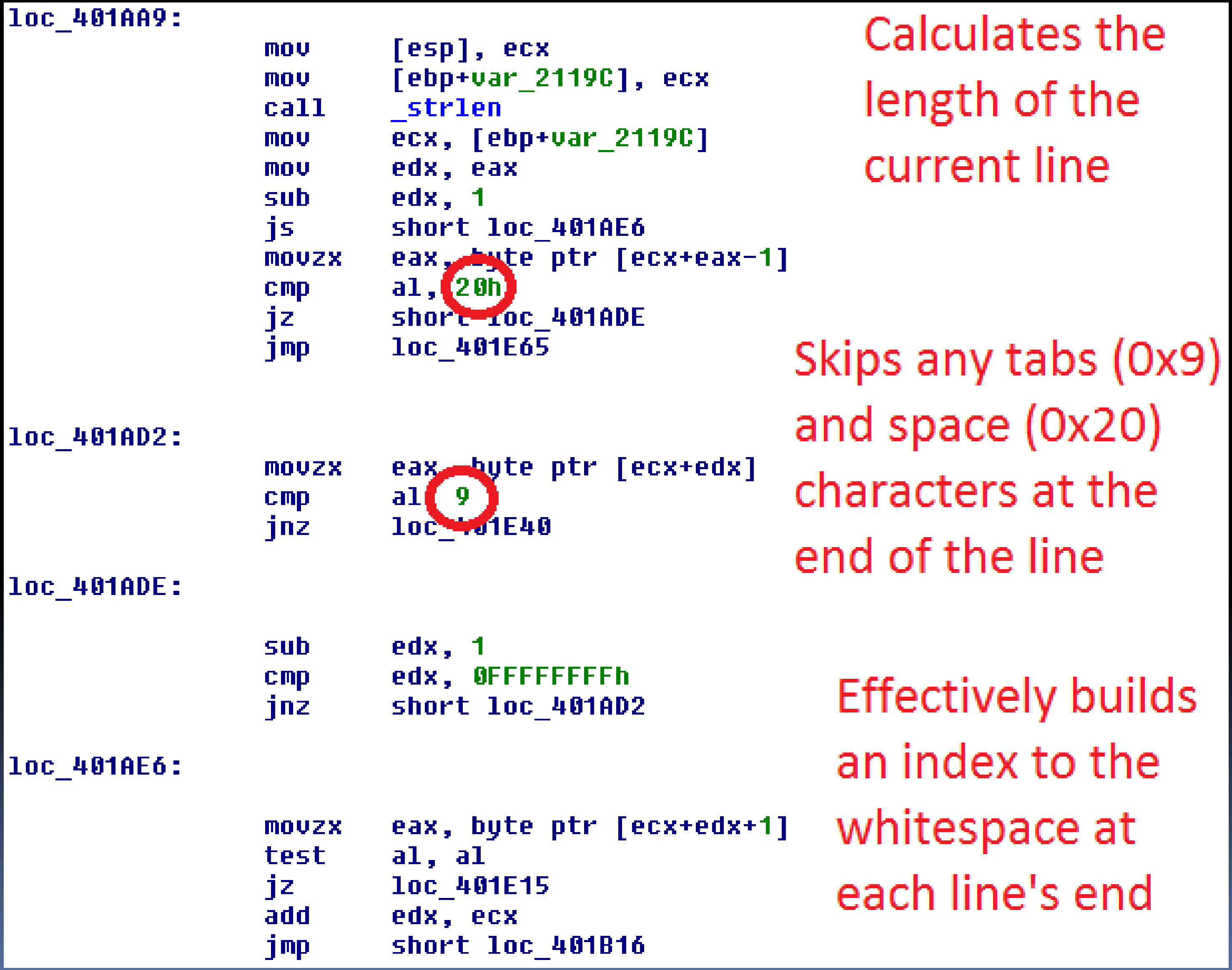
Initial call to _strtok to tokenize the file by line

Inside _tier2 (3)



```
loc_401AA9:
                mov     [esp], ecx
                mov     [ebp+var_2119C], ecx
                call    _strlen
                mov     ecx, [ebp+var_2119C]
                mov     edx, eax
                sub     edx, 1
                js      short loc_401AE6
                movzx   eax, byte ptr [ecx+eax-1]
                cmp     al, 20h
                jz      short loc_401ADE
                jmp     loc_401E65


loc_401AD2:
                movzx   eax, byte ptr [ecx+edx]
                cmp     al, 9
                jnz     loc_401E40

loc_401ADE:

                sub     edx, 1
                cmp     edx, 0FFFFFFFFh
                jnz     short loc_401AD2

loc_401AE6:

                movzx   eax, byte ptr [ecx+edx+1]
                test    al, al
                jz      loc_401E15
                add     edx, ecx
                jmp     short loc_401B16
```

Inside _tier2 (4)



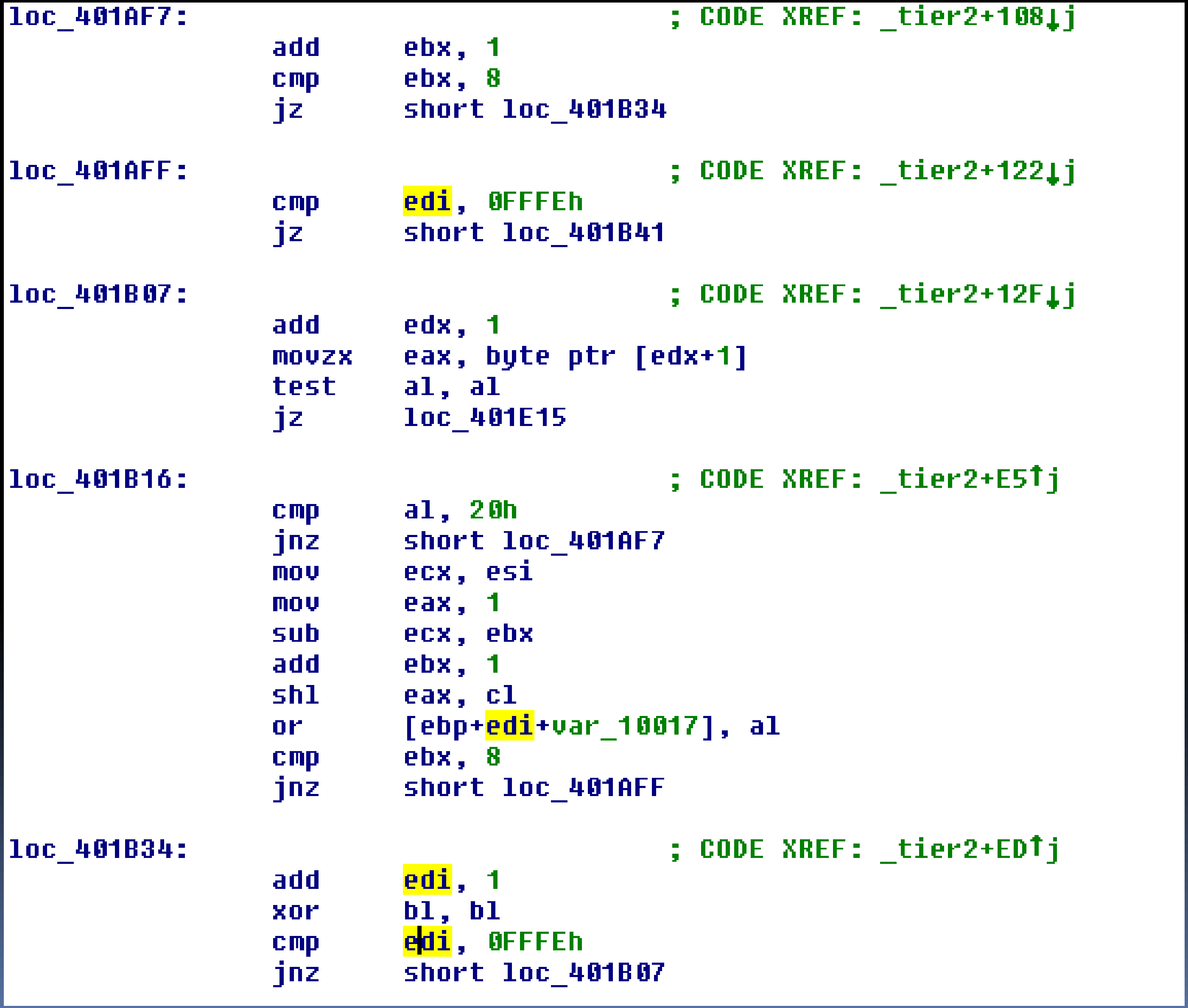
```
loc_401AA9:
                mov     [esp], ecx
                mov     [ebp+var_2119C], ecx
                call    _strlen
                mov     ecx, [ebp+var_2119C]
                mov     edx, eax
                sub     edx, 1
                js      short loc_401AE6
                movzx   eax, byte ptr [ecx+eax-1]
                cmp     al, 20h
                jz      short loc_401ADE
                jmp     loc_401E65

loc_401AD2:
                movzx   eax, byte ptr [ecx+edx]
                cmp     al, 9
                jnz     loc_401E40

loc_401ADE:

                sub     edx, 1
                cmp     edx, 0FFFFFFFFh
                jnz     short loc_401AD2

loc_401AE6:

                movzx   eax, byte ptr [ecx+edx+1]
                test    al, al
                jz      loc_401E15
                add     edx, ecx
                jmp     short loc_401B16
```

Calculates the length of the current line

Skips any tabs (0x9) and space (0x20) characters at the end of the line

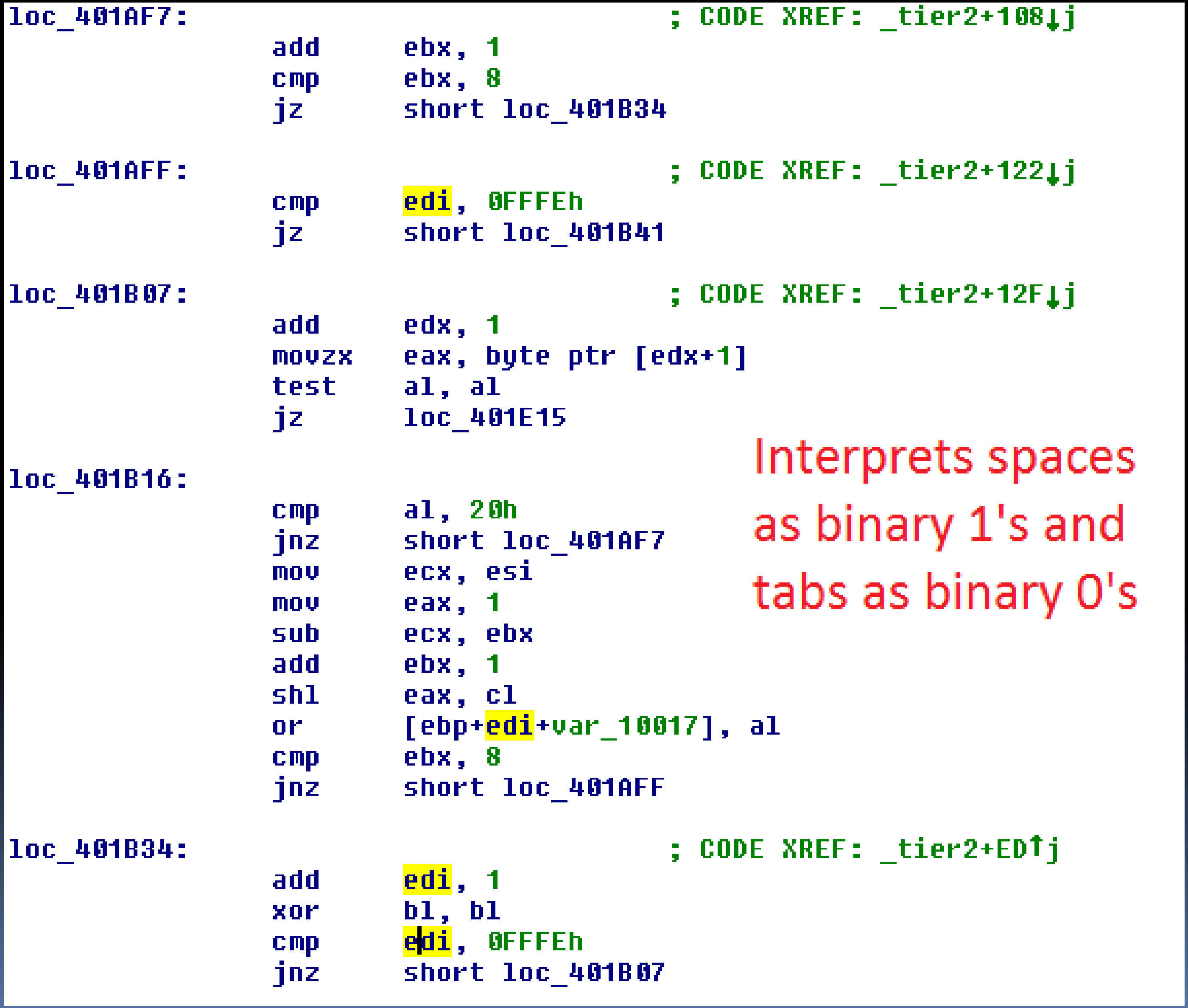
Effectively builds an index to the whitespace at each line's end

Inside _tier2 (5)



```
loc_401AF7:                                     ; CODE XREF: _tier2+108↓j
                add     ebx, 1
                cmp     ebx, 8
                jz      short loc_401B34

loc_401AFF:                                     ; CODE XREF: _tier2+122↓j
                cmp     edi, 0FFFEh
                jz      short loc_401B41

loc_401B07:                                     ; CODE XREF: _tier2+12F↓j
                add     edx, 1
                movzx   eax, byte ptr [edx+1]
                test    al, al
                jz      loc_401E15

loc_401B16:                                     ; CODE XREF: _tier2+E5↑j
                cmp     al, 20h
                jnz     short loc_401AF7
                mov     ecx, esi
                mov     eax, 1
                sub     ecx, ebx
                add     ebx, 1
                shl     eax, cl
                or      [ebp+edi+var_10017], al
                cmp     ebx, 8
                jnz     short loc_401AFF

loc_401B34:                                     ; CODE XREF: _tier2+ED↑j
                add     edi, 1
                xor     bl, bl
                cmp     edi, 0FFFEh
                jnz     short loc_401B07
```

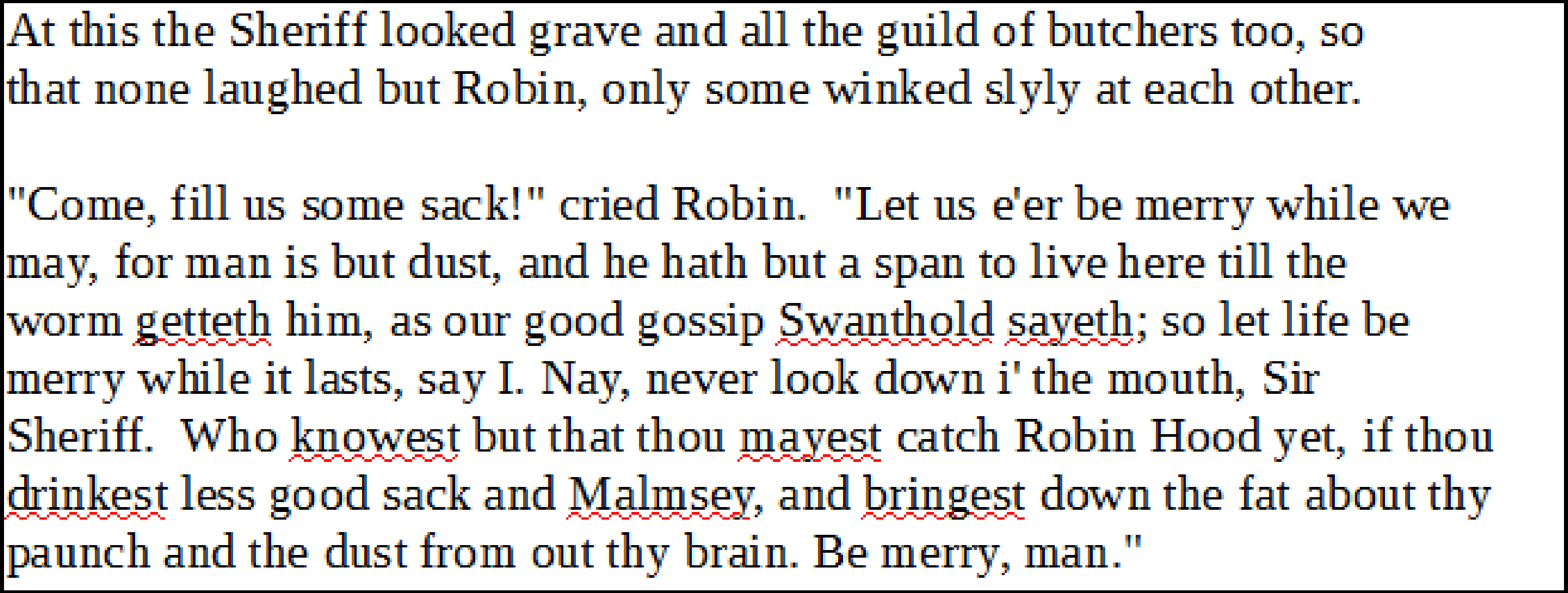
Inside _tier2 (6)



```
loc_401AF7:                             ; CODE XREF: _tier2+108↓j
                add     ebx, 1
                cmp     ebx, 8
                jz      short loc_401B34

loc_401AFF:                             ; CODE XREF: _tier2+122↓j
                cmp     edi, 0FFFEh
                jz      short loc_401B41

loc_401B07:                             ; CODE XREF: _tier2+12F↓j
                add     edx, 1
                movzx   eax, byte ptr [edx+1]
                test    al, al
                jz      loc_401E15

loc_401B16:
                cmp     al, 20h
                jnz     short loc_401AF7
                mov     ecx, esi
                mov     eax, 1
                sub     ecx, ebx
                add     ebx, 1
                shl     eax, cl
                or      [ebp+edi+var_10017], al
                cmp     ebx, 8
                jnz     short loc_401AFF

loc_401B34:                             ; CODE XREF: _tier2+ED↑j
                add     edi, 1
                xor     bl, bl
                cmp     edi, 0FFFEh
                jnz     short loc_401B07
```

Interprets spaces as binary 1's and tabs as binary 0's

Tabs and Spaces

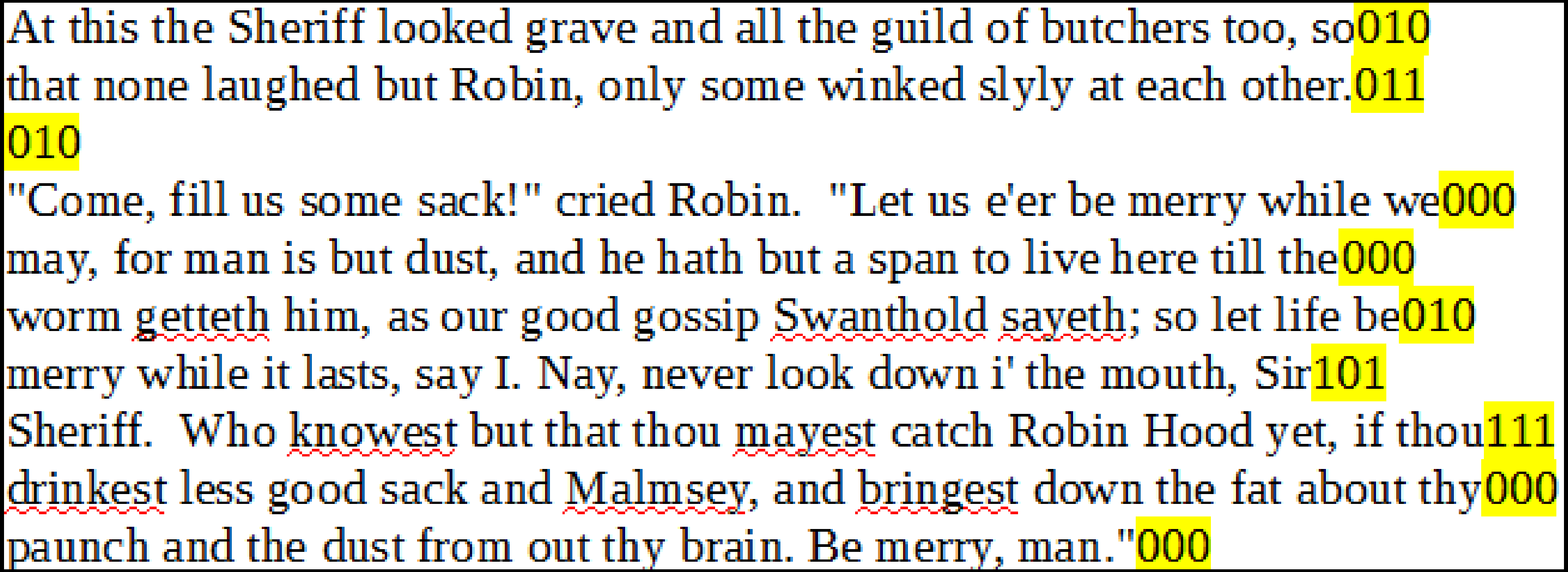


At this the Sheriff looked grave and all the guild of butchers too, so
that none laughed but Robin, only some winked slyly at each other.

"Come, fill us some sack!" cried Robin. "Let us e'er be merry while we
may, for man is but dust, and he hath but a span to live here till the
worm getteth him, as our good gossip Swanthold sayeth; so let life be
merry while it lasts, say I. Nay, never look down i' the mouth, Sir
Sheriff. Who knowest but that thou mayest catch Robin Hood yet, if thou
drinkest less good sack and Malmsey, and bringest down the fat about thy
paunch and the dust from out thy brain. Be merry, man."

...

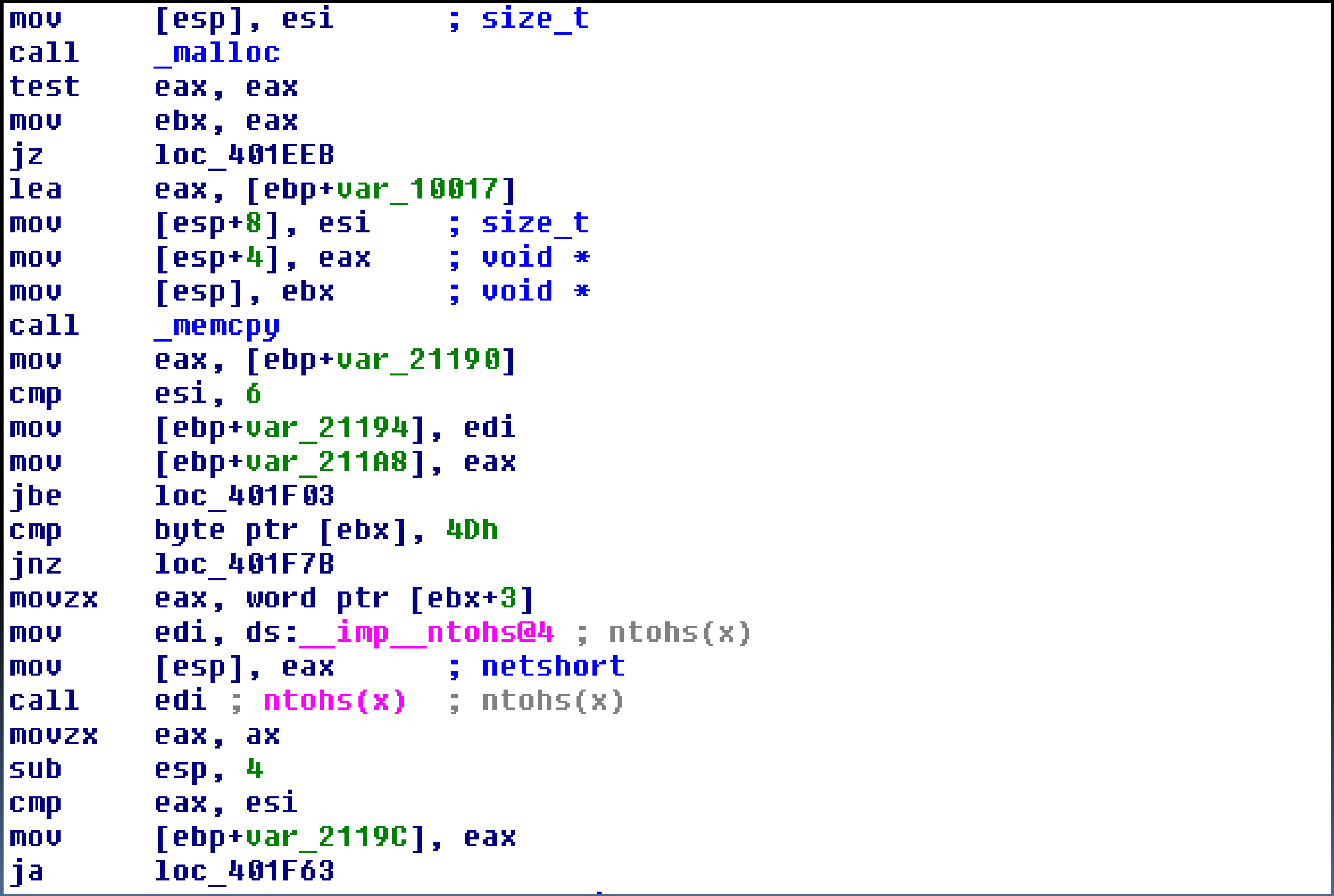
Tabs and Spaces – Revealed!



At this the Sheriff looked grave and all the guild of butchers too, so010
that none laughed but Robin, only some winked slyly at each other.011
010
"Come, fill us some sack!" cried Robin. "Let us e'er be merry while we000
may, for man is but dust, and he hath but a span to live here till the000
worm getteth him, as our good gossip Swanthold sayeth; so let life be010
merry while it lasts, say I. Nay, never look down i' the mouth, Sir101
Sheriff. Who knowest but that thou mayest catch Robin Hood yet, if thou111
drinkest less good sack and Malmsey, and bringest down the fat about thy000
paunch and the dust from out thy brain. Be merry, man."000

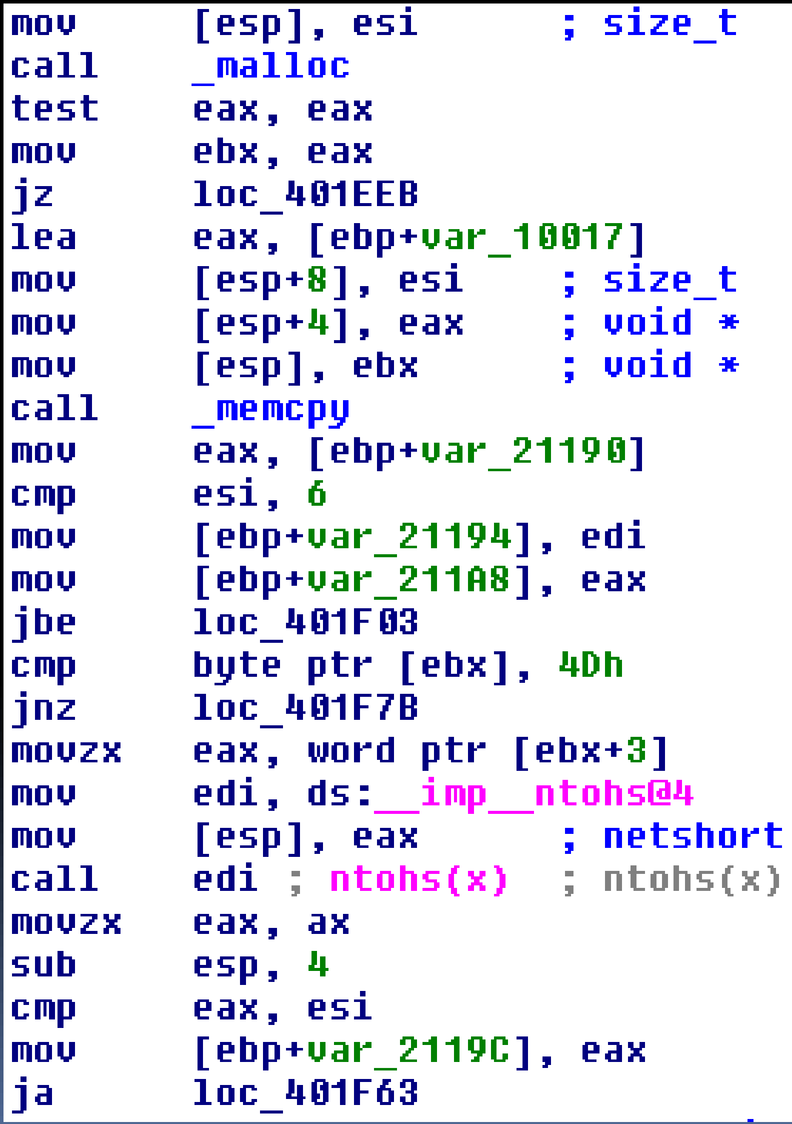
- …

Inside _tier2 (7)



```
mov     [esp], esi      ; size_t
call    _malloc
test    eax, eax
mov     ebx, eax
jz      loc_401EEB
lea     eax, [ebp+var_10017]
mov     [esp+8], esi    ; size_t
mov     [esp+4], eax    ; void *
mov     [esp], ebx      ; void *
call    _memcpy
mov     eax, [ebp+var_21190]
cmp     esi, 6
mov     [ebp+var_21194], edi
mov     [ebp+var_211A8], eax
jbe     loc_401F03
cmp     byte ptr [ebx], 4Dh
jnz     loc_401F7B
movzx   eax, word ptr [ebx+3]
mov     edi, ds:__imp__ntohs@4 ; ntohs(x)
mov     [esp], eax      ; netshort
call    edi ; ntohs(x)  ; ntohs(x)
movzx   eax, ax
sub     esp, 4
cmp     eax, esi
mov     [ebp+var_2119C], eax
ja      loc_401F63
```

Inside _tier2 (8)



```
mov     [esp], esi      ; size_t
call    _malloc
test    eax, eax
mov     ebx, eax
jz      loc_401EEB
lea     eax, [ebp+var_10017]
mov     [esp+8], esi    ; size_t
mov     [esp+4], eax    ; void *
mov     [esp], ebx      ; void *
call    _memcpy
mov     eax, [ebp+var_21190]
cmp     esi, 6
mov     [ebp+var_21194], edi
mov     [ebp+var_211A8], eax
jbe     loc_401F03
cmp     byte ptr [ebx], 4Dh
jnz     loc_401F7B
movzx   eax, word ptr [ebx+3]
mov     edi, ds:__imp__ntohs@4
mov     [esp], eax      ; netshort
call    edi ; ntohs(x)  ; ntohs(x)
movzx   eax, ax
sub     esp, 4
cmp     eax, esi
mov     [ebp+var_2119C], eax
ja      loc_401F63
```

Allocates dynamic space for the decoded data and copies it in

esi: size of decoded data

ebx: decoded data (heap)

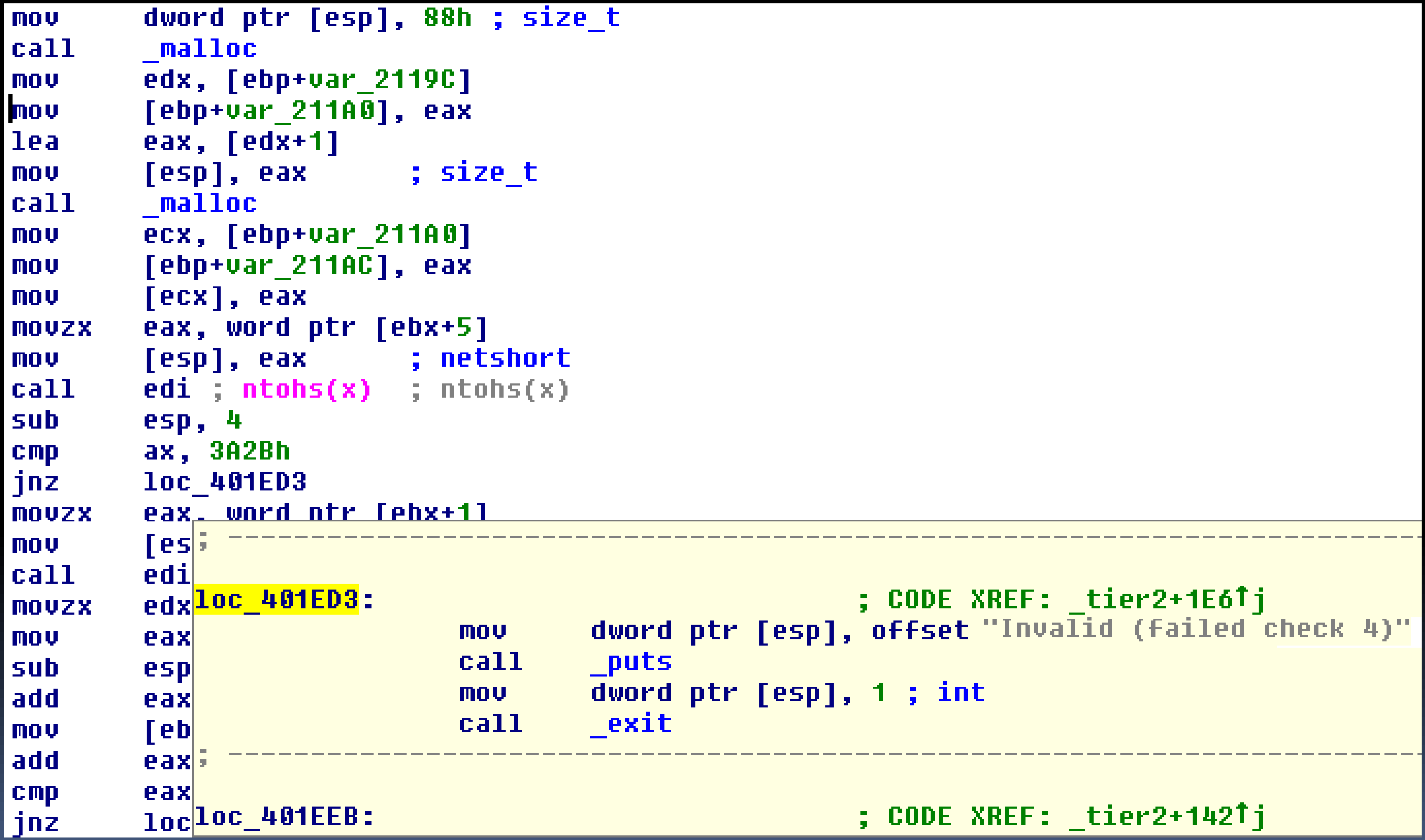
Three compares:

1. data size > 6
2. data[0] == 'M'
3. data[3-4] <= data size

Inside _tier2 - Fail cases

```
loc_401FO3:
mov     dword ptr [esp], offset "Invalid (failed check 1)"
call    _puts
mov     dword ptr [esp], 1
call    _exit

loc_401F7B:
mov     dword ptr [esp], offset "Invalid (failed check 2)"
call    _puts
mov     dword ptr [esp], 1
call    _exit

loc_401F63:
mov     dword ptr [esp], offset "Invalid (failed check 3)"
call    _puts
mov     dword ptr [esp], 1
call    _exit
```

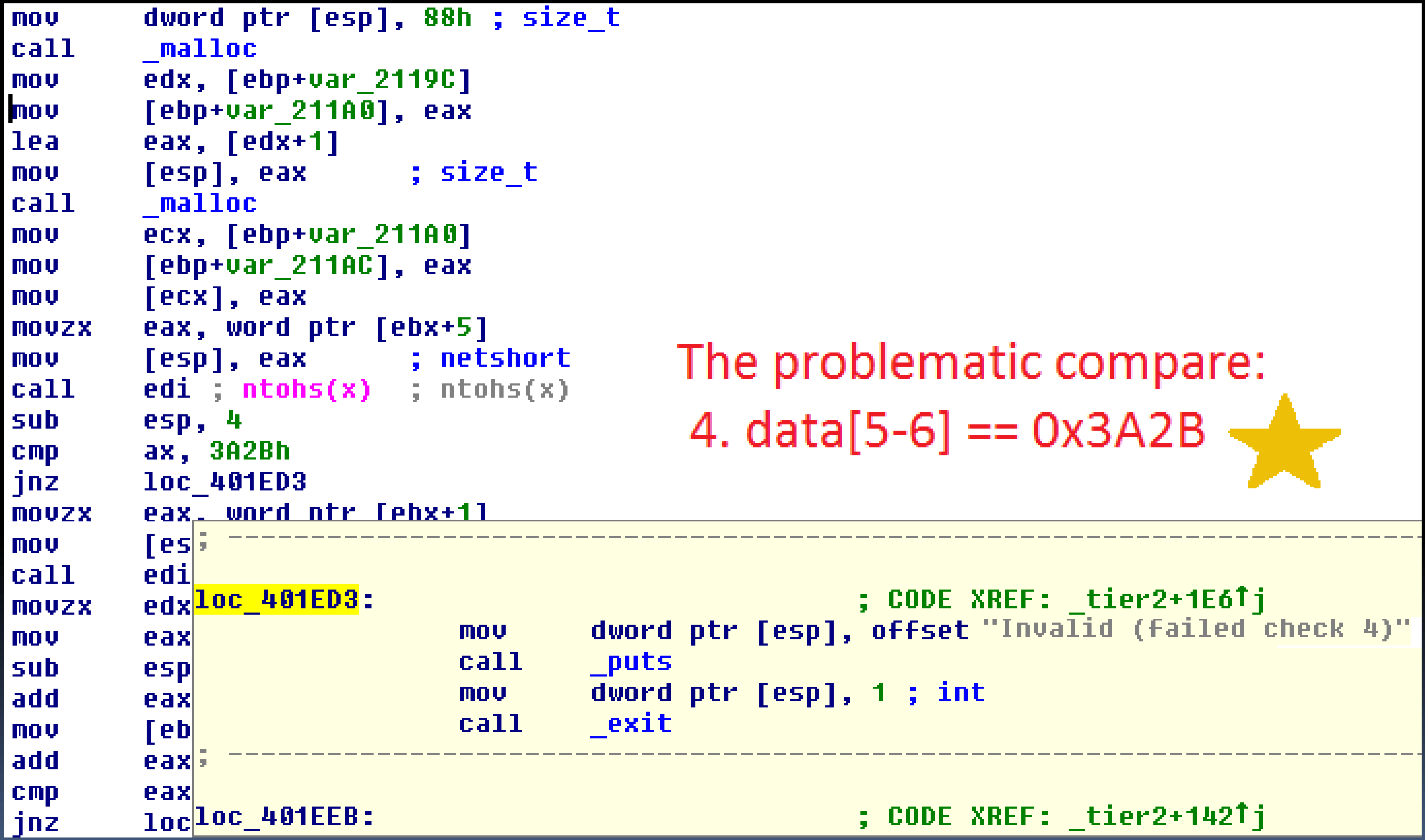
Inside _tier2 (9)



```
mov     dword ptr [esp], 88h ; size_t
call    _malloc
mov     edx, [ebp+var_2119C]
mov     [ebp+var_211A0], eax
lea     eax, [edx+1]
mov     [esp], eax      ; size_t
call    _malloc
mov     ecx, [ebp+var_211A0]
mov     [ebp+var_211AC], eax
mov     [ecx], eax
movzx   eax, word ptr [ebx+5]
mov     [esp], eax      ; netshort
call    edi ; ntohs(x)  ; ntohs(x)
sub     esp, 4
cmp     ax, 3A2Bh
jnz     loc_401ED3
movzx   eax, word ptr [ebx+1]
mov     [es
call    edi
movzx   edx
mov     eax
sub     esp
add     eax
mov     [eb
add     eax
cmp     eax
jnz     loc
```

```
; ---------------------------------------------------------------------------

loc_401ED3:                             ; CODE XREF: _tier2+1E6↑j
                mov     dword ptr [esp], offset "Invalid (failed check 4)"
                call    _puts
                mov     dword ptr [esp], 1 ; int
                call    _exit
; ---------------------------------------------------------------------------

loc_401EEB:                             ; CODE XREF: _tier2+142↑j
```

Inside _tier2 (10)



```
mov     dword ptr [esp], 88h ; size_t
call    _malloc
mov     edx, [ebp+var_2119C]
mov     [ebp+var_211A0], eax
lea     eax, [edx+1]
mov     [esp], eax      ; size_t
call    _malloc
mov     ecx, [ebp+var_211A0]
mov     [ebp+var_211AC], eax
mov     [ecx], eax
movzx   eax, word ptr [ebx+5]
mov     [esp], eax      ; netshort
call    edi ; ntohs(x)  ; ntohs(x)
sub     esp, 4
cmp     ax, 3A2Bh
jnz     loc_401ED3
movzx   eax, word ptr [ebx+1]
mov     [es
call    edi
movzx   edx
mov     eax
sub     esp
add     eax
mov     [eb
add     eax
cmp     eax
jnz     loc
```

```
; ---------------------------------------------------------------------------

loc_401ED3:                             ; CODE XREF: _tier2+1E6↑j
                mov     dword ptr [esp], offset "Invalid (failed check 4)"
                call    _puts
                mov     dword ptr [esp], 1 ; int
                call    _exit
; ---------------------------------------------------------------------------

loc_401EEB:                             ; CODE XREF: _tier2+142↑j
```

The problematic compare:

4. data[5-6] == 0x3A2B

From the Task 2 backstory

- “Through SIGINT we have collected a new message file - this one appears to have been sent to a field operative”

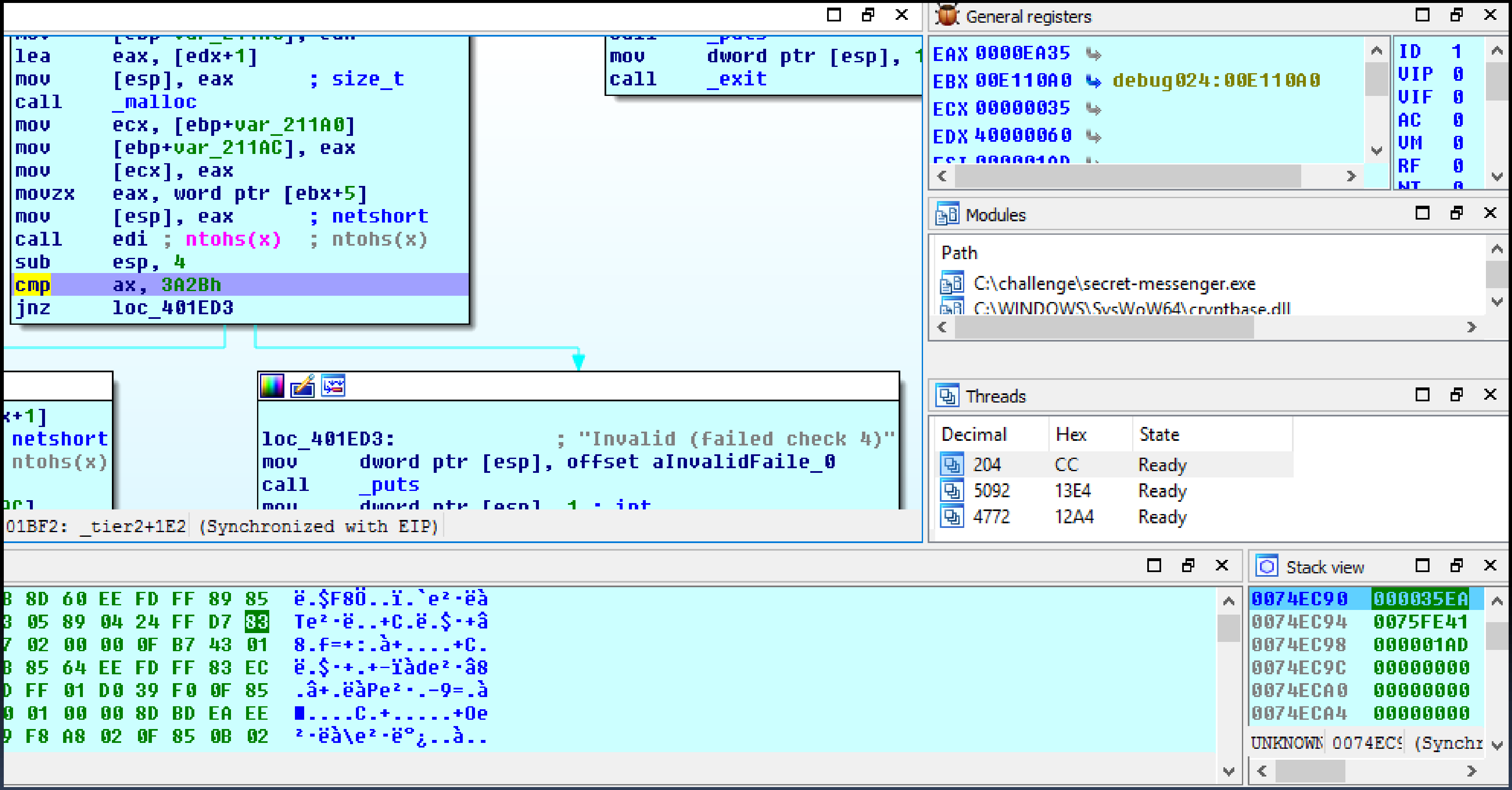
- The first message didn’t have this problem…

- Messages must have an ID associating them to a given operative.

We have the binary...

- So bypass the check dynamically!
- Set a breakpoint at the comparison in IDA
  - Click the circle to the left of that line of code
- Prepare the debugger
  - Debugger -> Set Debugger (Local Win32 debugger)
  - Debugger -> Process Options...
  - Specify the program parameters for Task 2 from earlier
  - Start Process...

At the breakpoint



```
lea     eax, [edx+1]
mov     [esp], eax       ; size_t
call    _malloc
mov     ecx, [ebp+var_211A0]
mov     [ebp+var_211AC], eax
mov     [ecx], eax
movzx   eax, word ptr [ebx+5]
mov     [esp], eax       ; netshort
call    edi ; ntohs(x)   ; ntohs(x)
sub     esp, 4
cmp     ax, 3A2Bh
jnz     loc_401ED3
```

```
mov     dword ptr [esp], 1
call    _exit
```

```
loc_401ED3:                 ; "Invalid (failed check 4)"
mov     dword ptr [esp], offset aInvalidFaile_0
call    _puts
```

```
01BF2: _tier2+1E2 (Synchronized with EIP)
```

```
B 8D 60 EE FD FF 89 85  ë.$F8Ö..ï.`e²·ëà
3 05 89 04 24 FF D7 83  Te²·ë..+C.ë.$·+â
7 02 00 00 0F B7 43 01  8.f=+:.à+....+C.
B 85 64 EE FD FF 83 EC  ë.$·+.+–ïàde²·â8
D FF 01 D0 39 F0 0F 85  .â+.ëàPe²·.–9=.à
0 01 00 00 8D BD EA EE  █....C.+.....+0e
9 F8 A8 02 0F 85 0B 02  ²·ëà\e²·ë°¿..à..
```

General registers

```
EAX 0000EA35
EBX 00E110A0  debug024:00E110A0
ECX 00000035
EDX 40000060
```

```
ID  1
VIP 0
VIF 0
AC  0
VM  0
RF  0
```

Modules

Path
C:\challenge\secret-messenger.exe
C:\WINDOWS\SysWoW64\cryptbase.dll

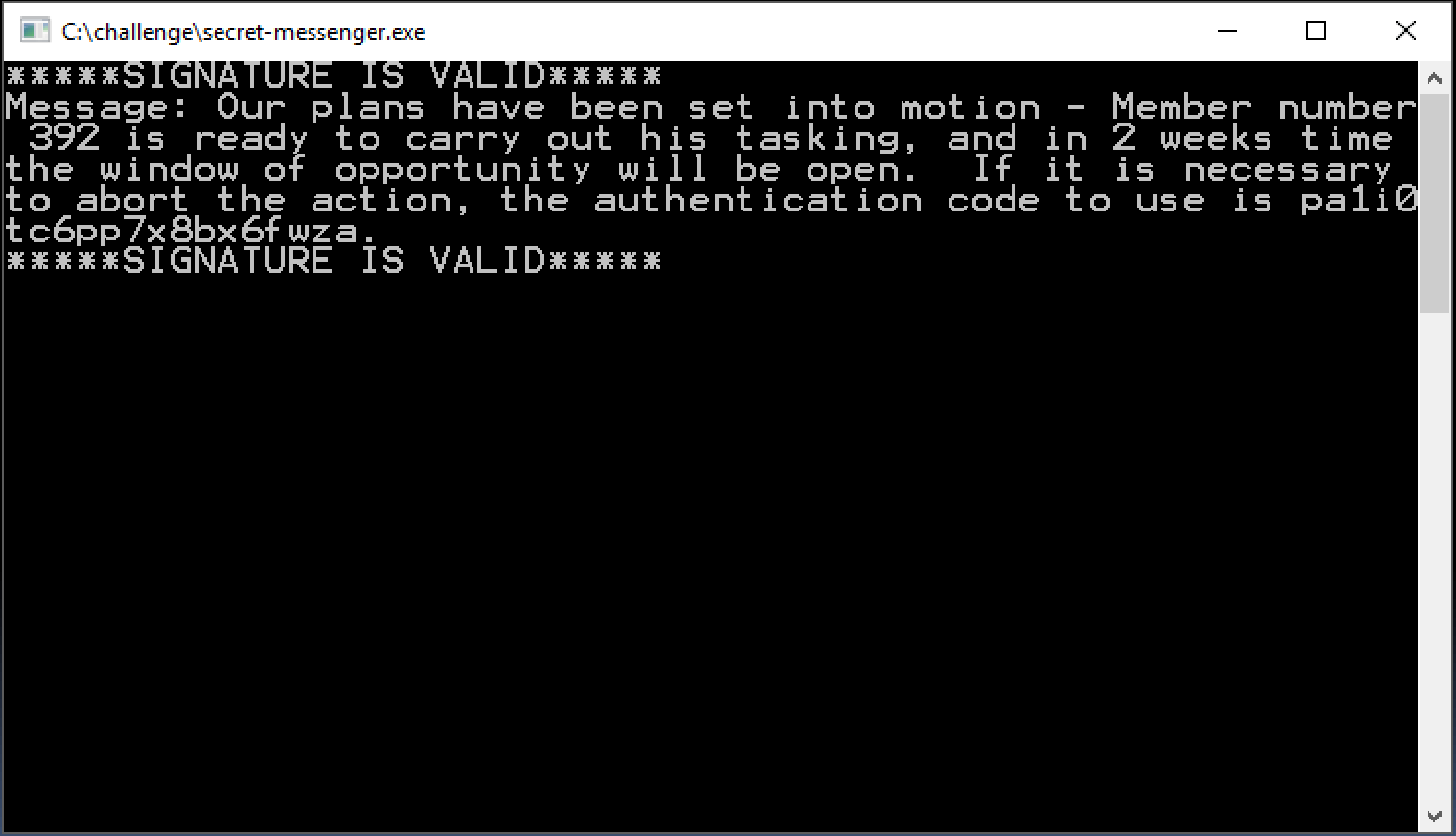
Threads

Decimal	Hex	State
204	CC	Ready
5092	13E4	Ready
4772	12A4	Ready

Stack view

```
0074EC90  000035EA
0074EC94  0075FE41
0074EC98  000001AD
0074EC9C  00000000
0074ECA0  00000000
0074ECA4  00000000
UNKNOWN 0074EC9 (Synchr
```

The result:



```
*****SIGNATURE IS VALID*****
Message: Our plans have been set into motion - Member number
 392 is ready to carry out his tasking, and in 2 weeks time
the window of opportunity will be open.  If it is necessary
to abort the action, the authentication code to use is pa1i0
tc6pp7x8bx6fwza.
*****SIGNATURE IS VALID*****
```

Task 2 Complete!

- Required either bypassing the check as we demonstrated, or modifying the binary / message

- … on to Task 3!

2015 Challenge – Task 3

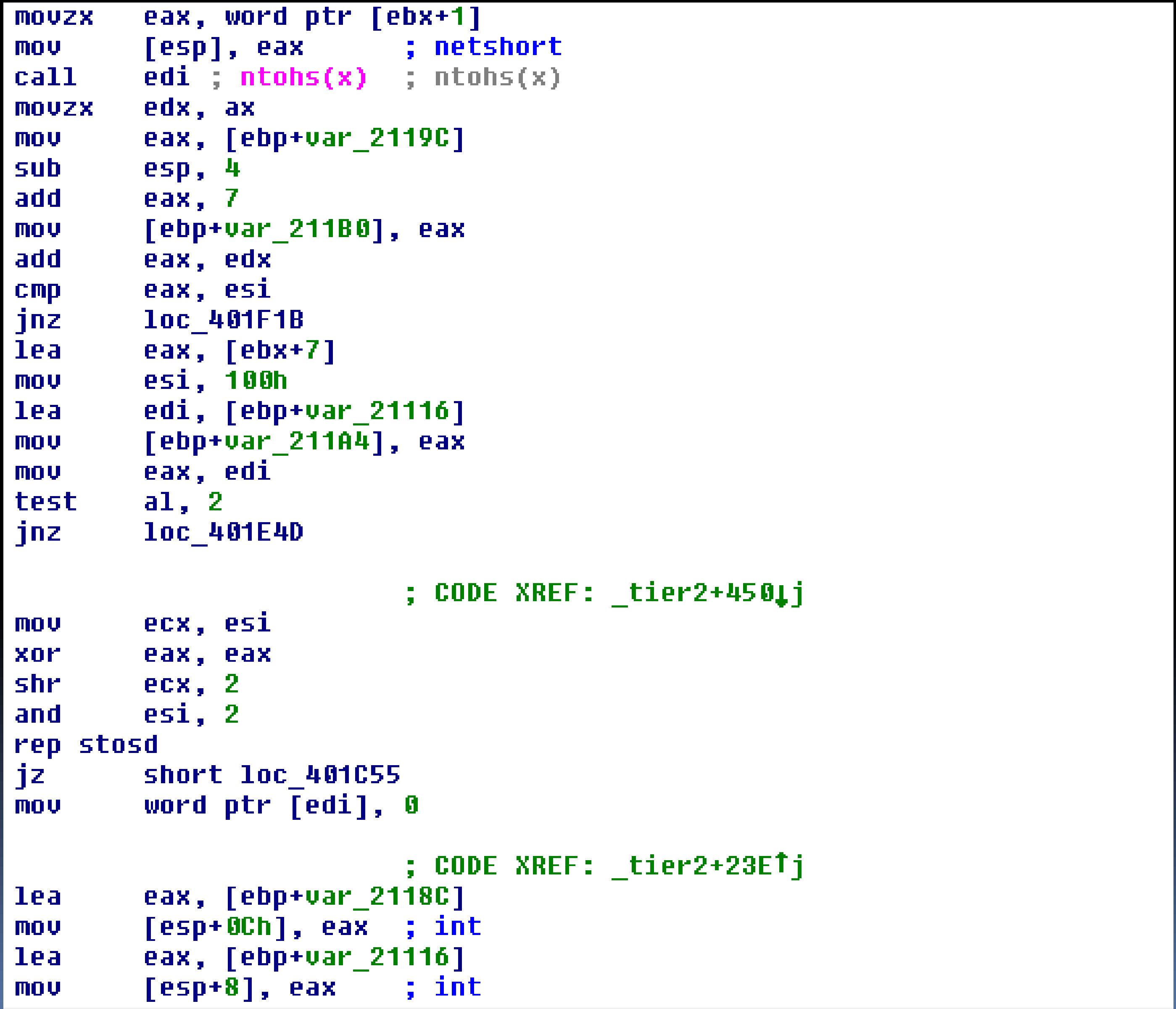
- The copy of the program you have is only capable of decoding secret messages and lacks the ability to encode new messages to other operatives. We need this capability in order to infiltrate the terrorist network and send encoded messages…
- Provided:
  - A message to encode
  - A text file to encode the message into
  - A public/private key pair

Recap – What we know so far

- Messages are encoded using tabs and spaces
- Once decoded, they have certain properties:
  - data[] size > 6
  - data[0] == 'M'
  - data[3-4] <= data size
  - data[5-6] == 0x3A2B

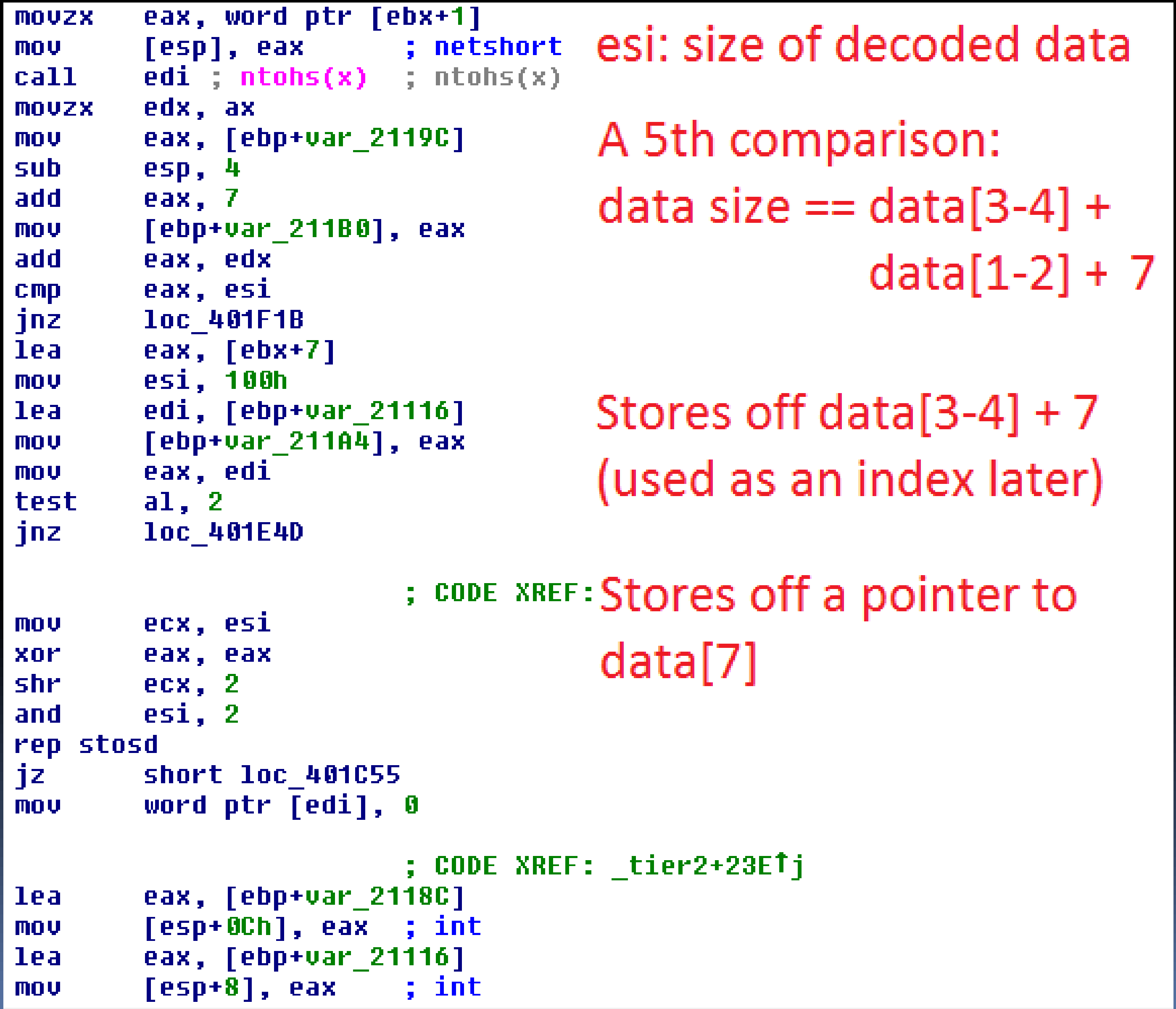
- So, message must take the form:
  - `‘M’ | ???? ???? | length ? | 0x3A 0X2B | ????????`

Inside _tier2 (11)



```
movzx   eax, word ptr [ebx+1]
mov     [esp], eax        ; netshort
call    edi ; ntohs(x)    ; ntohs(x)
movzx   edx, ax
mov     eax, [ebp+var_2119C]
sub     esp, 4
add     eax, 7
mov     [ebp+var_211B0], eax
add     eax, edx
cmp     eax, esi
jnz     loc_401F1B
lea     eax, [ebx+7]
mov     esi, 100h
lea     edi, [ebp+var_21116]
mov     [ebp+var_211A4], eax
mov     eax, edi
test    al, 2
jnz     loc_401E4D

                          ; CODE XREF: _tier2+450↓j
mov     ecx, esi
xor     eax, eax
shr     ecx, 2
and     esi, 2
rep stosd
jz      short loc_401C55
mov     word ptr [edi], 0

                          ; CODE XREF: _tier2+23E↑j
lea     eax, [ebp+var_2118C]
mov     [esp+0Ch], eax    ; int
lea     eax, [ebp+var_21116]
mov     [esp+8], eax      ; int
```

Inside _tier2 (12)



```
movzx   eax, word ptr [ebx+1]
mov     [esp], eax        ; netshort
call    edi ; ntohs(x)    ; ntohs(x)
movzx   edx, ax
mov     eax, [ebp+var_2119C]
sub     esp, 4
add     eax, 7
mov     [ebp+var_211B0], eax
add     eax, edx
cmp     eax, esi
jnz     loc_401F1B
lea     eax, [ebx+7]
mov     esi, 100h
lea     edi, [ebp+var_21116]
mov     [ebp+var_211A4], eax
mov     eax, edi
test    al, 2
jnz     loc_401E4D

                          ; CODE XREF:
mov     ecx, esi
xor     eax, eax
shr     ecx, 2
and     esi, 2
rep stosd
jz      short loc_401C55
mov     word ptr [edi], 0

                          ; CODE XREF: _tier2+23E↑j
lea     eax, [ebp+var_2118C]
mov     [esp+0Ch], eax    ; int
lea     eax, [ebp+var_21116]
mov     [esp+8], eax      ; int
```

esi: size of decoded data

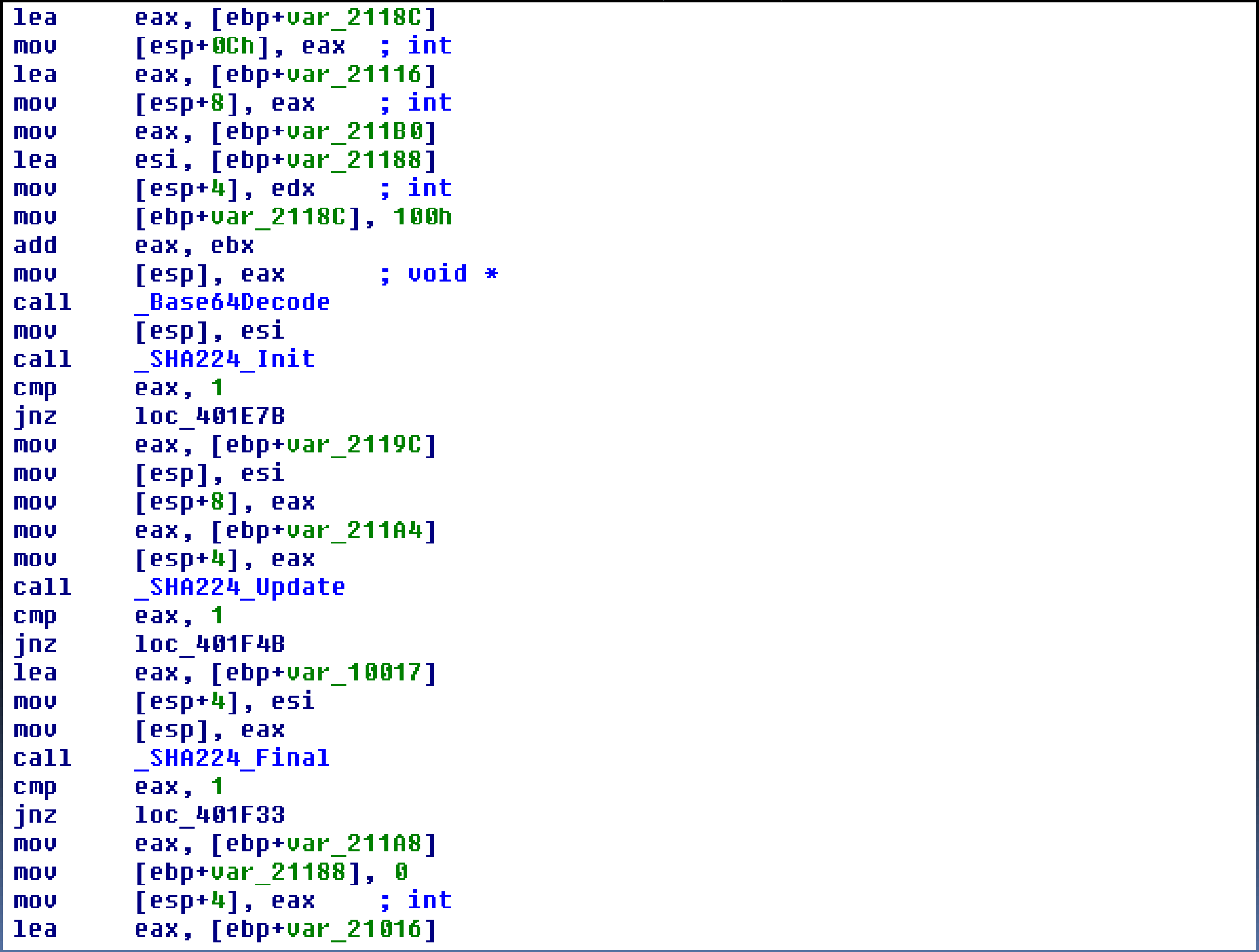
A 5th comparison:

data size == data[3-4] +
data[1-2] + 7

Stores off data[3-4] + 7
(used as an index later)

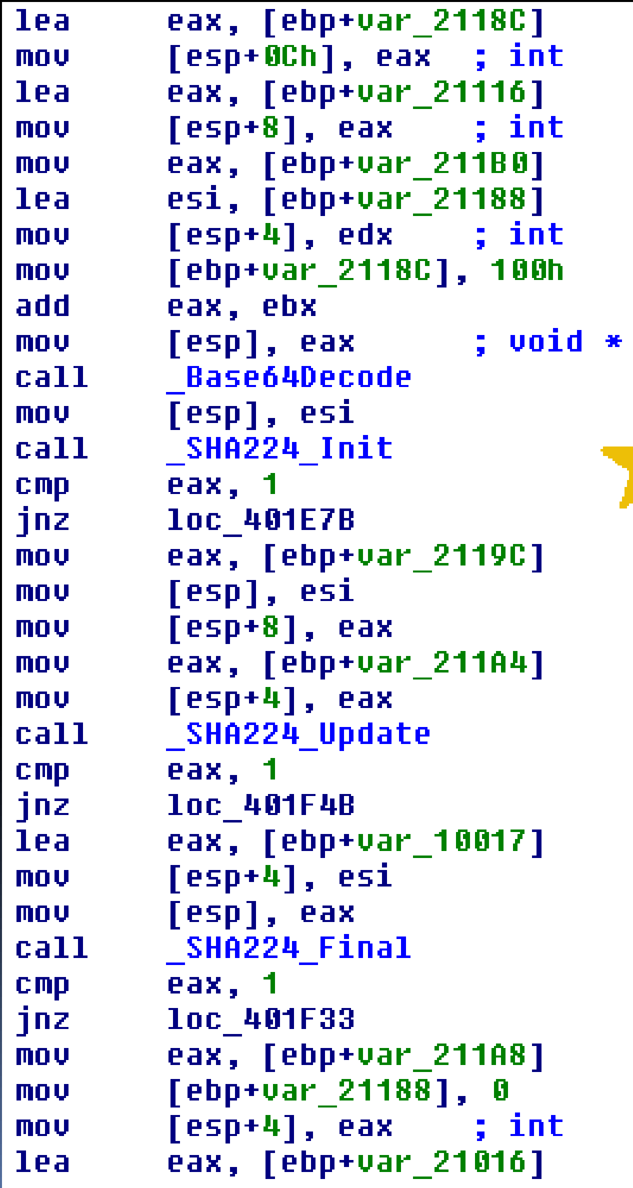
Stores off a pointer to
data[7]

Inside _tier2 (13)



```
lea     eax, [ebp+var_2118C]
mov     [esp+0Ch], eax  ; int
lea     eax, [ebp+var_21116]
mov     [esp+8], eax    ; int
mov     eax, [ebp+var_211B0]
lea     esi, [ebp+var_21188]
mov     [esp+4], edx    ; int
mov     [ebp+var_2118C], 100h
add     eax, ebx
mov     [esp], eax      ; void *
call    _Base64Decode
mov     [esp], esi
call    _SHA224_Init
cmp     eax, 1
jnz     loc_401E7B
mov     eax, [ebp+var_2119C]
mov     [esp], esi
mov     [esp+8], eax
mov     eax, [ebp+var_211A4]
mov     [esp+4], eax
call    _SHA224_Update
cmp     eax, 1
jnz     loc_401F4B
lea     eax, [ebp+var_10017]
mov     [esp+4], esi
mov     [esp], eax
call    _SHA224_Final
cmp     eax, 1
jnz     loc_401F33
mov     eax, [ebp+var_211A8]
mov     [ebp+var_21188], 0
mov     [esp+4], eax    ; int
lea     eax, [ebp+var_21016]
```

Inside _tier2 (14)



```
lea     eax, [ebp+var_2118C]
mov     [esp+0Ch], eax  ; int
lea     eax, [ebp+var_21116]
mov     [esp+8], eax    ; int
mov     eax, [ebp+var_211B0]
lea     esi, [ebp+var_21188]
mov     [esp+4], edx    ; int
mov     [ebp+var_2118C], 100h
add     eax, ebx
mov     [esp], eax      ; void *
call    _Base64Decode
mov     [esp], esi
call    _SHA224_Init
cmp     eax, 1
jnz     loc_401E7B
mov     eax, [ebp+var_2119C]
mov     [esp], esi
mov     [esp+8], eax
mov     eax, [ebp+var_211A4]
mov     [esp+4], eax
call    _SHA224_Update
cmp     eax, 1
jnz     loc_401F4B
lea     eax, [ebp+var_10017]
mov     [esp+4], esi
mov     [esp], eax
call    _SHA224_Final
cmp     eax, 1
jnz     loc_401F33
mov     eax, [ebp+var_211A8]
mov     [ebp+var_21188], 0
mov     [esp+4], eax    ; int
lea     eax, [ebp+var_21016]
```

Computes a pointer to data[(data[3-4] + 7)] and passes it to _Base64Decode

Calls to the SHA224 standard hashing functions

Passes data[7] pointer as 2nd arg to _Sha224_Update

Passes data[3-4] as 3rd arg to _Sha224_Update

SHA224_Update

```
SHA224_Update(
    SHA224_CTX *context,
    const uint8_t *data,
    size_t len);
```

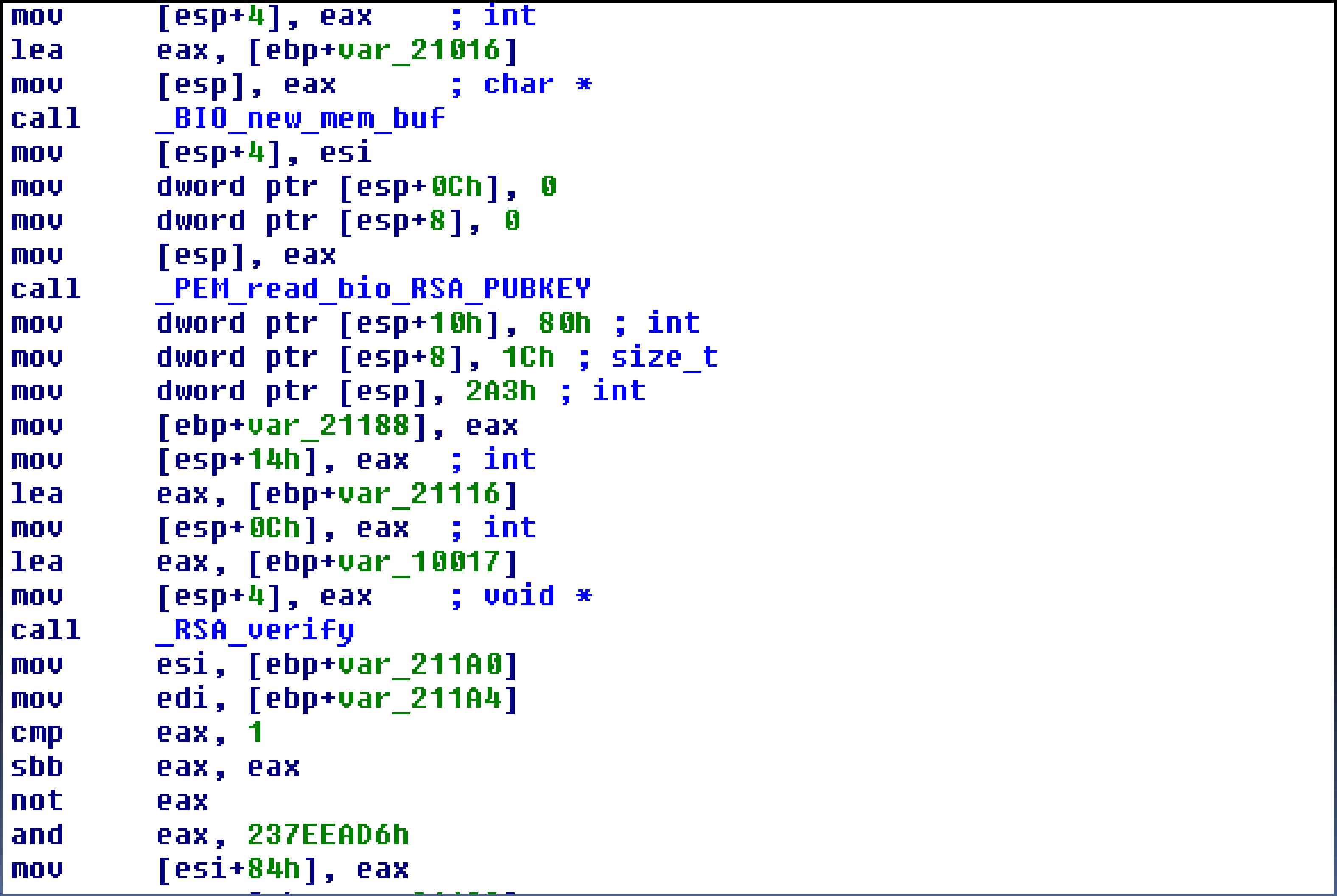
```
SHA224_Update(
    CTX_obj,
    pointer to data[7],
    data[3-4]);
```

Piecing together the clues

- data[3-4] length of data starting at data[7] (that gets hashed)
- data[1-2] length of the remaining data
  - data size == data[3-4] + data[1-2] + 7
- data after data1 is base64 decoded

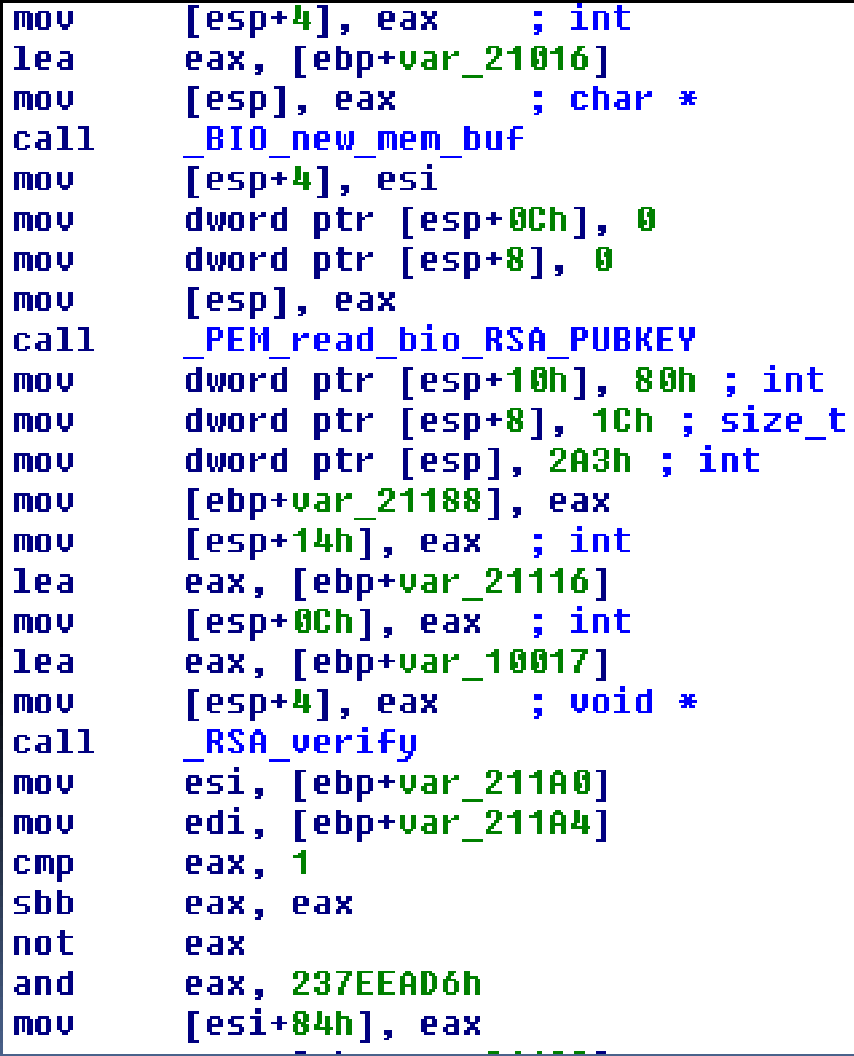
```
'M' | len data2 | len data1 | 0x3A2B | data1 | data2
data1 = ???
data2 = b64( ??? )
```

Inside _tier2 (15)



```
mov     [esp+4], eax    ; int
lea     eax, [ebp+var_21016]
mov     [esp], eax      ; char *
call    _BIO_new_mem_buf
mov     [esp+4], esi
mov     dword ptr [esp+0Ch], 0
mov     dword ptr [esp+8], 0
mov     [esp], eax
call    _PEM_read_bio_RSA_PUBKEY
mov     dword ptr [esp+10h], 80h ; int
mov     dword ptr [esp+8], 1Ch ; size_t
mov     dword ptr [esp], 2A3h ; int
mov     [ebp+var_21188], eax
mov     [esp+14h], eax  ; int
lea     eax, [ebp+var_21116]
mov     [esp+0Ch], eax  ; int
lea     eax, [ebp+var_10017]
mov     [esp+4], eax    ; void *
call    _RSA_verify
mov     esi, [ebp+var_211A0]
mov     edi, [ebp+var_211A4]
cmp     eax, 1
sbb     eax, eax
not     eax
and     eax, 237EEAD6h
mov     [esi+84h], eax
```

Inside _tier2 (16)



```
mov     [esp+4], eax    ; int
lea     eax, [ebp+var_21016]
mov     [esp], eax      ; char *
call    _BIO_new_mem_buf
mov     [esp+4], esi
mov     dword ptr [esp+0Ch], 0
mov     dword ptr [esp+8], 0
mov     [esp], eax
call    _PEM_read_bio_RSA_PUBKEY
mov     dword ptr [esp+10h], 80h ; int
mov     dword ptr [esp+8], 1Ch ; size_t
mov     dword ptr [esp], 2A3h ; int
mov     [ebp+var_21188], eax
mov     [esp+14h], eax  ; int
lea     eax, [ebp+var_21116]
mov     [esp+0Ch], eax  ; int
lea     eax, [ebp+var_10017]
mov     [esp+4], eax    ; void *
call    _RSA_verify
mov     esi, [ebp+var_211A0]
mov     edi, [ebp+var_211A4]
cmp     eax, 1
sbb     eax, eax
not     eax
and     eax, 237EEAD6h
mov     [esi+84h], eax
```

Creates a new RSA_PUBKEY object from the key file that was read in

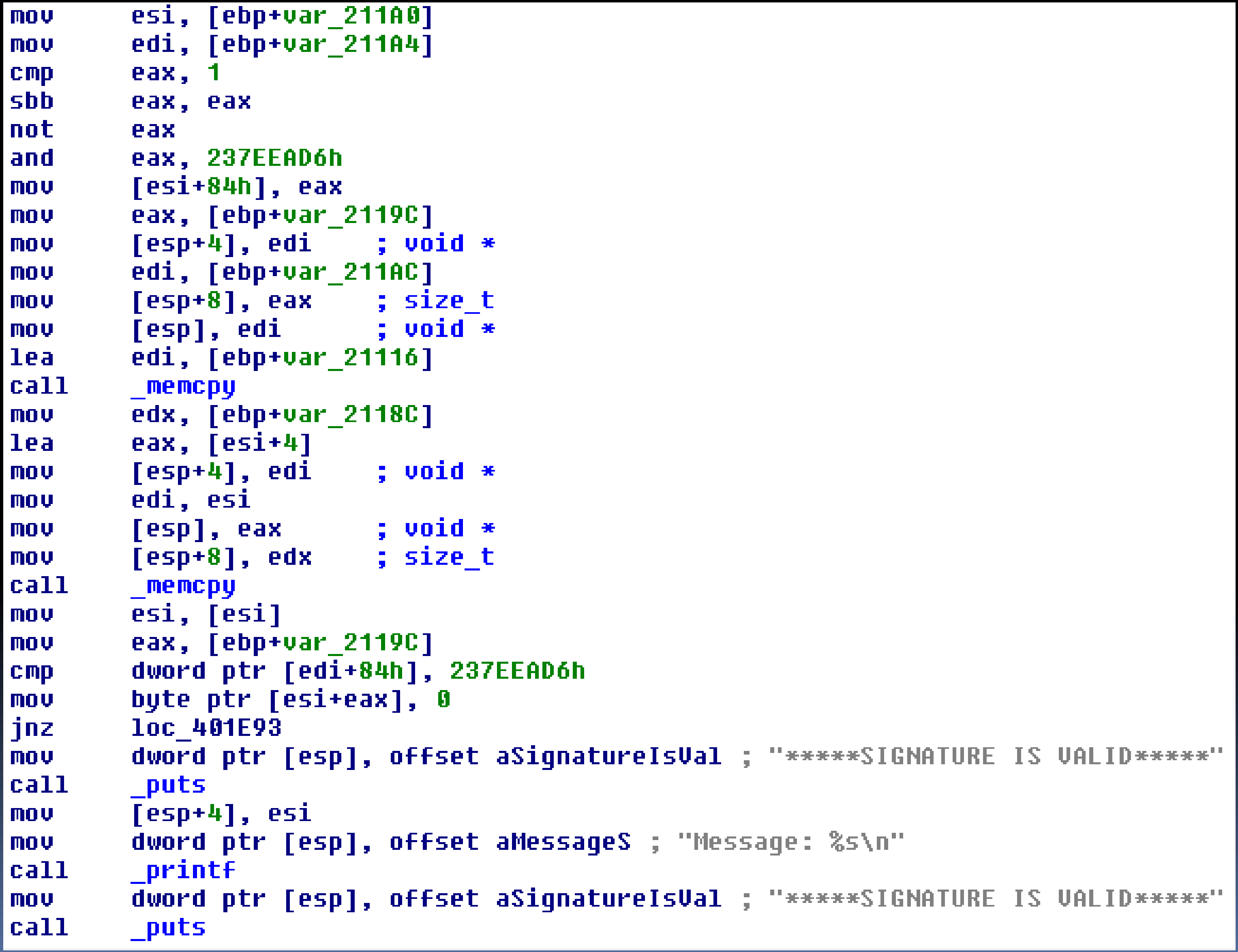
```
_RSA_verify(
    0x2A3,
    SHA224_hash,
    0x1C,
    b64_decoded_data,
    0x80,
    RSA_PUBKEY obj);
```

RSA_verify

```
RSA_verify(                          RSA_verify(
    int type,                            0x2A3,
    unsigned char *hash,                 sha224_hash,
    unsigned int hash_len,               0x1c,
    unsigned char *sigbuf,               b64_decoded_data,
    unsigned int siglen,                 0x80,
    RSA *rsa);                           RSA_PUBKEY_obj);
```

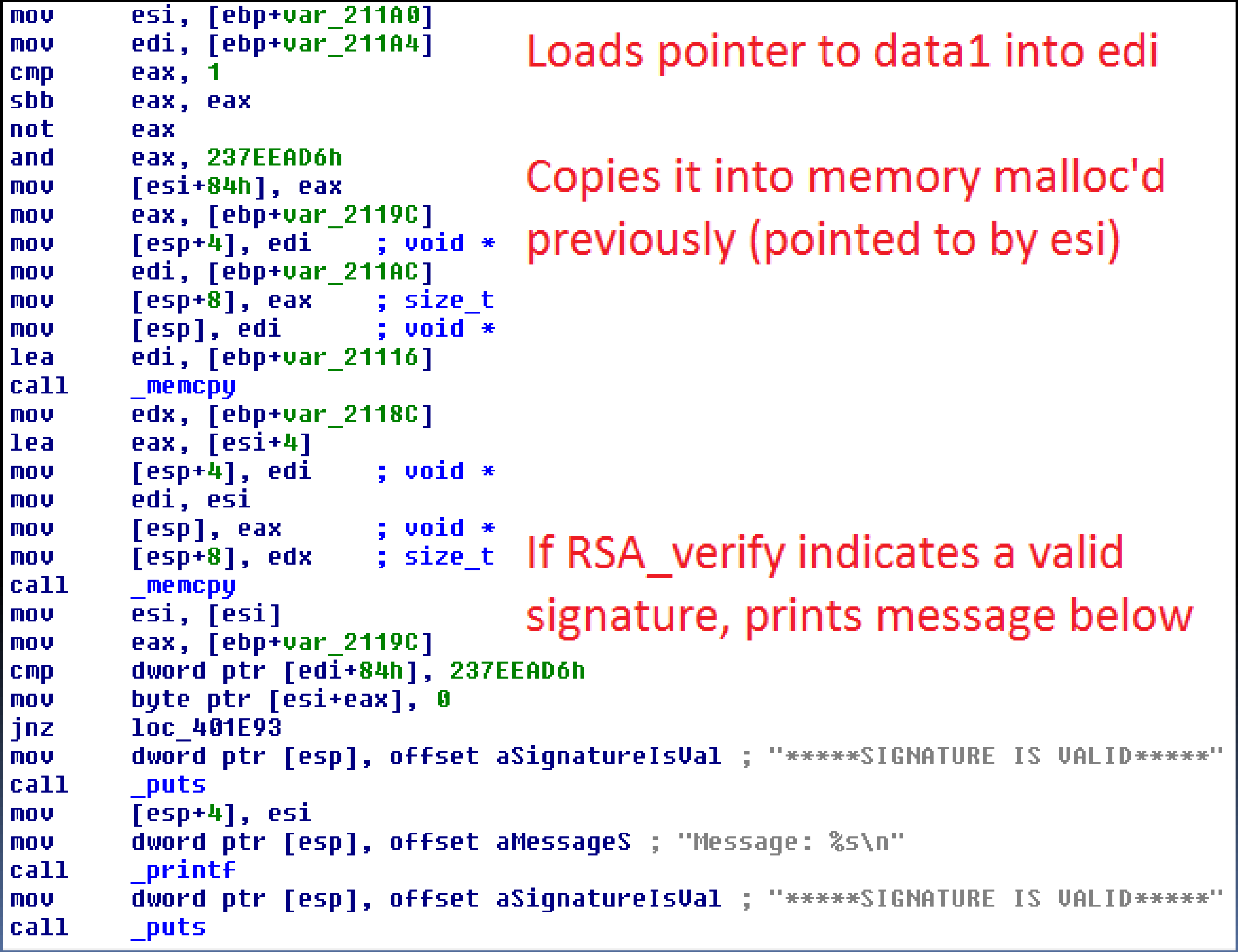
```
So,
'M' | len data2 | len data1 | 0x3A2B | data1 | data2
data1 = ???
data2 = b64( RSA_sign ( SHA224 ( data1 ) ) )
```

Inside _tier2 (17)



```
mov     esi, [ebp+var_211A0]
mov     edi, [ebp+var_211A4]
cmp     eax, 1
sbb     eax, eax
not     eax
and     eax, 237EEAD6h
mov     [esi+84h], eax
mov     eax, [ebp+var_2119C]
mov     [esp+4], edi    ; void *
mov     edi, [ebp+var_211AC]
mov     [esp+8], eax    ; size_t
mov     [esp], edi      ; void *
lea     edi, [ebp+var_21116]
call    _memcpy
mov     edx, [ebp+var_2118C]
lea     eax, [esi+4]
mov     [esp+4], edi    ; void *
mov     edi, esi
mov     [esp], eax      ; void *
mov     [esp+8], edx    ; size_t
call    _memcpy
mov     esi, [esi]
mov     eax, [ebp+var_2119C]
cmp     dword ptr [edi+84h], 237EEAD6h
mov     byte ptr [esi+eax], 0
jnz     loc_401E93
mov     dword ptr [esp], offset aSignatureIsVal ; "*****SIGNATURE IS VALID*****"
call    _puts
mov     [esp+4], esi
mov     dword ptr [esp], offset aMessageS ; "Message: %s\n"
call    _printf
mov     dword ptr [esp], offset aSignatureIsVal ; "*****SIGNATURE IS VALID*****"
call    _puts
```

Inside _tier2 (18)



```
mov     esi, [ebp+var_211A0]
mov     edi, [ebp+var_211A4]
cmp     eax, 1
sbb     eax, eax
not     eax
and     eax, 237EEAD6h
mov     [esi+84h], eax
mov     eax, [ebp+var_2119C]
mov     [esp+4], edi    ; void *
mov     edi, [ebp+var_211AC]
mov     [esp+8], eax    ; size_t
mov     [esp], edi      ; void *
lea     edi, [ebp+var_21116]
call    _memcpy
mov     edx, [ebp+var_2118C]
lea     eax, [esi+4]
mov     [esp+4], edi    ; void *
mov     edi, esi
mov     [esp], eax      ; void *
mov     [esp+8], edx    ; size_t
call    _memcpy
mov     esi, [esi]
mov     eax, [ebp+var_2119C]
cmp     dword ptr [edi+84h], 237EEAD6h
mov     byte ptr [esi+eax], 0
jnz     loc_401E93
mov     dword ptr [esp], offset aSignatureIsVal ; "*****SIGNATURE IS VALID*****"
call    _puts
mov     [esp+4], esi
mov     dword ptr [esp], offset aMessageS ; "Message: %s\n"
call    _printf
mov     dword ptr [esp], offset aSignatureIsVal ; "*****SIGNATURE IS VALID*****"
call    _puts
```

Loads pointer to data1 into edi

Copies it into memory malloc'd previously (pointed to by esi)

If RSA_verify indicates a valid signature, prints message below

We can now craft messages!

```
‘M’ | len data2 | len data1 | 0x3A2B | data1 | data2
data1 = message text
data2 = b64( RSA_sign ( SHA224 ( data1 ) ) )
```

- Compute hash of the message text
- Compute RSA signature of message text hash using provided RSA private key
- Base64 encode the RSA signature
- Calculate lengths
- Build header
- Encode in tabs and spaces

Task 3 Complete!

- Required reverse engineering the algorithm and writing a complimentary solution
- … on to Task 4!

2015 Challenge – Task 4

- A military organization wants to make the messages appear to come from the group's leadership. … Program binaries and keys have already been distributed throughout the terrorist organization, though, so achieving this effect must be done only via the message file.

- Craft a message that can be sent to the same high-ranking member that the message from Task 1 was originally sent to

- Provided:
  - A message to encode
  - A text file to encode the message into

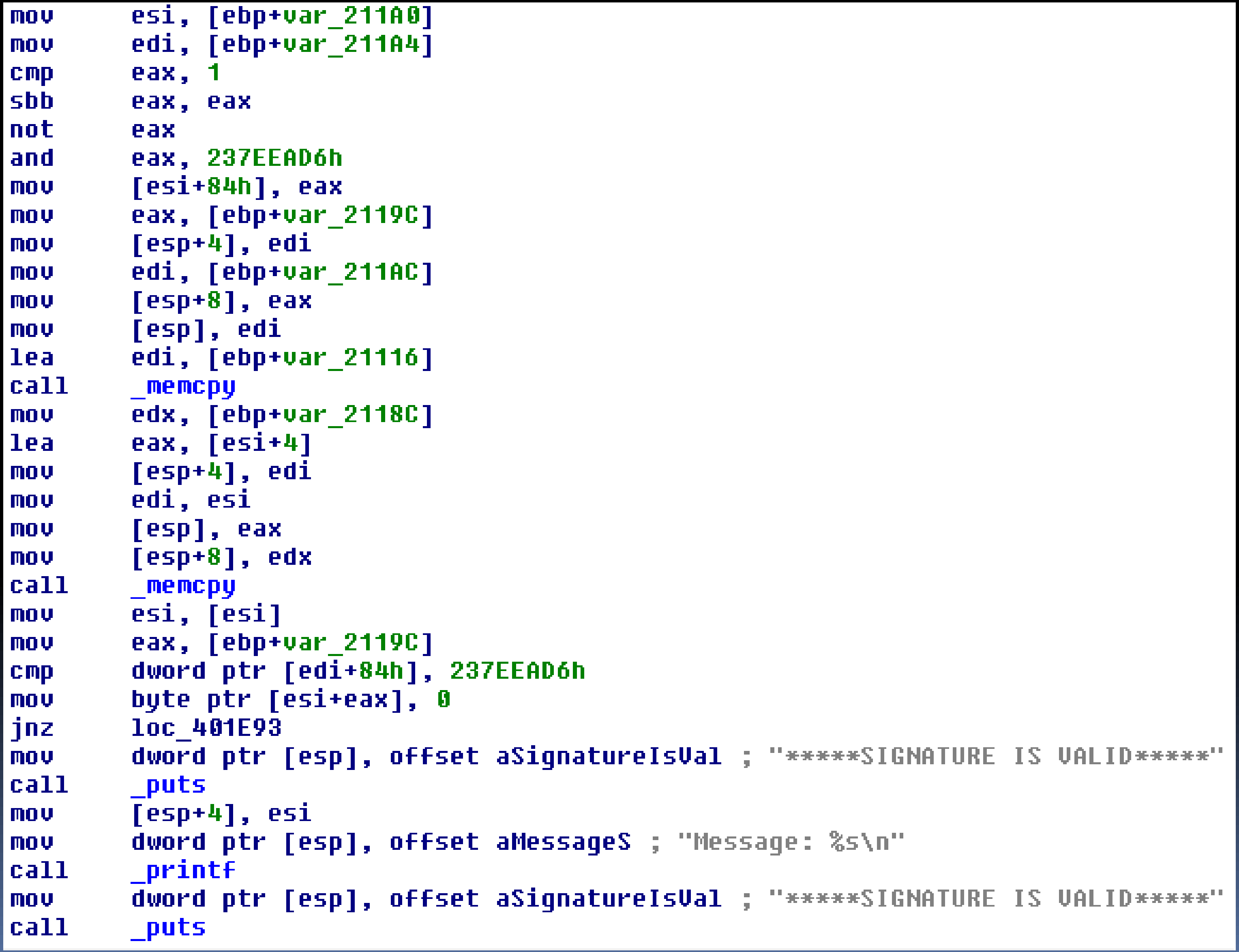
The problem... No private key ☹

```
M | len data2 | len data1 | 0x3A 0x2B | data1 | data2
data1 = message text
```

```
data2 = b64( RSA_sign ( SHA224( data1 ) ) )
```

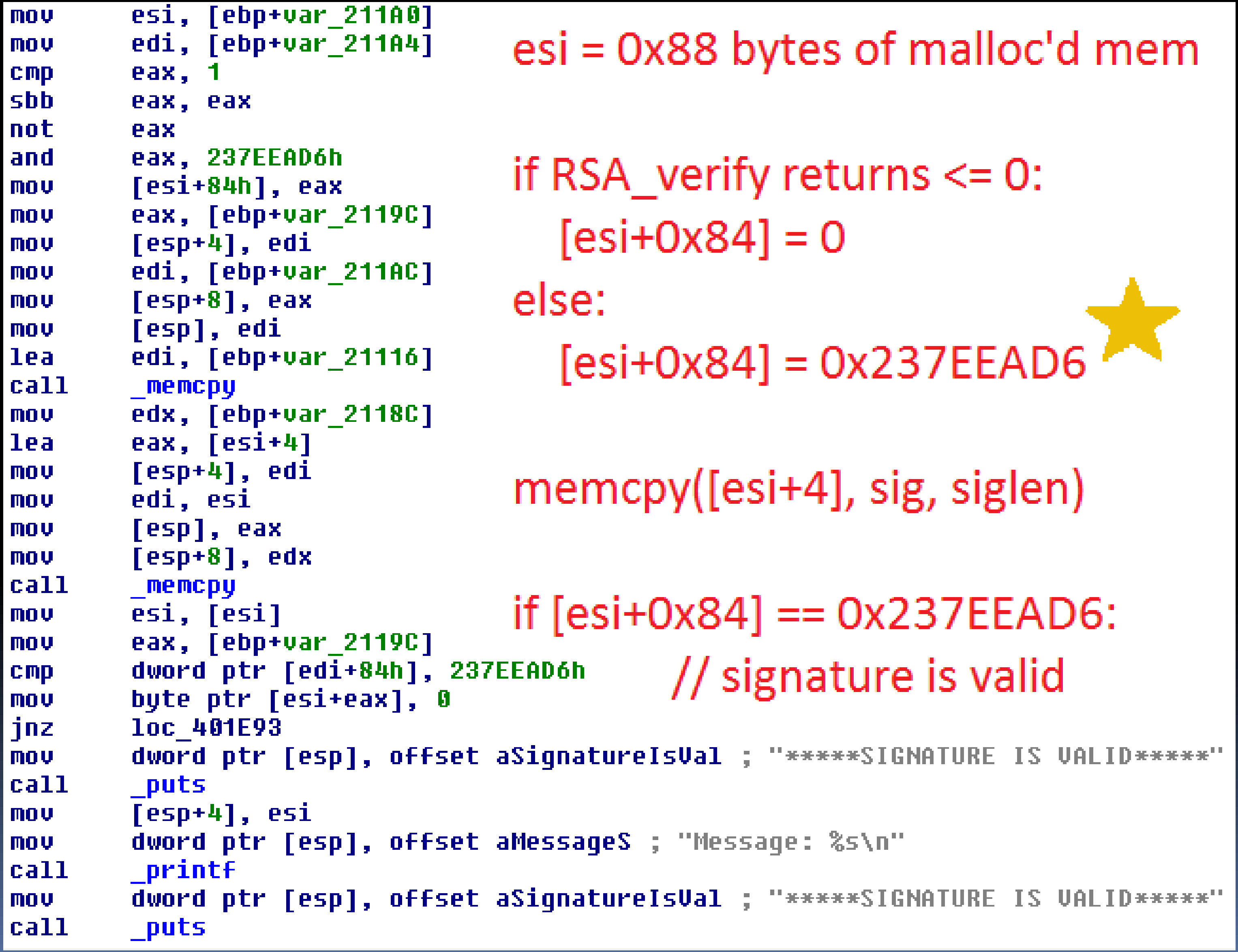
- We have the person’s public key, but computing the RSA signature requires the private key
- Maybe there is a flaw we can exploit?

A further look ...



```
mov     esi, [ebp+var_211A0]
mov     edi, [ebp+var_211A4]
cmp     eax, 1
sbb     eax, eax
not     eax
and     eax, 237EEAD6h
mov     [esi+84h], eax
mov     eax, [ebp+var_2119C]
mov     [esp+4], edi
mov     edi, [ebp+var_211AC]
mov     [esp+8], eax
mov     [esp], edi
lea     edi, [ebp+var_21116]
call    _memcpy
mov     edx, [ebp+var_2118C]
lea     eax, [esi+4]
mov     [esp+4], edi
mov     edi, esi
mov     [esp], eax
mov     [esp+8], edx
call    _memcpy
mov     esi, [esi]
mov     eax, [ebp+var_2119C]
cmp     dword ptr [edi+84h], 237EEAD6h
mov     byte ptr [esi+eax], 0
jnz     loc_401E93
mov     dword ptr [esp], offset aSignatureIsVal ; "*****SIGNATURE IS VALID*****"
call    _puts
mov     [esp+4], esi
mov     dword ptr [esp], offset aMessageS ; "Message: %s\n"
call    _printf
mov     dword ptr [esp], offset aSignatureIsVal ; "*****SIGNATURE IS VALID*****"
call    _puts
```

A further look ... (2)



```
mov     esi, [ebp+var_211A0]
mov     edi, [ebp+var_211A4]
cmp     eax, 1
sbb     eax, eax
not     eax
and     eax, 237EEAD6h
mov     [esi+84h], eax
mov     eax, [ebp+var_2119C]
mov     [esp+4], edi
mov     edi, [ebp+var_211AC]
mov     [esp+8], eax
mov     [esp], edi
lea     edi, [ebp+var_21116]
call    _memcpy
mov     edx, [ebp+var_2118C]
lea     eax, [esi+4]
mov     [esp+4], edi
mov     edi, esi
mov     [esp], eax
mov     [esp+8], edx
call    _memcpy
mov     esi, [esi]
mov     eax, [ebp+var_2119C]
cmp     dword ptr [edi+84h], 237EEAD6h
mov     byte ptr [esi+eax], 0
jnz     loc_401E93
mov     dword ptr [esp], offset aSignatureIsVal ; "*****SIGNATURE IS VALID*****"
call    _puts
mov     [esp+4], esi
mov     dword ptr [esp], offset aMessageS ; "Message: %s\n"
call    _printf
mov     dword ptr [esp], offset aSignatureIsVal ; "*****SIGNATURE IS VALID*****"
call    _puts
```

```
esi = 0x88 bytes of malloc'd mem

if RSA_verify returns <= 0:
    [esi+0x84] = 0
else:
    [esi+0x84] = 0x237EEAD6 ★

memcpy([esi+4], sig, siglen)

if [esi+0x84] == 0x237EEAD6:
    // signature is valid
```

A problem

```
'M' | len data2 | len data1 | 0x3A2B | data1 | data2
```

```
esi = 0x88 bytes of malloc'd mem
if RSA_verify returns <= 0:
  [esi+0x84] = 0
else:
  [esi+0x84] = 0x237EEAD6

memcpy([esi+4], sig, siglen)

if [esi+0x84] == 0x237EEAD6:
    // signature is valid
```

Both are set by Base64Decode, based on 'data2' and 'len data2'

A problem

```
'M' | len data2 | len data1 | 0x3A2B | data1 | data2
```

```
esi = 0x88 bytes of malloc'd mem
if RSA_verify returns <= 0:
   [esi+0x84] = 0
else:
   [esi+0x84] = 0x237EEAD6

memcpy([esi+4], sig, siglen)

if [esi+0x84] == 0x237EEAD6:
       // signature is valid
```

If siglen is greater than 0x80, the memcpy will overwrite the signature verification value with data from sig

To exploit, craft data2 such that base64 decodes into a buffer with 0x237EEAD6 at byte 0x80

An alternate solution...

- Recall:
  - Intelligence suggests … a customized cryptographic implementation is used to generate a public/private key pair, which is then used to authenticate messages from leadership
- Maybe there's a problem with the keys…

With your powers combined

- Task 1: public key for high-ranking member
- Task 2: public key for field operative

- From Wikipedia, regarding attacks on the RSA cryptosystem:

```
If n = pq is one public key and n' = p'q' is
another, then if by chance p = p' … then a
simple computation of gcd(n,n') = p factors both
n and n', totally compromising both keys.
```

https://en.wikipedia.org/wiki/RSA_(cryptosystem)#Security_and_practical_considerations

The keys share a common factor ☺

- Computing the GCD on both keys reveals the private key to both
- This can be used to sign a message to either recipient

- Idea for this attack:
  - 2012 research paper from U of Mich:

```
We were able to remotely obtain the RSA private keys for
0.50% of TLS hosts and 0.03% of SSH hosts because their
public keys shared nontrivial common factors due to poor
randomness.
```

factorable.net

Task 4 Complete!

- Required:
  - Exploiting the four-byte buffer overflow vuln, or
  - Computing the GCD of the provided public keys

Questions

?

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

Check the site for an event code to use when applying (to associate yourself with the Codebreaker Challenge)