ECE560
Computer and Information Security

Fall 2020

Reverse Engineering

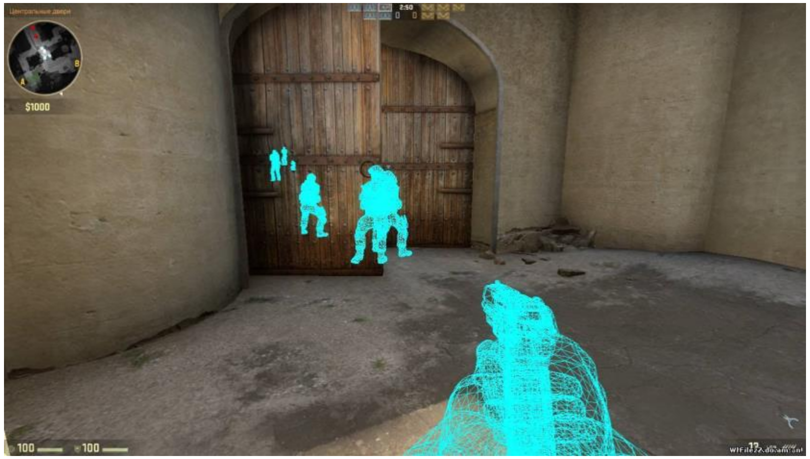
Tyler Bletsch

Duke University

With additional content by Jiaming Li, NC State University, 2015

What is software reverse engineering?

- Determine and possibly change program logic
  - “Logic” ≠ Just observed behavior
- Ethics
  - Useful for good:
    - Analyze malware
    - Understand undocumented legacy code
    - Watch/read/play the stuff you paid for
  - Useful for evil
    - “Crack” software
      (remove restrictions)
    - Find exploits
    - Cheat at games



Types of tools

- Disassembler
  - Turn compiled program into assembly
  - Not perfect
  - Static tool
- Debugger
  - Step through running program
  - Dynamic tool
- Hex editor
  - Make changes to binaries
- Monitoring tools
  - Watch system calls, library calls, etc.

Also, **decompiler**:

- Attempts to turn assembly back into source code.
- Usually awful at machine code, but managed code (e.g. Java, Python) can produce decent results.

Examples of tools

- Linux:
  - Disassember: objdump (free), **IDA Pro** (free and paid versions), Ghidra (free, from NSA!)
  - Debugger: **gdb** and its front-ends
  - Hex editor: okteta, bless, lots more…
  - Monitoring: **strace**, ltrace

IDA Pro eats basically anything

- Windows:
  - Disassembler: **IDA Pro** (free and paid versions), Ghidra (free, from NSA!)
  - Debugger: WinDBG (basic), **OllyDbg** (shareware), SoftICE ($1000+)
  - Hex editor: XVI32, Notepad++ with plugin, etc.
  - Monitoring tools: Process Monitor, Explorer, and more.

- X86 in general: A hypervisor (VMware, KVM, etc.)

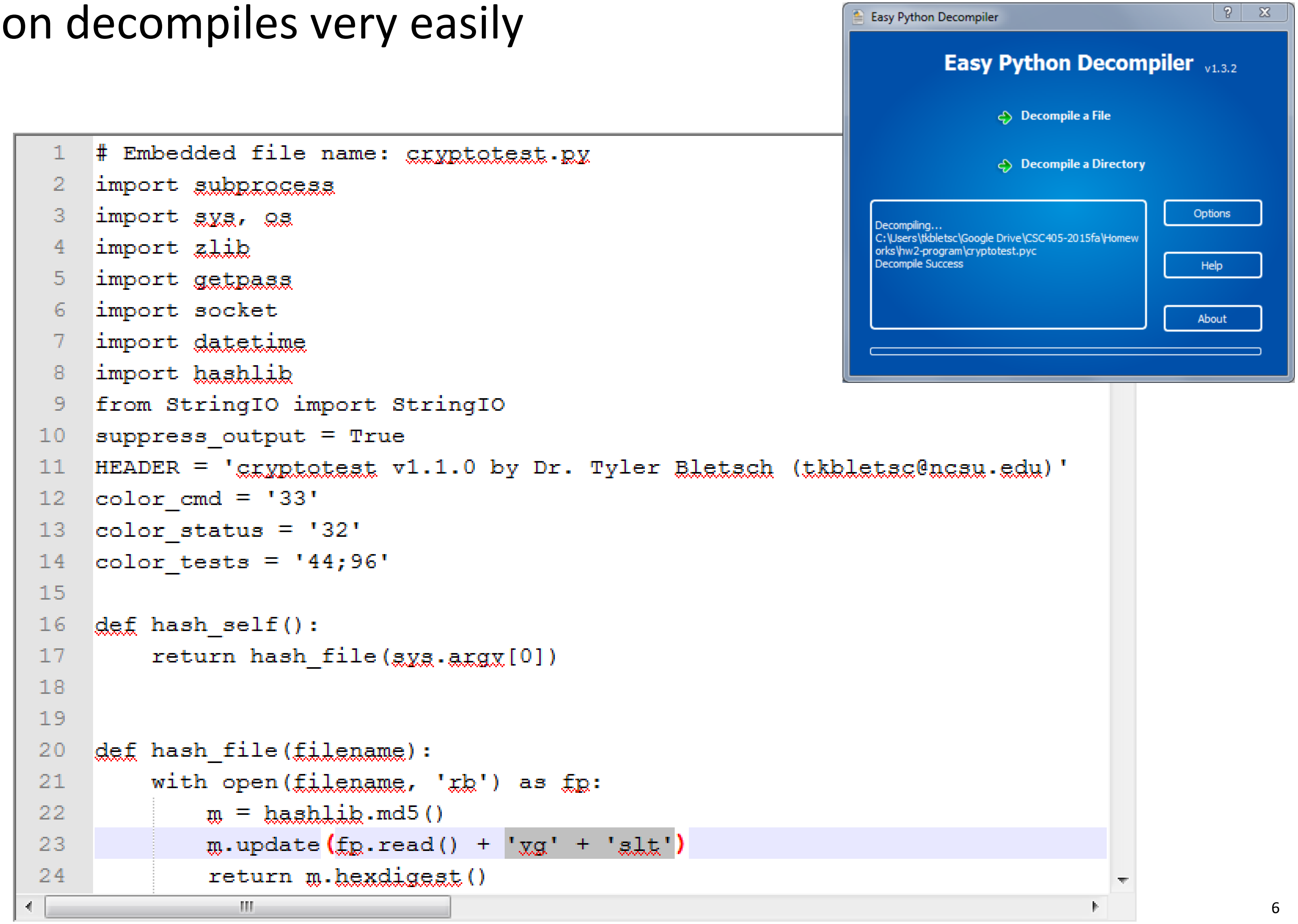
Debug or disassemble? Both.

- Disassembler gives **static** results
  - Good overview of program logic
  - But need to “mentally execute” program
  - Difficult to jump to specific functionality in the code
- Debugger is **dynamic**
  - Can set break points; fast forward to code for relevant functionality
  - Can treat complex code as “black box”
  - Not all code disassembles correctly
- Disassembler **and** debugger both required for any serious reverse engineering task

From "Computer Science 654 Lecture 5: Software Reverse Engineering" by Wayne Patterson, Howard Univ. 2009.

Example 1: HW2 auto-grader

- Python decompiles very easily



```
Decompiling...
C:\Users\tkbletsc\Google Drive\CSC405-2015fa\Homeworks\hw2-program\cryptotest.pyc
Decompile Success
```

```python
# Embedded file name: cryptotest.py
import subprocess
import sys, os
import zlib
import getpass
import socket
import datetime
import hashlib
from StringIO import StringIO
suppress_output = True
HEADER = 'cryptotest v1.1.0 by Dr. Tyler Bletsch (tkbletsc@ncsu.edu)'
color_cmd = '33'
color_status = '32'
color_tests = '44;96'

def hash_self():
    return hash_file(sys.argv[0])


def hash_file(filename):
    with open(filename, 'rb') as fp:
        m = hashlib.md5()
        m.update(fp.read() + 'yg' + 'slt')
        return m.hexdigest()
```

Example 2: Minecraft

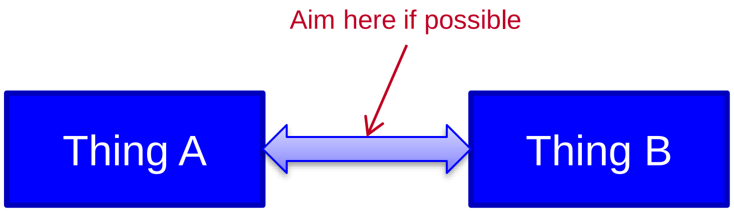
- Minecraft is a Java program, no mod support
- All mods use something like the Mod Coder Pack (MCP):

> "Use MCP to decompile the Minecraft client and server jar files.
>
> Use the decompiled source code to create mods for Minecraft.
>
> Recompile modified versions of Minecraft.
>
> Reobfuscate the classes of your mod for Minecraft."

- Entire mod community is built on reverse engineering!

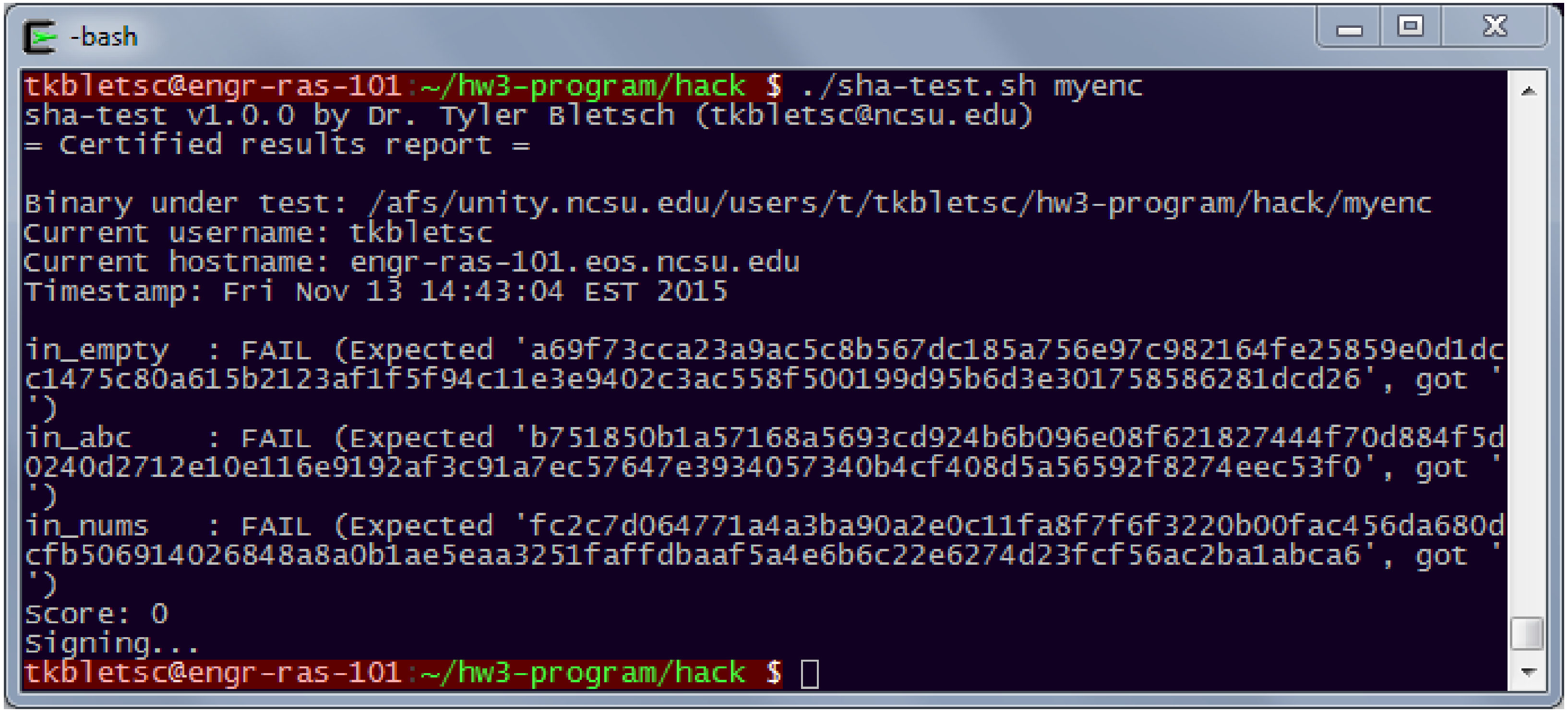
Examining multi-component systems

- Weaknesses often at the seams – where parts of system come together
  - Most visible, often exploitable
  - Example: SQL injection
- If not the seams, at least focus on the least protected part



Example 3: Auto-grader for a homework question you didn't get

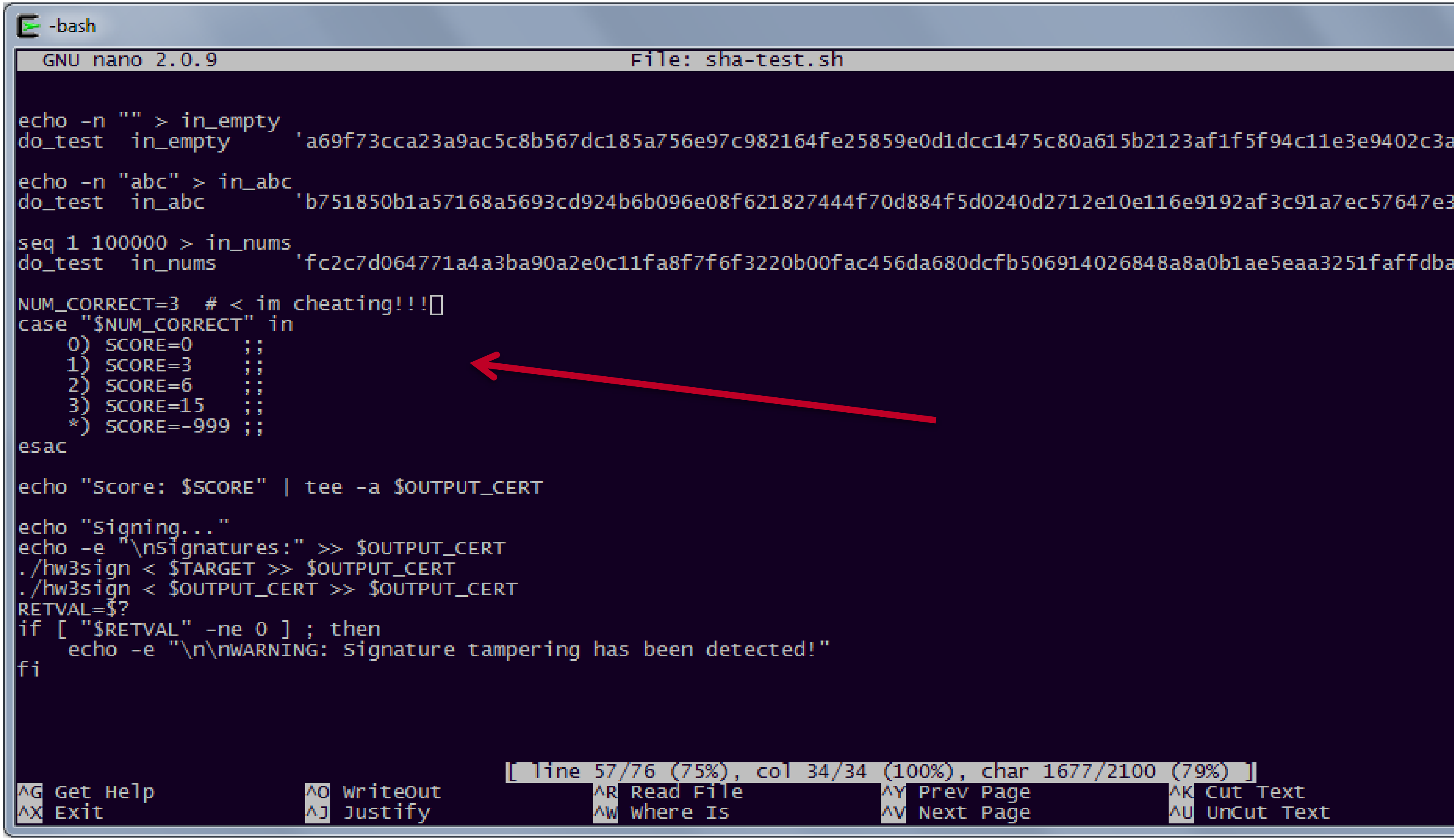
- Was used last year, but cut because it involves implementing SHA-3, which is commonly supported in most libraries by now.
- We'll focus on the auto-grader and its anti-tamper mechanisms.
- Two files: (1) Binary **`hw3sign`**, (2) shell script **`sha-test.sh`**
- Normal usage:



```
tkbletsc@engr-ras-101:~/hw3-program/hack $ ./sha-test.sh myenc
sha-test v1.0.0 by Dr. Tyler Bletsch (tkbletsc@ncsu.edu)
= Certified results report =

Binary under test: /afs/unity.ncsu.edu/users/t/tkbletsc/hw3-program/hack/myenc
Current username: tkbletsc
Current hostname: engr-ras-101.eos.ncsu.edu
Timestamp: Fri Nov 13 14:43:04 EST 2015

in_empty  : FAIL (Expected 'a69f73cca23a9ac5c8b567dc185a756e97c982164fe25859e0d1dc
c1475c80a615b2123af1f5f94c11e3e9402c3ac558f500199d95b6d3e301758586281dcd26', got '
')
in_abc    : FAIL (Expected 'b751850b1a57168a5693cd924b6b096e08f621827444f70d884f5d
0240d2712e10e116e9192af3c91a7ec57647e3934057340b4cf408d5a56592f8274eec53f0', got '
')
in_nums   : FAIL (Expected 'fc2c7d064771a4a3ba90a2e0c11fa8f7f6f3220b00fac456da680d
cfb506914026848a8a0b1ae5eaa3251faffdbaaf5a4e6b6c22e6274d23fcf56ac2ba1abca6', got '
')
Score: 0
Signing...
tkbletsc@engr-ras-101:~/hw3-program/hack $ 
```

Example 3: Auto-grader for a homework question you didn't get

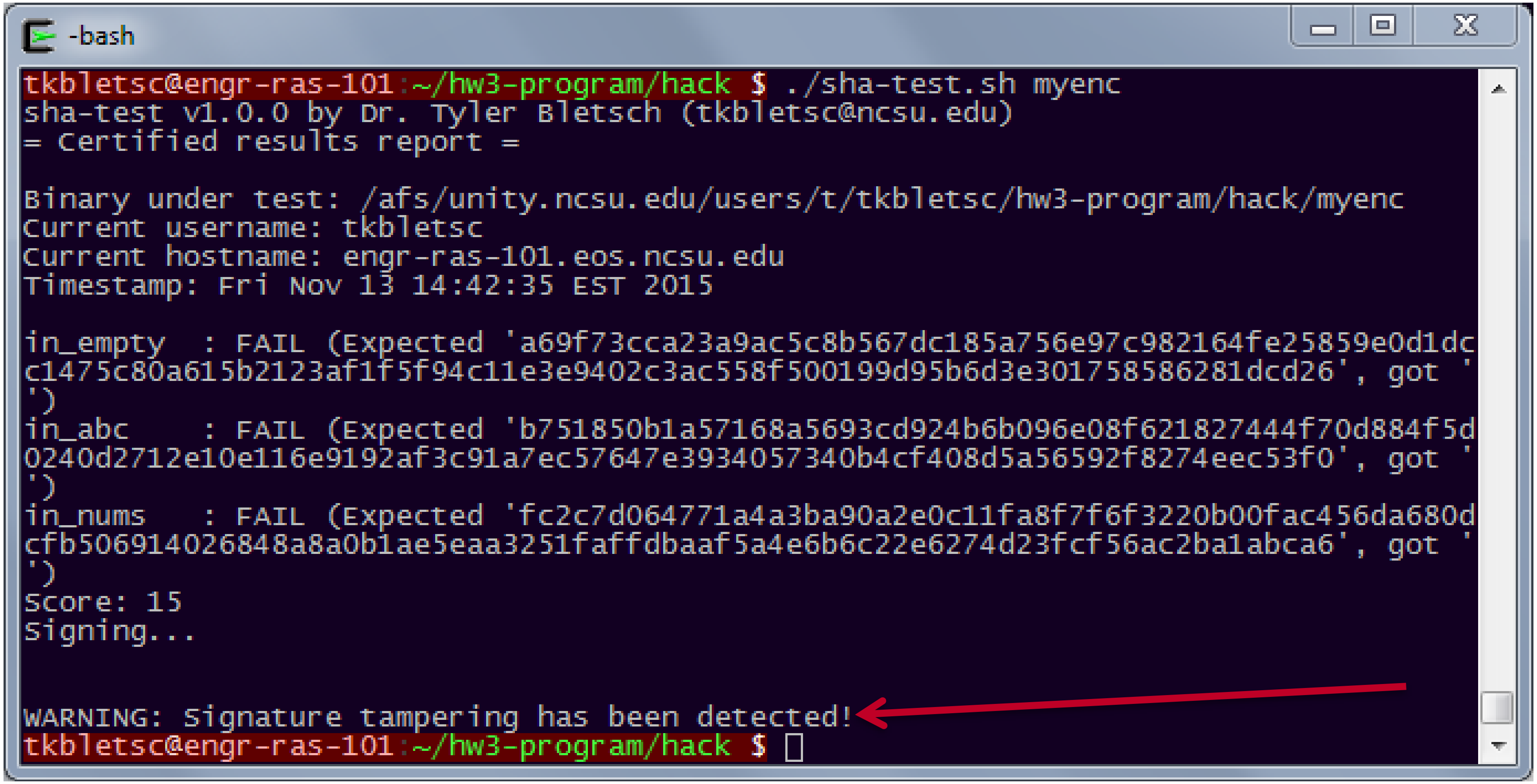
- Naïve attack: Just change the script



```
  GNU nano 2.0.9                                    File: sha-test.sh


echo -n "" > in_empty
do_test  in_empty      'a69f73cca23a9ac5c8b567dc185a756e97c982164fe25859e0d1dcc1475c80a615b2123af1f5f94c11e3e9402c3a

echo -n "abc" > in_abc
do_test  in_abc        'b751850b1a57168a5693cd924b6b096e08f621827444f70d884f5d0240d2712e10e116e9192af3c91a7ec57647e3

seq 1 100000 > in_nums
do_test  in_nums       'fc2c7d064771a4a3ba90a2e0c11fa8f7f6f3220b00fac456da680dcfb506914026848a8a0b1ae5eaa3251faffdba

NUM_CORRECT=3  # < im cheating!!!
case "$NUM_CORRECT" in
    0) SCORE=0    ;;
    1) SCORE=3    ;;
    2) SCORE=6    ;;
    3) SCORE=15   ;;
    *) SCORE=-999 ;;
esac

echo "Score: $SCORE" | tee -a $OUTPUT_CERT

echo "Signing..."
echo -e "\nSignatures:" >> $OUTPUT_CERT
./hw3sign < $TARGET >> $OUTPUT_CERT
./hw3sign < $OUTPUT_CERT >> $OUTPUT_CERT
RETVAL=$?
if [ "$RETVAL" -ne 0 ] ; then
    echo -e "\n\nWARNING: Signature tampering has been detected!"
fi



                                                [ line 57/76 (75%), col 34/34 (100%), char 1677/2100 (79%) ]
^G Get Help          ^O WriteOut          ^R Read File         ^Y Prev Page         ^K Cut Text
^X Exit              ^J Justify           ^W Where Is          ^V Next Page         ^U UnCut Text
```

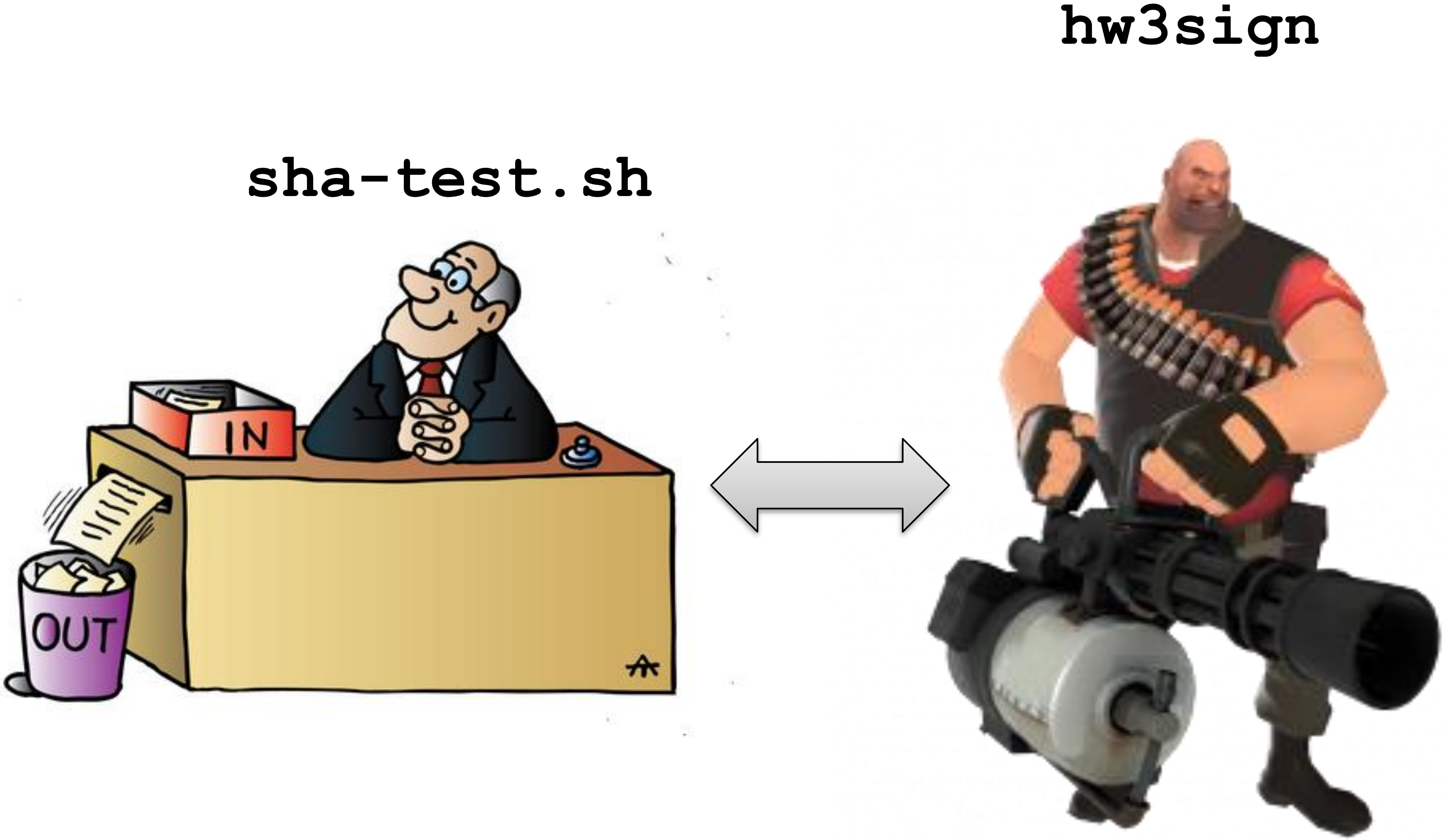
Example 3: Lost HW autograder

- Naïve attack: Just change the script
  - Failed: **`hw3sign`** must be checking it somehow!



```
tkbletsc@engr-ras-101:~/hw3-program/hack $ ./sha-test.sh myenc
sha-test v1.0.0 by Dr. Tyler Bletsch (tkbletsc@ncsu.edu)
= Certified results report =

Binary under test: /afs/unity.ncsu.edu/users/t/tkbletsc/hw3-program/hack/myenc
Current username: tkbletsc
Current hostname: engr-ras-101.eos.ncsu.edu
Timestamp: Fri Nov 13 14:42:35 EST 2015

in_empty  : FAIL (Expected 'a69f73cca23a9ac5c8b567dc185a756e97c982164fe25859e0d1dc
c1475c80a615b2123af1f5f94c11e3e9402c3ac558f500199d95b6d3e301758586281dcd26', got '
')
in_abc    : FAIL (Expected 'b751850b1a57168a5693cd924b6b096e08f621827444f70d884f5d
0240d2712e10e116e9192af3c91a7ec57647e3934057340b4cf408d5a56592f8274eec53f0', got '
')
in_nums   : FAIL (Expected 'fc2c7d064771a4a3ba90a2e0c11fa8f7f6f3220b00fac456da680d
cfb506914026848a8a0b1ae5eaa3251faffdbaaf5a4e6b6c22e6274d23fcf56ac2ba1abca6', got '
')
Score: 15
Signing...


WARNING: Signature tampering has been detected!
tkbletsc@engr-ras-101:~/hw3-program/hack $
```

Example 3: Lost HW autograder Topology



hw3sign

sha-test.sh

Example 3: Lost HW autograder

- Could look at behavior with **strace**:

```
$ strace -f -o trace.txt ./sha-test.sh myenc
          ...
$ cat trace.txt
4127  execve("./sha-test.sh", ["./sha-test.sh", "myenc"], [/* 46 vars */]) = 0
4127  brk(0)                                   = 0x1700000
4127  mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
    = 0x7f55d5a17000
4127  access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or
    directory)
4127  open("/etc/ld.so.cache", O_RDONLY) = 3
4127  fstat(3, {st_mode=S_IFREG|0644, st_size=210058, ...}) = 0
4127  mmap(NULL, 210058, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f55d59e3000
4127 close(3)                                 = 0
          ...
```

- But `hw3sign` never appears to open `sha-test.sh`:

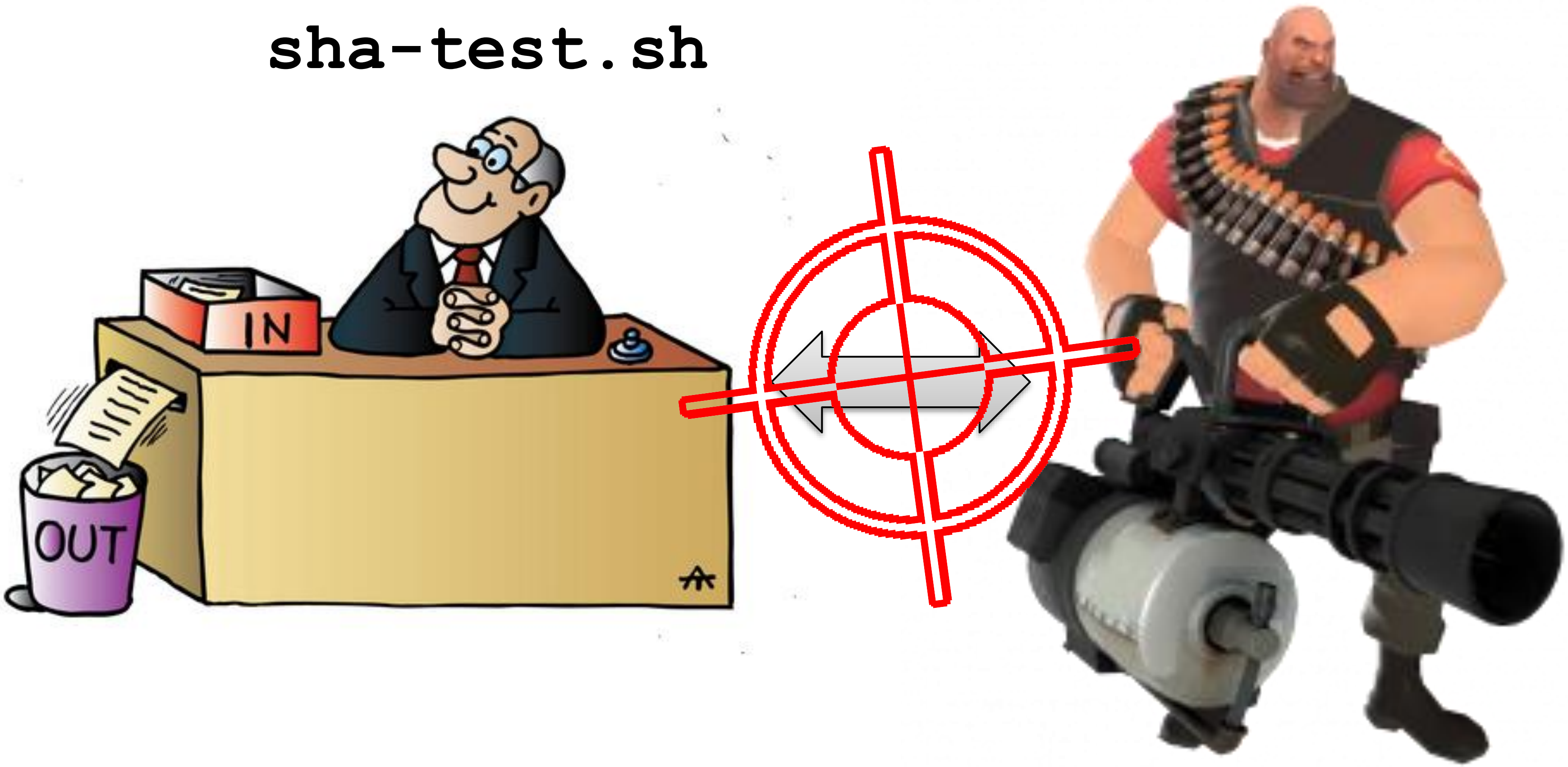
```
$ grep open trace.txt | grep sha-test.sh
4127  open("./sha-test.sh", O_RDONLY)    = 3
```

  - This one line is from when `sha-test.sh` itself is started
  - There’s more mystery here that I’ll leave to you…

Example 3: Lost HW autograder
Best place to attack?

hw3sign

sha-test.sh



Example 3: Lost HW autograder

- Two past successful student attacks
- **Black box attack:**
  - hw3sign signs the binary, then the certificate itself
  - What if we ask it to "test" a doctored certificate as a binary – it will sign it for us! No understanding needed!
- **Chameleon attack:**
  - Add cheating to sha-test.sh; also add code to copy a legit sha-test.sh over itself before doing signings
  - Malicious behavior occurs then hides before check occurs
  - Example of a TOCTOU attack (Time-Of-Check/Time-Of-Use)!

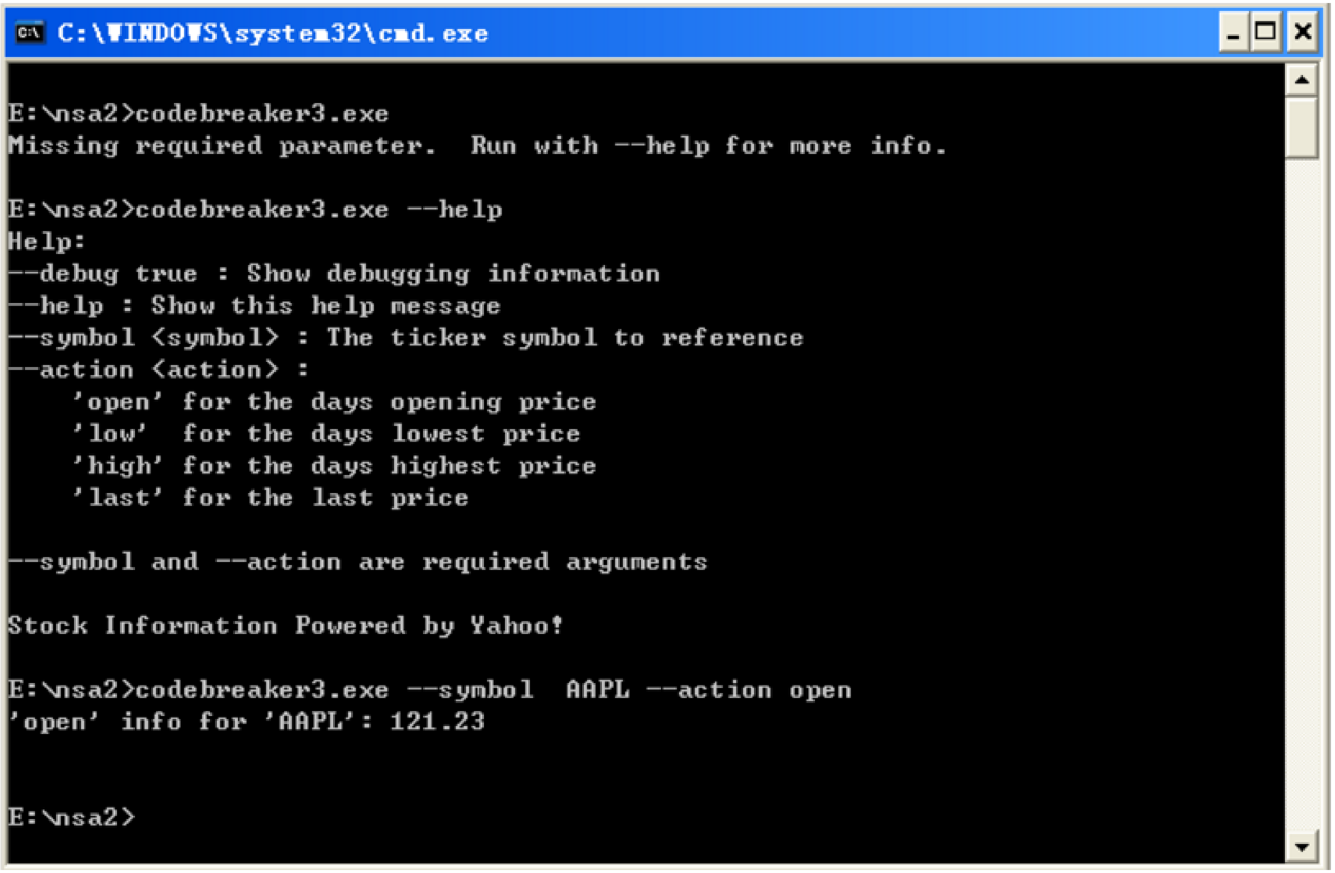
Example 4:
NSA Codebreaker challenge, 2015

- Scenario:
  - Terrorists using a cryptography program to decrypt/authenticate messages from leadership
  - What we have:
    - The program: **`codebreaker3.exe`**
    - A member's key: **`tier1_key.pem`**
    - A text file with a hidden message: **`tier1_msg.txt`**
  - At first glance, the program appears to simply check stock information, but that's a ruse.
  - Need to reverse engineer it:
    Challenge has 4 tasks, we'll do 2.

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 1: Decrypt

- Need to decode message we have.
- The program:



```
E:\nsa2>codebreaker3.exe
Missing required parameter.  Run with --help for more info.

E:\nsa2>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!

E:\nsa2>codebreaker3.exe --symbol  AAPL --action open
'open' info for 'AAPL': 121.23


E:\nsa2>
```

Adapted from content by Jiaming Li, NC State University, 2015

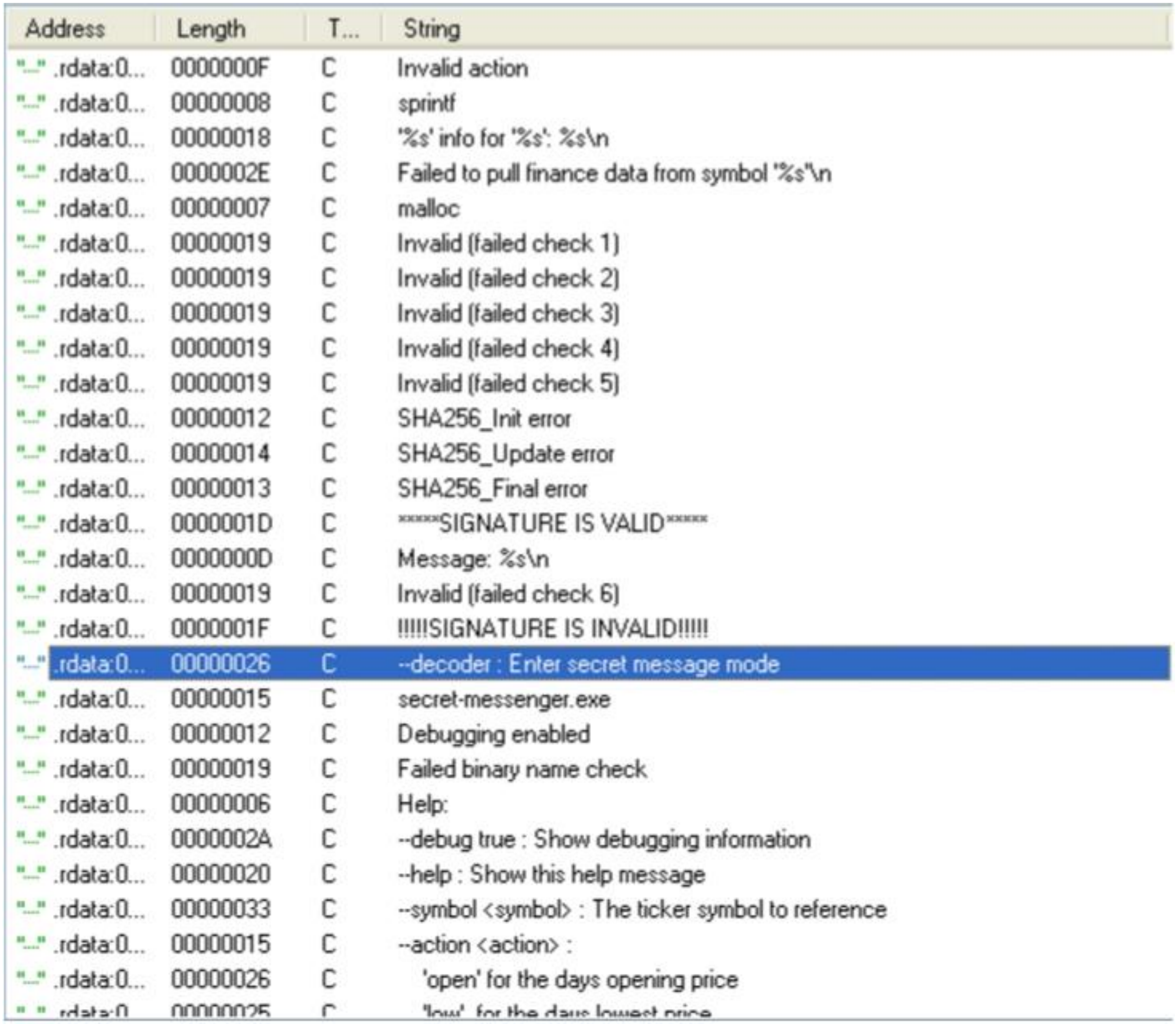
Codebreaker Task 1: Decrypt

- Do static analysis with IDA Pro
  - Load binary
  - Confirm binary format options
  - Process:
    - Code is disassembled
    - Call graph of assembly code built
    - All memory references are cross-referenced, especially string literals

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 1: Decrypt

- Do static analysis with IDA Pro, check the all string information

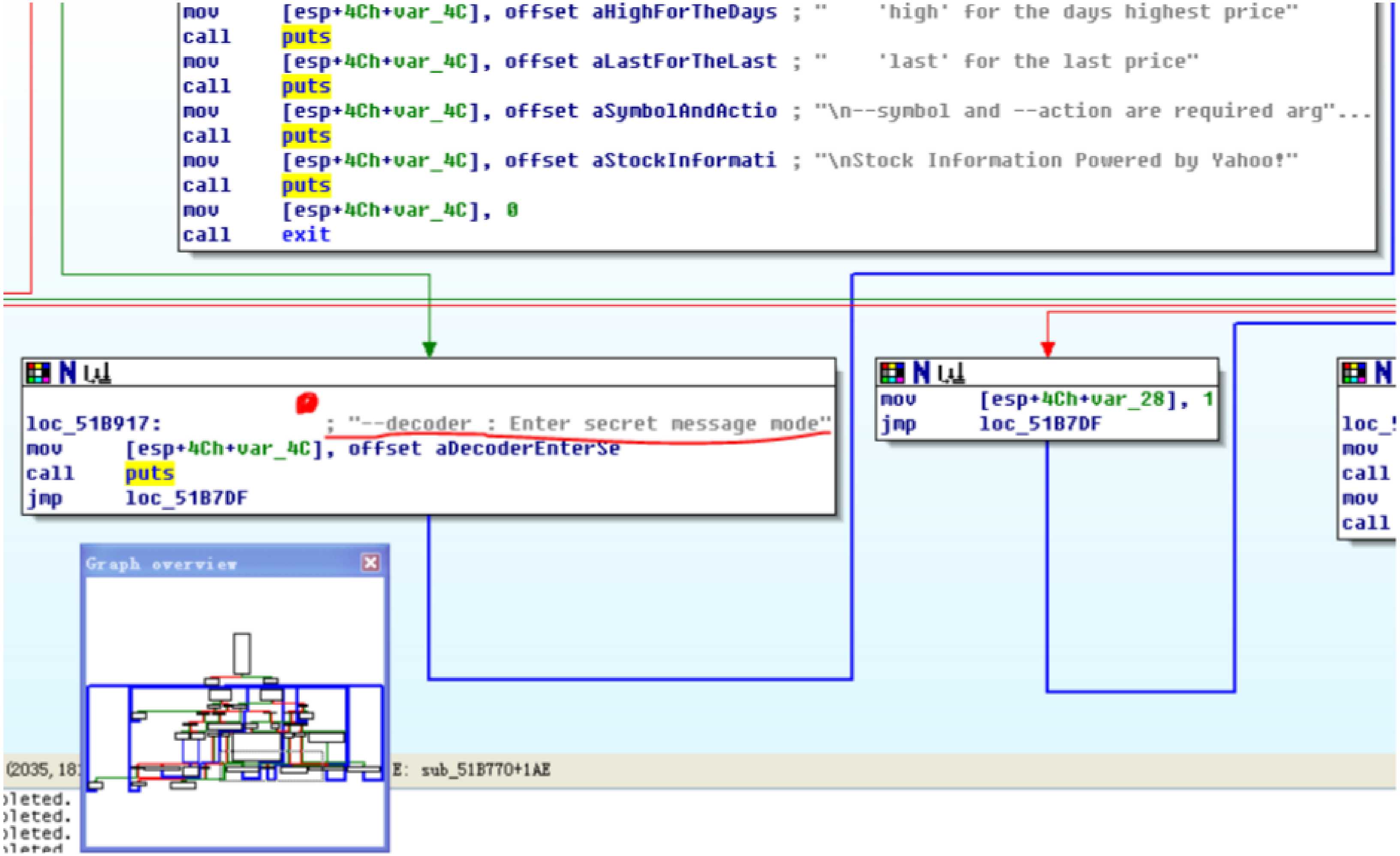


Address	Length	T...	String
.rdata:0...	0000000F	C	Invalid action
.rdata:0...	00000008	C	sprintf
.rdata:0...	00000018	C	'%s' info for '%s': %s\n
.rdata:0...	0000002E	C	Failed to pull finance data from symbol '%s'\n
.rdata:0...	00000007	C	malloc
.rdata:0...	00000019	C	Invalid (failed check 1)
.rdata:0...	00000019	C	Invalid (failed check 2)
.rdata:0...	00000019	C	Invalid (failed check 3)
.rdata:0...	00000019	C	Invalid (failed check 4)
.rdata:0...	00000019	C	Invalid (failed check 5)
.rdata:0...	00000012	C	SHA256_Init error
.rdata:0...	00000014	C	SHA256_Update error
.rdata:0...	00000013	C	SHA256_Final error
.rdata:0...	0000001D	C	*****SIGNATURE IS VALID*****
.rdata:0...	0000000D	C	Message: %s\n
.rdata:0...	00000019	C	Invalid (failed check 6)
.rdata:0...	0000001F	C	!!!!!SIGNATURE IS INVALID!!!!!
.rdata:0...	00000026	C	--decoder : Enter secret message mode
.rdata:0...	00000015	C	secret-messenger.exe
.rdata:0...	00000012	C	Debugging enabled
.rdata:0...	00000019	C	Failed binary name check
.rdata:0...	00000006	C	Help:
.rdata:0...	0000002A	C	--debug true : Show debugging information
.rdata:0...	00000020	C	--help : Show this help message
.rdata:0...	00000033	C	--symbol <symbol> : The ticker symbol to reference
.rdata:0...	00000015	C	--action <action> :
.rdata:0...	00000026	C	'open' for the days opening price
.rdata:0	00000025	C	'low' for the days lowest price

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 1: Decrypt

- Press x, this leads us to the location where this string appears:



```
mov     [esp+4Ch+var_4C], offset aHighForTheDays ; "    'high' for the days highest price"
call    puts
mov     [esp+4Ch+var_4C], offset aLastForTheLast ; "    'last' for the last price"
call    puts
mov     [esp+4Ch+var_4C], offset aSymbolAndActio ; "\n--symbol and --action are required arg"...
call    puts
mov     [esp+4Ch+var_4C], offset aStockInformati ; "\nStock Information Powered by Yahoo!"
call    puts
mov     [esp+4Ch+var_4C], 0
call    exit
```

```
loc_51B917:                             ; "--decoder  : Enter secret message mode"
mov     [esp+4Ch+var_4C], offset aDecoderEnterSe
call    puts
jmp     loc_51B7DF
```

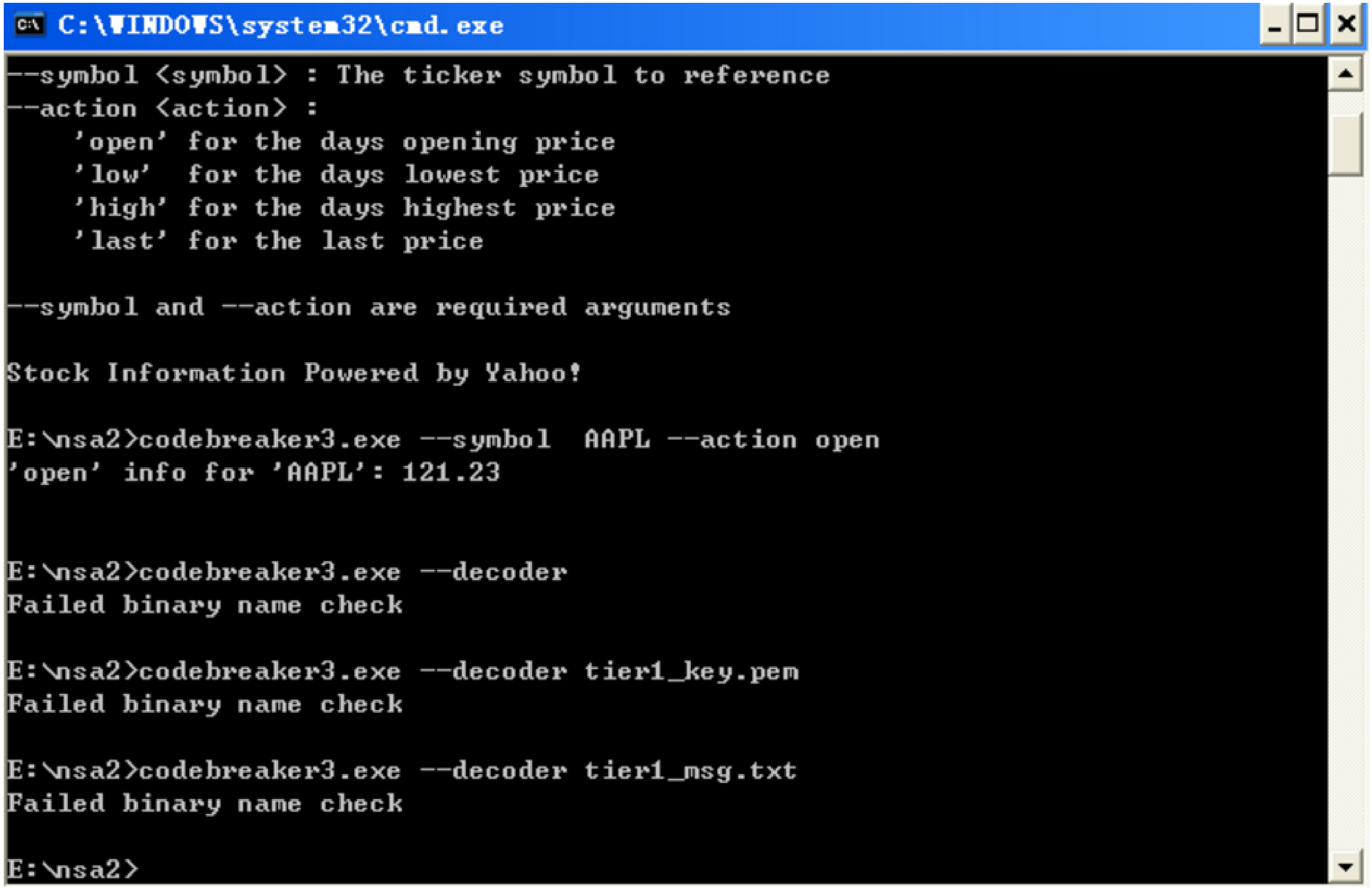
```
mov     [esp+4Ch+var_28], 1
jmp     loc_51B7DF
```

```
loc_!
mov
call
mov
call
```

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 1: Decrypt

- OK, let’s try “decoder” parameter:

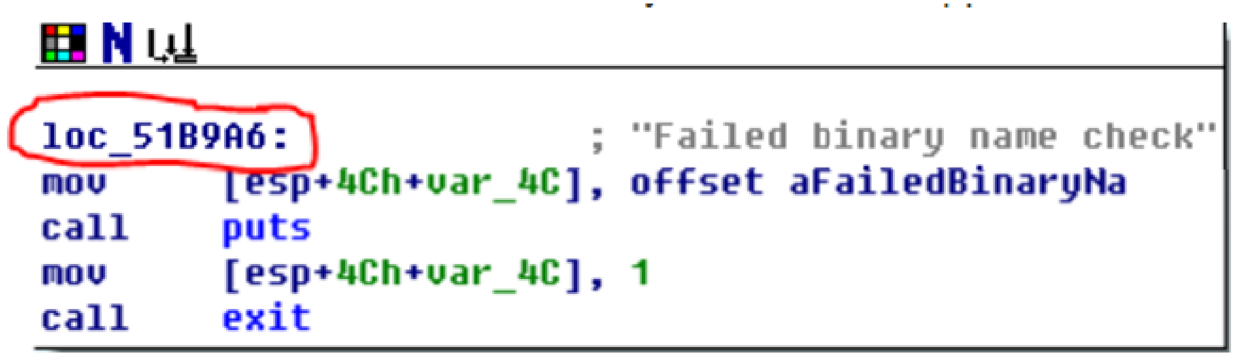


```
--symbol <symbol> : The ticker symbol to reference
--action <action> :
    'open' for the days opening price
    'low'  for the days lowest price
    'high' for the days highest price
    'last' for the last price

--symbol and --action are required arguments

Stock Information Powered by Yahoo!

E:\nsa2>codebreaker3.exe --symbol  AAPL --action open
'open' info for 'AAPL': 121.23


E:\nsa2>codebreaker3.exe --decoder
Failed binary name check

E:\nsa2>codebreaker3.exe --decoder tier1_key.pem
Failed binary name check

E:\nsa2>codebreaker3.exe --decoder tier1_msg.txt
Failed binary name check

E:\nsa2>
```

Adapted from content by Jiaming Li, NC State University, 2015

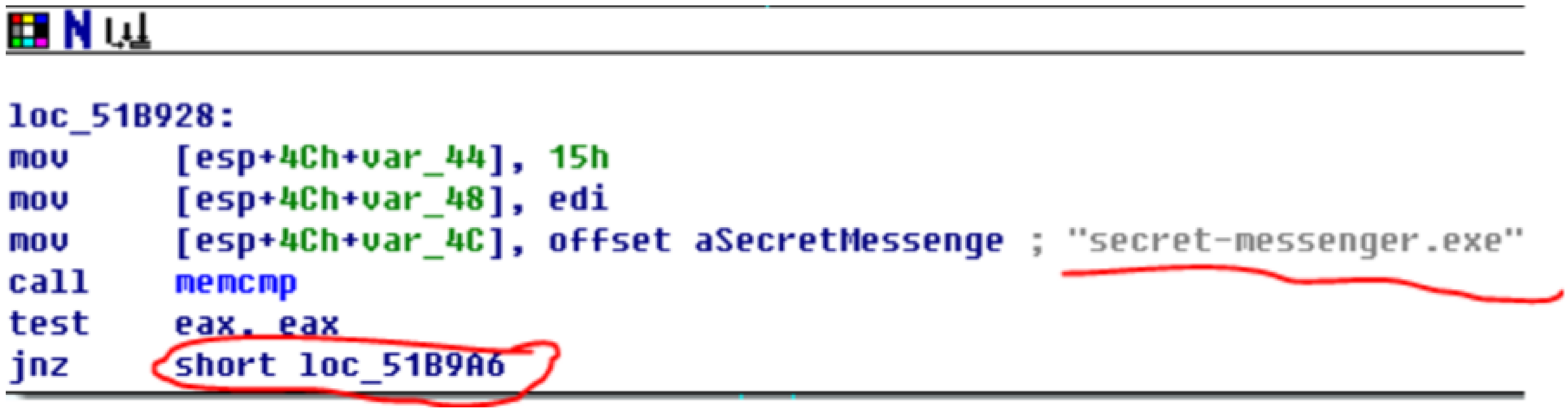
Codebreaker Task 1: Decrypt

- We need to find where “Failed binary name check” appears:



```
loc_51B9A6:                          ; "Failed binary name check"
mov      [esp+4Ch+var_4C], offset aFailedBinaryNa
call     puts
mov      [esp+4Ch+var_4C], 1
call     exit
```

- and this comes from:

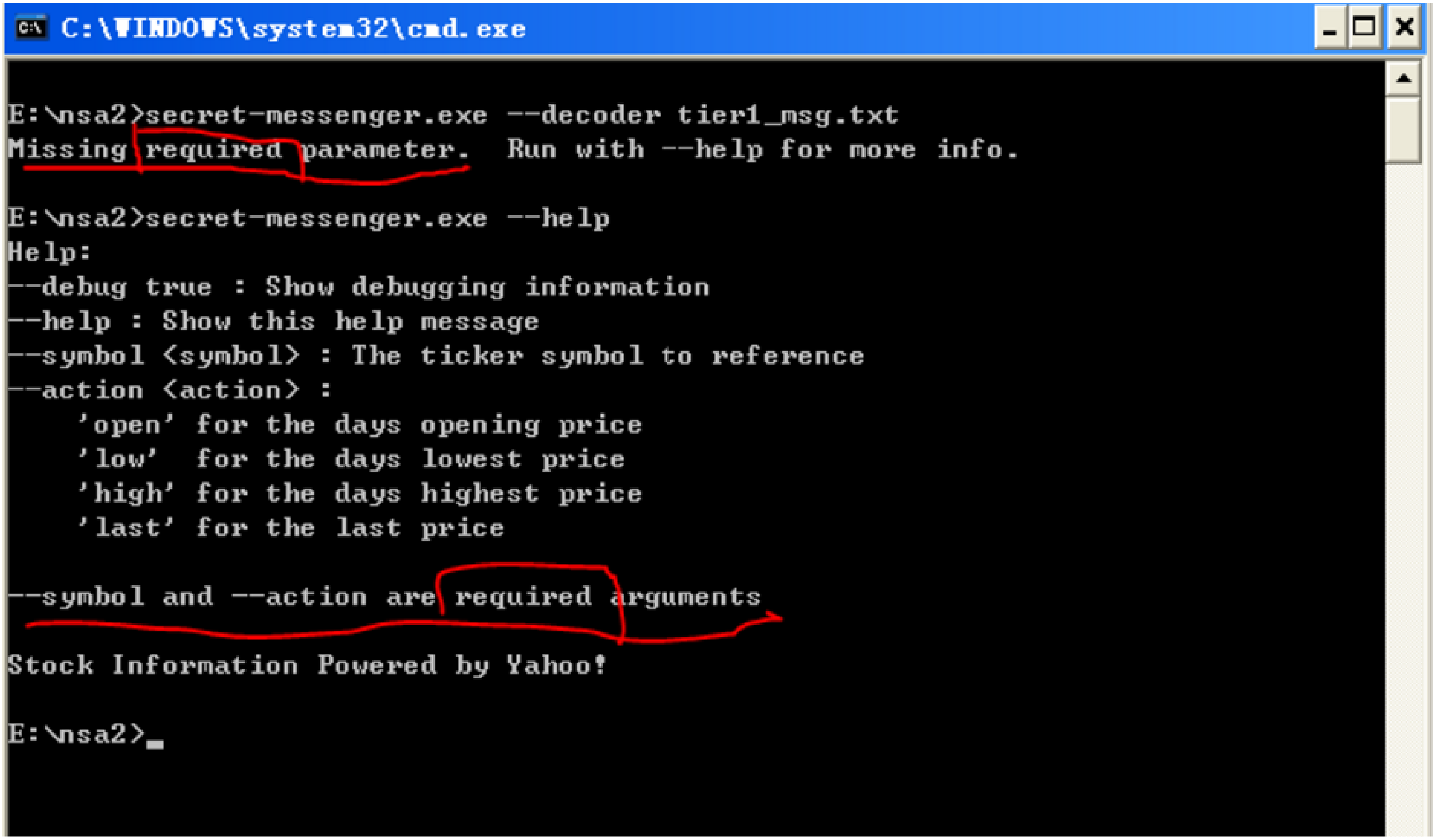


```
loc_51B928:
mov      [esp+4Ch+var_44], 15h
mov      [esp+4Ch+var_48], edi
mov      [esp+4Ch+var_4C], offset aSecretMessenge ; "secret-messenger.exe"
call     memcmp
test     eax, eax
jnz      short loc_51B9A6
```

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 1: Decrypt

- Then we change our program name to “secret-messenger.exe” and try again:



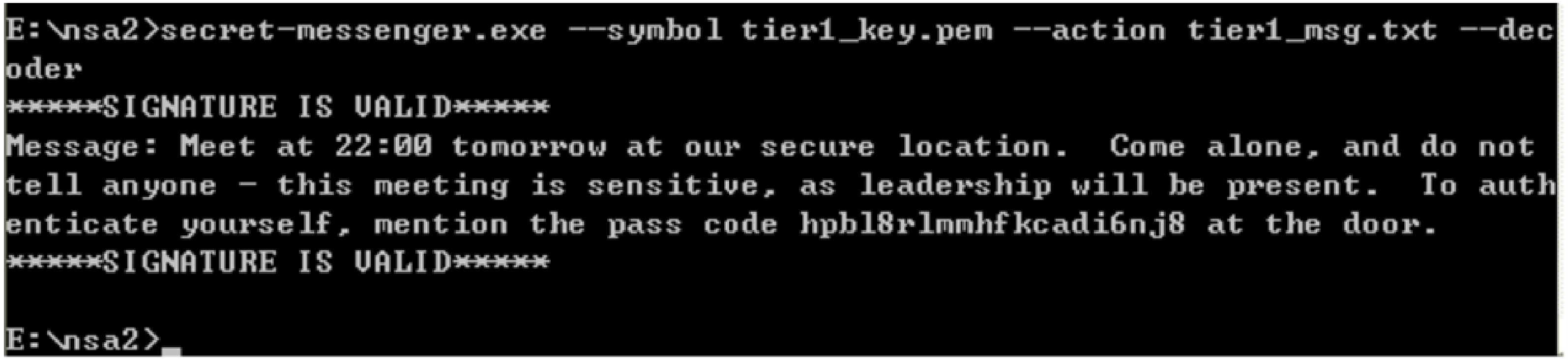
```
E:\nsa2>secret-messenger.exe --decoder tier1_msg.txt
Missing required parameter.  Run with --help for more info.

E:\nsa2>secret-messenger.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!

E:\nsa2>_
```

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 1: Decrypt

- Ideas?

- Let’s jam the stuff into symbol and action fields



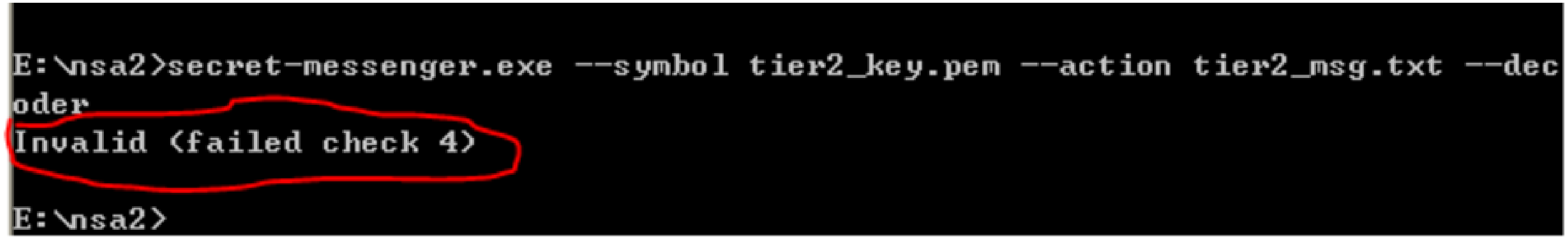
```
E:\nsa2>secret-messenger.exe --symbol tier1_key.pem --action tier1_msg.txt --dec
oder
*****SIGNATURE IS VALID*****
Message: Meet at 22:00 tomorrow at our secure location.  Come alone, and do not
tell anyone - this meeting is sensitive, as leadership will be present.  To auth
enticate yourself, mention the pass code hpbl8rlmmhfkcadi6nj8 at the door.
*****SIGNATURE IS VALID*****

E:\nsa2>_
```

MISSION ACCOMPLISHED

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 2: Bypass access limitation

- We've collected a **new message file** - this one to a different field operative **whose key we also have**.
- Each operative has their own decrypt tool, each tool will only decrypt content "addressed" to its owner.
- Need to defeat this access limitation to decrypt the message.

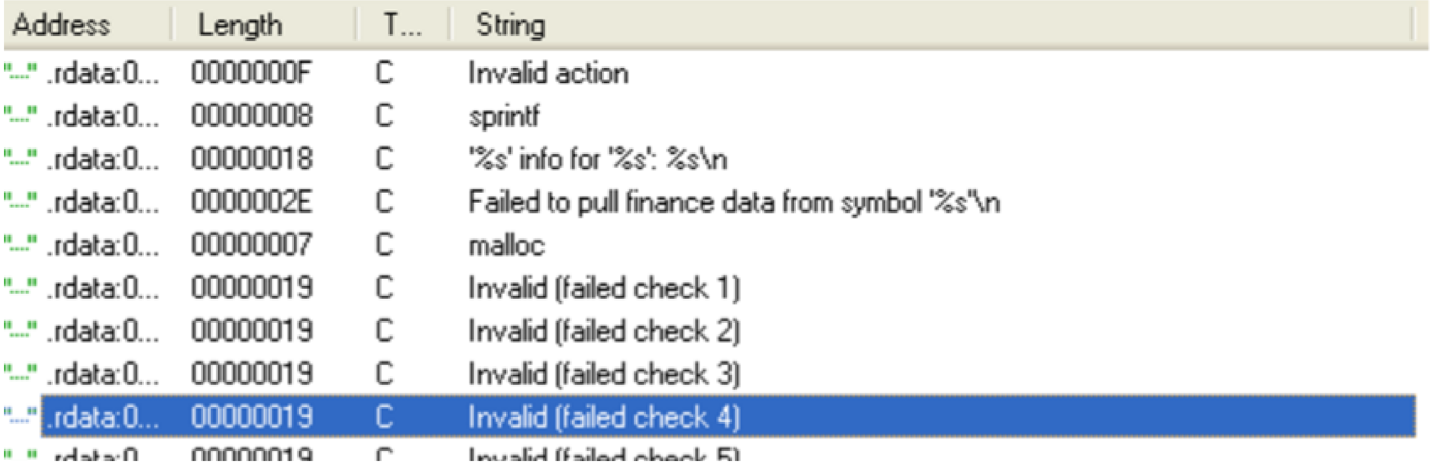


```
E:\nsa2>secret-messenger.exe --symbol tier2_key.pem --action tier2_msg.txt --dec
oder
Invalid (failed check 4)

E:\nsa2>
```

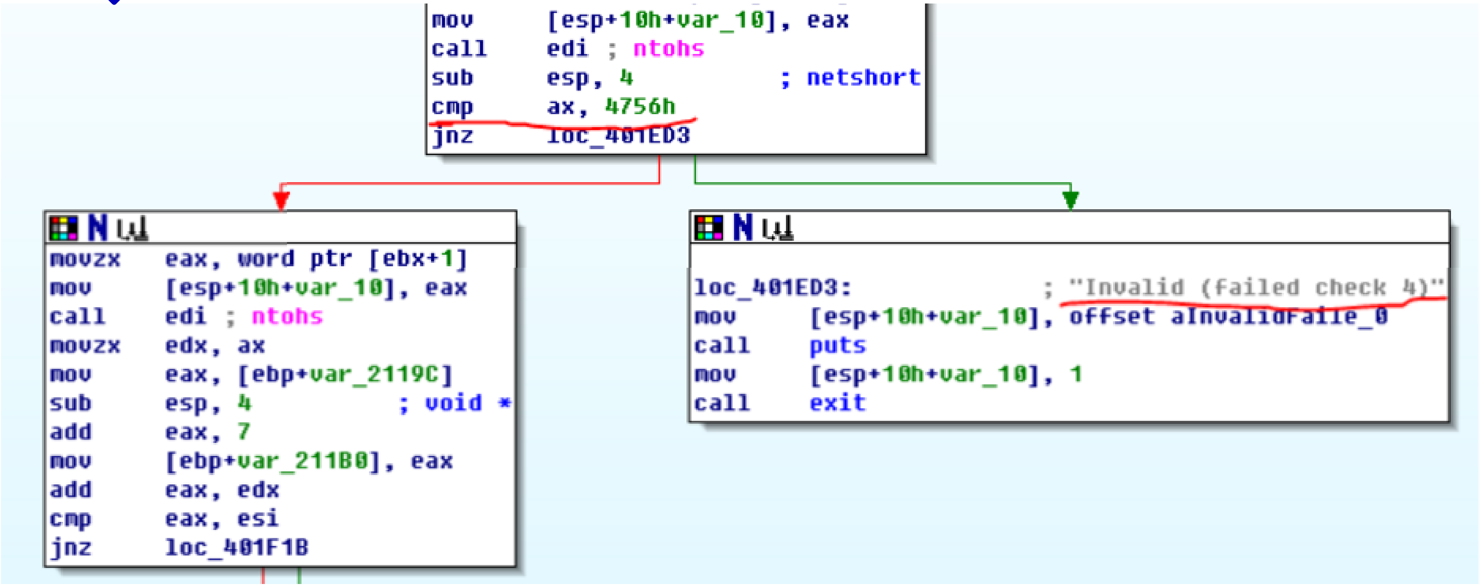
Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 2:
Bypass access limitation

- Let’s go back to IDA to find where this error appears:



Address	Length	T...	String
.rdata:0...	0000000F	C	Invalid action
.rdata:0...	00000008	C	sprintf
.rdata:0...	00000018	C	'%s' info for '%s': %s\n
.rdata:0...	0000002E	C	Failed to pull finance data from symbol '%s'\n
.rdata:0...	00000007	C	malloc
.rdata:0...	00000019	C	Invalid (failed check 1)
.rdata:0...	00000019	C	Invalid (failed check 2)
.rdata:0...	00000019	C	Invalid (failed check 3)
.rdata:0...	00000019	C	Invalid (failed check 4)
.rdata:0...	00000019	C	Invalid (failed check 5)



```
mov     [esp+10h+var_10], eax
call    edi ; ntohs
sub     esp, 4              ; netshort
cmp     ax, 4756h
jnz     loc_401ED3
```

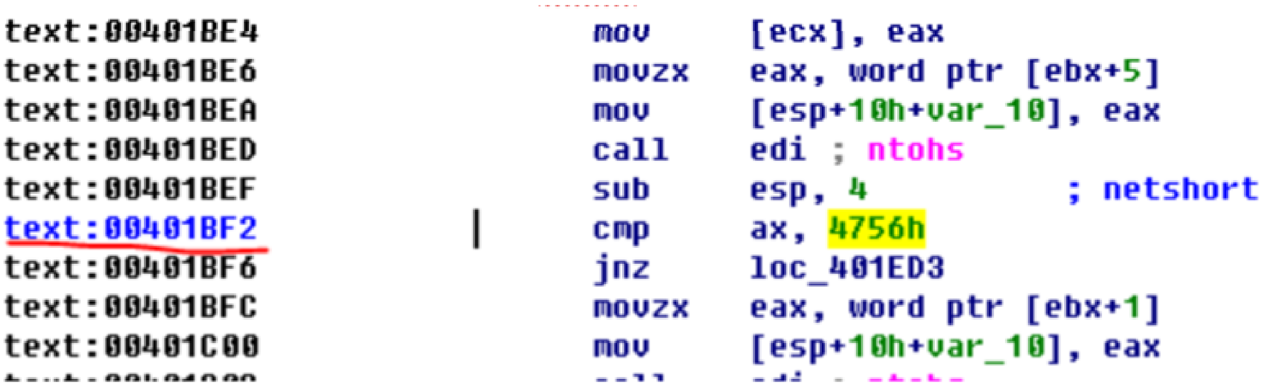
```
movzx   eax, word ptr [ebx+1]
mov     [esp+10h+var_10], eax
call    edi ; ntohs
movzx   edx, ax
mov     eax, [ebp+var_2119C]
sub     esp, 4              ; void *
add     eax, 7
mov     [ebp+var_211B0], eax
add     eax, edx
cmp     eax, esi
jnz     loc_401F1B
```

```
loc_401ED3:                 ; "Invalid (failed check 4)"
mov     [esp+10h+var_10], offset aInvalidFaile_0
call    puts
mov     [esp+10h+var_10], 1
call    exit
```

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 2: Bypass access limitation

- Note down the address of “cmp ax,4756h”, press SPACE:



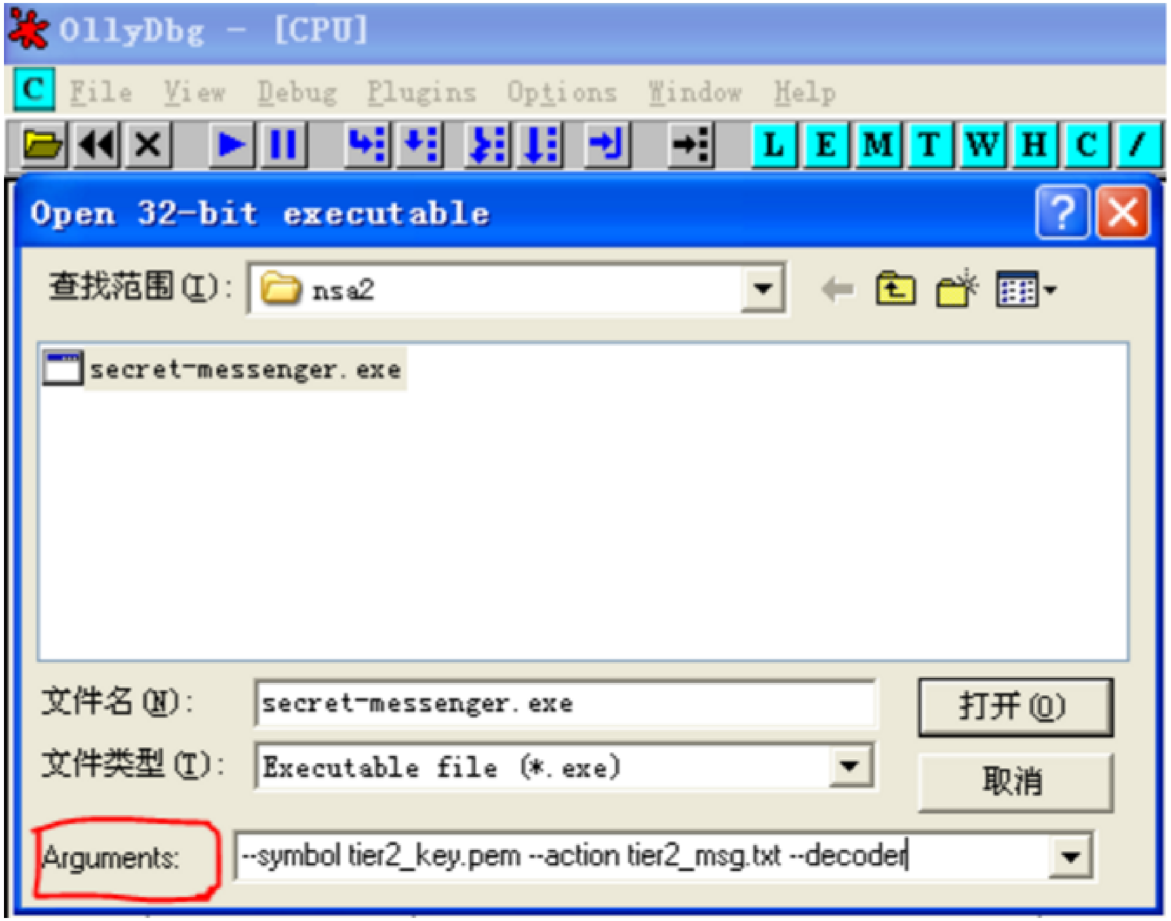
```
text:00401BE4                 mov     [ecx], eax
text:00401BE6                 movzx   eax, word ptr [ebx+5]
text:00401BEA                 mov     [esp+10h+var_10], eax
text:00401BED                 call    edi ; ntohs
text:00401BEF                 sub     esp, 4          ; netshort
text:00401BF2                 cmp     ax, 4756h
text:00401BF6                 jnz     loc_401ED3
text:00401BFC                 movzx   eax, word ptr [ebx+1]
text:00401C00                 mov     [esp+10h+var_10], eax
```

- How to test if this is the check?
- How to bypass the check?

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 2: Bypass access limitation

- In order to bypass this check as easily as possible, we can just modify the assembly code or change the specific flag during execution. Load the program with ollydbg:



文件名(N): secret-messenger.exe

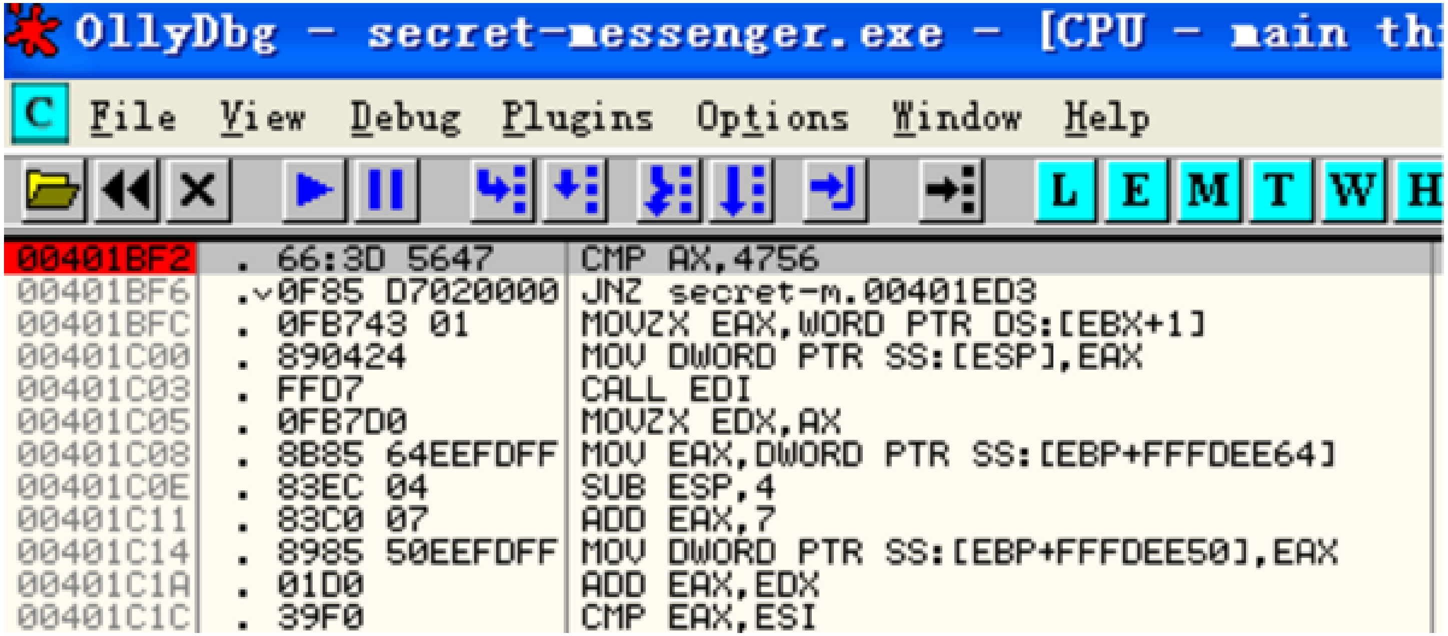
文件类型(T): Executable file (*.exe)

Arguments: --symbol tier2_key.pem --action tier2_msg.txt --decoder

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 2: Bypass access limitation

- Press CTRL+g go to the address 00401bf2, press F2 set breakpoint:



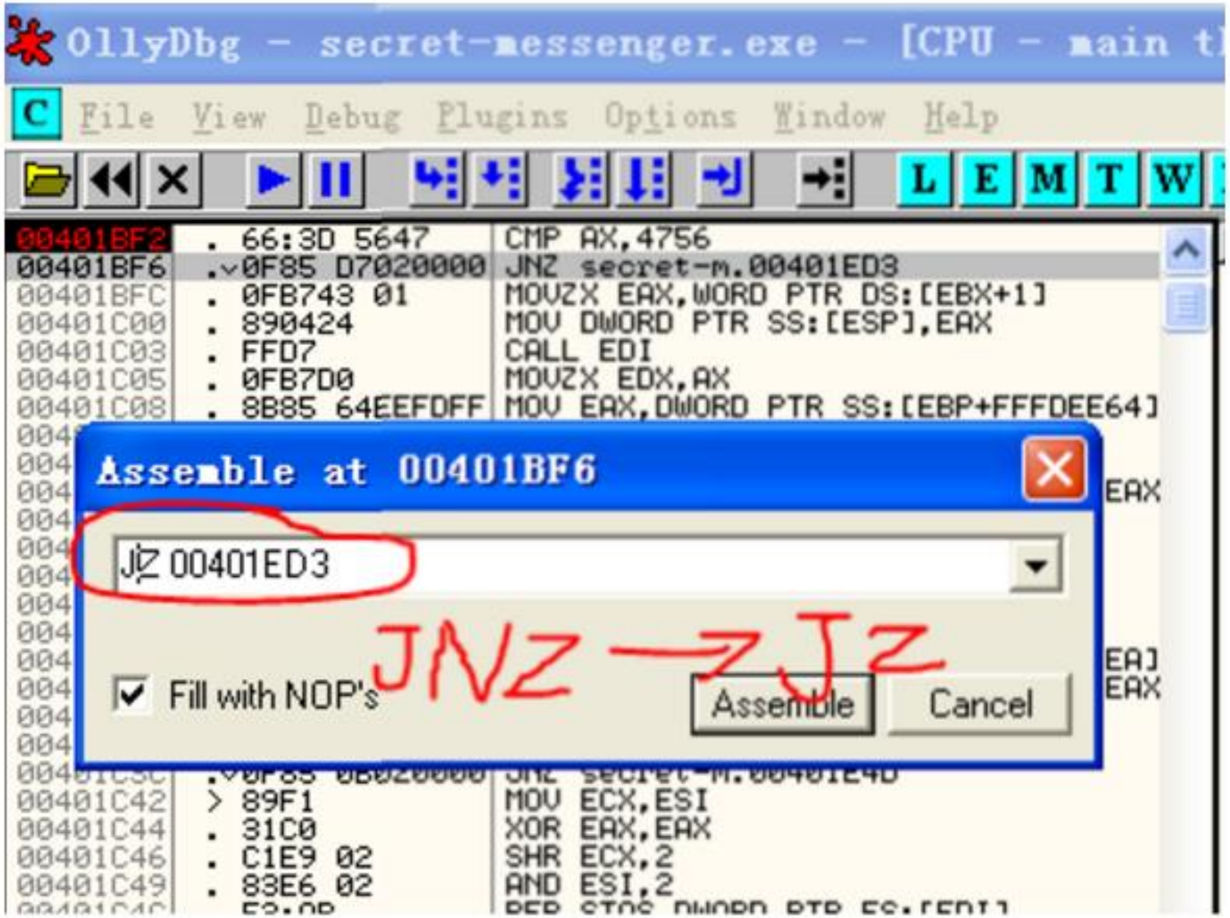
OllyDbg - secret-messenger.exe - [CPU - main th
C File View Debug Plugins Options Window Help
L E M T W H

```
00401BF2  .  66:3D 5647    CMP AX,4756
00401BF6  .v0F85 D7020000  JNZ secret-m.00401ED3
00401BFC  .  0FB743 01     MOVZX EAX,WORD PTR DS:[EBX+1]
00401C00  .  890424        MOV DWORD PTR SS:[ESP],EAX
00401C03  .  FFD7          CALL EDI
00401C05  .  0FB7D0        MOVZX EDX,AX
00401C08  .  8B85 64EEFDFF MOV EAX,DWORD PTR SS:[EBP+FFFDEE64]
00401C0E  .  83EC 04       SUB ESP,4
00401C11  .  83C0 07       ADD EAX,7
00401C14  .  8985 50EEFDFF MOV DWORD PTR SS:[EBP+FFFDEE50],EAX
00401C1A  .  01D0          ADD EAX,EDX
00401C1C  .  39F0          CMP EAX,ESI
```

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 2: Bypass access limitation

- Let’s run the program and it will stop at this breakpoint, press F8 to run one more step and we modify the conditional JUMP instruction manually:



```
00401BF2  .  66:3D 5647      CMP AX,4756
00401BF6  .v0F85 D7020000    JNZ secret-m.00401ED3
00401BFC  .  0FB743 01       MOVZX EAX,WORD PTR DS:[EBX+1]
00401C00  .  890424          MOV DWORD PTR SS:[ESP],EAX
00401C03  .  FFD7            CALL EDI
00401C05  .  0FB7D0          MOVZX EDX,AX
00401C08  .  8B85 64EEFDFF   MOV EAX,DWORD PTR SS:[EBP+FFFDEE64]
00401C3C  .v0F85 0B020000    JNZ secret-m.00401E4D
00401C42  >  89F1            MOV ECX,ESI
00401C44  .  31C0            XOR EAX,EAX
00401C46  .  C1E9 02         SHR ECX,2
00401C49  .  83E6 02         AND ESI,2
00401C4C  .  F3:AB           REP STOS DWORD PTR ES:[EDI]
```

Assemble at 00401BF6

JZ 00401ED3

☑ Fill with NOP's

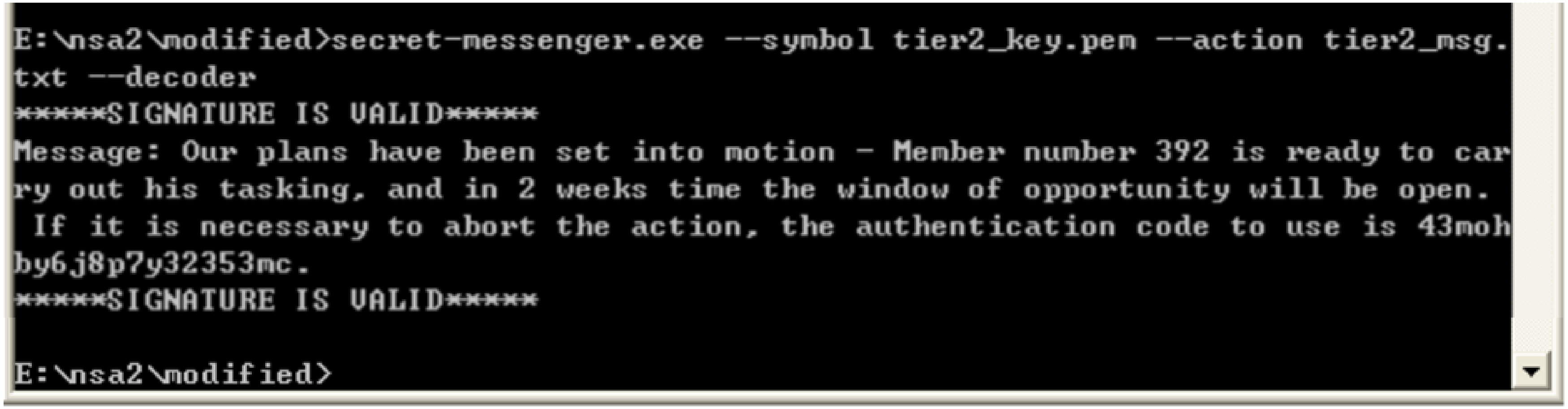
Assemble | Cancel

JNZ → JZ

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Task 2:
Bypass access limitation

- Then, right click → copy to executable→ all modification, so we just saved our new program, let’s try to run it:



```
E:\nsa2\modified>secret-messenger.exe --symbol tier2_key.pem --action tier2_msg.
txt --decoder
*****SIGNATURE IS VALID*****
Message: Our plans have been set into motion - Member number 392 is ready to car
ry 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 43moh
by6j8p7y32353mc.
*****SIGNATURE IS VALID*****

E:\nsa2\modified>
```

Adapted from content by Jiaming Li, NC State University, 2015

Codebreaker Tasks 3 and 4

- Task 3: Analyze decryption logic and develop a compatible encryption tool

- Task 4: Spoof messages so they appear to come from group leadership. Tell all recipients:
  “Leadership has arranged a meeting with the local authorities…Meet at the city police station at 18:00. Be discreet, and come unarmed as to not draw attention.” (LOL)

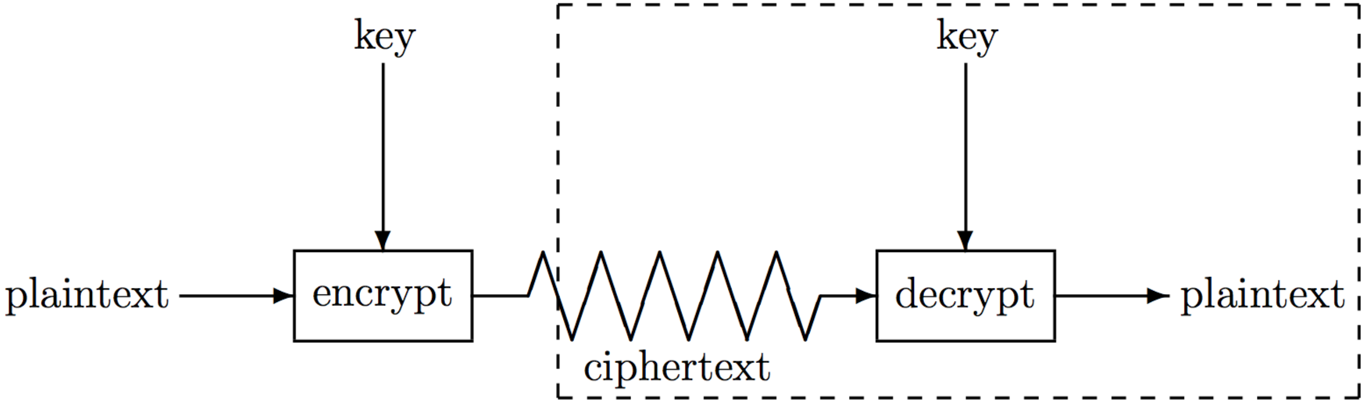
Adapted from content by Jiaming Li, NC State University, 2015

Anti-reverse engineering

- **Basics:**
  - Turn off debug symbols (-g)
  - Strip other symbols (e.g. “strip” tool on *NIX)
  - Consider static linking (no external calls to standard libraries to trace)
- **Anti-disassembly:**
  - Encrypted or self-modifying code
  - Code riddled with junk that is jumped over
    - Can especially confuse x86 assemblers due to variable-length instructions
- **Anti-debugging:**
  - Identify if debugger is in use (effects on real time, use of debug registers, etc.) and act differently
  - Use threads in complex ways to get less deterministic behavior
- **Tamper resistance:**
  - Hash parts of ones own code/data and verify periodically
  - The verification code is also code, though…
- **Obfuscation:**
  - Include lots of unreachable code to increase work the reverse engineer must do

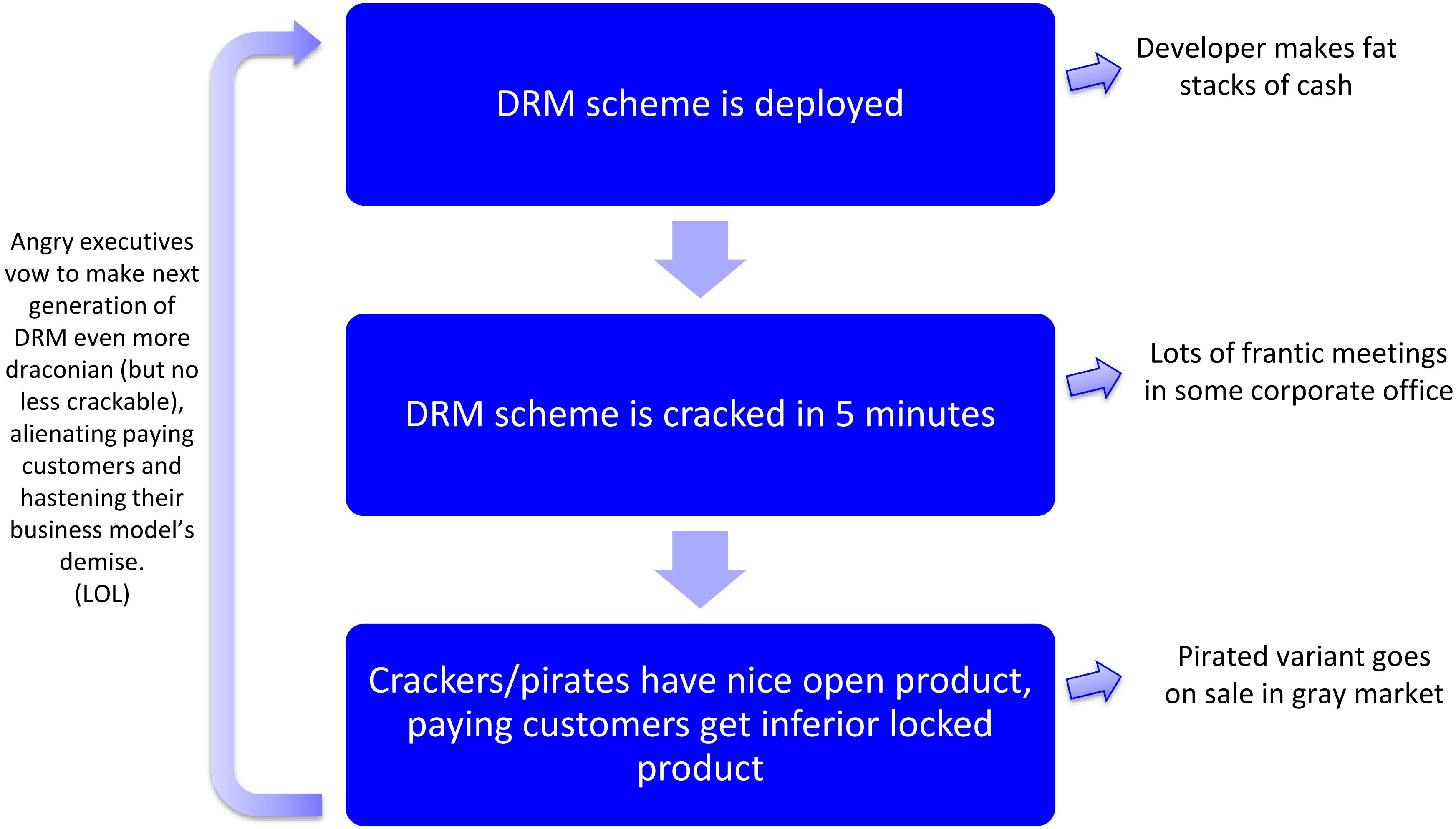
DRM: Digital Rights Management

- Attempt to restrict what users can do with *data they have* on a *computer they own*
- Almost every implementation looks like this:



  - Customer gets everything in the dashed box
  - Problem?

DRM deployment process



DRM scheme is deployed

Developer makes fat stacks of cash

DRM scheme is cracked in 5 minutes

Lots of frantic meetings in some corporate office

Crackers/pirates have nice open product, paying customers get inferior locked product

Pirated variant goes on sale in gray market

Angry executives vow to make next generation of DRM even more draconian (but no less crackable), alienating paying customers and hastening their business model's demise.
(LOL)