REVERSE ENGINEERING 1 (STATIC ANALYSIS)

COMP 6016

Malware Analysis

Dr Muhammad Hilmi Kamarudin

ACTIVITY - 1

- Divide into small groups
- 5 minutes
- 3 questions:
  - What do you understand by reverse engineering?
  - Why use it?
  - Given an executable, how do we find out what it does?

WHAT WILL WE LEARN TODAY?

- What is Reverse Engineering?
- Static vs Dynamic analysis
- RE and Assembly
- Disassembly, Decompilers
- IDA PRO
- Cutter
- Key Fishing `i.e. finding passwords`
- Reversing and Patching Java Bytecode

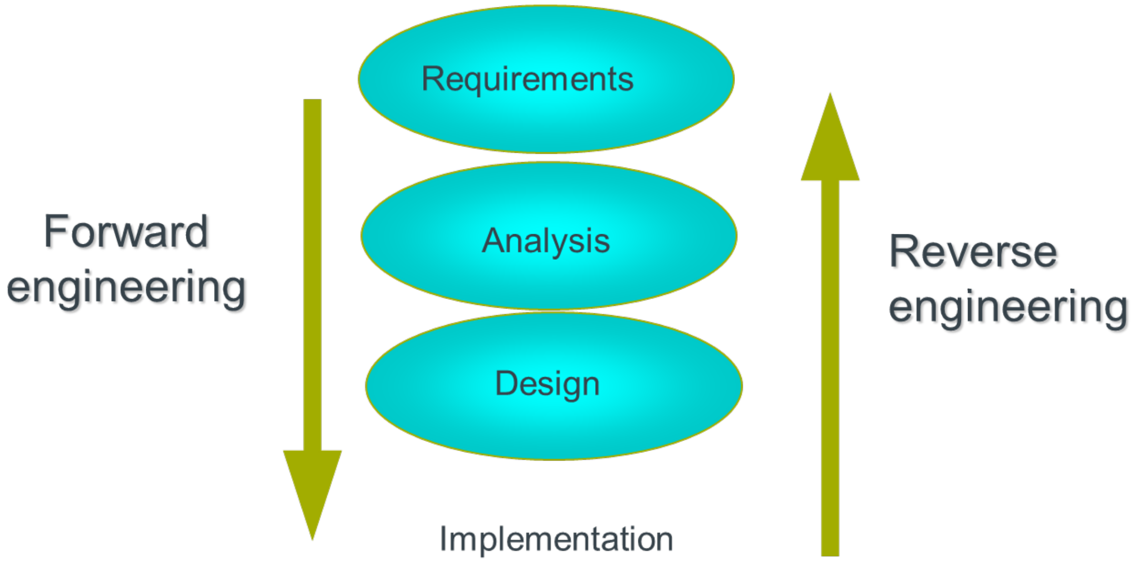
CODE ANALYSIS

- Given an executable, how do we find out what it does?
  - Find the program online.
    - Analyze source code to find clues.
    - Search for the name of the program.
  - **Perform source code review**.
  - Execute the program in a sandbox.
    - Some programs can break out of a sandbox / jail.

- Look for possible attacks: How?
  - **Denial of service:** initiate a memory leak, or infinite recursion, or infinite loop that will eventually cause the application to crash or hang.
  - **SPAM bot:** cause the application to start sending SPAM messages to remote hosts in the form of network packets.
  - **File modifier**: Add, delete, or replace files on the file system with malicious programs with the same file name.

REVERSE ENGINEERING

- ‘Trying to figure out the structure and behaviour of existing software by building general-level static and dynamic models’



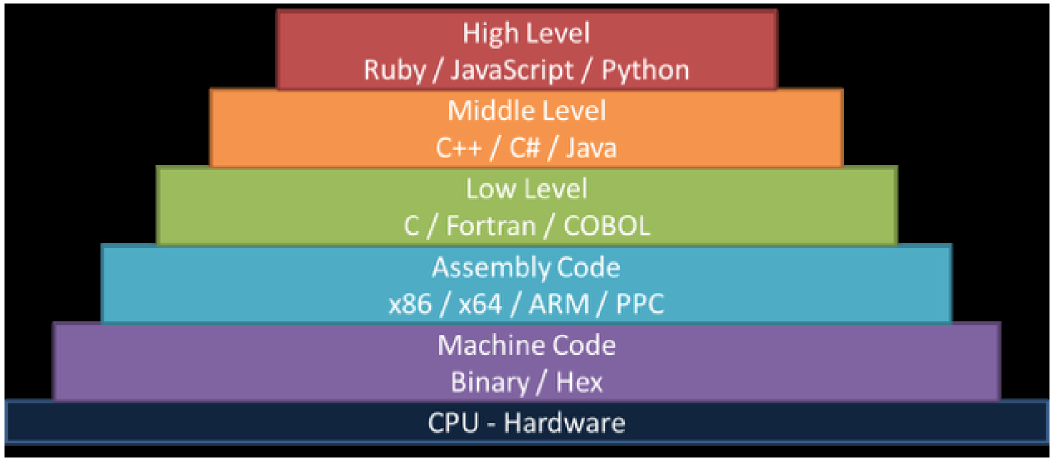
STATIC VS DYNAMIC ANALYSIS- OVERVIEW

- Static Analysis
  - Looking at the code, figure things out
  - Can be an easy way to find signatures –URLs, filenames, registry keys
  - Examines malware without running it - Not running the code!
  - A safer approach
  - Tools: VirusTotal, strings, a disassembler like IDA Pro
- Dynamic Analysis
  - Run the malware and monitor its effect - Examine the process during execution
  - Find new files made, processes created, - websites contacted, files downloaded/ executed, etc
  - Can see the values in real time
    - Registers, memory contents, etc.
  - Allows manipulation of the process
  - Shows you the effect the malware has on the system/network
  - Use a virtual machine and take snapshots
  - Tools: RegShot, Process Monitor, Process Hacker, CaptureBAT

Use Snapshots to save the state before detonating the malware in case it messes things up real bad!

REVERSE ENGINEERING AND ASSEMBLY

- Assembly is (usually) the highest level abstraction layer that can be reliably and consistently recovered
- This is why understanding assembly is so important for a malware analyst
  - x86, x64, MIPS, ARM, PowerPC, etc.



REVERSE ENGINEERING TOOLS (1)

- Disassemblers are usually the tool of choice for static analysis
  - Decodes binary machine code into a readable assembly language text
  - IDA Pro, Cutter, objdump, etc.
  - A good disassembler will have several useful features
    - Commenting
    - Renaming variables
    - Changing function prototypes
    - Coloring, grouping and renaming nodes
- Decompilers
  - Attempt to produce a high-level language source-code-like representation from a binary.
  - Never completely possible because
    - The compiler removes some information,
    - The compiler optimizes the code.
    - .Net (ILDasm, Remotesoft Salamander, Reflector for .Net)
    - Java (JODE, JAD)

REVERSE ENGINEERING TOOLS (2)

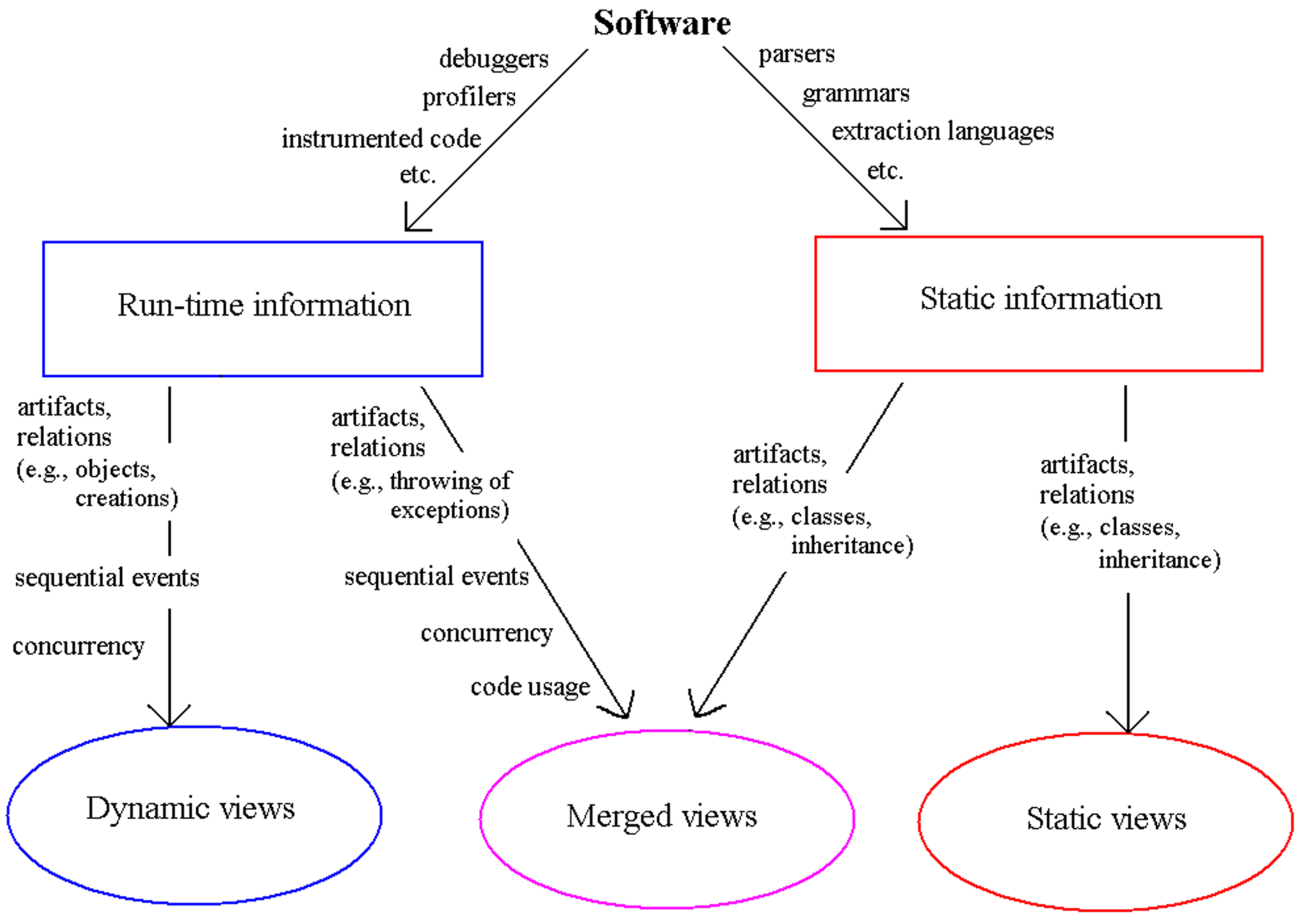
- Debuggers are used for dynamic analysis
  - Windows - WinDBG, Immunity, OllyDBG, IDA, Radare2
  - Linux – GDB, Radare2
  - A good debugger will have several useful features
    - Set breakpoints
    - Step into / over
    - Show loaded modules, SEH chain, etc.
    - Memory searching
- Hex Editor
  - To patch (make changes to) exe file

BASIC VS ADVANCED ANALYSIS

- Basic static analysis
  - View malware without looking at instructions – (Names of functions especially API functions, specific data strings like Names of constant strings, Names of directories, Identification of compiler
  - Quick and easy but fails for advanced malware and can miss important behavior
- Advanced static analysis
  - Reverse-engineering with a disassembler
  - Complex, requires understanding of assembly code

- Basic dynamic analysis
  - Easy but requires a safe test environment
  - Not effective on all malware (Virus Total)
- Advanced Dynamic Analysis
  - Run code in a debugger
  - Examines internal state of a running malicious executable

STATIC AND DYNAMIC ANALYSIS IS REQUIRED



DISASSEMBLY

- Malware on a disk is in **binary** form at the **machine code** level
- Disassembly converts the binary form to **assembly language**
- IDA Pro is the most popular disassembler (but is very expensive)
- Radare2 is an open source equivalent (but is complex to use)
- Cutter is a graphical wrapper to Radare2 (and is what we will use)

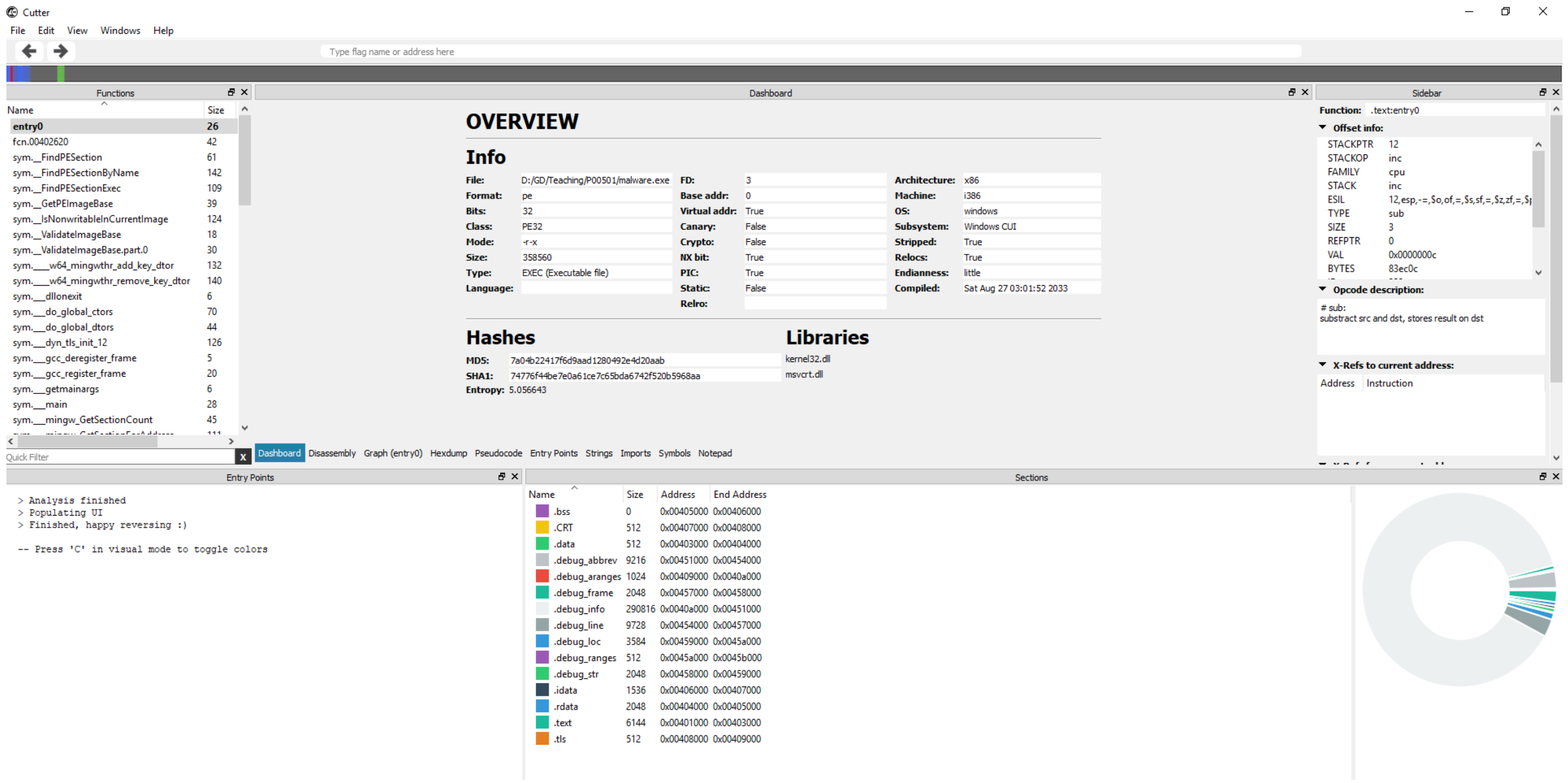
IDA PRO - OVERVIEW

- IDA Pro was originally designed as a powerful disassembler
- Supports 30+ processors
- It has since been broadened to include a built in debugger
- Designed for reverse engineers with quickness and robustness in mind
  - This sometimes makes the learning curve steep
- Extensible plugin architecture and scripting language

RADARE2- OVERVIEW

- Designed as an open source alternative to IDA Pro
- Runs on MS Windows, Linux, *BSD, Mac OS X, Solaris, Haiku, Android, iOS . Supports x86, x64, MIPS and ARM.
- Designed for reverse engineers with quickness and robustness in mind
  - This sometimes makes the learning curve steep
- Extensible plugin architecture aided by very wide range of possible scripting languages (Python, Javascript, Go).
- Collaborative analysis supported by in built web server

CUTTER - OVERVIEW



Cutter

File Edit View Windows Help

Type flag name or address here

Functions

Name	Size
entry0	26
fcn.00402620	42
sym.__FindPESection	61
sym.__FindPESectionByName	142
sym.__FindPESectionExec	109
sym.__GetPEImageBase	39
sym.__IsNonwritableInCurrentImage	124
sym.__ValidateImageBase	18
sym.__ValidateImageBase.part.0	30
sym.___w64_mingwthr_add_key_dtor	132
sym.___w64_mingwthr_remove_key_dtor	140
sym.___dllonexit	6
sym.___do_global_ctors	70
sym.___do_global_dtors	44
sym.___dyn_tls_init_12	126
sym.___gcc_deregister_frame	5
sym.___gcc_register_frame	20
sym.___getmainargs	6
sym.___main	28
sym.___mingw_GetSectionCount	45

Quick Filter

Dashboard

OVERVIEW

Info

File:	D:/GD/Teaching/P00501/malware.exe	**FD:**	3	**Architecture:**	x86
Format:	pe	**Base addr:**	0	**Machine:**	i386
Bits:	32	**Virtual addr:**	True	**OS:**	windows
Class:	PE32	**Canary:**	False	**Subsystem:**	Windows CUI
Mode:	-r-x	**Crypto:**	False	**Stripped:**	True
Size:	358560	**NX bit:**	True	**Relocs:**	True
Type:	EXEC (Executable file)	**PIC:**	True	**Endianness:**	little
Language:		**Static:**	False	**Compiled:**	Sat Aug 27 03:01:52 2033
		Relro:			

Hashes

MD5: 7a04b22417f6d9aad1280492e4d20aab
SHA1: 74776f44be7e0a61ce7c65bda6742f520b5968aa
Entropy: 5.056643

Libraries

kernel32.dll
msvcrt.dll

Dashboard Disassembly Graph (entry0) Hexdump Pseudocode Entry Points Strings Imports Symbols Notepad

Sidebar

Function: .text:entry0

Offset info:

STACKPTR	12
STACKOP	inc
FAMILY	cpu
STACK	inc
ESIL	12,esp,-=,$o,of,=,$s,sf,=,$z,zf,=,$
TYPE	sub
SIZE	3
REFPTR	0
VAL	0x0000000c
BYTES	83ec0c

Opcode description:

sub:
substract src and dst, stores result on dst

X-Refs to current address:

Address	Instruction

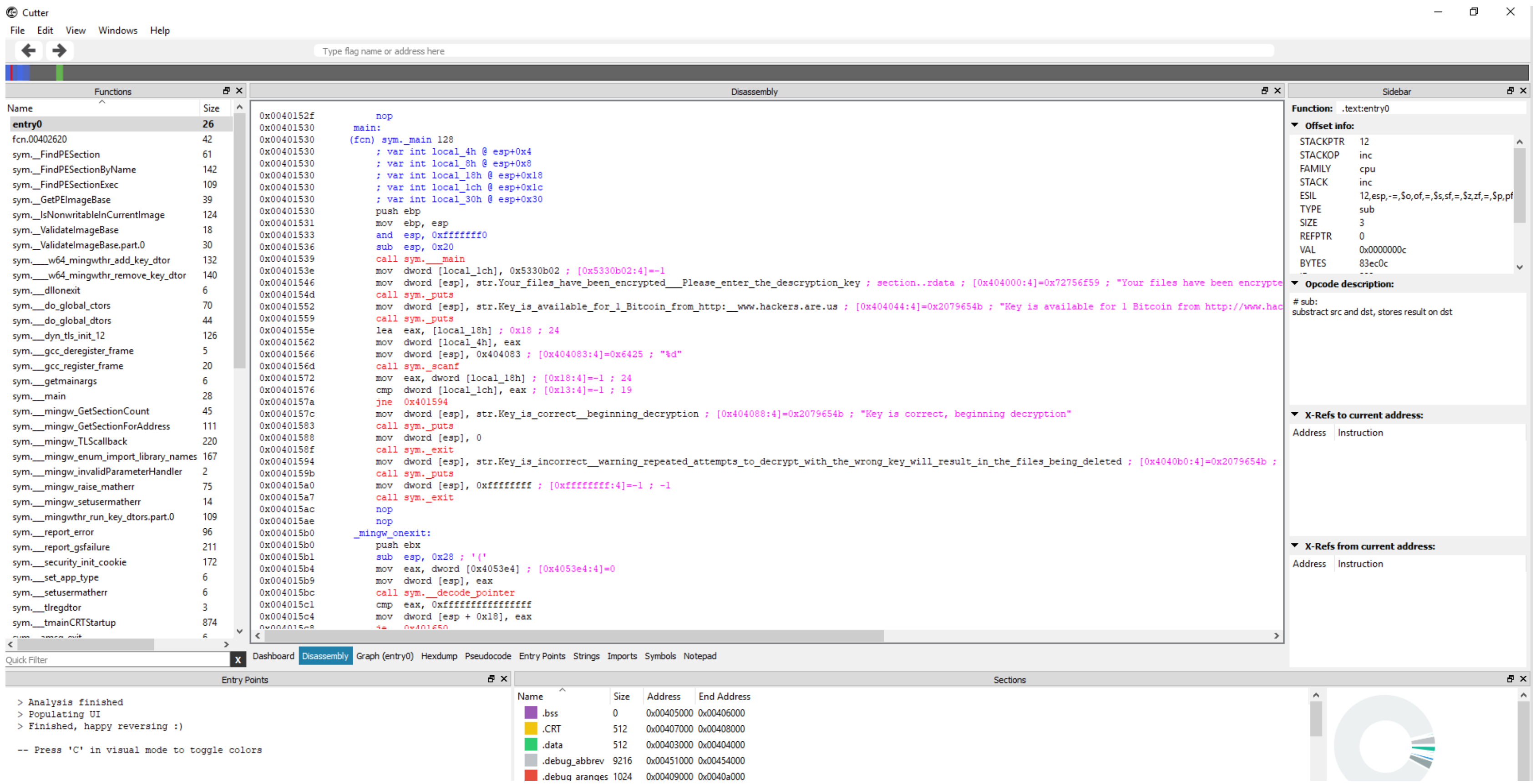
Entry Points

```
> Analysis finished
> Populating UI
> Finished, happy reversing :)

-- Press 'C' in visual mode to toggle colors
```

Sections

Name	Size	Address	End Address
🟪 .bss	0	0x00405000	0x00406000
🟨 .CRT	512	0x00407000	0x00408000
🟩 .data	512	0x00403000	0x00404000
⬜ .debug_abbrev	9216	0x00451000	0x00454000
🟥 .debug_aranges	1024	0x00409000	0x0040a000
🟩 .debug_frame	2048	0x00457000	0x00458000
⬜ .debug_info	290816	0x0040a000	0x00451000
⬜ .debug_line	9728	0x00454000	0x00457000
🟦 .debug_loc	3584	0x00459000	0x0045a000
🟪 .debug_ranges	512	0x0045a000	0x0045b000
🟩 .debug_str	2048	0x00458000	0x00459000
⬛ .idata	1536	0x00406000	0x00407000
🟦 .rdata	2048	0x00404000	0x00405000
🟩 .text	6144	0x00401000	0x00403000
🟧 .tls	512	0x00408000	0x00409000

CUTTER – DISASSEMBLER MODE



Name	Size
entry0	**26**
fcn.00402620	42
sym.__FindPESection	61
sym.__FindPESectionByName	142
sym.__FindPESectionExec	109
sym.__GetPEImageBase	39
sym.__IsNonwritableInCurrentImage	124
sym.__ValidateImageBase	18
sym.__ValidateImageBase.part.0	30
sym.____w64_mingwthr_add_key_dtor	132
sym.____w64_mingwthr_remove_key_dtor	140
sym.___dllonexit	6
sym.___do_global_ctors	70
sym.___do_global_dtors	44
sym.___dyn_tls_init_12	126
sym.___gcc_deregister_frame	5
sym.___gcc_register_frame	20
sym.___getmainargs	6
sym.___main	28
sym.___mingw_GetSectionCount	45
sym.___mingw_GetSectionForAddress	111
sym.___mingw_TLScallback	220
sym.___mingw_enum_import_library_names	167
sym.___mingw_invalidParameterHandler	2
sym.___mingw_raise_matherr	75
sym.___mingw_setusermatherr	14
sym.___mingwthr_run_key_dtors.part.0	109
sym.___report_error	96
sym.___report_gsfailure	211
sym.___security_init_cookie	172
sym.___set_app_type	6
sym.___setusermatherr	6
sym.___tlregdtor	3
sym.___tmainCRTStartup	874

```
0x0040152f          nop
0x00401530      main:
0x00401530     (fcn) sym._main 128
0x00401530          ; var int local_4h @ esp+0x4
0x00401530          ; var int local_8h @ esp+0x8
0x00401530          ; var int local_18h @ esp+0x18
0x00401530          ; var int local_1ch @ esp+0x1c
0x00401530          ; var int local_30h @ esp+0x30
0x00401530          push ebp
0x00401531          mov  ebp, esp
0x00401533          and  esp, 0xfffffff0
0x00401536          sub  esp, 0x20
0x00401539          call sym.___main
0x0040153e          mov  dword [local_1ch], 0x5330b02 ; [0x5330b02:4]=-1
0x00401546          mov  dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [0x404000:4]=0x72756f59 ; "Your files have been encrypte
0x0040154d          call sym._puts
0x00401552          mov  dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654b ; "Key is available for 1 Bitcoin from http://www.hac
0x00401559          call sym._puts
0x0040155e          lea  eax, [local_18h] ; 0x18 ; 24
0x00401562          mov  dword [local_4h], eax
0x00401566          mov  dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
0x0040156d          call sym._scanf
0x00401572          mov  eax, dword [local_18h] ; [0x18:4]=-1 ; 24
0x00401576          cmp  dword [local_1ch], eax ; [0x13:4]=-1 ; 19
0x0040157a          jne  0x401594
0x0040157c          mov  dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, beginning decryption"
0x00401583          call sym._puts
0x00401588          mov  dword [esp], 0
0x0040158f          call sym._exit
0x00401594          mov  dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decrypt_with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x4040b0:4]=0x2079654b ;
0x0040159b          call sym._puts
0x004015a0          mov  dword [esp], 0xffffffff ; [0xffffffff:4]=-1 ; -1
0x004015a7          call sym._exit
0x004015ac          nop
0x004015ae          nop
0x004015b0      _mingw_onexit:
0x004015b0          push ebx
0x004015b1          sub  esp, 0x28 ; '('
0x004015b4          mov  eax, dword [0x4053e4] ; [0x4053e4:4]=0
0x004015b9          mov  dword [esp], eax
0x004015bc          call sym.__decode_pointer
0x004015c1          cmp  eax, 0xffffffffffffffff
0x004015c4          mov  dword [esp + 0x18], eax
0x004015c8          je   0x401650
```

Function: .text:entry0

Offset info:

STACKPTR	12
STACKOP	inc
FAMILY	cpu
STACK	inc
ESIL	12,esp,-=,$o,of,=,$s,sf,=,$z,zf,=,$p,pf
TYPE	sub
SIZE	3
REFPTR	0
VAL	0x0000000c
BYTES	83ec0c

Opcode description:

# sub:
substract src and dst, stores result on dst

X-Refs to current address:

Address	Instruction

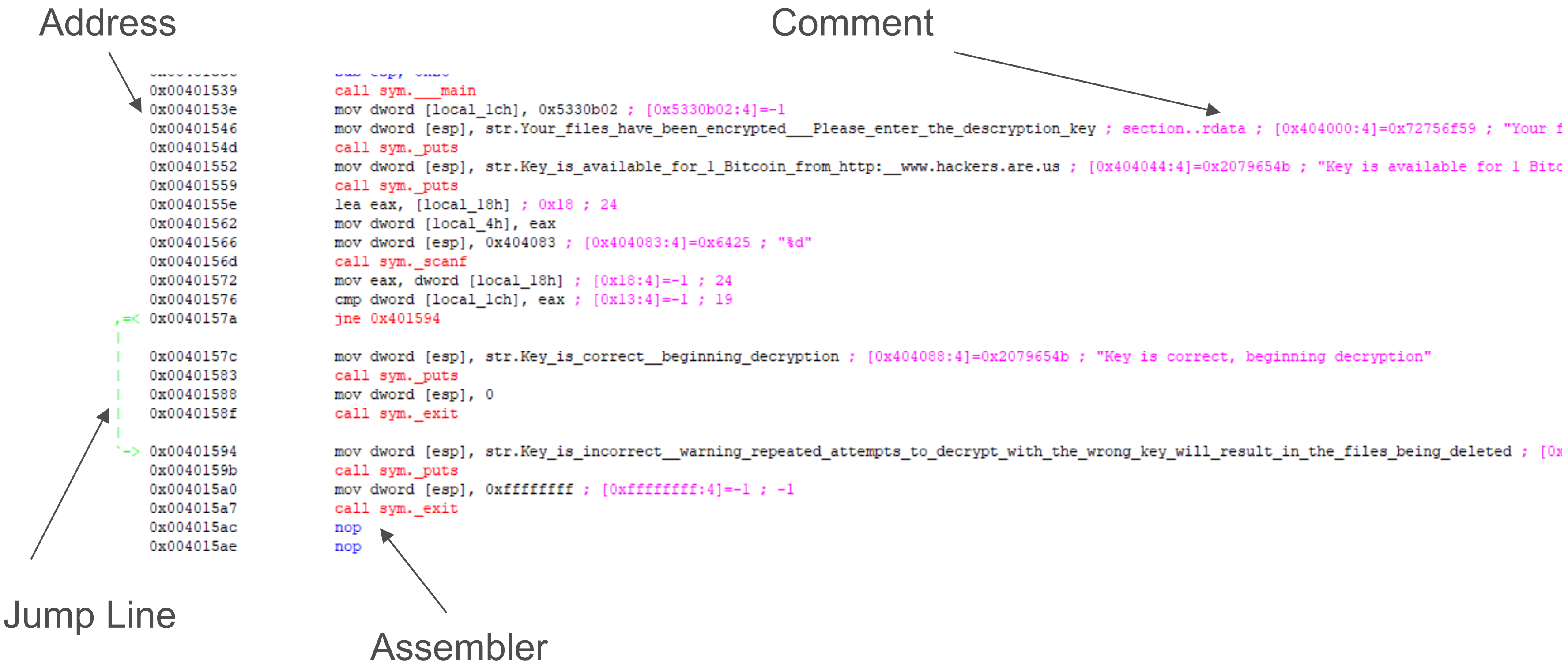
X-Refs from current address:

Address	Instruction

```
> Analysis finished
> Populating UI
> Finished, happy reversing :)

-- Press 'C' in visual mode to toggle colors
```

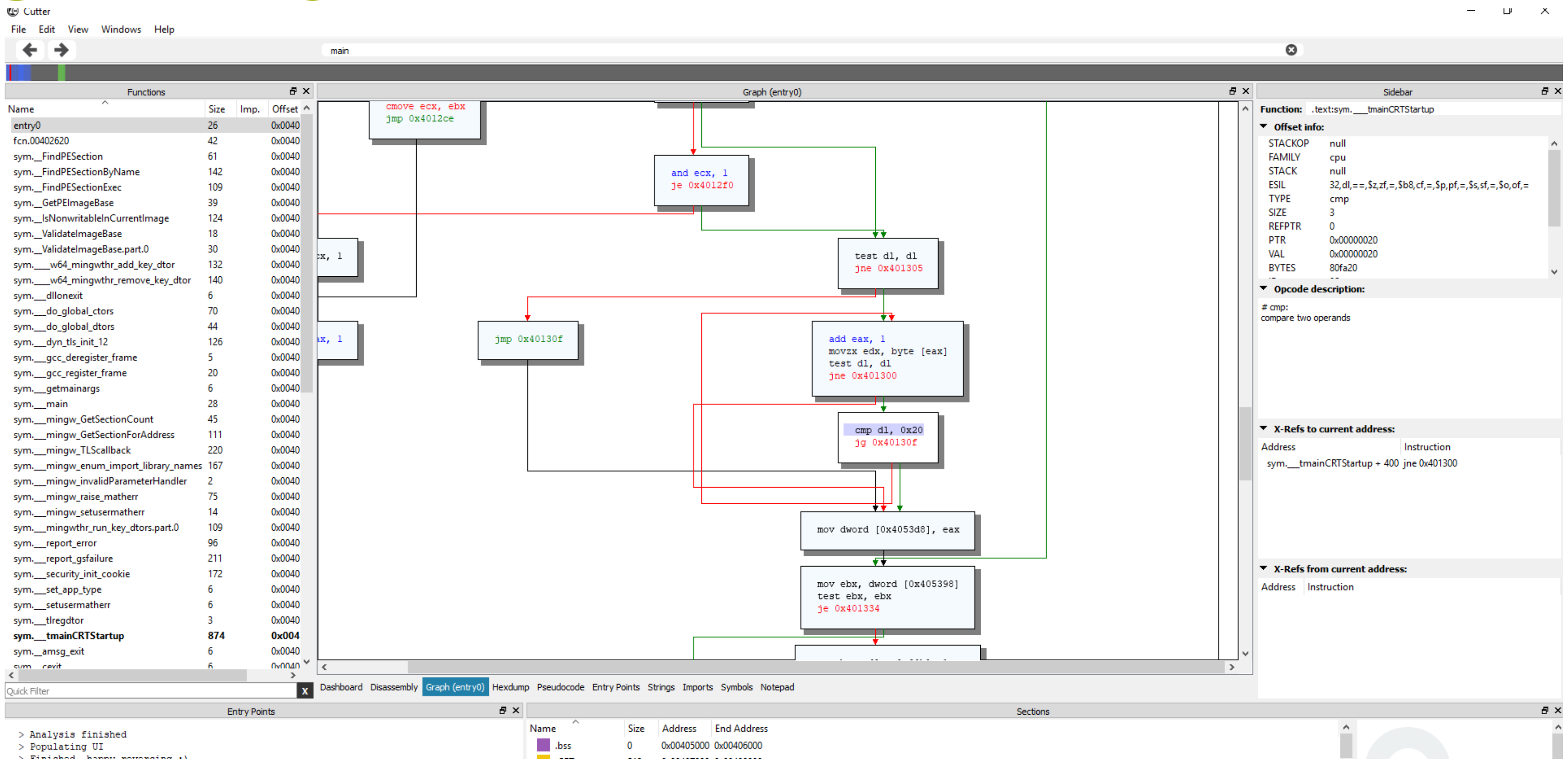
Name	Size	Address	End Address
🟪 .bss	0	0x00405000	0x00406000
🟨 .CRT	512	0x00407000	0x00408000
🟩 .data	512	0x00403000	0x00404000
⬜ .debug_abbrev	9216	0x00451000	0x00454000
🟥 .debug_aranges	1024	0x00409000	0x0040a000

CUTTER – DISASSEMBLER MODE



```
                 [illegible]
                 0x00401539           call sym.___main
                 0x0040153e           mov dword [local_1ch], 0x5330b02 ; [0x5330b02:4]=-1
                 0x00401546           mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [0x404000:4]=0x72756f59 ; "Your f
                 0x0040154d           call sym._puts
                 0x00401552           mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654b ; "Key is available for 1 Bitc
                 0x00401559           call sym._puts
                 0x0040155e           lea eax, [local_18h] ; 0x18 ; 24
                 0x00401562           mov dword [local_4h], eax
                 0x00401566           mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
                 0x0040156d           call sym._scanf
                 0x00401572           mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
                 0x00401576           cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
              ,=< 0x0040157a           jne 0x401594
              |
              |   0x0040157c           mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, beginning decryption"
              |   0x00401583           call sym._puts
              |   0x00401588           mov dword [esp], 0
              |   0x0040158f           call sym._exit
              |
              `-> 0x00401594           mov dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decrypt_with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x
                 0x0040159b           call sym._puts
                 0x004015a0           mov dword [esp], 0xffffffff ; [0xffffffff:4]=-1 ; -1
                 0x004015a7           call sym._exit
                 0x004015ac           nop
                 0x004015ae           nop
```

CUTTER - GRAPH


Cutter
File Edit View Windows Help
main
Functions
Graph (entry0)
Sidebar
Dashboard Disassembly Graph (entry0) Hexdump Pseudocode Entry Points Strings Imports Symbols Notepad
Quick Filter
Entry Points
Sections

Name	Size	Imp.	Offset
entry0	26		0x0040
fcn.00402620	42		0x0040
sym.__FindPESection	61		0x0040
sym.__FindPESectionByName	142		0x0040
sym.__FindPESectionExec	109		0x0040
sym.__GetPEImageBase	39		0x0040
sym.__IsNonwritableInCurrentImage	124		0x0040
sym.__ValidateImageBase	18		0x0040
sym.__ValidateImageBase.part.0	30		0x0040
sym.___w64_mingwthr_add_key_dtor	132		0x0040
sym.___w64_mingwthr_remove_key_dtor	140		0x0040
sym.___dllonexit	6		0x0040
sym.___do_global_ctors	70		0x0040
sym.___do_global_dtors	44		0x0040
sym.___dyn_tls_init_12	126		0x0040
sym.___gcc_deregister_frame	5		0x0040
sym.___gcc_register_frame	20		0x0040
sym.___getmainargs	6		0x0040
sym.___main	28		0x0040
sym.___mingw_GetSectionCount	45		0x0040
sym.___mingw_GetSectionForAddress	111		0x0040
sym.___mingw_TLScallback	220		0x0040
sym.___mingw_enum_import_library_names	167		0x0040
sym.___mingw_invalidParameterHandler	2		0x0040
sym.___mingw_raise_matherr	75		0x0040
sym.___mingw_setusermatherr	14		0x0040
sym.___mingwthr_run_key_dtors.part.0	109		0x0040
sym.___report_error	96		0x0040
sym.___report_gsfailure	211		0x0040
sym.___security_init_cookie	172		0x0040
sym.___set_app_type	6		0x0040
sym.___setusermatherr	6		0x0040
sym.___tlregdtor	3		0x0040
sym.___tmainCRTStartup	**874**		**0x004**
sym.__amsg_exit	6		0x0040

```
cmove ecx, ebx
jmp 0x4012ce

and ecx, 1
je 0x4012f0

x, 1

test dl, dl
jne 0x401305

jmp 0x40130f

x, 1

add eax, 1
movzx edx, byte [eax]
test dl, dl
jne 0x401300

cmp dl, 0x20
jg 0x40130f

mov dword [0x4053d8], eax

mov ebx, dword [0x405398]
test ebx, ebx
je 0x401334
```

Function: .text:sym.___tmainCRTStartup

Offset info:

STACKOP	null
FAMILY	cpu
STACK	null
ESIL	32,dl,==,$z,zf,=,$b8,cf,=,$p,pf,=,$s,sf,=,$o,of,=
TYPE	cmp
SIZE	3
REFPTR	0
PTR	0x00000020
VAL	0x00000020
BYTES	80fa20

Opcode description:

cmp:
compare two operands

X-Refs to current address:

Address	Instruction
sym.___tmainCRTStartup + 400	jne 0x401300

X-Refs from current address:

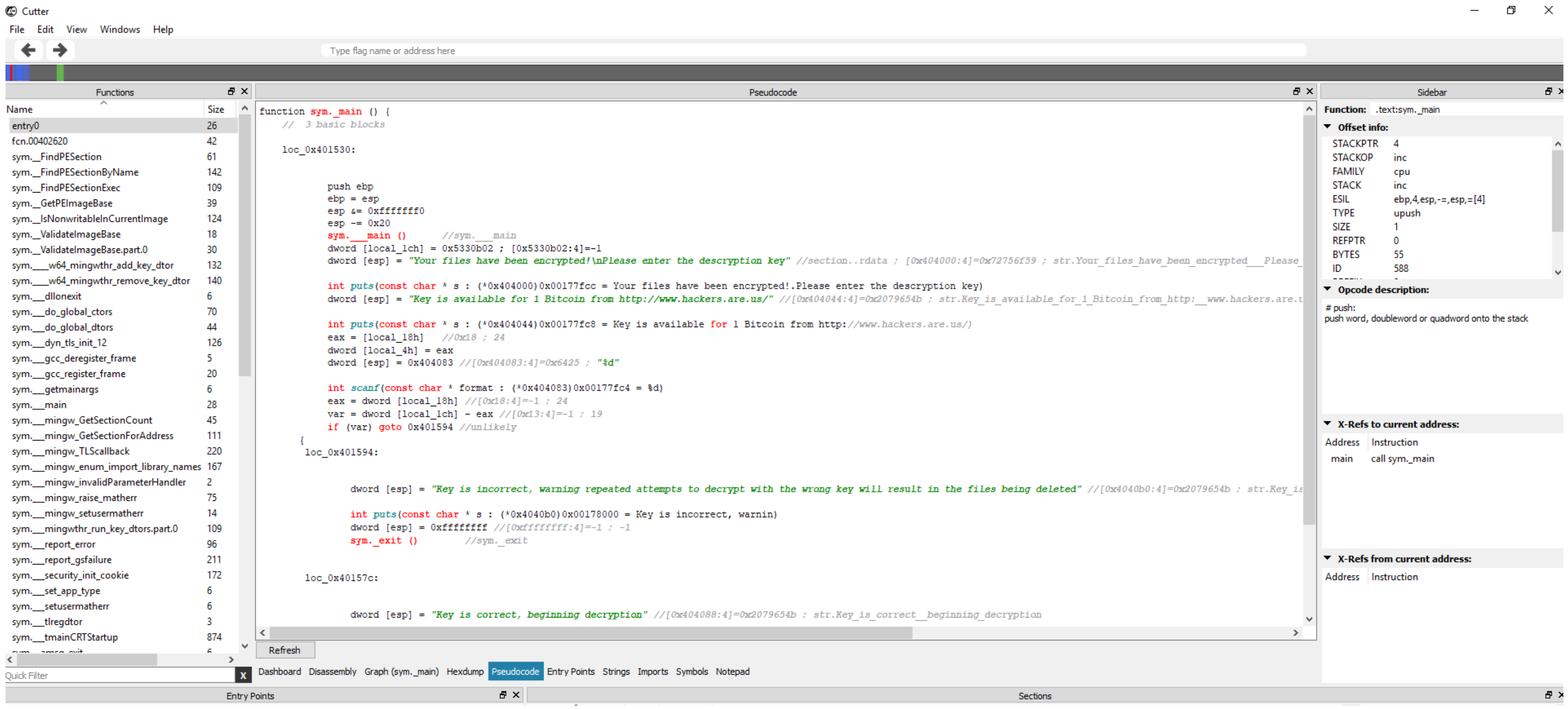
Address	Instruction

```
> Analysis finished
> Populating UI
```

Name	Size	Address	End Address
.bss	0	0x00405000	0x00406000

- Colors
  - Red — Conditional jump not taken
  - Green — Conditional jump taken
  - Black — Unconditional jump (or continuation)

CUTTER – PSUEDO-CODE



Name	Size
entry0	26
fcn.00402620	42
sym.__FindPESection	61
sym.__FindPESectionByName	142
sym.__FindPESectionExec	109
sym.__GetPEImageBase	39
sym.__IsNonwritableInCurrentImage	124
sym.__ValidateImageBase	18
sym.__ValidateImageBase.part.0	30
sym.___w64_mingwthr_add_key_dtor	132
sym.___w64_mingwthr_remove_key_dtor	140
sym.___dllonexit	6
sym.___do_global_ctors	70
sym.___do_global_dtors	44
sym.___dyn_tls_init_12	126
sym.___gcc_deregister_frame	5
sym.___gcc_register_frame	20
sym.___getmainargs	6
sym.___main	28
sym.___mingw_GetSectionCount	45
sym.___mingw_GetSectionForAddress	111
sym.___mingw_TLScallback	220
sym.___mingw_enum_import_library_names	167
sym.___mingw_invalidParameterHandler	2
sym.___mingw_raise_matherr	75
sym.___mingw_setusermatherr	14
sym.___mingwthr_run_key_dtors.part.0	109
sym.___report_error	96
sym.___report_gsfailure	211
sym.___security_init_cookie	172
sym.___set_app_type	6
sym.___setusermatherr	6
sym.___tlregdtor	3
sym.___tmainCRTStartup	874

```
function sym._main () {
    //  3 basic blocks

    loc_0x401530:


          push ebp
          ebp = esp
          esp &= 0xfffffff0
          esp -= 0x20
          sym.___main ()       //sym.___main
          dword [local_1ch] = 0x5330b02 ; [0x5330b02:4]=-1
          dword [esp] = "Your files have been encrypted!\nPlease enter the descryption key" //section..rdata ; [0x404000:4]=0x72756f59 ; str.Your_files_have_been_encrypted___Please_

          int puts(const char * s : (*0x404000)0x00177fcc = Your files have been encrypted!.Please enter the descryption key)
          dword [esp] = "Key is available for 1 Bitcoin from http://www.hackers.are.us/" //[0x404044:4]=0x2079654b ; str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.u

          int puts(const char * s : (*0x404044)0x00177fc8 = Key is available for 1 Bitcoin from http://www.hackers.are.us/)
          eax = [local_18h]   //0x18 ; 24
          dword [local_4h] = eax
          dword [esp] = 0x404083 //[0x404083:4]=0x6425 ; "%d"

          int scanf(const char * format : (*0x404083)0x00177fc4 = %d)
          eax = dword [local_18h] //[0x18:4]=-1 ; 24
          var = dword [local_1ch] - eax //[0x13:4]=-1 ; 19
          if (var) goto 0x401594 //unlikely
       {
     loc_0x401594:


              dword [esp] = "Key is incorrect, warning repeated attempts to decrypt with the wrong key will result in the files being deleted" //[0x4040b0:4]=0x2079654b ; str.Key_is

              int puts(const char * s : (*0x4040b0)0x00178000 = Key is incorrect, warnin)
              dword [esp] = 0xffffffff //[0xffffffff:4]=-1 ; -1
              sym._exit ()         //sym._exit


     loc_0x40157c:


              dword [esp] = "Key is correct, beginning decryption" //[0x404088:4]=0x2079654b ; str.Key_is_correct__beginning_decryption
```

Function: .text:sym._main

Offset info:

STACKPTR	4
STACKOP	inc
FAMILY	cpu
STACK	inc
ESIL	ebp,4,esp,-=,esp,=[4]
TYPE	upush
SIZE	1
REFPTR	0
BYTES	55
ID	588

Opcode description:

push:
push word, doubleword or quadword onto the stack

X-Refs to current address:

Address	Instruction
main	call sym._main

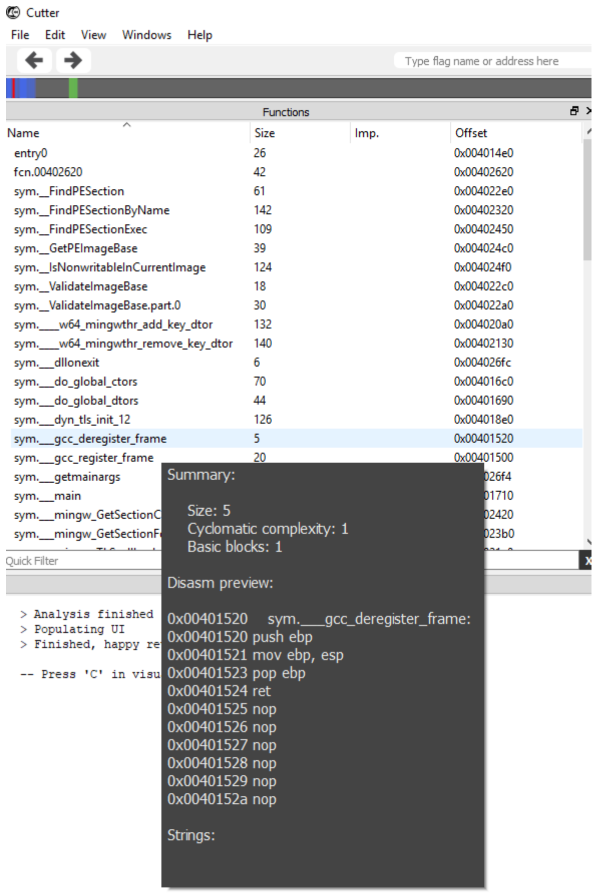
X-Refs from current address:

Address	Instruction

Cutter
File Edit View Windows Help
Type flag name or address here
Functions
Pseudocode
Sidebar
Quick Filter
Refresh
Dashboard Disassembly Graph (sym._main) Hexdump Pseudocode Entry Points Strings Imports Symbols Notepad
Entry Points
Sections

CUTTER- USEFUL WINDOWS FOR ANALYSIS - FUNCTIONS

- Shows each function, size and offset
- Sortable
  - Large functions usually more important



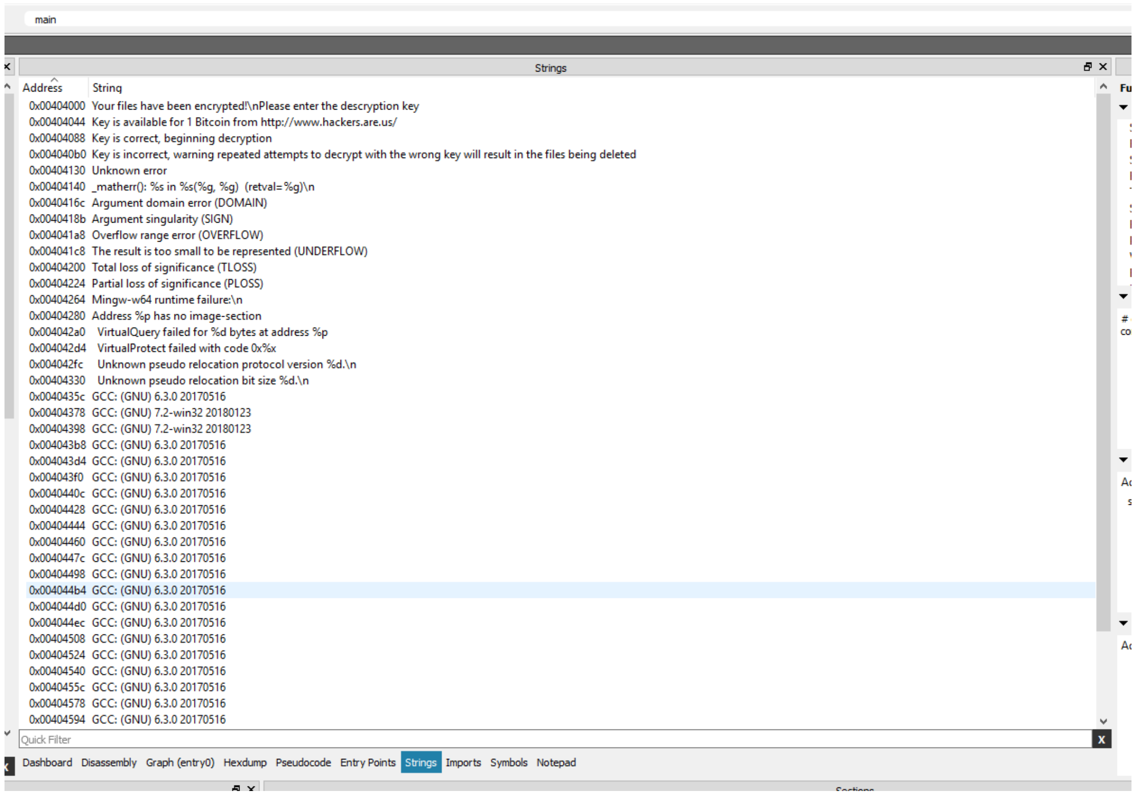
Name	Size	Imp.	Offset
entry0	26		0x004014e0
fcn.00402620	42		0x00402620
sym.__FindPESection	61		0x004022e0
sym.__FindPESectionByName	142		0x00402320
sym.__FindPESectionExec	109		0x00402450
sym.__GetPEImageBase	39		0x004024c0
sym.__IsNonwritableInCurrentImage	124		0x004024f0
sym.__ValidateImageBase	18		0x004022c0
sym.__ValidateImageBase.part.0	30		0x004022a0
sym.___w64_mingwthr_add_key_dtor	132		0x004020a0
sym.___w64_mingwthr_remove_key_dtor	140		0x00402130
sym.___dllonexit	6		0x004026fc
sym.___do_global_ctors	70		0x004016c0
sym.___do_global_dtors	44		0x00401690
sym.___dyn_tls_init_12	126		0x004018e0
sym.___gcc_deregister_frame	5		0x00401520
sym.___gcc_register_frame	20		0x00401500
sym.___getmainargs			026f4
sym.___main			01710
sym.___mingw_GetSectionC			02420
sym.___mingw_GetSectionF			023b0

```
Summary:

    Size: 5
    Cyclomatic complexity: 1
    Basic blocks: 1

Disasm preview:

0x00401520    sym.___gcc_deregister_frame:
0x00401520 push ebp
0x00401521 mov ebp, esp
0x00401523 pop ebp
0x00401524 ret
0x00401525 nop
0x00401526 nop
0x00401527 nop
0x00401528 nop
0x00401529 nop
0x0040152a nop

Strings:
```

```
> Analysis finished
> Populating UI
> Finished, happy re

-- Press 'C' in visu
```

CUTTER - USEFUL WINDOWS FOR ANALYSIS - STRINGS

- Any strings in your code

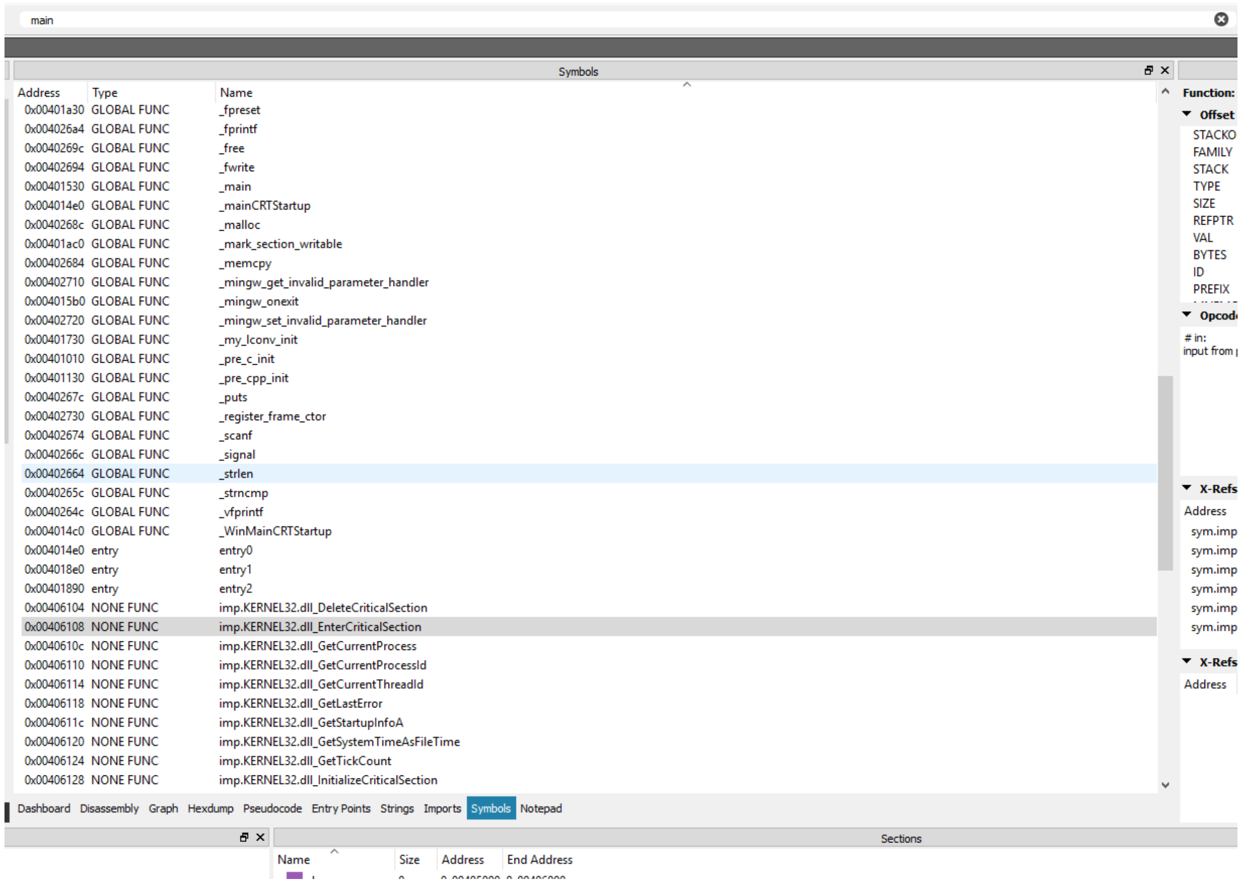


```
Address     String
0x00404000  Your files have been encrypted!\nPlease enter the descryption key
0x00404044  Key is available for 1 Bitcoin from http://www.hackers.are.us/
0x00404088  Key is correct, beginning decryption
0x004040b0  Key is incorrect, warning repeated attempts to decrypt with the wrong key will result in the files being deleted
0x00404130  Unknown error
0x00404140  _matherr(): %s in %s(%g, %g)  (retval=%g)\n
0x0040416c  Argument domain error (DOMAIN)
0x0040418b  Argument singularity (SIGN)
0x004041a8  Overflow range error (OVERFLOW)
0x004041c8  The result is too small to be represented (UNDERFLOW)
0x00404200  Total loss of significance (TLOSS)
0x00404224  Partial loss of significance (PLOSS)
0x00404264  Mingw-w64 runtime failure:\n
0x00404280  Address %p has no image-section
0x004042a0    VirtualQuery failed for %d bytes at address %p
0x004042d4    VirtualProtect failed with code 0x%x
0x004042fc    Unknown pseudo relocation protocol version %d.\n
0x00404330    Unknown pseudo relocation bit size %d.\n
0x0040435c  GCC: (GNU) 6.3.0 20170516
0x00404378  GCC: (GNU) 7.2-win32 20180123
0x00404398  GCC: (GNU) 7.2-win32 20180123
0x004043b8  GCC: (GNU) 6.3.0 20170516
0x004043d4  GCC: (GNU) 6.3.0 20170516
0x004043f0  GCC: (GNU) 6.3.0 20170516
0x0040440c  GCC: (GNU) 6.3.0 20170516
0x00404428  GCC: (GNU) 6.3.0 20170516
0x00404444  GCC: (GNU) 6.3.0 20170516
0x00404460  GCC: (GNU) 6.3.0 20170516
0x0040447c  GCC: (GNU) 6.3.0 20170516
0x00404498  GCC: (GNU) 6.3.0 20170516
0x004044b4  GCC: (GNU) 6.3.0 20170516
0x004044d0  GCC: (GNU) 6.3.0 20170516
0x004044ec  GCC: (GNU) 6.3.0 20170516
0x00404508  GCC: (GNU) 6.3.0 20170516
0x00404524  GCC: (GNU) 6.3.0 20170516
0x00404540  GCC: (GNU) 6.3.0 20170516
0x0040455c  GCC: (GNU) 6.3.0 20170516
0x00404578  GCC: (GNU) 6.3.0 20170516
0x00404594  GCC: (GNU) 6.3.0 20170516
```

CUTTER - USEFUL WINDOWS FOR ANALYSIS - SYMBOLS

OXFORD BROOKES UNIVERSITY

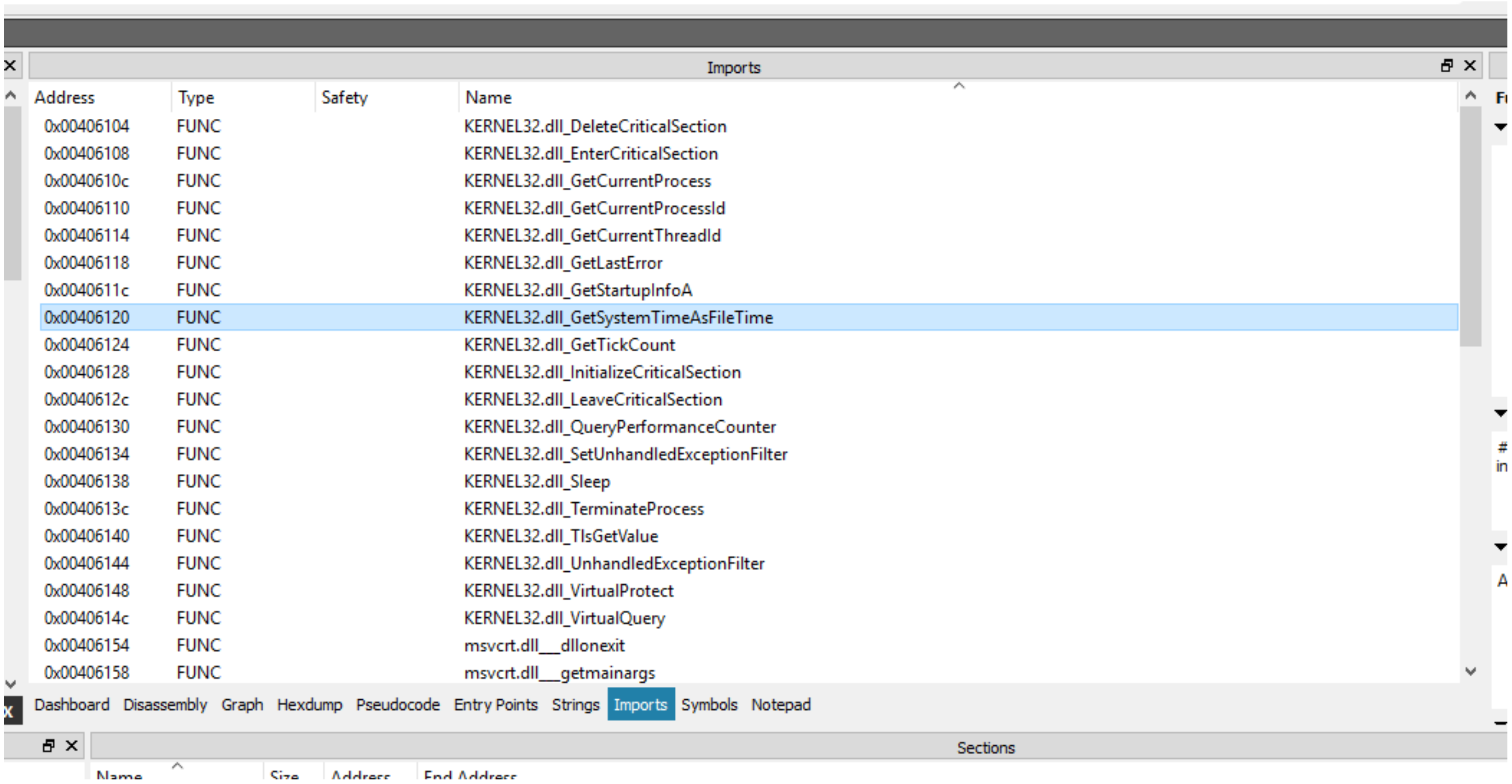
- Any key names in your code



Address	Type	Name
0x00401a30	GLOBAL FUNC	_fpreset
0x004026a4	GLOBAL FUNC	_fprintf
0x0040269c	GLOBAL FUNC	_free
0x00402694	GLOBAL FUNC	_fwrite
0x00401530	GLOBAL FUNC	_main
0x004014e0	GLOBAL FUNC	_mainCRTStartup
0x0040268c	GLOBAL FUNC	_malloc
0x00401ac0	GLOBAL FUNC	_mark_section_writable
0x00402684	GLOBAL FUNC	_memcpy
0x00402710	GLOBAL FUNC	_mingw_get_invalid_parameter_handler
0x004015b0	GLOBAL FUNC	_mingw_onexit
0x00402720	GLOBAL FUNC	_mingw_set_invalid_parameter_handler
0x00401730	GLOBAL FUNC	_my_lconv_init
0x00401010	GLOBAL FUNC	_pre_c_init
0x00401130	GLOBAL FUNC	_pre_cpp_init
0x0040267c	GLOBAL FUNC	_puts
0x00402730	GLOBAL FUNC	_register_frame_ctor
0x00402674	GLOBAL FUNC	_scanf
0x0040266c	GLOBAL FUNC	_signal
0x00402664	GLOBAL FUNC	_strlen
0x0040265c	GLOBAL FUNC	_strncmp
0x0040264c	GLOBAL FUNC	_vfprintf
0x004014c0	GLOBAL FUNC	_WinMainCRTStartup
0x004014e0	entry	entry0
0x004018e0	entry	entry1
0x00401890	entry	entry2
0x00406104	NONE FUNC	imp.KERNEL32.dll_DeleteCriticalSection
0x00406108	NONE FUNC	imp.KERNEL32.dll_EnterCriticalSection
0x0040610c	NONE FUNC	imp.KERNEL32.dll_GetCurrentProcess
0x00406110	NONE FUNC	imp.KERNEL32.dll_GetCurrentProcessId
0x00406114	NONE FUNC	imp.KERNEL32.dll_GetCurrentThreadId
0x00406118	NONE FUNC	imp.KERNEL32.dll_GetLastError
0x0040611c	NONE FUNC	imp.KERNEL32.dll_GetStartupInfoA
0x00406120	NONE FUNC	imp.KERNEL32.dll_GetSystemTimeAsFileTime
0x00406124	NONE FUNC	imp.KERNEL32.dll_GetTickCount
0x00406128	NONE FUNC	imp.KERNEL32.dll_InitializeCriticalSection

CUTTER- USEFUL WINDOWS FOR ANALYSIS - IMPORTS

This shows the malware sample is not "packed", i.e. it calls external libraries. They're not part of the binary itself!

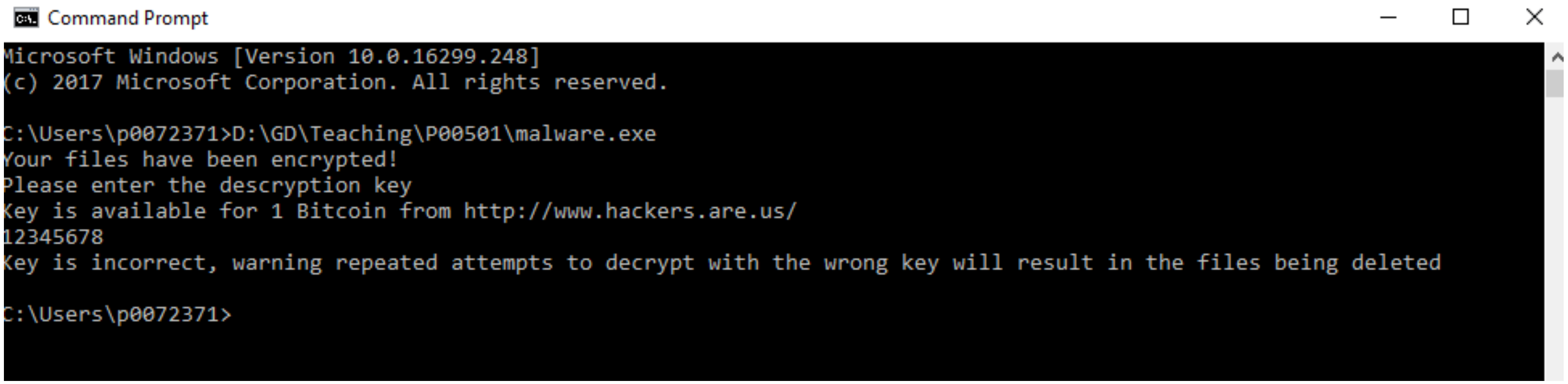


Address	Type	Safety	Name
0x00406104	FUNC		KERNEL32.dll_DeleteCriticalSection
0x00406108	FUNC		KERNEL32.dll_EnterCriticalSection
0x0040610c	FUNC		KERNEL32.dll_GetCurrentProcess
0x00406110	FUNC		KERNEL32.dll_GetCurrentProcessId
0x00406114	FUNC		KERNEL32.dll_GetCurrentThreadId
0x00406118	FUNC		KERNEL32.dll_GetLastError
0x0040611c	FUNC		KERNEL32.dll_GetStartupInfoA
0x00406120	FUNC		KERNEL32.dll_GetSystemTimeAsFileTime
0x00406124	FUNC		KERNEL32.dll_GetTickCount
0x00406128	FUNC		KERNEL32.dll_InitializeCriticalSection
0x0040612c	FUNC		KERNEL32.dll_LeaveCriticalSection
0x00406130	FUNC		KERNEL32.dll_QueryPerformanceCounter
0x00406134	FUNC		KERNEL32.dll_SetUnhandledExceptionFilter
0x00406138	FUNC		KERNEL32.dll_Sleep
0x0040613c	FUNC		KERNEL32.dll_TerminateProcess
0x00406140	FUNC		KERNEL32.dll_TlsGetValue
0x00406144	FUNC		KERNEL32.dll_UnhandledExceptionFilter
0x00406148	FUNC		KERNEL32.dll_VirtualProtect
0x0040614c	FUNC		KERNEL32.dll_VirtualQuery
0x00406154	FUNC		msvcrt.dll___dllonexit
0x00406158	FUNC		msvcrt.dll___getmainargs

Imports are external functions used by the code (often libraries)

REVERSE ENGINEERING EXAMPLE

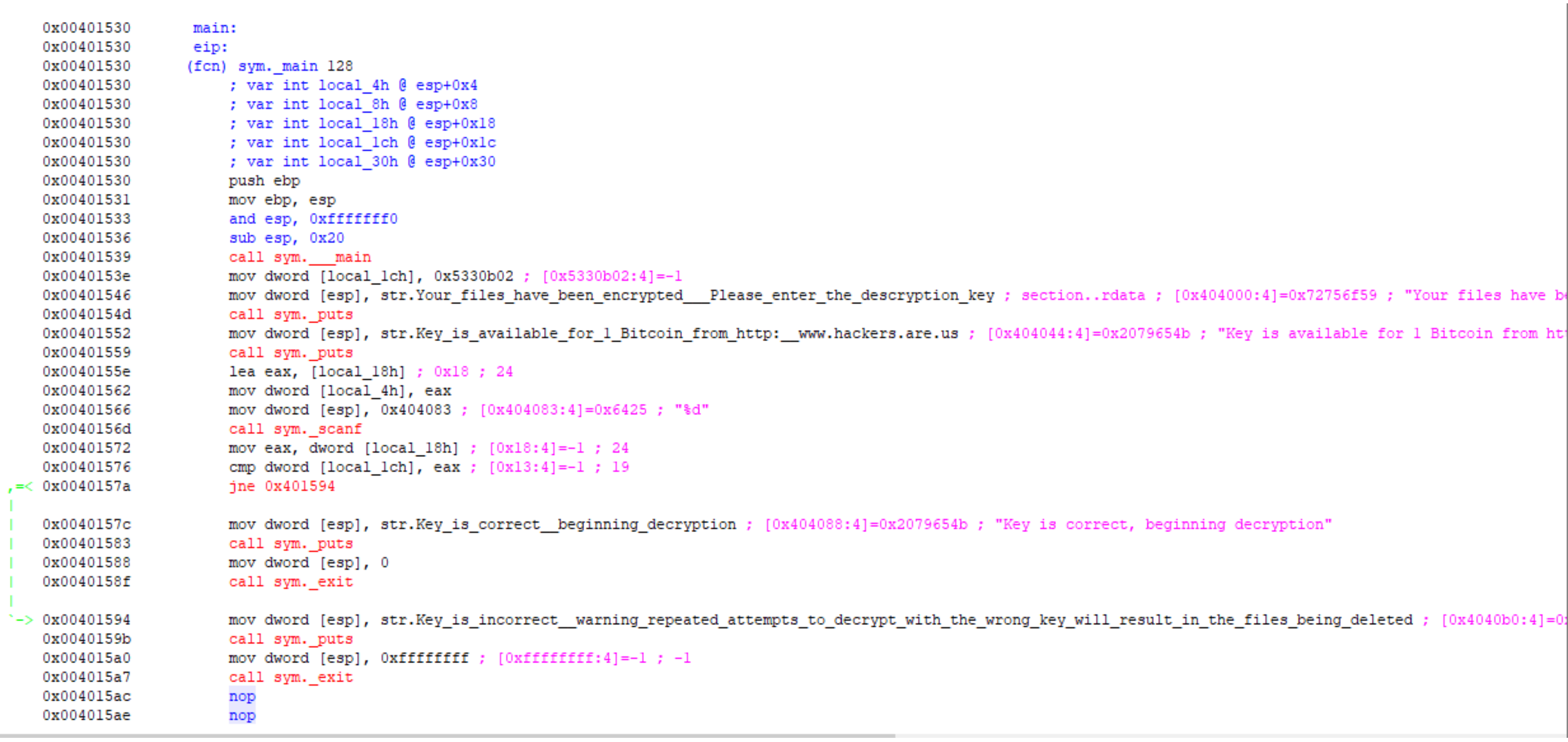
- Lets consider a simple example – Key Fishing
- This example only requires disassembler (cutter) and hex editor
  - disassembles to understand code and also patch the code
- For most real-world code, also need a debugger (SoftICE or OllyDbg)
- We have been the victim of ransomware
- Can we find the key to decrypt our disk?



```
Microsoft Windows [Version 10.0.16299.248]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\p0072371>D:\GD\Teaching\P00501\malware.exe
Your files have been encrypted!
Please enter the descryption key
Key is available for 1 Bitcoin from http://www.hackers.are.us/
12345678
Key is incorrect, warning repeated attempts to decrypt with the wrong key will result in the files being deleted

C:\Users\p0072371>
```

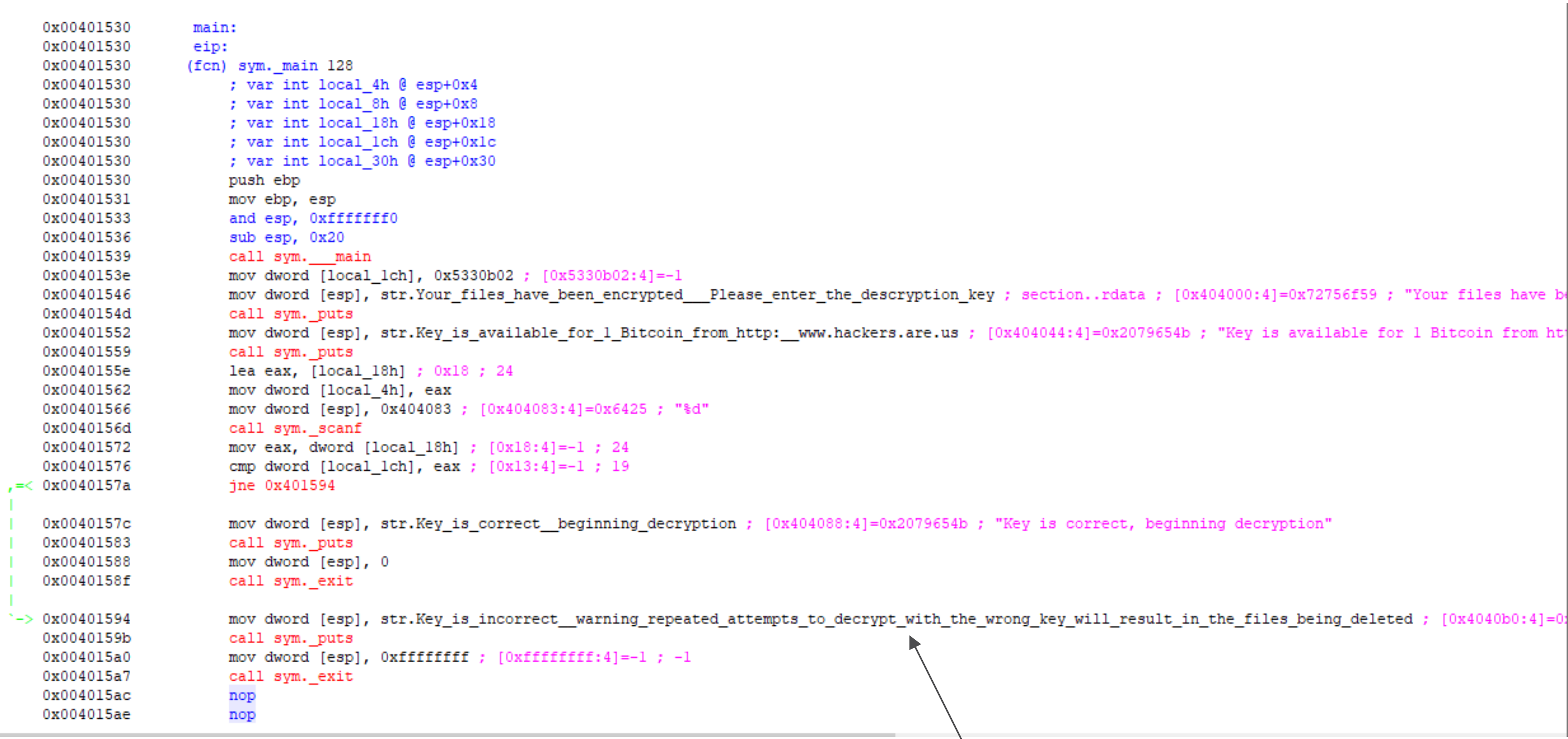
REVERSE ENGINEERING EXAMPLE

- Open the executable using Cutter



```
    0x00401530      main:
    0x00401530      eip:
    0x00401530     (fcn) sym._main 128
    0x00401530          ; var int local_4h @ esp+0x4
    0x00401530          ; var int local_8h @ esp+0x8
    0x00401530          ; var int local_18h @ esp+0x18
    0x00401530          ; var int local_1ch @ esp+0x1c
    0x00401530          ; var int local_30h @ esp+0x30
    0x00401530          push ebp
    0x00401531          mov ebp, esp
    0x00401533          and esp, 0xfffffff0
    0x00401536          sub esp, 0x20
    0x00401539          call sym.___main
    0x0040153e          mov dword [local_1ch], 0x5330b02 ; [0x5330b02:4]=-1
    0x00401546          mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [0x404000:4]=0x72756f59 ; "Your files have b
    0x0040154d          call sym._puts
    0x00401552          mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654b ; "Key is available for 1 Bitcoin from ht
    0x00401559          call sym._puts
    0x0040155e          lea eax, [local_18h] ; 0x18 ; 24
    0x00401562          mov dword [local_4h], eax
    0x00401566          mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
    0x0040156d          call sym._scanf
    0x00401572          mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
    0x00401576          cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
,=< 0x0040157a          jne 0x401594
|
|   0x0040157c          mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, beginning decryption"
|   0x00401583          call sym._puts
|   0x00401588          mov dword [esp], 0
|   0x0040158f          call sym._exit
|
`-> 0x00401594          mov dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decrypt_with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x4040b0:4]=0
    0x0040159b          call sym._puts
    0x004015a0          mov dword [esp], 0xffffffff ; [0xffffffff:4]=-1 ; -1
    0x004015a7          call sym._exit
    0x004015ac          nop
    0x004015ae          nop
```

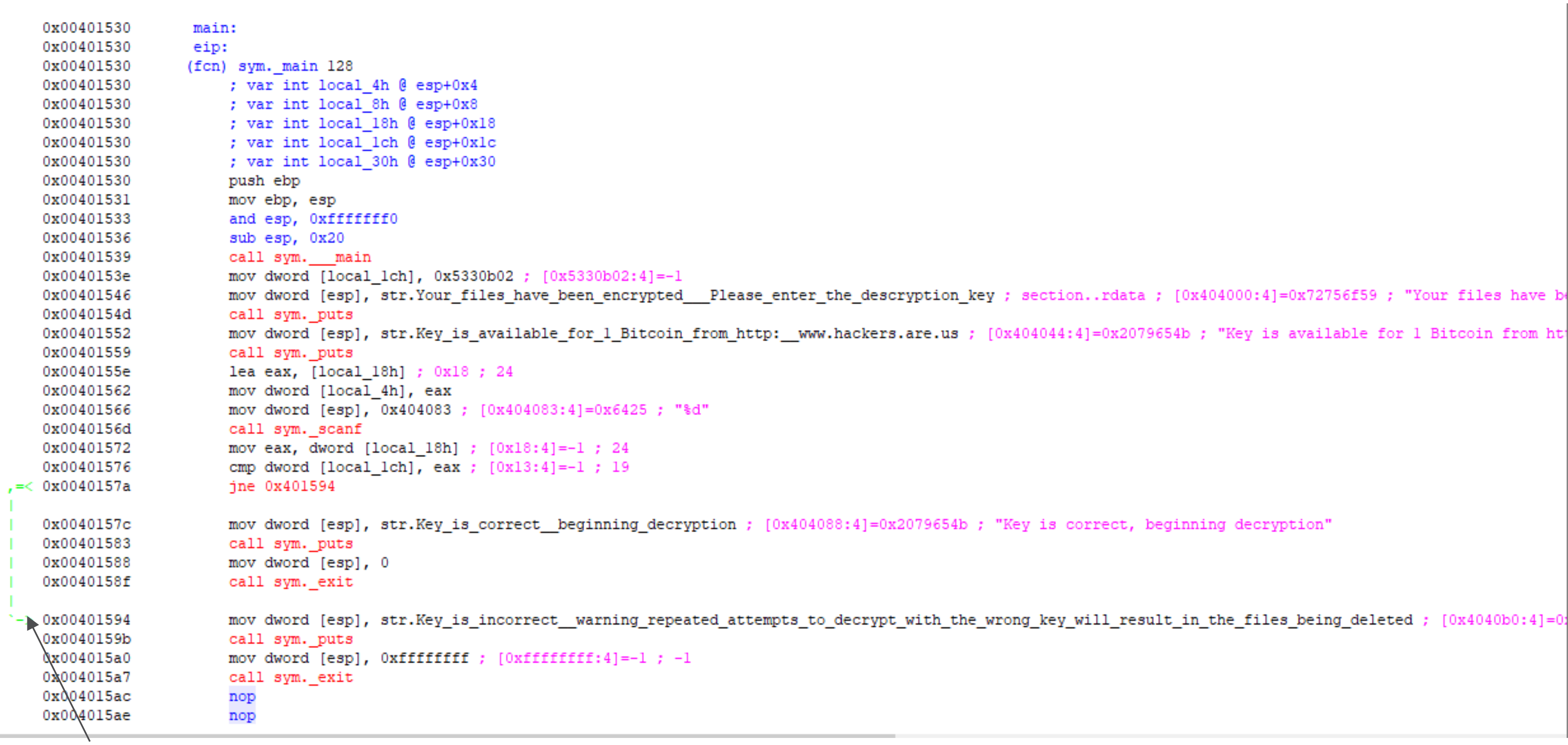
REVERSE ENGINEERING EXAMPLE



```
     0x00401530        main:
     0x00401530        eip:
     0x00401530       (fcn) sym._main 128
     0x00401530            ; var int local_4h @ esp+0x4
     0x00401530            ; var int local_8h @ esp+0x8
     0x00401530            ; var int local_18h @ esp+0x18
     0x00401530            ; var int local_1ch @ esp+0x1c
     0x00401530            ; var int local_30h @ esp+0x30
     0x00401530            push ebp
     0x00401531            mov ebp, esp
     0x00401533            and esp, 0xfffffff0
     0x00401536            sub esp, 0x20
     0x00401539            call sym.___main
     0x0040153e            mov dword [local_1ch], 0x5330b02 ; [0x5330b02:4]=-1
     0x00401546            mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [0x404000:4]=0x72756f59 ; "Your files have b
     0x0040154d            call sym._puts
     0x00401552            mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654b ; "Key is available for 1 Bitcoin from ht
     0x00401559            call sym._puts
     0x0040155e            lea eax, [local_18h] ; 0x18 ; 24
     0x00401562            mov dword [local_4h], eax
     0x00401566            mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
     0x0040156d            call sym._scanf
     0x00401572            mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
     0x00401576            cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
 ,=< 0x0040157a            jne 0x401594
 |
 |   0x0040157c            mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, beginning decryption"
 |   0x00401583            call sym._puts
 |   0x00401588            mov dword [esp], 0
 |   0x0040158f            call sym._exit
 |
 `-> 0x00401594            mov dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decrypt_with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x4040b0:4]=0
     0x0040159b            call sym._puts
     0x004015a0            mov dword [esp], 0xffffffff ; [0xffffffff:4]=-1 ; -1
     0x004015a7            call sym._exit
     0x004015ac            nop
     0x004015ae            nop
```

Looks like this is warning message, but how do we get to it?

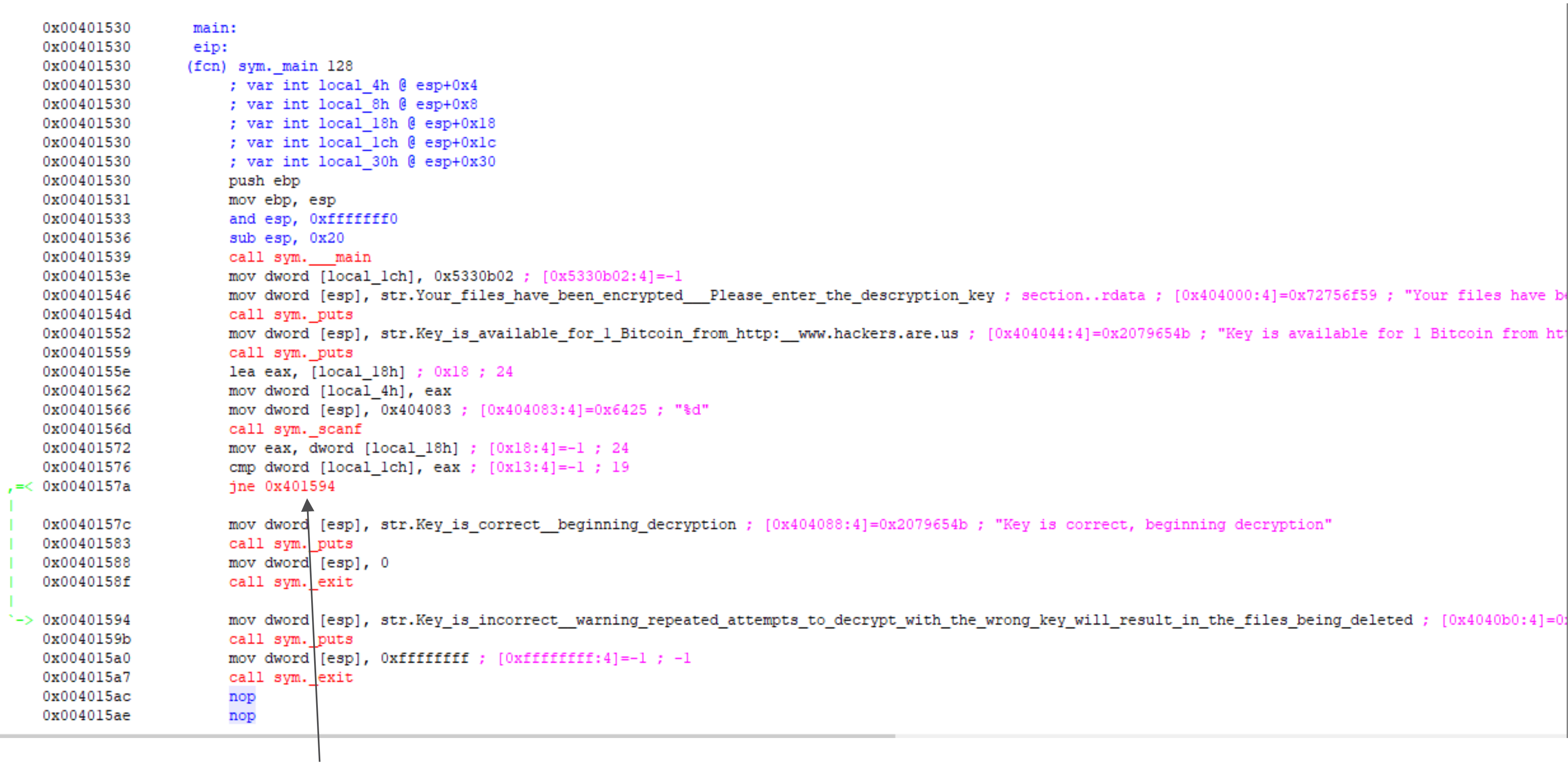
REVERSE ENGINEERING EXAMPLE



```
0x00401530       main:
0x00401530       eip:
0x00401530      (fcn) sym._main 128
0x00401530           ; var int local_4h @ esp+0x4
0x00401530           ; var int local_8h @ esp+0x8
0x00401530           ; var int local_18h @ esp+0x18
0x00401530           ; var int local_1ch @ esp+0x1c
0x00401530           ; var int local_30h @ esp+0x30
0x00401530           push ebp
0x00401531           mov ebp, esp
0x00401533           and esp, 0xfffffff0
0x00401536           sub esp, 0x20
0x00401539           call sym.___main
0x0040153e           mov dword [local_1ch], 0x5330b02 ; [0x5330b02:4]=-1
0x00401546           mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [0x404000:4]=0x72756f59 ; "Your files have b
0x0040154d           call sym._puts
0x00401552           mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654b ; "Key is available for 1 Bitcoin from ht
0x00401559           call sym._puts
0x0040155e           lea eax, [local_18h] ; 0x18 ; 24
0x00401562           mov dword [local_4h], eax
0x00401566           mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
0x0040156d           call sym._scanf
0x00401572           mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
0x00401576           cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
,=< 0x0040157a           jne 0x401594
|
|   0x0040157c           mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, beginning decryption"
|   0x00401583           call sym._puts
|   0x00401588           mov dword [esp], 0
|   0x0040158f           call sym._exit
|
`-> 0x00401594           mov dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decrypt_with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x4040b0:4]=0
    0x0040159b           call sym._puts
    0x004015a0           mov dword [esp], 0xffffffff ; [0xffffffff:4]=-1 ; -1
    0x004015a7           call sym._exit
    0x004015ac           nop
    0x004015ae           nop
```

Here's our jump line – we need to follow it back

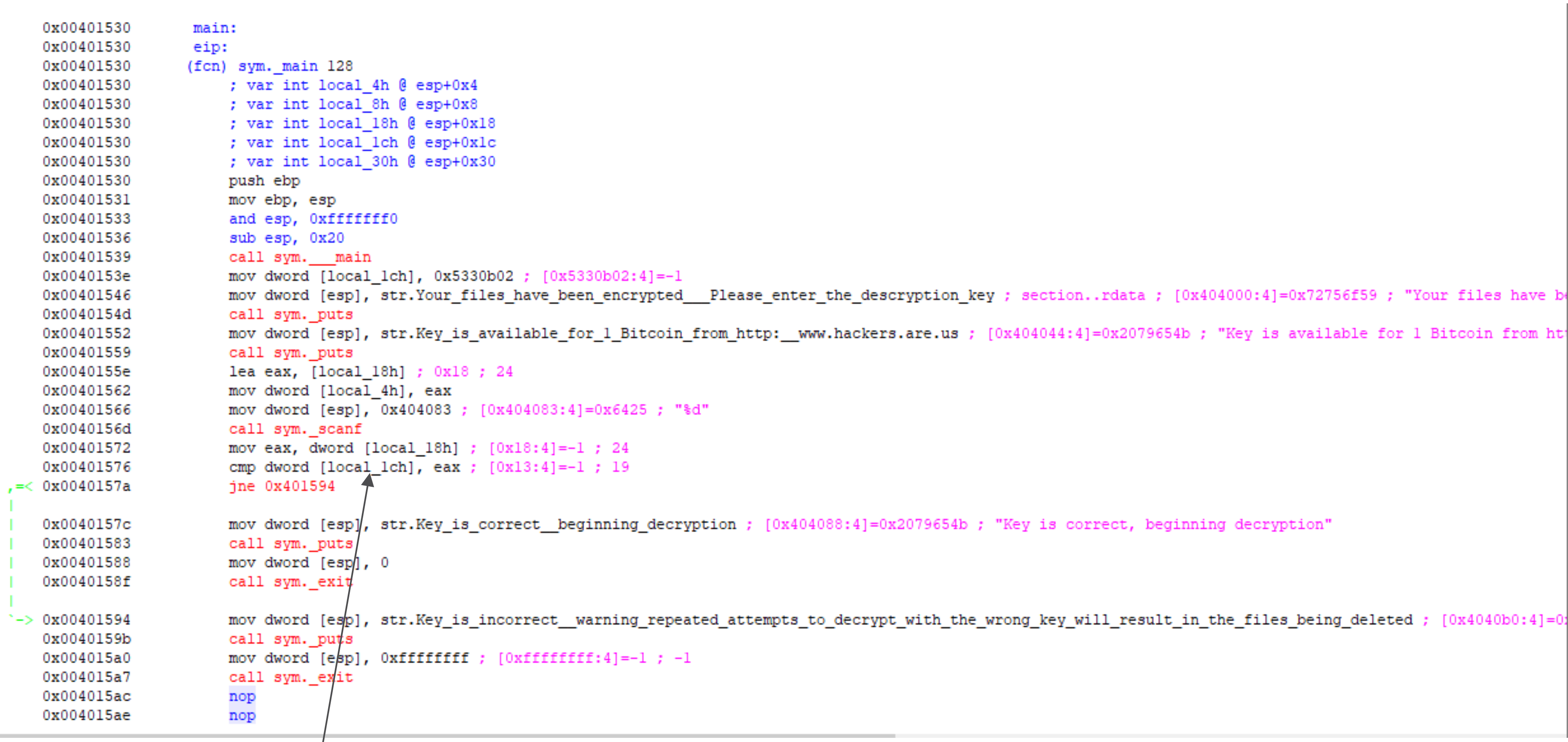
REVERSE ENGINEERING EXAMPLE



```
     0x00401530      main:
     0x00401530      eip:
     0x00401530     (fcn) sym._main 128
     0x00401530          ; var int local_4h @ esp+0x4
     0x00401530          ; var int local_8h @ esp+0x8
     0x00401530          ; var int local_18h @ esp+0x18
     0x00401530          ; var int local_1ch @ esp+0x1c
     0x00401530          ; var int local_30h @ esp+0x30
     0x00401530          push ebp
     0x00401531          mov ebp, esp
     0x00401533          and esp, 0xfffffff0
     0x00401536          sub esp, 0x20
     0x00401539          call sym.___main
     0x0040153e          mov dword [local_1ch], 0x5330b02 ; [0x5330b02:4]=-1
     0x00401546          mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [0x404000:4]=0x72756f59 ; "Your files have b
     0x0040154d          call sym._puts
     0x00401552          mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654b ; "Key is available for 1 Bitcoin from ht
     0x00401559          call sym._puts
     0x0040155e          lea eax, [local_18h] ; 0x18 ; 24
     0x00401562          mov dword [local_4h], eax
     0x00401566          mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
     0x0040156d          call sym._scanf
     0x00401572          mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
     0x00401576          cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
 ,=< 0x0040157a          jne 0x401594
 |
 |   0x0040157c          mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, beginning decryption"
 |   0x00401583          call sym._puts
 |   0x00401588          mov dword [esp], 0
 |   0x0040158f          call sym._exit
 |
 `-> 0x00401594          mov dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decrypt_with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x4040b0:4]=0
     0x0040159b          call sym._puts
     0x004015a0          mov dword [esp], 0xffffffff ; [0xffffffff:4]=-1 ; -1
     0x004015a7          call sym._exit
     0x004015ac          nop
     0x004015ae          nop
```

We jump if the last comparison is not equal to zero

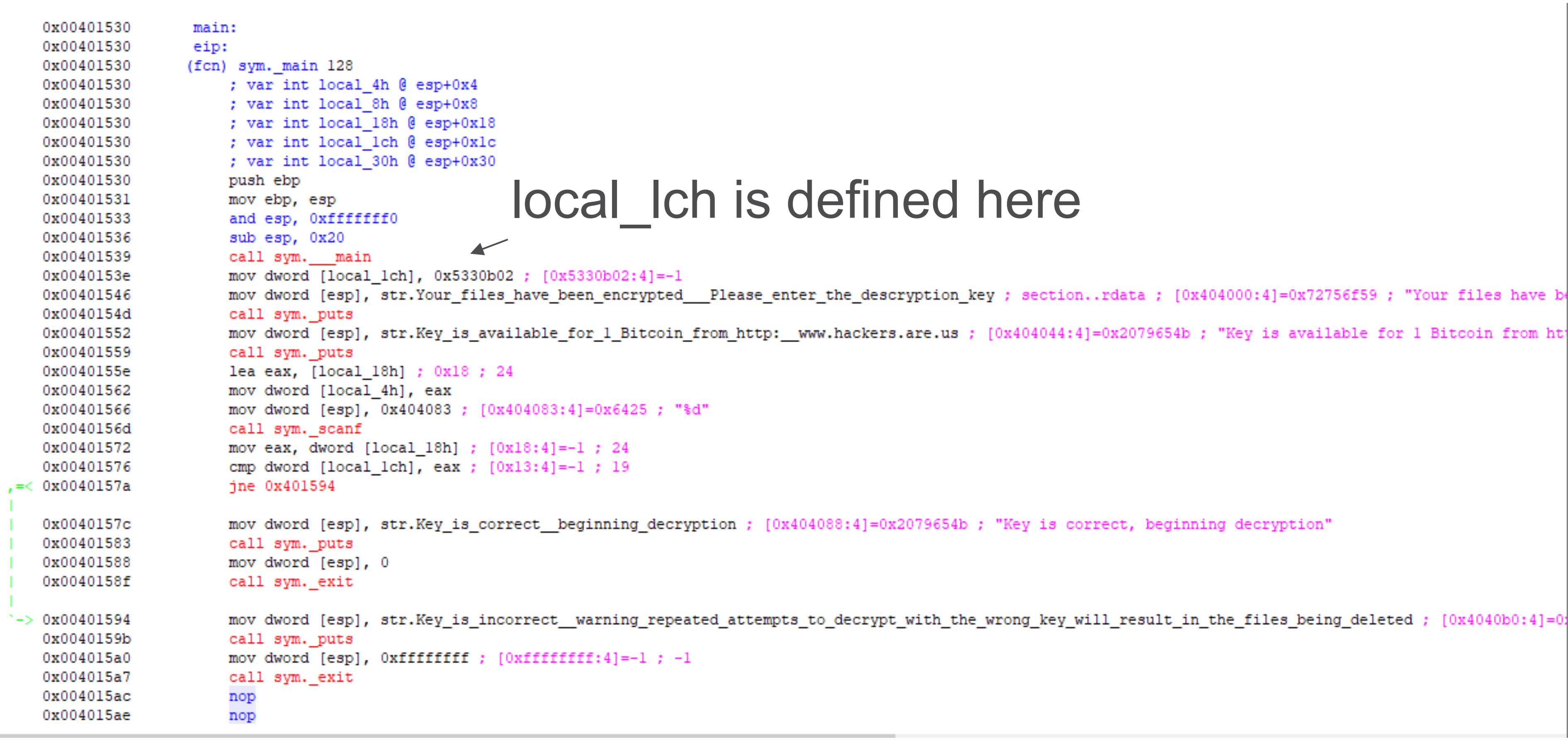
REVERSE ENGINEERING EXAMPLE



```
        0x00401530      main:
        0x00401530      eip:
        0x00401530     (fcn) sym._main 128
        0x00401530          ; var int local_4h @ esp+0x4
        0x00401530          ; var int local_8h @ esp+0x8
        0x00401530          ; var int local_18h @ esp+0x18
        0x00401530          ; var int local_1ch @ esp+0x1c
        0x00401530          ; var int local_30h @ esp+0x30
        0x00401530          push ebp
        0x00401531          mov ebp, esp
        0x00401533          and esp, 0xfffffff0
        0x00401536          sub esp, 0x20
        0x00401539          call sym.___main
        0x0040153e          mov dword [local_1ch], 0x5330b02 ; [0x5330b02:4]=-1
        0x00401546          mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [0x404000:4]=0x72756f59 ; "Your files have be
        0x0040154d          call sym._puts
        0x00401552          mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654b ; "Key is available for 1 Bitcoin from ht
        0x00401559          call sym._puts
        0x0040155e          lea eax, [local_18h] ; 0x18 ; 24
        0x00401562          mov dword [local_4h], eax
        0x00401566          mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
        0x0040156d          call sym._scanf
        0x00401572          mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
        0x00401576          cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
    ,=< 0x0040157a          jne 0x401594
    |
    |   0x0040157c          mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, beginning decryption"
    |   0x00401583          call sym._puts
    |   0x00401588          mov dword [esp], 0
    |   0x0040158f          call sym._exit
    |
    `-> 0x00401594          mov dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decrypt_with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x4040b0:4]=0
        0x0040159b          call sym._puts
        0x004015a0          mov dword [esp], 0xffffffff ; [0xffffffff:4]=-1 ; -1
        0x004015a7          call sym._exit
        0x004015ac          nop
        0x004015ae          nop
```

local_lch is therefore likely to hold our key

REVERSE ENGINEERING EXAMPLE

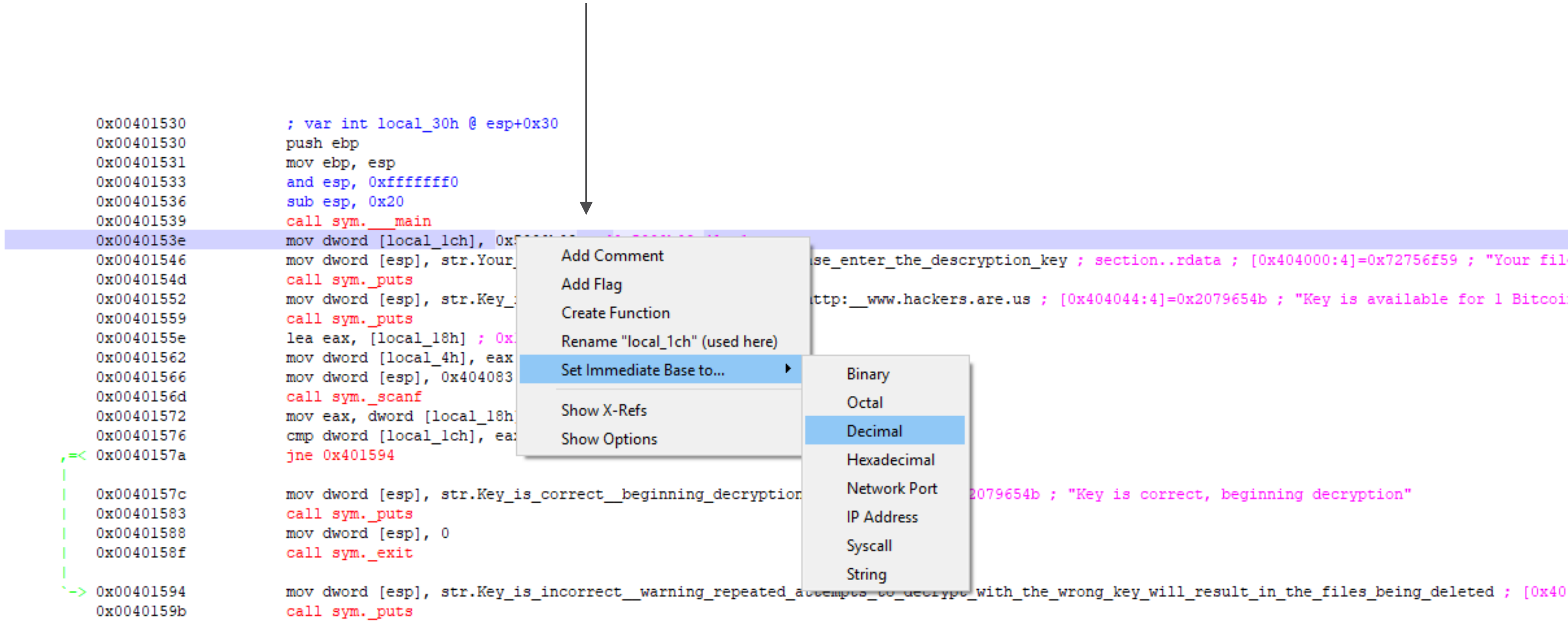


local_lch is defined here

```
       0x00401530        main:
       0x00401530        eip:
       0x00401530       (fcn) sym._main 128
       0x00401530            ; var int local_4h @ esp+0x4
       0x00401530            ; var int local_8h @ esp+0x8
       0x00401530            ; var int local_18h @ esp+0x18
       0x00401530            ; var int local_1ch @ esp+0x1c
       0x00401530            ; var int local_30h @ esp+0x30
       0x00401530            push ebp
       0x00401531            mov ebp, esp
       0x00401533            and esp, 0xfffffff0
       0x00401536            sub esp, 0x20
       0x00401539            call sym.___main
       0x0040153e            mov dword [local_1ch], 0x5330b02 ; [0x5330b02:4]=-1
       0x00401546            mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [0x404000:4]=0x72756f59 ; "Your files have b
       0x0040154d            call sym._puts
       0x00401552            mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654b ; "Key is available for 1 Bitcoin from ht
       0x00401559            call sym._puts
       0x0040155e            lea eax, [local_18h] ; 0x18 ; 24
       0x00401562            mov dword [local_4h], eax
       0x00401566            mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
       0x0040156d            call sym._scanf
       0x00401572            mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
       0x00401576            cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
   ,=< 0x0040157a            jne 0x401594
   |
   |   0x0040157c            mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, beginning decryption"
   |   0x00401583            call sym._puts
   |   0x00401588            mov dword [esp], 0
   |   0x0040158f            call sym._exit
   |
   `-> 0x00401594            mov dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decrypt_with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x4040b0:4]=0:
       0x0040159b            call sym._puts
       0x004015a0            mov dword [esp], 0xffffffff ; [0xffffffff:4]=-1 ; -1
       0x004015a7            call sym._exit
       0x004015ac            nop
       0x004015ae            nop
```

REVERSE ENGINEERING EXAMPLE

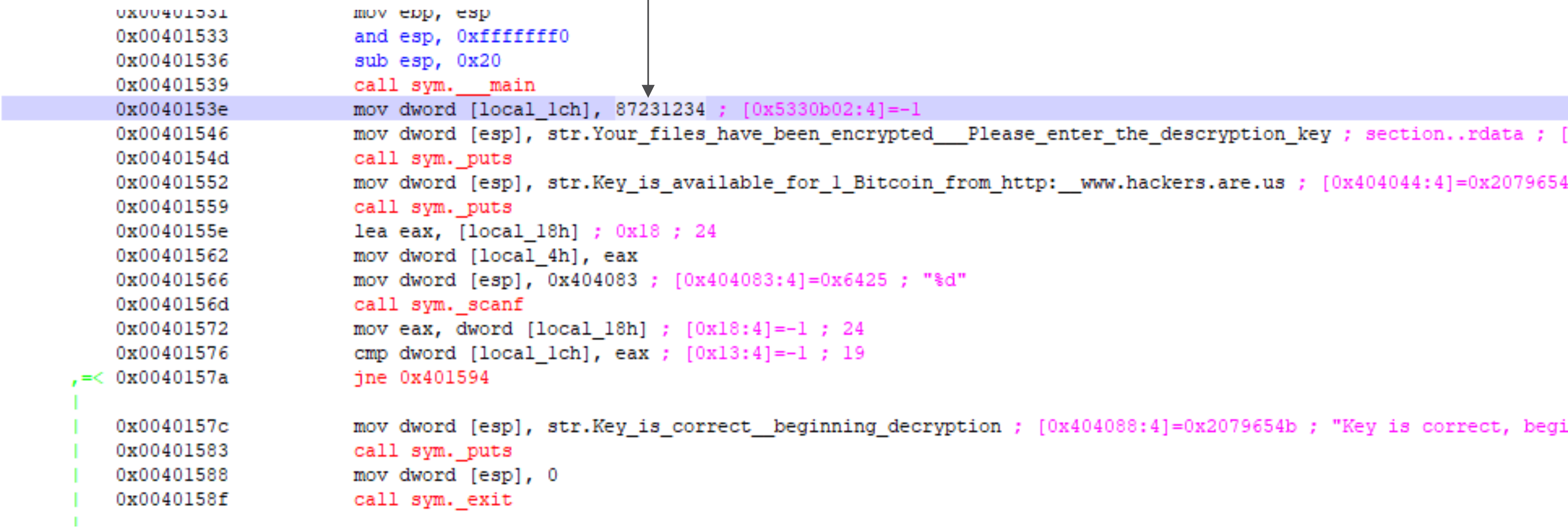
Lets make life easier and turn it to decimal



```
     0x00401530          ; var int local_30h @ esp+0x30
     0x00401530          push ebp
     0x00401531          mov ebp, esp
     0x00401533          and esp, 0xfffffff0
     0x00401536          sub esp, 0x20
     0x00401539          call sym.___main
     0x0040153e          mov dword [local_1ch], 0x
     0x00401546          mov dword [esp], str.Your_          se_enter_the_descryption_key ; section..rdata ; [0x404000:4]=0x72756f59 ; "Your fil
     0x0040154d          call sym._puts
     0x00401552          mov dword [esp], str.Key_           ttp:__www.hackers.are.us ; [0x404044:4]=0x2079654b ; "Key is available for 1 Bitcoi
     0x00401559          call sym._puts
     0x0040155e          lea eax, [local_18h] ; 0x
     0x00401562          mov dword [local_4h], eax
     0x00401566          mov dword [esp], 0x404083
     0x0040156d          call sym._scanf
     0x00401572          mov eax, dword [local_18h]
     0x00401576          cmp dword [local_1ch], ea
 ,=< 0x0040157a          jne 0x401594
 |
 |   0x0040157c          mov dword [esp], str.Key_is_correct__beginning_decryption          2079654b ; "Key is correct, beginning decryption"
 |   0x00401583          call sym._puts
 |   0x00401588          mov dword [esp], 0
 |   0x0040158f          call sym._exit
 |
 `-> 0x00401594          mov dword [esp], str.Key_is_incorrect__warning_repeated_a                 with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x40
     0x0040159b          call sym._puts
```

REVERSE ENGINEERING EXAMPLE

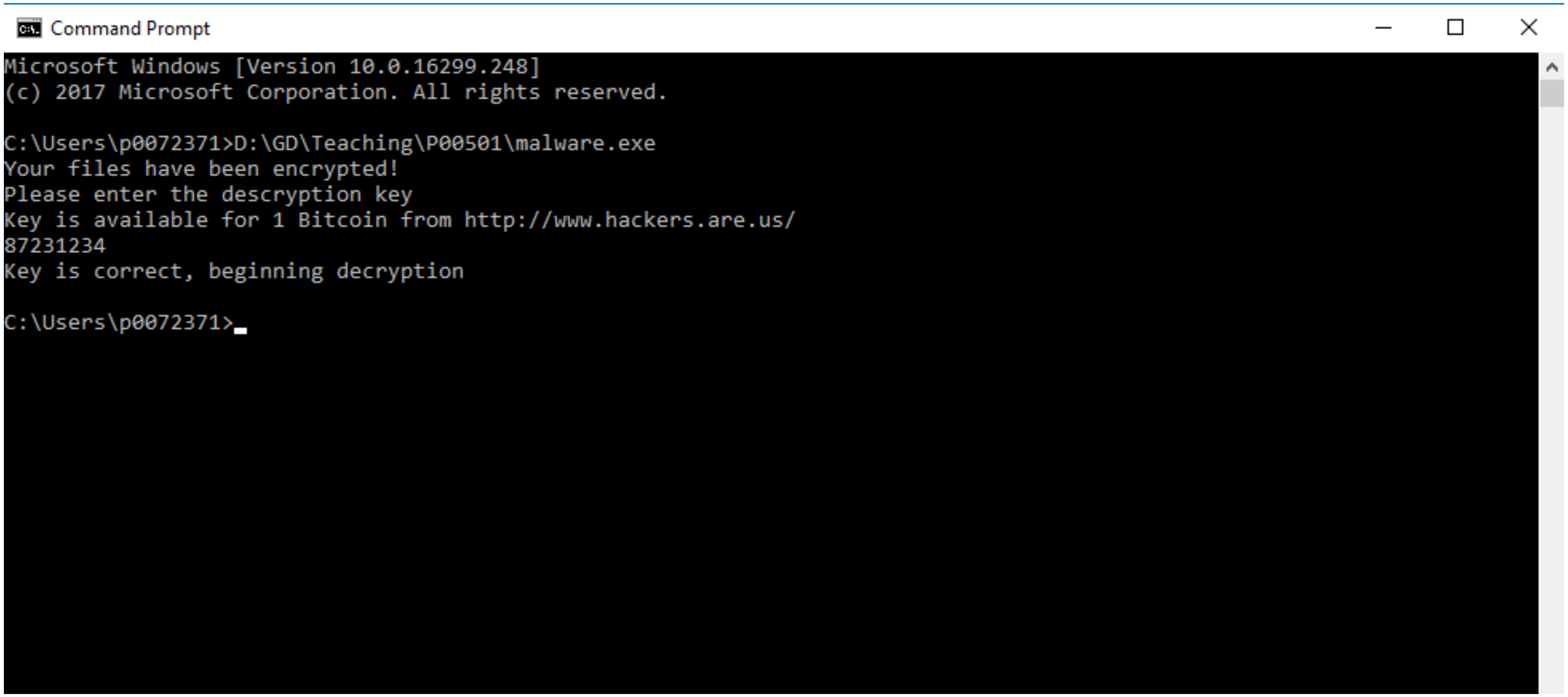
Looks like the key is 87231234



```
           0x00401531         mov ebp, esp
           0x00401533         and esp, 0xfffffff0
           0x00401536         sub esp, 0x20
           0x00401539         call sym.___main
           0x0040153e         mov dword [local_1ch], 87231234 ; [0x5330b02:4]=-1
           0x00401546         mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [
           0x0040154d         call sym._puts
           0x00401552         mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654
           0x00401559         call sym._puts
           0x0040155e         lea eax, [local_18h] ; 0x18 ; 24
           0x00401562         mov dword [local_4h], eax
           0x00401566         mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
           0x0040156d         call sym._scanf
           0x00401572         mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
           0x00401576         cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
       ,=< 0x0040157a         jne 0x401594
       |
       |   0x0040157c         mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, begi
       |   0x00401583         call sym._puts
       |   0x00401588         mov dword [esp], 0
       |   0x0040158f         call sym._exit
       |
```

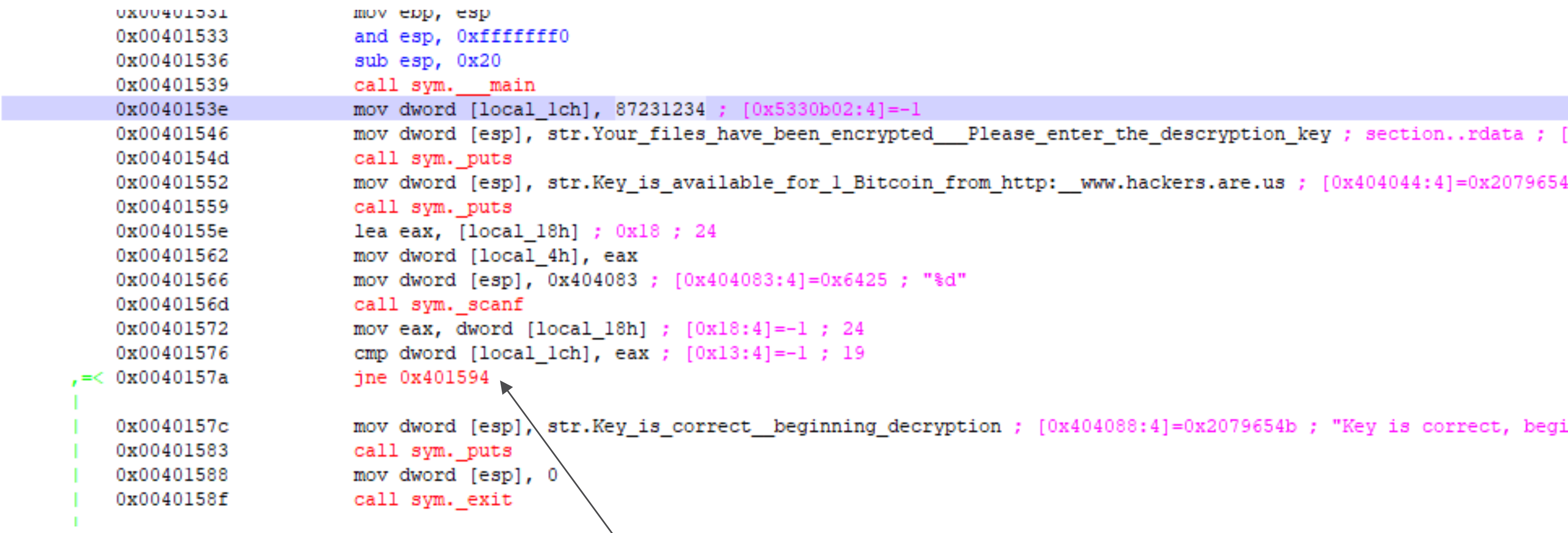
REVERSE ENGINEERING EXAMPLE

- Let’s try it



```
Microsoft Windows [Version 10.0.16299.248]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\p0072371>D:\GD\Teaching\P00501\malware.exe
Your files have been encrypted!
Please enter the descryption key
Key is available for 1 Bitcoin from http://www.hackers.are.us/
87231234
Key is correct, beginning decryption

C:\Users\p0072371>_
```

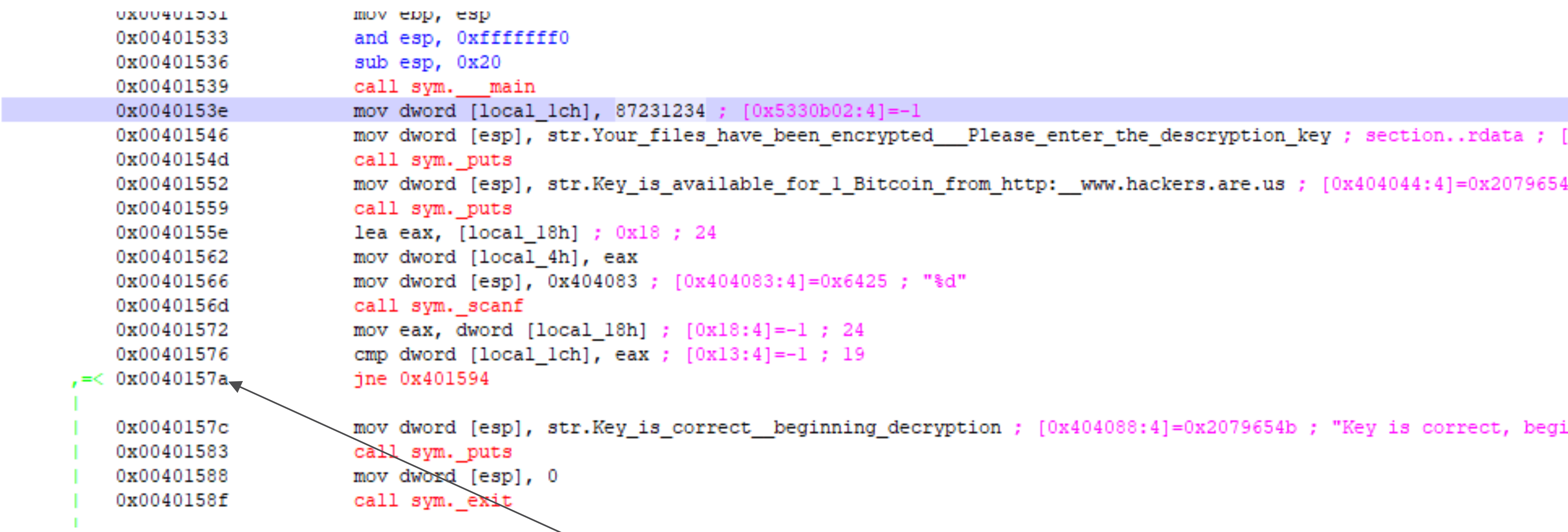
REVERSE ENGINEERING EXAMPLE



```
     0x00401531          mov ebp, esp
     0x00401533          and esp, 0xfffffff0
     0x00401536          sub esp, 0x20
     0x00401539          call sym.___main
     0x0040153e          mov dword [local_1ch], 87231234 ; [0x5330b02:4]=-1
     0x00401546          mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [
     0x0040154d          call sym._puts
     0x00401552          mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654
     0x00401559          call sym._puts
     0x0040155e          lea eax, [local_18h] ; 0x18 ; 24
     0x00401562          mov dword [local_4h], eax
     0x00401566          mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
     0x0040156d          call sym._scanf
     0x00401572          mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
     0x00401576          cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
 ,=< 0x0040157a          jne 0x401594
 |
 |   0x0040157c          mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, begi
 |   0x00401583          call sym._puts
 |   0x00401588          mov dword [esp], 0
 |   0x0040158f          call sym._exit
 |
```

Ideally, we want to patch our code to ignore this jump

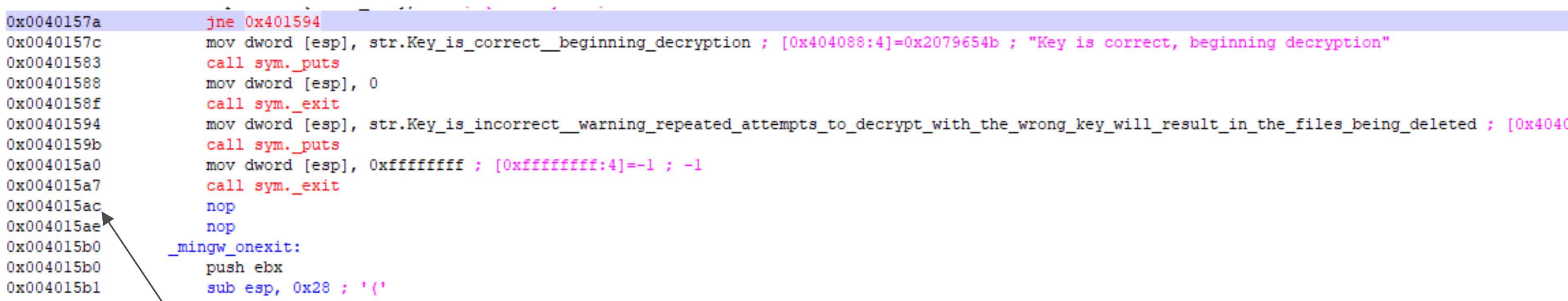
REVERSE ENGINEERING EXAMPLE



```
      0x00401531        mov ebp, esp
      0x00401533        and esp, 0xfffffff0
      0x00401536        sub esp, 0x20
      0x00401539        call sym.___main
      0x0040153e        mov dword [local_1ch], 87231234 ; [0x5330b02:4]=-1
      0x00401546        mov dword [esp], str.Your_files_have_been_encrypted___Please_enter_the_descryption_key ; section..rdata ; [
      0x0040154d        call sym._puts
      0x00401552        mov dword [esp], str.Key_is_available_for_1_Bitcoin_from_http:__www.hackers.are.us ; [0x404044:4]=0x2079654
      0x00401559        call sym._puts
      0x0040155e        lea eax, [local_18h] ; 0x18 ; 24
      0x00401562        mov dword [local_4h], eax
      0x00401566        mov dword [esp], 0x404083 ; [0x404083:4]=0x6425 ; "%d"
      0x0040156d        call sym._scanf
      0x00401572        mov eax, dword [local_18h] ; [0x18:4]=-1 ; 24
      0x00401576        cmp dword [local_1ch], eax ; [0x13:4]=-1 ; 19
   ,=< 0x0040157a       jne 0x401594
   |
   |  0x0040157c        mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, begi
   |  0x00401583        call sym._puts
   |  0x00401588        mov dword [esp], 0
   |  0x0040158f        call sym._exit
   |
```

Look up this address in the hexdump

REVERSE ENGINEERING EXAMPLE


```
0x0040157a          jne 0x401594
0x0040157c          mov dword [esp], str.Key_is_correct__beginning_decryption ; [0x404088:4]=0x2079654b ; "Key is correct, beginning decryption"
0x00401583          call sym._puts
0x00401588          mov dword [esp], 0
0x0040158f          call sym._exit
0x00401594          mov dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decrypt_with_the_wrong_key_will_result_in_the_files_being_deleted ; [0x404
0x0040159b          call sym._puts
0x004015a0          mov dword [esp], 0xffffffff ; [0xffffffff:4]=-1 ; -1
0x004015a7          call sym._exit
0x004015ac          nop
0x004015ae          nop
0x004015b0      _mingw_onexit:
0x004015b0          push ebx
0x004015b1          sub esp, 0x28 ; '('
```

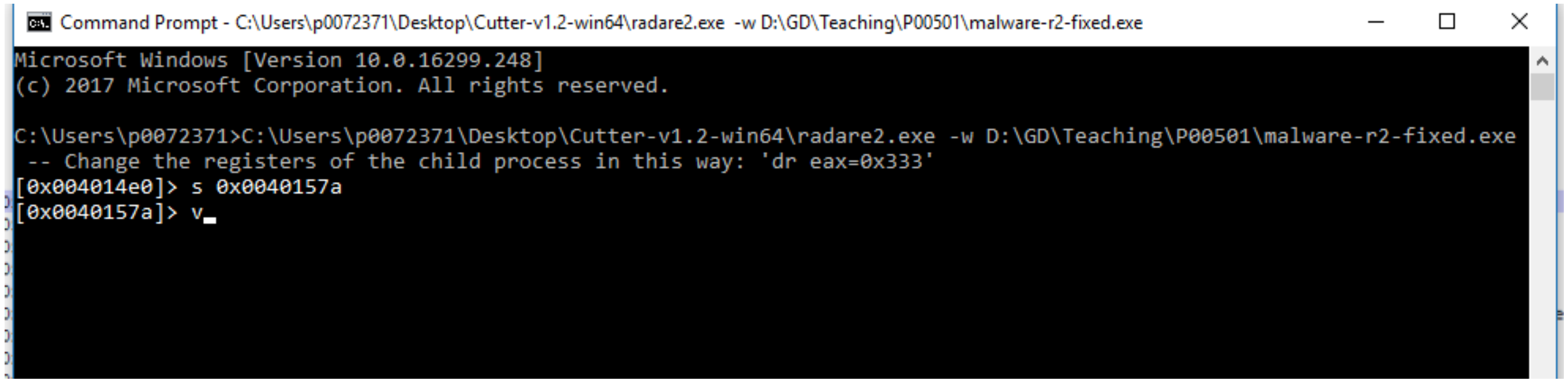
Look up this address in the hexdump

If we could replace our JNE with a NOP, we would always go to the decryption routine!

Unfortunately, Cutter doesn't support editing files

Or we could convert it into a JMP, delete it…

REVERSE ENGINEERING EXAMPLE

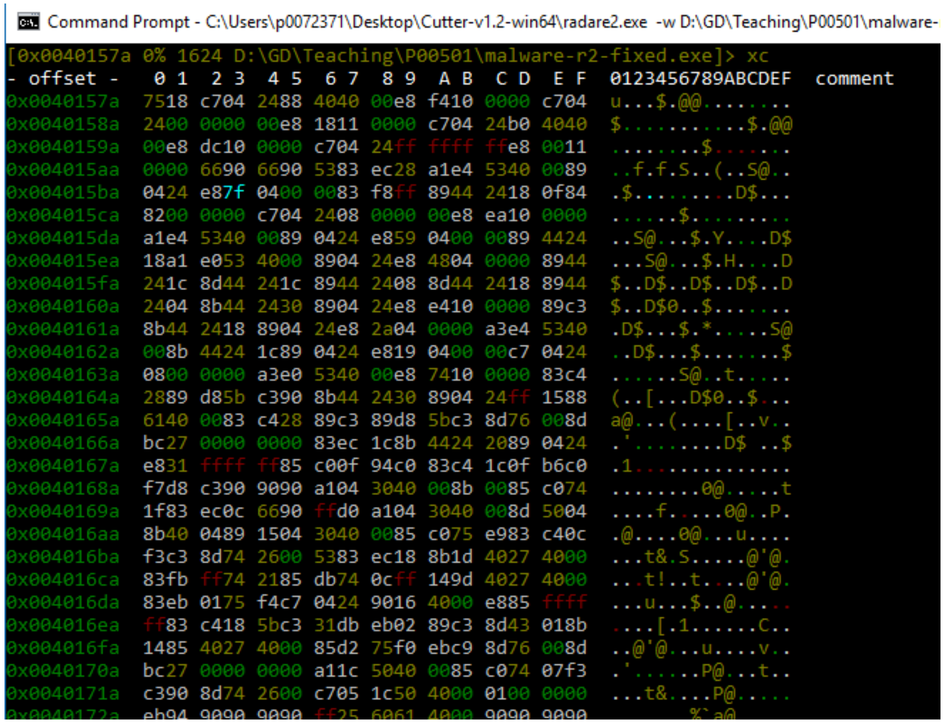


Command Prompt - C:\Users\p0072371\Desktop\Cutter-v1.2-win64\radare2.exe -w D:\GD\Teaching\P00501\malware-r2-fixed.exe

```
Microsoft Windows [Version 10.0.16299.248]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\p0072371>C:\Users\p0072371\Desktop\Cutter-v1.2-win64\radare2.exe -w D:\GD\Teaching\P00501\malware-r2-fixed.exe
 -- Change the registers of the child process in this way: 'dr eax=0x333'
[0x004014e0]> s 0x0040157a
[0x0040157a]> v_
```

- Cutter doesn’t support easy editing, but cutter is a front end to radare2 and that does support editing
- The command `s <address>` will take us to that part of the code
- The command `V` will enter visual mode

REVERSE ENGINEERING EXAMPLE



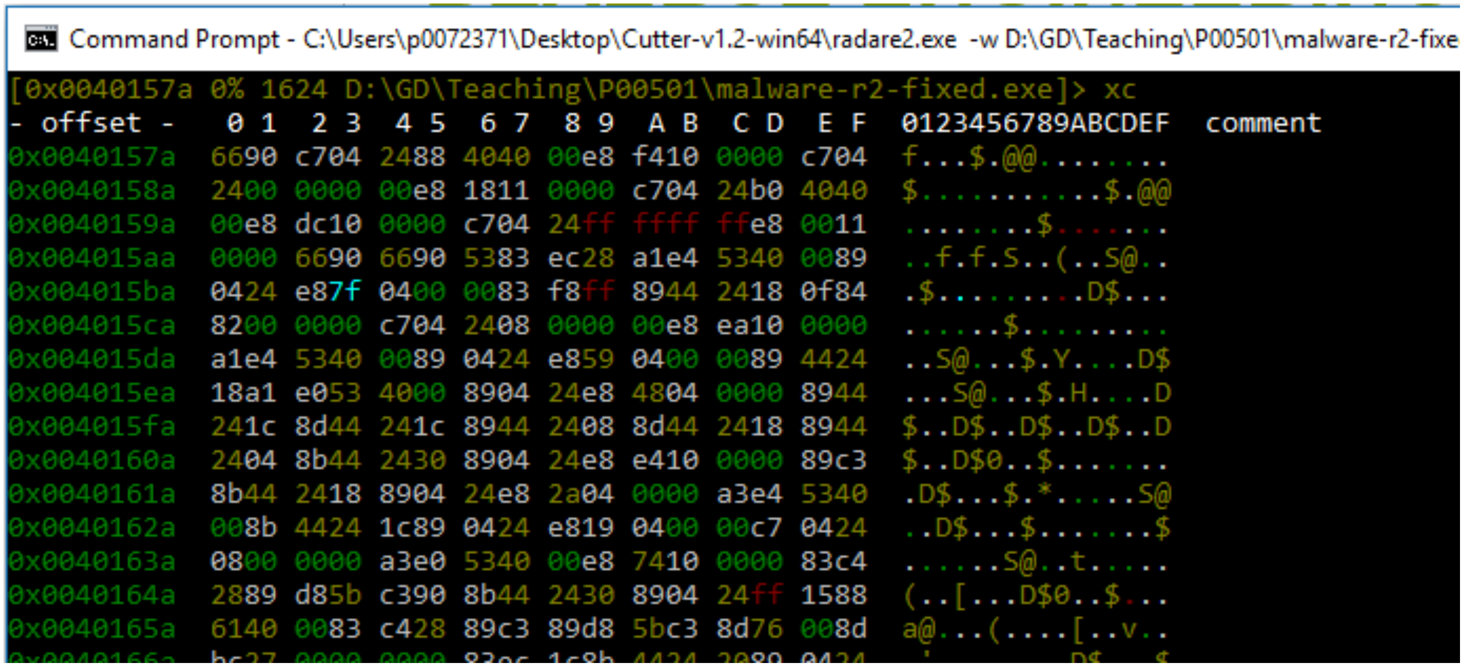
Command Prompt - C:\Users\p0072371\Desktop\Cutter-v1.2-win64\radare2.exe -w D:\GD\Teaching\P00501\malware-

```
[0x0040157a 0% 1624 D:\GD\Teaching\P00501\malware-r2-fixed.exe]> xc
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF  comment
0x0040157a  7518 c704 2488 4040 00e8 f410 0000 c704  u...$.@@........
0x0040158a  2400 0000 00e8 1811 0000 c704 24b0 4040  $...........$.@@
0x0040159a  00e8 dc10 0000 c704 24ff ffff ffe8 0011  ........$.......
0x004015aa  0000 6690 6690 5383 ec28 a1e4 5340 0089  ..f.f.S..(..S@..
0x004015ba  0424 e87f 0400 0083 f8ff 8944 2418 0f84  .$.........D$...
0x004015ca  8200 0000 c704 2408 0000 00e8 ea10 0000  ......$.........
0x004015da  a1e4 5340 0089 0424 e859 0400 0089 4424  ..S@...$.Y....D$
0x004015ea  18a1 e053 4000 8904 24e8 4804 0000 8944  ...S@...$.H....D
0x004015fa  241c 8d44 241c 8944 2408 8d44 2418 8944  $..D$..D$..D$..D
0x0040160a  2404 8b44 2430 8904 24e8 e410 0000 89c3  $..D$0..$.......
0x0040161a  8b44 2418 8904 24e8 2a04 0000 a3e4 5340  .D$...$.*.....S@
0x0040162a  008b 4424 1c89 0424 e819 0400 00c7 0424  ..D$...$.......$
0x0040163a  0800 0000 a3e0 5340 00e8 7410 0000 83c4  ......S@..t.....
0x0040164a  2889 d85b c390 8b44 2430 8904 24ff 1588  (..[...D$0..$...
0x0040165a  6140 0083 c428 89c3 89d8 5bc3 8d76 008d  a@...(....[..v..
0x0040166a  bc27 0000 0000 83ec 1c8b 4424 2089 0424  .'........D$ ..$
0x0040167a  e831 ffff ff85 c00f 94c0 83c4 1c0f b6c0  .1..............
0x0040168a  f7d8 c390 9090 a104 3040 008b 0085 c074  ........0@.....t
0x0040169a  1f83 ec0c 6690 ffd0 a104 3040 008d 5004  ....f.....0@..P.
0x004016aa  8b40 0489 1504 3040 0085 c075 e983 c40c  .@....0@...u....
0x004016ba  f3c3 8d74 2600 5383 ec18 8b1d 4027 4000  ...t&.S.....@'@.
0x004016ca  83fb ff74 2185 db74 0cff 149d 4027 4000  ...t!..t....@'@.
0x004016da  83eb 0175 f4c7 0424 9016 4000 e885 ffff  ...u...$..@.....
0x004016ea  ff83 c418 5bc3 31db eb02 89c3 8d43 018b  ....[.1......C..
0x004016fa  1485 4027 4000 85d2 75f0 ebc9 8d76 008d  ..@'@...u....v..
0x0040170a  bc27 0000 0000 a11c 5040 0085 c074 07f3  .'......P@...t..
0x0040171a  c390 8d74 2600 c705 1c50 4000 0100 0000  ...t&....P@.....
0x0040172a  eb94 9090 9090 ff25 6061 4000 9090 9090  ........%`a@....
```

Useful keys –

- `c` – switch to cursor mode
- `hjkl` – move
- `i` – insert mode
- `q` - quit

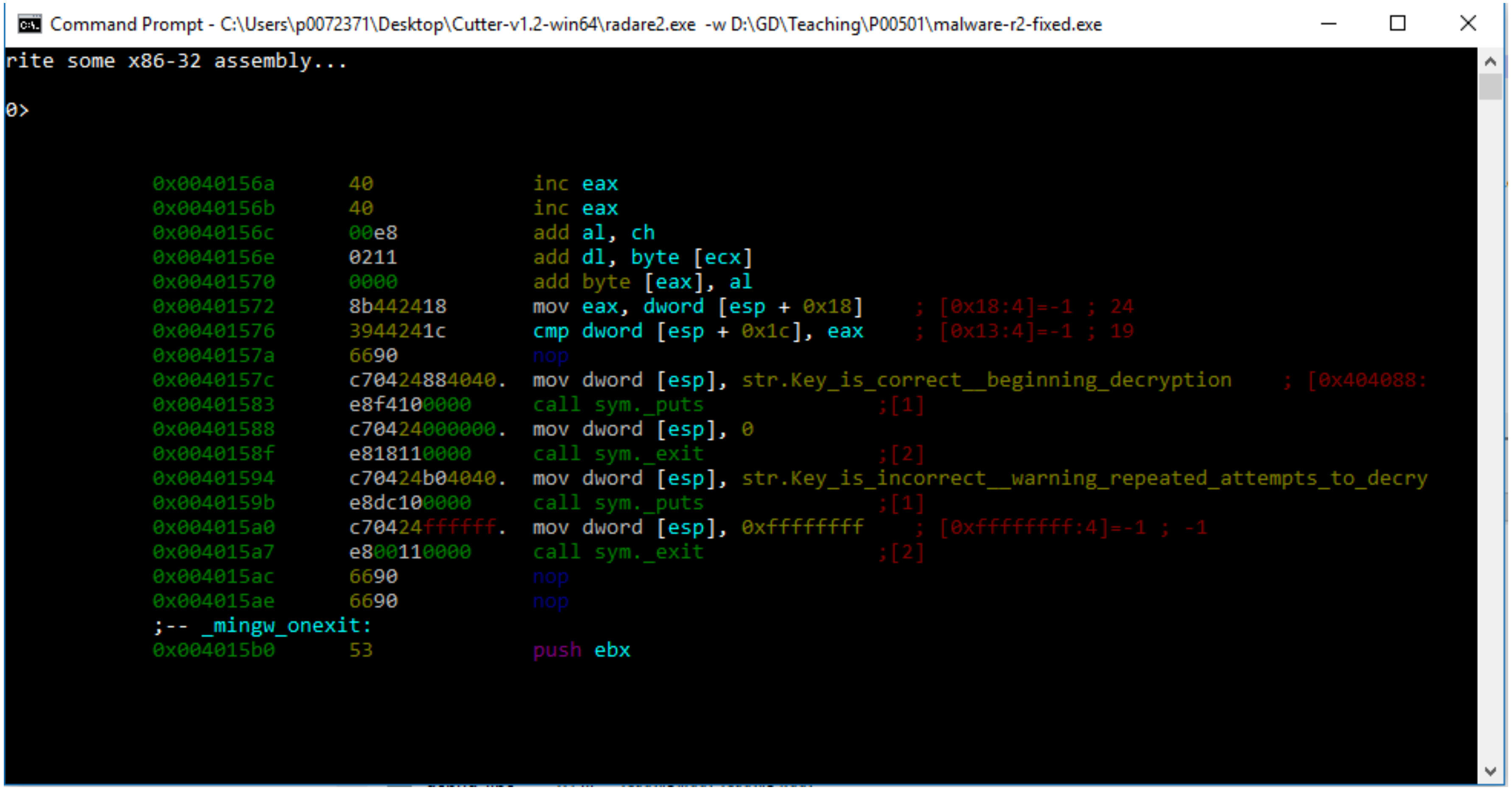
REVERSE ENGINEERING EXAMPLE



```
Command Prompt - C:\Users\p0072371\Desktop\Cutter-v1.2-win64\radare2.exe  -w D:\GD\Teaching\P00501\malware-r2-fixe
[0x0040157a 0% 1624 D:\GD\Teaching\P00501\malware-r2-fixed.exe]> xc
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF  comment
0x0040157a  6690 c704 2488 4040 00e8 f410 0000 c704  f...$.@@........
0x0040158a  2400 0000 00e8 1811 0000 c704 24b0 4040  $...........$.@@
0x0040159a  00e8 dc10 0000 c704 24ff ffff ffe8 0011  ........$.......
0x004015aa  0000 6690 6690 5383 ec28 a1e4 5340 0089  ..f.f.S..(..S@..
0x004015ba  0424 e87f 0400 0083 f8ff 8944 2418 0f84  .$.........D$...
0x004015ca  8200 0000 c704 2408 0000 00e8 ea10 0000  ......$.........
0x004015da  a1e4 5340 0089 0424 e859 0400 0089 4424  ..S@...$.Y....D$
0x004015ea  18a1 e053 4000 8904 24e8 4804 0000 8944  ...S@...$.H....D
0x004015fa  241c 8d44 241c 8944 2408 8d44 2418 8944  $..D$..D$..D$..D
0x0040160a  2404 8b44 2430 8904 24e8 e410 0000 89c3  $..D$0..$.......
0x0040161a  8b44 2418 8904 24e8 2a04 0000 a3e4 5340  .D$...$.*.....S@
0x0040162a  008b 4424 1c89 0424 e819 0400 00c7 0424  ..D$...$.......$
0x0040163a  0800 0000 a3e0 5340 00e8 7410 0000 83c4  ......S@..t.....
0x0040164a  2889 d85b c390 8b44 2430 8904 24ff 1588  (..[...D$0..$...
0x0040165a  6140 0083 c428 89c3 89d8 5bc3 8d76 008d  a@...(....[..v..
0x0040166a  bc27 0000 0000 83ec 1c8b 4424 2089 0424
```

We've replace the JNE with NOP
(Pressing `A` will show us the updated assembler)

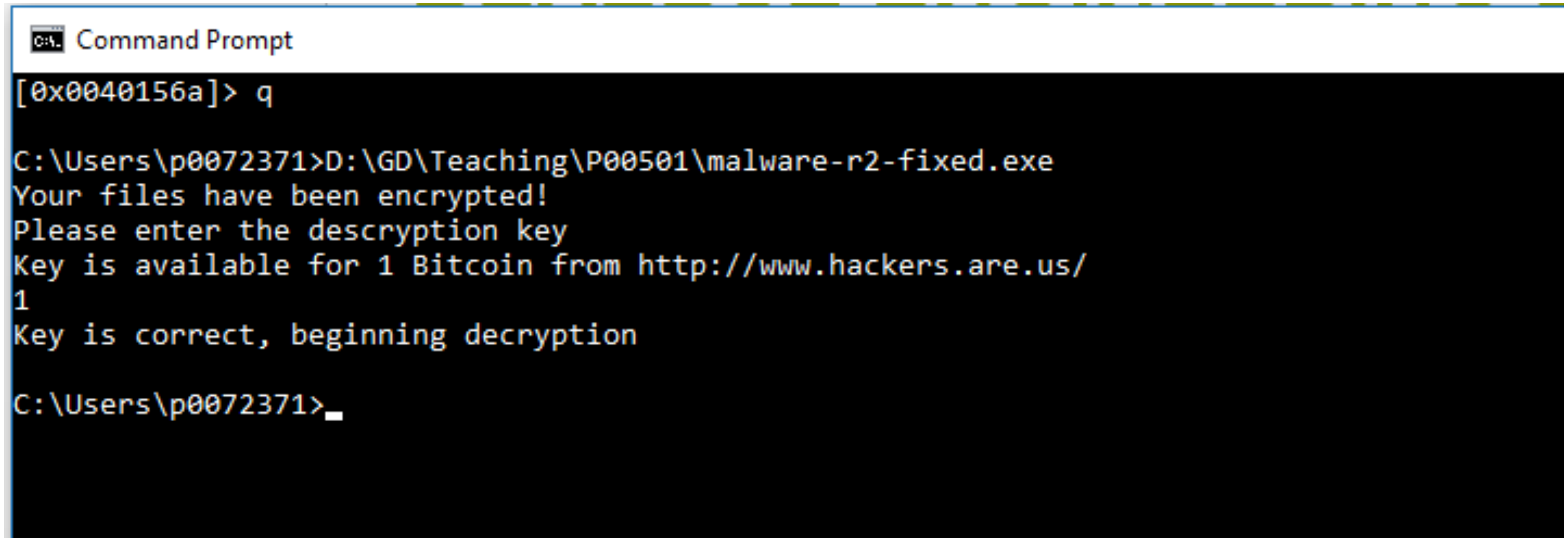
REVERSE ENGINEERING EXAMPLE



Command Prompt - C:\Users\p0072371\Desktop\Cutter-v1.2-win64\radare2.exe -w D:\GD\Teaching\P00501\malware-r2-fixed.exe

```
rite some x86-32 assembly...

0>


            0x0040156a      40              inc eax
            0x0040156b      40              inc eax
            0x0040156c      00e8            add al, ch
            0x0040156e      0211            add dl, byte [ecx]
            0x00401570      0000            add byte [eax], al
            0x00401572      8b442418        mov eax, dword [esp + 0x18]    ; [0x18:4]=-1 ; 24
            0x00401576      3944241c        cmp dword [esp + 0x1c], eax    ; [0x13:4]=-1 ; 19
            0x0040157a      6690            nop
            0x0040157c      c70424884040.   mov dword [esp], str.Key_is_correct__beginning_decryption    ; [0x404088:
            0x00401583      e8f4100000      call sym._puts              ;[1]
            0x00401588      c70424000000.   mov dword [esp], 0
            0x0040158f      e818110000      call sym._exit              ;[2]
            0x00401594      c70424b04040.   mov dword [esp], str.Key_is_incorrect__warning_repeated_attempts_to_decry
            0x0040159b      e8dc100000      call sym._puts              ;[1]
            0x004015a0      c70424ffffff.   mov dword [esp], 0xffffffff    ; [0xffffffff:4]=-1 ; -1
            0x004015a7      e800110000      call sym._exit              ;[2]
            0x004015ac      6690            nop
            0x004015ae      6690            nop
            ;-- _mingw_onexit:
            0x004015b0      53              push ebx
```

REVERSE ENGINEERING EXAMPLE



Command Prompt

```
[0x0040156a]> q

C:\Users\p0072371>D:\GD\Teaching\P00501\malware-r2-fixed.exe
Your files have been encrypted!
Please enter the descryption key
Key is available for 1 Bitcoin from http://www.hackers.are.us/
1
Key is correct, beginning decryption

C:\Users\p0072371>_
```

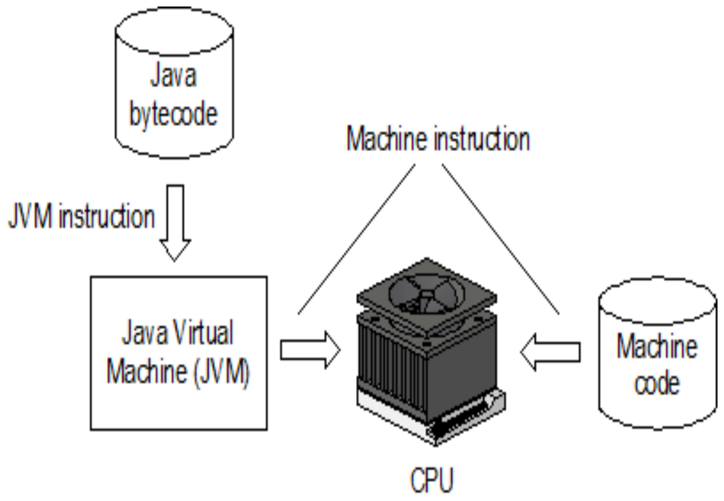
Now it doesn’t matter what the key the user types in!

BOOMERANG – DECOMPILER

Original source code	Disassembled binary code	Decompiled source code
#include <stdio.h>		
int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};	8049460 01000000 02000000 03000000 04000000	int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
	8049470 05000000 06000000 07000000 08000000	
	8049480 09000000 0a000000	
int main() {	8048328: push %ebp	int main(int argc, char** argv, char** envp)
	8048329: mov %esp,%ebp	{
	804832b: sub $0x8,%esp	int local1; // m[r28{0} - 8] // sum
	804832e: and $0xfffffff0,%esp	int local2; // m[r28{0} - 12] // i
	8048331: mov $0x0,%eax	
	8048336: sub %eax,%esp	
int sum = 0;	8048338: movl $0x0,0xfffffffc(%ebp)	local1 = 0;
int i;	804833f: movl $0x0,0xfffffff8(%ebp)	local2 = 0;
for (i=0; i < 10; i++) {	8048346: cmpl $0x9,0xfffffff8(%ebp)	while (local2 <= 9) {
	804834a: jle 804834e <main+0x26>	
	804834c: jmp 8048364 <main+0x3c>	
sum += a[i];	804834e: mov 0xfffffff8(%ebp),%eax	local1 += a[local2]; // sum += a[i]
	8048351: mov 0x8049460(,%eax,4),%edx	
	8048358: lea 0xfffffffc(%ebp),%eax	
	804835b: add %edx,(%eax)	
	804835d: lea 0xfffffff8(%ebp),%eax	local2++; // i++
	8048360: incl (%eax)	
}	8048362: jmp 8048346 <main+0x1e>	}
printf("Sum is %d\n", sum);	8048364: sub $0x8,%esp	printf("Sum is %d\n", local1);
	8048367: pushl 0xfffffffc(%ebp)	
	804836a: push $0x804842c	
	804836f: call 8048268 <printf@plt>	
	8048374: add $0x10,%esp	
return 0;	8048377: mov $0x0,%eax	return 0;
}	804837c: leave	}
	804837d: ret	
	804842c 53756d20 69732025 Sum is %	
	8048434 640a00 d..	

REVERSING AND PATCHING JAVA BYTECODE

- The following formal definitions of machine code and Java bytecode apply:
  - **Machine code**: "Machine code or machine language is a system of instructions and data executed directly by a computer's central processing unit". Machine code contains the platform-specific machine instructions to execute on the target processor.
  - **Java bytecode**: "Bytecode is the intermediate representation of Java programs just as assembler is the intermediate representation of C or C++ programs". Java bytecode contains platform-independent instructions that are translated to platform-specific instructions by a Java Virtual Machine.



REVERSING AND PATCHING JAVA BYTECODE

- Good-quality Java source can often be generated from Java bytecode with little difficulty due to certain characteristics of bytecode:
  - Platform-independent (consistent) instruction set and layout/format.
  - Very rich, well-structured metadata about Classes, Methods, and Variables:
    - names and datatypes (e.g., String personName, Map personRecord).
    - Method signatures (includes Constructors).
- Generating HLL source (e.g., C/C++) from machine code is challenging due to high variation in the output of compilers on different platforms and unavoidable loss of information that occurs when compiling a HLL down to machine code.

REVERSING AND PATCHING JAVA BYTECODE

- Machine code is stored in files with varying extensions (**.exe, *.dll, …)*
  - extensions are dependent upon the operating system.
- On the contrary…
  - Java bytecode is *always* stored in files that have a **.class* extension.
- The Java Language Specification allows at most one top-level public class to be defined per **.java* source file and requires that the bytecode be stored in a file whose name matches the pattern *TopLevelClassName.class*.
- Collections of Java classes, such as those for an application or class library, are stored together in an archive file with a **.jar* extension.

REVERSING AND PATCHING JAVA BYTECODE

DECOMPILING AND DISASSEMBLING JAVA BYTECODE

- Bytecode is stored in a binary format that is not human-readable and therefore must be “disassembled” in order to be read.

- Oracle’s Java Development Toolkit (JDK) comes with javap, a command-line tool for “disassembling” Java bytecode.

- javap disassembles bytecode – not entirely true
  - the output of javap is unstructured text which cannot be compiled back to bytecode.
  - The assumption is you already have the **.class* file.

- The output of javap is nonetheless useful as a debugging and performance tuning aid since one can see which JVM instructions are generated from high- level Java language statements.

REVERSING AND PATCHING JAVA BYTECODE

DECOMPILING AND DISASSEMBLING JAVA BYTECODE

- A better way to disassemble Java bytecode is to use the commercial product IntelliJ by Jet Brains
- The community version (free version) can disassemble but the commercial version (free to students) lets you do more to patch
- To use IntelliJ as a disassembler, just open the class/jar file instead of the source file

C TO ASSEMBLY (1)

```c
int a = 0;
int b = 1;
a = a + 11;
a = a - b;
a--;
b++;
b = a % 3;
```

```
00401006        mov [ebp+var_4], 0
0040100D        mov [ebp+var_8], 1
00401014        mov eax, [ebp+var_4]
00401017        add eax, 0Bh
0040101A        mov [ebp+var_4], eax
0040101D        mov ecx, [ebp+var_4]
00401020        sub ecx, [ebp+var_8]
00401023        mov [ebp+var_4], ecx
00401026        mov edx, [ebp+var_4]
00401029        sub edx, 1
0040102C        mov [ebp+var_4], edx
0040102F        mov eax, [ebp+var_8]
00401032        add eax, 1
00401035        mov [ebp+var_8], eax
00401038        mov eax, [ebp+var_4]
0040103B        cdq
0040103C        mov ecx, 3
00401041        idiv ecx
00401043        mov [ebp+var_8], edx
```

C TO ASSEMBLY (2)

```
int x = 1;
int y = 2;

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

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

WHAT DID WE LEARN TODAY?

- What is Reverse Engineering?
- Static vs dynamic analysis
- RE and Assembly
- Disassembly, Decompilers
- Radare2 & Cutter
- Ransomware protection
- Reversing and Patching Java Bytecode

Practical today will try out some of these tools discussed.

Next week we will learn about Dynamic Analysis