恶意代码分析与防治技术

第8章 动态调试

王志
zwang@nankai.edu.cn

updated on 2022-11-6

南开大学 网络空间安全学院
2022/2023

本章知识点

- 源码级动态调试与汇编级动态调试
- 内核模式动态调试与用户模式动态调试
- 动态调试器的使用
- Windows异常处理机制
  - 难点：Second Chance Exception，int3 中断，int1中断
- 恶意代码的动态修改

Source-Level vs. Assembly-Level Debuggers

Debugger

Is the debugger a piece of software or hardware?

Why we need a debugger?

- Why does a program give wrong results?
- What is a program doing?

Debugger

- Give you insight into what a program is doing
  - a dynamic view
- Measure and control the internal state and execution of a program
  - change anything at any time
  - memory, register, argument to every function

Disassemblers vs. Debuggers

IDA Pro

static overview of the whole program

Debugger

dynamic accurate view of one concrete execution path

change anything at any time during the execution

Two Debuggers

- Ollydbg
  - Most popular for malware analysis
  - User-mode debugging only
  - IDA Pro has a built-in debugger, but it's not as easy to use or powerful as Ollydbg
- Windbg
  - Supports kernel-mode debugging

Source-Level vs. Assembly-Level Debuggers

- Source-level
  - operate on source code
  - step through program execution one line at a time
- Assembly-level
  - Low-level debugger
  - Operate on assembly code
  - step through program execution one instruction at a time

单选题 10分

设置

Which item below could view a sequence of CPU instructions execution ?

A Source-level debugger

B Assembly-level debugger

C Disassembler

D ProcessMoniter

提交

User Mode vs. Kernel Mode

Debuggers

User Mode vs. Kernel Mode

- Review
  - What is User Mode ?
  - What is Kernel Mode ?

User mode debugger

•debug user mode codes

Kernel mode debugger

•debug kernel mode codes

User Mode Debugging

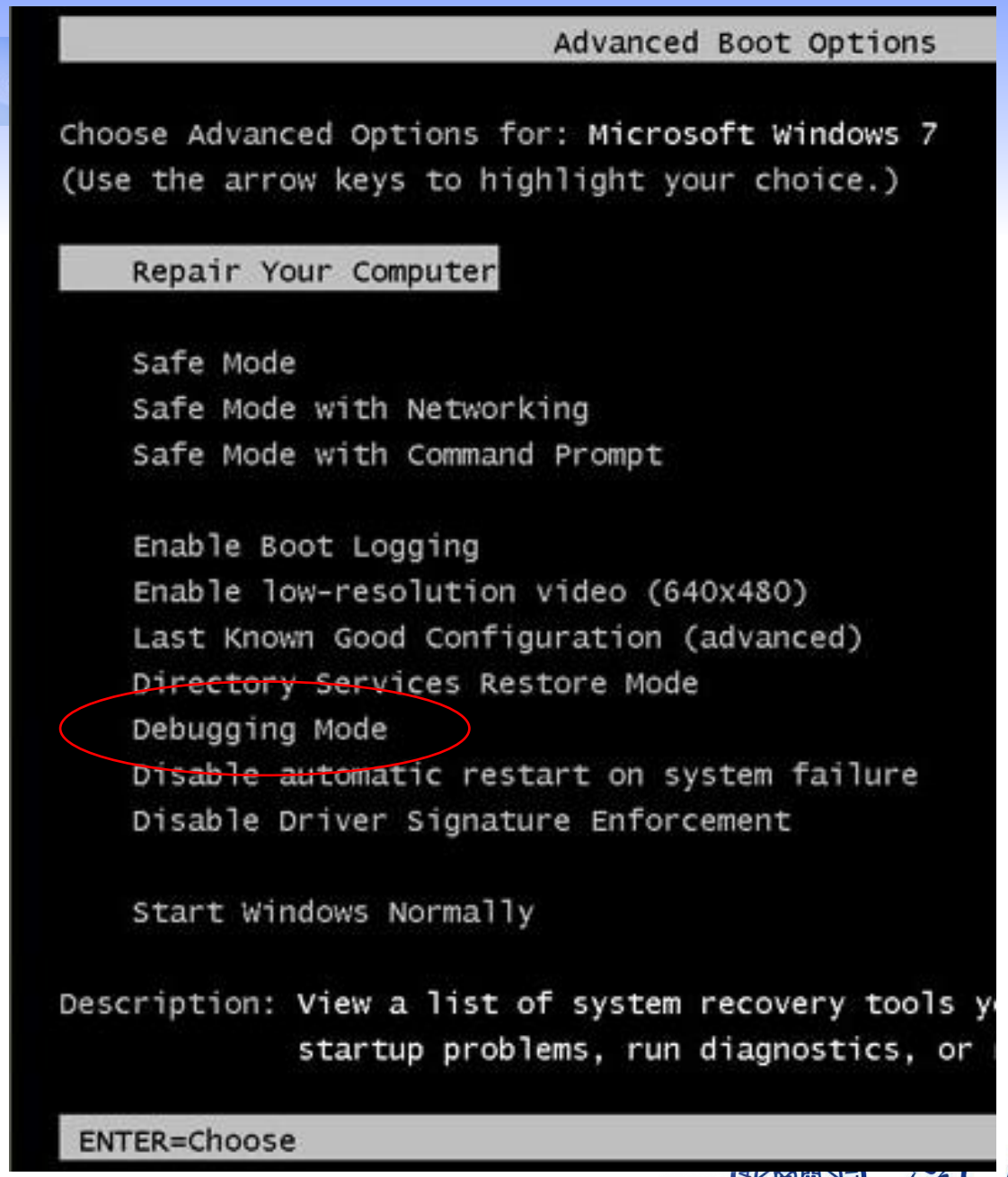
- OS supports multiple user mode program execution at the same time
- Debugger and the code being debugged could be running on the same OS.
- OllyDbg

Kernel Mode Debugging

- How many kernels on one OS?
- What will happen when kernel is at a breakpoint?
- Two computers for kernel mode debugging
  - Two different kernels
  - Enable kernel debugging

Windows 7 Advanced Boot Options

- Press F8 during startup
- "Debugging Mode"



单选题 10分

设置

Which item below must require two computers connected together?

A Source-level debugger

B Kernel-mode debugger

C User-mode debugger

D Assembly-level debuger

提交

单选题 10分

设置

Which item below is almost never used by malware analysts?

A Source-level debugger

B Assembly-level debugger

C User-mode debugger

D Kernel-mode debugger

提交

南開大學 Nankai University

Which item below is suggested by Windows automatically after it crashes?

A Source-level debugger

B Assembly-level debugger

C User-mode debugger

D Kernel-mode debugger

提交

Using a Debuggers

Two Ways

- Start the program with the debugger
  - It stops running immediately prior to the execution of its entry point
  - debug malicious code

Two Ways

- Attach a debugger to a program that is already running
  - All its threads are paused
  - debug a process that is affected by malware

Single-Stepping

- Run a single instruction and then return control to the debugger
- You can see everything
- Time consuming and tedious for complex code

- Focus on the big picture
- Do not get lost in the details

Example

- This code decodes the string with XOR

Example 9-1. Stepping through code

```
mov     edi, DWORD_00406904
mov     ecx, 0x0d
LOC_040106B2
xor     [edi], 0x9C
inc     edi
loopw   LOC_040106B2
...
DWORD:00406904:   F8FDF3D01
```

Example 9-2. Single-stepping through a section of code to see how it changes memory

```
D0F3FDF8 D0F5FEEE FDEEE5DD 9C (.............)
4CF3FDF8 D0F5FEEE FDEEE5DD 9C (L.............)
4C6FFDF8 D0F5FEEE FDEEE5DD 9C (Lo............)
4C6F61F8 D0F5FEEE FDEEE5DD 9C (Loa...........)
. . . SNIP . . .
4C6F6164 4C696272 61727941 00 (LoadLibraryA.)
```

Stepping-Over vs. Stepping-Into

- **Call instruction**
  - call a function, such as LoadLibrary
- **Stepping-over**
  - Complete the execution of the function
  - Stop at the instruction after call instruction
- **Stepping-into**
  - Stop at the first instruction of the called function

Stepping Out

- Stepping-Over
  - Risk of missing important functionality
  - Risk of never return
- Stepping-Into
  - Long nested call instructions
  - No relevance to what you are seeking
- Stepping-Out
  - Run until after the function returns

多选题 10分

设置

Which item might miss important functionality?

A single-step

B step-over

C step-into

D step-out

提交

南開大學 Nankai University

多选题 10分

设置

Which item will stop debugger at the first instruction of the called function?

A single-step

B step-into

C step-over

D step-out

提交

单选题 10分

设置

Which item will run after the function returns?

A single-step

B step-into

C step-out

D step-over

提交

单选题 10分

设置

Which item will stop at the first function after call instruction?

A single-step

B step-into

C step-over

D step-out

提交

南開大學 Nankai University

Pausing Execution with Breakpoints

- Breakpoints stop execution
  - Set breakpoints at interesting positions
- Example
  - You can't tell where this call is going
  - Set a breakpoint at the call and see what's in eax

Example 9-3. Call to EAX

```
00401008    mov     ecx, [ebp+arg_0]
0040100B    mov     eax, [edx]
0040100D    call    eax
```

- This code calculates a filename and then creates the file
- Set a breakpoint at CreateFileW and look at the stack to see the filename

Example 9-4. Using a debugger to determine a filename

```
0040100B  xor     eax, esp
0040100D  mov     [esp+0D0h+var_4], eax
00401014  mov     eax, edx
00401016  mov     [esp+0D0h+NumberOfBytesWritten], 0
0040101D  add     eax, 0FFFFFFFEh
00401020  mov     cx, [eax+2]
00401024  add     eax, 2
00401027  test    cx, cx
0040102A  jnz     short loc_401020
0040102C  mov     ecx, dword ptr ds:a_txt ; ".txt"
00401032  push    0                ; hTemplateFile
00401034  push    0                ; dwFlagsAndAttributes
00401036  push    2                ; dwCreationDisposition
00401038  mov     [eax], ecx
0040103A  mov     ecx, dword ptr ds:a_txt+4
00401040  push    0                ; lpSecurityAttributes
00401042  push    0                ; dwShareMode
00401044  mov     [eax+4], ecx
00401047  mov     cx, word ptr ds:a_txt+8
0040104E  push    0                ; dwDesiredAccess
00401050  push    edx              ; lpFileName
00401051  mov     [eax+8], cx
00401055  call    CreateFileW ; CreateFileW(x,x,x,x,x,x,x)
```

WinDbg

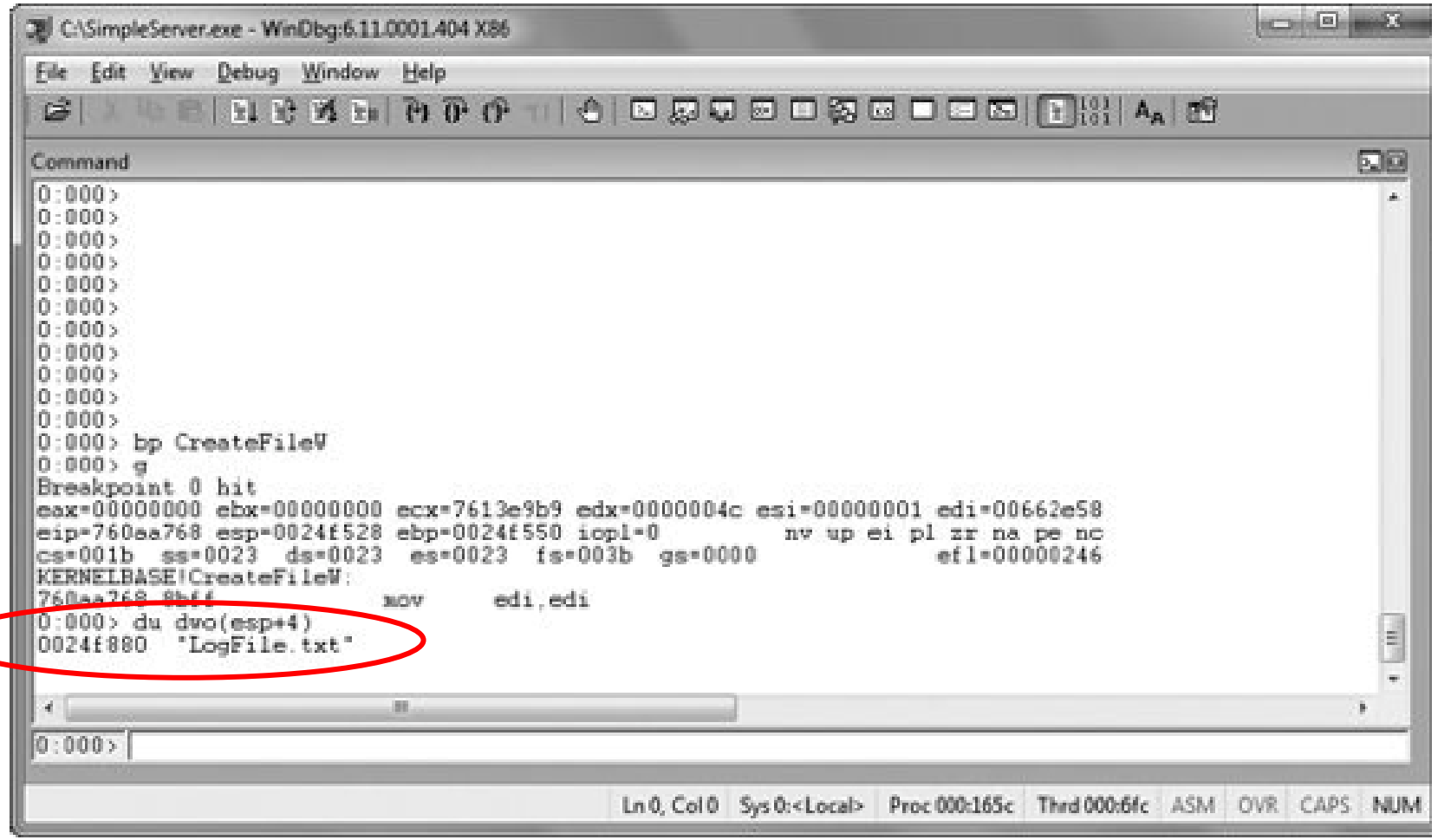


Figure 9-1. Using a breakpoint to see the parameters to a function call. We set a breakpoint on `CreateFileW` and then examine the first parameter of the stack.

Encrypted Data

- Suppose malware sends encrypted network data
- Set a breakpoint before the data is encrypted and view it

Example 9-5. Using a breakpoint to view data before the program encrypts it

```
004010D0  sub     esp, 0CCh
004010D6  mov     eax, dword_403000
004010DB  xor     eax, esp
004010DD  mov     [esp+0CCh+var_4], eax
004010E4  lea     eax, [esp+0CCh+buf]
004010E7  call    GetData
004010EC  lea     eax, [esp+0CCh+buf]
004010EF  call    EncryptData
004010F4  mov     ecx, s
004010FA  push    0               ; flags
004010FC  push    0C8h            ; len
00401101  lea     eax, [esp+0D4h+buf]
00401105  push    eax             ; buf
00401106  push    ecx             ; s
00401107  call    ds:Send
```

OllyDbg

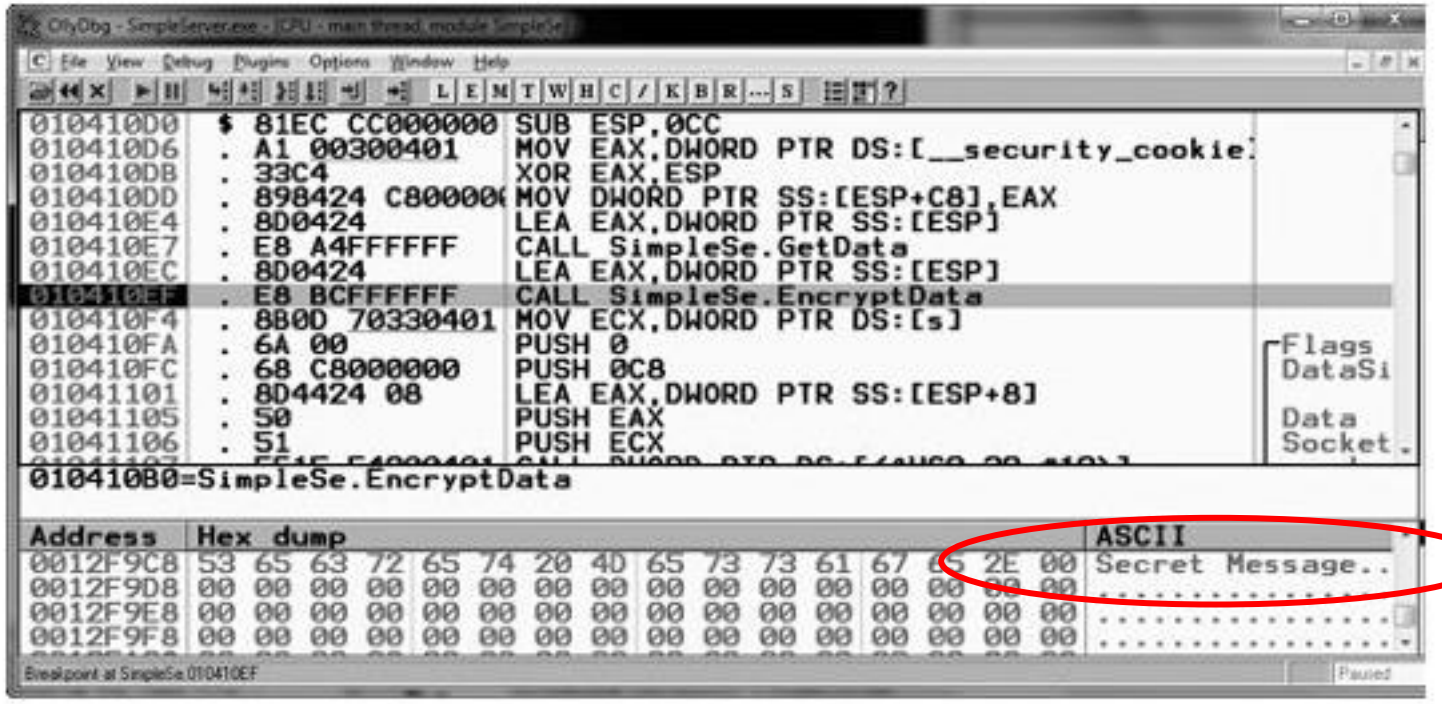


Figure 9-2. Viewing program data prior to the encryption function call

Types of Breakpoints

- Software execution breakpoint
- Hardware execution breakpoint
- Conditional breakpoint

Software Execution Breakpoints

- Set breakpoints at specified instructions
- Debugger overwrites the first byte of the instruction with 0xCC
  - The instruction for INT 3
  - An interrupt designed for use with debuggers
  - When the breakpoint is executed, the OS generates an exception and transfers control to the debugger

Memory Contents at a Breakpoint

- There's a breakpoint at the push instruction
- Debugger says it's 0x55, but it's really 0xCC

Table 9-1. Disassembly and Memory Dump of a Function with a Breakpoint Set

Disassembly view			Memory dump	
00401130 55	push	ebp	00401130 CC 8B EC 83	
00401131 8B EC	mov	ebp, esp	00401134 E4 F8 81 EC	
00401133 83 E4 F8	and	esp, 0FFFFFFF8h	00401138 A4 03 00 00	
00401136 81 EC A4 03 00 00	sub	esp, 3A4h	0040113C A1 00 30 40	
0040113C A1 00 30 40 00	mov	eax, dword_403000	00401140 00	

When Software Execution Breakpoints Fail

- If the 0xCC byte is changed during code execution, the breakpoint won't occur
- If other code reads the memory containing the breakpoint, it will read 0xCC instead of the original byte
- Code that verifies integrity will notice the discrepancy

Hardware Execution Breakpoints

- Four Hardware Debug Registers
  - DR0 through DR3 – addresses of breakpoints
  - DR7 stores control information
- if (EIP == Breakpoints) by hardware
- Can break on access or execution
  - Can set to break on read, write, or both
- No change in code bytes

Hardware Execution Breakpoints

- Running code can change the DR registers, to interfere with debuggers
- General Detect flag in DR7
  - Causes a breakpoint prior to any mov instruction that would change the contents of a Debug Register
  - Does not detect other instructions, however

Conditional Breakpoints

- Breaks only if a condition is true
  - Ex: Set a breakpoint on the GetProcAddress function
  - Only if parameter being passed in is RegSetValue
- Implemented as software breakpoints
  - The debugger always receives the break
  - If the condition is not met, it resumes execution without alerting the user

Conditional Breakpoints

- Conditional breakpoints take much longer than ordinary instructions
- A conditional breakpoint on a frequently-accessed instruction can slow a program down

多选题 10分

设置

What type of breakpoint uses Interrupt #3?

A software breakpoint

B hardware breakpoint

C conditional breakpoint

D single step

提交

多选题 10分

设置

What type of breakpoint may make a program run slowly?

A software breakpoint

B hardware breakpoint

C conditional breakpoint

提交

多选题 10分

设置

What type of breakpoint changes the binary code?

A software breakpoint

B hardware breakpoint

C conditional breakpoint

提交

多选题 10分

设置

What type of breakpoint uses the DR registers?

A software breakpoint

B hardware breakpoint

C conditional breakpoint

提交

南開大學 Nankai University

单选题 10分

设置

做多设置几个硬件中断?

A 1

B 2

C 4

D 8

提交

南開大學 Nankai University

Exceptions

Exceptions

- What is the exception?
  - Error?
  - Fault?
- How many different exceptions?

Exceptions

- Used by debuggers to gain control of a running program

- Breakpoints generate exceptions
  - Software BP
  - Hardware BP
  - Conditional BP

Exception

- Exceptions are also caused by
  - Access violation
  - Division by zero
  - Stack overflow

First- and Second-Chance Exceptions

- Debuggers are usually given two opportunities to handle the same exception
  - First-chance exception
  - Second-chance exception

First- and Second-Chance

- When a exception occurs while a debugger is attached
  - The program stops executing
  - The debugger is given **first chance** at control
  - Debugger can either handle the exception, or pass it on to the program
  - If it's passed on, the program's exception handler SEH takes it

Second Chance

- If the application doesn't handle the exception
- The debugger is given a **second chance** to handle it
  - This means the program would have crashed if the debugger were not attached

Second Chance

- In malware analysis, first-chance exceptions can usually be ignored
- Second-chance exceptions cannot be ignored
  - They usually mean that the malware doesn't like the environment in which it is running

多选题 10分

设置

Which item below handles exceptions during normal program execution and debugger attached?

A First chance

B Second chance

C SEH

D INT 3

提交

主观题 10分

设置

When analyzing malware, why the first-chance exceptions can often be ignored?

正常使用主观题需2.0以上版本雨课堂

作答

主观题 10分

设置

In malware analysis, why the second chance exception cannot be ignored?

正常使用主观题需2.0以上版本雨课堂

作答

Common Exceptions

- INT 3: Software Breakpoint Trap
  - 0xCC
- INT 1: Single-stepping Trap
  - If the **trap flag** in the flags register is set
  - The processor executes one instruction and then generates an exception

Common Exception

- Memory-access violation exception
  - Code tries to access a location that it cannot access, either because the address is invalid or because of access-control protections

Common Exceptions

- Violating Privilege Rules
  - Attempt to execute a kernel mode instruction in user mode
  - Attempt to execute Ring 0 instruction from Ring 3

List of Exceptions

The following chart lists the exceptions that can be generated by the Intel 80286, 80386, 80486, and Pentium processors:

Exception (dec/hex)	Description
0 00h	Divide error: Occurs during a DIV or an IDIV instruction when the divisor is zero or a quotient overflow occurs.
1 01h	Single-step/debug exception: Occurs for any of a number of conditions: - Instruction address breakpoint fault - Data address breakpoint trap - General detect fault - Single-step trap - Task-switch breakpoint trap
2 02h	Nonmaskable interrupt: Occurs because of a nonmaskable hardware interrupt.
3 03h	Breakpoint: Occurs when the processor encounters an INT 3 instruction.

设置

Which item could cause an exception?

A Access violation

B Division by zero

C Stack overflow

D Breakpoint

提交

多选题 10分

设置

Which item is used for single-stepping?

A first chance

B second chance

C SEH

D INT 3

E trap flag

提交

多选题 10分

设置

Which type of exception usually is ignored for malware analysis?

A First chance

B Second chance

C SEH

D INT 3

E trap flag

提交

多选题 10分

设置

A ring 3 process tries to access hardware directly. What exception will be thrown?

A First chance

B Second chance

C /0

D Invalid memory access

E Privilege violation

提交

Modifying Execution with a Debugger

Modifying Execution with a Debugger

- Why we have to modify the malware execution?
  - Skipping a function
  - Testing a function

Skipping a Function

- We can change control flags, the instruction pointer, or the code itself
- We can avoid a function call by setting a breakpoint where at the call, and then changing the instruction pointer to the instruction after it
  - This may cause the program to crash or malfunction, or course

Testing a Function

- You could run a function directly, without waiting for the main code to use it
  - You will have to set the parameters
  - This destroys a program's stack
  - The program won't run properly when the function completes

设置

Using binary modification, which operation we could do?

A skip a function

B test a function

C software cracking

D code reuse

提交

Modifying Program Execution in Practice

Real Virus

- Operation depends on language setting of a computer
  - Simplified Chinese
    - Uninstalls itself & does no harm
  - English
    - Display pop-up "Your luck's no good"
  - Japanese or Indonesian
    - Overwrite the hard drive with random data

Break at 1; Change Return Value

Example 9-6. Assembly for differentiating between language settings

```
00411349   call    GetSystemDefaultLCID
0041134F   1mov    [ebp+var_4], eax
00411352   cmp     [ebp+var_4], 409h          409 = English
00411359   jnz     short loc_411360
0041135B   call    sub_411037
00411360   cmp     [ebp+var_4], 411h          411 = Japanese
00411367   jz      short loc_411372
00411369   cmp     [ebp+var_4], 421h          421 = Indonesian
00411370   jnz     short loc_411377
00411372   call    sub_41100F
00411377   cmp     [ebp+var_4], 0C04h         C04 = Chinese
0041137E   jnz     short loc_411385
00411380   call    sub_41100A
```

Internal Logic

- English, loc_411360
- Japanese, loc_41372
- Indonesian, loc_411377
- Chinese, loc_411385

设置

How to force the malicious code to run different path without changing the settings on our system?

A Modify instruction pointer

B Change Windows API return value

C Breakpoint

D Set trap flag

提交

Conclusion

Conclusion

- Debugging
  - obtaining running information
  - single-step what's happening internally
  - set breakpoint to see particular section of code
  - modify code to get additional information

Next Chapters

- Debugging
  - OllyDbg
  - WinDbg

Outline

- Source-Level vs. Assembly-Level Debuggers
- Kernel vs. User-Mode Debugging
- Using a Debugger
- Exceptions
- Modifying Execution with a Debugger

恶意代码分析与防治技术

第8章 动态调试

王志
zwang@nankai.edu.cn

updated on 2022-11-6

南开大学 网络空间安全学院
2022/2023