Reverse Engineering: Towards Malware Analysis

Lecture – Debugging

Computer Security Practice

Outline

- Why Debuggers?
- Stepping
- Breakpoints
- Exceptions
- Tool in practice

Debuggers

- Test and examine the flow of a program
- Software devs - identify bugs and errors
- Helps to see how programs produce output and examine the internal state during execution
- Provide information that couldn't be gained from disassembly
  - Every memory location
  - Register
  - Arguments, return value of every function
  - (bonus) allows you to change them

Kernel Mode Vs. User Mode

- More challenging to debug at the kernel level
  - Kernel mode typically requires two systems
- In user mode, the debugger is allowed to run on the same OS as the code being debugged
- The OS must be configured to do kernel debugging
- WinDbg is currently the only popular tool for kernel debugging

gdb

- OllyDbg is the most popular user-mode debugger.
  - IDA Pro can do some basic debugging although it is a disassembler.

Source-Level Vs. Assembly-Level

- Most developers are familiar with source-level Dbg
  - Allow debugging during C++ programming
  - Typically comes with an IDE

VScode
QT Creator

- Assembly-level Debugger
  - Operates on assembly code (does not require source code)
  - Malware analysts don't have access to source code. So, this fits well into malware analysis!
  - Sometimes called low-level debugger
- Both allow for break points and stepping through the code 1 line at a time

Single Stepping

- Debuggers offer the ability to step through programs 1 line at a time
  - Should only be done for understanding sections of code, as stepping through an entire program would take a lot of time
  - Be selective of the code you single step through
  - "*Focus on the big picture, or you'll get lost in the details*"

Debugging: Options to stop, control and examine

Stepping Over Vs. Stepping Into

- Stepping into means you follow a function call and step through it as it runs
- Stepping over means you let the function run, but the next instruction you see is what is after the function returns
  - Some functions never return
- Stepping over helps save time, as you don't have to analyze functions you may not care about, such as normal OS functionality

Breakpoints

- Used to pause execution an allows for examining a program’s state
  - Can view register, memory and stack contents
  - Helps to see values that you can’t see in a disassembler
  - Can also help to view contents of encrypted data, by setting a breakpoint before the encrypting routine to see what is being passed

Breakpoint Example

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

- Set a breakpoint at 0x0040100D and view EAX

Breakpoint Example 2

```
• 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)
```

Debugging: Types of Breakpoints

Breakpoint Types

- **Software Execution Breakpoints** – stops a program when a particular instruction is executed. This is typically the default breakpoint
- **Hardware Execution Breakpoints** – Every time the processor executes an instruction, this detects if the instruction pointer is equal to the breakpoint address. It breaks based on address location regardless of what is stored there
  - Can break on access as opposed to execution

Breakpoint Types

- **Conditional Breakpoints** – break only if a certain condition is true
  - Implemented as software breakpoints
  - If the condition is not met, it automatically continues running instructions
- These can significantly slow program execution if not set properly

INT3

Software Breakpoints Illustrated

- Dissassembly

```
55                  push    ebp
8B EC               mov     ebp, esp
83 E4 F8            and     esp, 0FFFFFFF8h
81 EC A4 03 00 00   sub     esp, 3A4h
A1 00 30 40 00      mov     eax, dword_403000
```

C

M/C code

As code

INT0 — 16

- Same bytes dumped in memory

```
CC 8B EC 83
E4 F8 81 EC
A4 03 00 00
A1 00 30 40
00
```

CC

CPU

stop

Breakpoints
Syscalls
Errors

Interrupts

reg
mem

Debugging: Exceptions

Exceptions

- Typically how a debugger gains control of a running program
- Breakpoints cause exceptions, but program errors do as well
- There is functionality in place to allow the debugger and the program being debugged to both use exceptions

First and Second Chance Exceptions

- Debuggers are usually given 2 chances to handle an exception
- When an exception occurs, the program stops execution and the debugger is given the 1st chance at control
  - The debugger can handle or pass to the program
- If the program has a registered exception handler, it is given the chance to handle the exception
- If the program doesn't handle the exception, the debugger is given a 2nd chance to handle

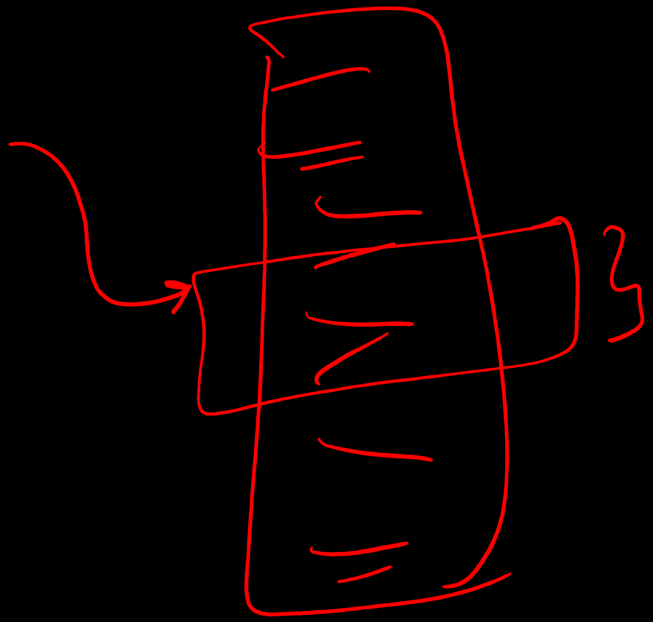
1st

reg

reg

Common Exceptions

- Most common occurs when the `INT 3` instruction is executed
  - Debuggers have special code to handle these, and the OS treats it as any other exception
- Single stepping uses the Trap flag, which is a flag in the flags register and is treated as an exception within the OS
- Memory access violations are exceptions that occur when the program attempts to access a memory address that it can't
  - Typically invalid memory address
- Certain instructions can only be carried out in kernel mode, and the processor generates an exception if they are attempted to be carried out in user mode



Debugging: Modifying control flow of program

Modifying Execution

- Debuggers can be used to modify program execution, and allow changing values such as the instruction pointer, flags and function calls
- This is useful for things like skipping encryption or obfuscation functions

A Real Virus

- Apparently written by a Russian group.
- Operations depend on the language settings:
  - Simplified Chinese
    - Uninstall itself
  - English
    - Displays a pop up “Your luck is no good”
  - Japanese or Indonesian
    - Overwrites hard drive with random data

Modifying Execution Example

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

lang pref

ger

eng

jz

english

Jap

indo

chinese

i=0

i = ~~0~~ 1

IP

?