

# Debugging Concepts and Tools

Aaron Sedlacek

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 100
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
sub_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Agenda

## Kernel land vs User land Debugging

### General Debugger Usage

### WinDbg

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Kernel Mode VS User Mode

- Debugging kernel mode code usually requires two different systems
  - One running the debugger, and the other running the kernel to be debugged
    - breakpoints in the kernel stop the entire system.
    - The target OS must also be configured to allow for kernel debugging
- We'll come back to this later in the semester

# Agenda

## Kernel land vs User land Debugging

### General Debugger Usage

### WinDbg

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Attaching to Processes

- You can either open executables or attach to the currently running processes

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], esi
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

# Single-Stepping

- Run a single instruction, then return control to the debugger
- Stepping into and stepping over
  - Stepping into lets you enter a function
    - Use as needed to save time
  - Pretty sure you don't want to step into printf or memcpy

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
call    [ebp+arg_0]
call    sub_31480a
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jnz     short loc_313066

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```



# Breakpoints

- Let you pause execution at specific points
  - See layout of memory at a specific point
- Software Breakpoints (traditional)
  - Your debugger inserts a **0xCC** instruction (**INT 3**)
    - This causes the OS to generate an exception and call the debugger
    - Malware has been known to check itself for integrity while it is executing, looking for things like **0xCC**'s

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

```
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31466A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
cmp eax, esi
jnz short loc_31308F
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push 0Dh
call sub_31466A
push esi
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

# Breakpoints

- Hardware Breakpoints

- Are tracked on the CPU
- Additionally, can be set to break on memory access
- Drawbacks:
  - Only four hardware registers store break addresses
    - DR0-DR3 store addresses, DR7 stores information about DR0-DR3
  - These registers can be modified by running programs
    - x86 can detect mov instructions that tamper with these registers

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

```
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_31306F:                                     ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push 0Dh
call sub_31411B
```

```
loc_313070:                                     ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3144F3
jnz short loc_31307D
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFh
or eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```



# Exceptions

- The way that a debugger gains control of a running program
- Programs regularly receive exceptions
  - Programs can register a function to be executed when a specific exception is received
- Attached debuggers grab exceptions first

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_4], eax
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
push    esi
push    [ebp+arg_4]
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jnz     short loc_313066
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

# Exceptions

- *First-chance Exceptions*

- Debuggers can either handle the exception or pass the exception to the program

- *Second-chance Exceptions*

- Exception is passed back to the debugger
- Created only when the program would have crashed, and must be handled

- Malware is infamous for using exceptions to subvert analysis

- We will cover this more later in the course!

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Patching - Modifying Execution

- Debuggers let you do this on the fly!
  - Great for changing how a program executes in order to analyze it
- You can change the contents of registers and memory
  - You can even change the instructions that a sample executes!
  - Note: It's particularly useful to flip the **ZF** and **SF** to control execution flow

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
short loc_31306E
eax, [ebp+var_70]
eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    eax
push    esi
push    esi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31306E
loc_313066:                                ; CODE XREF: sub_312FD8
; sub_312FD8+55
push    0Dh
call    sub_314118
loc_31306D:                                ; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
; -----
loc_31307D:                                ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
loc_31308C:                                ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

# Tracing

- The debugger can record actions that the program takes
  - Read Tracing
    - Memory read from
  - Write Tracing
    - Memory written to
  - Execution Tracing
    - Instructions a sample executes
- Particularly useful for samples that execute in intentionally convoluted ways

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jnz     short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Agenda

## Kernel land vs User land Debugging General Debugger Usage WinDbg

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```



# WinDbg

We'll do it Live!

He went to my high school lol →



```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

loc\_31307D: ; CODE XREF: sub\_312FD8

```
call    sub_3140F3
and     eax, 0FFFFFFFh
or      eax, 80070000h
```

loc\_31308C: ; CODE XREF: sub\_312FD8

```
mov     [ebp+var_4], eax
```



- The debugger can record actions that the program takes
  - Read Tracing
    - Memory read from
  - Write Tracing
    - Memory written to
  - Execution Tracing
    - Instructions a sample executes
- Particularly useful for samples that execute in intentionally convoluted ways

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

```

```

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

```

```

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55

```

```

push    0Dh
call    sub_31411B

```

```

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49

```

```

call    sub_3140F3
test    eax, eax
jnz     short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

```

```

loc_31307D:                                     ; CODE XREF: sub_312FD8

```

```

call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

```

```

loc_31308C:                                     ; CODE XREF: sub_312FD8

```

```

mov     [ebp+var_4], eax

```

# Questions?

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

```

```

loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55

```

```

push    0Dh
call    sub_31411B

```

```

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49

```

```

call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

```

```

; -----

```

```

loc_31307D:                                     ; CODE XREF: sub_312FD8

```

```

call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

```

```

loc_31308C:                                     ; CODE XREF: sub_312FD8

```

```

mov     [ebp+var_4], eax

```

# References

1. Sikorski, Michael, and Andrew Honig. Practical Malware Analysis the Hands-on Guide to Dissecting Malicious Software. San Francisco: No Starch, 2012. Print.

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    esi
mov     [ebp+arg_0], eax
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```