CMSC389R

Binaries II

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

recap

HW7

--

Questions?

Itinerary

- Review
- Reverse Engineering
  - Static analysis
  - Dynamic analysis
- Tools
- Exercises

Review

- x86 Assembly
  - Registers, instructions, conventions
- Tools
  - objdump, yasm, gdb

Calling Conventions

- Arguably one of the more important aspects when starting to learn reverse engineering
- Argument passing
  - rdi, rsi, rdx, rcx, r8, r9
- Function setup
  - Adjusting base/stack pointer
  - Where is return address stored?
  - How do we return?

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
	0x00405fe8	
rsp	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
	0x00405fe8	
rsp	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rsp	0x00405fe8	
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
rbp	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rbp rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rbp rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
rsp	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
rsp	0x00405fd0	
	0x00405fd8	
	0x00405fe0	
rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabcd
	0x00406000	1234
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
rsp	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
rsp	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rsp rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rsp rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rsp rbp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
rsp	0x00405fe8	0x00406000
	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
rbp	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
	0x00405fe8	0x00406000
rsp	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
rbp	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

	0x00405fb8	
	0x00405fc0	
	0x00405fc8	
	0x00405fd0	123
	0x00405fd8	456
	0x00405fe0	789
	0x00405fe8	0x00406000
rsp	0x00405ff0	MY RETURN ADDRESS
	0x00405ff8	0xabce
rbp	0x00406000	1235
	.....	
	0xffffffff	

```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```


```
my_func:
    push rbp
    mov rbp, rsp
    sub rsp, 24

    ...

    add rsp, 24
    leave
    ret
```

rip	MY RETURN ADDRESS

Static Analysis

- “lacking in movement, action, or change”
- Analyzing a binary without running it
- Useful for certain circumventions
  - Malware
  - Network access
  - System modifications

Static Analysis Tools

- objdump: disassembles binaries
  - *-D* will disassemble EVERYTHING
  - *-M intel* will output with Intel syntax

```
0000000000001200 <my_memset>:
    1200:       55                          push   rbp
    1201:       48 89 e5                    mov    rbp,rsp
    1204:       c9                          leave
    1205:       c3                          ret

0000000000001206 <my_strncpy>:
    1206:       55                          push   rbp
    1207:       48 89 e5                    mov    rbp,rsp
    120a:       c9                          leave
    120b:       c3                          ret
    120c:       0f 1f 40 00                 nop    DWORD PTR [rax+0x0]
```

Static Analysis Tools

- objdump: disassembles binaries
  - *-D* will disassemble EVERYTHING
  - *-M intel* will output with Intel syntax

```
0000000000001200 <my_memset>:
    1200:       55                          push   rbp
    1201:       48 89 e5                    mov    rbp,rsp
    1204:       c9                          leave
    1205:       c3                          ret

0000000000001206 <my_strncpy>:
    1206:       55                          push   rbp
    1207:       48 89 e5                    mov    rbp,rsp
    120a:       c9                          leave
    120b:       c3                          ret
    120c:       0f 1f 40 00                 nop    DWORD PTR [rax+0x0]
```

Static Analysis Tools

- readelf: view information about ELF files
  - ELF header
  - Section headers (.text .data .comment etc)
- *man readelf* for more details

Static Analysis Tools

- rea
  -
  -
- *man*

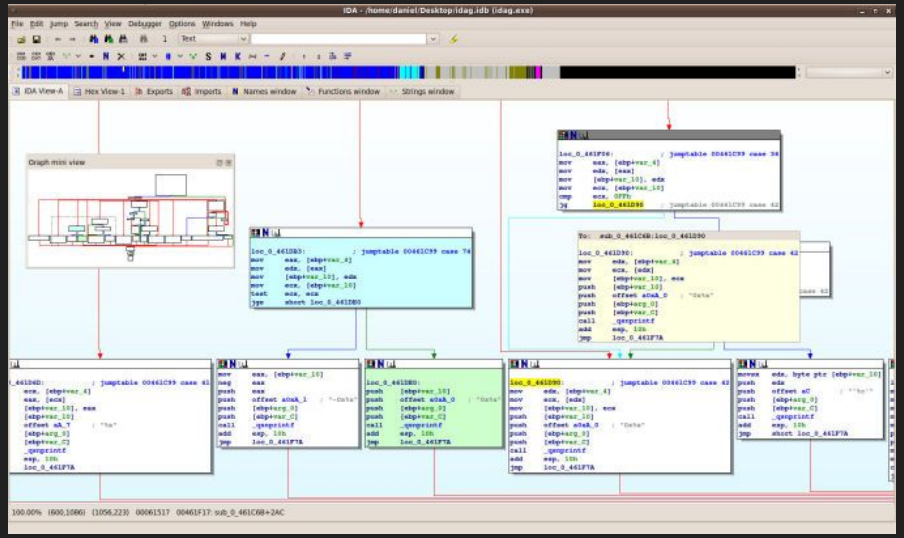
```
week/5 [ readelf -a main
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x1050
  Start of program headers:          64 (bytes into file)
  Start of section headers:          14864 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         11
  Size of section headers:           64 (bytes)
  Number of section headers:         29
  Section header string table index: 28
```

tc)

Static Analysis Tools

- strings: outputs human-readable strings

```
week/5 [ strings main
/lib64/ld-linux-x86-64.so.2
)H,=[
&~:e
libc.so.6
puts
__stack_chk_fail
__cxa_finalize
__libc_start_main
GLIBC_2.4
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u3UH
Hello WoH
rld!
rld!
```



```
[0x00003c9c 255 /usr/bin/r2]> pd $r @ sym..L94+4869 # 0x3c9c
            0x00003c9c    e970efffff   jmp 0x100002c11 ; (fcn.00002390) ;[1]
            0x00003ca1    8bbba4010000 mov edi, [ebx+0x1a4]
            0x00003ca7    8b74247c     mov esi, [esp+0x7c]
            0x00003cab    8b842494000. mov eax, [esp+0x94]
            0x00003cb2    c7442404000. mov dword [esp+0x4], 0x0
            0x00003cba    890424       mov [esp], eax
            0x00003cbd    e81ee2ffff   call 0x100001ee0 ; (sym.imp.r_core_prompt) ;[2]
               sym.imp.r_core_prompt()
            0x00003cc2    85c0         test eax, eax
            0x00003cc4    0f8eaa000000 jle 0x3d74 ;[3]
            0x00003cca    85f6         test esi, esi
            0x00003ccc    7408         jz 0x3cd6 ;[4]
            0x00003cce    893424       mov [esp], esi
            0x00003cd1    e84ae4ffff   call 0x100002120 ; (sym.imp.r_th_lock_enter) ;[5]
               sym.imp.r_th_lock_enter()
            0x00003cd6    8b942494000. mov edx, [esp+0x94]
            0x00003cdd    891424       mov [esp], edx
            0x00003ce0    e80be4ffff   call 0x1000020f0 ; (sym.imp.r_core_prompt_exec) ;[6]
               sym.imp.r_core_prompt_exec()
            0x00003ce5    8984249c000. mov [esp+0x9c], eax
            0x00003cec    83c001       add eax, 0x1
            0x00003cef    0f8424010000 jz 0x3e19 ;[7]
            0x00003cf5    85f6         test esi, esi
            0x00003cf7    7408         jz 0x3d01 ;[8]
```

```
4_t sub_40232c()
004023b4  mov     edx, 0x2
004023b9  mov     esi, 0x1
004023be  mov     edi, eax
004023c0  call    setsockopt
004023c5  cmp     eax, 0xffffffff
004023c8  jne     0x4023ec

004023ca  call    __errno_location
004023cf  mov     eax, dword [rax]
004023d1  mov     esi, eax
004023d3  mov     edi, 0x402d40  {"[ERROR] setsockopt() failed %x\n…"}
004023d8  mov     eax, 0x0
004023dd  call    printf
004023e2  mov     edi, 0xffffffff
004023e7

Define Name
Enter symbol name:
maybe_bind_socket
Cancel   OK

                  x, dword [rbp-0x14]
                  x, dword [rbp-0x18]
                  i, dword [rbp-0x4]
                  x, qword [rbp-0x20]
004023f9  mov     rdi, rax
004023fc  call    sub_40245e
00402401  test    eax, eax
00402403  je      0x40240f

00402405  mov     edi, 0xffffffff
0040240a  call    exit

0040240f  mov     eax, dword [rbp-0x4]
00402412  mov     esi, 0x14
00402417  mov     edi, eax
00402419  call    listen
0040241e  test    eax, eax
00402420  je      0x40242c

Options   Selection: 0x4023fc to 0x402401 (0x5 bytes)
```

Static Analysis Tools

- Disassemblers
  - IDA: “Interactive Disassembler”
    - Very expensive
    - State of the art, industry standard
  - Binary Ninja
    - Much cheaper
    - Fewer features than IDA, but fine if you’re only doing x86
  - radare2
    - Open-source (read: free)
    - Impressive features for a free product

IDA Pro

- Luckily there's a free trial for IDA Pro
- Download on your host OS, NOT in the VM
  - Better performance
  - Runs basically the same on all OSes
  - No need to shove it in a VM
- https://www.hex-rays.com/products/ida/support/download_freeware.shtml
  - *tar zxvf idafree70_mac.tgz* for macOS users
- Homework will require you to use IDA

IDA Basics

- Graph view -- NOT a text editor!
  - Pressing random keys is likely bad -- NO UNDO BUTTONS
  - Main view where most all work is done
- Hex view -- shows hex dump of program
- Structures/Enums -- lets you define struct/enum types to use in analysis
- Imports/exports -- shows functions used by the binary

IDA Basics

- Graph view
  - Clicking on a thing will highlight all uses of said thing in the current view
  - Double clicking on certain things (loc_##, variables, etc) may lead you to its definition
  - N -- renames highlighted value
  - X -- finds cross-references (xrefs) to highlighted value
  - ESC -- goes back to previous view
    - Views work like a stack, letting you go back to previous views as you go deep into a rabbit hole
  -

Key points

- Function argument order important
- Series of math instructions is usually the compiler trying to form a specific value efficiently
- Pointer sizes are based on architecture (64bit has 8byte ptrs)
- [rbp+*var_##*] grabs local variables
- Differences between var_## and var_$$ can help deduce type of data being worked with
- Renaming as many things before analysis helps a lot
  - Sometimes you can’t rename things until you start analysis, and that’s fine
- jz/jnz is often used the same as je/jne

Dynamic Analysis

- “stimulates change or progress”
- Analyzing a binary by running it
  - May be too complex to comprehend statically
  - May exhibit unique behavior based on environment in which it executes
- Behavioral Analysis
  - Flag obfuscation? No worries!
  - Breakpoint at strcmp, examine memory

Dynamic Analysis Tools

- gdb: your C debugger
  - Surprise! Most reverse engineers use this
  - Very powerful if you know what you’re looking for
  - Scriptable
- angr: programmatically interact with binaries
  - Symbolically execute binaries
  - Override function behavior at runtime
  - Many more things to do!

homework #11

will be posted soon.

Let us know if you have any questions!