Week 5: Reverse Engineering Part 2

Nobel Gautam
OSIRIS Lab Hacknight

OSIRIS

Objectives

- Setting up IDA
- Navigating IDA
- Reversing a binary
- Rename variables
- Define structs
- X-refs
- Shortcuts
- Demo

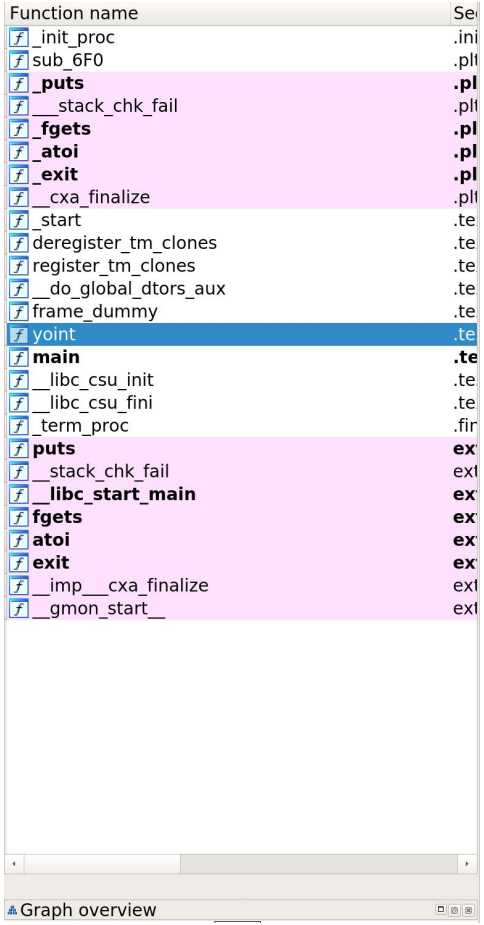
Setting Up IDA

Download link: https://www.hex-rays.com/products/ida/support/download_freeware.shtml

Sample Binary: https://github.com/osirislab/Hack-Night/tree/master/Rev/static

Navigating IDA

- On the left side, there is a list of functions
- Look for the entrypoint: “start” or “main”
    - Click the list of functions and type out function name to search
- Use the graph view! It helps with understanding program logic



Navigating IDA: Views

Top bar lists views

- IDA View: The disassembly! :D
- Hex View: The raw bytes of the program along with ASCII equivalent (if possible) on the right
- Structures: Allows you to define structures/types to make disassembly more readable
- Imports: List of functions imported by the program (typically libc)
- Exports: Functions defined by the program

Reversing a Binary

- Run the program to see what it does
- Take advantage of symbols (if available) and graph view to get a basic understanding.
- Pay attention to variables you control! Useful bugs will manifest here.
- With integers, you want to look for overflows or mistyped conversions.
- With strings, you want to look for out of bound read/writes.
- Look for common mistakes such as off-by-ones!

Reversing a Binary Checklist

- Look for the entrypoint (main, start)
- Identify interesting functions
- Examine areas where program branches
- Follow your input through
- Look for common bugs such as off by one, use after free, etc.
- Try to find ways to reach new branches
- You don’t always need to reverse the entire binary if you find the bad function.

Renaming Variables

n: Rename variables

Name it whatever, follow it, and rename to something sensible once you get a better understanding.

y: Change type

Often IDA will treat pointers as numeric types. You can fix that! And other stuff.

Define Structs

See something like

```
mov     cs:b, 0
mov     cs:dword_60102C, 1
mov     cs:dword_601030, 2
```

That's probably a struct. You can use the Structure view to define structures, to make these easier to read. We can infer that the stuff it's reading is 4 bytes here, so we define 4-byte variables at these offsets.

Define Structs

```
00000000 ; D/A/*   : create structure member (data/ascii/array)
00000000 ; N       : rename structure or structure member
00000000 ; U       : delete structure member
00000000 ; [00000018 BYTES. COLLAPSED STRUCT Elf64_Sym. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; ---------------------------------------------------------------------------
00000000
00000000 structy         struc ; (sizeof=0xC, mappedto_4)
00000000                                         ; XREF: .data:b/r
00000000 member_1        dd ?
00000004 member_2        dd ?
00000008 member_3        dd ?
0000000C structy         ends
0000000C
00000000 ; [00000018 BYTES. COLLAPSED STRUCT Elf64_Rela. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT Elf64_Dyn. PRESS CTRL-NUMPAD+ TO EXPAND]
```

Define Structs

```
mov     cs:b.member_1, 0
mov     cs:b.member_2, 1
mov     cs:b.member_3, 2
```

“y” or “t” to change type

X-Refs (Cross-References)

A function can be referenced by several other functions.

We see a bug in `func` and want to find out how we can get it called. What do we do? `x`!
IDA can also show a graph of xrefs to and from a function, which can be a nice visual aid.

IDA Shortcuts

https://www.hex-rays.com/products/ida/support/freefiles/IDA_Pro_Shortcuts.pdf

Notable:

- Changing types (y)
- Switch data length (d)
- Jump to address (g<addr>)
- Jump to function (g<name>, search in function list, etc. etc.)
- Rename (n)
- Xrefs (x)

Workshop

https://github.com/osirislab/Hack-Night/tree/master/Rev/static

Practice Challenge

http://wargames.osiris.cyber.nyu.edu/

We'll go over the challenge next week