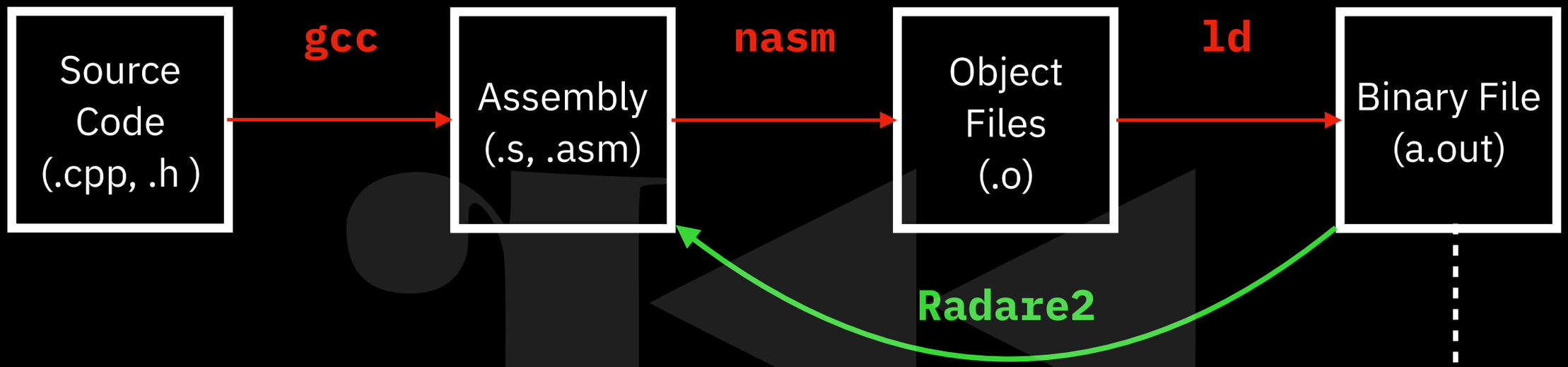
Evil Reverse Engineering with Radare2

By Derek Rodriguez

Radare2 is a…

- Scripting Environment
- Disassembler
- Digital Forensics Tool
- Hex Calculator
- In-line Assembler
- …Retro VideoGame IDE?

Reverse Engineering 101




```
0x100000b60      55              push rbp                    ; section 0 va=0x100000b60
0x100000b61      4889e5          mov rbp, rsp
0x100000b64      4883ec40        sub rsp, 0x40               ; '@'
0x100000b68      c745fc000000.   mov dword [local_4h], 0
0x100000b6f      897df8          mov dword [local_8h], edi
;-- rip:
0x100000b72      488975f0        mov qword [local_10h], rsi
0x100000b76      c745ec020000.   mov dword [local_14h], 2
0x100000b7d      c745e8060000.   mov dword [local_18h], 6
0x100000b84      c745e4000000.   mov dword [local_1ch], 0
0x100000b8b      488b75f0        mov rsi, qword [local_10h]
0x100000b8f      488b7608        mov rsi, qword [rsi + 8]    ; [0x8:8]=-1 ; 8
0x100000b93      0fbe3e          movsx edi, byte [rsi]
0x100000b96      488b75f0        mov rsi, qword [local_10h]
0x100000b9a      488b7608        mov rsi, qword [rsi + 8]    ; [0x8:8]=-1 ; 8
0x100000b9e      0fbe460a        movsx eax, byte [rsi + 0xa] ; [0xa:1]=255 ; 10
0x100000ba2      29c7            sub edi, eax
0x100000ba4      4088f9          mov cl, dil
0x100000ba7      884de3          mov byte [local_1dh], cl
0x100000baa      488b75f0        mov rsi, qword [local_10h]
0x100000bae      488b7608        mov rsi, qword [rsi + 8]    ; [0x8:8]=-1 ; 8
```

```
0x100000b60  5548 89e5 4883 ec40 c745 fc00 0000 0089
0x100000b70  7df8 4889 75f0 c745 ec02 0000 00c7 45e8
0x100000b80  0600 0000 c745 e400 0000 0048 8b75 f048
0x100000b90  8b76 080f be3e 488b 75f0 488b 7608 0fbe
0x100000ba0  460a 29c7 4088 f988 4de3 488b 75f0 488b
0x100000bb0  7608 0fbe 4601 488b 75f0 488b 7608 0fbe
0x100000bc0  7e09 29f8 88c1 884d e248 8b75 f048 8b76
0x100000bd0  080f be46 0248 8b75 f048 8b76 080f be7e
0x100000be0  0829 f888 c188 4de1 488b 75f0 488b 7608
0x100000bf0  0fbe 4603 488b 75f0 488b 7608 0fbe 7e07
0x100000c00  29f8 88c1 884d e048 8b75 f048 8b76 080f
```

Applications in Offensive Security

- DRM can be rendered ineffective through control flow manipulation

- Software updates can be reversed to understand update releases

- Exploits can be identified by tracing syscalls and library functions.

The Pirate Bay

DENUVO

By exploring the control flow of the program, we can decide a plan of attack for circumventing the authentication system

```
[0x100000ed0]> VV @ entry0 (nodes 7 edges 8 zoom 100%) BB-NORM mouse:canvas-y movements-speed:5

  0x100000f01 ;[ga]
      ; JMP XREF from 0x100000eea (entry0)
mov rax, qword [local_10h]
    ; [0x8:8]=-1
    ; 8
mov rax, qword [rax + 8]
mov qword [local_18h], rax
mov rdi, qword [local_18h]
call sym._superSecurePasswordCheck;[ge]
cmp rax, 0
je 0x100000f29;[gf]

f t

  0x100000f1f ;[gi]
call sym._triggerSuccessState;[gg]
jmp 0x100000f2e;[gh]

  0x100000f29 ;[gf]
      ; JMP XREF from 0x100000f19 (entry0)
call sym._indicateInvalidKey;[gj]

  0x100000f2e ;[gh]
      ; JMP XREF from 0x100000f24 (entry0)
xor eax, eax
add rsp, 0x20
pop rbp
ret
```

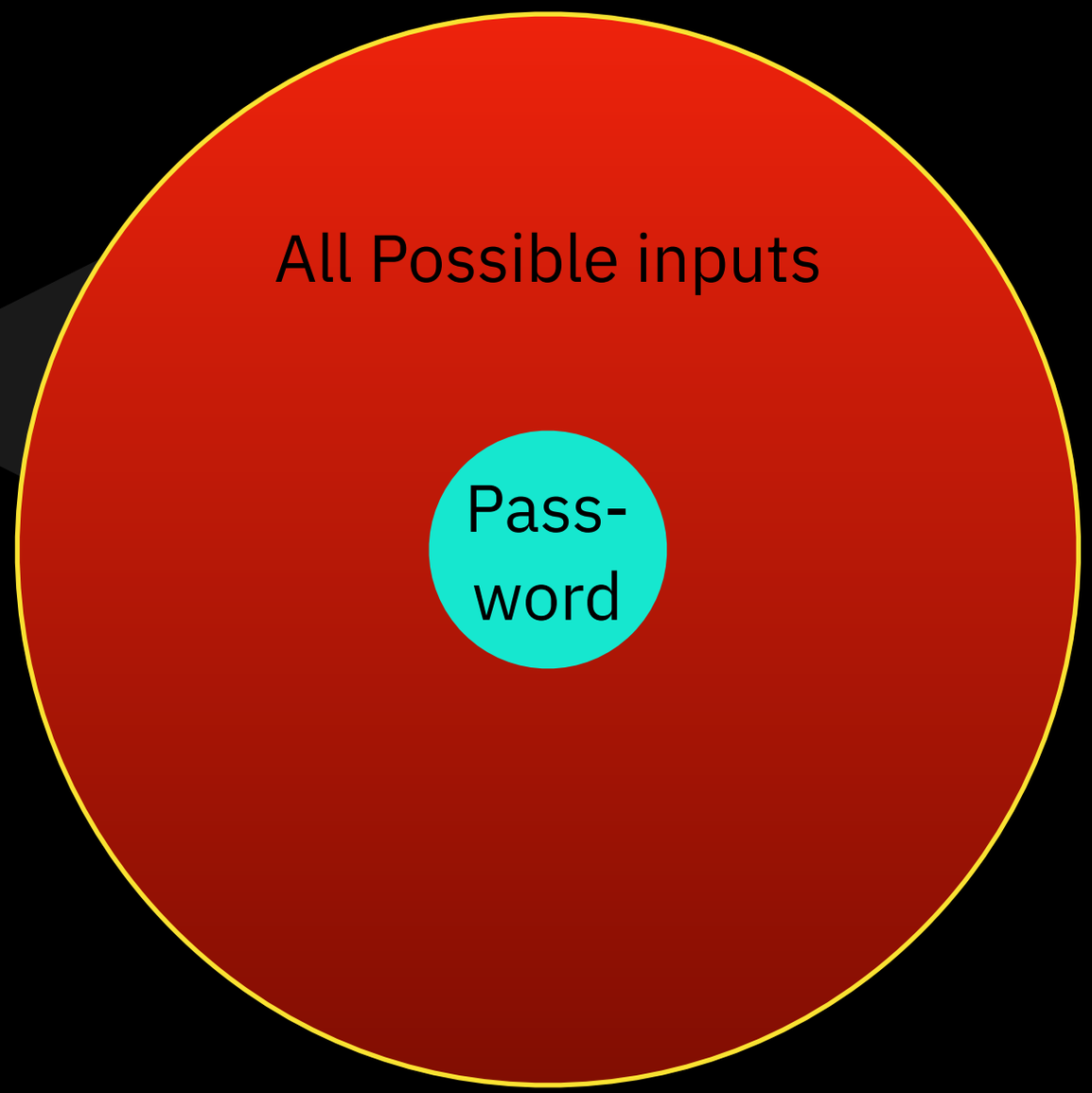
This jump instruction guards our success state, let's break it

```
[0x100000ed8 41% 205 (0x42:-1=1)]> pd $r @ entry0+74
|           0x100000ed8      c745fc000000.  mov dword [local_4h], 0
|           0x100000edf      897df8         mov dword [local_8h], edi
|           0x100000ee2      488975f0       mov qword [local_10h], rsi
|           0x100000ee6      837df802       cmp dword [local_8h], 2    ; [0x2:4]=-1 ; 2
|       ,=< 0x100000eea      0f8d11000000   jge 0x100000f01            ;[1]
|       |   0x100000ef0      488d3da80000.  lea rdi, str.Usage:_easy__password_ ; 0x100000f9f ; "Usage: easy [password]"
|       |   0x100000ef7      b000           mov al, 0
|       |   0x100000ef9      e838000000     call sym.imp.printf         ;[2] ; int printf(const char *format)
|       |   0x100000efe      8945e4         mov dword [local_1ch], eax
|       |       ; JMP XREF from 0x100000eea (entry0)
|       `-> 0x100000f01      488b45f0       mov rax, qword [local_10h]
|           0x100000f05      488b4008       mov rax, qword [rax + 8]   ; [0x8:8]=-1 ; 8
|           0x100000f09      488945e8       mov qword [local_18h], rax
|           0x100000f0d      488b7de8       mov rdi, qword [local_18h]
|           0x100000f11      e84affffff     call sym._superSecurePasswordCheck ;[3]
|           0x100000f16      83f800         cmp eax, 0
|       ,=< 0x100000f19   1  0f850a000000   jne 0x100000f29            ;[4]
|       |   0x100000f1f      e86cffffff     call sym._triggerSuccessState ;[5]
|       ,==< 0x100000f24      e905000000     jmp 0x100000f2e            ;[6]
|       ||      ; JMP XREF from 0x100000f19 (entry0)
|       |`-> 0x100000f29      e882ffffff     call sym._indicateInvalidKey ;[7]
|       |       ; JMP XREF from 0x100000f24 (entry0)
|       `--> 0x100000f2e      31c0           xor eax, eax
|           0x100000f30      4883c420       add rsp, 0x20
|           0x100000f34      5d             pop rbp
\           0x100000f35      c3             ret
            ;-- section_end.0.__TEXT.__text:
            ;-- section.1.__TEXT.__stubs:
/ (fcn) sym.imp.printf 6
|   sym.imp.printf ();
|              ; CALL XREF from 0x100000ef9 (entry0)
|              ; CALL XREF from 0x100000ec1 (sym._indicateInvalidKey)
|              ; CALL XREF from 0x100000ea1 (sym._triggerSuccessState)
\           0x100000f36      ff25d4000000   jmp qword [0x100001010]    ; section.6.__DATA.__la_symbol_ptr ; [0x100001010:8]
/ (fcn) sym.imp.strcmp 6
|   sym.imp.strcmp ();
|              ; CALL XREF from 0x100000e77 (sym._superSecurePasswordCheck)
\           0x100000f3c      ff25d6000000   jmp qword [0x100001018]    ; [0x100001018:8]=0x100000f5e
            ;-- section_end.1.__TEXT.__stubs:
            0x100000f42      ff             invalid
```

This highlighted instruction guards our desired state, change `je` to `jne`

So What did this do?

- Previously, of all possible inputs, we needed to find the input that met certain condition. Now it accepts all inputs that don’t meet those conditions.



Conclusions

- Enjoyable to use… if you like CLI programs and HJKL for navigation.
- Help command is amazing for learning the software
- Very unique niche in R.E. market (it’s free)
- Has some problems with run-time debugging on macOS
- Editing in debugger mode only edits the image in memory, not the file
- R2pipe opens up possibilities for advanced dynamic analysis, logic libraries like Z3.