Basic Tools

yuawn

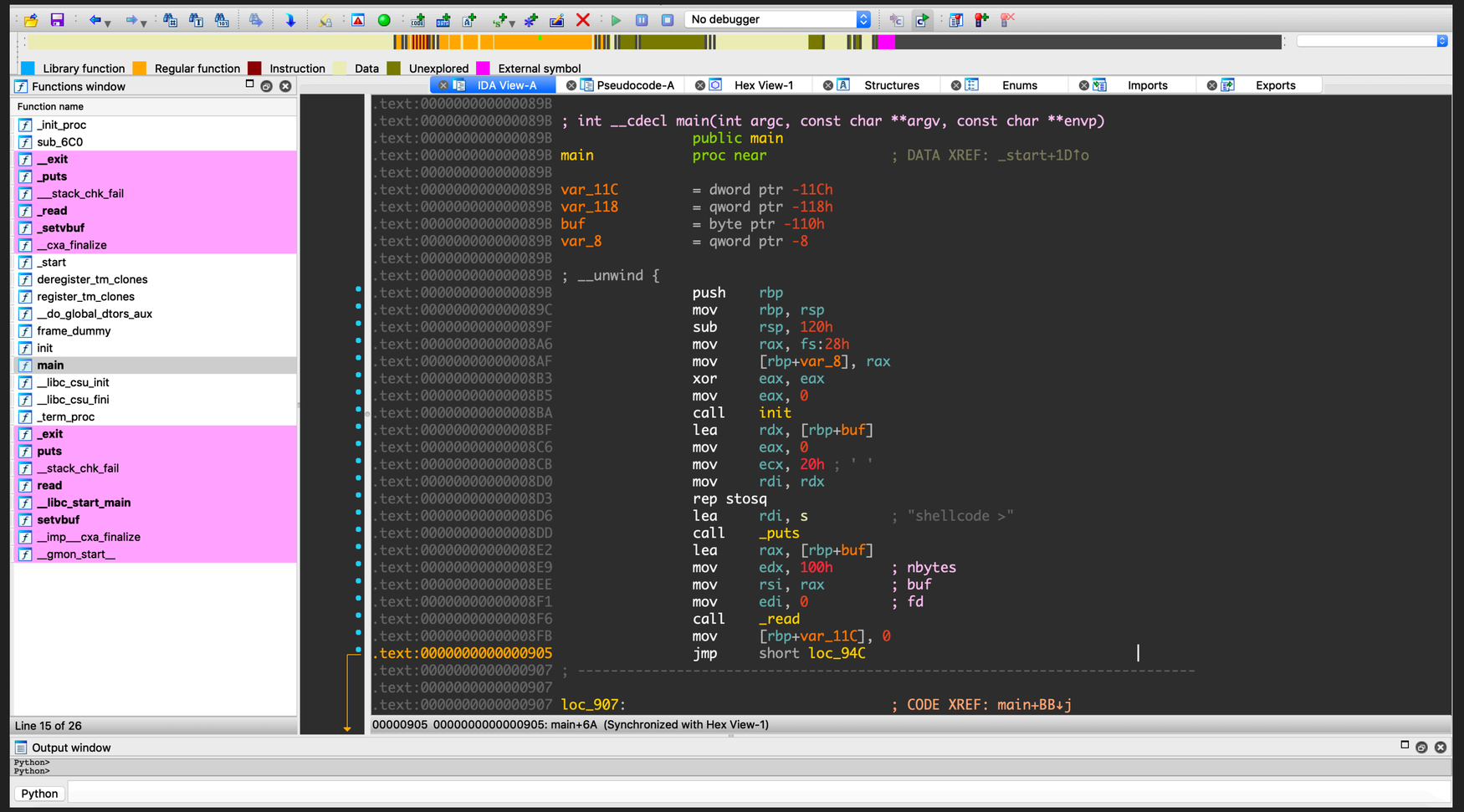
Reverse Engineering

- Static Analysis
  - strings
  - objdump, readelf
- Dynamic Analysis
  - gdb
  - ltrace, strace
  - ghidra, Radare2

```
 8ca:   48 8b 05 6f 07 20 00    mov    rax,QWORD PTR [rip+0x20076f]
 8d1:   b9 00 00 00 00          mov    ecx,0x0
 8d6:   ba 02 00 00 00          mov    edx,0x2
 8db:   be 00 00 00 00          mov    esi,0x0
 8e0:   48 89 c7                mov    rdi,rax
 8e3:   e8 78 fe ff ff          call   760 <setvbuf@plt>
 8e8:   90                      nop
 8e9:   5d                      pop    rbp
 8ea:   c3                      ret

00000000000008eb <main>:
 8eb:   55                      push   rbp
 8ec:   48 89 e5                mov    rbp,rsp
 8ef:   48 81 ec 20 01 00 00    sub    rsp,0x120
 8f6:   64 48 8b 04 25 28 00    mov    rax,QWORD PTR fs:0x28
 8fd:   00 00
 8ff:   48 89 45 f8             mov    QWORD PTR [rbp-0x8],rax
 903:   31 c0                   xor    eax,eax
 905:   b8 00 00 00 00          mov    eax,0x0
 90a:   e8 7b ff ff ff          call   88a <init>
 90f:   48 8d 85 f0 fe ff ff    lea    rax,[rbp-0x110]
 916:   ba 00 01 00 00          mov    edx,0x100
 91b:   be cc 00 00 00          mov    esi,0xcc
 920:   48 89 c7                mov    rdi,rax
 923:   e8 18 fe ff ff          call   740 <memset@plt>
 928:   48 8d 3d 55 01 00 00    lea    rdi,[rip+0x155]        # a84
 92f:   e8 ec fd ff ff          call   720 <puts@plt>
 934:   48 8d 85 f0 fe ff ff    lea    rax,[rbp-0x110]
 93b:   ba 00 01 00 00          mov    edx,0x100
 940:   48 89 c6                mov    rsi,rax
 943:   bf 00 00 00 00          mov    edi,0x0
 948:   e8 03 fe ff ff          call   750 <read@plt>
 94d:   c7 85 e4 fe ff ff 00    mov    DWORD PTR [rbp-0x11c],0x0
 954:   00 00 00
 957:   eb 59                   jmp    9b2 <main+0xc7>
 959:   8b 85 e4 fe ff ff       mov    eax,DWORD PTR [rbp-0x11c]
 95f:   48 98                   cdqe
 961:   0f b6 84 05 f0 fe ff    movzx  eax,BYTE PTR [rbp+rax*1-0x110
 968:   ff
 969:   84 c0                   test   al,al
 96b:   74 28                   je     995 <main+0xaa>
 96d:   8b 85 e4 fe ff ff       mov    eax,DWORD PTR [rbp-0x11c]
 973:   48 98                   cdqe
```

IDA Pro


No debugger
Library function
Regular function
Instruction
Data
Unexplored
External symbol
Functions window
Function name
_init_proc
sub_6C0
__exit
_puts
__stack_chk_fail
_read
_setvbuf
__cxa_finalize
_start
deregister_tm_clones
register_tm_clones
__do_global_dtors_aux
frame_dummy
init
main
__libc_csu_init
__libc_csu_fini
_term_proc
_exit
puts
__stack_chk_fail
read
__libc_start_main
setvbuf
__imp___cxa_finalize
__gmon_start__
Line 15 of 26
IDA View-A
Pseudocode-A
Hex View-1
Structures
Enums
Imports
Exports
.text:000000000000089B
.text:000000000000089B ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:000000000000089B public main
.text:000000000000089B main proc near ; DATA XREF: _start+1D↑o
.text:000000000000089B
.text:000000000000089B var_11C = dword ptr -11Ch
.text:000000000000089B var_118 = qword ptr -118h
.text:000000000000089B buf = byte ptr -110h
.text:000000000000089B var_8 = qword ptr -8
.text:000000000000089B
.text:000000000000089B ; __unwind {
.text:000000000000089B push rbp
.text:000000000000089C mov rbp, rsp
.text:000000000000089F sub rsp, 120h
.text:00000000000008A6 mov rax, fs:28h
.text:00000000000008AF mov [rbp+var_8], rax
.text:00000000000008B3 xor eax, eax
.text:00000000000008B5 mov eax, 0
.text:00000000000008BA call init
.text:00000000000008BF lea rdx, [rbp+buf]
.text:00000000000008C6 mov eax, 0
.text:00000000000008CB mov ecx, 20h ; ' '
.text:00000000000008D0 mov rdi, rdx
.text:00000000000008D3 rep stosq
.text:00000000000008D6 lea rdi, s ; "shellcode >"
.text:00000000000008DD call _puts
.text:00000000000008E2 lea rax, [rbp+buf]
.text:00000000000008E9 mov edx, 100h ; nbytes
.text:00000000000008EE mov rsi, rax ; buf
.text:00000000000008F1 mov edi, 0 ; fd
.text:00000000000008F6 call _read
.text:00000000000008FB mov [rbp+var_11C], 0
.text:0000000000000905 jmp short loc_94C
.text:0000000000000907 ; ---
.text:0000000000000907
.text:0000000000000907 loc_907: ; CODE XREF: main+BB↓j
00000905 0000000000000905: main+6A (Synchronized with Hex View-1)
Output window
Python>
Python>
Python

GDB (GNU Debugger)

- Basic commands
  - run - 執行程式
  - break *[addres] - 在該 address 設下斷點
  - continue - 繼續執行程式
  - disas [function/address] - 反組譯某個函式
  - delete [break point id]

GDB

- info registers (i r) - 查看暫存器狀態
- info breakpoint (i b) - 查看所有斷點
- info proc map (i porc m) - 查看 process memory mappings

GDB

- si - 執行一行指令，會跟進 function call
- ni - 執行一行指令，不跟進 function call
- backtrace (bt) - 顯示當前 stack frame 資訊

GDB

- x/nfu [address]
  - n, repeat count - 要列的次數
  - f, display format - 要顯示的格式
    - x - 十六進制, d - 十進制, s - 字串, i - 指令
  - u, unit size - b/h/w/g 分別為 1/2/4/8 bytes
- e.g.
  - x/2gx - 以十六進制顯示兩個 8byte memory

GDB

- set
  - set [reg]=[value] - 設置某個 register 的值
    - set $rax=0xfaceb00c
  - set [type][addres]=[value] - 將該記憶體地址填入該值
    - set {int}0x400000=0x123，將 0x400000 填入 0x00000123
    - set {long}0x400000=0x456，將 0x400000 填入 0x0000000000000456

GDB

- attach [pid] - attach 上正在運行的 process
- $ncat -vc [binary_path] -kl [port]
  - 將 binary 掛在該 port 上

GDB

- TUI
  - layout asm/src/reg
  - Ctrl + x + a 切换
  - fs cmd/src/asm

GDB-PEDA

- Python Exploit Development Assistance for GDB
- https://github.com/longld/peda.git
- https://github.com/scwuaptx/peda.git

Demo

Pwntools

- Pwntools is a CTF framework and exploit development library.
- https://github.com/Gallopsled/pwntools

Pwntools

- r = remote(host , port) - 建立連線至 host:port
  - r = remote(‘edu-ctf.csie.org’ , 10150)
- r.interactive() - 互動模式
- context.arch = ‘arch’
  - I386, amd64, arm, aarch64, thumb, mips, mips64, avr, s390 ...

Pwntools

- recv(n) - 接收 n bytes
- recvuntil(‘str’) - 接收至直到出現該 str pattern
  - s = r.recvuntil(’Osas’)
  - Server 送出 “Uvuvwevwevwe Onyetenyevwe Ugwemuhwem Osas”
  - s == “Uvuvwevwevwe Onyetenyevwe Ugwemuhwem Osas”
- recvline() - 接收至換行，recvuntil(’\n’)

Pwntools

- send(‘str’) - 向連線送出 str
- sendline(‘str’) - send(str + ‘\n’)
- sendafter(‘str1’ , ‘str2’) - 接收到 str1 後才送出 str2
  - r.recvuntil(‘str1’) 再 r.send(’str2’)
- sendlineafter(‘str1’ , ‘str2’)

Pwntools

- p32([4 bytes integer]) - Pack an integer (little-endian default)
  - p32(0x1234) == “\x34\x12\x00\x00”
- u32([4 bytes string]) - Unpack
  - u32(“abcd”) == 0x64636261
- p64, u64, p16, u16
- flat(iterable, ...) - 將參數的值全部進行 pack , 依據 context.arch 決定 32|64

Pwntools

- process(‘ELF_path’)
  - r = process(‘./shellc0de’)
- process([‘./binary’ , ‘argv1’ , ‘argv2’])

Demo

CTF

CTF

- Jeopardy
- A&D - Attack & Defense
- KoH - King of Hill
- https://ctftime.org/

Reverse engineering

- Reverse - 逆向工程 (reverse engineering)
- Pwn - Binary exploitation
- Web - Web security
- Crypto - Cryptography
- Misc
  - Forensic, PPC, stego

HITCON CTF

- https://ctftime.org/event/848

Balsn CTF

- https://balsn.tw/
- https://ctftime.org/event/811
- Balsn CTF 台灣之星(國內第一名)：$ 10,000 新台幣

Thanks