Introducing Reverse Engineering

Racterub @ YZU CS250

About me

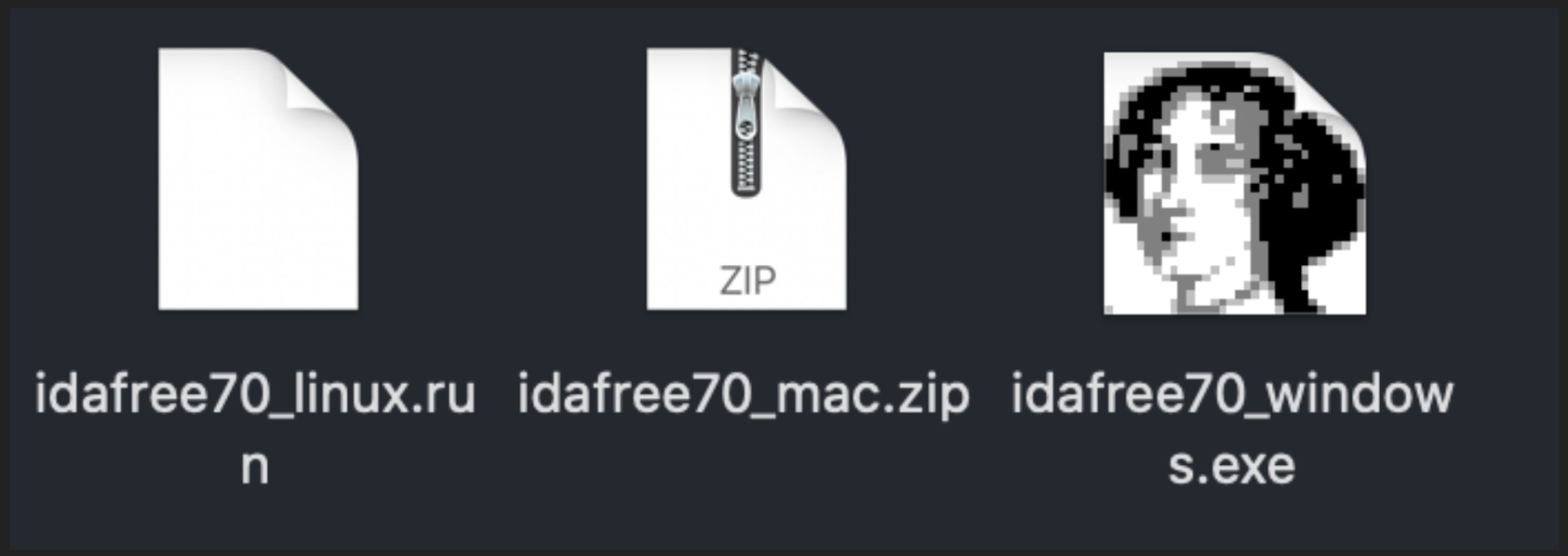
- 元智 電機通訊學院英語學士班 大一
- 目前主要玩的領域是資安
- 去年混分拿到了民生物聯網漏洞挖掘競賽第二期第三名

有電腦的可以載一下
今天會有 lab 讓你們試試

https://ppt.cc/f8C9Gx

工具包

- IDA Free 資料夾裡面選你的系統安裝就好了



Linux MacOS Windows

逆向？

你們熟悉的應該是

```
00000000004005a0 .text:
  4005a0: 31 ed                    xor     ebp, ebp
  4005a2: 49 89 d1                 mov     r9, rdx
  4005a5: 5e                       pop     rsi
  4005a6: 48 89 e2                 mov     rdx, rsp
  4005a9: 48 83 e4 f0              and     rsp, -16
  4005ad: 50                       push    rax
  4005ae: 54                       push    rsp
  4005af: 49 c7 c0 b0 07 40 00     mov     r8, 4196272
  4005b6: 48 c7 c1 40 07 40 00     mov     rcx, 4196160
  4005bd: 48 c7 c7 fb 06 40 00     mov     rdi, 4196091
  4005c4: ff 15 26 0a 20 00        call    qword ptr [rip + 2099750]
  4005ca: f4                       hlt
  4005cb: 0f 1f 44 00 00           nop     dword ptr [rax + rax]
  4005d0: f3 c3                    rep             ret
  4005d2: 66 2e 0f 1f 84 00 00 00 00 00 nop     word ptr cs:[rax + rax]
  4005dc: 0f 1f 40 00              nop     dword ptr [rax]
  4005e0: 55                       push    rbp
  4005e1: b8 48 10 60 00           mov     eax, 6295624
  4005e6: 48 3d 48 10 60 00        cmp     rax, 6295624
  4005ec: 48 89 e5                 mov     rbp, rsp
  4005ef: 74 17                    je      23 <.text+0x68>
  4005f1: b8 00 00 00 00           mov     eax, 0
  4005f6: 48 85 c0                 test    rax, rax
  4005f9: 74 0d                    je      13 <.text+0x68>
  4005fb: 5d                       pop     rbp
  4005fc: bf 48 10 60 00           mov     edi, 6295624
  400601: ff e0                    jmp     rax
  400603: 0f 1f 44 00 00           nop     dword ptr [rax + rax]
  400608: 5d                       pop     rbp
  400609: c3                       ret
  40060a: 66 0f 1f 44 00 00        nop     word ptr [rax + rax]
  400610: be 48 10 60 00           mov     esi, 6295624
  400615: 55                       push    rbp
  400616: 48 81 ee 48 10 60 00     sub     rsi, 6295624
  40061d: 48 89 e5                 mov     rbp, rsp
  400620: 48 c1 fe 03              sar     rsi, 3
  400624: 48 89 f0                 mov     rax, rsi
  400627: 48 c1 e8 3f              shr     rax, 63
  40062b: 48 01 c6                 add     rsi, rax
```

軟體逆向通常是夠過透過組語來了解一支程式的運行方式

今天有一題小 lab

這邊會先教一點 AMD64(x86-64) 的東西

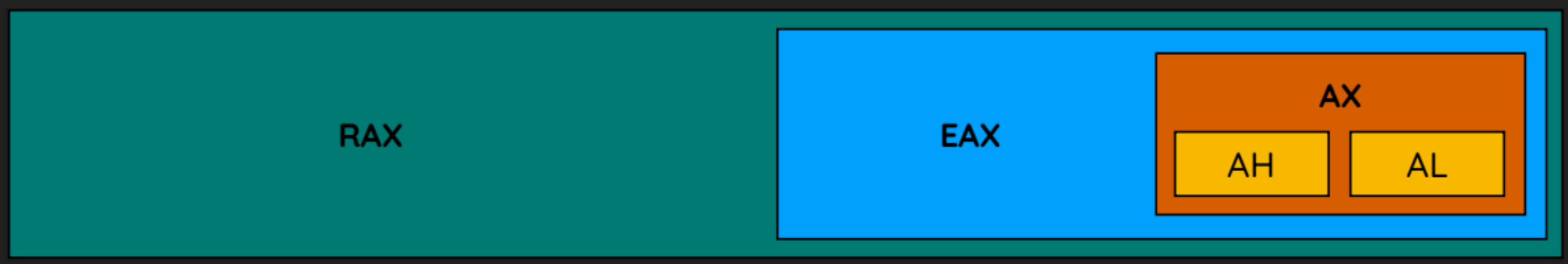
X64 calling convention

- 這裡講的是 AMD64 不是 Intel 64
- Function 的參數是以這樣的順序排的
  - RDI, RSI, RDX, RCX, R8, R9
- 所以 function(123, 456) 時
  - RDI => 123
  - RSI => 456

```
void function(int a, int b, int c, int d, int e, int f, int g) {
            //  rdi    rsi    rdx    rcx    r8     r9     ..stack
```

Registers

- R[A-D]X, RSI, RDI => 8 bytes
- E[A-D]X, ESI, EDI => 4 bytes
- [A-D]X, SI, DI => 2 bytes
- AX -> AH AL => 1 bytes
- RBP -> stack 底部
- RSP -> stack 頂部



Registers

RAX = 0x1234 5678 90ab cdef

- RAX -> 0x1234567890abcdef
  - EAX -> 0x90abcdef
    - AX -> 0xcdef
      - AH -> 0xcd
      - AL -> 0xef

一個古老的例子

- ++i v.s. i++
  - 在古老的時代，事實上 ++i 的效能會比較好
  - 這是由逆向程式後觀察組語發現的特性
- 不過現在 MSVC 跟 glibc 大概都優化到差不多了
  - (要調高優化)

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    int i=0;
    printf("Original i: %d\n", i);
    printf("i++: %d\n", i++);
    printf("++i: %d\n", ++i);
    return 0;
}
```

```
[vagrant@racterubctf:~/tmp] $ ./test
Original i: 0
i++: 0
++i: 2
```

*** gcc -no-pie test.c -o test -g * (預設優化為 -O1)**

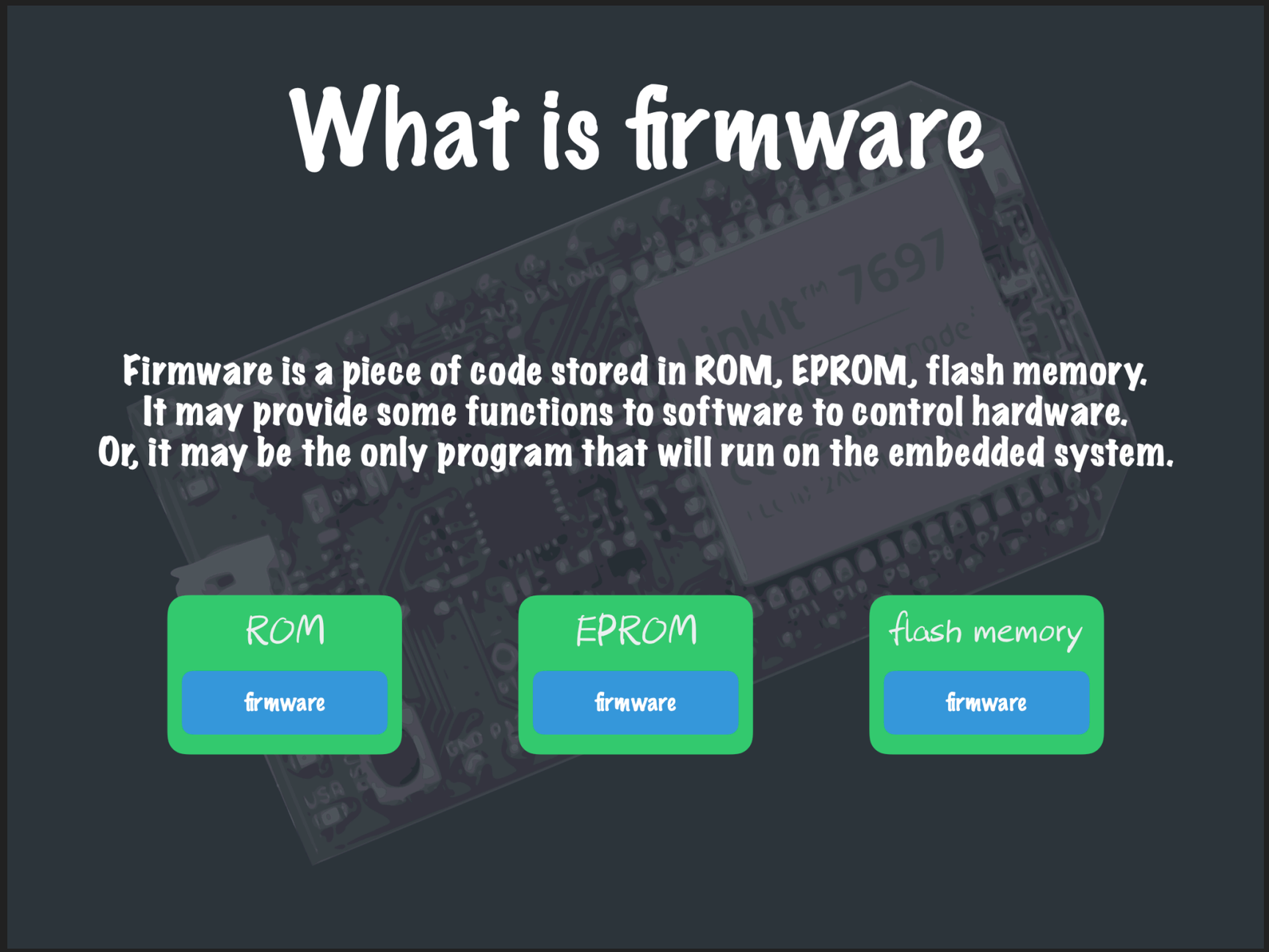
```
mov     eax,DWORD PTR [rbp-0x4]
lea     edx,[rax+0x1]
mov     DWORD PTR [rbp-0x4],edx
mov     esi,eax
lea     rdi,[rip+0xbf]        # 4005e4 <_IO_stdin_used+0x14>
mov     eax,0x0
call    4003f0 <printf@plt>
add     DWORD PTR [rbp-0x4],0x1
mov     eax,DWORD PTR [rbp-0x4]
mov     esi,eax
lea     rdi,[rip+0xae]        # 4005ed <_IO_stdin_used+0x1d>
mov     eax,0x0
call    4003f0 <printf@plt>
```

▲Compiled on Ubuntu 18.04 with GCC

逆向入門

逆向

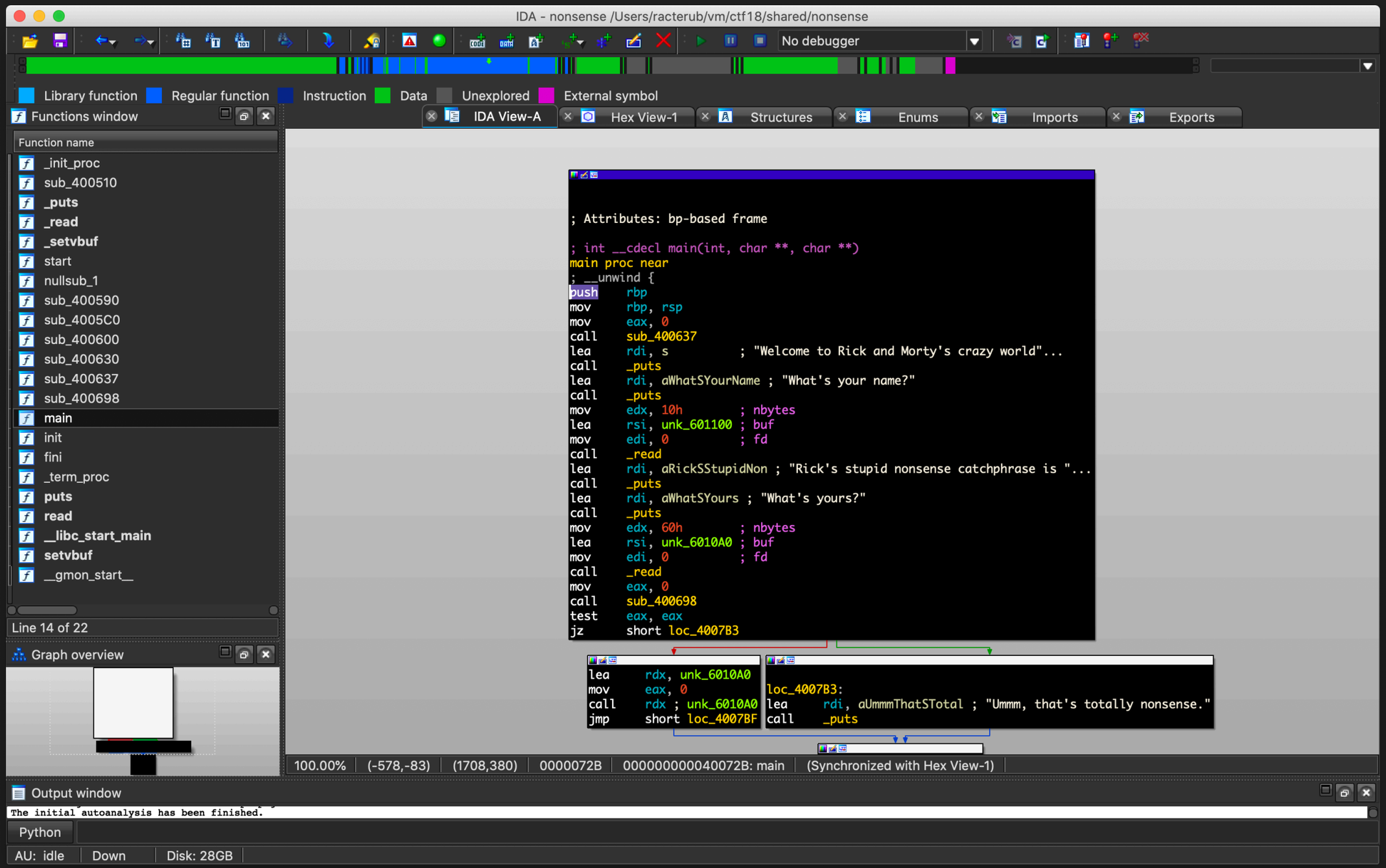
- 硬體



▲ Firmware analysis @ AIS3 2019

逆向

- 軟體



▲IDA Pro 分析 x64 binary

C/C++ 的逆向？

C/C++ Reversing

- 當你將一隻由 C/C++ 撰寫得程式編譯之後不論你是 object file 還是 binary ，都可以拿來逆向
- 通常不同平台、不同編譯器產出來的東西都不太一樣
- 但是透過 disassembler/debugger 還是可以正常了解程式的流程跟邏輯

C/C++ Reversing

- 動態分析
  - GDB

racterub@Racterubs-MacBook-Pro: ~/vm/ctf18 (ssh) ⌘1 | racterub@Racterubs-MacBook-Pro: ~/Desktop/fallenbeat/Fallen_Beat/songs/gekkou (zsh) ⌘2

```
Registers
RAX: 0x7fffffffdb25 --> 0x757a79006c69616d ('mail')
RBX: 0x0
RCX: 0x4
RDX: 0x5
RSI: 0x7fffffffdb25 --> 0x757a79006c69616d ('mail')
RDI: 0x7fffffffdb20 --> 0x69616d0074736574 ('test')
RBP: 0x7fffffffdb60 --> 0x4008c0 (<__libc_csu_init>:    push   r15)
RSP: 0x7fffffffdae0 --> 0x7fffffffdc48 --> 0x7fffffffdfc1 ("/home/vagrant/test")
RIP: 0x400828 (<main+158>:      cmp    QWORD PTR [rbp-0x50],0x0)
R8 : 0x5
R9 : 0x7ffff7fe64c0 (0x00007ffff7fe64c0)
R10: 0x3
R11: 0x7ffff7a82b50 (<strtok>:  lea    rdx,[rip+0x34ef11]        # 0x7ffff7dd1a68 <olds>)
R12: 0x400690 (<_start>:        xor    ebp,ebp)
R13: 0x7fffffffdc40 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
Code
   0x40081d <main+147>: mov    QWORD PTR [rbp-0x50],rax
   0x400821 <main+151>: cmp    QWORD PTR [rbp-0x58],0x0
   0x400826 <main+156>: je     0x40082f <main+165>
=> 0x400828 <main+158>: cmp    QWORD PTR [rbp-0x50],0x0
   0x40082d <main+163>: jne    0x40083b <main+177>
   0x40082f <main+165>: call   0x400777 <error>
   0x400834 <main+170>: mov    eax,0x1
   0x400839 <main+175>: jmp    0x4008a1 <main+279>
[rbp-0x50] : 0x7fffffffdb10 --> 0x7fffffffdb25 --> 0x757a79006c69616d ('mail')
Stack
0000| 0x7fffffffdae0 --> 0x7fffffffdc48 --> 0x7fffffffdfc1 ("/home/vagrant/test")
0008| 0x7fffffffdae8 --> 0x100000000
0016| 0x7fffffffdaf0 --> 0x0
0024| 0x7fffffffdaf8 --> 0x142e406547
0032| 0x7fffffffdb00 --> 0x9 ('\t')
0040| 0x7fffffffdb08 --> 0x7fffffffdb20 --> 0x69616d0074736574 ('test')
0048| 0x7fffffffdb10 --> 0x7fffffffdb25 --> 0x757a79006c69616d ('mail')
0056| 0x7fffffffdb18 --> 0xf0b5ff

Legend: code, data, rodata, heap, value
0x0000000000400828      25          if ((username == NULL) || (domain == NULL)) {
[gdb]>
```

C/C++

- 靜態分析
  - objdump 直接嗑組語
  - IDA Pro, Ghidra, Radare2
    - IDA Pro 是付費的，但是有出免費版 IDA Free

```
0000000000400777 <error>:
  400777:       55                      push   rbp
  400778:       48 89 e5                mov    rbp,rsp
  40077b:       48 8d 3d c2 01 00 00    lea    rdi,[rip+0x1c2]        # 400944 <_IO_stdin_used+0x4>
  400782:       e8 89 fe ff ff          call   400610 <puts@plt>
  400787:       90                      nop
  400788:       5d                      pop    rbp
  400789:       c3                      ret

000000000040078a <main>:
  40078a:       55                      push   rbp
  40078b:       48 89 e5                mov    rbp,rsp
  40078e:       48 83 c4 80             add    rsp,0xffffffffffffff80
  400792:       89 7d 8c                mov    DWORD PTR [rbp-0x74],edi
  400795:       48 89 75 80             mov    QWORD PTR [rbp-0x80],rsi
  400799:       64 48 8b 04 25 28 00    mov    rax,QWORD PTR fs:0x28
  4007a0:       00 00
  4007a2:       48 89 45 f8             mov    QWORD PTR [rbp-0x8],rax
  4007a6:       31 c0                   xor    eax,eax
  4007a8:       c6 45 9a 40             mov    BYTE PTR [rbp-0x66],0x40
  4007ac:       c6 45 9b 2e             mov    BYTE PTR [rbp-0x65],0x2e
  4007b0:       48 8d 3d a3 01 00 00    lea    rdi,[rip+0x1a3]        # 40095a <_IO_stdin_used+0x1a>
  4007b7:       b8 00 00 00 00          mov    eax,0x0
  4007bc:       e8 7f fe ff ff          call   400640 <printf@plt>
  4007c1:       48 8b 15 a8 08 20 00    mov    rdx,QWORD PTR [rip+0x2008a8]        # 601070 <stdin@@GLIBC_2.2.5>
  4007c8:       48 8d 45 c0             lea    rax,[rbp-0x40]
  4007cc:       be 32 00 00 00          mov    esi,0x32
  4007d1:       48 89 c7                mov    rdi,rax
  4007d4:       e8 87 fe ff ff          call   400660 <fgets@plt>
  4007d9:       48 8d 45 c0             lea    rax,[rbp-0x40]
  4007dd:       48 89 c7                mov    rdi,rax
  4007e0:       e8 3b fe ff ff          call   400620 <strlen@plt>
  4007e5:       83 e8 01                sub    eax,0x1
  4007e8:       89 45 9c                mov    DWORD PTR [rbp-0x64],eax
  4007eb:       8b 45 9c                mov    eax,DWORD PTR [rbp-0x64]
  4007ee:       48 98                   cdqe
  4007f0:       c6 44 05 c0 00          mov    BYTE PTR [rbp+rax*1-0x40],0x0
  4007f5:       48 8d 55 9a             lea    rdx,[rbp-0x66]
  4007f9:       48 8d 45 c0             lea    rax,[rbp-0x40]
  4007fd:       48 89 d6                mov    rsi,rdx
  400800:       48 89 c7                mov    rdi,rax
  400803:       e8 78 fe ff ff          call   400680 <strtok@plt>
  400808:       48 89 45 a8             mov    QWORD PTR [rbp-0x58],rax
  40080c:       48 8d 45 9a             lea    rax,[rbp-0x66]
  400810:       48 89 c6                mov    rsi,rax
```

IDA - test /Users/racterub/vm/ctf18/shared/test

No debugger

Library function | Regular function | Instruction | Data | Unexplored | External symbol

Functions window

Function name

- _init_proc
- sub_400600
- **_puts**
- **_strlen**
- ___stack_chk_fail
- **_printf**
- **_strtok_r**
- **_fgets**
- **_strcmp**
- **_strtok**
- _start
- _dl_relocate_static_pie
- deregister_tm_clones
- register_tm_clones
- __do_global_dtors_aux
- frame_dummy
- **error**
- **main**
- __libc_csu_init
- __libc_csu_fini
- _term_proc
- **puts**
- **strlen**

Line 18 of 31

Graph overview

IDA View-A | Hex View-1 | Structures | Enums | Imports | Exports

```
; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

argv= qword ptr -80h
argc= dword ptr -74h
domainDelim= byte ptr -66h
levelDelim= byte ptr -65h
inputLen= dword ptr -64h
save= qword ptr -60h
username= qword ptr -58h
domain= qword ptr -50h
level= qword ptr -48h
input= byte ptr -40h
var_8= qword ptr -8

; __unwind {
push    rbp
mov     rbp, rsp
add     rsp, 0FFFFFFFFFFFFFF80h
mov     [rbp+argc], edi
mov     [rbp+argv], rsi
mov     rax, fs:28h
mov     [rbp+var_8], rax
xor     eax, eax
mov     [rbp+domainDelim], 40h
mov     [rbp+levelDelim], 2Eh
lea     rdi, format     ; "Input email address: "
mov     eax, 0
call    _printf
mov     rdx, cs:stdin@@GLIBC_2_2_5 ; stream
lea     rax, [rbp+input]
mov     esi, 32h        ; n
mov     rdi, rax        ; s
call    _fgets
lea     rax, [rbp+input]
mov     rdi, rax        ; s
call    _strlen
sub     eax, 1
```

80.00% | (-941,192) | (2016,496) | 0000078A | 000000000040078A: main | (Synchronized with Hex View-1)

Output window

```
The initial autoanalysis has been finished.
```

Python

AU: idle | Down | Disk: 28GB

Lab

只需要對 demo_linux
逆向就好

https://ppt.cc/fttGHx

其他語言的逆向

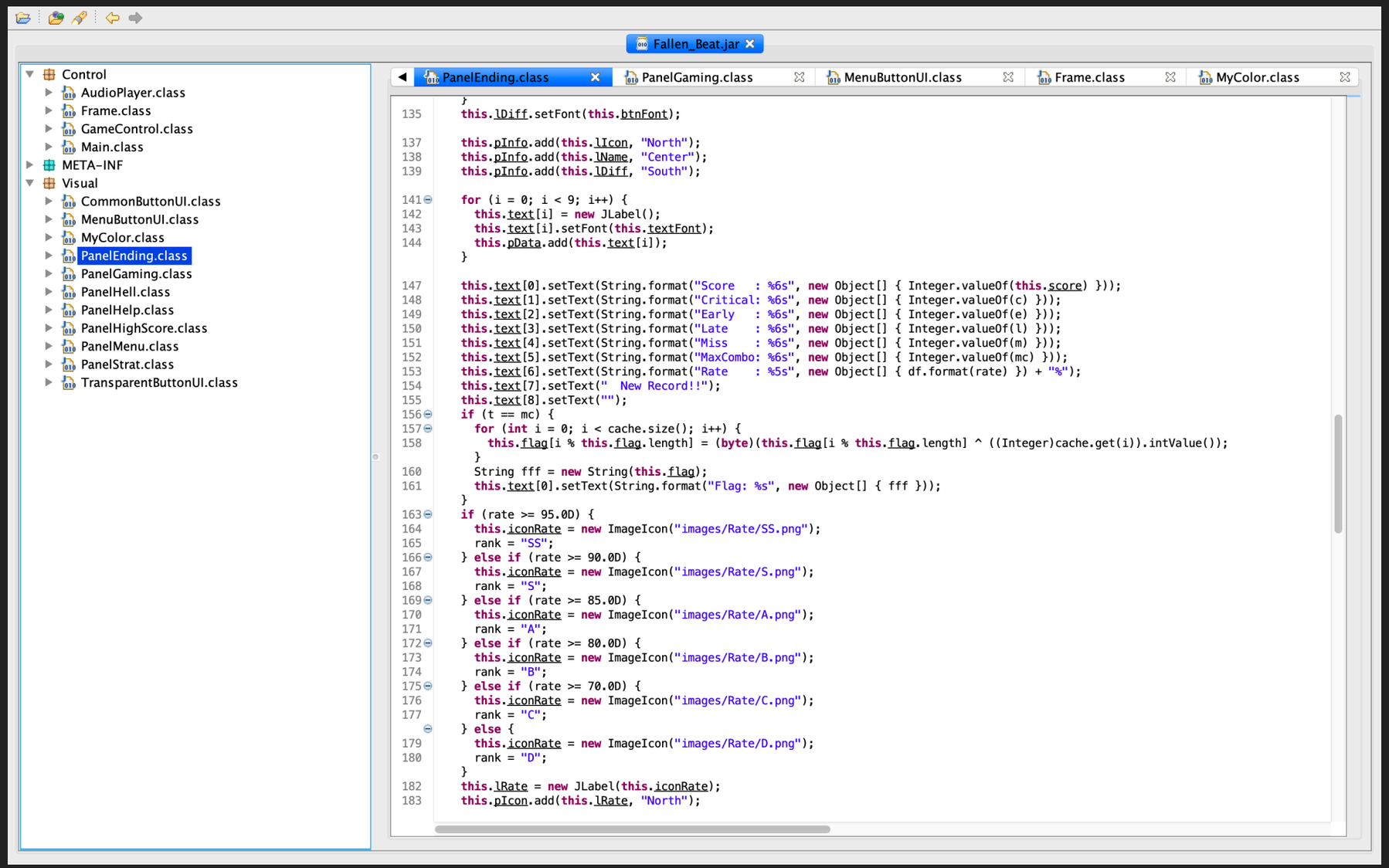
其他語言的逆向

- Python
  - Python 有一個功能是可以將 py 檔編譯檔
  - 使用 uncompyle6 就可以把編譯檔還原出程式碼
- 雖然有時候會爛掉

```
[vagrant@racterubctf:~/tmp/__pycache__] $ uncompyle6 test.cpython-36.pyc
# uncompyle6 version 3.7.1
# Python bytecode 3.6 (3379)
# Decompiled from: Python 3.6.9 (default, Apr 18 2020, 01:56:04)
# [GCC 8.4.0]
# Embedded file name: test.py
# Compiled at: 2020-06-12 12:14:40
# Size of source mod 2**32: 56 bytes
a = 1
b = 2
c = a + b
print(c)
# okay decompiling test.cpython-36.pyc
```

其他語言的逆向

- Java
  - 透過 JAD 或是 JD-GUI 反編譯



▲JD-GUI 反編譯的樣子

逆向可以幹嘛？

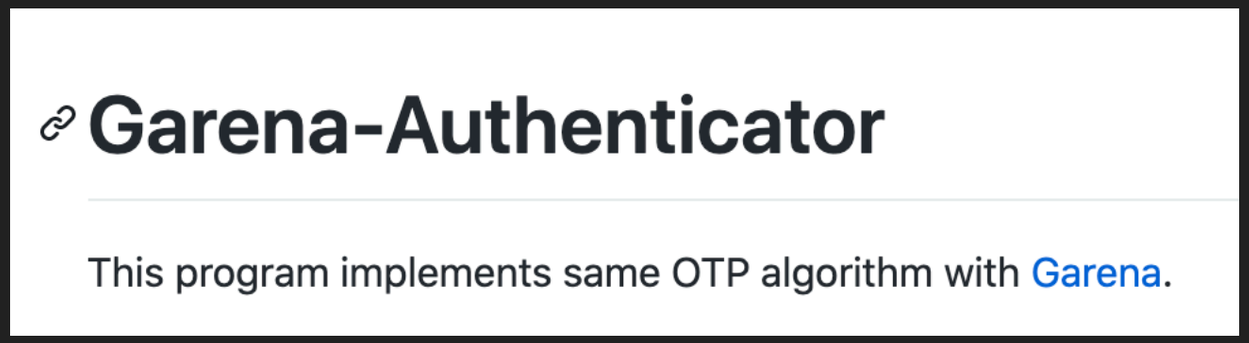
當你組員寫出了一個壞掉的程式作業，然後程式碼也被刪掉了

那就逆向硬幹

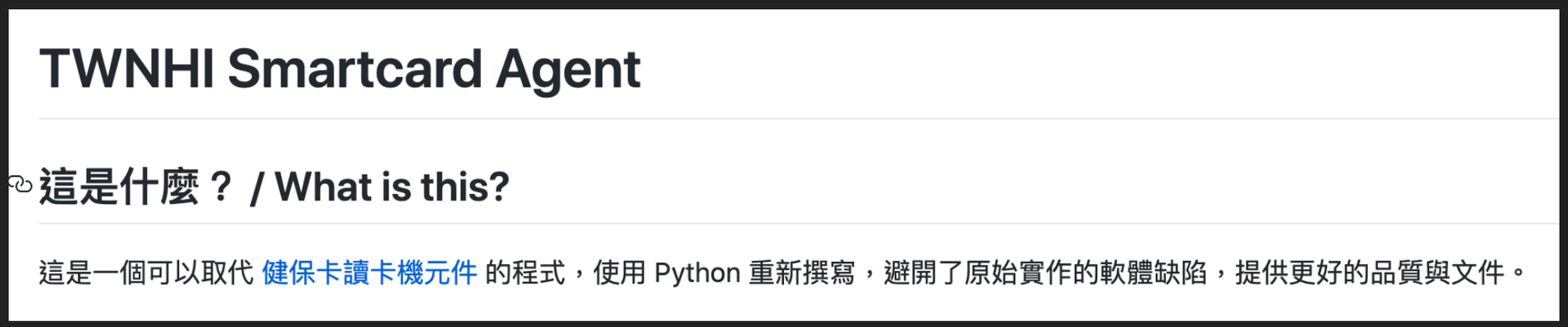
當你組員寫出了一個壞掉的程式作業，然後程式碼也被刪掉了

那就逆向硬幹

不過我猜這個情境發生的機率大概不高辣



▲https://github.com/Inndy/garena-authenticator



▲https://github.com/Inndy/twnhi-smartcard-agent

悠遊卡餘額明細查詢

作者 Zhi-Wei Cai 版本 v0.3 授權 MIT

> 由於悠遊卡公司不打算提供一般民眾方便查詢餘額的管道，目前已將查詢功能關閉（詳見官方聲明）。相關討論請瀏覽 PTT。

不受硬體限制

智慧卡公司對悠遊卡進行加密來禁止對卡片直接的餘額讀取功能，官方提供的查詢程式又僅支援少部分手機，且未對外開放 API 供大眾使用，十分不便。

▲https://github.com/x43x61x69/Easy-Card

```
# lol
key = 'EasyCardToKingay23456789'
iv = '01234567'
salt = 'L0CalKing'
const = 0x2160
```

▲https://github.com/x43x61x69/Easy-Card/blob/8f981ff718480d5a67b7afdec9324559685607cb/bin/easycard.py

yzuCourseBot 元智選課機器人

Update log

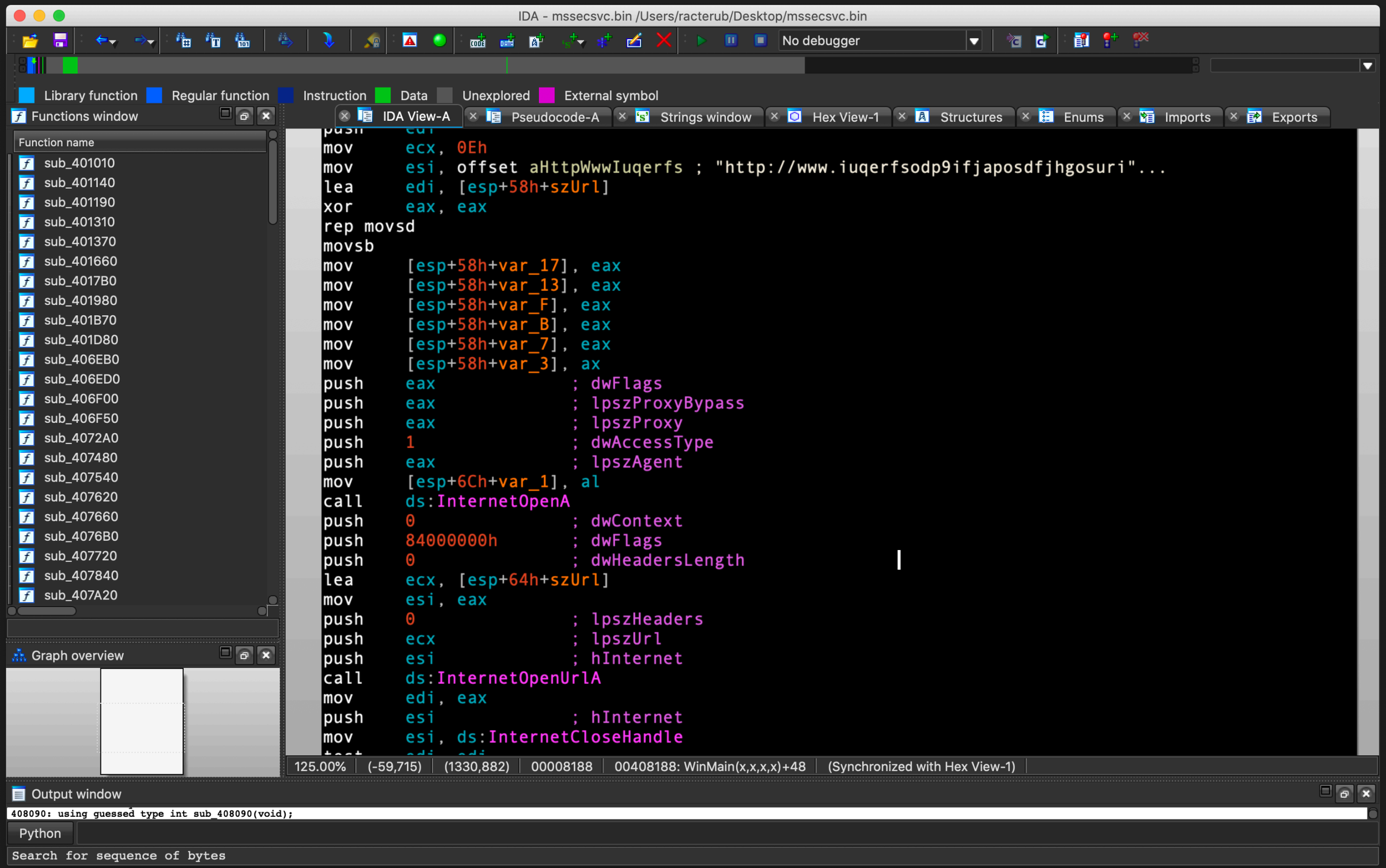
- 2020/02/26
  - [新增] 異常登入判斷，會休息10分鐘後再繼續
  - [新增] 系所編號判斷，填錯會報錯誤訊息
  - [新增] 新增delay功能，預設為1秒 (太快容易被server判定為異常request，請自行斟酌)
- 2019/12/11
  - [修改] 優化判斷是否登入成功的方式
  - [修改] 優化部分hard code的地方
  - [新增] 判斷使用者的課程ID是否存在


MapleStory

```
/*地圖內數量計數資訊 1.51.2*/
long MapMobCount[2]      = {0x0E920B4,0x24}; //紅點數量基址+4=怪物數量基址
/*地圖ID 1.51.2*/
long MapIdPointer[2] = {0x0E962B0,0x10d0};
/*人物本身資料 1.51.2*/
long SkillActionBase[2] = {0x0E8DA4C,0x4b8};
long HumActBase[2]      = {0x0E8DA4C,0x4b4};
long HumanPokeCount[2] =  {0x0E8DA4C,0x7338};
/*撿取物品 151.2*/
long PickItemAdr=0x04F923F; //6a 28 b8 ?? ?? ?? ?? e8 ?? ?? ?? ?? 8b f1 8b
/*取得人物完美XY Call 1.51.2*/
long GameGetHumXY_Call=0x04AC765; //33 c0 5e c3 55 8b ec 51 51 8d 45 ?? 50
/*血模警告 151.2*/
long HpMpNoticeBase[3] = {0x0E920AC→,0x50,0x54};
/*血模值 151.2*/
long HpMpValueBase[3] = {0x00E92358,0x2168,0x216c};
```


Wana Decrypt0r 2.0
Ooops, your files have been encrypted!
English
What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.
Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.
Payment will be raised on
5/16/2017 00:47:55
Time Left
02:23:57:37
Your files will be lost on

▲WannaCry



▲IDA Pro 分析 WannaCry

```
strcpy(&szUrl, "http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com");
v7 = 0;
v8 = 0;
v9 = 0;
v10 = 0;
v11 = 0;
v12 = 0;
v13 = 0;
v4 = InternetOpenA(0, 1u, 0, 0, 0);
InternetOpenUrlA(v4, &szUrl, 0, 0, 0x84000000, 0);
InternetCloseHandle(v4);
InternetCloseHandle(0);
sub_408090();
return 0;
```

void *(__fastcall *)(void *dst, const char *)

▲上一張組語轉成偽代碼的部分

Magisk

Cydia

▲越獄 / Root 所需要的漏洞也是由逆向而來的

逆向就是要了解程式碼跟組合語言之間關係

要知道你的 code 編譯會長怎樣，也要知道組合語言的邏輯合起來會等同於什麼 code

跟你寫程式一樣，你寫 code 就要知道你的 code 運作的方式

QA?

https://ppt.cc/f7FKFx