

HITCON GIRLS 的第一個軟體 破解課程

Charles

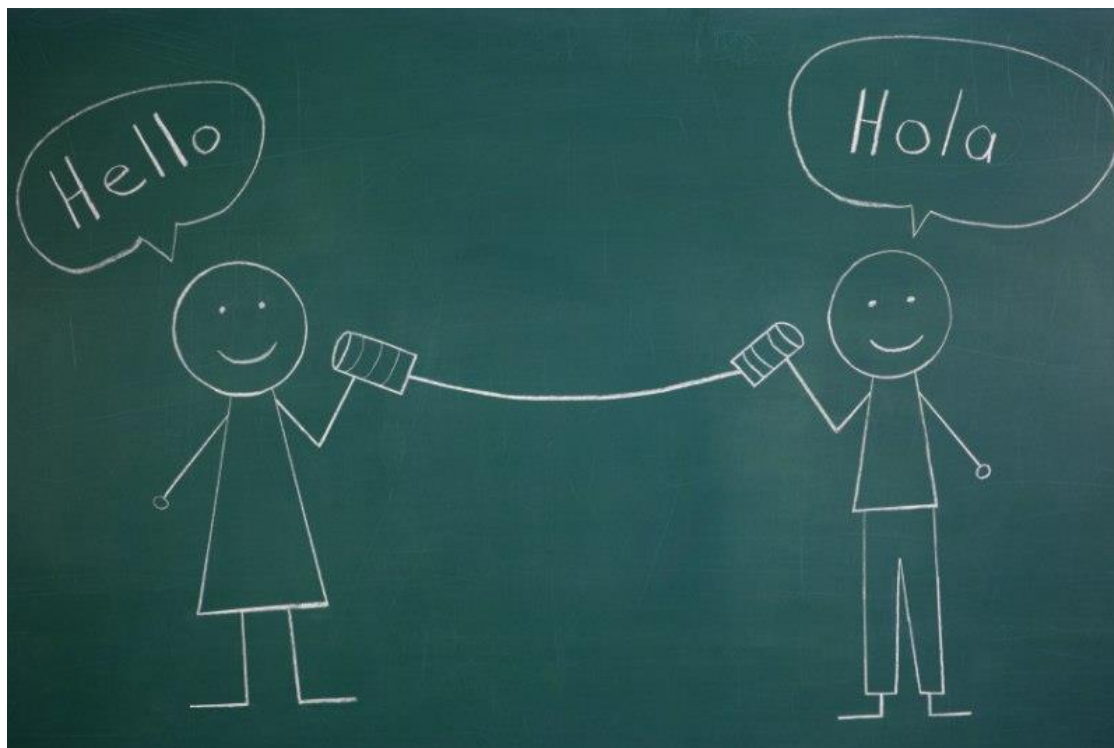


課程內容

- 逆向工程與軟體破解概念簡介
- Lab 0
- 軟體破解前你必須知道的二三事
- Lab 1
- 通往破解點的道路 – Strings 與 API
- Lab 2
- 軟體破解實戰Demo – 010 Editor

逆向工程與軟體破解

- 何謂逆向工程 – 從宅仔和小P的愛情故事說起



逆向工程與軟體破解

- 何謂逆向工程 – 宅仔深愛著小 p，卻發現自己聽不懂她說的話

!@#\$%^&*
()_+



```
int main(){  
    printf("I love you\n!!");  
}
```



逆向工程與軟體破解

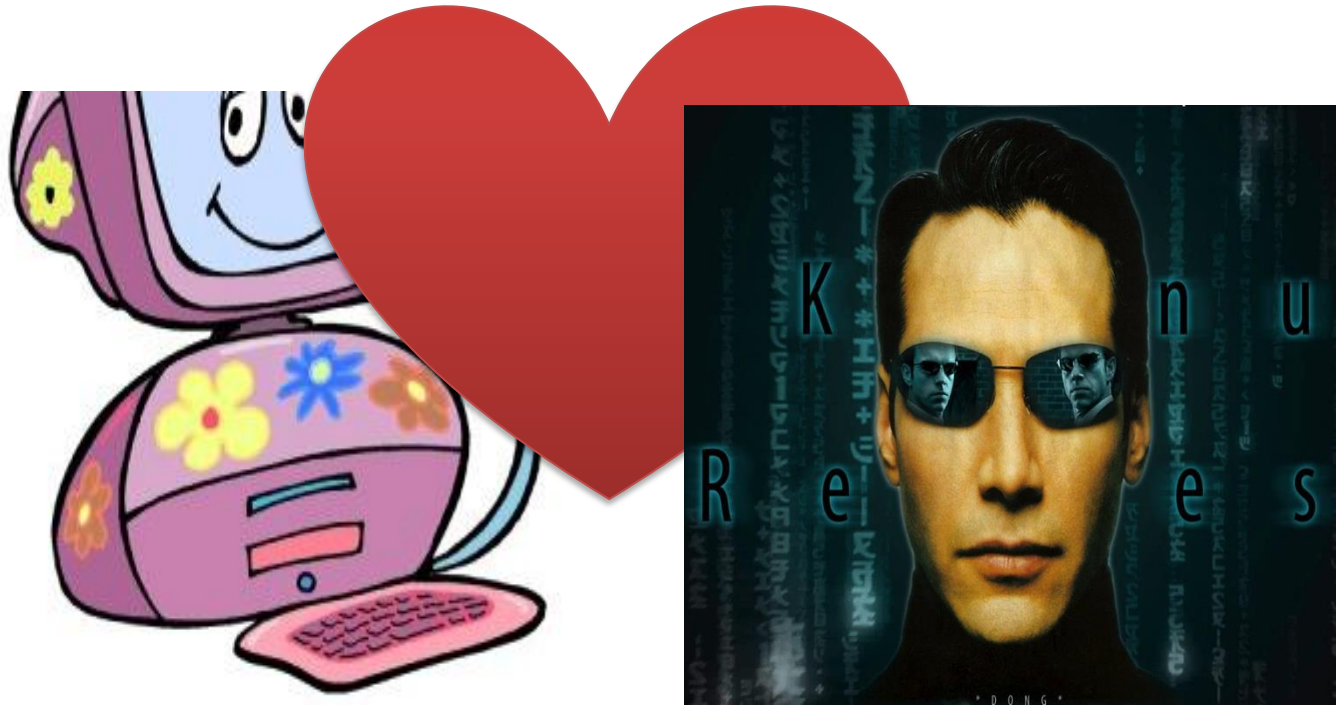
- 何謂逆向工程 – 痴心的宅仔想更了解小p的內心世界，於是努力學習小p的語言 – 機器語言。

```
817C24 08 1101>CMP DWORD PTR SS:[ESP+8],111
75 74      JNZ SHORT ncrackme.004010CE
8B4424 0C   MOV EAX,DWORD PTR SS:[ESP+C]
66:3D EA03  CMP AX,3EA
75 42      JNZ SHORT ncrackme.004010A6
E8 C70100  CALL ncrackme.00401000
85C0
```



逆向工程與軟體破解

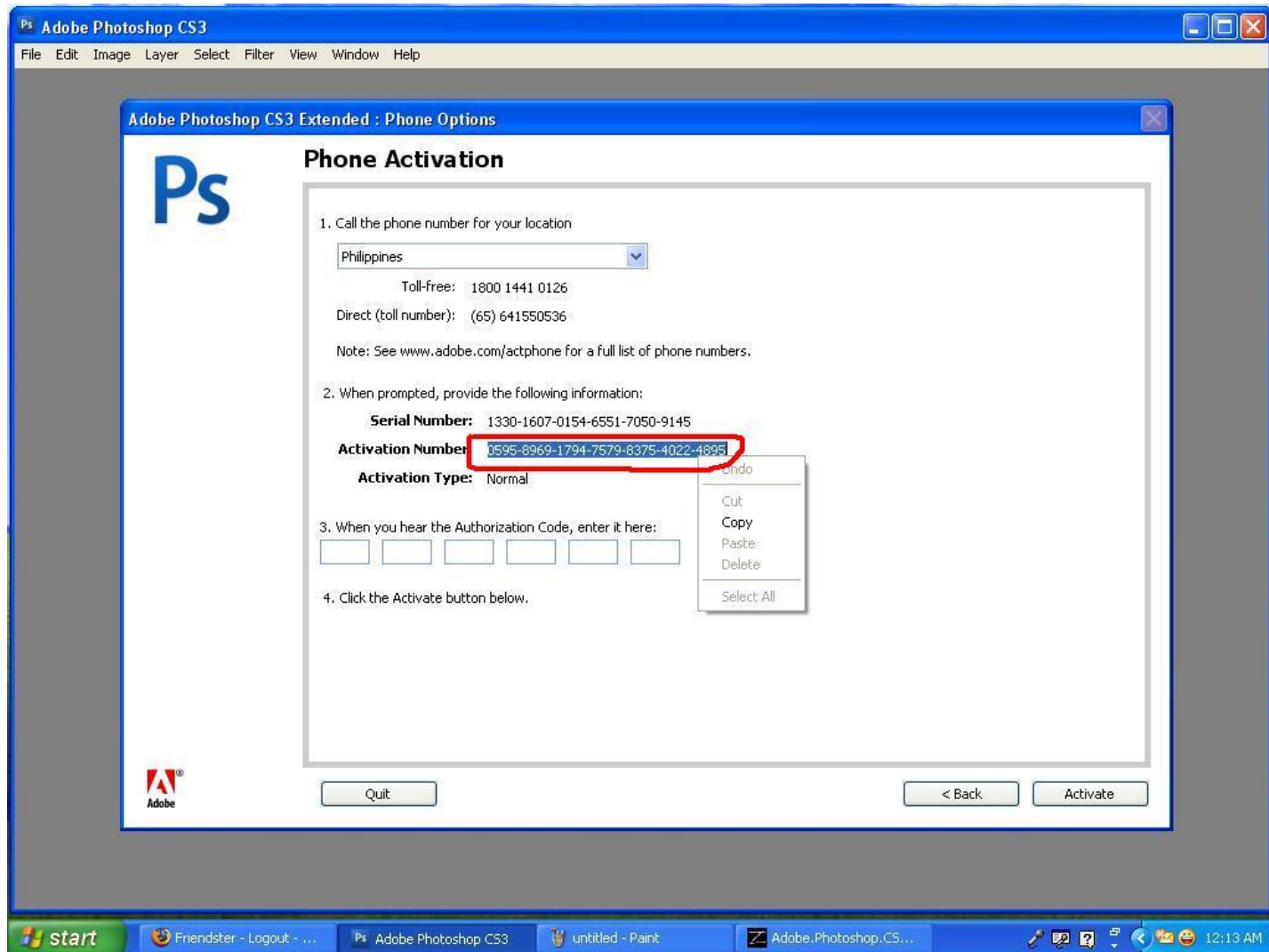
- 何謂逆向工程 –宅仔成功了，他學會了逆向工程，成為了駭客
- 成功擄獲小p的心。



逆向工程與軟體破解

- 逆向工程：工程師在寫程式時，
 - 通常是以高階語言(**high level language**)作撰寫，
 - 再藉由**Compiler**的幫助將其轉化成電腦所能接受的機器語言(**Machine language**)。
 - 逆向工程是將一個編譯過的二進位程式檔，試圖重製，或了解該程式運作方式的過程。
- 軟體破解：藉由逆向工程的手法，了解程式的架構或流程後，嘗試對其進行修改，以改變其原始運作方式
→(讓它乖乖聽話)。

軟體破解實例 – 序號破解



逆向工程與軟體破解

- 為什麼我們要學逆向工程：
 - 破解軟體保護 (跟朋友炫耀，~~用免錢的程式~~)
 - 讓沒有原始碼的軟體程式可以二次利用
 - 評估軟體程式品質和健全性 (黑箱測試)
 - 研究病毒和惡意程式 (like Charles 😊)
 - 幫現有程式增加功能 (寫遊戲外掛)

逆向工程與軟體破解

- 作逆向工程前，你必須：
 - 寫過簡單程式，了解何謂程式架構、邏輯判斷及流程跳轉
 - 碰過一點組合語言
 - 具備耐心與熱忱

逆向工程與軟體破解

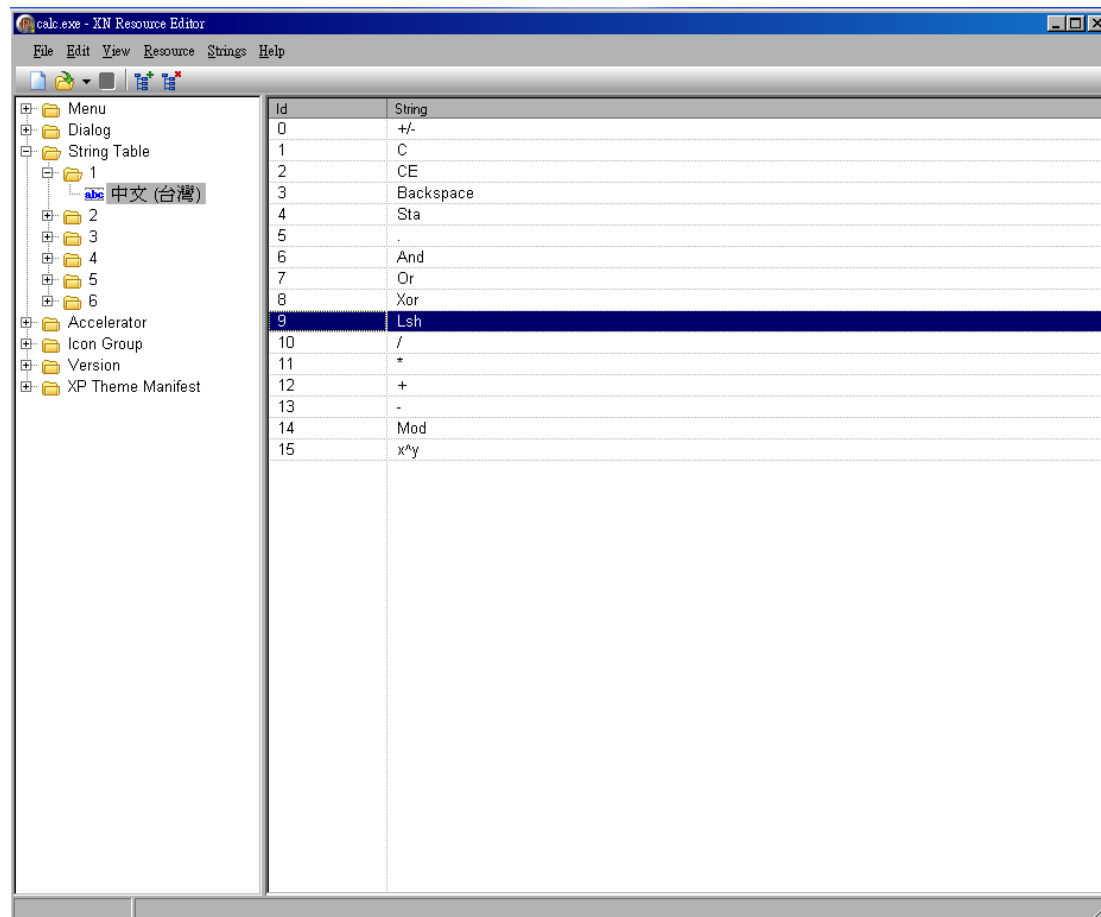
- 逆向工程常用到的工具：
 - 除錯器(Debuggers)
 - 16進位編輯器(Hex editors)
 - PE and resource viewers/editors
 - System Monitoring tools

課程內容

- 逆向工程與軟體破解概念簡介
- Lab 0
- 軟體破解前你必須知道的二三事
- Lab 1
- 通往破解點的道路 – Strings 與 API
- Lab 2
- 軟體破解實戰Demo – 010 Editor

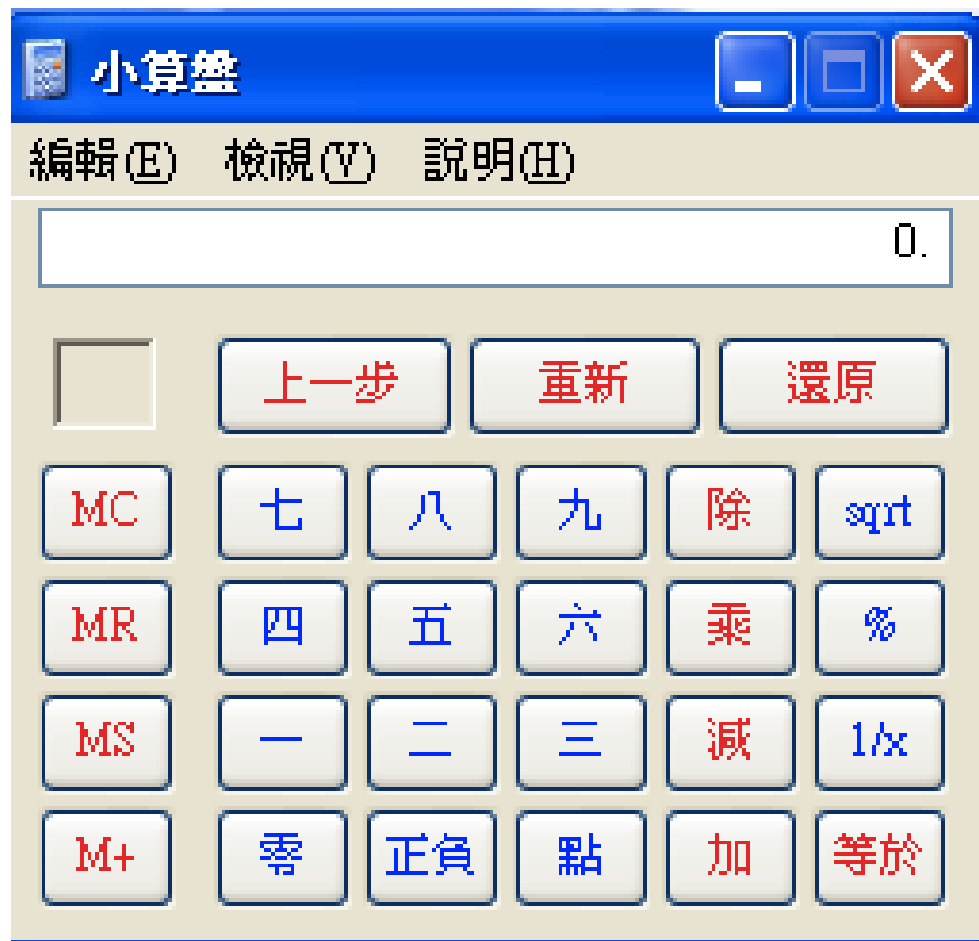
Lab 0

- 用資源編輯器來作我們的第一個程式修改：



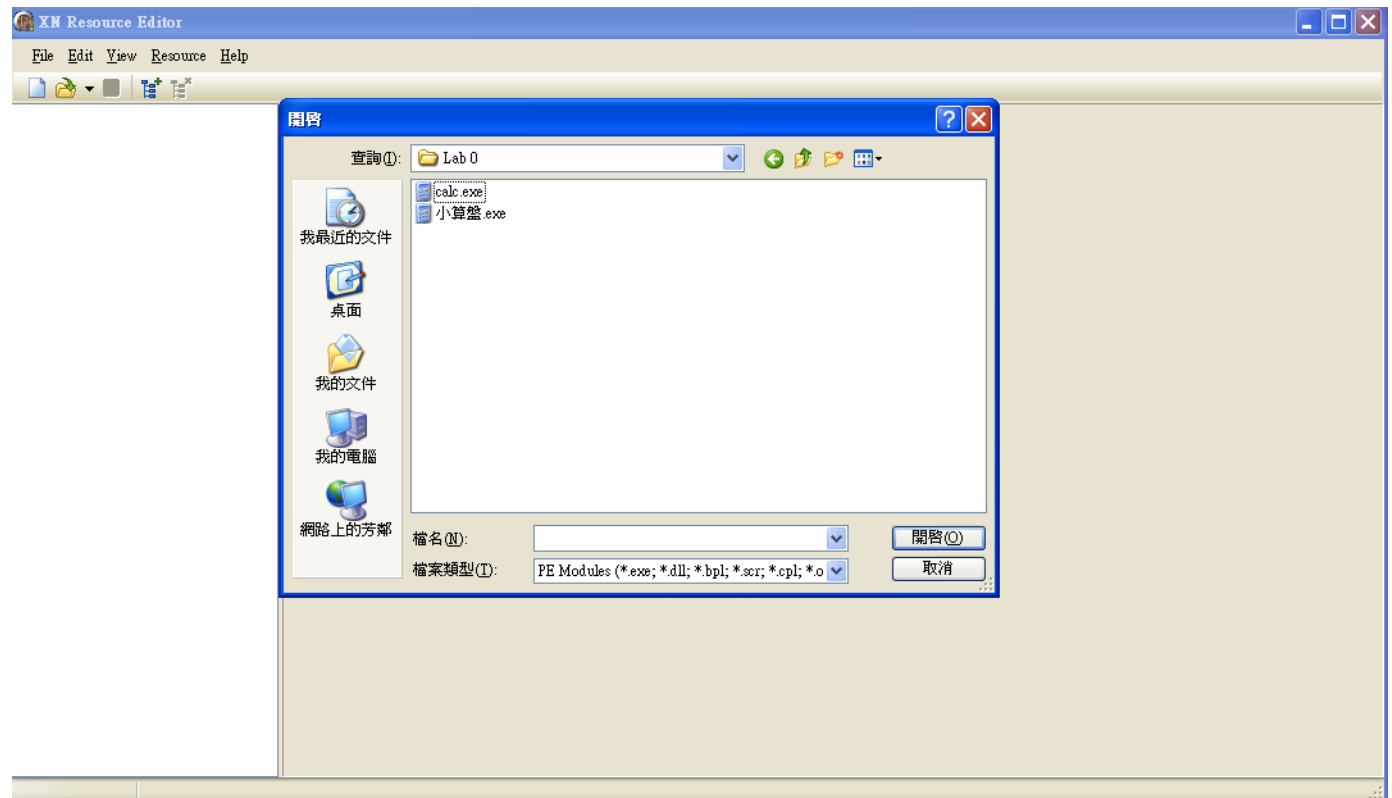
Lab 0

- 目的：修改出中文的小算盤：



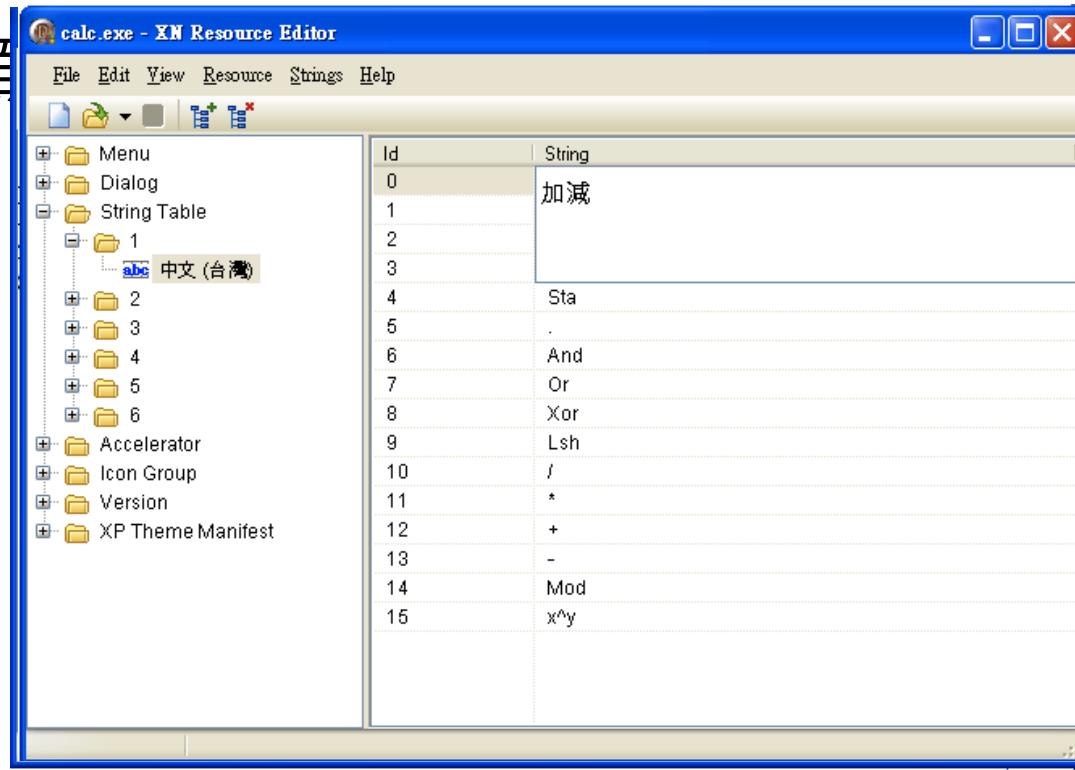
Lab 0

- 用XN Resource Editor打開”Lab 0”裡面附的calc.exe [File] → [Open]



Lab 0

- 選擇左方欄String Table, 依次展開，尋找想要修改的項目
- 找到想要修改的項目，修改



Lab 0

- 動手時間：10 mins

課程內容

- 逆向工程與軟體破解概念簡介
- Lab 0
- 軟體破解前你必須知道的二三事
- Lab 1
- 通往破解點的道路 – Strings 與 API
- Lab 2
- 軟體破解實戰Demo – 010 Editor

程式破解前你必須知道的二三事

- 程式流程跳轉: if-else

```
int main(){  
  
    unsigned int age;  
    printf("Please enter your age: ");  
    scanf("%d", &age);  
    printf("\n");
```

Action1

程式操作

```
    if(age < 18){
```

```
        printf
```

```
    }
```

```
    else{
```

```
        printf
```

```
    }
```

```
}
```

C:\Documents and Settings\Administrator\桌面

\Hitcon_girls\Lab>age.exe

Please enter your age: 18

Let's have a crazy party !! :D

C:\Documents and Settings\Administrator\桌面

\Hitcon_girls\Lab>age.exe

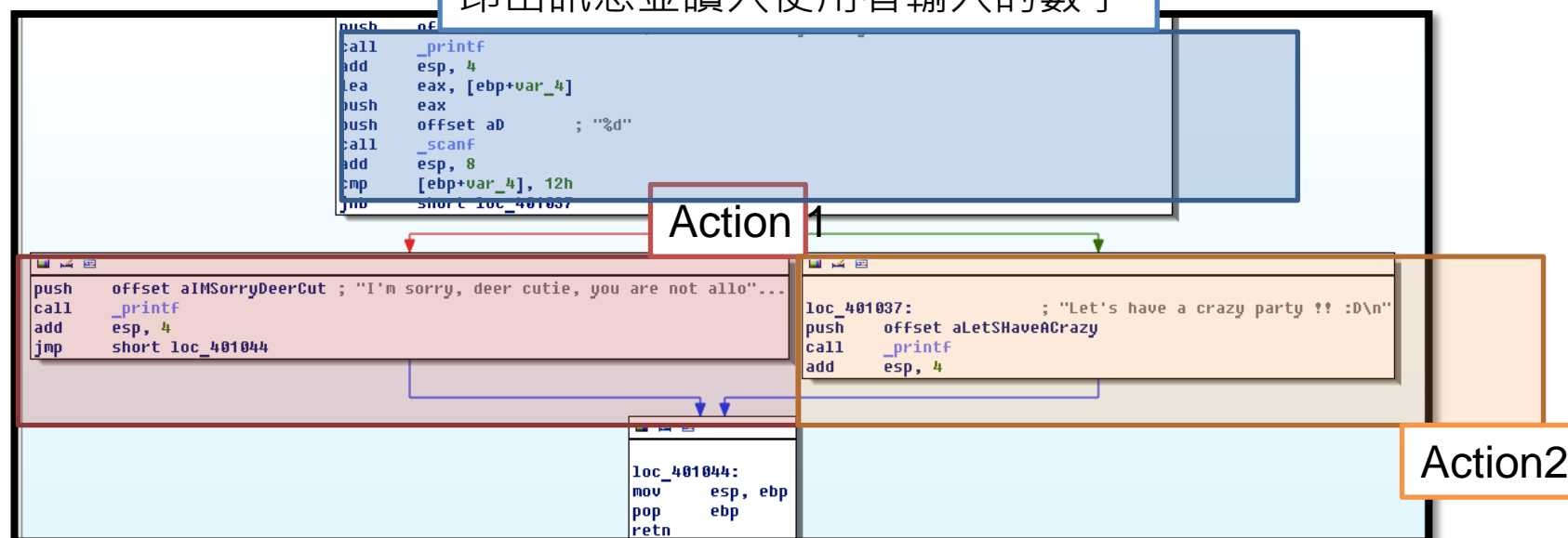
Please enter your age: 12

I'm sorry, deer cutie, you are not allowed to drink beer :(

程式破解前你必須知道的二三事

- 程式流程跳轉(示音圖)

印出訊息並讀入使用者輸入的數字



```
printf("Please enter your age: ");
scanf("%d", &age);
```

```
if(age < 18){
    printf("I'm sorry, deer cutie, you are not allowed to drink beer :(\n");
}
else{
    printf("Let's have a crazy party !! :D\n");
}
```

程式破解前你必須知道的二三事

- 除錯器：藉由除錯器(debugger)的幫助，可以一步步觀察目標程式在各時間點的狀態
 - 記憶體(memory)狀態、
 - 程式碼跳轉(code branch)。
- 有效幫助我們了解目標程式的運作邏輯。

程式破解前你必須知道的一二事

• 除

```
00401090 55 PUSH EBP
00401091 8BEC MOV EBP,ESP
00401093 6A FF PUSH -1
00401095 68 A8604000 PUSH age.004060A8
0040109A 68 712F4000 PUSH age.00402F94
0040109F 6A 01 00000000 MOV EAX,DWORD PTR FS:[0]
004010A5 50 PUSH EAX
004010A6 6A 10 92F40000 MOV DWORD PTR FS:[0],ESP
004010AD 8FEC 10 SUB ESP,10
004010B0 50 PUSH EAX
004010B1 56 PUSH ESI
004010B2 57 PUSH EDI
004010B3 8965 E8 MOV DWORD PTR SS:[EBP-18],ESP
004010B6 FF15 04604000 CALL DWORD PTR DS:[40604000]
004010BC 33D2 XOR EDX,EDX
004010BE 8AD4 MOV DL,AH
004010C0 8915 58794000 MOV DWORD PTR DS:[407958],EDX
004010C6 8BC8 MOV EAX,EAX
004010C8 81E1 FF000000 AND EAX,0
004010CE 890D 54794000 MOV DWORD PTR DS:[407954],ECX
004010D4 C1E1 08 SHL ECX,8
004010D7 03CA ADD ECX,EDX
004010D9 890D 50794000 MOV DWORD PTR DS:[407950],ECX
004010DF C1E8 10 SHR EAX,10
004010E2 A3 4C794000 MOV DWORD PTR DS:[40794C],EAX
004010E7 6A 00 PUSH 0
004010E9 E8 711D0000 CALL age.00402E5F
004010EE 59 POP ECX
004010EF 85C0 TEST EAX,EAX
004010F1 75 08 JNZ SHORT age.004010FB
004010F3 6A 1C PUSH 1C
004010F5 E8 9A000000 CALL age.00401194
004010FA 59 POP ECX
004010FB 8365 FC 00 AND DWORD PTR SS:[EBP-4],0
004010FF E8 B01B0000 CALL age.00402CB4
00401104 FF15 00604000 CALL DWORD PTR DS:[40604000]
0040110A A3 648E4000 MOV DWORD PTR DS:[408E64],EAX
EBP=0012FFF0
```

Code

```
Registers <FPU>
EAX 00000000
ECX 0012FFB0
EDX 7C92E514 ntdll.KiFastSystemCallRet
EBX 7FFD4000
ESP 0012FFC4
EBP 0012FFF0
ESI 00790074
EDI 0069006E
EIP 00401090 age.<ModuleEntryPoint>

C 0 ES 0023 32bit 0<FFFFFFFF>
P 1 CS 001B 32bit 0<FFFFFFFF>
A 0 SS 0023 32bit 0<FFFFFFFF>
Z 1 DS 0023 32bit 0<FFFFFFFF>
S 0 FS 003B 32bit 7FFDF000<FFF>
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS <00000000>
EFL 00000246 <NO,NB,E,BE,NS,PE,GE,LE>

ST0 empty
ST1 empty
ST2 empty
ST3 empty
ST4 empty
ST5 empty
ST6 empty
ST7 empty

FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 <GT>
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1
```

Register

Address	Hex dump
00407000	00 00 00 00 00 00 00 00 00 00 00 00 69 1C 40 00
00407010	C4 3F 40 00 00 00 00 00 00 00 00 00 0E 1D 40 00
00407020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00407030	50 6C 65 61 73 65 20 65 6E 74 65 72 20 79 6F 75
00407040	72 20 61 67 65 3A 20 00 25 64 00 00 49 27 6D 20
00407050	73 6F 72 72 20 2C 20 64 65 65 72 20 63 75 74 69
00407060	65 20 6E 6F 74 20 61 20 64 72 69 6E 6B 20 60
00407070	6C 6E 00 00 00 00 00 00 00 00 00 00 4C 65 74 27
00407080	62 6E 00 00 00 00 00 00 00 00 00 00 63 72 61 7A
00407090	73 27 44 0A 00 34 26 40 00 00 00 00 00 00 00 00
004070A0	61 70 01 00 00 20 09 2D 0D 5D 00 00 00 5D 00 00
004070B0	00 7E 40 00 00 00 00 60 7E 40 00 01 01 00 00 00
004070C0	00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00
004070D0	00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00
004070E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004070F0	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00407100	00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00
00407110	02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00407120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Memory

```
0012FFC4 7C817077 00 RETURN to kernel32.7C817077
0012FFC8 0069006E n.i.
0012FFCC 00790074 t.y.
0012FFD0 7FFD4000 .0?
0012FFD4 80545CFD T
0012FFD8 0012FFC8 ??.
0012FFDC 81C96DA8
0012FFE0 FFFFFFFF chain
0012FFE4 7C839AD8
0012FFE8 7C817080 17080
0012FFEC 00000000
0012FFF0 00000000
0012FFF4 00000000
0012FFF8 00401090 ?0. age.<ModuleEntryPoint>
0012FFFC 00000000
```

Stack

程式破解前你必須知道的二三事

- 除錯器 – Debugger 指令：
- 中斷點：cpu的特殊instruction，會使程式流程在該處中斷
 - bp xxxx: 在位址xxxx設置軟體斷點
 - F9: 執行程式，直到遇到斷點為止
 - F8: 執行單一指令
- Debugger操作

程式破解前你必須知道的二三事

- 我們的程式經編譯過後長什麼樣子呢？

記憶體位址

組合語言

記憶體位址	組合語言	機器碼
00401000	PUSH EBP	55
00401001	MOV EBP,ESP	8BEC
00401003	PUSH ECX	51
00401004	PUSH age.00407030	68 30704
00401009	CALL age.0040105F	E8 51000
0040100E	ADD ESP,4	83C4 04
00401011	LEA EAX,DWORD PTR SS:[EBP-4]	8D45 FC
00401014	PUSH EAX	50
00401015	PUSH age.00407048	68 48704
0040101A	CALL age.00401048	E8 29000
0040101F	ADD ESP,8	83C4 08
00401022	CMP DWORD PTR SS:[EBP-4],12	837D FC
00401026	INB SHORT age.00401032	73 0F
00401028	PUSH age.0040704C	68 4C704
0040102D	CALL age.0040105F	E8 2D000
00401032	ADD ESP,4	83C4 04
00401035	IMP SHORT age.00401041	EB 0D
00401037	PUSH age.0040708C	68 8C704
0040103C	CALL age.0040105F	E8 1E000
00401041	ADD ESP,4	83C4 04
00401044	MOV ESP,EBP	8BE5
00401046	POP EBP	5D
00401047	RETN	C3

ASCII "Please enter your age: "

ASCII "%d"

Action 1

ASCII "I'm sorry, deer cutie, you are not

Action 2

ASCII "Let's have a crazy party !! :D"

機器碼

程式破解前你必須知道的二三事

- 機器語言如何實作「流程分支」的概念：

The image displays a snippet of assembly code with annotations explaining a conditional jump. The code is as follows:

```
00401000 55      PUSH EBP
00401001 8BEC    MOV EBP,ESP
00401003 51      PUSH ECX
00401004 68 30704 PUSH age.00407030
00401005 8B45    MOV EAX,[EBP+4]
00401006 8B45    MOV EAX,[EBP+4]
00401007 8B45    MOV EAX,[EBP+4]
00401008 8B45    MOV EAX,[EBP+4]
00401009 8B45    MOV EAX,[EBP+4]
0040100A 8B45    MOV EAX,[EBP+4]
0040100B 8B45    MOV EAX,[EBP+4]
0040100C 8B45    MOV EAX,[EBP+4]
0040100D 8B45    MOV EAX,[EBP+4]
0040100E 8B45    MOV EAX,[EBP+4]
0040100F 83C4 08  ADD ESP,8
00401010 837D FC  CMP DWORD PTR SS:[EBP-4],12
00401011 73 0F    JNB SHORT age.00401037
00401012 68 4C704 PUSH age.0040704C
00401013 E8 2D000 CALL age.0040105F
00401014 83C4 04  ADD ESP,4
00401015 EB 0D    JMP SHORT age.00401044
00401016 68 8C704 PUSH age.0040708C
00401017 E8 1E000 CALL age.0040105F
00401018 83C4 04  ADD ESP,4
00401019 8BE5    MOV ESP,EBP
0040101A 5D      POP EBP
0040101B C3      RETN
```

Annotations and flow:

- A green box highlights the instruction `CMP DWORD PTR SS:[EBP-4],12` with the text "比較變數跟0x12(18)的大小值".
- A green arrow points from this instruction to a label `cmp: compare`.
- A red arrow points from the `JNB` instruction to a box labeled "JNB: Jmp if Not Below".
- Two red arrows indicate the flow of execution: one from the `JNB` instruction to "Action 1" (at address 0x401015) and another from "Action 1" to "Action 2" (at address 0x401016).

如果條件(不小於)成立，程式跳到0x401037(Action 2)去執行

課程內容

- 逆向工程與軟體破解概念簡介
- Lab 0
- 軟體破解前你必須知道的二三事
- Lab 1
- 通往破解點的道路 – Strings 與 API
- Lab 2
- 軟體破解實戰Demo – 010 Editor

Lab 1

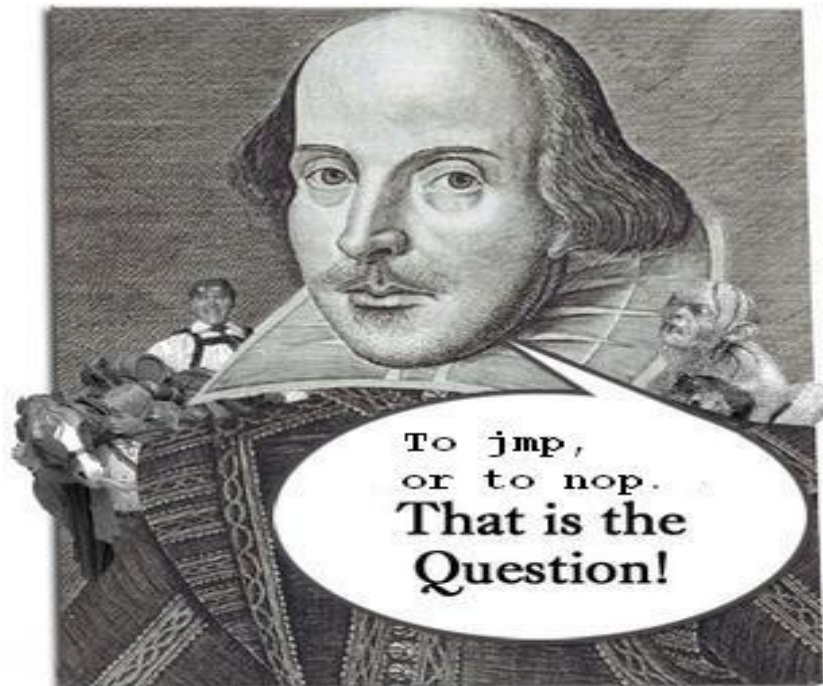
- 目標：修改我們的程式age.exe，讓它無論輸入年紀多少都顯示：“Let's have a crazy party !! :D”

```
C:\Documents and Settings\Administrator\桌面\Hitcon_girls\Lab 1>noage.exe  
Please enter your age: 10  
Let's have a crazy party !! :D
```

```
C:\Documents and Settings\Administrator\桌面\Hitcon_girls\Lab 1>noage.exe  
Please enter your age: 99  
Let's have a crazy party !! :D
```

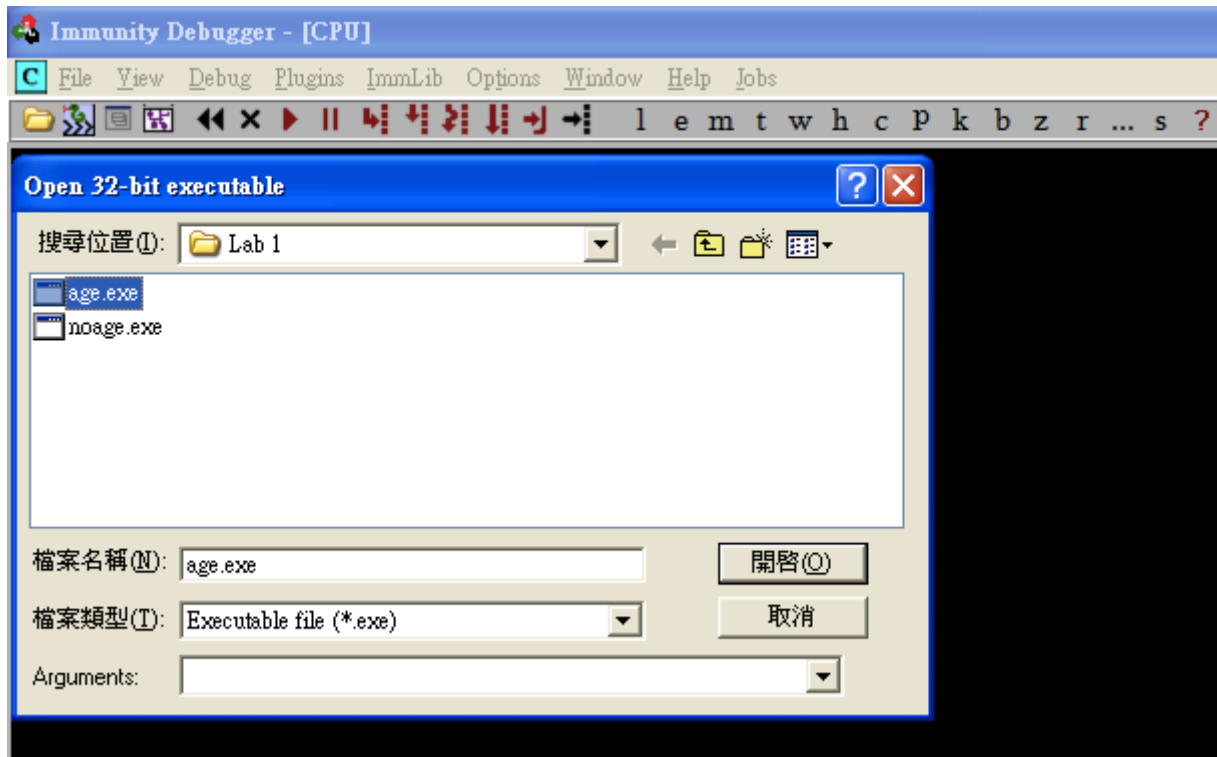
Lab 1

- 兩個必須熟知的組合語言指令：
 - **nop**: No Operation, 不作任何事
 - **jmp** xxxx: 將程式流程強制跳至位址xxxx去執行



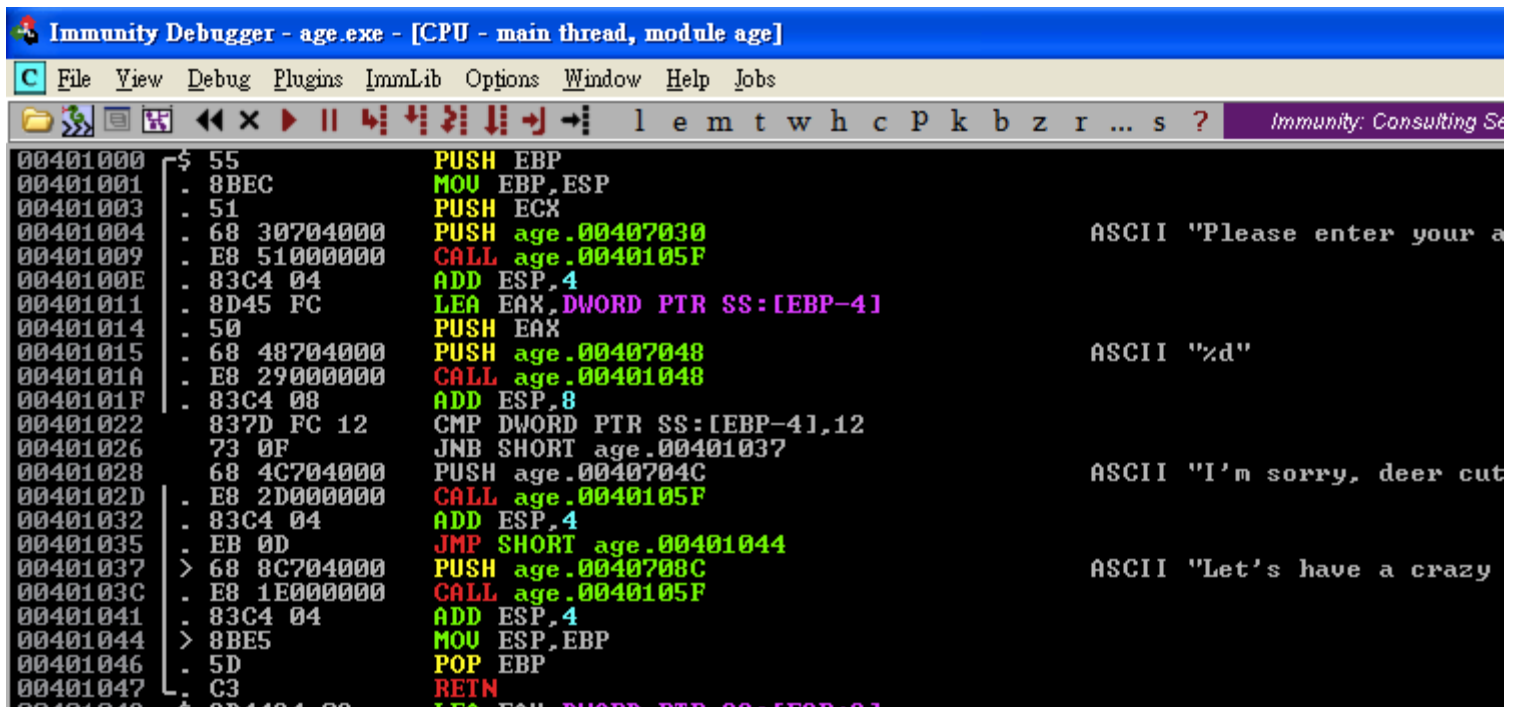
Lab 1

- 用debugger開啟我們的 age.exe:
- [File] → [Open], 選擇age.exe



Lab 1

- 在code視窗(左上角)，作上下捲動，直到找到**0x401000** – 我們的main function 所在為止(或按Ctrl+G，輸入0x401000直接到該位址)



The screenshot shows the Immunity Debugger interface with the CPU window displaying assembly code for the main thread of age.exe. The code starts at address 00401000 and includes instructions for pushing EBP, ECX, and EAX, calling age.0040105F, and adding ESP. Comments on the right side of the code indicate ASCII strings like "Please enter your a", "%d", "I'm sorry, deer cut", and "Let's have a crazy".


```
00401000 55      PUSH EBP
00401001 8BEC    MOV EBP,ESP
00401003 51      PUSH ECX
00401004 68 30704000  PUSH age.00407030      ASCII "Please enter your a
00401009 E8 51000000  CALL age.0040105F
0040100E 83C4 04    ADD ESP,4
00401011 8D45 FC    LEA EAX,DWORD PTR SS:[EBP-4]
00401014 50      PUSH EAX
00401015 68 48704000  PUSH age.00407048      ASCII "%d"
0040101A E8 29000000  CALL age.00401048
0040101F 83C4 08    ADD ESP,8
00401022 837D FC 12  CMP DWORD PTR SS:[EBP-4],12
00401026 73 0F     JNB SHORT age.00401037
00401028 68 4C704000  PUSH age.0040704C      ASCII "I'm sorry, deer cut
0040102D E8 2D000000  CALL age.0040105F
00401032 83C4 04    ADD ESP,4
00401035 EB 0D     JMP SHORT age.00401044
00401037 68 8C704000  PUSH age.0040708C      ASCII "Let's have a crazy
0040103C E8 1E000000  CALL age.0040105F
00401041 83C4 04    ADD ESP,4
00401044 8BE5     MOV ESP,EBP
00401046 5D      POP EBP
00401047 C3      RETN
```

Lab 1

- To jmp or to nop, that's the question:
 - 在紅色箭頭所在處，程式有一個分歧點：
 - 繼續往下一個指令執行(不jmp)
 - 亦或跳到0x401037執行

```
00401000 $ 55      PUSH EBP
00401001 . 8BEC    MOV EBP,ESP
00401003 . 51      PUSH ECX
00401004 . 68 30704 PUSH age.00407030      ASCII "Please enter your age: "
00401009 . E8 51000 CALL age.0040105F
0040100E . 83C4 04  ADD ESP,4
00401011 . 8D45 FC  LEA EAX,DWORD PTR SS:[EBP-4]
00401014 . 50      PUSH EAX
00401015 . 68 48704 PUSH age.00407048
0040101A . E8 29000 CALL age.00401048
0040101F . 83C4 08  ADD ESP,8
00401022 . 837D FC  CMP DWORD PTR SS:[EBP-4],12
00401026 . 73 0F    JNB SHORT age.00401037
00401028 . 68 40704 PUSH age.0040704C      ASCII "I'm sorry, deer cutie, you are not
0040102D . E8 20000 CALL age.0040105F
00401032 . 83C4 04  ADD ESP,4
00401035 . EB 01    JMP SHORT age.00401044
00401037 > 68 8C704 PUSH age.0040708C      ASCII "Let's have a crazy party ?? :D"
0040103C . E8 1E000 CALL age.0040105F
00401041 . 83C4 04  ADD ESP,4
00401044 > 8BE5    MOV ESP,EBP
00401046 . 5D      POP EBP
00401047 . C3      RETN
```

比較式，0x12=18



Lab 1

- 假設你希望把分歧點消除，讓程式永遠只作某件事：
 - 把jnb指令改成nop→永遠作action 1
 - 把jnb指令改成jmp→永遠作action 2

```
00401000 $ 55 PUSH EBP
00401001 - 8BEC MOV EBP,ESP
00401003 - 51 PUSH ECK
00401004 - 68 30704 PUSH age.00407030 ASCII "Please enter your age: "
00401009 - E8 51000 CALL age.0040105F
0040100E - 83C4 04 ADD ESP,4
00401011 - 8D45 FC LEA EAX,DWORD PTR SS:[EBP-4]
00401014 - 50 PUSH EAX
00401015 - 68 48704 PUSH age.00407048 ASCII "%d"
0040101A - E8 29000 CALL age.00401048
0040101F - 83C4 08 ADD ESP,8
00401022 - 837D FC CMP DWORD PTR SS:[EBP-4],12
00401026 - 73 0F JMP SHORT age.00401037
00401028 - 68 4C704 PUSH age.0040704C ASCII "I'm sorry, deer cutie, you are not
0040102D - E8 2D000 CALL age.0040105F Action 1
00401032 - 83C4 04 ADD ESP,4
00401035 - EB 0D JMP SHORT age.00401044
00401037 > 68 8C704 PUSH age.0040708C ASCII "Let's have a crazy party !! :D"
0040103C - E8 1E000 CALL age.0040105F
00401041 - 83C4 04 ADD ESP,4
00401044 > 8BE5 MOV ESP,EBP
00401046 - 5D POP EBP
00401047 - C3 RETN
```

Lab 1

- 動手時間 – 10 mins – 怎麼修改指令？
- 我們的目標是希望程式永遠執行action 2，因此要將jnb所在的指令改為**jmp**
 - 將滑鼠移至該指令上方選取它
 - 按右鍵
 - 選“**Assemble**”
 - 將原本的
 - “JNB SHORT 00401037”，改成→
 - “**Jmp** SHORT 00401037”
 - 按**F9**執行程式，看結果

課程內容

- 逆向工程與軟體破解概念簡介
- Lab 0
- 軟體破解前你必須知道的二三事
- Lab 1
- 通往破解點的道路 – Strings 與 API
- Lab 2
- 軟體破解實戰Demo – 010 Editor

通往破解點的道路

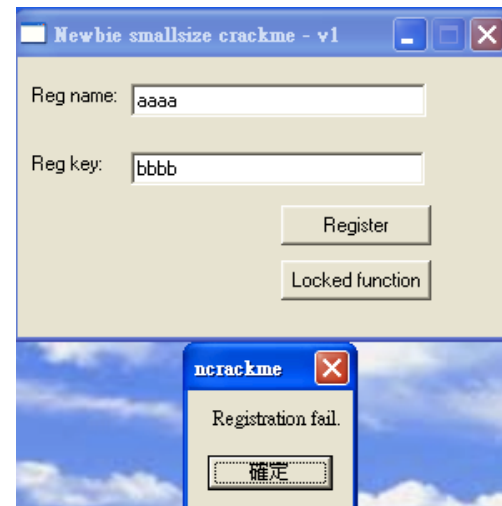


通往破解點的道路 – 字串引用

- 範例程式：簡單的序號驗證程式。
- 程式操作示範
- 假想程式流程：
 - 讀進reg name 和 reg key
 - 將reg name, reg key當作參數，傳入自己的演算法，驗證是否正確
 - 驗證失敗，跳出視窗顯示錯誤

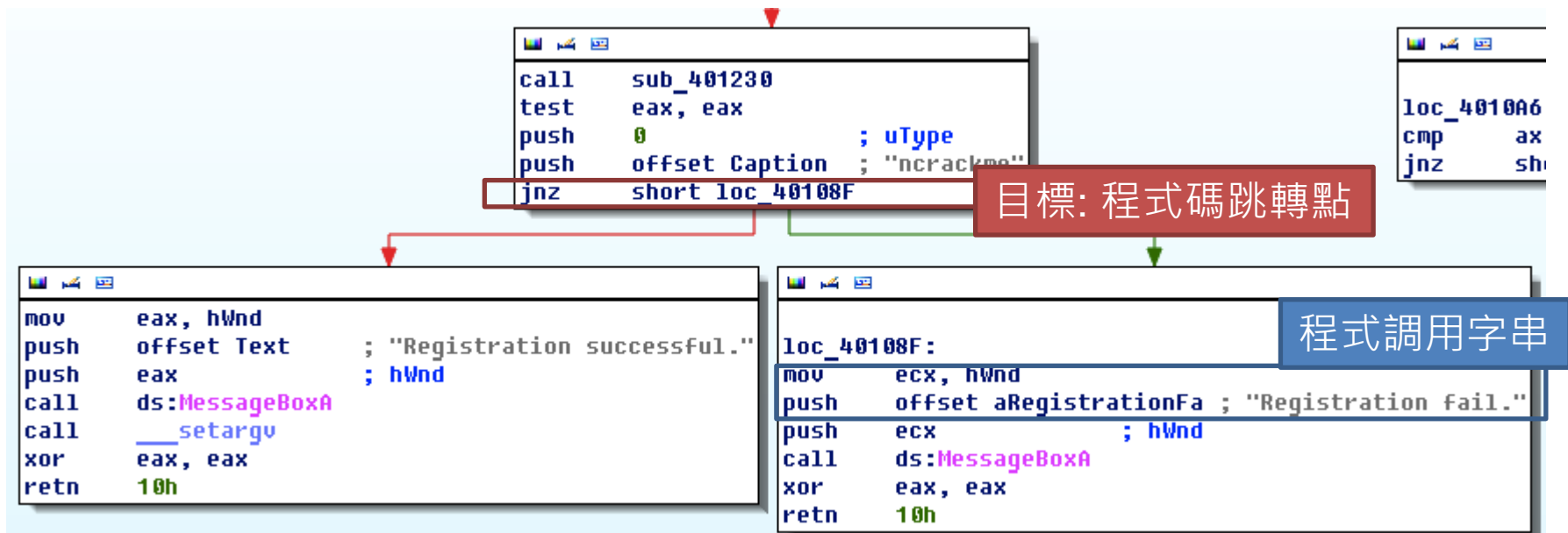


猜測：程式引用到“Registration fail”處的程式碼，可能很接近作驗證的程式碼(我們要修改的點)



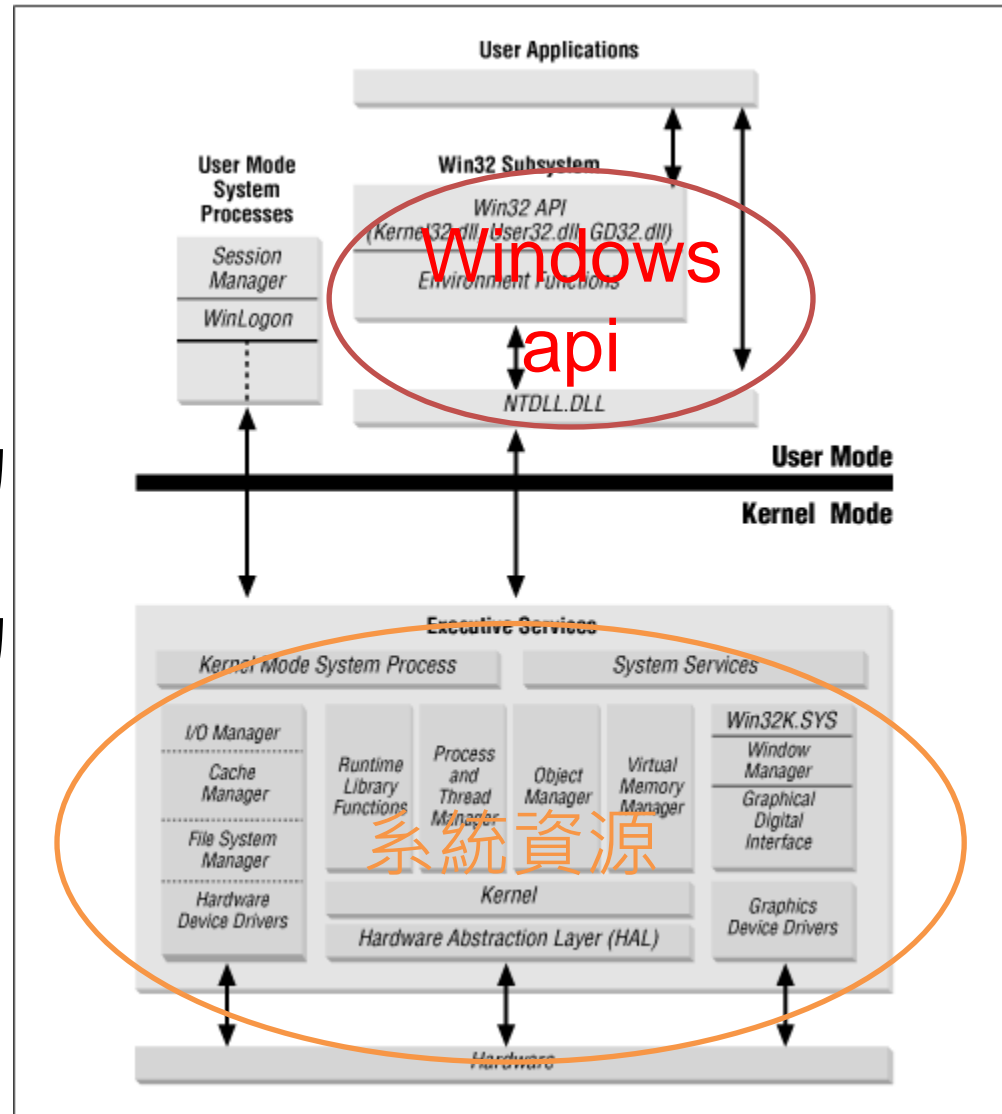
通往破解點的道路 – 字串引用

- 程式實際流程圖：



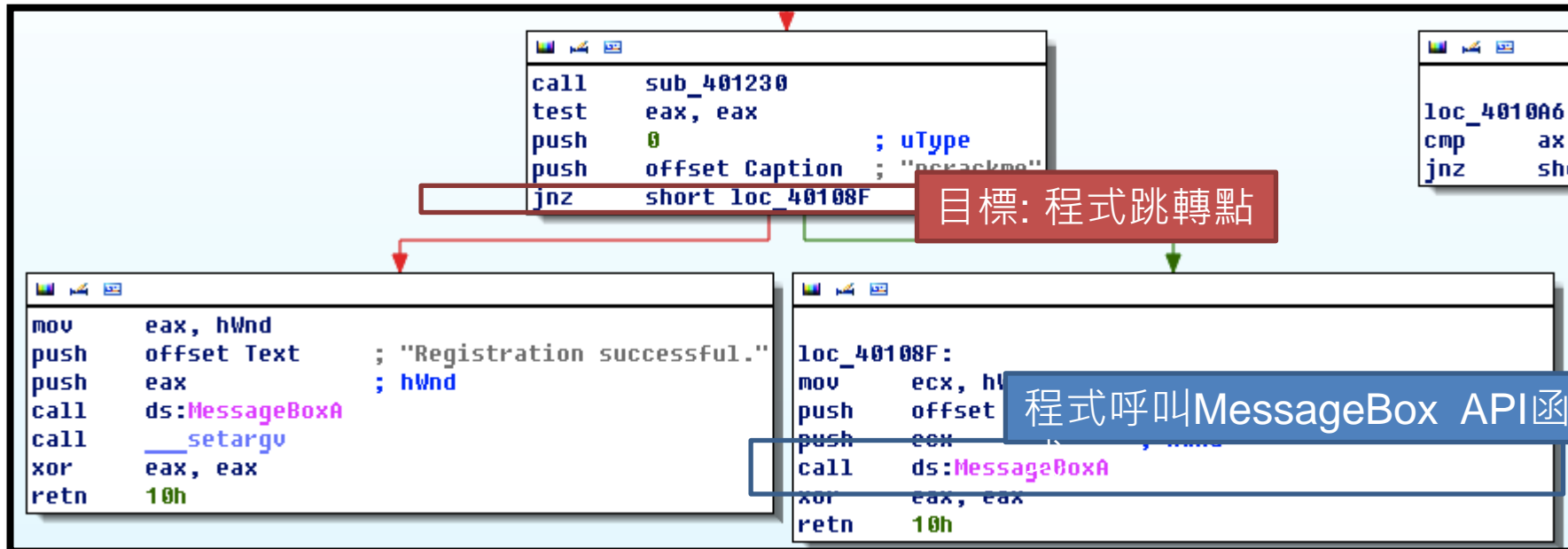
通往破解點的道路 - Windows API

- 視窗作業系統應用程式介面 (Windows API)
 - Windows提供的函式
 - 用來讓使用者作底層的系統資源存取
 - 分門別類存放於不同的函式庫(DLL)中



通往破解點的道路 - Windows API

- 掌握適當Windows API, 可以幫助我們快速找到關鍵程式碼(破解點)。



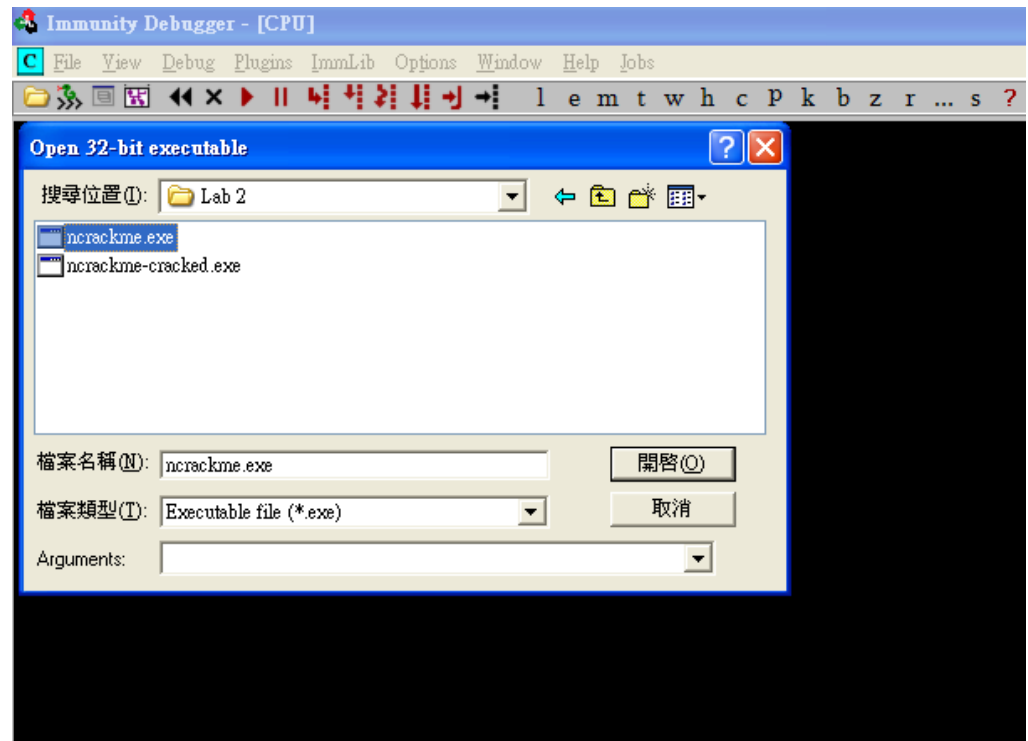
課程內容

- 逆向工程與軟體破解概念簡介
- Lab 0
- 軟體破解前你必須知道的二三事
- Lab 1
- 通往破解點的道路 – Strings 與 API
- Lab 2
- 軟體破解實戰Demo – 010 Editor

Lab 2

- 目標：用Debugger來追蹤ncrackme.exe，找出程式作序號檢查的程式點
- 用Debugger開啟ncrackme.exe:

[File] → [Open]



Lab 2

- 方法一 – 利用字串引用來尋找：
- 按右鍵→
- “Search for”
- “All referenced text strings”

The screenshot shows a debugger window with assembly code on the left and a context menu on the right. The assembly code is for a function named `EBP=0012FFF0`. The context menu is open, showing options like `Backup`, `Copy`, `Binary`, `Assemble`, `Label`, `Comment`, `Add Header`, `Modify Variable`, `Breakpoint`, `Hit trace`, `Run trace`, `Go to`, `Follow in Dump`, `View call tree`, `Search for`, `Find references to`, `View`, `Copy to executable`, `Analysis`, `Bookmark`, and `Appearance`. The `Search for` option is selected, and a sub-menu is open showing options like `Name (label) in current module`, `Name in all modules`, `All Commands in all modules`, `All sequences in all modules`, `Command`, `Sequence of commands`, `Constant`, `Binary string`, `All intermodular calls`, `All commands`, `All sequences`, `All constants`, `All switches`, and `All referenced text strings`.

Address	Hex dump	ASCII
00405000	00 00 00 00 00 00 00 00 00 00 00 00 06 25 40 00
00405010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00405020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00405030	67 6F 6F 64 20 66 75 6E 63 74 69 6F 6E 2C 20 69	good funct

Lab 2

- 程式中會引用到的字串列表：

0040106D	PUSH ncrackme.00405080	ASCII "ncrackme"
00401079	PUSH ncrackme.00405064	ASCII "Registration successful."
00401095	PUSH ncrackme.00405050	ASCII "Registration fail."
004010BD	PUSH ncrackme.00405080	ASCII "ncrackme"
004010C2	PUSH ncrackme.00405030	ASCII "good function, i was cracked"
00401140	MOV DWORD PTR SS:[ESP+58],ncrackme.004040D8	ASCII "myWindowClass"
00401176	PUSH ncrackme.0040508C	ASCII "Newbie smallsize crackme - v1"
0040117B	PUSH ncrackme.004040D8	ASCII "myWindowClass"
00401368	PUSH EBP	<Initial CPU selection>
0040202A	PUSH ncrackme.004043CC	ASCII "<program name unknown>"
0040206C	PUSH ncrackme.004043C8	ASCII "..."
00402080	PUSH ncrackme.004043AC	ASCII "Runtime Error! Program: "
0040209E	PUSH ncrackme.004043A8	ASCII "[00]"
004020C6	PUSH ncrackme.00404380	ASCII "Microsoft Visual C++ Runtime Libr
004032DB	PUSH ncrackme.00404414	ASCII "user32.dll"
004032F2	PUSH ncrackme.00404408	ASCII "MessageBoxA"
00403303	PUSH ncrackme.004043F8	ASCII "GetActiveWindow"
0040330B	PUSH ncrackme.004043E4	ASCII "GetLastActivePopup"

- 在想追蹤的string上按右鍵→"Follow in dissembler"

0040106D	PUSH ncrackme.00405080	ASCII "ncrackme"		
00401079	PUSH ncrackme.00405064	ASCII "Registration successful."		
00401095	PUSH ncrackme.00405050	ASCII "Registration fail."	Follow in Disassembler	Enter
004010BD	PUSH ncrackme.00405080	ASCII "ncrackme"		
004010C2	PUSH ncrackme.00405030	ASCII "good function, i was cracked"	Search for text	
00401140	MOV DWORD PTR SS:[ESP+58],ncrackme.004040D8	ASCII "myWindowClass"		
00401176	PUSH ncrackme.0040508C	ASCII "Newbie smallsize crackme - v1"	Toggle breakpoint	F2
0040117B	PUSH ncrackme.004040D8	ASCII "myWindowClass"	Conditional breakpoint	Shift+F2
00401368	PUSH EBP	<Initial CPU selection>	Conditional log breakpoint	Shift+F4
0040202A	PUSH ncrackme.004043CC	ASCII "<program name unknown>"		
0040206C	PUSH ncrackme.004043C8	ASCII "..."		
00402080	PUSH ncrackme.004043AC	ASCII "Runtime Error! Program: "		
0040209E	PUSH ncrackme.004043A8	ASCII "[00]"	Set breakpoint on every command	

Lab 2

- 即可找到關鍵程式碼區塊：

```
00401050 . 817C24 08 1101 CMP DWORD PTR SS:[ESP+8],111
00401058 . 75 74 JNZ SHORT ncrackme.004010CE
0040105A . 8B4424 0C MOV EAX,DWORD PTR SS:[ESP+C]
0040105E . 66:3D EA03 CMP AX,3EA
00401062 . 75 42 JNZ SHORT ncrackme.004010A6
00401064 . E8 C7010000 CALL ncrackme.00401230
00401069 . 85C0 TEST EAX,EAX
0040106B . 6A 00 PUSH 0
0040106D . 68 80504000 PUSH ncrackme.00405080
00401072 . 75 1B JNZ SHORT ncrackme.0040108F
00401074 . A1 B8564000 MOV EAX,DWORD PTR DS:[4056B8]
00401079 . 68 64504000 PUSH ncrackme.00405064
0040107E . 50 PUSH EAX
0040107F . FF15 C0404000 CALL DWORD PTR DS:[<&USER32.MessageBoxA:
00401085 . E8 A6020000 CALL ncrackme.00401330
0040108A . 33C0 XOR EAX,EAX
0040108C . C2 1000 RETN 10
0040108F > 8B0D B8564000 MOV ECX,DWORD PTR DS:[4056B8]
00401095 . 68 50504000 PUSH ncrackme.00405050
0040109A . 51 PUSH ECX
0040109B . FF15 C0404000 CALL DWORD PTR DS:[<&USER32.MessageBoxA:
004010A1 . 33C0 XOR EAX,EAX
004010A3 . C2 1000 RETN 10
004010A6 > 66:3D EB03 CMP AX,3EB
004010AA . 75 22 JNZ SHORT ncrackme.004010CE
004010AC . A1 C0564000 MOV EAX,DWORD PTR DS:[4056C0]
004010B1 . 85C0 TEST EAX,EAX
004010B3 . 74 19 JE SHORT ncrackme.004010CE
004010B5 . 8B15 B8564000 MOV EDX,DWORD PTR DS:[4056B8]
004010BB . 6A 00 PUSH 0
004010BD . 68 80504000 PUSH ncrackme.00405080
004010C2 . 68 30504000 PUSH ncrackme.00405030
004010C7 . 52 PUSH EDX
004010C8 . FF15 C0404000 CALL DWORD PTR DS:[<&USER32.MessageBoxA:
004010CE > 33C0 XOR EAX,EAX
004010D0 . C2 1000 RETN 10
004010D3 . 90 NOP
004010D4 . 90 NOP
004010D5 . 90 NOP
```

```
Style = MB_OK|MB_APPLMODAL
Title = "ncrackme"

Text = "Registration successful."
hOwner => NULL
MessageBoxA

Text = "Registration fail."
hOwner => NULL
MessageBoxA

Style = MB_OK|MB_APPLMODAL
Title = "ncrackme"
Text = "good function, i was cracked"
hOwner => NULL
MessageBoxA
```

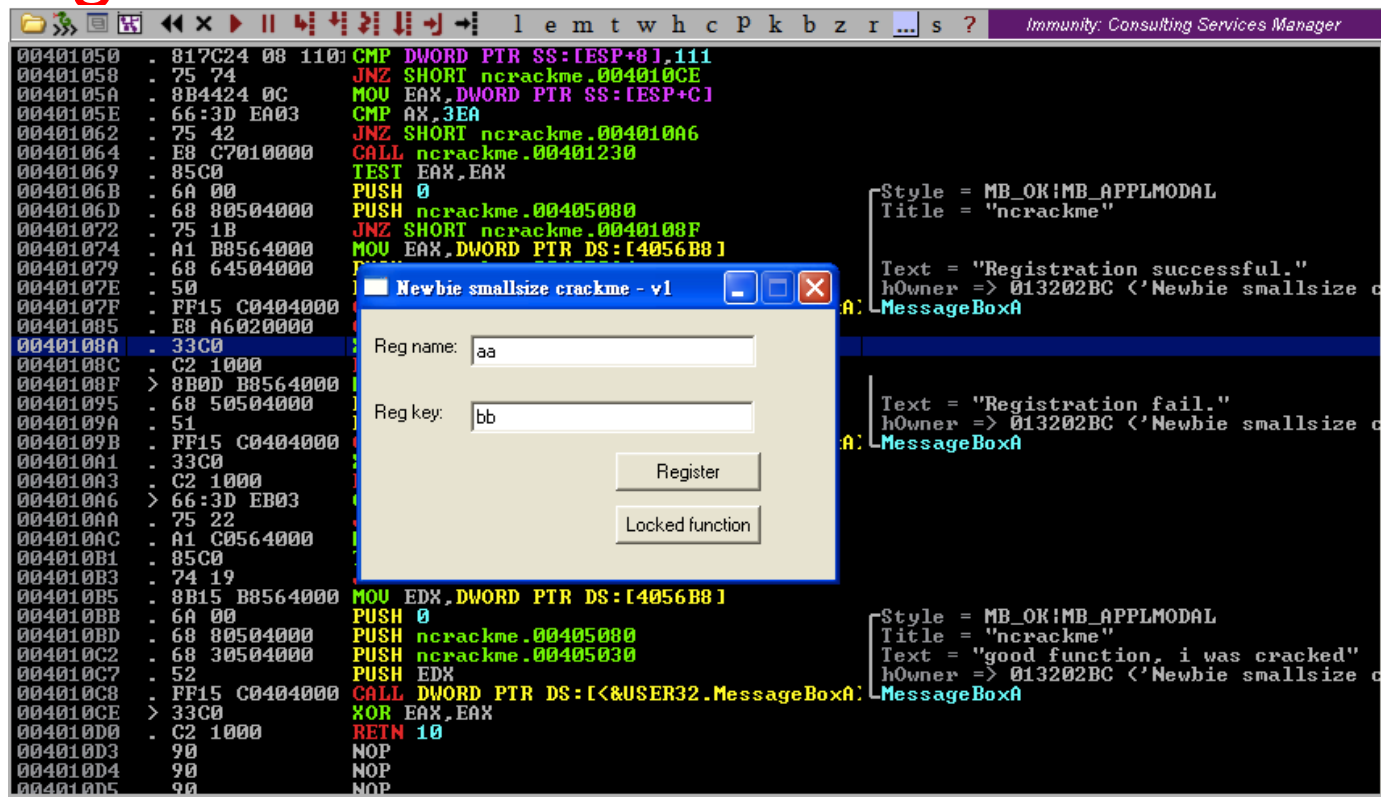

Lab 2

- 方法二— 利用API MessageBoxA來設置中斷點找尋：
- 按「**Ctrl + N**」，debugger會列出所有引用到的函式

0040402C	.rdata	Import	KERNEL32.HeapAlloc
00404080	.rdata	Import	KERNEL32.HeapCreate
0040407C	.rdata	Import	KERNEL32.HeapDestroy
00404088	.rdata	Import	KERNEL32.HeapFree
00404024	.rdata	Import	KERNEL32.HeapReAlloc
00404014	.rdata	Import	KERNEL32.LCMapStringA
00404010	.rdata	Import	KERNEL32.LCMapStringW
0040409C	.rdata	Import	USER32.LoadCursorA
00404098	.rdata	Import	USER32.LoadIconA
0040401C	.rdata	Import	KERNEL32.LoadLibraryA
004040C0	.rdata	Import	USER32.MessageBoxA
00401368	.text	Export	<ModuleEntryPoint>
00404018	.rdata	Import	KERNEL32.MultiByteToWideChar
004040CC	.rdata	Import	USER32.PostQuitMessage
004040A0	.rdata	Import	USER32.RegisterClassExA
0040408C	.rdata	Import	KERNEL32.RtlUnwind
00404070	.rdata	Import	KERNEL32.SetHandleCount
004040AC	.rdata	Import	USER32.ShowWindow
0040404C	.rdata	Import	KERNEL32.TerminateProcess
004040B8	.rdata	Import	USER32.TranslateMessage
00404054	.rdata	Import	KERNEL32.UnhandledExceptionFilter
004040B0	.rdata	Import	USER32.UpdateWindow
00404028	.rdata	Import	KERNEL32.VirtualAlloc
00404084	.rdata	Import	KERNEL32.VirtualFree
00404000	.rdata	Import	KERNEL32.WideCharToMultiByte
00404060	.rdata	Import	KERNEL32.WriteFile

Lab 2

- 按“F9”讓程式執行，跳出視窗後在“Reg name”，“Reg key”欄位任意輸入東西，按“Register”



The screenshot shows the Immunity Debugger interface. The assembly window on the left displays the following code:

```
00401050 - 817C24 08 1101 CMP DWORD PTR SS:[ESP+8],111
00401058 - 75 74 JNZ SHORT ncrackme.004010CE
0040105A - 8B4424 0C MOV EAX,DWORD PTR SS:[ESP+C]
0040105E - 66:3D EA03 CMP AX,3EA
00401062 - 75 42 JNZ SHORT ncrackme.004010A6
00401064 - E8 C7010000 CALL ncrackme.00401230
00401069 - 85C0 TEST EAX,EAX
0040106B - 6A 00 PUSH 0
0040106D - 68 80504000 PUSH ncrackme.00405080
00401072 - 75 1B JNZ SHORT ncrackme.0040108F
00401074 - A1 B8564000 MOV EAX,DWORD PTR DS:[4056B8]
00401079 - 68 64504000 PUSH ncrackme.00405080
0040107E - 50 POP EAX
0040107F - FF15 C0404000 JMP ncrackme.004040C0
00401085 - E8 A6020000 CALL ncrackme.00401230
0040108A - 33C0 XOR EAX,EAX
0040108C - C2 1000 RETN 10
0040108F > 8B0D B8564000 MOV EAX,DWORD PTR DS:[4056B8]
00401095 - 68 50504000 PUSH ncrackme.00405080
0040109A - 51 POP EAX
0040109B - FF15 C0404000 JMP ncrackme.004040C0
004010A1 - 33C0 XOR EAX,EAX
004010A3 - C2 1000 RETN 10
004010A6 > 66:3D EB03 CMP AX,3
004010AA - 75 22 JNZ SHORT ncrackme.004010A6
004010AC - A1 C0564000 MOV EAX,DWORD PTR DS:[4056B8]
004010B1 - 85C0 TEST EAX,EAX
004010B3 - 74 19 JZ SHORT ncrackme.004010C2
004010B5 - 8B15 B8564000 MOV EDI,DWORD PTR DS:[4056B8]
004010BB - 6A 00 PUSH 0
004010BD - 68 30504000 PUSH ncrackme.00405080
004010C2 - 68 30504000 PUSH ncrackme.00405030
004010C7 - 52 POP EDI
004010C8 - FF15 C0404000 JMP ncrackme.004040C0
004010CE > 33C0 XOR EAX,EAX
004010D0 - C2 1000 RETN 10
004010D3 - 90 NOP
004010D4 - 90 NOP
004010D5 - 90 NOP
```

The 'Newbie smallsize crackme - v1' dialog box is open, showing the following fields and buttons:

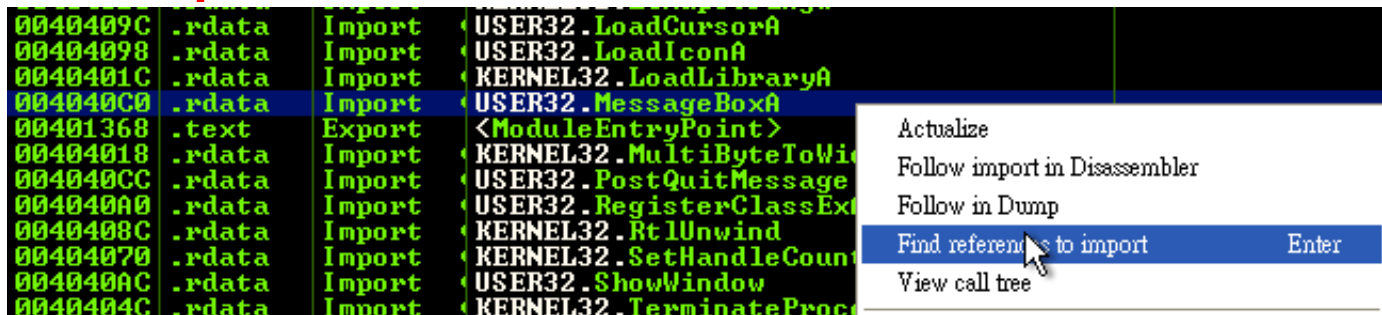
- Reg name: aa
- Reg key: bb
- Buttons: Register, Locked function

The background assembly window shows the following code:

```
00401050 - 817C24 08 1101 CMP DWORD PTR SS:[ESP+8],111
00401058 - 75 74 JNZ SHORT ncrackme.004010CE
0040105A - 8B4424 0C MOV EAX,DWORD PTR SS:[ESP+C]
0040105E - 66:3D EA03 CMP AX,3EA
00401062 - 75 42 JNZ SHORT ncrackme.004010A6
00401064 - E8 C7010000 CALL ncrackme.00401230
00401069 - 85C0 TEST EAX,EAX
0040106B - 6A 00 PUSH 0
0040106D - 68 80504000 PUSH ncrackme.00405080
00401072 - 75 1B JNZ SHORT ncrackme.0040108F
00401074 - A1 B8564000 MOV EAX,DWORD PTR DS:[4056B8]
00401079 - 68 64504000 PUSH ncrackme.00405080
0040107E - 50 POP EAX
0040107F - FF15 C0404000 JMP ncrackme.004040C0
00401085 - E8 A6020000 CALL ncrackme.00401230
0040108A - 33C0 XOR EAX,EAX
0040108C - C2 1000 RETN 10
0040108F > 8B0D B8564000 MOV EAX,DWORD PTR DS:[4056B8]
00401095 - 68 50504000 PUSH ncrackme.00405080
0040109A - 51 POP EAX
0040109B - FF15 C0404000 JMP ncrackme.004040C0
004010A1 - 33C0 XOR EAX,EAX
004010A3 - C2 1000 RETN 10
004010A6 > 66:3D EB03 CMP AX,3
004010AA - 75 22 JNZ SHORT ncrackme.004010A6
004010AC - A1 C0564000 MOV EAX,DWORD PTR DS:[4056B8]
004010B1 - 85C0 TEST EAX,EAX
004010B3 - 74 19 JZ SHORT ncrackme.004010C2
004010B5 - 8B15 B8564000 MOV EDI,DWORD PTR DS:[4056B8]
004010BB - 6A 00 PUSH 0
004010BD - 68 30504000 PUSH ncrackme.00405080
004010C2 - 68 30504000 PUSH ncrackme.00405030
004010C7 - 52 POP EDI
004010C8 - FF15 C0404000 JMP ncrackme.004040C0
004010CE > 33C0 XOR EAX,EAX
004010D0 - C2 1000 RETN 10
004010D3 - 90 NOP
004010D4 - 90 NOP
004010D5 - 90 NOP
```

Lab 2

- 在API函式上按右鍵，選擇“Find reference to import”



- 除錯器會列出程式中所有引用此API的位址，在想追蹤的位址上按“enter”

Address	Disassembly	Comment
0040107F	CALL DWORD PTR DS:[&USER32.MessageBoxA]	USER32.MessageBoxA
0040109B	CALL DWORD PTR DS:[&USER32.MessageBoxA]	USER32.MessageBoxA
004010C8	CALL DWORD PTR DS:[&USER32.MessageBoxA]	USER32.MessageBoxA

Lab 2

- 即可找到呼叫"MessageBoxA"的程式碼：

程式跳轉點

註冊成功的情況

註冊失敗的情況

Address	Disassembly	Comment
00401058	JNZ SHORT ncrackme.004010CE	
0040105A	MOV EAX,DWORD PTR SS:[ESP+4]	
0040105E	CMP AX,3EA	
00401062	JNZ SHORT ncrackme.004010A6	
00401064	CALL ncrackme.00401230	
00401069	TEST EAX,EAX	
0040106B	PUSH 0	
68 80504000	PUSH ncrackme.00405080	
75 1B	JNZ SHORT ncrackme.0040108F	
A1 B8564000	MOV EAX,DWORD PTR DS:[4056B8]	
68 64504000	PUSH ncrackme.00405064	
15 C0504000	PUSH EAX	
A6021000	CALL DWORD PTR DS:[&USER32.MessageBoxA]	Text = "Registration successful." hOwner => 013202BC (<'Newbie smallsize crackme - v1', MessageBoxA)
0040108A	XOR EAX,EAX	
0040108C	RETN 10	
0040108F	MOV EAX,DWORD PTR DS:[4056B8]	
68 64504000	PUSH ncrackme.00405050	
15 C0504000	PUSH EAX	
A6021000	CALL DWORD PTR DS:[&USER32.MessageBoxA]	Text = "Registration fail." hOwner => 013202BC (<'Newbie smallsize crackme - v1', MessageBoxA)
00401093	XOR EAX,EAX	
00401095	RETN 10	
004010A3	CMP AX,3EB	
004010A6	JNZ SHORT ncrackme.004010CE	
004010AA	MOV EAX,DWORD PTR DS:[4056C0]	
004010AC	TEST EAX,EAX	
004010B1	JE SHORT ncrackme.004010CE	
004010B3	MOV EDI,DWORD PTR DS:[4056B8]	
004010B5	PUSH 0	
004010BB	PUSH ncrackme.00405080	
004010BD	PUSH ncrackme.00405030	
004010C2	PUSH EDI	
004010C7	CALL DWORD PTR DS:[&USER32.MessageBoxA]	Text = "good function, i was cracked" hOwner => 013202BC (<'Newbie smallsize crackme - v1', MessageBoxA)
004010C8	XOR EAX,EAX	
004010CE	RETN 10	
004010D0	NOP	
004010D3	NOP	
004010D4	NOP	
004010D5	NOP	
004010D6	NOP	

現在，試著修改程式，讓我們不管輸入什麼序號，永遠都會顯示註冊成功 😊

Lab 2

- 動手時間 – 10 mins
- 將位址0x401072的指令由原本的
 - JNZ SHORT 0040108F，改為
 - NOP
- 按F9執行，觀察結果

課程內容

- 逆向工程與軟體破解概念簡介
- Lab 0
- 軟體破解前你必須知道的二三事
- Lab 1
- 通往破解點的道路 – Strings 與 API
- Lab 2
- 軟體破解實戰Demo – 010 Editor

demo

- 010序號驗證機制破解 (僅供教學目的)

如何儲存修改過的檔案

- 之前的Lab都在記憶體中作修改，下次重新啟動效果就消失

The screenshot displays the Immunity Debugger interface. On the left, the assembly window shows a list of instructions with their addresses and hex values. The instruction at address 00401058, `JNZ SHORT 8B4424 0C`, is highlighted. A context menu is open over this instruction, listing various actions. The 'Copy to executable' option is selected, and a sub-menu is visible showing 'Selection' and 'All modifications'. On the right, the C++ source code window shows the corresponding code for the assembly, including a `DefWindowProcA` function and a `PostQuitMessage` call. The code is written in a dark theme.

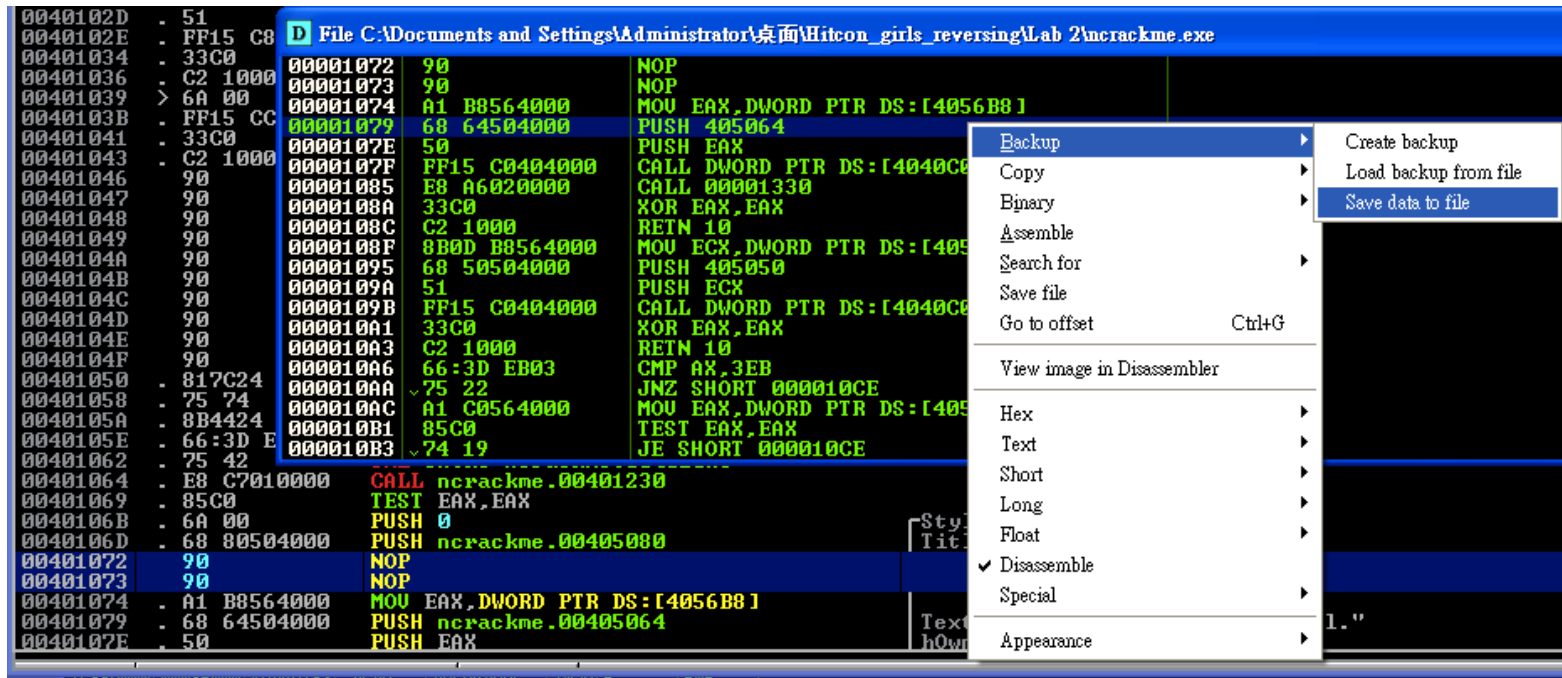
```
00401014 . 8B5424 0C MOV EDX, DWORD PTR DS:[4056B8]
00401018 . 50 PUSH EAX
00401019 . 8B4424 08 MOV EAX, DWORD PTR DS:[4056B8]
0040101D . 52 PUSH EDI
0040101E . 51 PUSH ESI
0040101F . 50 PUSH EAX
00401020 . FF15 C4404000 CALL DWORD PTR DS:[C4404000]
00401026 . C2 1000 RETN 10
00401029 > 8B4C24 04 MOV ECX, DWORD PTR DS:[4056B8]
0040102D . 51 PUSH EDI
0040102E . FF15 C8404000 CALL DWORD PTR DS:[C8404000]
00401034 . 33C0 XOR EAX, EAX
00401036 . C2 1000 RETN 10
00401039 > 6A 00 PUSH 0
0040103B . FF15 CC404000 CALL DWORD PTR DS:[CC404000]
00401041 . 33C0 XOR EAX, EAX
00401043 . C2 1000 RETN 10
00401046 . 90 NOP
00401047 . 90 NOP
00401048 . 90 NOP
00401049 . 90 NOP
0040104A . 90 NOP
0040104B . 90 NOP
0040104C . 90 NOP
0040104D . 90 NOP
0040104E . 90 NOP
0040104F . 90 NOP
00401050 . 817C24 08 1101 CMP DWORD PTR DS:[7C240811], 0
00401058 . 75 74 JNZ SHORT 8B4424 0C
0040105A . 8B4424 0C MOV EAX, DWORD PTR DS:[4056B8]
0040105E . 66:3D EA03 CMP AX, 3
00401062 . 75 42 JNZ SHORT 8B4424 0C
00401064 . E8 C7010000 CALL ncrackme.00401030
00401069 . 85C0 TEST EAX, EAX
0040106B . 6A 00 PUSH 0
0040106D . 68 80504000 PUSH ncrackme.00405080
00401072 . 90 NOP
00401073 . 90 NOP
00401074 . A1 B8564000 MOV EAX, DWORD PTR DS:[4056B8]
00401079 . 68 64504000 PUSH ncrackme.00405064
0040107E . 50 PUSH EAX
0040107F . FF15 C0404000 CALL DWORD PTR DS:[C0404000]
00401085 . E8 A6020000 CALL ncrackme.00401030
0040108A . 33C0 XOR EAX, EAX
0040108C . C2 1000 RETN 10
0040108F > 8B0D B8564000 MOV ECX, DWORD PTR DS:[4056B8]
00401095 . 68 50504000 PUSH ncrackme.00405050
0040109A . 51 PUSH ESI
0040109B . FF15 C0404000 CALL DWORD PTR DS:[C0404000]
004010A1 . 33C0 XOR EAX, EAX
```

Backup
Copy
Binary
Undo selection Alt+BkSp
Assemble Space
Label :
Comment ;
Add Header
Modify Variable
Breakpoint
Hit trace
Run trace
New origin here Ctrl+Gray *
Go to
Follow in Dump
Search for
Find references to
View
Copy to executable Selection
Analysis All modifications
Bookmark
Appearance

```
lParam  
wParam  
Message  
hWnd  
DefWindowProcA  
Case 10 of switch 00401006  
hWnd  
DestroyWindow  
ExitCode = 0; Case 2 of switch 00401006  
PostQuitMessage  
Style = MB_OK|MB_APPLMODAL  
Title = "ncrackme"  
Text = "Registration successful."  
hOwner => NULL  
MessageBoxA  
Text = "Registration fail."  
hOwner => NULL  
MessageBoxA
```


如何儲存修改過的檔案

- 按右鍵 → “Backup” → “Save data to file”
- 另存新檔即可
- 現在執行看新存的檔案吧



Charles@teamt5.net

QUESTION?

