工具：VS2012  OD  
  
直接进入正题吧：  
简单介绍下基础知识，根据三个（结构体）：

* TEB结构---线程信息块
* SEH结构---结构化异常处理
* PEB结构---枚举用户模块列表

把思路扩展开。  
  
**FS段寄存器指向当前的TEB结构**:

* 在TEB偏移0x00处指向SEH链指针
* 在TEB偏移0x18处是指向TEB结构体自身
* 在TEB偏移0x30处是指向PEB结构体指针

**反调试思路**：

1. 添加异常链
2. 制作异常
3. 在异常回调函数得到BeingDebugged(PEB结构体第三个成员)值   1 代表正在被调试  0 代表正常运行
4. 备份正确的异常返回地址
5. 根据标志位执行不同的SEH异常处理流程

**俩个内联汇编块**：

代码:

\_\_asm{ /\* 添加SEH异常链A \*/

        push XXXXXXX;      //将回调函数XXXXXXX压入堆栈

        push dword ptr fs:[0];    //将先前的EXCEPTION\_REGISTRATION结构保存

        mov dword ptr fs:[0],esp;  // 将线程信息块（TIB）中的第一个DWORD指向

                                // 新的EXCEPTION\_REGISTRATION结构。

    };

接下来就代码了。  是不是感觉有点简单粗暴。。。。。。  
嗯 是这样滴 但**理解本文**必要的知识点都列出了。  
有不明白的地方回帖   
-------------------------------------反调试------------------------------------  
**主函数**：

代码:

\_\_asm{ /\* 添加SEH异常链A \*/

push ExpHandel\_A;

push dword ptr fs:[0];

mov dword ptr fs:[0],esp;

};

\_\_asm{ /\* 添加SEH异常链B \*/

push ExpHandel\_B;

push dword ptr fs:[0];

mov dword ptr fs:[0],esp;

};

// 2. 人为制造异常

int \*pTest = nullptr;

\*pTest = 0;

return 0;

**ExpHandel\_B回调**：

代码:

//可注释  存在的意义 是更好理解SEH执行流程

EXCEPTION\_DISPOSITION ExpHandel\_B(

    EXCEPTION\_RECORD  \*pExceptionRecord,   // 异常状态描述

    EXCEPTION\_REGISTRATION\_RECORD \*pEstablisherFrame,  // 异常注册框架

    CONTEXT  \*pContextRecord,     // 返回线程上下文

    PVOID  pDispatcherContext ) // 分发器上下文（系统使用，无需关注）

{

    // 返回“运行下一个异常处理器”

    return ExceptionContinueSearch;

}

ExpHandel\_A回调：

代码:

// 1. 获取调试标志位

BYTE IsBeginDebug = false;

\_\_asm {

push eax;

mov eax, dword ptr fs:[0x30]; // 获取PEB

mov al,byte ptr ds:[eax+2]; // 获取PEB.BeginDebug

mov IsBeginDebug,al;

pop eax;

};

// 2. 备份正确的异常返回地址

g\_funExceptionReturn = (void (\*)())pContextRecord->Eip;

// 3. 根据标志位执行不同的SEH异常处理流程

if ( IsBeginDebug )

pContextRecord->Eip = (DWORD)WrongFun;

else

pContextRecord->Eip = (DWORD)RightFun;

// 4. 返回“已经正常处理”

return ExceptionContinueExecution;

正常运行状态：

代码:

**被调试状态**：

代码:

代码:

void RightFun()

{

MessageBox(0,L"正常运行！",L"SEH反调试测试",MB\_OK);

// 删除异常处理器

// 运行至此处时栈顶的信息是EXCEPTION\_REGISTRATION\_RECORD，里面

// 保存着其最初运行的已成处理器，我们将其保存到FS:[0]处后就相当于

// 摘除了前面注册的SEH异常链信息。

\_\_asm pop dword ptr fs:[0];

g\_funExceptionReturn();

}

这样反调试便是完成了。

-------------------------------------反反调试------------------------------------  
  
将生成的程序拖入OD   我们将注意力放在寄存器窗口  
  
来到这里（PS:我在前文说什么来着？）：  
  
  
我们再去PEB找找调试标记 看是否如我所说的那样。  
点击上图30H位置  回车   或者 CTRL + G  
  
  
我们再回到开始 直接找到MAIN函数 选中 回车  
  
  
左键选中第一个（第二个也行） 右键 -数据窗口跟随 - 立即常数    
我们再把注意防止**数据窗口** 右键-反汇编  
  
然后左键点击

代码:

002813DE   /E9 2D0D0000     JMP ExpHandel\_A

当然 你也可以点击

代码:

00281488   /E9 430D0000     JMP ExpHandel\_B

由于我知道 你难道还不知道？  
操作放在 **ExpHandel\_A**回调函数内   直接选中 回车  
当然了 在不知道源码的情况 我们每个都看一下最好   
懒一点 根据经验、运气之类的随机吧。。。  
红色高亮为关键代码 &nbsp    
为什么？  
这段代码 就是检测是否在调试状态**TEST**检测！  我们肯定在调试状态。  
至于怎么做 这个我就不说了 很简单了。

代码:

CPU Disasm

地址 十六进制数据 指令 注释

00282110 55 PUSH EBP ; INT SEH\_AntiDebug\_Native.ExpHandel\_A(void)

00282111 8BEC MOV EBP,ESP

00282113 81EC D0000000 SUB ESP,0D0

00282119 53 PUSH EBX

0028211A 56 PUSH ESI

0028211B 57 PUSH EDI

0028211C 8DBD 30FFFFFF LEA EDI,[EBP-0D0]

00282122 B9 34000000 MOV ECX,34

00282127 B8 CCCCCCCC MOV EAX,CCCCCCCC

0028212C F3:AB REP STOS DWORD PTR ES:[EDI]

0028212E A1 00C02900 MOV EAX,DWORD PTR DS:[\_\_security\_cookie]

00282133 33C5 XOR EAX,EBP

00282135 8945 FC MOV DWORD PTR SS:[EBP-4],EAX

00282138 C645 F7 00 MOV BYTE PTR SS:[EBP-9],0

0028213C 50 PUSH EAX

0028213D 64:A1 30000000 MOV EAX,DWORD PTR FS:[30]

00282143 3E:8A40 02 MOV AL,BYTE PTR DS:[EAX+2]

00282147 8845 F7 MOV BYTE PTR SS:[EBP-9],AL

0028214A 58 POP EAX

0028214B 8B45 10 MOV EAX,DWORD PTR SS:[EBP+10]

0028214E 8B88 B8000000 MOV ECX,DWORD PTR DS:[EAX+0B8]

00282154 890D 80D12900 MOV DWORD PTR DS:[g\_funExceptionReturn],

0028215A 0FB645 F7 MOVZX EAX,BYTE PTR SS:[EBP-9]

0028215E 85C0 TEST EAX,EAX

00282160 74 0F JE SHORT 00282171

00282162 8B45 10 MOV EAX,DWORD PTR SS:[EBP+10]

00282165 C780 B8000000 B MOV DWORD PTR DS:[EAX+0B8],002817BC

0028216F EB 0D JMP SHORT 0028217E

00282171 8B45 10 MOV EAX,DWORD PTR SS:[EBP+10]

00282174 C780 B8000000 E MOV DWORD PTR DS:[EAX+0B8],002813E8

0028217E 33C0 XOR EAX,EAX

00282180 5F POP EDI

00282181 5E POP ESI

00282182 5B POP EBX

00282183 8B4D FC MOV ECX,DWORD PTR SS:[EBP-4]

00282186 33CD XOR ECX,EBP

00282188 E8 36EFFFFF CALL 002810C3

0028218D 81C4 D0000000 ADD ESP,0D0

00282193 3BEC CMP EBP,ESP

00282195 E8 7EF4FFFF CALL 00281618

0028219A 8BE5 MOV ESP,EBP

0028219C 5D POP EBP

0028219D C3 RETN

成功！  
-------------------------  
源码：  
[SEH\_AntiDebug\_Native.rar](http://bbs.pediy.com/attachment.php?attachmentid=101529&d=1448631046" \o "名称:  SEH_AntiDebug_Native.rar查看次数: 144文件大小:  15.6 KB).