

窗口类名	EnumWindow	ZwQueryInformationProcess	EnableWindow	关 闭(C)
IsDebuggerPresent	SeDebugPrivilege	ZwSetInformationThread		
枚举进程	NTQueryObject	OutputDebugString		关 于(A)
父进程Explorer	断点检测	单步异常		
GetTickCount	函数断点检测	保护页Guard Pages		
GetStartupInfo	Checksum	HardwareBreakpoint		
PebFlags	BlockInput	GetEntryPoint		

```

1 // DetectODDlg.cpp : implementation file
2 //
3
4 #include "stdafx.h"
5 #include "DetectOD.h"
6 #include "DetectODDlg.h"
7 #include "Shlwapi.h"
8 #include "tlhelp32.h"
9 #include "Windows.h"
10 #include "Winable.h"
11 #include "eh.h"
12 #ifdef _DEBUG
13 #define new DEBUG_NEW
14 #undef THIS_FILE
15 static char THIS_FILE[] = __FILE__;
16 #endif
17 static DWORD NewEip;
18 ///////////////////////////////////////////////////
19 // CAboutDlg dialog used for App About
20
21 class CAboutDlg : public CDialog
22 {
23 public:
24     CAboutDlg();
25
26     // Dialog Data
27    //{{AFX_DATA(CAboutDlg)
28     enum { IDD = IDD_ABOUTBOX };
29     //}}AFX_DATA
30
31     // ClassWizard generated virtual function overrides
32    //{{AFX_VIRTUAL(CAboutDlg)
33 protected:
34     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
35     //}}AFX_VIRTUAL
36
37     // Implementation
38 protected:
39    //{{AFX_MSG(CAboutDlg)
40     afx_msg void OnMypage();
41     afx_msg void OnMouseMove(UINT nFlags, CPoint point);
42     virtual BOOL OnInitDialog();
43     afx_msg void OnComeon();
44     afx_msg void OnMyicon();
45     //}}AFX_MSG
46     DECLARE_MESSAGE_MAP()
47 };
48
49 CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
50 {
51     //{{AFX_DATA_INIT(CAboutDlg)
52     //}}AFX_DATA_INIT
53 }
54
55 void CAboutDlg::DoDataExchange(CDataExchange* pDX)

```

```

56 {
57     CDialog::DoDataExchange(pDX);
58     //{AFX_DATA_MAP(CAboutDlg)
59     //}}AFX_DATA_MAP
60 }
61
62 BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
63     //{AFX_MSG_MAP(CAboutDlg)
64     ON_BN_CLICKED(IDC_MYPAGE, OnMypage)
65     ON_WM_MOUSEMOVE()
66     ON_BN_CLICKED(IDC_COMEON, OnComeon)
67     ON_BN_CLICKED(IDC_MYICON, OnMyicon)
68     //}}AFX_MSG_MAP
69 END_MESSAGE_MAP()
70
71 ///////////////////////////////////////////////////
72 // CDetectODDlg dialog
73
74 CDetectODDlg::CDetectODDlg(CWnd* pParent /*=NULL*/)
75 : CDialog(CDetectODDlg::IDD, pParent)
76 {
77     //{AFX_DATA_INIT(CDetectODDlg)
78     // NOTE: the ClassWizard will add member initialization here
79     //}}AFX_DATA_INIT
80     // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
81     m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
82 }
83
84 void CDetectODDlg::DoDataExchange(CDataExchange* pDX)
85 {
86     CDialog::DoDataExchange(pDX);
87     //{AFX_DATA_MAP(CDetectODDlg)
88     // NOTE: the ClassWizard will add DDX and DDV calls here
89     //}}AFX_DATA_MAP
90 }
91
92 BEGIN_MESSAGE_MAP(CDetectODDlg, CDialog)
93     //{AFX_MSG_MAP(CDetectODDlg)
94     ON_WM_SYSCOMMAND()
95     ON_WM_PAINT()
96     ON_WM_QUERYDRAGICON()
97     ON_BN_CLICKED(IDC_WNDCLS, OnWndcls)
98     ON_BN_CLICKED(IDC_ISDEBUGGERPRESENT, OnIsdebuggerpresent)
99     ON_BN_CLICKED(IDC_ENUMWINDOW, OnEnumwindow)
100    ON_BN_CLICKED(IDC_EnumProcess, OnEnumProcess)
101    ON_BN_CLICKED(IDC_Explorer, OnExplorer)
102    ON_BN_CLICKED(IDC_GetTickCount, OnGetTickCount)
103    ON_BN_CLICKED(IDC_GetStartupInfo, OnGetStartupInfo)
104    ON_BN_CLICKED(IDC_PEBFLAGS, OnPebflags)
105    ON_BN_CLICKED(IDC_CHECKREMOTEDBUGGERPRESENT, OnCheckremotedebuggerpresent)
106    ON_BN_CLICKED(IDC_SetUnhandledExceptionFilter, OnSetUnhandledExceptionFilter)
107    ON_BN_CLICKED(IDC_ZwQueryInformationProcess, OnZwQueryInformationProcess)
108    ON_BN_CLICKED(IDC_SeDebugPrivilege, OnSeDebugPrivilege)
109    ON_BN_CLICKED(IDC_NTQueryObject, OnNTQueryObject)
110    ON_BN_CLICKED(IDC_DetectBreakpoints, OnDetectBreakpoints)
111    ON_BN_CLICKED(IDC_DetectFuncBreakpoints, OnDetectFuncBreakpoints)
112    ON_BN_CLICKED(IDC_BlockInput, OnBlockInput)
113    ON_BN_CLICKED(IDC_CHECKSUM, OnChecksum)
114    ON_BN_CLICKED(IDC_EnableWindow, OnEnableWindow)
115    ON_BN_CLICKED(IDC_ZwSetInformationThread, OnZwSetInformationThread)
116    ON_BN_CLICKED(IDC_OutputDebugString, OnOutputDebugString)
117    ON_BN_CLICKED(IDC_GetEntryPoint, OnGetEntryPoint)
118    ON_BN_CLICKED(IDC_TrapFlag, OnTrapFlag)
119    ON_BN_CLICKED(IDC_GuardPages, OnGuardPages)
120    ON_BN_CLICKED(IDC_HARDWAREBREAKPOINT, OnHardwarebreakpoint)
121    ON_BN_CLICKED(IDC_ABOUT, OnAbout)
122    ON_BN_CLICKED(IDC_MYPAGE2, OnMypage2)
123    //}}AFX_MSG_MAP
124 END_MESSAGE_MAP()
125
126 ///////////////////////////////////////////////////
127 // CDetectODDlg message handlers
128
129 BOOL CDetectODDlg::OnInitDialog()

```

```

130 {
131     CDialog::OnInitDialog();
132
133     // Add "About..." menu item to system menu.
134
135     // IDM_ABOUTBOX must be in the system command range.
136     ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
137     ASSERT(IDM_ABOUTBOX < 0xF000);
138
139     CMenu* pSysMenu = GetSystemMenu(FALSE);
140     if (pSysMenu != NULL)
141     {
142         CString strAboutMenu;
143         strAboutMenu.LoadString(IDS_ABOUTBOX);
144         if (!strAboutMenu.IsEmpty())
145         {
146             pSysMenu->AppendMenu(MF_SEPARATOR);
147             pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
148         }
149     }
150
151     // Set the icon for this dialog. The framework does this automatically
152     // when the application's main window is not a dialog
153     // SetIcon(m_hIcon, TRUE);         // Set big icon
154     // SetIcon(m_hIcon, FALSE);        // Set small icon
155
156     // TODO: Add extra initialization here
157     SetClassLong(m_hWnd, GCL_HICON, (LONG)(LoadIcon(AfxGetApp()-
>m_hInstance, MAKEINTRESOURCE(IDI_DOG))));
158     return TRUE; // return TRUE unless you set the focus to a control
159 }
160
161 void CDetectODDlg::OnSysCommand(UINT nID, LPARAM lParam)
162 {
163     if ((nID & 0xFFF0) == IDM_ABOUTBOX)
164     {
165         CAboutDlg dlgAbout;
166         dlgAbout.DoModal();
167     }
168     else
169     {
170         CDialog::OnSysCommand(nID, lParam);
171     }
172 }
173
174 // If you add a minimize button to your dialog, you will need the code below
175 // to draw the icon. For MFC applications using the document/view model,
176 // this is automatically done for you by the framework.
177
178 void CDetectODDlg::OnPaint()
179 {
180     if (IsIconic())
181     {
182         CPaintDC dc(this); // device context for painting
183
184         SendMessage(WM_ICONERASEBKGND, (LPARAM) dc.GetSafeHdc(), 0);
185
186         // Center icon in client rectangle
187         int cxIcon = GetSystemMetrics(SM_CXICON);
188         int cyIcon = GetSystemMetrics(SM_CYICON);
189         CRect rect;
190         GetClientRect(&rect);
191         int x = (rect.Width() - cxIcon + 1) / 2;
192         int y = (rect.Height() - cyIcon + 1) / 2;
193
194         // Draw the icon
195         dc.DrawIcon(x, y, m_hIcon);
196     }
197     else
198     {
199         CDialog::OnPaint();
200     }
201 }
202

```

```

203 // The system calls this to obtain the cursor to display while the user drags
204 // the minimized window.
205 HCURSOR CDetectODDlg::OnQueryDragIcon()
206 {
207     return (HCURSOR) m_hIcon;
208 }
209
210 void CDetectODDlg::OnWndcls()
211 {
212     // TODO: Add your control notification handler code here
213     HWND hWnd;
214     if(hWnd==::FindWindow("OllyDbg",NULL))
215     {
216         MessageBox("发现OD");
217     }else{
218         MessageBox("没发现OD");
219     }
220 }
221
222 void CDetectODDlg::OnIsdebuggerpresent()
223 {
224     // TODO: Add your control notification handler code here
225     if(IsDebuggerPresent())
226     {
227         MessageBox("发现OD");
228     }
229     else
230     {
231         MessageBox("没有OD");
232     }
233 }
234 /*****/
235 BOOL CALLBACK EnumWindowsProc(
236     HWND hwnd,      // handle to parent window
237     LPARAM lParam    // application-defined value
238 )
239 {
240     char ch[100];
241     CString str="Ollydbg";
242     if(IsWindowVisible(hwnd))
243     {
244         ::GetWindowText(hwnd,ch,100);
245         //AfxMessageBox(ch);
246         if(::StrStrI(ch,str))
247         {
248             AfxMessageBox("发现OD");
249             return FALSE;
250         }
251     }
252     return TRUE;
253 }
254
255 void CDetectODDlg::OnEnumwindow()
256 {
257     // TODO: Add your control notification handler code here
258     EnumWindows(EnumWindowsProc,NULL);
259     AfxMessageBox("枚举窗口结束, 未提示发现OD, 则没有OD");
260 }
261
262 /*****/
263 void CDetectODDlg::OnEnumProcess()
264 {
265     // TODO: Add your control notification handler code here
266
267     HANDLE hwnd;
268     PROCESSENTRY32 tp32; //结构体
269     CString str="OLLYDBG.EXE";
270     BOOL bFindOD=FALSE;
271     hwnd=::CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,NULL);
272     if(INVALID_HANDLE_VALUE!=hwnd)
273     {
274         Process32First(hwnd,&tp32);
275         do{
276             if(0==lstrcmpi(str,tp32.szExeFile))

```

```

277     {
278         AfxMessageBox("发现OD");
279         bFindOD=TRUE;
280         break;
281     }
282     }while(Process32Next(hwnd,&tp32));
283     if(!bFindOD)
284         AfxMessageBox("没有OD");
285     }
286     CloseHandle(hwnd);
287 }
288
289 void CDetectODDlg::OnExplorer()
290 {
291     // TODO: Add your control notification handler code here
292     HANDLE hwnd;
293     PROCESSENTRY32 tp32; //结构体
294     CString str="Explorer.EXE";
295
296     DWORD ExplorerID;
297     DWORD SelfID;
298     DWORD SelfParentID;
299     SelfID=GetCurrentProcessId();
300     ::GetWindowThreadProcessId(::FindWindow("Progman",NULL),&ExplorerID);
301     hwnd=::CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,NULL);
302     if(INVALID_HANDLE_VALUE!=hwnd)
303     {
304         Process32First(hwnd,&tp32);
305         do{
306             if(0==lstrcmp(str,tp32.szExeFile))
307             {
308                 // ExplorerID=tp32.th32ProcessID;
309                 // AfxMessageBox("aaa");
310             }
311             if(SelfID==tp32.th32ProcessID)
312             {
313                 SelfParentID=tp32.th32ParentProcessID;
314             }
315             }while(Process32Next(hwnd,&tp32));
316
317         str.Format("本进程: %d 父进程: %d Explorer进程: %d",SelfID,SelfParentID,ExplorerID);
318         MessageBox(str);
319         if(ExplorerID==SelfParentID)
320         {
321             AfxMessageBox("没有OD");
322         }
323         else
324         {
325             AfxMessageBox("发现OD");
326         }
327     }
328     CloseHandle(hwnd);
329 }
330
331 void CDetectODDlg::OnGetTickCount()
332 {
333     // TODO: Add your control notification handler code here
334     DWORD dTime1;
335     DWORD dTime2;
336     dTime1=GetTickCount();
337     GetCurrentProcessId();
338     GetCurrentProcessId();
339     GetCurrentProcessId();
340     GetCurrentProcessId();
341     dTime2=GetTickCount();
342     if(dTime2-dTime1>100)
343     {
344         AfxMessageBox("发现OD");
345     }
346     else{
347         AfxMessageBox("没有OD");
348     }
349 }

```

```

350
351 void CDetectODDlg::OnGetStartupInfo()
352 {
353     // TODO: Add your control notification handler code here
354     STARTUPINFO info={0};
355     GetStartupInfo(&info);
356     if(info.dwX!=0 || info.dwY!=0 || info.dwXCountChars!=0 || info.dwYCountChars!=0
357         || info.dwFillAttribute!=0 || info.dwXSize!=0 || info.dwYSize!=0)
358     {
359         AfxMessageBox("发现OD");
360     }
361     else{
362         AfxMessageBox("没有OD");
363     }
364 }
365
366
367 //*****
368 typedef ULONG NTSTATUS;
369 typedef ULONG PPEB;
370 typedef ULONG KAFFINITY;
371 typedef ULONG KPRIORITY;
372
373 typedef struct _PROCESS_BASIC_INFORMATION { // Information Class 0
374     NTSTATUS ExitStatus;
375     PPEB PebBaseAddress;
376     KAFFINITY AffinityMask;
377     KPRIORITY BasePriority;
378     ULONG UniqueProcessId;
379     ULONG InheritedFromUniqueProcessId;
380 } PROCESS_BASIC_INFORMATION, *PPROCESS_BASIC_INFORMATION;
381
382 typedef enum _PROCESSINFOCLASS {
383     ProcessBasicInformation, // 0 Y N
384     ProcessQuotaLimits, // 1 Y Y
385     ProcessIoCounters, // 2 Y N
386     ProcessVmCounters, // 3 Y N
387     ProcessTimes, // 4 Y N
388     ProcessBasePriority, // 5 N Y
389     ProcessRaisePriority, // 6 N Y
390     ProcessDebugPort, // 7 Y Y
391     ProcessExceptionPort, // 8 N Y
392     ProcessAccessToken, // 9 N Y
393     ProcessLdtInformation, // 10 Y Y
394     ProcessLdtSize, // 11 N Y
395     ProcessDefaultHardErrorMode, // 12 Y Y
396     ProcessIoPortHandlers, // 13 N Y
397     ProcessPooledUsageAndLimits, // 14 Y N
398     ProcessWorkingSetWatch, // 15 Y Y
399     ProcessUserModeIoPL, // 16 N Y
400     ProcessEnableAlignmentFaultFixup, // 17 N Y
401     ProcessPriorityClass, // 18 N Y
402     ProcessWx86Information, // 19 Y N
403     ProcessHandleCount, // 20 Y N
404     ProcessAffinityMask, // 21 N Y
405     ProcessPriorityBoost, // 22 Y Y
406     ProcessDeviceMap, // 23 Y Y
407     ProcessSessionInformation, // 24 Y Y
408     ProcessForegroundInformation, // 25 N Y
409     ProcessWow64Information // 26 Y N
410 } PROCESSINFOCLASS;
411
412 typedef NTSTATUS (_stdcall *ZwQueryInformationProcess)(
413     HANDLE ProcessHandle,
414     PROCESSINFOCLASS ProcessInformationClass,
415     PVOID ProcessInformation,
416     ULONG ProcessInformationLength,
417     PULONG ReturnLength
418 ); //定义函数指针
419 void CDetectODDlg::OnPebflags()
420 {
421     // TODO: Add your control notification handler code here
422
423     //定义函数指针变量

```

```

424 ZwQueryInformationProcess MyZwQueryInformationProcess;
425
426 HANDLE hProcess = NULL;
427 PROCESS_BASIC_INFORMATION pbi = {0};
428 ULONG peb = 0;
429 ULONG cnt = 0;
430 ULONG PebBase = 0;
431 ULONG AddrBase;
432 BOOL bFoundOD=FALSE;
433 WORD flag;
434 DWORD dwFlag;
435 DWORD bytesrw;
436 DWORD ProcessId=GetCurrentProcessId();
437 hProcess = OpenProcess(PROCESS_QUERY_INFORMATION | PROCESS_VM_READ, FALSE,
ProcessId);
438 if (hProcess != NULL) {
439     //函数指针变量赋值
440     MyZwQueryInformationProcess=
(ZwQueryInformationProcess)GetProcAddress(LoadLibrary("ntdll.dll"),"ZwQueryInformation
Process");
441     //函数指针变量调用
442     if (MyZwQueryInformationProcess(
443         hProcess,
444         ProcessBasicInformation,
445         &pbi,
446         sizeof(PROCESS_BASIC_INFORMATION),
447         &cnt) == 0)
448     {
449         PebBase = (ULONG)pbi.PebBaseAddress;
450         AddrBase=PebBase;
451         if (ReadProcessMemory(hProcess, (LPCVOID)(PebBase+0x68),&flag,2,&bytesrw)
&& bytesrw==2)
452         { //PEB.NtGlobalFlag
453             if(0x70==flag){
454                 bFoundOD=TRUE;
455             }
456         }
457         if (ReadProcessMemory(hProcess, (LPCVOID)(PebBase+0x18),&dwFlag,4,&bytesrw)
&& bytesrw==4)
458         {
459             AddrBase=dwFlag;
460         }
461         if (ReadProcessMemory(hProcess, (LPCVOID)(AddrBase+0x0c),&flag,2,&bytesrw)
&& bytesrw==2)
462         { //PEB.ProcessHeap.Flags
463             if(2!=flag){
464                 bFoundOD=TRUE;
465             }
466         }
467         if (ReadProcessMemory(hProcess, (LPCVOID)(AddrBase+0x10),&flag,2,&bytesrw)
&& bytesrw==2)
468         { //PEB.ProcessHeap.ForceFlags
469             if(0!=flag){
470                 bFoundOD=TRUE;
471             }
472         }
473         if(bFoundOD==FALSE)
474         {
475             AfxMessageBox("没有OD");
476         }
477         else
478         {
479             AfxMessageBox("发现OD");
480         }
481     }
482     CloseHandle(hProcess);
483 }
484 }
485
486 //*****
487 typedef BOOL (WINAPI *CHECK_REMOTE_DEBUGGER_PRESENT)(HANDLE, PBOOL);
488
489 void CDetectODDlg::OnCheckremotedebuggerpresent()
490 {

```

```

491 // TODO: Add your control notification handler code here
492 HANDLE hProcess;
493 HINSTANCE hModule;
494 BOOL bDebuggerPresent = FALSE;
495 CHECK_REMOTE_DEBUGGER_PRESENT CheckRemoteDebuggerPresent;
496 hModule = GetModuleHandleA("Kernel32");
497 CheckRemoteDebuggerPresent =
498 (CHECK_REMOTE_DEBUGGER_PRESENT)GetProcAddress(hModule,
"CheckRemoteDebuggerPresent");
499 hProcess = GetCurrentProcess();
500 CheckRemoteDebuggerPresent(hProcess,&bDebuggerPresent);
501 if(bDebuggerPresent==TRUE)
502 {
503     AfxMessageBox("发现OD");
504 }
505 else
506 {
507     AfxMessageBox("没有OD");
508 }
509 }
510 //*****
511 typedef NTSTATUS (_stdcall *ZW_QUERY_INFORMATION_PROCESS)(
512 HANDLE ProcessHandle,
513 PROCESSINFOCLASS ProcessInformationClass, //该参数也需要上面声明的数据结构
514 PVOID ProcessInformation,
515 ULONG ProcessInformationLength,
516 PULONG ReturnLength
517 ); //定义函数指针
518
519 void CDetectODDlg::OnZwQueryInformationProcess()
520 {
521     // TODO: Add your control notification handler code here
522     HANDLE hProcess;
523     HINSTANCE hModule;
524     DWORD dwResult;
525     ZW_QUERY_INFORMATION_PROCESS MyFunc;
526     hModule = GetModuleHandle("ntdll.dll");
527     MyFunc=
(ZW_QUERY_INFORMATION_PROCESS)GetProcAddress(hModule,"ZwQueryInformationProcess");
528     hProcess = GetCurrentProcess();
529     MyFunc(
530         hProcess,
531         ProcessDebugPort,
532         &dwResult,
533         4,
534         NULL);
535     if(dwResult!=0)
536     {
537         AfxMessageBox("发现OD");
538     }
539     else
540     {
541         AfxMessageBox("没有OD");
542     }
543 }
544 //*****
545 static DWORD lpOldHandler;
546 typedef LPTOP_LEVEL_EXCEPTION_FILTER (_stdcall *pSetUnhandledExceptionFilter)(
547     LPTOP_LEVEL_EXCEPTION_FILTER lpTopLevelExceptionFilter
548 );
549 pSetUnhandledExceptionFilter lpSetUnhandledExceptionFilter;
550
551 LONG WINAPI TopUnhandledExceptionFilter(
552     struct _EXCEPTION_POINTERS *ExceptionInfo
553 )
554 {
555     _asm pushad
556     AfxMessageBox("回调函数");
557     lpSetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER )lpOldHandler);
558     ExceptionInfo->ContextRecord->Eip=NewEip;//转移到安全位置
559     _asm popad
560     return EXCEPTION_CONTINUE_EXECUTION;
561 }
562

```



```

563 void CDetectODDllg::OnSetUnhandledExceptionFilter()
564 {
565     bool isDebugged=0;
566     // TODO: Add your control notification handler code here
567     lpSetUnhandledExceptionFilter =
568     (pSetUnhandledExceptionFilter)GetProcAddress(LoadLibrary(("kernel32.dll")),
569     "SetUnhandledExceptionFilter");
570     lpOldHandler=(DWORD)lpSetUnhandledExceptionFilter(TopUnhandledExceptionFilter);
571     _asm{ //获取这个安全地址
572         call me //方式一, 需要NewEip加上一个偏移值
573     me:
574         pop NewEip //方式一结束
575         mov NewEip,offset safe //方式二, 更简单
576         int 3 //触发异常
577     }
578     AfxMessageBox("检测到OD");
579     isDebugged=1;
580     _asm{
581     safe:
582     }
583     if(1==isDebugged){
584     }else{
585         AfxMessageBox("没有OD");
586     }
587 }
588 //*****
589 void CDetectODDllg::OnSeDebugPrivilege()
590 {
591     // TODO: Add your control notification handler code here
592     HANDLE hProcessSnap;
593     HANDLE hProcess;
594     PROCESSENTRY32 tp32; //结构体
595     CString str="csrss.exe";
596     hProcessSnap=::CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,NULL);
597     if(INVALID_HANDLE_VALUE!=hProcessSnap)
598     {
599         Process32First(hProcessSnap,&tp32);
600         do{
601             if(0==lstrcmpi(str,tp32.szExeFile))
602             {
603
604                 hProcess=OpenProcess(PROCESS_QUERY_INFORMATION,NULL,tp32.th32ProcessID);
605                 if(NULL!=hProcess)
606                 {
607                     AfxMessageBox("发现OD");
608                 }
609                 else
610                 {
611                     AfxMessageBox("没有OD");
612                 }
613                 CloseHandle(hProcess);
614             }while(Process32Next(hProcessSnap,&tp32));
615         }
616         CloseHandle(hProcessSnap);
617     }
618
619     //*****
620     #ifndef STATUS_INFO_LENGTH_MISMATCH
621     #define STATUS_INFO_LENGTH_MISMATCH ((UINT32)0xC000004L)
622     #endif
623
624     typedef enum _POOL_TYPE {
625         NonPagedPool,
626         PagedPool,
627         NonPagedPoolMustSucceed,
628         DontUseThisType,
629         NonPagedPoolCacheAligned,
630         PagedPoolCacheAligned,
631         NonPagedPoolCacheAlignedMustS
632     } POOL_TYPE;
633
634     typedef struct _UNICODE_STRING {

```

```

635     USHORT Length;
636     USHORT MaximumLength;
637     PWSTR Buffer;
638 } UNICODE_STRING;
639 typedef UNICODE_STRING *PUNICODE_STRING;
640 typedef const UNICODE_STRING *PCUNICODE_STRING;
641
642 typedef enum _OBJECT_INFORMATION_CLASS
643 {
644     ObjectBasicInformation,           // Result is OBJECT_BASIC_INFORMATION structure
645     ObjectNameInformation,           // Result is OBJECT_NAME_INFORMATION structure
646     ObjectTypeInformation,           // Result is OBJECT_TYPE_INFORMATION structure
647     ObjectAllTypesInformation,        // Result is OBJECT_ALL_INFORMATION structure
648     ObjectDataInformation            // Result is OBJECT_DATA_INFORMATION structure
649
650 } OBJECT_INFORMATION_CLASS, *POBJECT_INFORMATION_CLASS;
651
652 typedef struct _OBJECT_TYPE_INFORMATION {
653     UNICODE_STRING TypeName;
654     ULONG TotalNumberOfHandles;
655     ULONG TotalNumberOfObjects;
656     WCHAR Unused1[8];
657     ULONG HighWaterNumberOfHandles;
658     ULONG HighWaterNumberOfObjects;
659     WCHAR Unused2[8];
660     ACCESS_MASK InvalidAttributes;
661     GENERIC_MAPPING GenericMapping;
662     ACCESS_MASK ValidAttributes;
663     BOOLEAN SecurityRequired;
664     BOOLEAN MaintainHandleCount;
665     USHORT MaintainTypeList;
666     POOL_TYPE PoolType;
667     ULONG DefaultPagedPoolCharge;
668     ULONG DefaultNonPagedPoolCharge;
669 } OBJECT_TYPE_INFORMATION, *POBJECT_TYPE_INFORMATION;
670
671 typedef struct _OBJECT_ALL_INFORMATION {
672     ULONG NumberOfObjectsTypes;
673     OBJECT_TYPE_INFORMATION ObjectTypeInformation[1];
674 } OBJECT_ALL_INFORMATION, *POBJECT_ALL_INFORMATION;
675
676 typedef struct _OBJECT_ALL_TYPES_INFORMATION {
677     ULONG NumberOfTypes;
678     OBJECT_TYPE_INFORMATION TypeInformation[1];
679 } OBJECT_ALL_TYPES_INFORMATION, *POBJECT_ALL_TYPES_INFORMATION;
680
681 typedef UINT32 (__stdcall *ZwQueryObject_t) (
682     IN HANDLE ObjectHandle,
683     IN OBJECT_INFORMATION_CLASS ObjectInformationClass,
684     OUT PVOID ObjectInformation,
685     IN ULONG Length,
686     OUT PULONG ResultLength );
687
688 void CDetectODDlg::OnNTQueryObject()
689 {
690     // TODO: Add your control notification handler code here
691     // 调试器必须正在调试才能检测到, 仅打开OD是检测不到的
692     HMODULE hNtDLL;
693     DWORD dwSize;
694     UINT i;
695     UCHAR KeyType=0;
696     OBJECT_ALL_TYPES_INFORMATION *Types;
697     OBJECT_TYPE_INFORMATION *t;
698     ZwQueryObject_t ZwQueryObject;
699
700     hNtDLL = GetModuleHandle("ntdll.dll");
701     if(hNtDLL){
702         ZwQueryObject = (ZwQueryObject_t)GetProcAddress(hNtDLL, "ZwQueryObject");
703         UINT32 iResult = ZwQueryObject(NULL, ObjectAllTypesInformation, NULL, NULL,
704 &dwSize);
705         if(iResult==STATUS_INFO_LENGTH_MISMATCH)
706         {
707             Types =
708 (OBJECT_ALL_TYPES_INFORMATION*)VirtualAlloc(NULL,dwSize,MEM_COMMIT,PAGE_READWRITE);

```

```

707         if (Types == NULL) return;
708         if (iResult=ZwQueryObject(NULL,ObjectAllTypesInformation, Types, dwSize,
&dwSize)) return;
709         for (t=Types->TypeInfo, i=0; i<Types->NumberOfTypes; i++)
710         {
711             if ( !_wcsicmp(t->TypeName.Buffer, L"DebugObject")) //比较两个是否相等,
这个L很特殊, 本地的意思
712             {
713                 if(t->TotalNumberOfHandles > 0 || t->TotalNumberOfObjects > 0)
714                 {
715                     AfxMessageBox("发现OD");
716                     VirtualFree (Types, 0, MEM_RELEASE);
717                     return;
718                 }
719                 break; // Found Anyways
720             }
721             t=(OBJECT_TYPE_INFORMATION *)((char *)t->TypeName.Buffer+((t-
>TypeName.MaximumLength+3)&~3));
722         }
723     }
724     AfxMessageBox("没有OD!");
725     VirtualFree (Types, 0, MEM_RELEASE);
726 }
727 }
728 /*****
729 BOOL DetectBreakpoints()
730 {
731     BOOL bFoundOD;
732     bFoundOD=FALSE;
733     __asm
734     {
735         jmp     CodeEnd
736     CodeStart: mov     eax, ecx    ;被保护的程序段
737         nop
738         push   eax
739         push   ecx
740         pop    ecx
741         pop    eax
742     CodeEnd:
743         cld                    ;检测代码开始
744         mov     edi, offset CodeStart
745         mov     edx, offset CodeStart
746         mov     ecx, offset CodeEnd
747         sub     ecx, edx
748
749         mov     al, 0CCH
750         repne   scasb
751         jnz     ODNotFound
752         mov     bFoundOD, 1
753     ODNotFound:
754     }
755     return bFoundOD;
756 }
757 void CDetectODDlg::OnDetectBreakpoints()
758 {
759     // TODO: Add your control notification handler code here
760     if(DetectBreakpoints())
761     {
762         AfxMessageBox("发现OD");
763     }
764     else
765     {
766         AfxMessageBox("没有OD");
767     }
768 }
769 /*****
770 BOOL DetectFuncBreakpoints()
771 {
772     BOOL bFoundOD;
773     bFoundOD=FALSE;
774     DWORD dwAddr;
775     dwAddr=(DWORD)::GetProcAddress(LoadLibrary("user32.dll"), "MessageBoxA");
776     __asm
777     {

```

```

778         cld             ;检测代码开始
779         mov     edi,dwAddr
780         mov     ecx,100    ;100bytes
781         mov     al,0CCH
782         repne   scasb
783         jnz     ODNotFound
784         mov     bFoundOD,1
785     ODNotFound:
786     }
787     return bFoundOD;
788 }
789 void CDetectODDlg::OnDetectFuncBreakpoints()
790 {
791     // TODO: Add your control notification handler code here
792     if(DetectFuncBreakpoints())
793     {
794         AfxMessageBox("发现OD");
795     }
796     else
797     {
798         AfxMessageBox("没有OD");
799     }
800 }
801
802 void CDetectODDlg::OnBlockInput()
803 {
804     // #include "Winable.h"
805     // TODO: Add your control notification handler code here
806     DWORD dwNoUse;
807     DWORD dwNoUse2;
808     ::BlockInput(TRUE);
809     dwNoUse=2;
810     dwNoUse2=3;
811     dwNoUse=dwNoUse2;
812     ::BlockInput(FALSE);
813 }
814 /*****
815 BOOL CheckSum()
816 {
817     BOOL bFoundOD;
818     bFoundOD=FALSE;
819     DWORD CHECK_SUM=5555; //正确校验值
820     DWORD dwAddr;
821     dwAddr=(DWORD)Checksum;
822     __asm
823     {
824         ;检测代码开始
825         mov     esi,dwAddr
826         mov     ecx,100
827         xor     eax,eax
828     checksum_loop:
829         movzx   ebx,byte ptr [esi]
830         add     eax,ebx
831         rol     eax,1
832         inc     esi
833         loop    checksum_loop
834
835         cmp     eax,CHECK_SUM
836         jz      ODNotFound
837         mov     bFoundOD,1
838     ODNotFound:
839     }
840     return bFoundOD;
841 }
842 void CDetectODDlg::OnChecksum()
843 {
844     // TODO: Add your control notification handler code here
845     if(CheckSum())
846     {
847         AfxMessageBox("发现OD");
848     }
849     else
850     {
851         AfxMessageBox("没有OD");
852     }
853 }

```

```

852     }
853     /*****
854
855 void CDetectODDlg::OnEnableWindow()
856 {
857     // TODO: Add your control notification handler code here
858     CWnd *wnd;
859     wnd=GetForegroundWindow();
860     wnd->EnableWindow(FALSE);
861     DWORD dwNoUse;
862     DWORD dwNoUse2;
863     dwNoUse=2;
864     dwNoUse2=3;
865     dwNoUse=dwNoUse2;
866     wnd->EnableWindow(TRUE);
867 }
868     *****/
869     typedef enum _THREADINFOCLASS {
870     ThreadBasicInformation, // 0 Y N
871     ThreadTimes, // 1 Y N
872     ThreadPriority, // 2 N Y
873     ThreadBasePriority, // 3 N Y
874     ThreadAffinityMask, // 4 N Y
875     ThreadImpersonationToken, // 5 N Y
876     ThreadDescriptorTableEntry, // 6 Y N
877     ThreadEnableAlignmentFaultFixup, // 7 N Y
878     ThreadEventPair, // 8 N Y
879     ThreadQuerySetWin32StartAddress, // 9 Y Y
880     ThreadZeroTlsCell, // 10 N Y
881     ThreadPerformanceCount, // 11 Y N
882     ThreadAmILastThread, // 12 Y N
883     ThreadIdealProcessor, // 13 N Y
884     ThreadPriorityBoost, // 14 Y Y
885     ThreadSetTlsArrayAddress, // 15 N Y
886     ThreadIsIoPending, // 16 Y N
887     ThreadHideFromDebugger // 17 N Y
888     } THREAD_INFO_CLASS;
889
890     typedef NTSTATUS (NTAPI *ZwSetInformationThread)(
891     IN HANDLE ThreadHandle,
892     IN THREAD_INFO_CLASS ThreadInformationClass,
893     IN PVOID ThreadInformation,
894     IN ULONG ThreadInformationLength
895     );
896
897 void CDetectODDlg::OnZwSetInformationThread()
898 {
899     // TODO: Add your control notification handler code here
900     CString str="利用我定位";
901     HANDLE hwnd;
902     HMODULE hModule;
903     hwnd=GetCurrentThread();
904     hModule=LoadLibrary("ntdll.dll");
905     ZwSetInformationThread myFunc;
906     myFunc=(ZwSetInformationThread)GetProcAddress(hModule,"ZwSetInformationThread");
907     myFunc(hwnd,ThreadHideFromDebugger,NULL,NULL);
908 }
909     *****/
910 void CDetectODDlg::OnOutputDebugString()
911 {
912     // TODO: Add your control notification handler code here
913     ::OutputDebugString("%s%s");
914 }
915     *****/
916 void CDetectODDlg::OnGetEntryPoint()
917 {
918     // TODO: Add your control notification handler code here
919     IMAGE_DOS_HEADER *dos_head=(IMAGE_DOS_HEADER *)GetModuleHandle(NULL);
920     PIMAGE_NT_HEADERS32 nt_head=(PIMAGE_NT_HEADERS32)((DWORD)dos_head+(DWORD)dos_head-
>e_lfanew);
921     DWORD EP=(nt_head->OptionalHeader.AddressOfEntryPoint);
922     CString str;
923     str.Format("%x",EP);
924     AfxMessageBox(str);

```

```

925     BYTE*OEP=(BYTE*)(nt_head->OptionalHeader.AddressOfEntryPoint+(DWORD)dos_head);
926     for(unsigned long index=0;index<20;index++){
927         if(OEP[index]==0xcc){
928             ExitProcess(0);
929         }
930     }
931 }
932
933 }
934 /*****
935 void terminateFunc()
936 {
937     AfxMessageBox("set_terminate指定的函数\n");
938     exit(0);
939 }
940 void CDetectODDlg::OnButton1()
941 {
942     // TODO: Add your control notification handler code here
943
944     set_terminate(terminateFunc);
945     try{
946         div(10,0);
947     }catch(int){
948         AfxMessageBox("仅捕获整型异常");
949     }catch(...){
950         terminate(); //所有其它异常
951     }
952     AfxMessageBox("啊哈");
953 }
954 //*****
955
956 void CDetectODDlg::OnTrapFlag()
957 {
958     try{
959         _asm{
960             pushfd                //触发单步异常
961             or     dword ptr [esp],100h    ;TF=1
962             popfd
963         }
964         AfxMessageBox("检测到OD");
965     }catch(...){
966         AfxMessageBox("没有OD");
967     }
968 }
969 //*****
970 static bool isDebugged=1;
971 LONG WINAPI TopUnhandledExceptionFilter2(
972     struct _EXCEPTION_POINTERS *ExceptionInfo
973 )
974 {
975     _asm pushad
976     AfxMessageBox("回调函数");
977     lpSetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER )lpOldHandler);
978     ExceptionInfo->ContextRecord->Eip=NewEip;
979     isDebugged=0;
980     _asm popad
981     return EXCEPTION_CONTINUE_EXECUTION;
982 }
983
984 void CDetectODDlg::OnGuardPages()
985 {
986     // TODO: Add your control notification handler code here
987
988     ULONG dwOldType;
989     DWORD dwPageSize;
990     LPVOID lpvBase;          // 获取内存的基地址
991     SYSTEM_INFO sSysInfo;    // 系统信息
992     GetSystemInfo(&sSysInfo); // 获取系统信息
993     dwPageSize=sSysInfo.dwPageSize; //系统内存页大小
994
995     lpSetUnhandledExceptionFilter =
996     (pSetUnhandledExceptionFilter)GetProcAddress(LoadLibrary(("kernel32.dll")),
997     "SetUnhandledExceptionFilter");
998     lpOldHandler=(DWORD)lpSetUnhandledExceptionFilter(TopUnhandledExceptionFilter2);

```

```

998
999 // 分配内存
1000 lpvBase = VirtualAlloc(NULL,dwPageSize,MEM_COMMIT,PAGE_READWRITE);
1001 if (lpvBase==NULL) AfxMessageBox("内存分配失败");
1002 _asm{
1003     mov     NewEip,offset safe //方式二, 更简单
1004     mov     eax,lpvBase
1005     push    eax
1006     mov     byte ptr [eax],0C3H //写一个 RETN 到保留内存, 以便下面的调用
1007 }
1008 if(0==:VirtualProtect(lpvBase,dwPageSize,PAGE_EXECUTE_READ |
PAGE_GUARD,&dwOldType)){
1009     AfxMessageBox("执行失败");
1010 }
1011 _asm{
1012     pop     ecx
1013     call    ecx //调用时压栈
1014 safe:
1015     pop     ecx //堆栈平衡, 弹出调用时的压栈
1016 }
1017 if(1==isDebugged){
1018     AfxMessageBox("发现OD");
1019 }else{
1020     AfxMessageBox("没有OD");
1021 }
1022 VirtualFree(lpvBase,dwPageSize,MEM_DECOMMIT);
1023 }
1024 //*****
1025 static bool isDebuggedHBP=0;
1026 LONG WINAPI TopUnhandledExceptionFilterHBP(
1027     struct _EXCEPTION_POINTERS *ExceptionInfo
1028 )
1029 {
1030     _asm pushad
1031     AfxMessageBox("回调函数被调用");
1032     ExceptionInfo->ContextRecord->Eip=NewEip;
1033     if(0!=ExceptionInfo->ContextRecord->Dr0||0!=ExceptionInfo->ContextRecord->Dr1||
1034         0!=ExceptionInfo->ContextRecord->Dr2||0!=ExceptionInfo->ContextRecord->Dr3)
1035         isDebuggedHBP=1; //检测有无硬件断点
1036     ExceptionInfo->ContextRecord->Dr0=0; //禁用硬件断点, 置0
1037     ExceptionInfo->ContextRecord->Dr1=0;
1038     ExceptionInfo->ContextRecord->Dr2=0;
1039     ExceptionInfo->ContextRecord->Dr3=0;
1040     ExceptionInfo->ContextRecord->Dr6=0;
1041     ExceptionInfo->ContextRecord->Dr7=0;
1042     ExceptionInfo->ContextRecord->Eip=NewEip; //转移到安全位置
1043     _asm popad
1044     return EXCEPTION_CONTINUE_EXECUTION;
1045 }
1046
1047 void CDetectODDlg::OnHardwarebreakpoint()
1048 {
1049     // TODO: Add your control notification handler code here
1050
1051     lpSetUnhandledExceptionFilter =
1052     (pSetUnhandledExceptionFilter)GetProcAddress(LoadLibrary(("kernel32.dll")),
1053     "SetUnhandledExceptionFilter");
1054     lpOldHandler=(DWORD)lpSetUnhandledExceptionFilter(TopUnhandledExceptionFilterHBP);
1055     _asm{
1056         mov     NewEip,offset safe //方式二, 更简单
1057         int     3
1058         mov     isDebuggedHBP,1 //调试时可能也不会触发异常去检测硬件断点
1059 safe:
1060     }
1061     if(1==isDebuggedHBP){
1062         AfxMessageBox("发现OD");
1063     }else{
1064         AfxMessageBox("没有OD");
1065     }
1066 }
1067 //*****
1068
1069 void CDetectODDlg::OnCancel()

```

```

1070 {
1071     // TODO: Add extra cleanup here
1072     CDialog::OnCancel();
1073 }
1074
1075 void CAboutDlg::OnMypage()
1076 {
1077     // TODO: Add your control notification handler code here
1078     ::ShellExecute(NULL, "open", "http://ucooper.com", NULL, NULL, SW_SHOWNORMAL);
1079 }
1080
1081 void CDetectODDlg::OnAbout()
1082 {
1083     // TODO: Add your control notification handler code here
1084     CAboutDlg dlg;
1085     dlg.DoModal();
1086 }
1087
1088 void CAboutDlg::OnMouseMove(UINT nFlags, CPoint point)
1089 {
1090     // TODO: Add your message handler code here and/or call default
1091     CRect rect(60, 20, 100, 100);
1092     if(rect.PtInRect(point)){
1093         SetClassLong(m_hWnd, GCL_HCURSOR, (LONG)(LoadCursor(NULL, IDC_HELP)));
1094     }else{
1095         SetClassLong(m_hWnd, GCL_HCURSOR, (LONG)(LoadCursor(AfxGetApp()-
1096 >m_hInstance, IDC_ARROW)));
1097     }
1098     CDialog::OnMouseMove(nFlags, point);
1099 }
1100
1101 BOOL CAboutDlg::OnInitDialog()
1102 {
1103     CDialog::OnInitDialog();
1104
1105     // TODO: Add extra initialization here
1106     SetClassLong(m_hWnd, GCL_HICON, (LONG)(LoadIcon(AfxGetApp()-
1107 >m_hInstance, MAKEINTRESOURCE(IDI_DOG))));
1108     return TRUE; // return TRUE unless you set the focus to a control
1109     // EXCEPTION: OCX Property Pages should return FALSE
1110 }
1111
1112 void CDetectODDlg::OnOK()
1113 {
1114     // TODO: Add extra validation here
1115     CDialog::OnOK();
1116 }
1117
1118 void CAboutDlg::OnComeon()
1119 {
1120     // TODO: Add your control notification handler code here
1121     ::ShellExecute(NULL, "open", "http://ucooper.com", NULL, NULL, SW_SHOWNORMAL);
1122 }
1123
1124 void CAboutDlg::OnMyicon()
1125 {
1126     // TODO: Add your control notification handler code here
1127     ::ShellExecute(NULL, "open", "http://ucooper.com", NULL, NULL, SW_SHOWNORMAL);
1128 }
1129
1130 void CDetectODDlg::OnMypage2()
1131 {
1132     // TODO: Add your control notification handler code here
1133     ::ShellExecute(NULL, "open", "http://ucooper.com", NULL, NULL, SW_SHOWNORMAL);
1134 }

```













