3. Обработка необрабатываемых исключений

Отладчики, работающие через MS Debugging API (OllyDbg, IDA-Pro, MS VC),

вынуждены мириться с тем, что отладочные процессы «страдают» хроничес-

кими особенностями поведения. Они «ломают» логику программы, и это

с огромной выгодой используют защитные механизмы. В частности, АРІ-фун-

кция SetUnhandledExceptionFilter() под отладчиком вообще не вызывается —

Энциклопедия антиотладочных приемов

вовсе не баг отладчика, а документированная фича системы!

FUNDAMENTALS Рассматривая обработку структурных исключений в предыдущем выпуске, мы мельком упомянули, что всякий процесс от рождения получает первичный обработчик структурных исключений, назначаемый операционной системой по умолчанию. Если программист забыл (или не захотел) назначать свои собственные обработчики, то все исключения, возникающие при выполнении программы, попадают в пасть первичного обработчика. Он расположен в NTDLL.DLL и, в зависимости от настроек оси, либо вызывает «Доктора Ватсона»,

либо выводит знаменитый диалог о критической ошибке с вариантами: ОК — завершить приложение в аварийном режиме и Cancel — вызывать Just-in-Time-отладчик (в роли которого может выступать и «Ольга»). OllyDbg 1.х имеет коварный баг! «Ольга» версии 1.10 имеет неприятный баг — если непосредственно за INT 03h следует команда PUSHFD, заталкивающая флаги в стек, отладчик едет крышей и теряет управление

над отлаживаемой программой, даже если мы нажимаем F7/F8 (Step Into / Step Over).

Для демонстрации бага достаточно воткнуть в листинг пару команд PUSHFD/POPFD. А вот те

же самые команды, отделенные от INT 03h одной или несколькими инструкциями NOP (или любыми другими), работают вполне нормально. В «Ольге» 2.х ошибка уже исправлена, однако, учитывая, что 2.х все еще находится в стадии разработки, основным инструментом хакеров остается «Ольга» 1.10 с багом на борту. То же самое происходит, если программист устанавливает один или несколько обработчиков структурных исключений, но никто из них не в состоянии справиться с ситуацией —

вот они и передают исключение друг другу, пока оно не докатится до системного обработчика. Системный обработчик легко подменить своим (было бы желание). Достаточно вместо ссылки на предыдущий EXCEPTION_REGISTRATION затолкать в поле prev значение -1. Это будет свидетельствовать, что данный обработчик — последний в цепочке.

Как вариант, можно воспользоваться API-функцией SetUnhandledExceptionFilter(), перек-

рывающей обработчик исключений верхнего уровня (top-level exception handler). Да, именно «верхнего», поскольку Windows создавалась в Америке, расположенной на противоположной стороне Земли, где люди ходят вверх ногами. Первичный системный обработчик, с их

точки зрения, находится на вершине пирамиды структурных исключений, в то время как русские программисты склонны рассматривать его как «основание». Но это все лирика, а дело-то в том, что... 🐕 MSDN Library Visual Studio 6.0 $\underline{\Phi}$ айл <u>П</u>равка <u>В</u>ид Переход <u>С</u>правка становить Обновить Ломой Печат SetUnhandledExceptionFilter Содержание <u>Указатель</u> Поиск Избранное The SetUnhandledExceptionFilter function lets an application supersede the top-level exception handler that Win32 places at the top of each thread and process. EXCEPTION POINTERS

After calling this function, if an exception occurs in a process that is not being debugged, and

the exception makes it to the Win32 unhandled exception filter, that filter will call the

exception filter function specified by the IpTopLevelExceptionFilter parameter.

EXCEPTION_RECORD exceptionClear exceptionDescribe ExceptionInInitializerError exceptionOccurred LPTOP LEVEL EXCEPTION FILTER SetUnhandledExceptionFilter(LPTOP LEVEL EXCEPTION FILTER lpTopLevelExceptionFilter Exceptions additional reading archive bad_cast bad_typeid C++ // exception filter function SetUnhandledExceptionFilter() на MSDN Функция SetUnhandledExceptionFilter(), перекрывая системный обработчик, в неволе работать отказывается, то есть получает управление, только когда процесс не находится под отладчиком. В противном случае исключение передается непосредственно самому отладчику. Это — задумка проектировщиков, кстати сказать довольно оригинальная и полезная. Если отладчика нет — установленный программистом обработчик берет управление на себя и завершает работу программы максимально корректным образом. Если же процесс находится под отладкой, операционная система передает бразды правления отладчику, позволяя разобраться с ситуацией, поскольку после завершения программы разбираться будет

ЭКСПЕРИМЕНТ — PRO ET CONTRA SETUNHANDLEDEXCEPTIONFILTER Напишем простейшую тестовую программу, позволяющую исследовать реакцию отладчиков на фильтр, установленный функцией SetUnhandledExceptionFilter(). На crackme она никак не тянет (слишком прозрачна и элементарна), но crackme мы написать завсегда успеем! Сейчас главное — врубиться в тему и выяснить, насколько надежен этот трюк, можно ли его обойти и если да, то как. Один из примеров реализации тестового стенда приведен ниже.

Исходный текст программы SetUnhandledExceptionFilter.c, демонстрирующей технику использования функции для борьбы с отладчиками

// We expect debugger by default

А теперь, внимание, вопрос! Что произойдет, если в обработчик, установленный SetUnhandledExceptionFilter(), воткнуть не код аварийного завершения приложения,

а кусок функциональности, например расшифровщик какой или просто пару строк на С, меняющих значение флага under_debuuger? Правильно — мы получим великолепный спо-

LONG souriz(struct _EXCEPTION_POINTERS *ExceptionInfo)

char dbgnoo[]= "debugger is not detected"; char dbgyes[]= "debugger is detected :-)";

не с чем и некому.

#include <windows.h>

char *p = dbgyes;

соб детекта отладчиков прикладного уровня!

```
// if we're here, process is _not_ under debugger
    p = dbgnoo;
    // Skip INT 03 (CCh) command
    ExceptionInfo->ContextRecord->Eip++;
   // We want the program to continue execution
    return EXCEPTION_CONTINUE_EXECUTION;
nezumi()
    // Supersede the default top-level exception handler by souriz() proc
    SetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER)&souriz);
    \_\_\mathsf{asm}\{
        int 03
                            ; // generate an exception
    // Terminate the program, showing the result
    ExitProcess(MessageBox(∅,p,p,∅));
Собираем программу компилятором MS Visual C++ со следующими ключами (см. ниже)
и получаем файл размером всего в 832 байта (и это не предел — если выкинуть заглушку MS-
DOS, программа похудеет еще на полсотни байтов).
Сборка программы SetUnhandledExceptionFilter.c в командной строке среды
Microsoft Visual C++
SET NIK=SetUnhandledExceptionFilter
cl.exe /c /Ox %NIK%.c
link.exe %NIK%.obj /FIXED /ENTRY:nezumi \
    /SUBSYSTEM:CONSOLE /ALIGN:16
    /MERGE:.data=.text /MERGE:.rdata=.text KERNEL32.LIB USER32.LIB
C:\WINNT\system32\CMD.EXE - SetUnhandledExceptionFilter.exe
```

Запуск тестовой программы под «чистой» осью проходит нормально Запустив файл на выполнение, мы получим сообщение, что отладчик не обнаружен. А как

насчет работы под отладчиком? Загружаем SetUnhandledExceptionFilter.exe в «Ольгу»

и давим F9 (Run). Ara! «Ольга» стопорится на INT 03h, что выглядит подозрительно (нормальные программы INT 03h не вызывают), хотя и не смертельно. Жмем F9 еще раз для продол-

:\ARTICLE\hacker\anti-debug-#03>SetUnhandledExceptionFilter.exe

debugger is not detected

debugger is not detected

Microsoft Windows 2000 [Версия 5.00.2195]

(С) Корпорация Майкрософт, 1985-2000

0040028A

0040028B

ции отладчика

90

90

информацию об исключении.

популярный мод «Ольги» и, естественно, бесплатный.

8DA424 00000 LEA 8D6424 00 LEA 8D5424 08 LEA

8DA424 30FDF LEA

E8 F623FFFF EB 1C

8BCC 6A 01

C File <u>View D</u>3bug <u>P</u>lugins Options <u>W</u>indow <u>H</u>elp

CD 2E

💥 SND - SetUnhandledExceptionFilter.exe - [DSM - main thr34d, modul3 ntdll]

INT

PUSH

PUSH

PUSH PUSH

CALL

MOV

PUSH

PUSH PUSH

MOU

равленности).

7C82ED54

7C82ED55

7C82ED5C

7C82ED64

7C82ED66

7C82ED68

7C82ED6A

7C82ED71

7C82EDAA 7C82EDAC

7C82EDAD 7C82EDB2

7C82EDB4

7C82EDB6

7C82EDB8 2C82EDB9

0040109C

004010A2

004010A4

004010AA 004010B0

004010B3

004010B5

004010BB

004010BE

004010C3

004010C5

004010CA

004010CB

004010CD

004010CF

004010D1

004010D6

00401050

00401051

00401057

00401060

00401067

00401069

0040106A

0040106B

0040106C

00401071

00401076

0040107B

00401081

0040106D 8BEC

0040106F 6AFF

00401089 83EC10

ASTERIX_HIDE_DEBUGGER.

while(1)

68A0404000

68341B4000

64A1000000000

00401068 5E

00401058 6A00

0040105A FF1594404000

00401061 FF1508404000

8B0D68504000

ваемых исключений

```
жения выполнения и ловим месседж: «дебаггер из детектед».
  OllyDbg - SetUnhandledExceptionFilter.exe - [CPU - main thread, module SetUnhan]
 C File <u>V</u>iew <u>D</u>ebug <u>P</u>lugins Options <u>W</u>indow <u>H</u>elp
 MOV EAX, DWORD PTR DS: [40022C]
 0040026C
          . A1 2C024000
 00400271
            6A 00
                         PUSH 0
                                                               -Style = MB_OK|MB_AF
                         PUSH EAX
                                                                Title => "debugger
 00400273
            50
                                                               Text => "debugger
 00400274
          . 50
                         PUSH EAX
 00400275
          . 6A 00
                                                                hOwner = NULL
                         PUSH 0
 00400277
          . FF15 EC014000 CALL DWORD PTR DS:[<&USER32.MessageBoxA]
                                                               LMessageBoxA
 0040027D
          . 50
                         PUSH EAX
                                                               rExitCode
          .. FF15 E4014000 CALL DWORD PTR DS:[<&KERNEL32.ExitProced LexitProcess
 0040027E
  10400284
 00400285
            90
                                                            debugger is detected :-)
 00400286
            90
 00400287
            90
                                                             debugger is detected :-)
 00400288
            90
 00400289
            90
                                                                     ОК
```

сюда управление каждый раз, когда отладчик ловит исключение, не обрабатываемое SEHфильтрами, установленными программистом (если они вообще есть). Итак, необходимо: отловить вызов SetUnhandledExceptionFilter(), запомнив (записав на бумажку) указатель на фильтр, а при возникновении исключения — раскрутить цепочку SEH-фильтров. Если последний фильтр в цепочке направлен в отладчик, мы должны вручную переместить EIP на фильтр, установленный SetUnhandledExceptionFilter(), передав

ей в качестве аргумента указатель на структуру _EXCEPTION_POINTERS, содержащую

Чтобы не париться, эту работу можно автоматизировать — написать свой собственный скрипт/плагин или воспользоваться уже готовым. Например, «Hide Debugger» plug-in by Asterix, который можно бесплатно скачать с OpenRCE (или другого сайта хакерской нап-

Кстати сказать, Hide_Debugger изначально входит в состав YDbg, представляющий собой

F DEMTWHC7KBR

Save

Close

X

DWORD PTR SS:[ESP] DWORD PTR SS:[ESP]

Hide Debugger [Options]

Detach

DUORD PIR SS:[FRP+8]

EDX, DWORD PTR SS:[ESP+8]

Запуск тестовой программы под «чистой» «Ольгой» (без специальных плагинов) приводит к детек-

Начинаем соображать — как можно это отловить, обломав защите рога. Что ж, достаточно поставить точку останова на API-функцию SetUnhandledExceptionFilter(), запомнить

передаваемый ей указатель на процедуру-обработчик (в данном случае это souriz) и кидать

E8 AD010000 7C82ED72 CALL 7C82E Protect against: 7C82ED7A EAX, IsDebuggerPresent 7C82ED7D 838424 C4000 ADD DWORI DWORD DWORD 7C82ED85 FindWindow/EnumWindows C70424 07000 MOV 7C82ED88 ■ TerminateProcess 64:A1 300000 MOV F640 02 FF TES EAX, BYTE 7C82ED8F 7C82ED95 TEST ✓ Unhandled exception tricks 75 19 54 SHORT ESP JNZ PUSH 7C82ED99 OutputDebugString exploit 7C82ED9B FF75 08 E8 9852FEFF 84C0 PUSH CALL **DWORI** 7C82ED9C Heap-checking tricks 7C814 AL. A SHORT 7C82ED9F 7C82EDA4 TEST 7C82EDA6 7C82EDA8 MOU

ECX.

ECX

7082:

SHORT

ECX.

1

ECX

«Hide Debugger» plug-in позволяет бороться с защитами, основанными на обработке необрабаты-

Заходим в меню Plug-ins, находим там Hide debugger. В опциях взводим галочку Unhanded

EBP.

ESP

exception tricks, затем нажимаем Ctrl-O. В открывавшемся диалоговом окне выбираем вкладку Exceptions и взводим галочку INT 03 breaks для передачи ломаемой программе исключений, генерируемых INT 03h. В конфигурации по умолчанию «Ольга», как и большинство других отладчиков прикладного уровня, молчаливо поглощает INT 03h — и потому никакой обработчик исключений вообще не вызывается. На самом деле INT 03h не имеет отношения к SetUnhandledExceptionFilter, и, если бы мы, например, генерировали исключения через обращение к нулевому указателю, последнего действия не потребовалось бы. Достаточно было бы просто взвести INT 03 breaks (подробнее о передаче исключений программе мы поговорим в следующем выпуске, а пока вернемся к нашим баранам). Перезапускаем отладчик, чтобы изменения вступили в силу, и «пытаем» нашу тестовую программу еще раз. На экране победно отображается debugger is not detected. Открываем пиво на радостях! Мы нашли способ, как обломать этот антиотладочный прием (между прочим, весьма популярный). 🔭 SND - SetUnhandledExceptionFilter.exe - [DSM - main thr34d, modul3 SetUnhan] C File View D3bug Plugins Options Window Help. LEMTWHC7 KB | Running PUSH MOV EBP, ESP 0040106D 0040106F 6A FF PUSH -1A040400 PUSH 341B400 PUSH 00401071 4040A0 00401076 401B34 SE handler EAX, DWORD PTR FS:[0] 0040107B 64:A1 0000 MOV 00401081 PUSH 64:8925 00 MOV 00401082 DWORD PTR FS:[0], ESP 83EC 10 00401089 SUB ESP, 10 53 0040108C PUSH EBX 0040108D PUSH ES I PUSH 0040108E 0040108F 8965 E8 **ELOCAL.61, ESP** MOU NEAR DWORD PTR DS:[404014 kernel32 00401092 FF15 14404 CALL 00401098 DL, AH DWORD PTR DS:[4052E4], ED 0040109A MOU

ECX, EAX
ECX, OFF
DWORD PTR DS:[4052E0], EC
ECX, 8

ECX, EDX DWORD PTR DS:[4052DC], EQ

debugger is not detected

debugger is not detected

X

DWORD PTR DS: [4052D8]

EAX, 10

004019FD

EAX, EAX

00401170

SHORT 004010

ECX

«Hide Debugger» plug-in позволил спрятать отладчик от защиты

Что же касается отладчиков типа Soft-Ice и Syser, то они никак не воздействуют на поведение функции SetUnhandledExceptionFilter(), поскольку отладочного процесса не порож-

дают и ведут себя, будто их здесь вообще нет. Но если, терзая нашу программу, сказать отладчику I3HERE ON и заставить его всплывать на программных точках останова (у многих хакеров эта команда пробита в строке инициализации), то отладчик «зажует» исключение, генерируемое инструкцией INT 03h. Поэтому до ломаемой программы оно вообще не дойдет, а значит, установленный фильтр не будет вызван. На экране снова появится улыбающаяся

E4524 MOV

FF000 AND

890D E0524 MOU C1E1 08 SHL

890D DC524MOV

A3 D852400 MOV

E8 3309000 CALL

9A00000 CALL

81E1

03CA

59 85C0

75 08

1C

PUSH EAX

PUSH ECX

PUSH +00

PUSH EAX

POP EDI

PUSH EBP

PUSH -01

PUSH EAX

00401082 64892500000000 MOV FS:[00000000],ESP

MOV EBP, ESP

PVSH 004040A0

PUSH 00401B34

SUB ESP, +10

MOV EAX, FS: [00000000]

POP

POP

POP

RET

MOV ECX, [00405068]

CALL MessageBoxA

CALL ExitProcess

ESI

EBX

EBP

C1E8 10

MOV

ADD

SHR

PUSH

POP

JNZ

POP

PUSH

TEST

рожа, подтверждающая детект отладчика. 00000010 00000020 00000030 00000040 00000050 00000060 00000070 00401048 00401049 IN UINT uType PUSH MOV EAX, [00405068] A168504000 0040104B

IN LPCSTR lpCaption

int MessageBoxA(IN HWND hWnd,IN LPCSTR lpText,IN LPCSTR lpCap

DECLSPEC_MORETURN VOID ExitProcess(IN VINT uExitCode)

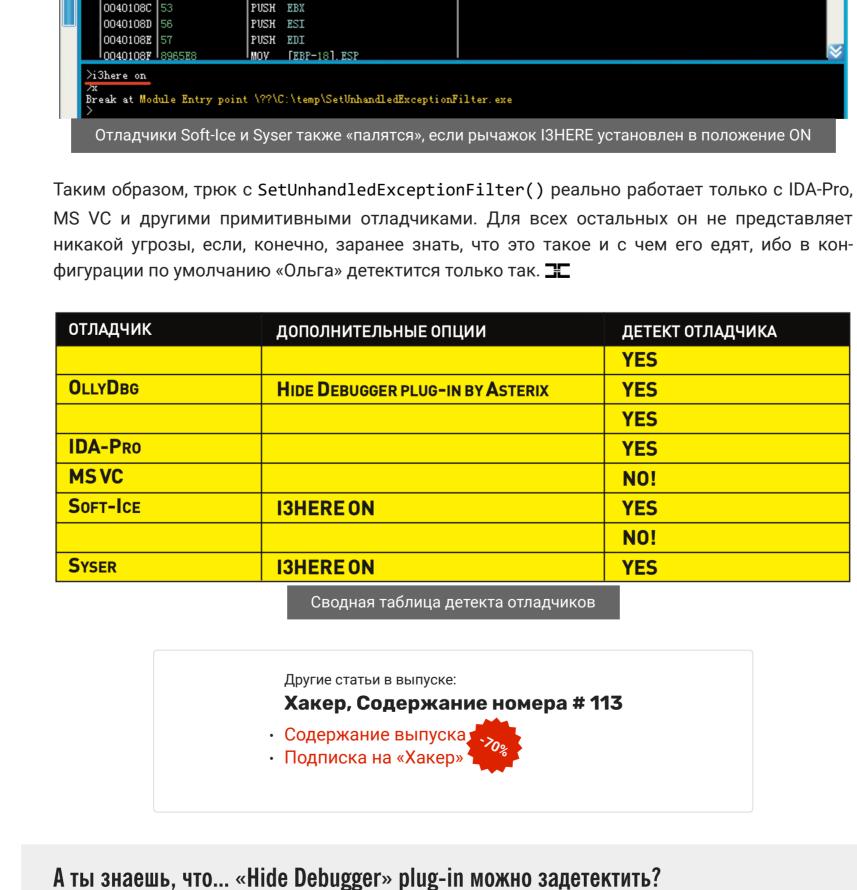
IN LPCSTR lpText

IN UINT uExitCode

debugger is detected:-) 🔣

debugger is detected :-)

IN HWND hWnd



метод детекции перестанет работать. Однако нетрудно реализовать универсальный детектор, основанный на самом факте подмены адреса NtQueryInformationProcess(). Достаточно распарсить таблицу импорта KERNEL32.DLL, и если адрес NtQueryInformationProcess() выходит за пределы модуля NTDLL.DLL (откуда эта функция, собственного, и импортируется), то, значит, мы имеем дело с «нестерильной» системой (строго говоря, это может быть

Плагин Hide Debugger от Asterix (как и большинство других плагинов подобного типа) дос-

Debugger изменяет адрес функции NtQueryInformationProcess() в таблице импорта биб-

лиотеки KERNEL32.DLL. Он записывает сюда команду перехода на свой собственный обработчик, который расположен в одном из блоков динамической памяти отлаживаемого про-

цесса и легко находится сканированием кучи и прямым поиском плагина по сигнатурам. В этом случае функция check_for_asterix_hide_debugger_plugin() возвратит значение

Естественно, после того как Asterix перепишет свой плагин, сигнатуры уйдут лесом и данный

таточно легко задетектить, и тогда защита вновь обломает отладчик. При работе Hide

не только HIDE DEBUGGER, но и какой-нибудь rootkit, однако разработчиков защит подобные мелочи не волнуют). Код детекции Hide Debugger // Very dirty anti-anti-debug trick check_for_asterix_hide_debugger_plugin() int a; int ret;int p=0; BYTE *x; MEMORY_BASIC_INFORMATION meminfo; // Asterix's Hide Debugger plugin changes addr of NtQueryInformationProcess

> // in the KERNEL32.DLL IAT to his own handler placed in the heap rwe block // so, to detect the plug-in, we have to find the _certain_ heap-block and // check signature out. It'll work until asterix doesn't rewrite the code.

1	
	<pre>if (!VirtualQuery((void*)p, &meminfo, sizeof(meminfo))) break;</pre>
	<pre>if ((meminfo.RegionSize==0x1000) && (meminfo.Type==MEM_PRIVATE)</pre>
	&&(meminfo.State==MEM_COMMIT)&&(meminfo.Protect==PGE_EXECUTE_READWRITE
	&&(*((unsigned int*)p)==0x04247C83)) return ASTERIX_HIDE_DEBUGGER;
	<pre>p += meminfo.RegionSize;</pre>
}	
//	I'm too lazy to parse IAT of the KERNEL32.DLL, so I just check out
//	the address of the NtQueryInformationProcess, found in GetLogicalDrives
//	of course, I have no guarantee the code of GetLogicalDrives will be
//	unchanged in the next versions of Windows. I know to parse IAT, but
//	I don't want to. I told you, I'm too lazzy.
Χ =	<pre>= (BYTE*) GetProcAddress(GetModuleHandle("KERNEL32.DLL"),"GetLogicalDriv</pre>
for	^ (a = 0; a < 0x69; a++)
{	
	if($(*((DWORD*)x)==0x15FFFF6A)$ && $(*((WORD*)(x+8))==0x0C085)$
	&&(*(DWORD*)(*(DWORD*)(x+4)) < (DWORD)GetModuleHandle("NTDLL.DLL")))
	return MessageBox(0,new, hid,0); return UNKNOWN_HIDE_DEBUGGER;
	X++;

// If we're here, well... hide-debugger plug-in is not detected :-)