# 4. Самотрассировка и прочие

Энциклопедия антиотладочных приемов

## **ГОЛОВОЛОМКИ** Крис Касперски, 01.08.2008 С Комментарии О 320 О Добавить в закладки

Содержание статьи

и на ошибках в debug engine, о которых я расскажу по ходу дела. А заодно

продемонстрирую интимные подробности основных хакерских инструмен-



### 02. Алгоритм 03. Счастливый финал Сегодня мы будем ломать мой crackme, напичканный антиотладочными приемами. Они основаны на особенностях обработки исключений отладчиками

01. Что мы будем ломать

тов — «Ольги», «Иды», Syser'a, х86emu и прочих. Разгадывать загадки намного интереснее, чем читать готовые решения. А потому, пока еще не поздно, оторвись от статьи и попробуй расковырять JohoR crackme. В подсказку не заглядывать! Исходный текст не смотреть! Впрочем, сам по себе исходный текст (даже с учетом всех комментариев) совершенно не объясняет, как же его отлаживать.

www Скачать crackme-jhr; копия также есть в моем репозитории на OpenRCE.

Здесь отсутствуют шифровка, самомодификация и прочие приемы, ослепляющие статический анализ. Дизассемблер выдает аккуратный листинг, каждая машинная команда которого

```
абсолютно понятна. Однако результат действия программы в целом очень трудно предска-
зуем и требует довольно глубоких знаний устройства процессора и операционной системы.
Поистине танталовы муки! Какой-то несчастный десяток машинных инструкций (ядро
crackme) отделяет нас от победы! Что ж, тем большее наслаждение испытываешь от взлома!
Ну а на случай, если самому взломать никак не получается, я даю развернутое объяснение.
```

ЧТО МЫ БУДЕМ ЛОМАТЬ Исходный текст JohoR crackme приведен в листинге 1. Это чудо моей инженерной мысли после компиляции занимает всего 832 байта, большая часть которых приходится на РЕзаголовок. Конечно, его можно было бы ужать, программируя в hex-кодах, но это ж сколько труда надо потратить! А так — файлы легко компилируются штатными утилитами Microsoft. \_ 🗆 × Syser : Active Hotkey [Ctrl+F12]

#### (gfsProvider: NPOpenEnum fgfsProvider: NPOpenEnum returning status O

🖺 🐧 🗊 🖫 💹 🚳 💌

🦨 Loader 💹 Debugger 📃 View 🌟 Tools 🍪 Help

fgfsProvider: MPEnumResource Count Requested -1 HgfsProvider: Enum: Enumerate root. HgfsProvider: Enum: Success. remote name = \\.host, provider = VMware Shared Folders. HgfsProvider: MPCloseEnum HgfsProvider: MPGetConnection: localName = Z:, size: 260 HgfsLockSharedMemory: Entered HgfsLockSharedMemory: Exiting HgfsUnlockSharedMemory: Entered HgfsUnlockSharedMemory: Exiting

```
HgfsProvider: MPGetConnection: localName = Z:, size: 40
  igfsLockSharedMemory: Entered∶
 HgfsLockSharedMemory: Exiting
HgfsUnlockSharedMemory: Entered
  MgfsUnlockSharedMemory: Exiting
 HgfsProvider: MPGetConnection: localName = Z:, size: 260
HgfsLockSharedMemory: Entered
HgfsLockSharedMemory: Exiting
  igfsUnlockSharedMemory: Entered
 HgfsUnlockSharedMemory: Exiting
HgfsProvider: MPGetConnection: localName = Z:, size: 260
HgfsLockSharedMemory: Entered
  MgfsLockSharedMemory: Exiting
  KgfsUnlockSharedMemory: Entered
 HgfsUnlockSharedMemory: Exiting
 Syser : Fail to clear left BP [crackme_jhr] 004001E0 Module Entry point \??\C:\KPNC\crackme_jhr.exe
 🔀 Conzole & Debug Mezzage
                               Personal Version : Lincense to Kris Kaspersky
                  Syser напрочь отказывается грузить JohoR crackme в release build'e
Кстати, о компиляции. По многочисленным просьбам трудящихся я отказался от командных
файлов и перешел на макак (в смысле, на .mak), обрабатываемых утилитой NMAKE из ком-
плекта поставки MS Visual Studio. NMAKE /f crackme_jhr.mak собирает релиз, а NMAKE /f
"crackme_jhr.mak" CFG="crackme_jhr - Win32 Debug" — отладочную версию. Только все
равно отладить ее с помощью MS Visual Studio не удастся — нет смысла даже пытаться.
Также поддерживается сборка и из IDE - достаточно открыть макаку и сделать build. Тупая
«Студия» всегда ищет скомпилированный файл в каталогах \Debug и \Release, тогда
как мыщъх создает его в текущей директории, поэтому запуск файла непосредственно
из IDE невозможен (хотя, может быть, в последних версиях MS уже пофиксила этот косяк).
```

int count; char str[]="0123456789ABCDEF!";  $\_\_$ asm ${}$ ; //int 03 ; // For soft-ice xor eax, eax; // eax := 0 mov ebx, fs:[eax] ; // Old SEH

```
;//-[hacker time]-;
```

push offset l1 ; // Handler proc

pushfd ; // save EFLAGS

;//-[new seh]-;

Исходный текст JohoR crackme

#include <windows.h>

```
push -1 ; // The last handler in the chain
    mov fs:[eax], esp ; // Assign the new handler
   xor eax,[eax] ; // <-- ACCESS VIOLATION</pre>
   ;//-[set TF bit]-;
   push -1; // TF := 1
   xor eax,[eax] ; // <-- ACCESS VIOLATION</pre>
   popfd ; // EFLAGS := 00244ED7h
   ;//-[TRACE-ZONE]-;
    mov eax, [eax] ; // <-- ACCESS VIOLATION
   nop ; // <-- INT 01
   ud2 ; // <-- ILLEGAL INSTRUCTION</pre>
   nop ; // <-- INT 01
   nop ; // <-- INT 01
   int 03; // <-- INT 01
    jmp end_of_line ; // :-- to exit -->
   ;//-[seh handler]- -;
   11: mov eax, [esp + 04h]; // *EXCEPTION_RECORD
   12: mov edx, [esp + OCh]; // EDX -> ContextRecord
    mov eax, [eax] ; // EXCEPTION CODE
    cmp eax, OCOOOOOO1Dh ; // ILLEGAL INSTRUCTION
    jz x2
            ; // X-->
    cmp eax, 080000003h; // INT 03
    jz x1 ; // -- skip 1 byte -->
    cmp eax, OCOOOOOO5h; // ACCESS VIOLATION
   jnz set_tf_bit ; // -- don't skip -->
   x2:inc dword ptr [edx+0B8h]; // Skip one byte
   x1:inc dword ptr [edx+0B8h]; // Skip one byte
    set_tf_bit: ; // <--X
    cmp dword ptr [edx + 0B8h], offset end_of_line
    jae end_of_handler ; // dont set TF-bit _outside_ trace-zone
    or dword ptr [edx+0C0h],100h ; // <---- set TF-bit _inside_ trace-zone
    end_of_handler:xor eax,eax ; // EXCEPTION_CONTINUE_SEARCH
   inc [count] ; // EXCEPTION COUNT
   ret ; // end of the handler
   ;//-[exit]-;
   end_of_line:mov fs:[eax],ebx ; // Restore the old SEH
   sub esp, 8; // restore the stack
    popfd ; // restore the flags
    // Print EXCEPTION COUNT
    count = str[(count>0x10)?0x10:count]; MessageBox(0,&count, "JohoR", MB_OK);
    ExitProcess(∅);
A problem has been detected and Windows has been shut down to prevent damage.
to your computer.
DRIVER_IRQL_NOT_LESS_OR_EQUAL
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x000000D1 (0x000000CC,0x000000FF,0x00000000,0xF87E04F5)
        Syser.sys - Address F87E04F5 base at F876B000, DateStamp 476e603f
```

#### его и передает управление на метку l1, с кодом C0000005h, расположенным в двойном слове по адресу [ESP + 04h]. Обработчик видит, что это ACCESS VIOLATION, и, зная, что его вырабатывает инструкция XOR EAX, [EAX], лезет в регистровый контекст. При этом он увеличивает значение EIP на два байта — sizeof(XOR EAX, [EAX]), поскольку ACCESS

Beginning dump of physical memory

равно не собираются, зачем же тогда усложнять код?).

**JohoR** 

Contact your system administrator or technical support group for further

Попытка отладки билда JohoR crackme под Syser'ом. Отладчик в панике, система в ауте, хакер

Первые три команды сохраняют указатель на текущую (системную) SEH-запись в регистре ЕВХ и заталкивают в стек флаги процессора, попутно обнуляя ЕАХ. Следующие три команды устанавливают новый SEH-обработчик, находящийся по смещению I1, замыкая SEH-цепочку терминирующим указателем -1 (FFFFFFFFh). Он сигнализирует системе о том, что данный

обработчик — последний. Вот такой маленький трюк (почему-то большинство хакеров

добавляют свой обработчик к цепочке существующих — хотя передавать им управление все

OK

Результат работы JohoR crackme при запуске без отладчика

Сразу же после установки SEH-обработчика выполняется команда XOR EAX, [EAX], «выб-

расывающая» исключение доступа типа ACCESS VIOLATION. Операционная система ловит

VIOLATION представляет собой fault. Если пояснять, то в момент генерации исключения

X

Physical memory dump complete.

assistance.

на измене

**АЛГОРИТМ** 

регистр EIP указывает на начало возбудившей его машинной команды. При выходе из обработчика процессор будет выполнять XOR EAX, [EAX] снова и снова, пока мы либо не изменим ЕАХ так, чтобы он указывал на валидную область памяти, либо не увеличим значение EIP, переходя к выполнению следующей машинной команды. Что мы и делаем, попутно увеличивая счетчик вызова исключений (count) на единицу. Зачем нам это? При пошаговой трассировке программы, когда взведен ТҒ-бит, операционная система следом за ACCESS VIOLATION генерирует SINGLE STEP. В результате вместо одного исключения мы получаем целых два, и SEH-обработчик под отладчиком вызывается дважды, позволяя программе обнаружить, что ее ломает злобный хакер! Отладчики MS VC,

MS WinDbg, Soft-Ice (и ряд других) давят исключение SINGLE STEP, сбрасывая флаг трассировки через контекст, а вот «Ольга» 1.1х об этом не заботится. Ошибка была исправлена только в версии 2.х (все еще находящейся в разработке). Подробнее об этом мыщъх расска-

Три следующие машинные команды взводят флаг трассировки. На самом деле флаг трассировки взводится с помощью всего двух команд — PUSH -1/POPFD, a XOR EAX, [EAX], pac-

положенная между ними, вставлена для борьбы с одним экспериментальным отладчиком, что «отлавливает» инструкцию POPFD. Если ломаемая программа взводит бит трассировки,

отладчик врубает модуль эмуляции, но, если управление на POPFD передается посредством

зывает в своем блоге (bug in Olly, Windows behavior and Peter Ferrie).

правки регистрового контекста, отладчик этого не просекает. Точнее, раньше не просекал, а сейчас ошибка исправлена. POPFD выталкивает -1 (FFFFFFFh) из стека, устанавливая процессорные флаги в единицу. Конечно, далеко не все флаги, а только те, которые позволено модифицировать прикладной программе. О чем, кстати говоря, «догадывается» далеко не каждый эмулирующий отладчик, а потому значение EFLAGS под «живым» процессором и, например, х86emu сильно отличаются. Но х86ети, в общем-то, и не подряжался эмулировать все флаги процессора. В принципе, здесь можно вставить проверку — если EFLAGS не равняется 00244ED7h, то одно из двух: либо нас ломают на эмуляторе, либо это какой-то очень левый процессор. Впрочем, поскольку достойных эмулирующих отладчиков под х86 все равно нет, подобная проверка лишена смысла.

Но не будем отвлекаться. Флаг трассировки успешно взведен, и по завершении команды, следующей за POPFD, процессор генерирует пошаговое исключение. А этой командой является наша старая знакомая XOR EAX, [EAX], «выбрасывающая» ACCESS VIOLATION, что

предшествует пошаговому исключению. В данном случае система генерирует два вполне законных исключения. Вот только большинство отладчиков ошибочно принимают исключение, сгенерированное программой, за свое собственное и давят его. В результате чего SEH-обработчик вызывается на один раз меньше. «Ольга» 1.1х эту ситуацию обрабатывает

вполне правильно (точнее, никак не обрабатывает, тупо передавая все исключения программе), а вот исправленная «Ольга» 2.х путается в исключениях и «давит» лишнее (с ее точки зрения) пошаговое исключение. Другими словами, под «Ольгой» 1.1х SEH-обработчик вызывается на один раз больше, чем под «живым» процессором (с учетом первой команды MOV EAX, [EAX]), а под «Ольгой» 2.х — на один раз меньше. Красота! И какую же версию нам

выбирать? Что касается Soft-Ice (и некоторых других отладчиков), он «кушает» все пошаго-

вые исключения, генерируемые отлаживаемой программой, обламывая самотрассировку. Потому SEH-обработчик вызывается только на MOV EAX, [EAX] — как следствие, счетчик

вызовов count оказывается намного меньше, чем ожидает защита, сразу же понимающая, с кем она имеет дело. Команда NOP «честно» генерирует пошаговое исключение (ведь бит трассировки взведен!), но Soft-Ice его поглощает. Остальные отладчики (типа «Ольги» и IDA-Pro) хотя бы можно настроить на отдачу пошаговых исключений ломаемой программе. Причем IDA-Pro 5.2 предложит сделать это автоматически, в то время как «Ольга» требует ручной настройки (вкладка Exceptions в опциях отладчика). 🚅 Debug 💹 Monitor 📋 View 🔢 Information 🔆 Tools 🕜 Help Ver: 1,95,1900,089 🥦 📭 🝗 🔰 🚚 🔻 🔯 📭 👺 🖺 📙 🕒 🧸 👰 🗟 🚇 Register|Value[F2]|Context 📗 🔯 00000000 00000000 00000010: 7FFDF000 00000020 00000030

00000040:

00000050: 00000060

00000070:

00401015 90

00401012

00401016

0040101B

0040101D

00401020

00401022

00401024

00401026

00401027

00401029

0040102A

0040102B

00401031

00401010 3300

648B18

6AFF

3300

6AFF

3300

8B00

INVALID

8B442404

OB909OCCEB4A

648920

6831104000

?? ?? ?? ?? ?? ?? ?? ??

INT3

PUSHFD

PUSH -01

PUSH -01

POPFD

NOP

DB

OR

XOR EAX, EAX

PUSH 00401031

MOV EBX, FS: [EAX]

MOV FS: [EAX], ESP

XOR EAX, [EAX]

XOR EAX, [EAX]

MOV EAX, [EAX]

EDX, [EAX+4AEBCC90]

EAX, [ESP+04]

0012FFB0

00000000 0012FFF0

0012FFC4

00401005

001B

0023

0023

003B

0000

0023

0012FFA0|F9036CA0|

0012FFA4|80A7962D

8003F000

Address | Value[F2] | Context

REG ALT+R FPU ALT+F MXMM ALT+X

0012FF9C|80822BA5||PsGetThreadFreezeCo|

HalClearSoftwareInt

# EFLAG 00000246

± CS

⊞ DS

± ES

± FS

⊞ GS

± SS

**∃** GDTR

7C82ED54 KiFastSystemCallRet

00401035 8B54240C EDX, [ESP+OC] MOV 0012FFA8|00000000 00401039 MOV EAX, [EAX] 8800 0012FFAC 00000000 0040103B 3D1D0000C0 EAX, C000001D CMP 0012FFB0 7C821E74 NtSetInformationThr 00401040 740E JΖ 00401050 0012FFB4|77E523CA IsProcessorFeatureP 00401042 | 3D03000080 CMP EAX, 80000003 0012FFB8 FFFFFFFE JΖ 00401056 00401047 740D 0012FFBC 00000009 3D050000C0 00401049 CMP EAX, C0000005 0012FFC0 0012FFF8  $\Pi Z$ 0040104E 750C 0040105C 0012FFC4|77E523CD <-ESP IsProcessorFe [EDX+000000B8] 00401050 FF82B8000000 INC 0012FFC8 00000000 00401056 FF82B8000000 INC [EDX+0000000B8] 0012FFCC 00000000 0040105C | 81BAB80000007B | CMP [EDX+000000B8],0040107B 0012FFD0|7FFDF000 00401072 0012FFD4 0000000F Syser Debugger Personal Version : License to Kris Kaspersky Break at Module Entry point \??\C:\KPNC\crackme\_jhr\_dbg.exe Break at Module Entry point \??\C:\KPNC\crackme\_jhr\_dbg.exe Entry point \??\C:\KPNC\crackme\_jhr\_dbg.exe 0012FFD8 0012FFC8 0012FFDC|F9036CF8 0012FFEO|FFFFFFFF Реакция Syser'a на машинную команду UD2 Инструкция UD2 генерирует исключение типа ILLEGAL INSTRUCTION, перехватываемое Soft-Ice и не отдаваемое отлаживаемой программой вплоть до отдачи команды faults off. Syser такой команды вообще не знает, обзывая ее как DB («объявить байт») и неверно дизассемблируя весь последующий код (что не покажется удивительным, если вспомнить, что UD2 двухбайтовая команда). Пара последующих NOP'ов не делает ничего, кроме генерации пошагового исключения, особенность обработки которого мы обсуждали двумя абзацами выше. Так зачем же тогда они нужны? Подсказка — разные отладчики имеют разные баги, «съедая» различное количество исключений. Правильно! Данный crackme определяет тип отладчика по значению count, уникальность которого обеспечивается соотношением команд, генерирующих свои собственные исключения, к общему количеству трассируемых инструкций. Если убрать NOP'ы, crackme продолжит детектить активную отладку, но уже не сможет определить, какой именно отладчик используется хакером. JohoR

руживаемую подсчетом контрольной суммы и «разваливающую» самомодифицирующий код. Впрочем, ни того, ни другого в crackme нет, а есть только INT 03h. Чисто теоретически отладчик может (и должен) отличать свои собственные INT 03h от чужих, отдавая программе только те исключения, которые она сама же и сгенерировала. Но на практике отладчики путаются. «Ольга», настроенная на отдачу INT 03h ломаемой программе, при установке программной точки останова поверх INT 03h циклится, вынуждая хакера применять аппаратные точки останова (которых всего четыре) или точки останова на регион памяти, реализованные через подмену атрибутов страниц, что также легко обнаруживается. Кстати, с точки зрения процессора, INT 03h генерирует trap, а не fault. То есть регистр EIP в момент генерации исключения смотрит на команду, следующую за INT 03h, которой в нашем случае является двухбайтовая инструкция JMP END\_OF\_LINE. В чем же подвох? А в том, что SEH-обработчик отлавливает BREAKPOINT-исключение (соответствующее коду 080000003h) и увеличивает значение EIP на единицу. Неужели управление передается в середину инструкции JMP END\_OF\_LINE? Какой хитрый прием против дизассемблера! Гм, вот только непонятно... первый байт опкода JMP SHORT равен EBh, второй представляет

относительное смещение целевого перехода. И чтобы оно соответствовало осмысленной

машинной инструкции, необходимо, чтобы метка END\_OF\_LINE располагалась на определен-

ном смещении от команды JMP. А в crackme между ними расположен SEH-обработчик.

Выходит, если его изменить, то crackme cpasy перестанет работать? Такая хитрая защита

OΚ

Результат работы JohoR crackme при запуске под IDA-Pro 4.7

Команда INT 03h также вставлена неспроста. Если даже настроить отладчик на отдачу INT

03h ломаемой программе, наличие INT 03h существенно затрудняет отладку. Если бы INT

03h не было, то, чтобы быстро выбраться из глубин системного обработчика исключений назад к ломаемой программе, достаточно было бы покрыть трассируемый блок программными точками останова (в «Ольге» для этого нужно на каждой команде нажать F2). Программные точки останова представляют собой однобайтовую инструкцию CCh, легко обна-

исходных текстов от изменения! И чего только со страху не покажется. Да, в руководствах от Intel черным по белому написано, что BREAKPOINT — это trap, а не fault. Вот только SEH-обработчик вызывается не процессором, а операционной системой. Той, что написана компанией Microsoft. A Microsoft Way умом не понять. Ну чем можно объяснить, что она подменяет процессорный контекст, умышленно уменьшая EIP на единицу? Парни из Microsoft впопыхах забыли, что BREAKPOINT может генерироваться как опкодом CCh, так и CDh 03h, а потому если внедрить CDh 03h в программу и никак ее не обрабатывать, то после выхода из исключения регистр EIP будет смотреть на опкод 03h, соответствующий команде ADD чего-то там. Допустим, за CDh 03h следует CCh (еще один INT 03h, только слегка другой). Тогда процессор выполнит опкод 03h CCh — ADD ECX, ESP. Вот и попробуй догадаться об этом при дизассемблировании! 🌟 OllyDbg - crackme-jhr.exe - [CPU - main thread, module crackme-] C File <u>View D</u>ebug Plugins Options <u>W</u>indow <u>H</u>elp 004001E0 \$ 33C0 XOR EAX, EAX

004001E2 . 64:8B18 MOV EBX, DWORD PTR FS: [EAX] 004001E5 . 90 PUSHFD . 68 01024000 004001E6 PUSH crackme-.00400201 004001EB . 6A FF PUSH -1 MOU DWORD PTR FS:[EAX],ESP 004001ED . 64:8920 004001F0 XOR EAX, DWORD PTR DS: [EAX] . 3300 004001F2 . 6A FF PUSH -1 004001F4 . 3300 XOR EAX, DWORD PTR DS:[EAX] 004001F6 MOV EAX, DWORD PTR DS:[EAX] . 90 004001F9 NOP 004001FA . 0F 0B UD2 004001FC . 90 NOP . 90 NOP 004001FE . CC INT3 ., EB 4A JMP SHORT crackme-.0040024B . 8B4424 04 MOV EAX, DWORD PTR SS:[ESP+4] . 8B5424 OC 00400205 MOV EDX, DWORD PTR SS:[ESP+C] . 8B00 00400209 MOV EAX, DWORD PTR DS: [EAX] 0040020B . 3D 1D0000C0 CMP EAX,C000001D JE SHORT crackme-.00400220 00400210 ...74 GE . 3D 03000080 CMP EAX,80000003 00400212 ...74 OD 00400217 JE SHORT crackme-.00400226 . 3D 050000C0 00400219 CMP EAX,C0000005 JNZ SHORT crackme-.0040022C 0040021E ...75 OC 00400220 > FF82 B8000000 INC DWORD PTR DS:[EDX+B8] > FF82 B8000000 INC DWORD PTR DS:[EDX+B8] > 81BA B8000000 CMP DWORD PTR DS:[EDX+B8],crackme-.0040 JNB SHORT crackme-.00400242 8180 CRRRRRRR OR NUMBER OF TENSHCRI 188 Установка программных точек останова внутри «горячей» зоны Наконец, команда JMP END\_OF\_LINE выводит код из зоны трассировки. Программа восста-

навливает прежний SEH, выталкивает из стека флаги и распечатывает значение счетчика исключений, после чего завершает свое выполнение вызовом функции ExitProcess(0).

#### СЧАСТЛИВЫЙ ФИНАЛ Так как же все-таки ломают эти программы? И какими отладчиками? В случае статического кода (к которому относится мой crackme) проблема решается установкой точки останова за пределами «горячей» зоны, где происходит выброс исключений, с прогоном их на живом

ливается аппаратная точка останова.

процессоре (без пошаговой трассировки). Если же нам жизненно необходимо подсмотреть значение некоторых регистров или ячеек памяти внутри «горячей» зоны — на них устанав-

Динамический код (упакованный, зашифрованный, самомодифицирующийся) заломать намного сложнее, поскольку нам реально необходимо прогнать его через пошаговый трассировщик, с которым и борется защита. Заметим, весьма эффективно борется. BOCHS (бесплатная Скачано не селита стровно в не говорить. В дествен риссов диняйся! сколько грузится на нем Windows, лучше не говорить.