**********

**Приложение к статье.**



Материалы по работе механизмов навесной защиты SecuRom версии 7.33.0017

**********

| Связанная процедура | Возвращаемые значения | Адреса вызова | Замечания |
|---|---|---|---|
| Анти-отладка | | | |
| WinAPI **IsDebuggerPresent**[kernel32] | EAX = 0 (Отладчик отсутствует)<br><br>1 (обнаружен) | В самом начале из VM (до UD2)<br><br>00D07078<br>00D07626 | Осуществляется проверка целостности. В TEB.ProcessDataBase->BeginDebugged протектором заносятся контрольные значения (>1), вызывается ISDP, и проверяется результат, которое должен совпасть. Адреса: 0D0697B, 00D07078 |
| WinAPI **CheckRemoteDebuggerPresent**[kernel32] | EAX = 0 (Отладчик отсутствует)<br><br>1 (обнаружен)<br>[ARG.1] DWORD Buffer = 0 (Отладчик отсутствует)<br><br>-1 (обнаружен) | 00CF1319 | ***CheckRemoteDebuggerPresent*** является обверткой ***NtQueryInformationProcess*** (ProcessInfoClass = ProcessDebugPort) |
| NativeAPI **NtQueryInformationProcess** (ProcessInfoClass = ProcessDebugPort) | [ARG.3] DWORD Buffer = 0 (Отладчик отсутствует)<br><br>-1 (обнаружен, отладка текущего процесса)/другое любое ненулевое значение(обнаружен) | 00CF237E | Выводит информацию о любом активном отладчике, вне зависимости отлаживается ли защищаемый процесс или нет |
| **-NtQueryInformationProcess** (ProcessInfoClass = BasicInformation)<br>**-GetModuleFileNameExA**[PSAPI] | ProcessParentID/ Parent Process path | 00CF275F(NtQuery)<br>00CEE25D(GetModu)<br>00DDB720(процедура извлекает имя файла)<br>00DE43B0(сверяет имя файла родителя с именами популярных отладчиков) | Протектор получает *Process ID* родителя, через ***GetModuleFileNameExA*** устанавливает полный путь к образу, извлекает имя файла и проверяет не совпадает ли оно с именами популярных отладчиков или прочих нехороших программ (*ollydbg.exe; idag.exe;idag64.exe; windbg.exe;curerom.exe;antisecurom.exe; apilogger.exe и др.*) |
| WinAPI **FindWindow**[USER32] | EAX = 0 (окно отсутствует)<br><br>EAX = handle(окно присутствует) | 00D38699(jmp) | Протектор проверяет присутствие окон с заголовками популярных отладчиков и прочих нехороших программ(*OLLYDBG;Anti-Blaxxx;Virtual-CD-Hide,[LordPE Deluxe] by yoda,FileMonitor – Sysinternals: www.sysinternals.com; Alcohol 120% и др.*) |
| WinAPI **FindWindowExA**[USER32] | EAX = 0 (окно отсутствует)<br><br>EAX = handle(окно присутствует) | 00D26D8F<br>00D38699<br>00CD4369<br><br>00D26BEC<br>00D272A2 (OpenProc)<br>00D27774 | Дублируется проверка на присутствие окон с заголовками популярных отладчиков и прочих нехороших программ(несколько расширенная).<br><br>Дополнительно. Работает с explorer.exe: читает *TrayNotifyWnd, SysPager и ToolBarWindow32*, устанавливает Process |

| | | | |
|---|---|---|---|
| | | (ReadProcMem) | ID, и через ***OpenProcess/ReadVitrualMemory*** перечисляет имена иконок в трее, ищет нехорошие программы. |
| WinAPI **CreateFileA**[kernel32] | EAX = -1(файл/драйвер отсутствует)<br><br>EAX = handle (файл/драйвер присутствует) | 00CEBECE<br>00CD6465(jmp)<br>01189CFA<br>01189D19<br>01189D61 | Протектор проверяет присутствие драйверов *SoftICE; Daemon TOOLS и прочих нехороших программ* |
| WinAPI **OpenSCManager**[ADVAPI32]<br><br>WinAPI **OpenServiceA**[ADVAPI32]<br>WinAPI **QueryServiceStatus**[ADVAPI32] | EAX = 0 (служба отсутствует)<br><br>EAX = handle (служба присутствует) | 00D41AA4<br>00D41AC3<br>00D41AD5 | Имеются ли в списке служб *SoftICE* драйвера |
| WinAPI **RegOpenKeyA**[ADVAPI32] | EAX = 2 (ключ отсутствует)<br><br>EAX = handle (ключ существует) | 00D23555 | Существует ли ключ “CureROM.Profile\Shell\Open\Command\” в HKEY_CLASSES_ROOT |
| WinAPI **GetModuleHandleA**[kernel32] (“asr.dll”) | EAX = 0 (модуль отсутствует)<br><br>EAX = Module Handle (модуль загружен) | 00D32775 | Присутствует ли в памяти процесса библиотека “asr.dll” |
| Анти X-code injection | | | |
| WinAPI **FindFirstFileA**[kernel32](File Name = ntdll*.dll)<br>WinAPI **FindNextFileA**[kernel32] | EAX = -1 (файлы не найдены)<br><br>EAX = &handle(файл присутствует) | 00D249EE<br>00D25076 | Существует ли в каталоге с *cnc3game.dat* проинектированная библиотека низкоуровневых функций(ntdll.dll) (?) |
| ASM<br>CMP [ECX], E8<br>CMP [EDX], EB<br>CMP [EAX], EB<br>CMP [ESI], E9<br><br>MOVSX EAX,BYTE PTR DS:[EAX]<br>MOV ECX,EAX<br>SUB ECX,0CC | Zero Flag = 1 (опкод переходов/INT3 присутствует)<br><br>Zero Flag = 0 (опкод переходов/INT3 отсутствует) | 00D3403D<br>00D34709<br>00D34CBF<br>00D34D5F<br><br>00CF15C1 | У некоторых WinAPI в библиотеках kernel32, NTDLL и ADVAPI32 проверяется наличие переходников в самом начале.<br><br>Или проверяется наличие программной точки останова в начале(ex:*UnhandledExectionFileter[kernel32]*) |
| Проверка диска в приводе | | | |
| WinAPI **GetDriveTypeA**[kernel32] | EAX = 5 (CD/DVD) | 00D85F89<br>00D85F63 (ф-я сравнения)<br>00D8639 (ф-я записи найденных приводов) | Первоначально формирует массив из доступных приводов |
| WinAPI **CreateFileA**[kernel32] | EAX = хэндл открытого диска<br><br>EAX =-1(ошибка) | 00D86A4B<br>00D86A02 (ф-я открытия диска)<br>00D86AE4 (общая ф-я) | Открывает диск на секторном уровне |
| WinAPI **DeviceIoControl**[kernel32]<br><br>Control code(s):<br>0004D014 [InBuffer/OutBuffer = 50h]<br>00070020 (IOCTL_DISK_PERFOMANCE)<br>00041018 | EAX = 1 (без ошибок)<br>EAX = 0 (ошибка)<br><br>InBuffer<br>OutBuffer | 00D86B75 | Именно на ***DeviceIoControl*** и возложена основная задача по проверке носителя(Геометрия диска?). Если изначально диск присутствует в устройстве:<br><br>*00D86BA1  CMP BYTE PTR SS:[EBP-52],BL*<br><br>То следующие 4 раза происходит первичная его проверка. Если она пройдена, то далее следует процедура |

| | | | |
|---|---|---|---|
| | | | поиска программ эмуляции приводов (ex: *Daemon TOOLS*). Если таковых не найдено, происходит окончательная проверка диска создаются два потока(10F0F099), которые синхронно обращаются к приводу с **IOCTL_DISK_PERFOMANCE**,в тандеме используя WInAPI ***QueryPerfomanceCounter***(способ различить виртуальный привод от реального?), в это же время основной поток также опрашивает привод, но с **ControlCode** = 0004D014 и 00041018 |
| Анти-attach | | | |
| NativeAPI<br>**DbgUiRemoteBreakin** | JMP<br>[ExitProcess] | 00DDA6EC | С помощью VirtualProtect получает разрешение на запись, затем сохраняет первые 5 байт, устанавливает адрес *ExitProcess*[KERNEL32], вычисляет операнд прыжка и выполняет запись переходника. По окончании проверки(подготовка перехода в OEP) исходный вид процедуры возвращается(т.е. действие анти-attach распространяется только на время проверки привода). |
| Разное | | | |
| **CreateFileA** | | 00CF6B68 | Доступ на чтение к файлу процесса-родителя |
| **ReadFileA** | | 00CF6C9D | Чтение файла |
| **NtQueryInformationProcess** | | 00C78F4D<br>00CF275F<br>00C78F47 | ExecuteFlags<br>BasicInformation<br>StartupFlags |
| **ReadProcessMemory** | | 00D27774 | Чтение памяти explorer.exe |
| **NtQuerySystemInformation** | | 00D1A233 | Сбор различной информации |
| **IsBadWritePtr** | | 00D066DB | Проверяет TEB на запись (относится к проверке целостности IsDP) |
| **OpenProcess** | | 00CEE1F3 | Открывает процесс-родитель с атрибутами чтения и запроса информации |
| **OpenProcessToken**<br>**LookupPriviligeValueA**<br>**AdjustTokernPrivilegies** | | 00D422D3 | Операции с маркерами доступа. Устанавливает/проверяет у текущего процесса SeDebugPrivilege |

## Коды ошибок:

Требуемый модуль безопасности не может быть запущен

**2000** – Отладчик виден по TEB

**3000** – Отладчик виден по активным аппаратным точкам останова

**5000** – Обнаружение через FindWindow

**5001** – Обнаружение через CreateFileA

**5002** – Обнаружение через FindWindowExA

**6000** – Отладчик виден по активным программным точкам останова в начале WinAPI

**8002** – Отладчик виден через CheckRemoteDebuggerPresent

**8007** – Проверка на целостность IsDebuggerPresent не пройдена (возвращаемое контрольное значение изменено)

**8011** – Отладчик виден через NtQueryInformationProcess

**10000** – Отладчик виден по имени процесса-родителя

Вставьте оригинальный диск вместо резервной копии

**1000** – Скорее связано с функцией по адресу 10F0F099, которая проверяет DISK_PERFOMANCE

## Файлы, имеющие отношение к SecuRom 7.33.0017:

**C:\windows\system32\CmdLineExt.dll**

**C:\windows\system32\CmdLineExt03.dll**

*Временные файлы:*

**drm_dyndata_7330017.dll**

**>если не создавать - не повлияет на работу защиты**

>переход по адресу 0108302C

>6(?) аргументов (адрес, длинна, указатель, длинна, адрес)

>Операции копирования, получения адресов

**drm_dialogs.dll** (действительно отвечает за диалоги)

**>если не создавать – сообщения будут выводиться средствами MessageBoxA**

**temp.ani** (иконка курсора с диском)

**>если не создавать – во время проверки курсор не поменяется**

## Код, покрывающий процесс перехода от одной значимой части протектора к другой. Если защиту трассировать, то представляет основную проблему при исследовании:

```
MOV EAX,11D7B1C // Начальный адрес зоны проверки, предъявите ваши программные точки останова в развернутом виде

MOV ESI,DWORD PTR DS:[EAX] //грузим след DWORD

ADD DWORD PTR SS:[ESP+10],ESI //складываем с предыдущим DWORD'ом

ADD EAX,4 // the next offset DWORD

DEC WORD PTR SS:[ESP+0C] //114 DWORD'ов над сложить

JNE SHORT 011D7B76           //  - while (dword [ESP+0Ch] != 0)

OR BYTE PTR SS:[ESP+10],01 // добавляем в младший байт единицу

SUB DWORD PTR SS:[ESP+10],933 /* вычитаем "контрольную сумму", пасьянс сошелся если:

DWORD [ESP+10]  <= 0. (Вообще-то там всегда нуль должен быть, отрицательное число – разработчики просто решили подстраховаться) */

PUSHFD //сохраням флаги (EFL = 206h)

…
```

```
POPFD //выстаскивам флаги (EFL = 206h)

JBE SHORT 011D7BC1 //так сошелся ли все-таки наш пасьянс? (Zero Flag(Z) = 1 или(и) Carry Flag(C) = 1?)
```

Вместе с приведенным кодом выше, можно встретить и проверку взведенного TF:

```
00C49160  PUSHFD

00C49161  MOV EAX,DWORD PTR SS:[ESP]

00C49164  NOP

00C49165  TEST AH,1

00C49168  JE SHORT cnc3game.00C4916F

00C4916A  MOV ECX,7BE

00C4916F  XOR EAX,EAX
```

В VM часто проскакивает такой код:

```
MOV DWORD PTR SS:[ESP],ESI

XOR ESI,DWORD PTR DS:[ESI]

XOR ESI,DWORD PTR SS:[ESP]
```

*Это не что иное как:* **MOV ESI, DWORD PTR DS:[ESI]**

**Аппаратные точки останова, устанавливаемые SecuRom 7:**

**DR0 -** 00C979E1 / 00C97E69

**DR1 –** 00C979F9 / 00C97E7E

**DR2 -** 00C97A17 / 00C97E95

**DR3 -** 00C97A2F / 00C97EAC

**Послания разработчиков протектора реверсерам:**

Начало первого островка VM, инструкция JMP SHORT перепрыгивающая ASCIIZ строку:
<space for rent>
В более поздних версиях: &You Are Now Entering a Restricted Area

Строка: yates is still ere.something kinda Ooooh
Строка:-[ Masses Against the Classes <°>>< ]
Забавная асм инструкция: MOV EDX, DEADC0DE
Обозначение емкости( C ) и изображение конденсатора псевдографикой

**Примечания:**

- Для обнаружения окна утилиты PROCESS EXPLORER дополнительно берется имя компьютера и пользователя.

- Аппаратными точки останова можно спокойно пользоваться до и после их проверки SecuRom 7 в начале

- Программные точки останова проверяются только в первой инструкции WinAPI

- Кроме явного обнаружения отладчика, SecuRom 7 также может не выдавать никакой информации, и перенаправить на ошибочный код. В подавляющем большинстве случаев причиной служат:

    1) не сброшенный флаг трассировки(T/TF)

    2) Программные точки останова, которые обнаруживает покрывающий код(приведен выше)

- Кроме X-кода, запретить SecuRom 7 создавать во вмененной папке файлы можно вполне легально и с помощью антивируса, например с McAfee VSE 8.8/8.7i (VirusScan Console-> Access Protection -> Anti-spyware Maximum Protection: Prevent all programs from running files from the Temp folder)