Dr.WEB®
Антивирус

Защити созданное

От BackDoor.Tdss.565 и выше (aka TDL3)

125124, Россия, Москва,
3-я улица Ямского поля,
вл.2, корп. 12а

Телефон:
+7 (495) 789-45-87
(многоканальный)

Факс:
+7 (495) 789-45-97

www.drweb.com
www.freedrweb.com
www.av-desk.com

Инсталляция

Уже с первых минут знакомства данная вредоносная программа начинает преподносить сюрпризы. Например, на этапе инсталляции используется оригинальный способ внедрения в системный процесс. Способ документированный, но ранее ни в одном вирусе не применявшийся. Это позволило обойти (обмануть) большинство поведенческих анализаторов и безнаказанно инсталлировать свой драйвер.

Теперь инсталляция продолжается уже в режиме ядра. Руткит проходит по стеку устройств, обслуживающих системный диск, и определяет соответствующий драйвер - свою будущую жертву для заражения. На какой именно модуль упадет выбор, зависит от конфигурации оборудования. Так, например, для системы, где системный диск имеет IDE-интерфейс, это будет **atapi.sys**, а в другой системе им может оказаться **iastor.sys**. Инфицирование драйверов файловой системы, сетевых драйверов и даже самого ядра уже не раз встречалось (BackDoor.Bulknet,Win32.Ntldrbot, Trojan.Spambot и т.д.) для обеспечения автозагрузки, и данный случай не является исключением. Стоит отметить, что размер зараженного файла не изменяется, так как код вируса перезаписывает часть секции ресурсов. Причем код этот совсем небольшой - 896 байт (в более поздних версиях уже всего 481 байта) и представляет собой загрузчик основного тела руткита. При этом меняется точка входа, обнуляется ссылка на подпись драйвера и пересчитывается новая контрольная сумма файла. Адреса API-функций, необходимых этому загрузчику во время заражения, фиксируются в его теле в виде RVA. С одной стороны, это позволило существенно сократить его размер, а с другой - затруднить исследование зараженного драйвера на системе с другой версией ядра.

Затем вирус оценивает размер диска и выделяет для себя небольшой участок (24064 байта) с конца диска, где будет храниться основное тело руткита. Фактически, основным телом руткита становится часть драйвера, который занимается инсталляцией, но уже в виде бинарных данных, а не исполняемого образа. Этот блок начинается с маркера '**TDL3**', далее следует 896 байт оригинальной секции ресурсов из зараженного драйвера. Там же, в конце диска, организуется отдельный виртуальный диск для компонентов пользовательского режима и файла конфигурации. Очень похоже, что на этот шаг авторов вдохновил BackDoor.Maxplus, который также создавал отдельное виртуальное устройство-диск для внедряемых компонентов. Подробнее об этом будет рассказано ниже.

В более поздних версиях (**BackDoor.Tdss.1030**) оригинальные данные ресурсов и само тело руткита сохраняются уже непосредственно на скрытом шифрованном диске в файлах **rsrc.dat** и **tdl** соответственно, что позволяет заметно облегчить возможность его обновления.

После завершения инсталляции драйвер возвращает ошибку STATUS_SECRET_TOO_LONG (0xC0000154), что на самом деле информирует компоненты пользовательского режима (http://vms.drweb.com/search/?q=BackDoor.Tdss.565) об успехе, а систему заставляет выгрузить уже не нужный драйвер.

Загрузчик

Вместе со стартом зараженного драйвера управление получает вирусный загрузчик. Как уже было сказано выше, его задача - подгрузить основное тело руткита, которое располагается в конце жесткого диска. В силу того, что загрузчик становится активен при загрузке ядром драйвера порта жесткого диска, он еще не может обращаться ни к диску, ни тем более к файловой системе. Поэтому он регистрирует функцию уведомления о создании управляющих объектов-устройств **FS** (FileSystem), а уже потом подгружает основное тело. Первые версии вируса использовали для этого функцию **IoRegisterFsRegistrationChange**, а последующие – временный перехват **IRP_MJ_DEVICE_CONTROL** в **DRIVER_OBJECT** жертвы и ожидание обработчиком определенного запроса от файловой системы. Интересно, что и в том, и другом случае точка входа в зараженный драйвер является функцией двойного назначения, которая одновременно используется и для запуска оригинальной **DriverEntry,** и для ожидания **FS** (Рисунок 1).

Для удобства дальнейшего повествования будем считать, что в системе заражен драйвер **atapi.sys.**

Рассмотрим более детально работу загрузчика **BackDoor.Tdss.565**. Получив управление, он обходит таблицу секций своего носителя и модифицирует ее таким образом, чтобы усложнить опознание секции инициализации: обнуляет бит **IMAGE_SCN_MEM_DISCARDABLE** у каждой секции, меняет первый байт имени на ноль, если это **INIT**. Кроме того, выделяет вспомогательную структуру данных для записи указателя на объект драйвера **atapi**, поступивший от ядра в **DriverEntry**. И далее регистрируется у ядра на нотификации о создании **CDO** (Control Device Object) **FS**.

```
.rsrc:00026780                  push    ebp
.rsrc:00026781                  mov     ebp, esp
.rsrc:00026783                  sub     esp, 160h
.rsrc:00026789                  call    $+5
.rsrc:00026789
.rsrc:0002678E                  pop     eax
.rsrc:0002678F                  sub     eax, 0F9A82A0Bh
.rsrc:00026794                  mov     [ebp+var_C], eax
.rsrc:00026797                  mov     eax, [ebp+var_C]
.rsrc:0002679A                  add     eax, 350h
.rsrc:0002679F                  add     eax, 0F9A829FDh
.rsrc:000267A4                  mov     [ebp+pParamBlock], eax
.rsrc:000267A7                  cmp     [ebp+RegistryPath], 1
.rsrc:000267AB                  jbe     jIsFsChangeOccur ; atapi init, not jmp
.rsrc:000267AB
.rsrc:000267B1                  mov     eax, [ebp+DriverObject]
.rsrc:000267B4                  mov     eax, [eax+DRIVER_OBJECT.DriverSection]
.rsrc:000267B7                  mov     eax, [eax]
.rsrc:000267B9                  mov     [ebp+PsLoadedModuleList], eax
.rsrc:000267B9
.rsrc:000267BC
.rsrc:000267BC  jNextDriverLdrEntry:                    ; CODE XREF: DriverEntry+4F↓j
.rsrc:000267BC                  mov     eax, [ebp+PsLoadedModuleList]
.rsrc:000267BF                  movzx   eax, [eax+KLDR_DATA_TABLE_ENTRY.LoadCount]
.rsrc:000267C3                  test    eax, eax
.rsrc:000267C5                  jz      short jPsLoadedModuleListWasFound
```

Рисунок 1. Точка входа зараженного BackDoor.Tdss.565 драйвера atapi.sys

При поступлении сообщения от файловой системы запускается вторая часть вирусного загрузчика. В ней он обходит все объекты-устройства драйвера-порта (например, “**\Device\IdePort0”, ”\Device\IdeDeviceP0T0L0-3”**) и пытается прочитать основное тело руткита по смещению на диске, прописанное в теле загрузчика во время инсталляции. При этом просто и незатейливо используются обычные функции **ZwOpenFile, ZwReadFile**. Такой, казалось бы, нелогичный прием с перебором устройств является следствием компактности кода, и он вполне эффективен. Успешное считывание проверяется по сигнатуре **TDL3** в начале данных (Рисунок 2). После этого нотификация удаляется (**IoUnregisterFsRegistrationChange**), и управление передается основному телу руткита.

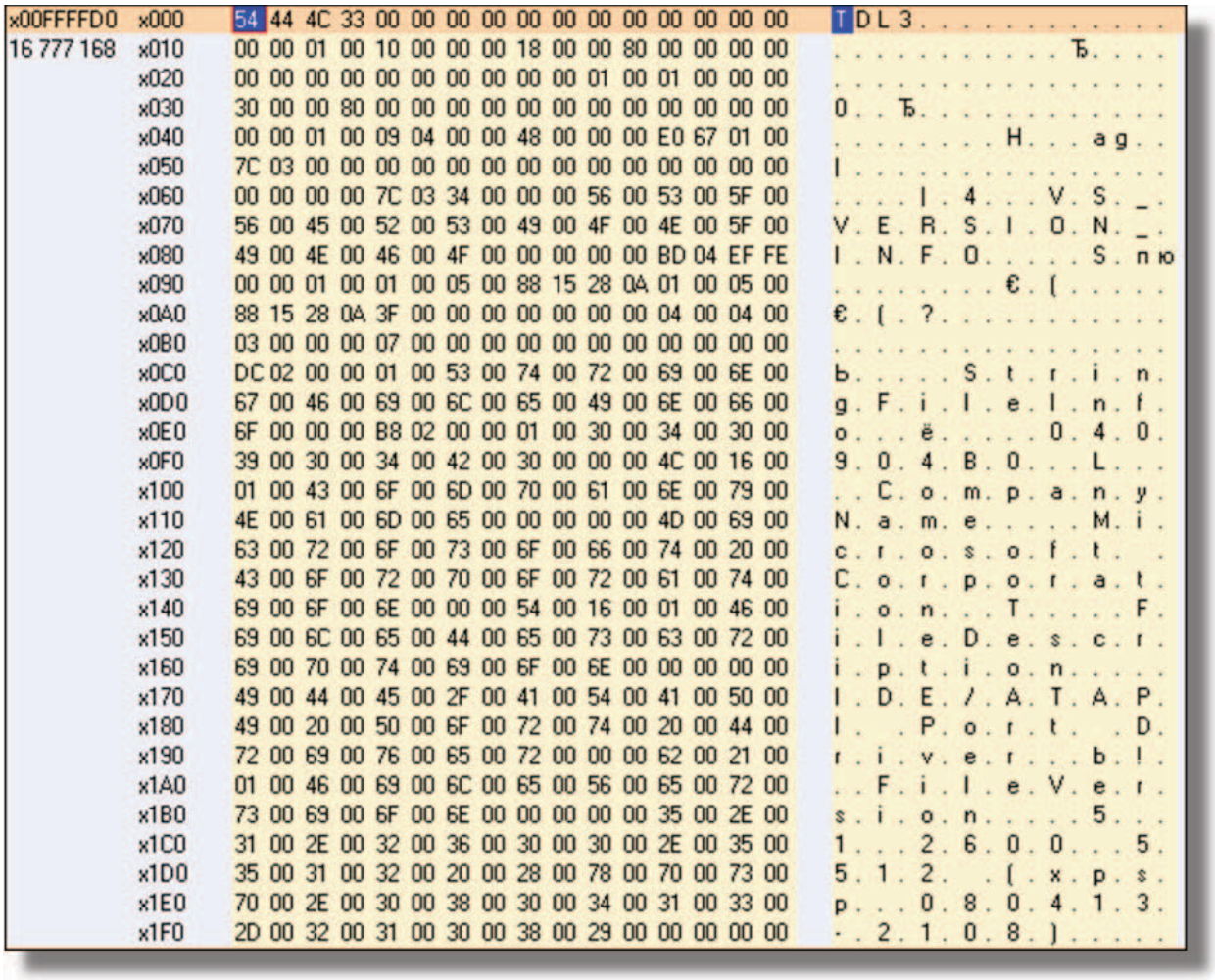


Рисунок 2. Первый сектор тела руткита в последних секторах диска

Руткит

Одной из основных технических особенностей TDL3 можно смело назвать создание скрытого шифрованного виртуального диска с **собственной** файловой системой. Не менее интересен и механизм сокрытия не только целого файла на уровне драйвера порта, но даже части произвольного сектора диска. В известных ранее руткитах полноценного воплощения этих идей не встречалось.

Известно, что основной особенностью виртуальной файловой системы NT является возможность представлять на уровне дескрипторов все устройства ввода-вывода. Ключевым звеном здесь является файловый объект, который создает ядро и объекты-устройства, представляющие данное физико-логическое устройство. Приложение открывает дескриптор на канал, жесткий диск, том, файл, и для их обслуживания подключаются разные слои стека устройств ввода-вывода. Все, что необходимо ядру - это знать, какой запрос поступил, и вызвать соответствующую функцию обработчик.

Разработчики руткита пошли схожим путем: они реализовали свою файловую систему, которая работает на уровне объекта-устройства драйвера порта. Вирус как бы *примонтирует* свою **FS** к этому объекту устройства.

Драйвер **atapi** создает несколько типов объектов устройств (Рисунок 3). Верхние два – это устройства, которые представляют собственно диски или CD-приводы, а вот два других выступают как контроллеры и фактически обслуживаются драйвером мини-порта, который является гибридным в Windows XP, т.е. представляет собой одновременно и порт, и мини-порт. Руткит выбирает для монтирования **FS** объект-устройство с типом **FILE_DEVICE_CONTROLLER**.

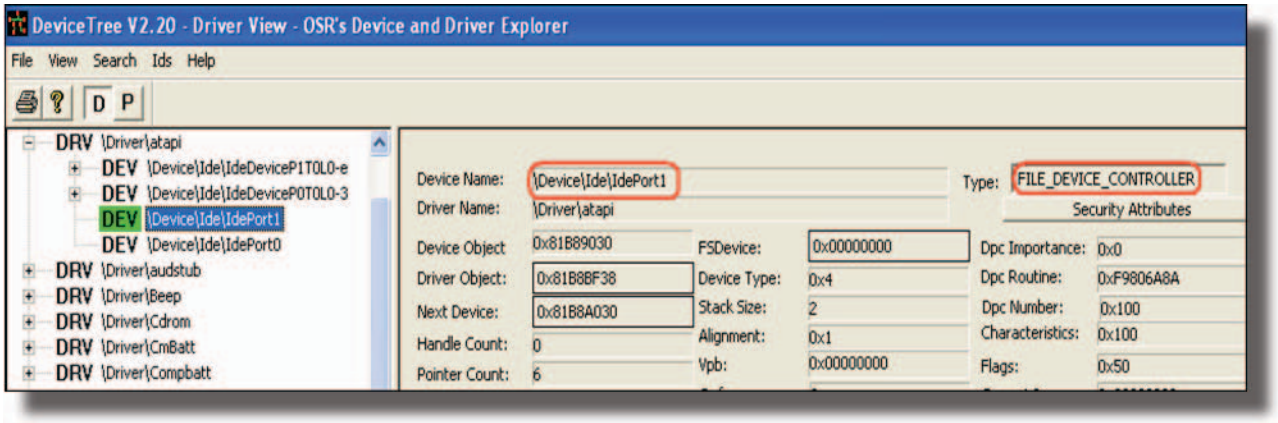


Рисунок 3. Устройства, созданные atapi.sys

Обычный (незараженный) **atapi** обслуживает запросы на чтение/запись с помощью только одного IRP – **IRP_MJ_SCSI** (**IRP_MJ_INTERNAL_DEVICE_CONTROL**). Клиент заполняет **Srb** и посылает его на дисковый объект-устройство. При обработке запросов **Create/Close atapi** всегда возвращает **SUCCESS**, они ему не нужны. Но операция **Create** очень важна для **FSD** (File System Driver), поскольку он инициализирует файлы **FILE_OBJECT**, используемый для обслуживания файлов.

Путь к файлам руткита, находящимся в защищенной (скрываемой) области, выглядит следующим образом: **\Device\Ide\IdePort1\mjqxtpex**, где **mjqxtpex** - это 8-байтовая сигнатура, которая генерируется случайным образом при загрузке системы. На этот скрытый диск компоненты пользовательского режима сохраняют свои файлы, полученные из сети, или читают с него настройки.

Пример полных путей к файлам:

```
\\?\globalroot\Device\Ide\IdePort1\mjqxtpex\tdlcmd.dll
\\?\globalroot\Device\Ide\IdePort1\mjqxtpex\tdlwsp.dll
\\?\globalroot\Device\Ide\IdePort1\mjqxtpex\config.ini
```

Чтобы понять, как руткит обслуживает свою файловую систему, обратимся к схеме, представляющей обработку create-запроса в обычном случае, т.е. **ntfs** или **fastfat**. Рассмотрим открытие файла **\Device\HarddiskVolume1\directory\config.ini** на скрытом диске **\Device\Ide\IdePort1\mjqxtpex\config.ini (**Рисунок 4).

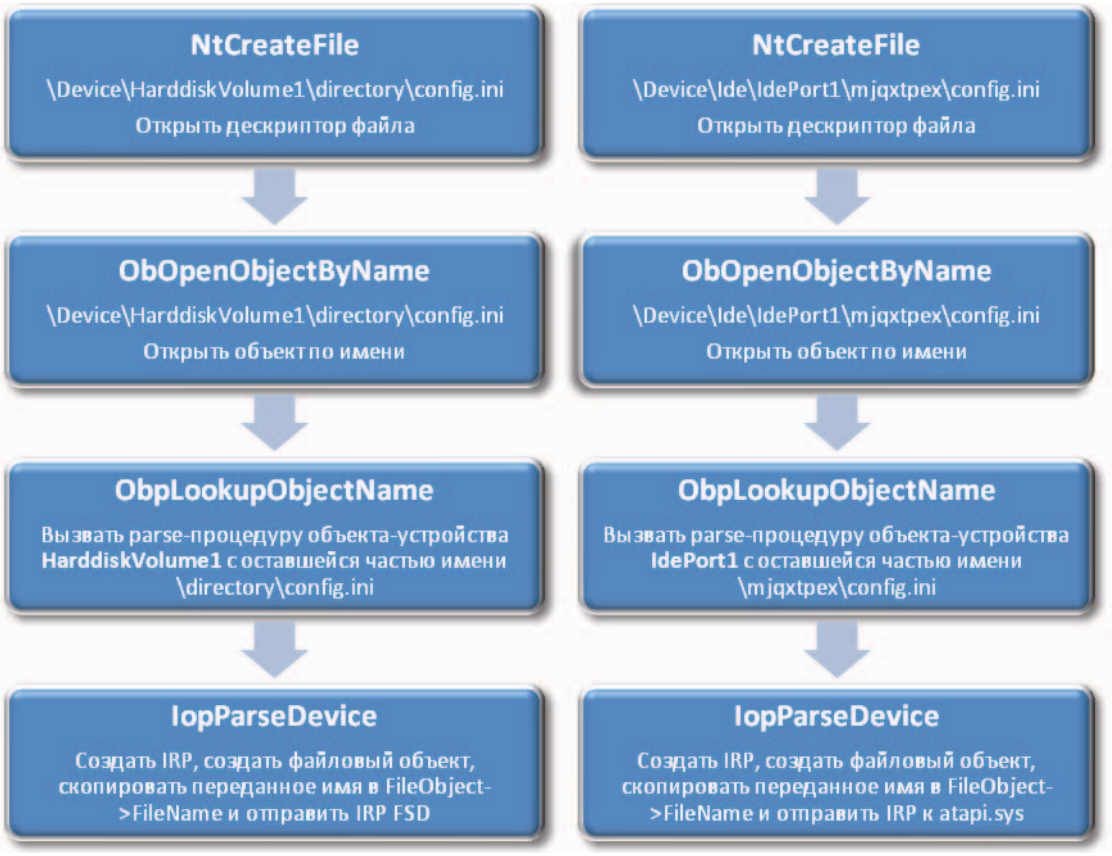


Рисунок 4. Открытие файла на обычном и скрытом диске

У руткита есть одна общая функция-обработчик (*диспатч-функция*), которая обслуживает все запросы, пришедшие как от клиентов **atapi**, так и от модулей режима пользователя (usermode). Таким образом, она выполняет две важные функции:

- скрывает от клиентов **atapi** данные в защищенной области, в конце диска, а при попытке чтения **atapi** с диска выдает клиенту оригинальный файл;

- как FSD, обслуживает операции типа **create/close/query information** для файлов в защищенной области, поступающих как от dll руткита в процессах, так и от самого руткита, который может запросить, например, чтение секции в config.ini.

Для подмены элементов в таблице указателей диспатч-функций руткит поступает следующим образом: находит конец первой секции файла **atapi.sys** в памяти и записывает в так называемый "**cave**" (пространство между фактически занятыми данными и размером всей секции) в конце секции следующий шаблон:

```
mov eax, ds:0FFDF0308h
jmp dword ptr [eax+0FCh],
```

что в некоторых случаях из-за отсутствия каких-либо проверок перезаписывает соседнюю секцию. Таким образом, перехваты по-прежнему ведут в образ **atapi.sys** (Рисунок 5), что сбивает с толку многие антируткиты, и они его попросту не замечают.

```
Dispatch routines:
[00] IRP_MJ_CREATE                      f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[01] IRP_MJ_CREATE_NAMED_PIPE           f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[02] IRP_MJ_CLOSE                       f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[03] IRP_MJ_READ                        f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[04] IRP_MJ_WRITE                       f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[05] IRP_MJ_QUERY_INFORMATION           f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[06] IRP_MJ_SET_INFORMATION             f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[07] IRP_MJ_QUERY_EA                    f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[08] IRP_MJ_SET_EA                      f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[09] IRP_MJ_FLUSH_BUFFERS               f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[0a] IRP_MJ_QUERY_VOLUME_INFORMATION    f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[0b] IRP_MJ_SET_VOLUME_INFORMATION      f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
[0c] IRP_MJ_DIRECTORY_CONTROL           f9756b3a    atapi!PortPassThroughZeroUnusedBuffers+0x34
```

Рисунок 5. Перехваты atapi.sys в Windows XP SP3

Руткит использует большую структуру, где хранит всю конфигурационную информацию, которая может понадобиться при осуществлении различных операций. Указатель на нее записывается по адресу **0xFFDF0308**, т.е. он использует часть **KUSER_SHARED_DATA**. Так, например, по смещению **+00FCh** расположен универсальный обработчик запросов, который и вызывается в приведенном выше примере (`jmp dword ptr [eax+0FCh]`). Там же хранятся структуры, описывающие, какие секторы и их части необходимо прятать и чем их подменять.

В случае если клиент **atapi** запрашивает данные из защищаемого руткитом раздела, тот просто обнуляет их или подменяет оригинальными. Рассмотрим псевдокод, описывающий его логику:

```
if( DeviceObject == ROOTKIT_PARAM_BLOCK. AtapiBootRootkitDevObj &&
   IoStack->MajorFunction == IRP_MJ_SCSI &&
   IoStack->Parameters.Scsi.Srb->Function == SRB_FUNCTION_EXECUTE_SCSI
)
{
        if( RequestedStartSector + cSectors >
ROOTKIT_PARAM_BLOCK.HideAreaStartSector)
        {
        if( IsRead )
```

```
        {
                подменить функцию завершения в текущей ячейке стека своей функцией
        }
        else if( IsWrite )
        {
                завершить операцию с ошибкой
        }
        else if( присутствует обращение к секции ресурсов atapi или oep, chksum,
        security data dir entry)
        {
                подменить функцию завершения в текущей ячейке стека своей функцией
        }
{
        И уже в функции завершения происходит подмена данных.
```

После выхода первых версий TDL3 некоторые разработчики антируткитов внесли соответствующие изменения для того, чтобы хотя бы детектировать присутствие руткита. Ответ вирусописателей не заставил себя долго ждать, и появились новые версии с принципиально новым и трудно детектируемым способом перехвата.

На этот раз таблица обработчиков зараженного драйвера остается чистой. Авторы руткита применили здесь интересное решение. Они просто «украли» у **atapi** объект-устройство, которое обслуживает необходимый им системный диск (Рисунок 6).

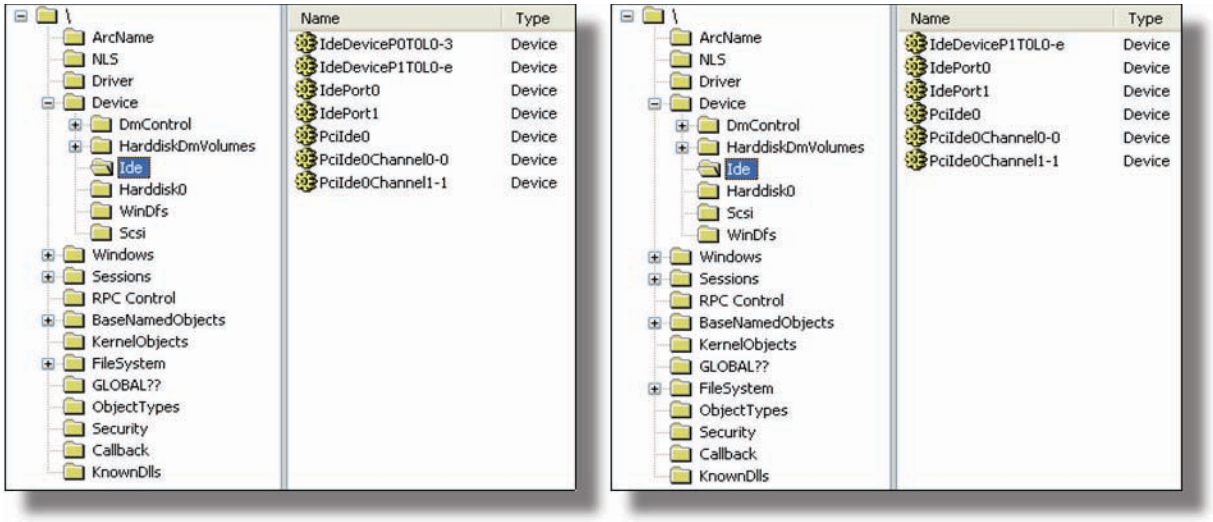


Рисунок 6. Чистая система (слева) и зараженная система (справа) с «пропавшим» устройством

И только в отладчике можно обнаружить аномалию (Рисунок 7) - неизвестное устройство, обслуживаемое неизвестным драйвером. Более того, заголовок DRIVER_OBJECT «неизвестного драйвера» испорчен, а сам он удален из общего системного списка драйверов (это касается и «украденного» устройства). Этот объект драйвера создан руткитом и прячет необходимые секторы, а так же обслуживает устройство для доступа к скрытому разделу. Он уже стал видимым, но для доступа, как и прежде, необходимо угадать (или найти) устройство, имя которого состоит из 8 случайных букв.

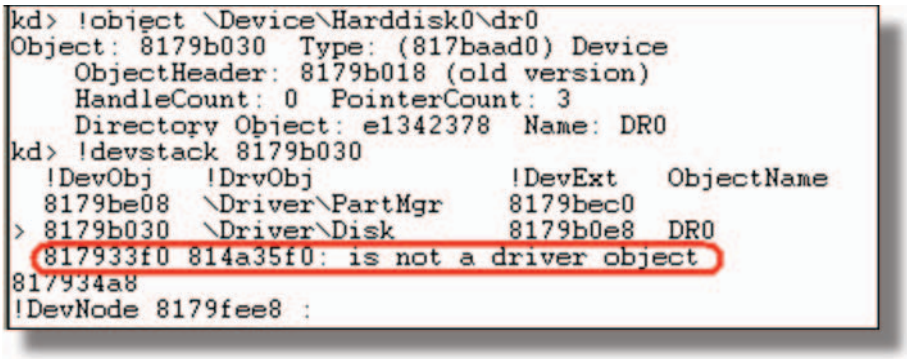


Рисунок 7. Обнаружение аномалии при помощи WinDbg

В связи с этим авторам антируткитов придется придумать новый способ, с помощью которого по заданному объекту-устройству можно будет найти настоящий драйвер, который это устройство обслуживает.

Также стоит обратить внимание на отладочный вывод, который делает руткит при старте, и любовь авторов к мультфильмам. Так, например, в отладчике может появиться одна из строк:

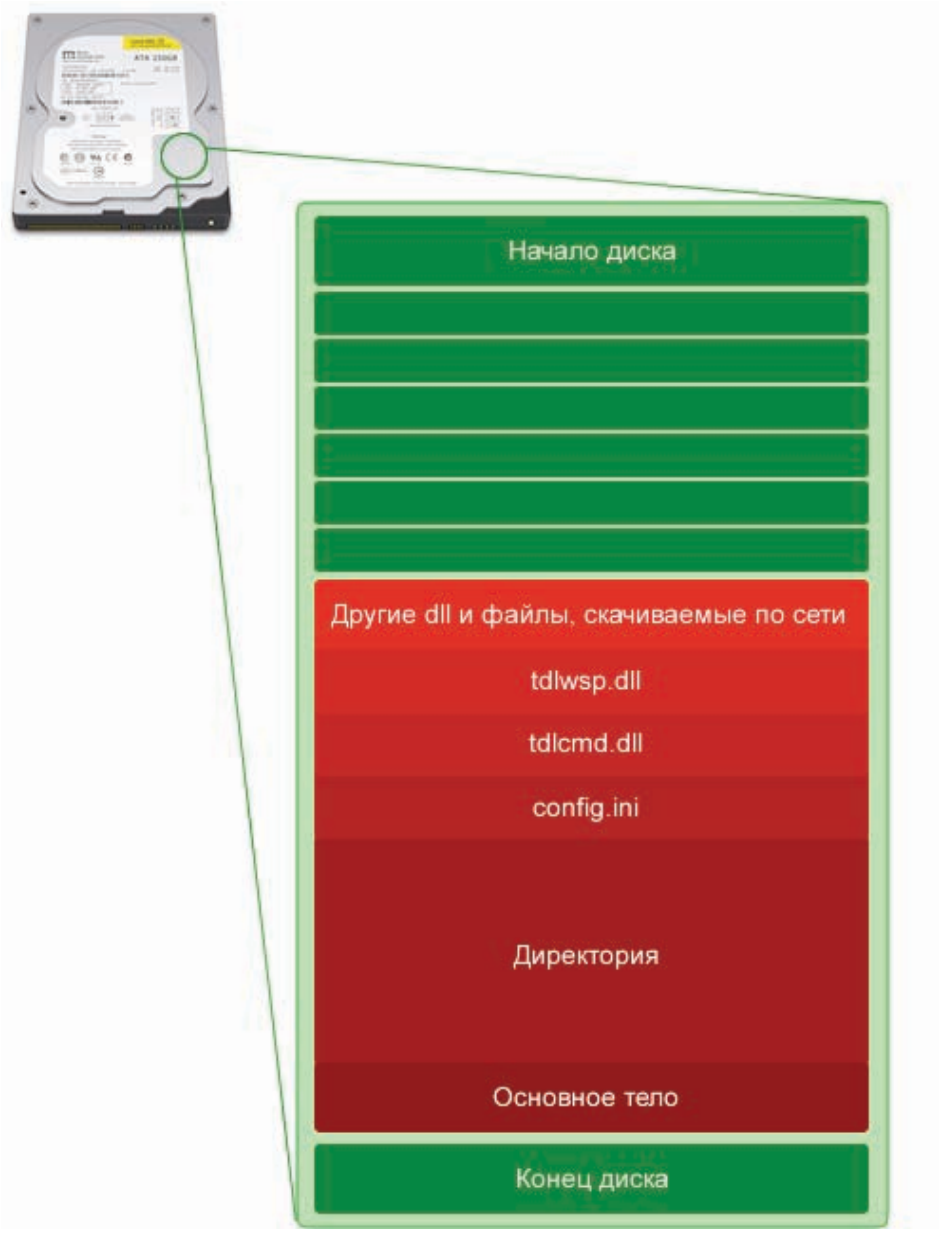
- Spider-Pig, Spider-Pig, does whatever a Spider-Pig does. Can he swing, from a web? No he can't, he's a pig. Look out! He is a Spider-Pig!
- This is your life, and it's ending one minute at a time
- The things you own end up owning you
- You are not your fucking khakis

Или в более поздних версиях:

- Alright Brain, you don't like me, and I don't like you. But lets just do this, and I can get back to killing you with beer
- I'm normally not a praying man, but if you're up there, please save me Superman.
- Dude, meet me in Montana XX00, Jesus (H. Christ)
- Jebus where are you? Homer calls Jebus!
- **TDL3 is not a new TDSS!**

Файловая система руткита

В конце диска руткит выделяет для себя область для хранения своего основного тела и виртуального диска. Структура физического диска в зараженной системе выглядит следующим образом:



Раздел виртуального диска растет от старших секторов к младшим, и внутренне руткит работает с отрицательными смещениями от сектора описателя виртуальной директории (Рисунок 8). Таким образом, расширяясь в обратную сторону, он способен затереть занятые секторы физического диска.

Хранимые в этом разделе файлы содержат одновременно и метаданные FS, и собственно сами данные файлов. Метаданные имеют размер 12 байт и представлены следующим форматом:

+00 Сигнатура [TDLD – для каталога, TDLF – для файлов, TDLN – для файла из сети]

+04 сквозной номер сектора с валидными данными

+08 размер данных, если они умещаются в сектор или если предыдущее поле не установлено в 0, смещение до следующего сектора файла, относительно данных файла (т. е. еще +0xC для метаданных, получается, что поле обычно содержит 0x3F4, 0x3F4 + 0xC = 0x400)

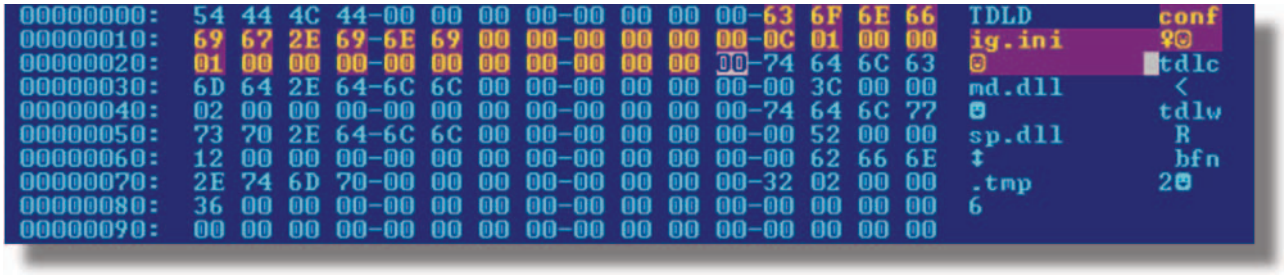


Рисунок 8. Описатель виртуальной директории BackDoor.Tdss.565

На рисунке 8 видны три файла, записанные на диск при инсталляции вируса (**config.ini**, **tdlcmd.dll** и **tdlwsp.dll**), а также временный **bfn.tmp,** загруженный из сети. Все секторы раздела зашифрованы при помощи алгоритма RC4. Этот же алгоритм используется и другими компонентами, не имеющими непосредственного отношения к работе виртуальной файловой системы. Так, например, упомянутый выше файл дополнительно зашифрован при помощи идентификатора бота, который хранится в config.ini и после расшифровки представляет собой набор команд для руткита (Рисунок 9).

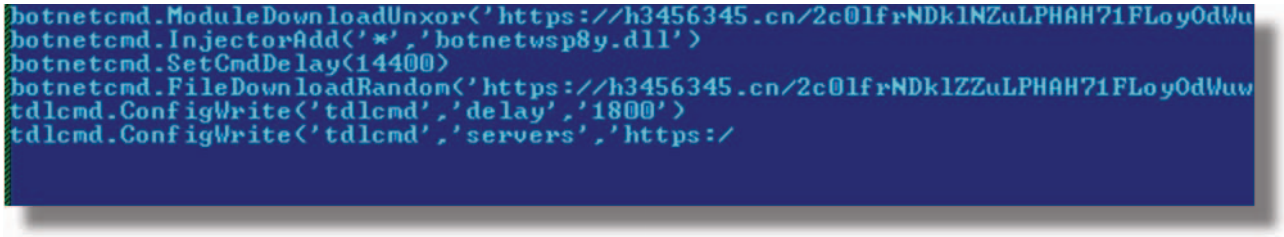


Рисунок 9. Содержимое файла bfn.tmp

На рисунке 10 представлен описатель директории BackDoor.Tdss.1030, на котором видны новые поля метаданных файлов, а также отдельные файлы для основного тела руткита (**tdl**) и оригинальных ресурсов зараженного файла (**rsrc.dat**):

Рисунок 10. Описатель виртуальной директории BackDoor.Tdss.1030

Каталог состоит из структуры метаданных с последующими записями о файлах. Размер одной записи 32 байта (на рисунке 8 запись выделена).

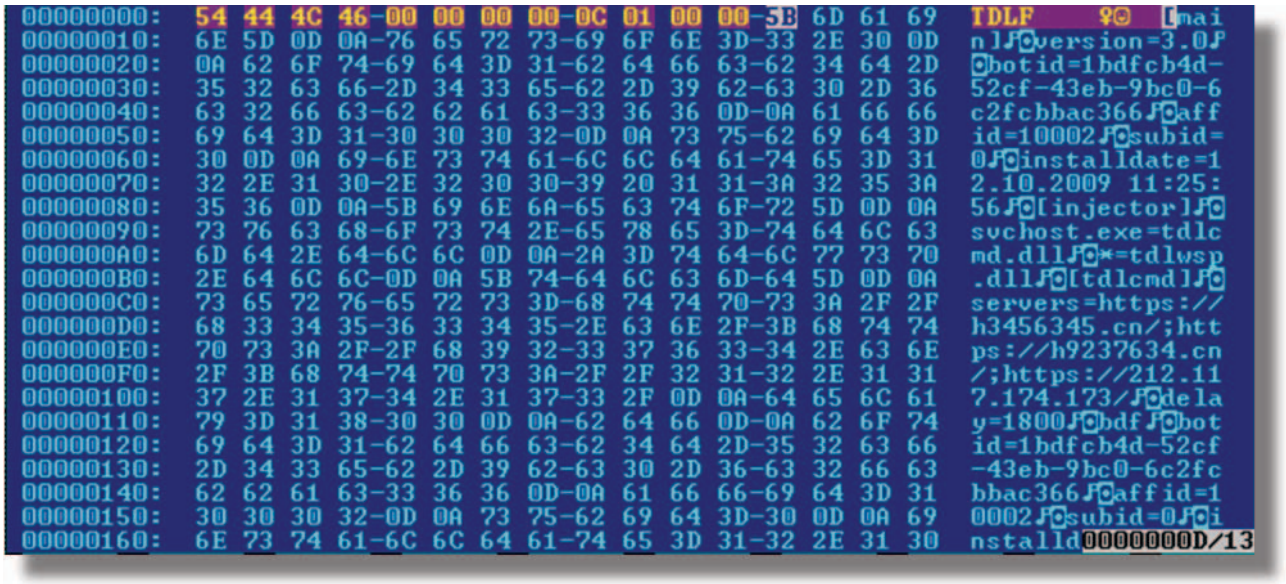


Рисунок 11. Описатель файла

Первые 12 байт описателя файла содержат метаданные, в начале которых идет сигнатура **TDLF** или **TDLN**, номер следующего сектора и размер**.** Например, на рисунке 11 размер файла - 0x10C байт.

Структура файловой системы руткита такова, что реальные данные файлов чередуются с “мусорными” секторами, т.к. он оперирует размером 0x400 байт (Рисунок 12) вместо 0x200 (для стандартной системы).

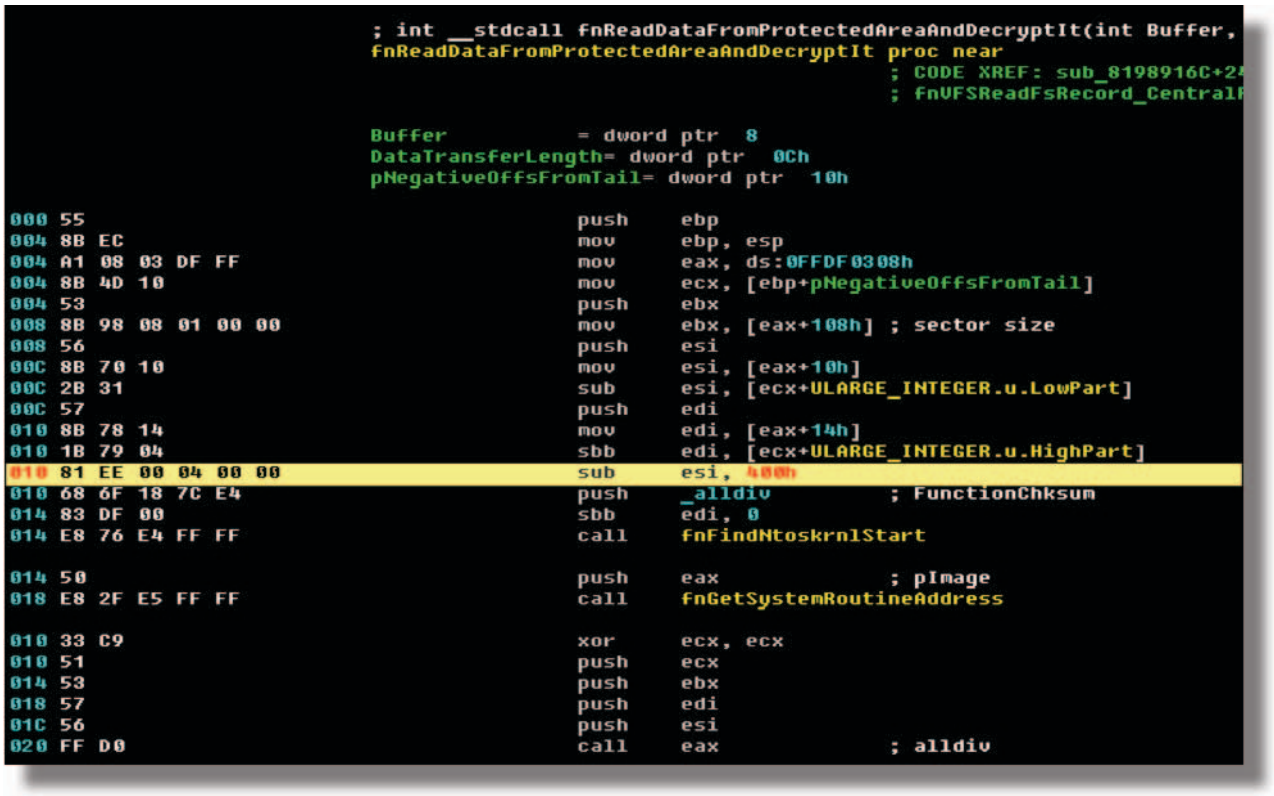


Рисунок 12. Функции чтения секторов виртуального раздела

Заключение

В целом новое поколение BackDoor.Tdss является весьма технологичным и интересным. Оно, бесспорно, поставило перед антивирусными компаниями сложную задачу детектирования и обезвреживания этого руткита. И не всем под силу справиться с этой задачей, как это уже бывало с BackDoor.MaosBoot, Win32.Ntldrbot и т.д.