

ROOTING ABOUT IN TDSS

*Aleksandr Matrosov, Eugene Rodionov*¹
ESET, Russia

Not so long ago one of our clients asked us to analyse a set of TDSS droppers, and to locate the source of the threat. As is described in a much lengthier report [1], we found evidence to suggest that a well-known cybercrime group was involved in the distribution of the rootkits.

The droppers were distributed using a pay-per-install (PPI) scheme that is already well known and gathering increasing popularity among cybercriminals. The PPI scheme is similar to those used for distributing toolbars for web browsers. If you are a partner distributing toolbars then you get a special build with an embedded identifier. This enables the number of installations for which you have been responsible to be calculated, and therefore also the calculation of the revenue due to you.

The same approach is used for distributing these rootkits: information about the distributor is embedded into the executable and special servers are used to calculate the number of installations.

EASY MONEY

The Dogma Millions cybercrime group started business in the autumn of 2009, placing a variety of advertisements on public forums offering 'easy money'. The group has a well-developed business infrastructure – from which many legitimate businesses could learn: for example, each affiliate is assigned a personal manager who can be consulted in case of any problems [2].

In order to reduce the likelihood of detection by anti-virus software, distributed malware is repacked every few hours (or even more frequently) and partners are specifically instructed *not* to check whether the malware can be detected by anti-virus products by using resources like *VirusTotal*. If these rules are violated, a partner may be fined. Usually, the cybercrime group uses all-too-reliable packers and protectors which ensure that the malware remains undetected by many anti-virus products.

ENCRYPTED FILE SYSTEM

One of the most interesting features of the rootkit is its file system, which is used to store its files and keep them hidden. The file system consists of:

- injectors (tdlcmd.dll)
- configuration information (config.ini)

¹With special thanks to David Harley for participating in this research.

- the rootkit body (tdl)
- overwritten resources of the infected file (rsrc.dat)
- additional files that are downloaded from the Internet.

We can see the layout of the file system in Figure 1.

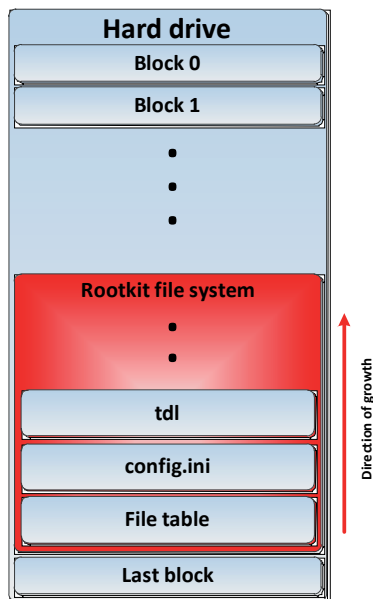


Figure 1: Rootkit file system layout.

The file system begins at the end of the disk, namely at the last logical block (sector), and grows towards the beginning of the disk. Thus, in theory it can overwrite users' data, if it grows large enough. It starts at the offset from the beginning of the disk which can be calculated with the following formula:

$$offset = (x - 1) * y$$

Here, x represents the total number of logical blocks on the disk, while y represents the size of the logical block (typically, the size of the logical block is 512 bytes). The file system of the rootkit is also divided into blocks. Each block has a size of 1,024 bytes. At the very beginning of the file system file there is a table which contains information about all the files stored in the file system. Each record in the table has the following format:

- file name (limited to 16 symbols);
- starting offset of the file from the beginning of the file system expressed in kilobytes (to get the actual offset of a file we need to subtract the starting offset multiplied by 1,024 from the offset of the beginning of the file system);
- size of the file;
- time of creation.

The structures that describe the file system are detailed in the next section.

FILE SYSTEM STRUCTURES

```
// Structure corresponding to file entry in the file table
typedef struct _TDL_FILE_TABLE_ENTRY
{
    char FileName[16];           // file name
    ULONG FileSize;             // size of the file
    ULONG FileOffset;           // offset of the file
                                // in kilobytes
    _int64 FileTime;            // time of creation
}TDL_FILE_TABLE_ENTRY, *PTDL_FILE_TABLE_ENTRY;

// Structure corresponding to block with file
typedef struct _TDL_FILE_OBJECT
{
    ULONG Signature;            // TDLF or TDLN if
                                // the block is free
    ULONG NextBlockOffset;     // offset to the
                                // next block with file data in kilobytes
    ULONG Reserved;
    UCHAR FileData[0x3F4];     // file data
}TDL_FILE_OBJECT, *PTDL_FILE_OBJECT;

//Structure corresponding to file table
typedef struct _TDL_FS_DIRECTORY
{
    ULONG Signature;           // TDLF
    ULONG NextBlockOffset;     // offset of the
                                // next block with file table if any
    ULONG Reserved;
    TDL_FILE_TABLE_ENTRY Files[0x1F]; // array of
                                // file entries in file table
}TDL_FS_DIRECTORY, *PTDL_FS_DIRECTORY;
```

Each block of the rootkit's file system has the following format:

- 0/3 bytes – signature:
 - TDLN – if the block contains file table information
 - TDLF – if the block contains a file
 - TDLN – if the block is free
- 4/7 – offset to the next block from the beginning of the file system expressed in kilobytes;
- 8/11 – size of the data;
- 12/1023 – data.

Figure 2 shows an example of the file table.

As we can see from Figure 2, the file system contains five files:

```

814AE79C 00 00 00 00 54 44 4C 44 00 00 00 00 00 00 00 00 .....TDL3.....
814AE7AC 63 6F 6E 66 69 67 2E 69 6E 69 00 00 00 00 00 00 config.ini.....
814AE7BC A1 02 00 00 01 00 00 00 B5 41 60 38 6C E0 CA 01 6.....;n 81p!
814AE7CC 74 64 6C 00 00 00 00 00 00 00 00 00 00 00 00 tdl.....
814AE7DC 9B 52 00 00 02 00 00 00 8A A4 62 38 6C E0 CA 01 WR.....Kab81p!
814AE7EC 72 73 72 63 2E 64 61 74 00 00 00 00 00 00 00 00 rsrc.dat.....
814AE7FC 93 03 00 00 17 00 00 00 82 E8 9B 38 6C E0 CA 01 U.....Buw81p!
814AE80C 74 64 6C 63 6D 64 2E 64 6C 6C 00 00 00 00 00 00 tdlcmd.dll.....
814AE81C 00 52 00 00 18 00 00 00 01 11 A3 38 6C E0 CA 01 R.....r81p!
814AE82C 00 78 61 79 2E 74 6D 70 00 00 00 00 00 00 00 00 .xay.tmp.....
814AE83C 00 52 00 00 2A 00 00 00 6D 24 56 DF 56 E1 CA 01 R.....n$U-Uc!
814AE84C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE85C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE86C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE87C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE88C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE89C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE8AC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE8BC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE8CC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE8DC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
814AE8EC 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figure 2: First block of the rootkit's file system.

- *tdl* – file containing the body of the rootkit
- *config.ini* – configuration file
- *rsrc.dat* – 915 (0x393) bytes of the overwritten resources of the infected driver
- *tdlcmd.dll* – the module that is injected into processes
- *?xay.tmp* – deleted temporary file.

Also, we can see that the file *config.ini* has a size of 0x2A1 bytes and starts at the next block (its offset is 1 kilobyte) of the file system.

Each block of the rootkit's file system is encrypted. In the latest version (3.273) the blocks are encrypted by XORing with a constant value (0x54) which is incremented at each XOR operation, while in the previous versions the RC4 cipher was used with the 'tdl' key.

In the course of our research into TDSS [1], from which this is a brief extract, we have developed a universal utility for dumping the rootkit's hidden file system. Our utility has worked correctly with all the samples with which we have been able to test to date, and is available from <http://www.eset.ru/viruslab/analytics/tdlfdumper.zip>.

This tool is useful for getting files stored into TDL3's encrypted file system (v. 2.23 and higher). It's used as follows:

Run the tool with the following parameters:

```
tfd.exe [-v] [directory_to_save_files]
```

-v for verbose output;

directory_to_save_files – specify the directory where content of the file system will be stored.

The tool requires administrative privileges (in order to load the driver). Here, just to give you a final flavour of how it looks, is an example of the sort of output you can expect using *tfd.exe*:

Output example: *tfd.exe*

Contents of TDL3 file system:

```

config.ini      MD5: C7562452A2D22E264CA936FD24169539
tdl             MD5: 19640E59F88B3EC86810F5CB92532A7F
rsrc.dat        MD5: EDF98E57B9A88A731FA016671C7E222
bckfg.tmp       MD5: 1BB9C4C278BAD9AEA26D36581679EC7E
tdlcmd.dll      MD5: 5250D03F8BA4337426AC928B64C10C2E

```

Output example: *tfd.exe -v*

Contents of TDL3 file system:

```

config.ini  Size: 505 bytes      MD5: C7562452A2D22E264
CA936FD24169539 Creation time: 24/05/2010 14:43:14

tdl        Size: 25159 bytes   MD5: 19640E59F88B3EC86
810F5CB92532A7F Creation time: 24/05/2010 14:43:14

rsrc.dat   Size: 917 bytes     MD5: EDF98E57B9A88A731
FA016671C7E222 Creation time: 24/05/2010 14:43:15

bckfg.tmp  Size: 191 bytes        MD5: 1BB9C4C278BAD9AEA2
6D36581679EC7E Creation time: 24/05/2010 14:43:15

tdlcmd.dll Size: 20480 bytes   MD5: 5250D03F8BA4337426
AC928B64C10C2E Creation time: 24/05/2010 14:43:15

```

config.ini:

[main]

quote=Tempers are wearing thin. Let's hope some robot doesn't kill everybody

version=3.273

botid=b79aea7d-ea32-4da4-bdd0-85af03bd91c7

affid=11418

subid=0

installdate=24.5.2010 14:43:16

builddate=8.4.2010 11:18:57

[injector]

*=tdlcmd.dll

[tdlcmd]

servers=https://873hg7xx60.com/;https://34jh7alm94.asia/;https://112.121.181.26/;https://61.61.20.132/

wspservers=http://lk0lha71gg1.cc/;http://z1091kha644.com/;http://91jjak4555j.com/

popupservers=http://zxcl9abnz72.com/

version=3.74

REFERENCES

- [1] Matrosov, A.; Rodionov, E. TDL3: The Rootkit Of All Evil? <http://www.eset.com/documentation/white-papers>.
- [2] Stevens, K. The Underground Economy of the Pay-Per-Install (PPI) Business. http://www.blackhat.com/presentations/bh-dc-10/Stevens_Kevin/BlackHat-DC-2010-Stevens-Underground-wp.pdf.