



What next? Trojan.Linkoptimizer

Mircea Ciubotariu
Symantec Security Response, Ireland

What next? Trojan.Linkoptimizer

Contents

The Trojan	4
The action	5
Spaghetti code	7
Reserved file names	8
Encrypting file system	9
The rootkit	10
Conclusion	12
References	12

Also known as Gromozon, the Linkoptimizer Trojan has created havoc within the antivirus industry. It has raised alarms signalling that there are other ingenious ways besides advanced rootkits to make the lives of both users and security providers a nightmare. Combining state-of-the-art techniques such as ‘spaghetti’ code, Encrypting File System (EFS), object rights manipulation, reserved file names, and user-mode rootkits, the Trojan manages gracefully to avoid detection by many antivirus engines and its removal can be a real nightmare.

The Trojan

Trojan.Linkoptimizer has pushed the limits of persistence and stealth to a point where it manages to evade antivirus detection most of the time.

Recently we have seen what a significant impact advanced rootkits can have on the antivirus industry, but in order to achieve a really good rootkit one has to go deeper into the system, making obscure undocumented changes, therefore introducing a greater risk of system instability. This means that such techniques can be applied only to a limited number of targets, and updates are required even for slight changes in the environment.

There are other ways to make code ‘stealthy’, some of which have already been discussed in various articles. But so far Linkoptimizer is the first to combine these techniques and add its own flavour to them by developing new ones.

Although there is a lot to be said about this complex threat, this article will focus on the big picture of the Trojan and the new elements it brings to the scene, especially its methods of evasion, how they work and how its authors have adapted them.

Linkoptimizer is a Trojan – in fact it is an army of Trojans, consisting mainly of droppers and downloaders with two ultimate purposes:

- To display advertisements.
- To dial premium-rate numbers.

The former has been identified as the sole purpose of the Trojan by various vendors, with the latter somehow having been neglected as it was not associated with the ‘adware’ component. This might be due to the fact that the two components responsible for these actions are dropped at different levels in the Linkoptimizer scheme. The adware component receives more attention since it is closer to the rootkit and EFS components.

The domain gromozon.com is considered to be responsible for spreading this Trojan, which also gives the initial name for the threat, Gromozon. Its registration date goes back to mid-February 2006, so it is expected that the first variants were released as early as March. As only some of its components

were submitted individually for analysis – mainly by retail users – it was only in June/July that it came to the attention of the anti-virus vendors and it took a while for all the pieces to be put together to get the full picture (see Figure 1).

The action

As shown in Figure 1 the Trojan gets into the victim’s computer while browsing malicious sites. So far it is known that these sites are only using old exploits (i.e. no zero-day exploits have been used) with a little social

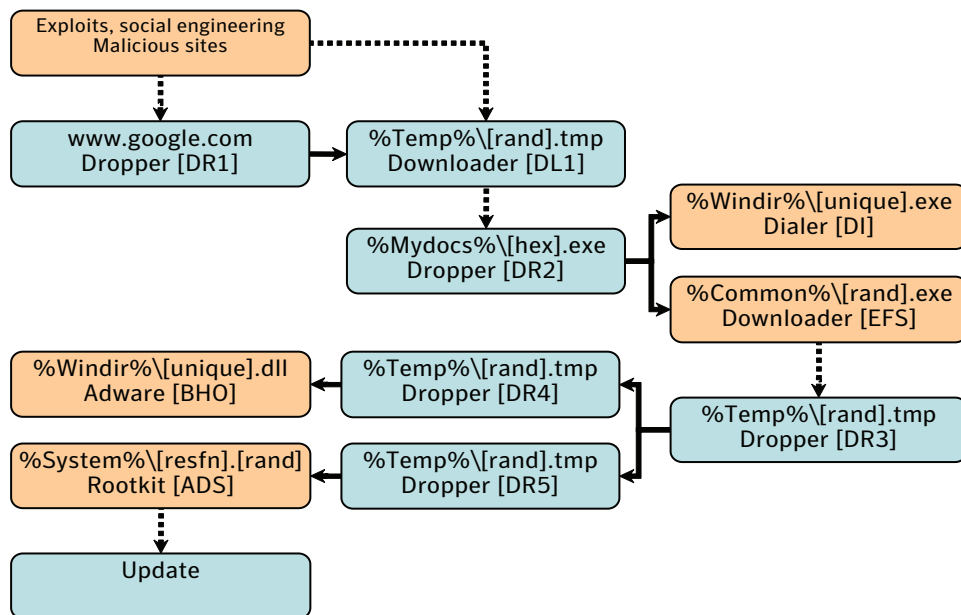


Figure 1: Trojan.Linkoptimizer scheme.

Component	Size (KB)	Packed	Spaghetti	Naming	Auto delete	Type
DR1	10 - 17	FSG,-	Yes	Social engineering	Yes	Exe
DL1	9 - 13	-	Yes	Temporary	Yes	DLL/OCX
DR2	61 - 67	UPX	Yes	6/8 random hex digits	Yes	Exe
DI (Dialer)	16 - 19	UPX, SUE	No	Generated machine unique	No	Exe
EFS	56 - 75	-	Yes	Reserved file names, other	No	Exe
DR3	156 - 193	UPX	Yes	Temporary	Yes	Exe
DR4	75 - 102	-	No	Temporary	Yes	Exe
BHO (Adware)	64 - 82	UPX + SUE	No	Generated machine unique	No	DLL/BHO
DR5	126 - 153	-	Yes	Temporary	Yes	Exe
ADS (Rootkit)	115 - 139	-	Yes	Reserved file names, other	No	DLL

engineering. This would make it hard to infect users with up-to-date patches and updates. However, two infection techniques have already been detailed^{1,2} and for the purpose of this paper, we will focus on the big picture.

The Trojan is set in motion once the first downloader, DL1, is executed on the computer through one of the two ways shown: it may be dropped and saved as www.google.com, or another seemingly innocuous name, or installed directly as an ActiveX Control (OCX file type) object, under the name ‘FreeAccess.ocx’. Once run, DL1 will attempt to download the following encrypted file:

```
shiptrop.com/1/pic.gif?<id_dec1>&<id_hex>&0
```

After decryption an executable is dropped into the current user's 'My Documents' folder. The name of the executable is made up of six or eight random hexadecimal digits (e.g. 3e22c2d.exe); this is the second dropper in the scheme, DR2. It will drop two other components: the dialer DI, and the EFS downloader.

The dialer executable is dropped into either the %Windir% or the %Windir%\Temp folder with the name generated pseudo-randomly so that it looks random, but it will always be the same on the same computer.

The dialer carries an .xml file that contains a long list of accounts, passwords and telephone numbers to be used when an active modem connection is detected in the system. Most of them are Italian premium-rate numbers starting with '899', but there are also some entries that use the Globalstar mobile satellite service, starting with '008819'.

Considering the authors decided to add the Globalstar satellite numbers just in case the compromised computer was outside Italy, it is worth noting that the number of countries from which these numbers can be dialed is limited because of the use of the prefix '00'. For example, from the USA and Canada the prefix '011' must be used to dial international numbers, so the prefix '00' will not work.

The EFS component will be discussed later, its main purpose being to download, decrypt and execute the following file:

```
shiptrop.com/2/pic2.gif?<id_dec2>& <id_hex>&0
```

This file is an intermediary dropper, DR3, which will drop two other droppers that we call DR4 and DR5. DR5 drops the rootkit component which is presented later on.

DR4 drops a DLL into the %Windir% folder. The name is similar to the dialer component and will be registered as a Browser Helper Object (BHO) using a pseudo-randomly generated CLSID that will remain the same for the same computer, so that it will be loaded automatically by Internet Explorer. The purpose of the BHO component is to display advertisements retrieved through this location:

```
wlow.net/common/hint_js.php?e=<request_enc64>
```

For quite a while the Trojan created an uninstall entry called 'Linkoptimizer' so that users could actually make use of the Add/Remove Programs function from the Control Panel, but the results would not be as expected. The following command would be executed:

```
iexplore.exe "http://notetol.com/uninstall.php"
```

This will render a page with only one small button in the middle reading 'Uninstall'. When the button is clicked it is replaced with a text box reading 'Thank you'. Needless to say, the threat is not touched at all, this being more of a practical joke meant to set a challenge in removing Linkoptimizer.

Recently, however, the Trojan's authors have decided to change the uninstall entry from 'Linkoptimizer' to a CLSID with various names, but still the same action.

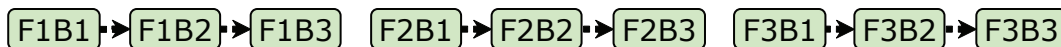
Upon execution, all of the components of the Trojan attempt to detect if the running environment is a virtual machine using the 'red pill' method³ and also check for the presence of the kernel mode debugger SoftICE. If either is detected they will simply exit.

Spaghetti code

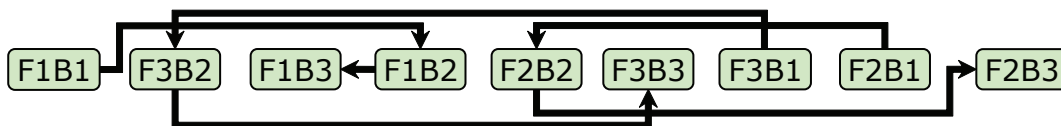
A common feature of Linkoptimizer is that it makes heavy use of spaghetti code⁴ in its binaries. This is most likely where the preferred name of the Trojan came from: an intermediary layer between the compiler and the linker that splits the code into many small blocks, then shuffles them and finally binds them together using jump instructions.

For example, let's consider a compiled binary having three consecutive functions, F1, F2 and F3, each of which is split into three basic code blocks, B1, B2 and B3.

The basic layout of the code right after compilation will be:



And one possible arrangement after the intermediary 'optimizer' layer might be:



From the source level – the creators' point of view – everything looks straightforward and meaningful, but what happens from the researcher's point of view? That's right, they only have the 'messy' code to deal with.

Previously, we have seen attempts to obfuscate code mainly at the source level by introducing extra garbage code – in most cases using macros that would generate different code each time and therefore different binaries. However Linkoptimizer has taken this a step further and mixed up the code with the following direct consequences:

- Analysis of such code is much slower since it is more difficult to follow.
- Automatic recognition of standard library functions

(e.g. IDA FLIRT signatures), as well as the recognition of the common code between different variants of the threat is no longer possible without further time-consuming processing.

- Detection signatures on this kind of code are not efficient at all; in most cases the detection would be limited to a single file only.

As will come as no surprise, this is combined with high-frequency updates of its components – about two per week – which ensures that the newly created binaries look different each time, and therefore they won't be detectable with regular definitions.

To make analysis of such code even more frustrating, the Trojan's authors have used two types of jump instructions in the binding process of the code blocks: direct jump instructions (opcode 0xe9/0xeb – jmp imm32/imm8) and indirect memory jumps (opcode 0xff, 0x25 – jmp [mem32]) where the target code block address is encoded in the data area.

All the spaghetti optimized binaries have their strings encrypted with RC4, using only one byte derived from the length of the string as the key. Decryption is performed only when the string is needed and only on the stack so that a memory dump of the module would not leak any useful information; and of course, each time the binary is compiled the string keys are changed.

Also, with the exception of a minimal set of APIs, the imports of such binaries are in some cases encrypted with RC4 as well. In other cases they are not stored at all in the file, in which case CRC32 hashes of the API names are stored.

Reserved file names

In order to maintain compatibility with certain DOS features, Windows operating systems reserve a number of file names which are used to access various physical devices. These are as follows:

Device file name	Description
AUX	Auxiliary device
COM1, COM2, ..., COM9	Serial ports
CON	Console device
LPT1, LPT2, ..., LPT9	Parallel ports
NUL	Null device
PRN	Printer device

Normally these file names cannot be used in conjunction with any extension, for example C:\LPT3.X would still refer to LPT3. However, 'LPT3X' is a valid file name and can be used to store data as any other file.

However with Windows NT, due to the limitations in the maximum path length, a new way of accessing files has been added, namely to use the '\\?\' prefix with a fully qualified file path. This not only increased the maximum allowed length of a path, but also allowed the use of the reserved device names in file or even folder names.

Two Linkoptimizer components, namely the EFS and the rootkit components, sometimes make use of this feature upon installation when naming their executables. The purpose is obvious: since most applications use the standard path names, they will be denied access to these files, therefore preventing curious eyes from seeing the contents of the files.

For example, the rootkit component may be dropped with the following name: '\\?\C:\Windows\System32\LPT1.IPJ', which will be treated as LPT1 unless the '\\?\' prefix is used.

Encrypting file system

Linkoptimizer effectively uses Windows Encrypting File System (EFS)⁵, shipped starting with Windows 2000, as an anti-antivirus technique. In doing so it creates a new administrator-equivalent account with a random name and random password. Then it copies itself with some garbage data appended into one of these locations:

- %CommonProgramFiles%\System
- %CommonProgramFiles%\Services
- %CommonProgramFiles%\Microsoft Shared
- %ProgramFiles%\Windows NT

Next, it encrypts the file through EFS and adds a new system service with a random name pointing to it, under the credentials of the newly created user. The service is instructed to run at boot time under the new account, so that no other common user or object has access to its contents.

Moreover, it manipulates the discretionary access control list (DACL) of the registry subkey that holds the service's details, as well as the executable's DACL, to permit access to these two objects only to its owner, which is set to be the new user. This means that if anyone – even the administrator – attempts to read or change the settings of the service or attempts to delete the file, he/she will be denied access.

However, there are means of getting access to the contents of the protected objects. In the case of registry subkeys, the system built-in account has access by default to any key of the hive. This means that, using the system credentials, one could read any protected information from the registry.

With the encrypted file, however, things are a bit different. One would need to make some changes in

the system in order to gain access to the decrypted contents of the file. In our case an easy solution would be to set the service's executable name to a different application that will run instead of the actual EFS and that will copy the contents of the EFS object to a new unencrypted container. At the end the service executable can be restored.

The implications of this are great: the biggest concern is not privacy invasion, since antivirus products may ask for permission before doing so, but access to the malicious contents, because there is no general way to get the contents of any EFS object.

The EFS component of Linkoptimizer goes a step further in preventing access to the file through the service's process, which contains the credentials of the user that it was run as. In doing so the service only runs for a small amount of time, just enough to create a new process, normally svchost.exe, inject its code into this process and perform all the subsequent actions with the appearance of an innocent and legitimate service.

Even for the small amount of time that the service runs, yet another protection measure has been introduced, namely to remove the SeDebugPrivilege from the administrator-equivalent, built-in, security group, which prevents any user in the system from manipulating processes or threads other than its own.

A consequence of this is that, without regaining SeDebugPrivilege, utility tools such as Filemon or Regmon will no longer run and some monitoring processes and information utilities will fail to work properly.

The rootkit

The rootkit component of Linkoptimizer is what attracted the most attention from the antivirus industry, although it does not bring anything new besides a certain persistence in staying up and running.

First, it is a user-mode rootkit that injects its own DLL into all running processes and hooks via patching 100 APIs from five libraries of the latest version. Initially the hooks were intended to cloak the rootkit and the BHO component, but over time protection mechanisms have been added.

One example is the blocking of known security tools. This is done in two steps. Firstly, by checking the name of the executable that is about to be run against a blacklist; and secondly, by checking the version information strings located in the executable's resources against another blacklist. When a string is found in the blacklist the execution request will simply be ignored. This way it effectively blocks many rootkit detectors and some fix tools provided by antivirus companies, which otherwise could remove or reveal some of its components.

It is noteworthy that the rootkit's cloak does not cover the dialer component. For some reason it was left outside, making it the most vulnerable component to detection and removal.

At the beginning the rootkit DLL was hidden in an Alternate Data Stream (ADS) of one of the following folders:

- System root folder (e.g. C:\kzpy.qmt)
- Windows folder (e.g. C:\Windows:hxlga.rbs)
- System folder (e.g. C:\Windows\System32:jicqr.nvd)

In time, the shelter provided by the ADS was no longer good enough since, although Alternate Data Streams can be used as normal files – can be executed, read and written – they don't have the same properties as normal files – for instance they don't have a security descriptor, a feature that would make them easier to remove.

In the end, the authors gave up using ADS for the rootkit storage and instead they started to use reserved file names or existing file names with a couple of characters changed. This time, however, they were able to use other tricks such as manipulating the file's security descriptor to allow execute permission only.

To ensure that the rootkit runs each time the system is restarted, it uses the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Windows\AppInit_DLLs registry value⁶, which instructs the system to load the DLLs enumerated by its string every time the user32 library initializes into a process. The trick with this value is that it is used even when the operating system boots in Safe Mode, with the exception of Windows 2003 Server.

In case AppInit_DLLs is not available the Trojan will use an alternate registry key to run at startup by creating a randomly named value containing the string 'rundll32 <rootkit_path>, <export_function>' into one of the following registry subkeys:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
```

The Windows API provides notification functions for registry subkeys and directories which can signal when the monitored subkeys or directories undertake any changes such as adding, deleting or changing objects and manipulating the security descriptor of any of the contained objects. The functions are FindFirstChangeNotification, FindNextChangeNotification for directories, and RegNotifyChangeKeyValue for subkeys.

The rootkit uses a notify function for the directory in which it is installed so that it monitors any

changes to its file. If the security descriptor is altered, or if the file attributes are changed or the file is deleted, it will revert any changes made. The same technique is used to monitor the startup registry subkey: if the value is removed or altered while the notification is active it will be restored.

Finally, it uses another registry subkey notification event, this time for HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager, in order to monitor the activity of the value 'PendingFileRenameOperations' which is responsible for renaming or deleting files at startup. The purpose of this is that if someone tries to delete the rootkit file using MoveFileEx, which uses that value, it will be able to replace the rootkit file name with random characters, preventing deletion in this way.

Conclusion

This isn't the work of an amateur. It may be one of those cases where a virus-writing teenager has refused to face reality and, as a grown up, has gone on to develop and improve his/her evil creatures with a new ultimate purpose: to make a living. It's no longer done for fun. But this is part of evolution and the antivirus industry must do more in response, since threats such as Linkoptimizer are at large, setting trends in malware development.

References

1. *Gromozon Evolution: From Spaghetti to Lasagna*. Elia Florio, Security Response Weblog.
http://www.symantec.com/enterprise/security_response/weblog/2006/10/gromozon_reloaded_everything_t.html
2. *GROMOZON.COM: The strange case of Dr.Rootkit and Mr.Adware*. Marco Giuliani, PrevX Virus Researcher.
<http://pcalsicuro.phpsoft.it/gromozon.pdf>
3. *Red Pill... or how to detect VMM using (almost) one CPU instruction*. Joanna Rutkowska.
<http://invisiblethings.org/papers/redpill.html>
4. *Spaghetti code*. Wikipedia.
http://en.wikipedia.org/wiki/Spaghetti_code
5. *Encrypting File System*. Wikipedia.
http://en.wikipedia.org/wiki/Encrypting_File_System
6. *Working with the AppInit_DLLs registry value*. Microsoft.
<http://support.microsoft.com/kb/197571>

About Symantec

Symantec is the global leader in information security, providing a broad range of software, appliances, and services designed to help individuals, small and mid-sized businesses, and large enterprises secure and manage their IT infrastructure.

Symantec's Norton™ brand of products is the worldwide leader in consumer security and problem-solving solutions.

Headquartered in Cupertino, California, Symantec has operations in 35 countries.

More information is available at www.symantec.com.

Symantec has worldwide operations in 35 countries. For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 800 745 6054.

Symantec Corporation
World Headquarters
20330 Stevens Creek Boulevard
Cupertino, CA 95014 USA
408 517 8000
800 721 3934
www.symantec.com

Symantec and the Symantec logo are U.S. registered trademarks of Symantec Corporation. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other brand and product names are trademarks of their respective holder(s). Any technical information that is made available by Symantec Corporation is the copyrighted work of Symantec Corporation and is owned by Symantec Corporation. NO WARRANTY. The technical information is being delivered to you as-is and Symantec Corporation makes no warranty as to its accuracy or use. Any use of the technical documentation or the information contained herein is at the risk of the user. Copyright © 2007 Symantec Corporation. All rights reserved. 04/05 10406630