New self-protecting USB trojan able to avoid detection

By Tomáš Gardoň

A unique data-stealing trojan has been spotted on USB devices in the wild – and it is different from typical data-stealing malware, reports ESET's Tomáš Gardoň.

23 Mar 2016 • , 5 min. read

A unique data-stealing trojan has been spotted on USB devices in the wild – and it is different from typical data-stealing malware. Each instance of this trojan relies on the particular USB device on which it is installed and it leaves no evidence on the compromised system. Moreover, it uses a very special mechanism to protect itself from being reproduced or copied, which makes it even harder to detect.

In this article we will examine the technical details of this interesting malware.

> ”What really sets this malware apart is its self-protection mechanism.”

Where other malware uses ‘good old-fashioned approaches’ like Autorun files or crafted shortcuts in order to get users to run it, USB Thief uses also another technique. This method depends on the increasingly common practice of storing portable versions of popular applications such as Firefox, NotePad++ and TrueCrypt on USB drives.

The malware takes advantage of this trend by inserting itself into the command chain of such applications, in the form of a plugin or a dynamically linked library (DLL). And therefore, whenever such an application is executed, the malware will also be run in the background.

What really sets this malware apart, however, is its self-protection mechanism.

The protection mechanism

The malware consists of six files. Four of them are executables and the other two contain configuration data. To protect itself from copying or reverse engineering, the malware uses two techniques. Firstly, some of the individual files are AES128-encrypted; secondly, their filenames are generated from cryptographic elements.

The AES encryption key is computed from the unique USB device ID, and certain disk properties of the USB drive hosting the malware. Hence, the malware can only run successfully from that particular USB device.

The name of the next file in malware execution chain is based on actual file content and its creation time. It is the first five bytes of SHA512 hash computed from mentioned attributes (file content concatenated with eight bytes of the creation time).

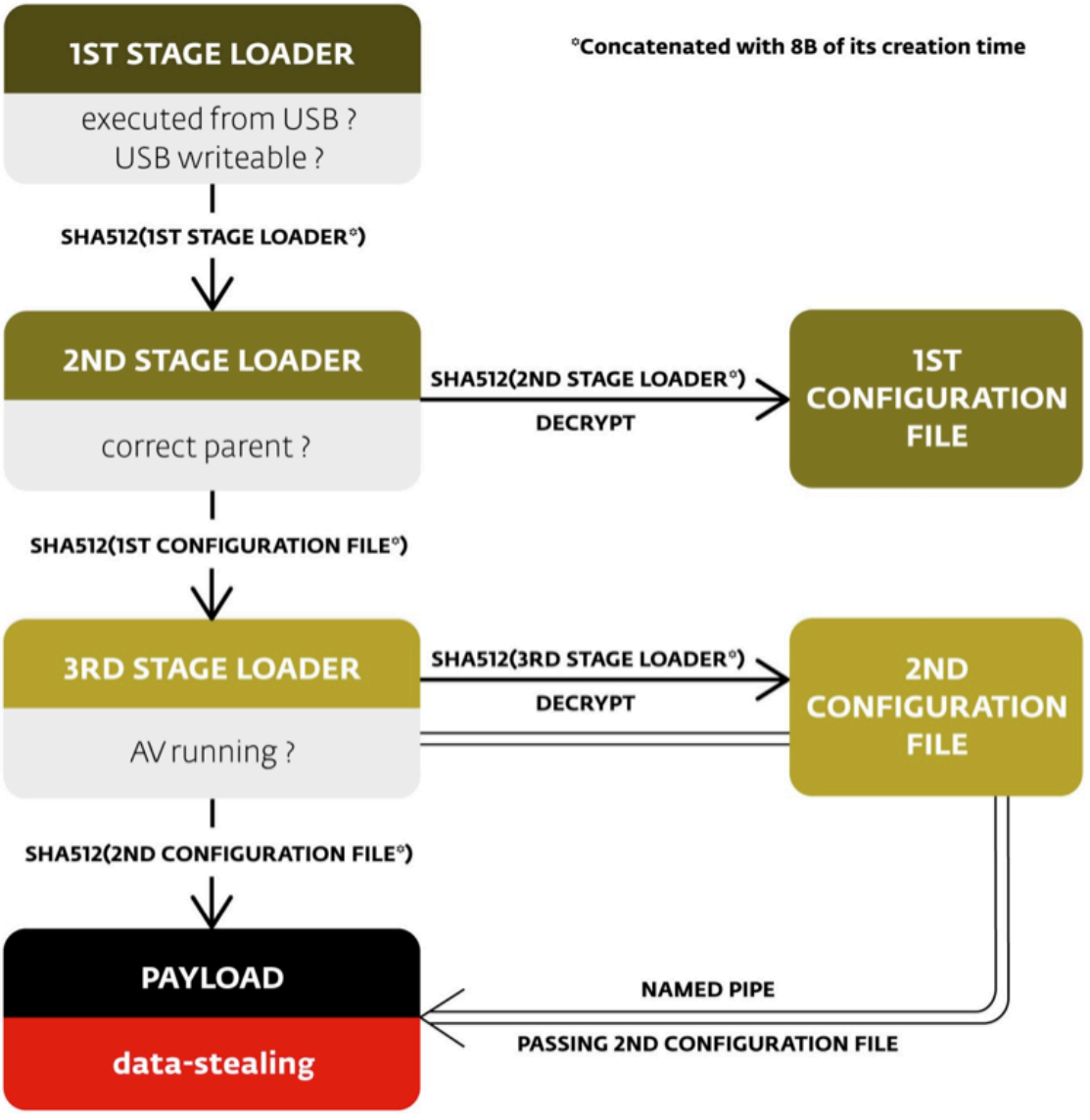
Because of this, filenames are different for every instance of this malware. Moreover, copying malware to a different place will replace the file creation time so that malicious actions associated with the previous locality

cannot be reproduced. For a better understanding of the naming technique, please see the image below.

It was quite challenging to analyze this malware because we had no access to any malicious USB device. Moreover, we had no dropper, so we could not create a suitably afflicted USB drive under controlled conditions for further analysis.

Only the submitted files can be analyzed, so the unique device ID had to be brute-forced and combined with common USB disk properties. Moreover, after successful decryption of the malware files, we had to find out the right order of the executables and configuration files, because the file copying process to get the samples to us had changed the file creation timestamp on the samples.

The execution flow of malware is quite simple. Each loader, in turn, loads and executes the following loader identified by computed hash according to the naming technique described above. However the execution must always start with the first stage loader, otherwise the malware terminates itself.



First stage loader

The first stage loader is just the malware's starting point and its main goal is to trick the user into running it. This can be done in several ways, but the most interesting is the use of portable applications. We have seen portable Notepad++ compromised by a malicious plugin as well as a TrueCrypt portable compromised by a malicious "RichEd20.dll". This loader also checks whether it is executed from a USB device and whether it is writeable, which is important because the payload will store stolen data here.

Second stage loader

The second stage loader is located using the first stage hash. Subsequently, its configuration file is found using its own hash. The configuration file contains the encrypted name of the parent process to be verified. This is an anti-debugging trick, which will cause termination of the malware if it is running under a different parent process, i.e. a debugger. Finally, the hash of the configuration file is used to compute the name of the third stage loader.

Third stage loader

The third stage loader handles some anti-AV checks. If one of the processes running is "avpui.exe" (Kaspersky security software) or "AVKTray.exe" (G Data security software) is running, execution is stopped.

Its configuration file is found by same technique as used by its predecessor, as is the payload executable. It also creates a named pipe to be used to pass the configuration data to the payload. The pipe name consists of the first 30 bytes of a SHA512 hash computed from the computer name.

Payload

Finally, the payload implements the actual data-stealing functionality. The executable is injected into a newly created "%windir%\system32\svchost.exe -k netsvcs" process. Configuration data includes information on what data should be gathered, how they should be encrypted, and where they should be stored.

The output destination must always be on the same removable device. In the case we analyzed, it was configured to steal all data files such as images or documents, the whole windows registry tree (HKCU), file lists from all of the drives, and information gathered using an imported open-source application called "WinAudit". It encrypts the stolen data using elliptic curve cryptography.

Conclusion

> "It would not be difficult to redesign the malware to change from a data-stealing payload to any other malicious payload."

In addition to the interesting concept of self-protecting multi-stage malware, the (relatively simple) data-stealing payload is very powerful, especially since it does not leave any evidence on the affected computer. After the USB is removed, nobody can find out that data was stolen.Also, it would not be difficult to redesign the malware to change from a data-stealing payload to any other malicious payload.

As ESET’s statistics shows, that malware is not very widespread. However, it possesses the ability to be used in targeted attacks - especially at computers that are not connected to the internet for security reasons.

Our detection name:

payload: Win32/PSW.Stealer.NAI trojan
loaders : Win32/TrojanDropper.Agent.RFT trojan

SHA1 hashes of decrypted binaries:

2C188C395AB32EAA00E6B7AA031632248FF38B2E
B03ABE820C0517CCEF98BC1785B7FD4CDF958278
66D169E1E503725A720D903E1DFAF456DB172767
4B2C60D77915C5695EC9D3C4364E6CD6946BD33C
76471B0F34ABB3C2530A16F39E10E4478CB6816D
