VB2020 localhost

30 September - 2 October, 2020 / vblocalhost.com

COMPFUN AUTHORS SPOOF VISA APPLICATION WITH HTTP STATUS-BASED TROJAN

Denis Legezo

Kaspersky, Russia

denis.legezo@kaspersky.com

ABSTRACT

From a researcher's point of view, it's always an exciting bonus when you uncover some really new and unusual techniques in the malware you are analysing. During 2019, one of actors that gave us some interesting puzzles to solve was the author of COMpfun. The malware was initially documented by *G DATA* in 2014 – although *G DATA* didn't identify which actor was using the malware – and we tentatively linked it to the Turla APT, based on the victimology.

In Autumn 2019, we covered one case of custom malware that was designed to compromise TLS-encrypted communications used in the HTTPS protocol [1]. Via a combination of installing digital certificates on the target's browsers and manipulating the TLS handshake to their own schema, the malware operators were able to distinguish the target's traffic – even after NAT routing – and decrypt it. To mark and distinguish the target's traffic, the developers came up with their own technically ingenious mechanisms – by patching the system's PRNG functions.

At the very end of 2019, we found another sample aimed at diplomatic entities, this time pretending to be a visa-related application on a LAN shared directory. These files with strong code similarities showed us that, with the same code base, developers can solve very different problems. This time, the code didn't manipulate TLS traffic at all. These newer samples used rare HTTP statuses (422-429) as C2 commands, targeting beacon C2s with a specific ETag and waiting for C2 response HTTP 402 (payment required) to proceed all the commands. The authors also solved the problem of spreading the malware to attached USB devices.

The method of injecting malware into the memory of system processes is also worth a mention. The necessary API functions addressed in this case were transmitted as parameters and as a result code was injected by itself (i.e. dumped from memory) that could barely be analysed without this additional data. Back in 2014, COMpfun developers were creative and potent in terms of their persistence – attributes which they still possess today.

INTRODUCTION

In autumn 2019 we presented a paper [2] at VB2019, detailing how a COMpfun successor known as Reductor infected files on the fly to compromise TLS traffic. Later in November 2019 our Attribution Engine revealed a new trojan with strong code similarities. Further research showed that it was obviously using the same code base as COMpfun.

WHAT'S OF THE MOST INTEREST INSIDE

The campaign operators retained their focus on diplomatic entities, this time in Europe, and spread the initial dropper as a spoofed visa application. It is not clear to us exactly how the malicious code is delivered to a target. The legitimate application is kept encrypted inside the dropper, along with the 32- and 64-bit next stage malware.

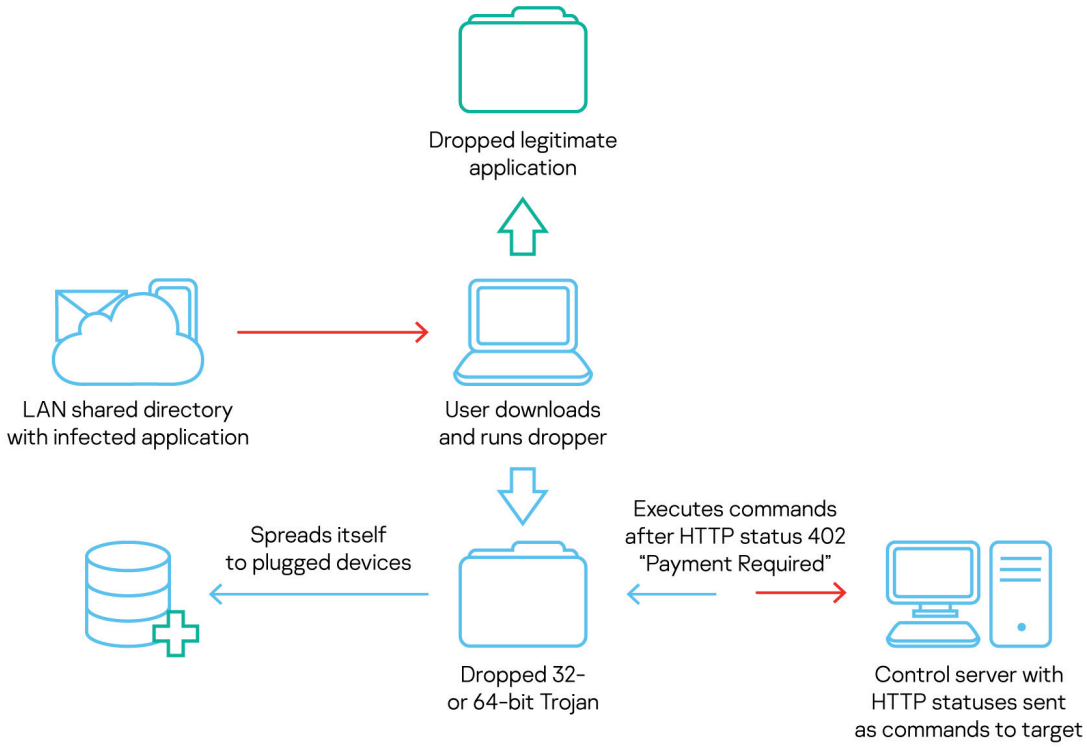

Figure 1: Overall infection chain. Interestingly, C2 commands are rare HTTP status codes.

We observed an interesting C2 communication protocol utilizing rare HTTP/HTTPS status codes (check IETF RFC 7231, 6585, 4918). Several HTTP status codes (422-429) from the Client Error class let the trojan know what the operators want to do. After the control server sends the status 'Payment Required' (402), all these previously received commands are executed.

The authors keep the RSA public key and unique HTTP ETag in encrypted configuration data. Created for web content caching reasons, this marker could also be used to filter unwanted requests to the C2, e.g. those that are from network scanners rather than targets. Besides the aforementioned RSA public key to communicate with the C2, the malware also uses a self-generated AES-128 key.

WHO IS THE AUTHOR?

We should mention here once again that the COMpfun malware was initially documented by *G DATA* in 2014, although the company did not identify which APT was using the malware. Based mostly on victimology, we were able to associate it with the Turla APT with a medium-to-low level of confidence.

WHAT THE TROJAN IS ABLE TO DO

The trojan's functions include the ability to acquire the target's geolocation, the gathering of host- and network-related data, keylogging and screenshots. In other words, it's a normal fully fledged trojan that is also capable of propagating itself to removable devices.

As in previous malware from the same authors, all the necessary function addresses resolve dynamically to complicate analysis. To exfiltrate the target's data to the C2 over HTTP/HTTPS, the malware uses RSA encryption. To hide data locally, the trojan implements LZNT1 compression and one-byte XOR encryption.

Encrypted data	Algorithm	Key source
Exfiltrated keystrokes, screenshots, etc.	RSA	Public key from configuration data
Configuration data in .rsrc section	XOR (plus LZNT1 compression)	Hard-coded one-byte key
Parameters inside the HTTP GET/ POST requests	AES-128 (plus ETag from config)	Generated by trojan and shared in beacon
Commands and arguments from C2 for HTTP status 427 (dir, upl, usb, net)	AES-128	Generated by trojan and shared in beacon

Encryption and compression used by the trojan for various tasks

Initial dropper

The first stage dropper was downloaded from the LAN shared directory. The file name related to the visa application process corresponds perfectly with the targeted diplomatic entities. As with all modules with a similar code base, the dropper begins by dynamically resolving all the required *Windows* API function addresses and puts them into structures. It then decrypts the next stage malware from its resource (.rsrc) section. The algorithm used to decrypt the next stage is a one-byte XOR using the key '0x55', followed by LZNT1 decompression.

The following files are dropped to the disk in addition to the original application that the malware tries to mimic:

MD5 hash	File name	Features
1BB03CBAD293CA9EE3DDCE6F054FC325	ieframe.dll.mui	64-bit trojan version
A6AFA05CBD04E9AF256D278E5B5AD050	ExplorerFrame.dll.mui	32-bit trojan version

The dropper urges users to run the file as administrator (using messages such as 'need to run as admin'), then drops a version corresponding to the host's architecture and sets the file system timestamp to 2013.12.20 22:31.

Interestingly, the dropper's abilities aren't limited to PE lures; as an alternative, this stage is also able to use .doc and .pdf files. In such cases, the dropper will open the files using the 'open' shell command instead of running the legitimate spoofed executable application.

Main module – HTTP status-based trojan

SHA256	710b0fafe5fd7b3d817cf5c22002e46e2a22470cf3894eb619f805d43759b5a3
MD5	a6afa05cbd04e9af256d278e5b5ad050
Compiled	2015.06.26 09:42:27 (GMT)
Type	I386 Windows GUI DLL
Size	593408
Internal name	ExplorerFrame.dll.mui

The analysis below is based on the 32-bit sample from the table above. The legitimate ExplorerFrame.dll.mui is a language resource for the ExplorerFrame.dll file used by *Windows Explorer*.

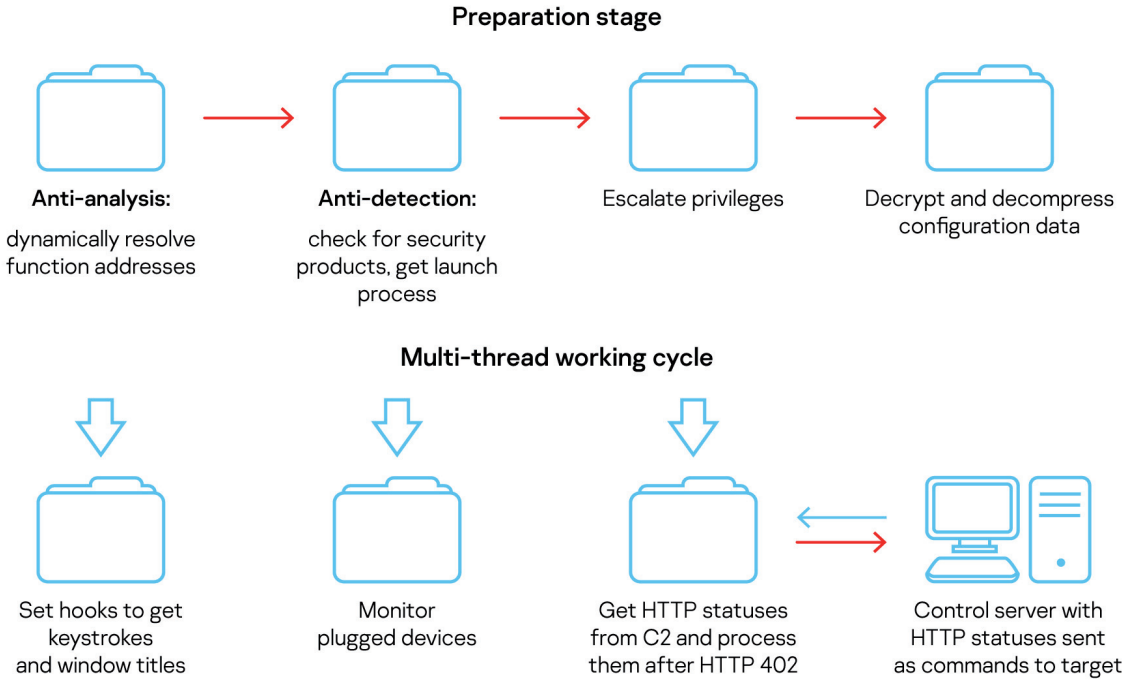

Figure 2: Multi-threaded trojan features such as monitoring USB devices to spread further and receiving commands as HTTP status codes.

Initialization

As usual in this malware family's code, a huge number of short standalone functions return all the readable strings. This is done to complicate analysis by not allowing the strings to be visible to researchers at a glance. The module's preparation stage dynamically resolves all required *Windows* API function addresses into corresponding custom structures. Afterwards the malware uses indirect function calls only.

The module obtains the processor architecture (32- or 64-bit) and *Windows* OS version. It includes a number of anti-analysis checks for virtual machine related devices (VEN_VMWARE, VBOX_HARDDISK, Virtual_DVD_ROM, etc.) to avoid controlled execution. It also notes which security products are running on the host (*Symantec*, *Kaspersky*, *Dr.Web*, *Avast*).

Before every communication with the C2, the malware checks if software such as debuggers (*WinDbg*, *OllyDbg*, *Visual Studio*) and host monitoring (*Process Explorer* or *Monitor*, etc.) or network monitoring (*Wireshark*, *TCPView*, etc.) programs are running. It also checks for Internet connectivity and does not attempt to communicate if the checks fail.

The DLL also checks for potentially available launch processes that it can inject itself into. In the case of PaymentRequired, this could be system, security product or browser processes. Then the malware forms the corresponding code to drop files, delete files, etc.

The last step in the initialization procedure is to decrypt and decompress the configuration file. Decryption is done via a one-byte XOR using the 0xAA key, followed by decompression using the LZNT1 algorithm. From the configuration, the malware parses the RSA public key, ETag and IP addresses to communicate with its control servers.

Figure 3: Decrypted configuration data contains an RSA public key to encrypt exfiltrated data, C2 IPs and unique ETag to communicate with them.

HTTP status-based communication module

First, the module generates the following:

- An AES-128 encryption key used in HTTP GET/POST parameters and HTTP status code 427 (request new command)
- A four-byte unique hardware ID (HWID) based on the host network adapters, CPU and first fixed logical drive serial number.

The module then chooses a process into which to inject the code, in order of decreasing priority, starting from *Windows* (cmd.exe, smss.exe), security-related applications (*Symantec*'s nis.exe, *Dr.Web*'s spideragent.exe) and browsers (*IE*, *Opera*, *Firefox*, *Yandex* browser, *Chrome*).

The main thread checks if the C2 supports TLS in its configuration. If it does, communication will be over HTTPS and port 443; otherwise, the HTTP protocol and port 80 are used.

Config parameter	**Value**
Encryption key	RSA public key on the image above
ETag	C8E9CEAD2E084F58A94AEDC14D423E1A
C2 IPs	95.183.49[.]10 95.183.49[.]29 200.63.45[.]35

Decrypted configuration content inside the analysed sample.

The first GET request sent contains an ETag 'If-Match' header that is built using data from its decrypted configuration. ETags are normally used by web servers for caching purposes in order to be more efficient and save bandwidth by not resending redundant information if an ETag value matches. The implementation of ETags means the C2 may ignore all requests that are not sent from its intended targets if they don't have the required ETag value.

HTTP status	RFC status meaning	Corresponding command functionality
200	OK	Send collected target data to C2 with current tickcount.
402	Payment required	This status is the signal to process received (and stored in binary flag) HTTP statuses as commands.
422	Unprocessable entity (WebDAV)	Uninstall. Delete COM-hijacking persistence and corresponding files on disk.
423	Locked (WebDAV)	Install. Create COM-hijacking persistence and drop corresponding files to disk.
424	Failed dependency (WebDAV)	Fingerprint target. Send host, network and geolocation data.
427	Undefined HTTP status	Get new command into IEA94E3.tmp file in %TEMP%, decrypt and execute appended command.
428	Precondition required	Propagate self to USB devices on target.
429	Too many requests	Enumerate network resources on target.

C2 HTTP status code descriptions, including installation, USB propagation, fingerprinting, etc.

HTTP 427 can receive any of the following appended commands:

Command	Command functionality
dir	Send directory content to C2 encrypted with RSA public key from config.
upl	Send file to C2 encrypted with RSA public key from config.
usb	Not implemented yet. Possibly same function planned as for HTTP status 428.
net	Not implemented yet. Possibly same function planned as for HTTP status 429.

Removable device propagation module

If initialization is successful, the malware starts one more thread for dispatching *Windows* messages, looking for removable devices related to a WM_DEVICECHANGE event. The module runs its own handlers in the event of a USB device being plugged into or unplugged from the host.

Other spying modules: keylogger, screenshot tool and more

The user's activity is monitored using several hooks. All of them gather the target's data independently of any C2 command. Keystrokes are encrypted using the RSA public key stored in the configuration data and sent once every two seconds, or when more than 512 bytes are recorded. These 512 characters also include left mouse button clicks (written as the 'MSLBTN' string) and *Windows* title bar texts. For clipboard content, the module calculates an MD5 hash and if it changes, encrypts the clipboard content with the same RSA public key and then sends it.

In a separate thread, the trojan takes a bitmap screenshot using the GDIPlus library, compresses it with the LZNT1 algorithm, encrypts it using the key from the configuration data and sends it to the control server. A screenshot will be taken of the target and sent anyway, independently of any C2 command.

Last but not least

There are several choices – albeit not major additional technical ones – that the malware author made which we consider to be noteworthy.

The COM-hijacking-based persistence method injects its corresponding code and structure as a parameter into the memory of a legitimate process. The malware geolocates victims using legitimate web services: geoplugin.net/json.gp, ip-api.com/json and telize.com/geoip.

The unusual thread synchronization timeout calculation in the HTTP status thread is peculiar. Mathematically, the partial sum of the series is precisely:

$$\text{timeout} = \sum_{n=0}^{19} \frac{a^n}{n!}$$

This series, in the case of a full sum, is just a representation of the exponent. The developers probably used the exponent to make timeouts in the communication thread more unpredictable and grow at a fast rate (the normal approach in high-loaded backend systems), and the compiler calculated it this way.

SO WHAT DID THE COMPFUN AUTHORS ACHIEVE?

We saw innovative approaches from the COMpfun developers twice in 2019. First, they bypassed TLS encrypted traffic via PRNG system function patching [1], and then we observed a unique implementation of C2 communications using uncommon HTTP status codes.

The malware operators retained their focus on diplomatic entities and the choice of a visa-related application – stored on a directory shared within the local network – as the initial infection vector worked in their favour. The combination of a tailored approach to their targets and the ability to generate and execute their ideas certainly makes the developers behind COMpfun a strong offensive team.

REFERENCES

[1] COMpfun successor Reductor infects files on the fly to compromise TLS traffic. 3 October 2019. https://securelist.com/compfun-successor-reductor/93633/.

[2] Legezo, D. Targeted attacks through ISPs. https://www.virusbulletin.com/conference/vb2019/abstracts/targeted-attacks-through-isps.

INDICATORS OF COMPROMISE

File MD5 Hashes

A6AFA05CBD04E9AF256D278E5B5AD050 trojan 32-bit

1BB03CBAD293CA9EE3DDCE6F054FC325 trojan 64-bit

IPs

95.183.49.10

95.183.49.29

200.63.45.35