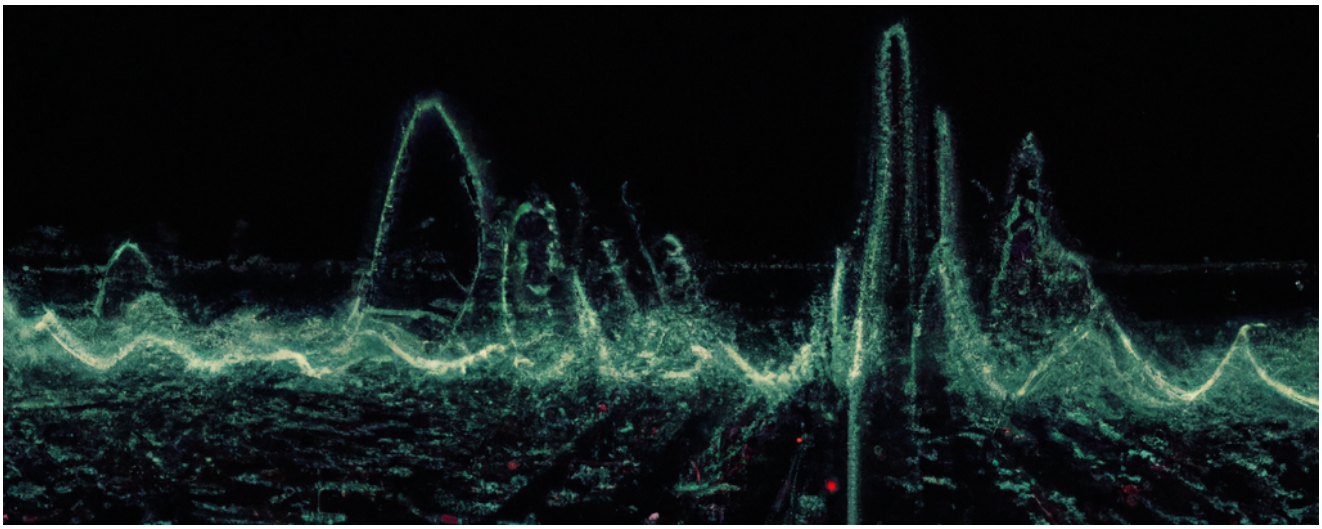
HijackLoader Targets Hotels: A Technical Analysis

alpine-sec.medium.com/hijackloader-targets-hotels-a-technical-analysis-c2795fc4f3a3

alpine-security September 23, 2023



alpine-security

--

18 Sep 2023 — Borja Merino

During the last months, the Alpine Security Hunting Team has observed several malware campaigns against various hotel chains in Andorra using HijackLoader as the main weapon of attack. Recently detailed and analyzed by , HijackLoader is a new malware loader that is used to load different malware families such as Danabot, SystemBC and RedLine Stealer. This Malware is characterized by using a modular design and implement several layers of obfuscation, anti-analysis and evasion techniques (DLL Stomping, Direct syscalls, process migration, etc.) to execute code as stealthily as possible.

In the campaigns observed, the attackers establish contact via email with the hotels to reserve a room and, under the pretext of suffering from food allergies, send a download link containing a compressed file with the malicious binary. The links usually point to a service with a good reputation (dropbox.com, drive.google.com, etc.)

In this other campaign the attackers contact via Booking with the hotel using a very similar apology; they inform on certain contraindications that should be taken into account by the hotel in order to avoid certain allergy problems described in an attached medical prescription.

The attachment in this case comes from the Discord CDN (hxxps://cdn.discordapp[.]com/attachments/1146064438449946687/1146072433867116564/my_contraindications.zip)

The present analysis shows the triage carried out on a certain binary following an alert in one of our clients, where the legitimate "ftp.exe" process (SysWOW64) was invoked from a recently created suspicious binary (previously unobserved in the company).

The analyzed binary, "**my contraindications.exe**", has a size of 1,538,816 bytes (1.5 MB), is developed in C++ for 32-bit architectures (Windows GUI Subsystem) and tries to pretend to be a legitimate McAffe binary, even embedding a certificate of said AV company. File-Header Timestamp reflects "2021–06–10 06:57:08".

The execution of the harmful code starts from the function (which is part of the Microsoft Visual C++ Runtime Library) where they have inserted a hook/call to the function from which the infection process starts. The process's main thread will therefore begin its harmful actions before reaching *WinMain()*.

The code will walk the PEB_LDR_DATA structure from PEB with the goal of retrieving the Kernel32 base address and traversing its EAT with which to retrieve symbols.

The malicious code will also load "winhttp.dll" in order to carry out network communications. One of these connections is to the legitimate domain "*doi.org*". The code, in a loop, will wait for communication with it to proceed with the infection actions.

The loader makes some direct syscalls in order to bypass certain security solutions; in the image below, *NtDelayExecution* (which is used recurrently during the infection chain).

HijackLoader will retrieve a PNG image from different image hosting services with the aim of recovering the next stage along with the corresponding malware family. The encrypted URIs, are listed below:

```
hxxxs://i.ibb[.]co/MMdnckd/alcocain.pnghxxxs://i.imgur[.]com/tGBX8NN.pnghxxxs://files.
```

The images will be recovered via WinHTTPReadData. The following image shows, already in memory, the PNG from "i.imgur[.]com/tGBX8NN.png"

The way the loader retrieves the payload is as follows:

1. Identifies a certain DWORD TAG within the image (). This TAG will serve as a starting point to reconstruct the XOR'ed payload.
2. Recovers and concatenates chunks of bytes separated by another ID (0x49444154).
3. Decrypt, via XOR, the set of bytes previously concatenated using the DWORD located right after the TAG as a KEY.

4. After applying the XOR, the resulting buffer will be decompressed (LZNT1) via (COMPRESSION_FORMAT_LZNT1). The bytes that accompany the XOR KEY will determine the size of the compressed buffer and its uncompressed size.

The following image show the buffer with the header in which the TAG and the XOR key are located. At the bottom you can see the routine in charge of applying the encryption using the KEY (in the example: **0xC5A2B15F**).

Finally, after applying the decryption, the payload embedded and compressed in the image will be recovered using *RtlDecompressBuffer.*

The was made to automate the extraction of the payloads of the different images during the analysis.

The uncompressed buffer contains the configuration file, some of the HijackLoader's modules described by Nikolaos Pantazopoulos and certain shellcode that will orchestrate the infection process until the final payload is executed.

For example, the DLL highlighted in the configuration ("C:\Windows\SysWOW64\mshtml.dll") is used to do DLL Stomping where the following stage is copied. After loading this DLL, it will invoke *VirtualProtect* function to modify the .text section to RWX (PAGE_EXECUTE_READWRITE) permissions. Subsequently, it will proceed to copy and write the next stage (one of the modules embedded in the previous buffer).

After writing the shellcode, the permissions will be reset to RX and a jump to the new stage will be made ("*call esi*" in the following image). The previously described logic is shown below.

The new stage will create a new process from the legitimate binary "*C:\Windows\Syswow64\ftp.exe*" in hidden mode (CREATE_NO_WINDOW flag) to inject the next stage.

The harmful code will force the "mshtml.dll" DLL to be loaded into the address space of the newly created "ftp.exe" process and will modify its *.text* section again to replace it with the last stage that will trigger the execution of the final payload. Finally, after its copy, it will modify the context of its main thread to point to the new payload vía *NtSetContextThread*.

The final payload is currently being analyzed. Yara is shared below for the described sample.

```
rule HijackLoader{
meta:
 description = "HijackLoader (Andorra Hotel campaign)"
 author = "@BorjaMerino (Alpine Security)"
 version = "1.0"
 date = "2023-09-18"
strings:
 $x1 = {4? 39 ?? 89 ?? 74 ?? 0F B6 ?? ?? 18 30 ?? ?? 4? 83 ?? ?? B? 00 00 00 00 74 ??
89 ?? EB ??}
 $x2 = {64 8B ?? 30 00 00 00 8B ?? 0C 83 ?? 0C}
 $x3 = {90 90 0F B7 ?? 01 ?? 0F B7 ?? 83 C? 02 66 85 ?? 74 ??}
 $x4 = {39 ?? 74 14 8D ?? 01 8B ?? 24 0C 8B ?? 24 39 ?? ?? 01 89 ?? 75 EA}
 $x5 = {90 90 31 ?? ?? 83 C? 04 39 ?? 72 f6}

condition: uint16(0) == 0x5A4D and uint16(uint32(0x3C)+0x18) == 0x010B  and
(pe.number_of_signatures > 0) and (filesize > 1MB and filesize < 5MB)  and 2 of
($x*)}
```

Referencies

Stealing More Than Towels: The New InfoStealer Campaign Hitting Hotels and Travel Agencies

Technical Analysis of HijackLoader

Alpine Security