Innovation in Cyber Intrusions: The Evolution of TA544

December 18, 2023

12/18/2023

Introduction

Innovation is not only an activity performed by companies, committed to protecting their perimeter, but is also an provided by threat actors. In fact, while organizations are investing in cybersecurity operations, such buying or implementing digital defenses, threat actors are implementing new strategies to bypass those protections.

An example of this type of innovation is TA544, also known as Narwhal Spider, Gold Essex, and recently known as Ursnif Gang, the notorious group hit Italy in past with massive attacks waves of Ursnif malware past years.

During last weeks, we observed a significant evolution in its TTPS, involving the adoption of new cyber weapons in all its infection chain, such as the abandon of Ursnif in favor of HijackLoader, aka IDAT Loader, and the delivery of other malware payloads, likes Remcos and SystemBC, passing through a massive abuse the DLL sideloading.

In the case under observation, the goal of the infection is to lead to the execution of the RAT (Remote Administration Tool) RemCosRAT, a lightweight and legitimate software used for remote control which is used by cybercriminals to facilitate access to infected machines and purse its new goal of Initial Access broker inside the new cybercriminal business model.

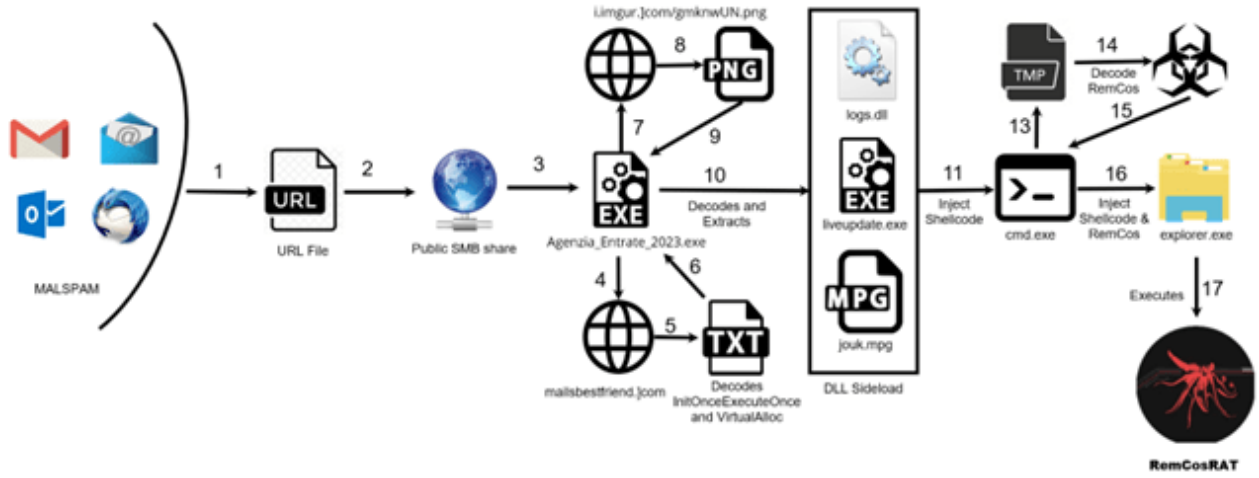

Figure 1: TA544 brand new Infection Chain

Technical Analysis

During last weeks, we observed a serious variation in TA544's cyber intrusions. The new infection chain involves new components and attack procedures. For this report, we take in exam the campaign spread on 21th November and reported by the independent Security Researcher @JAMESWT_MHT.

The infection chain starts with a malicious mail containing a malicious link, which downloads a URL file, having the following static information.

SHA256	e3454a40e1903c9369f74b323df4dda0931449a0321cd3ae21f3e8d0ff92b93c
Threat	IDAT Loader/Remcos
Threat Description	Url downloading IDAT Loader payload

This file can the treated as a Internet shortcut, containing a pointer to a remote resource in the Internet. Generally this kind of threat contains a HTTP link, but a recent TTP is to abuse the SMB protocol and point to a public share, so enabling the next stage of the infection.

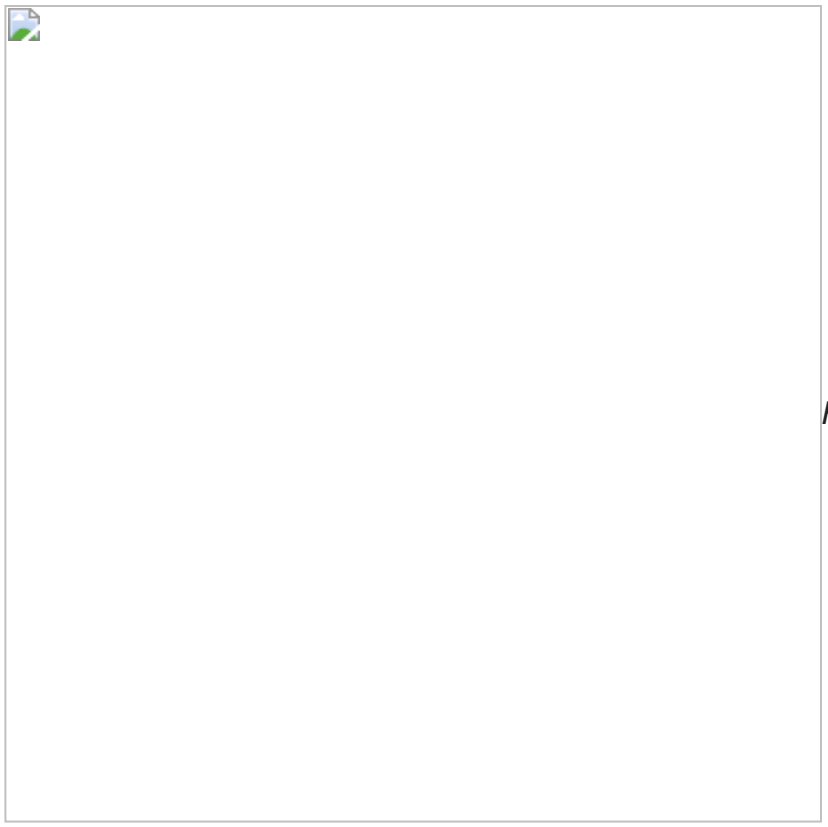

Figure 2: URL downloading the first executable

So, the URL downloads the first executable of the infection chain, which, after attribution, is a new version of IDAT Loader. This is a relatively new malware, first reported by Rapid7 researchers. During this infection chain, IDATLoader is widely used in all the intermediate stages in both Executable and DLL version.

The First IDATLoader packer is a trojanized executable written in C++, containing a simple, but sometimes effective anti-analysis trick: if the name is exactly the one indented to be by the attacker, the infection goes on, otherwise the malware evades by showing a MessageBox of a generic error.

The algorithm is quite easy. The malware retrieves the file name thanks to the **GetModuleFileNameW** API call. Then it performs two checks on that filename. The first one is quite easy: it is only the lengh of the name compared to the hardcoded one; the second one iterates the characters of the filename and sums the hexadecimal value of each character with the next one, the result of this operation then is checked against an harcoded value in the rdata section:

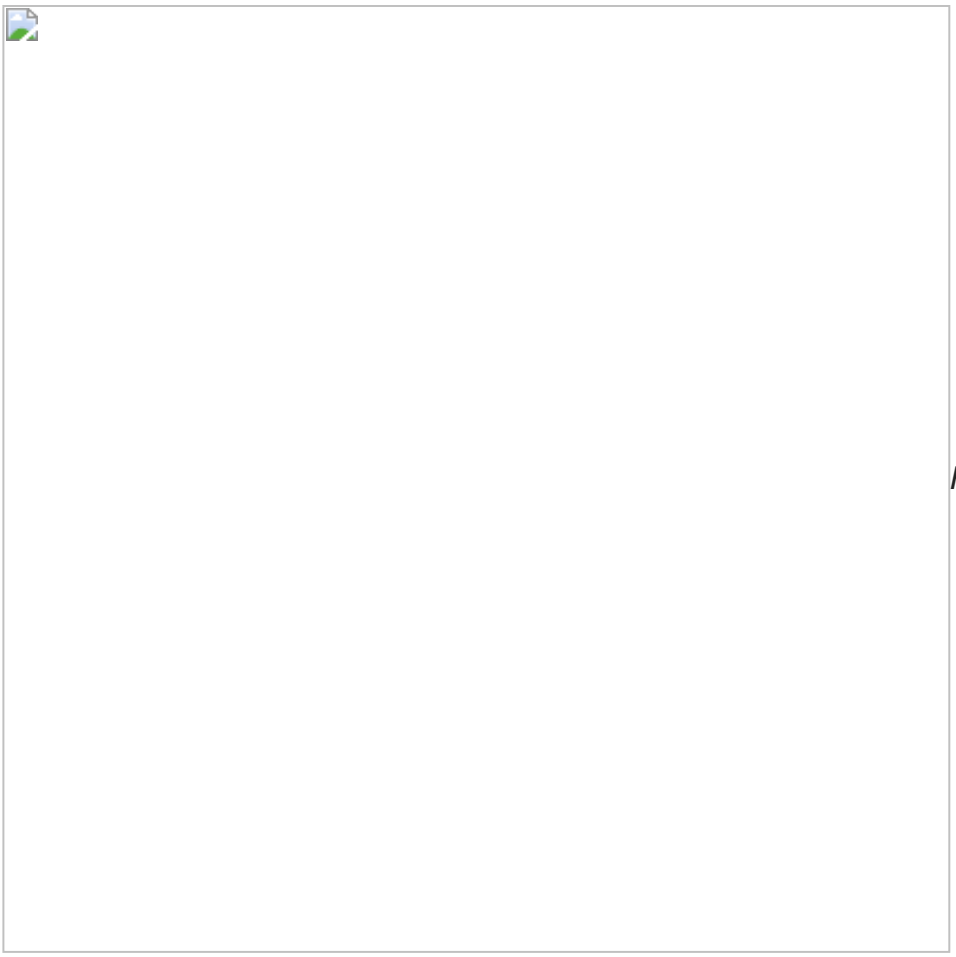

Figure 3: Filename check

If these checks pass, the malware downloads the file ***hxxps://mailsbestfriend.]com/downloads/Filters/FILTER-SOLICIT.txt*** and from its content builds the string ***InitOnceExecuteOnce***, which is a function used to execute the next subroutine. This API call is extensively abused in this malware because of its callback design, even useful when dealing with the execution of shellcodes. Then it uses the same technique for VirtualAlloc and writes shellcode to the allocated memory, which is a trampoline to decode and inject another stage of shellcode inside the PLA.DLL library, a legit Microsoft library (Performance Logs and Alerts Library) which provides the ability to generate alert notifications based on performance counter thresholds.

After dynamically loading the APIs in this new shellcode hosted inside PLA.dll, the malware downloads a png hosted on ***hxxps://i.imgur.]com/gmknwUN.png.*** At this point the behaviour of IDAT Loader emerges: the shellcode is responsible for looking for “IDAT” structures inside the PNG file and extracting the next stage code.

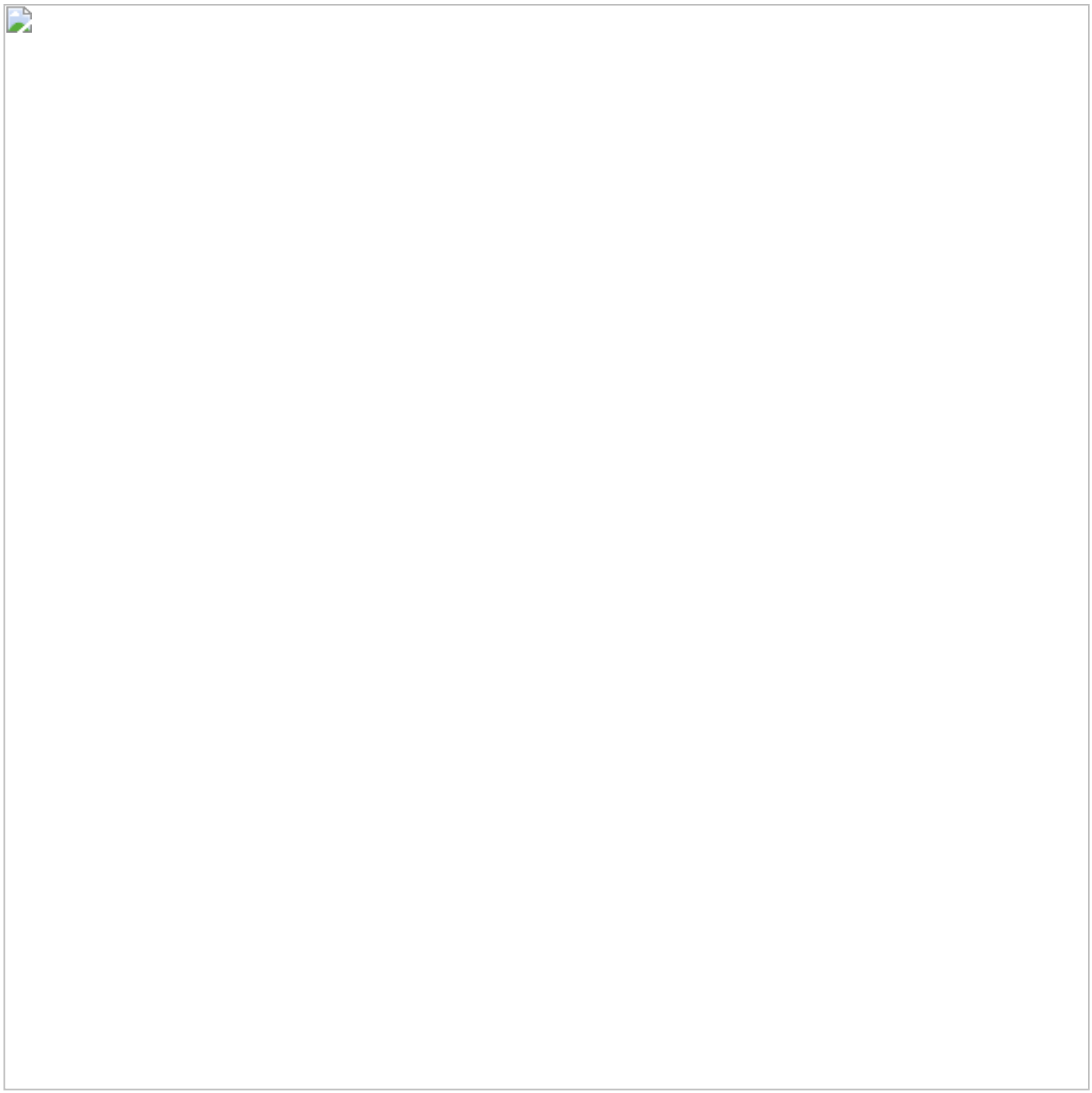

Figure 4: PNG containing the next stage using steganography.

At a first instance, this image seems to be a legit image, but in the bottom part there is not rendered well, indicating the possibility of an hidden payload, with a sort of steganography. This hypothesis is confirmed by inspecting the code and viewing a particular routine aimed at comparing the next 4 bytes after "**IDAT**" header of the png file with a hardcoded value:

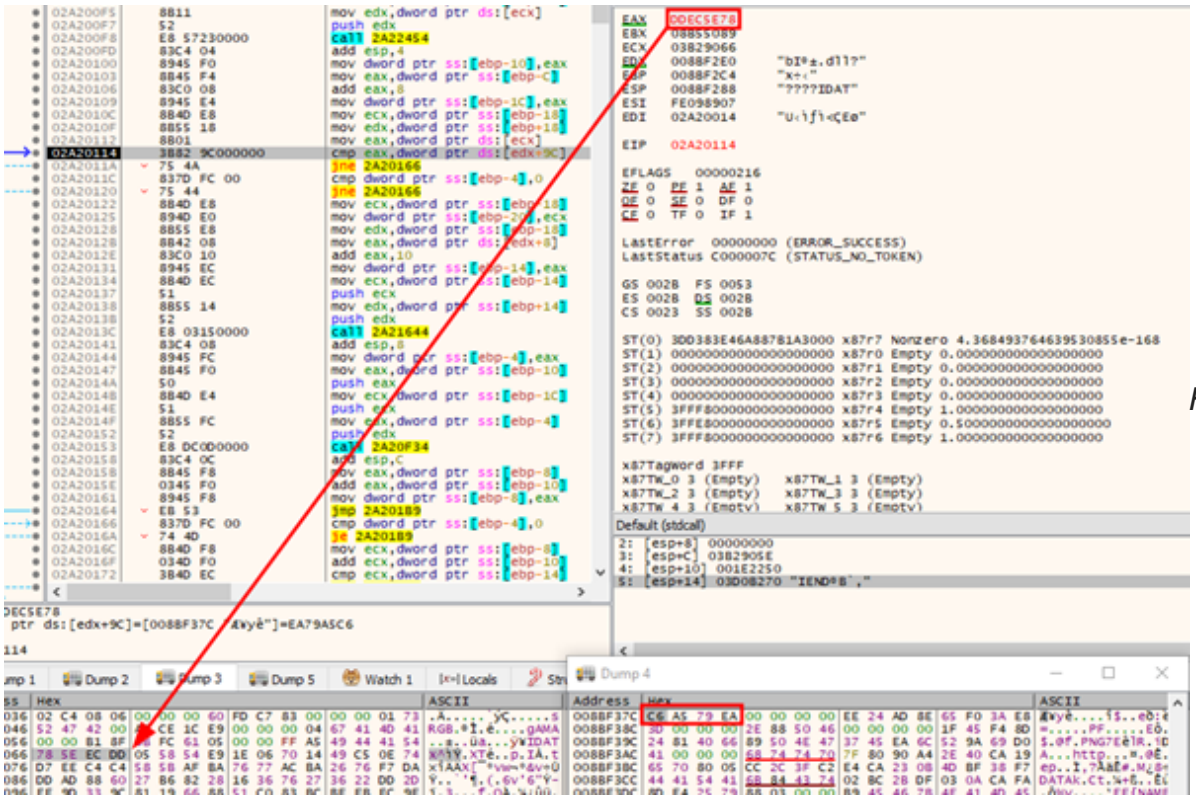

Figure 5: Comparing the next 4 bytes after IDAT

When the hardcoded value is checked, the malware starts the decryption and decompression routine for the next stage of the malware:

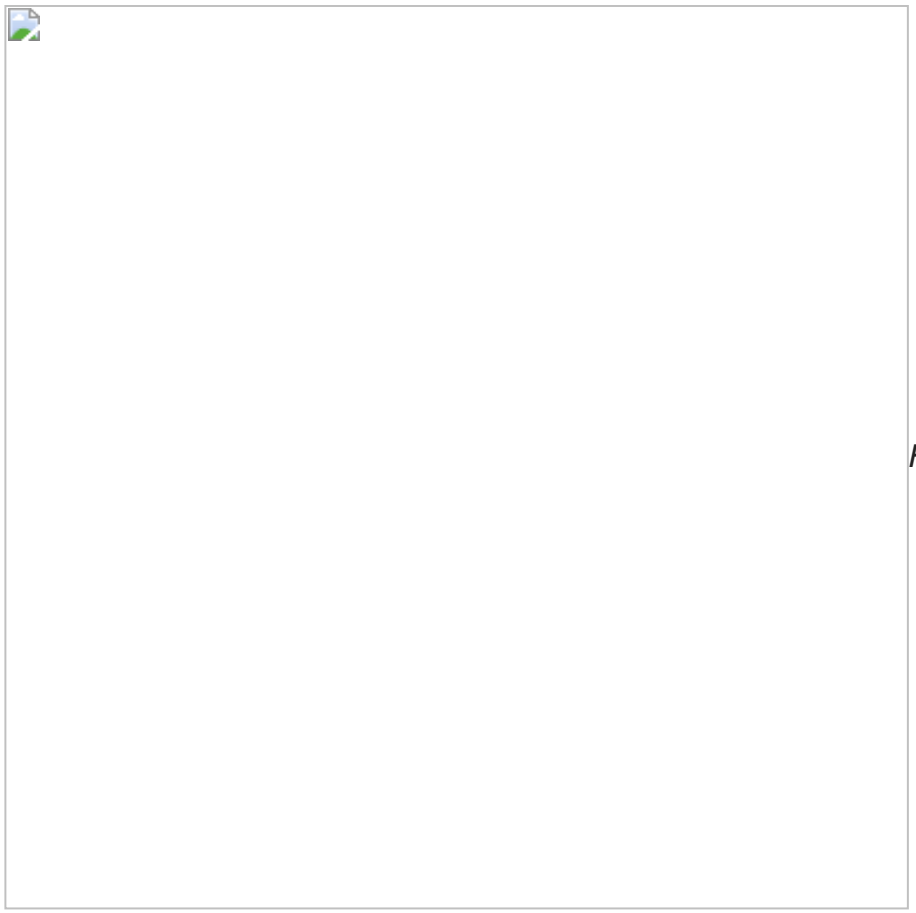

Figure 6: Decryption and decompression

Then, after the decoding phase, the malware writes all the extracted files inside the "%*appdata%\Roaming\DebugApp_v1*" directory. After writing the files, the malware invokes the API call ***CreateProcessW*** in order to execute "**liveupdate.exe**", which will sideload "log.dll" library.

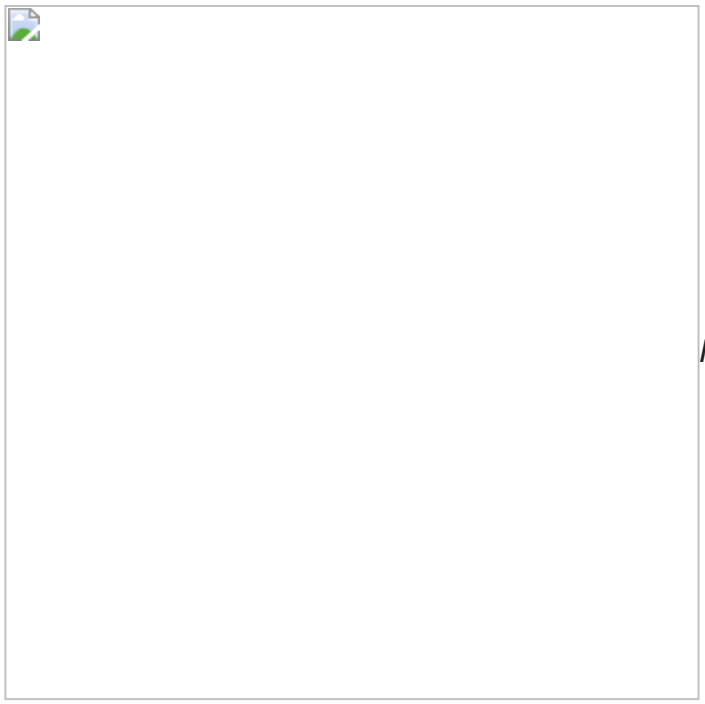

Figure 7: Next stage containing the trojanized log.dll

The "***log.dll***" library is a trojanized dependency read by the "liveupdate.exe" process, which immediately reads the "jouk.mpg", an encrypted file containing the shellcode to load in memory aimed at propagating the infection to the next stages. This new piece of code has the goal to set as an environment variable with the same code thanks to the **SetEnvironmentVariableW** API call, in to retrieve it in the next stage through the **GetEnvironmentVariableW** call.

This new step is to is to run a cmd.exe process through the ***CreateProcessW*** API call inject a piece of shellcode, performed though the Heaven's Gate technique and a series of direct syscalls. Heaven's Gate technique in malware analysis refers to a sophisticated method employed by malicious software to obscure its code and evade detection. This technique involves switching between 32-bit and 64-bit execution modes during runtime, complicating the analysis process. By utilizing specific opcodes, such as the 0x33 operand prefix, malware can dynamically transition from 32-bit to 64-bit mode or vice versa. Direct syscalls, on the other hand, represent a low-level approach in malware execution, in this way, threat actors are able to bypass standard library functions, allowing malware to interact with the operating system kernel at a more fundamental level.

The principal syscalls aimed at the injection routine are ***NtCreateSection***, ***NtMapViewOfSection*** and ***NtWriteVirtualMemory*** to remotely load even this time the ***pla.dll*** library inside the just created cmd.exe process and then inject the shellcode inside its *.text* section.

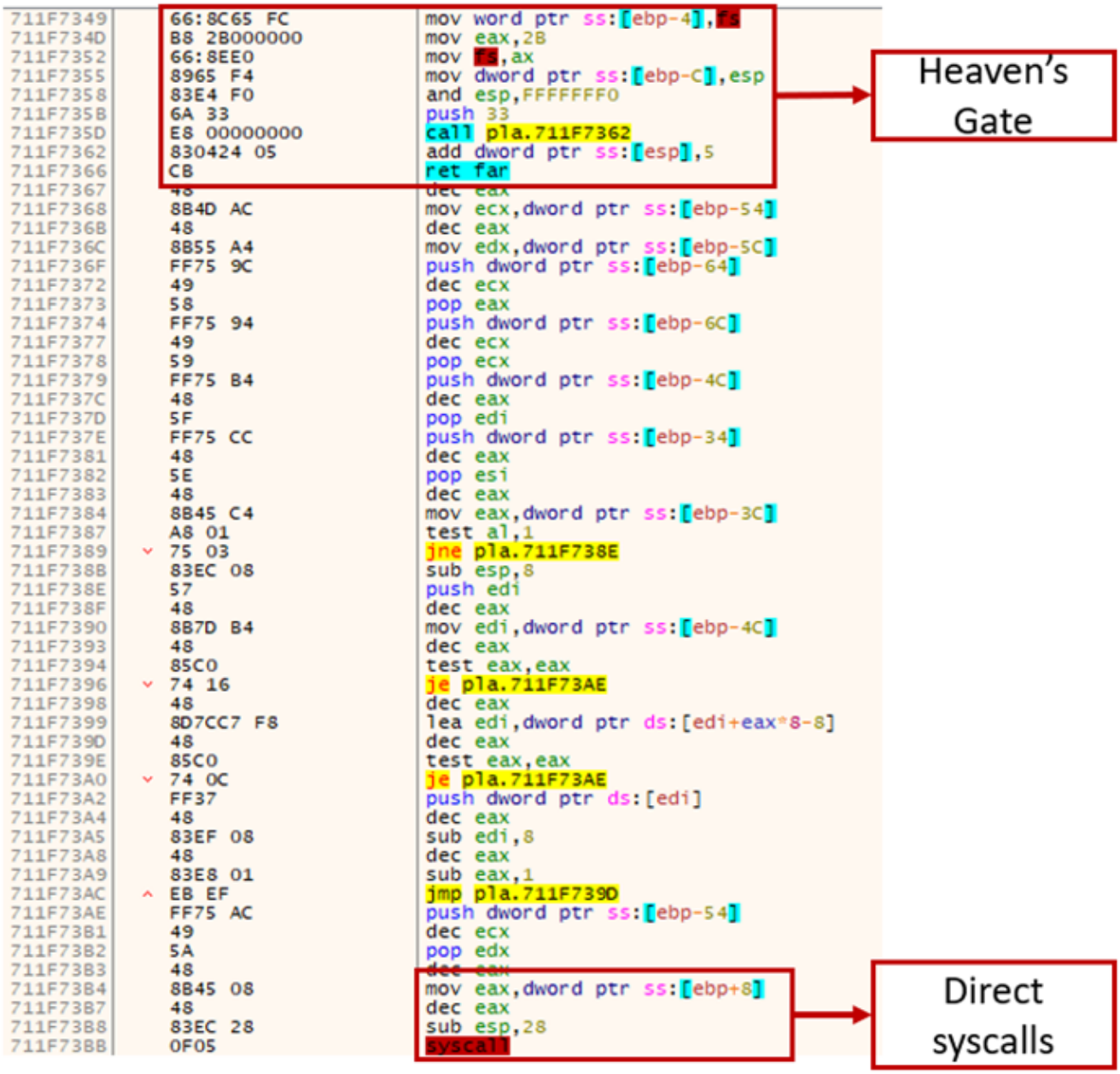

Figure 8: Using heaven's gate and direct syscall for the injection.

An instance of direct syscall used by the malware is the case of ***NtWriteVirtualMemory,*** the routine aimed at write the code inside the remote process' memory. In the following figure, it represented the opcode of the syscall along with the parameters pushed on the stack.

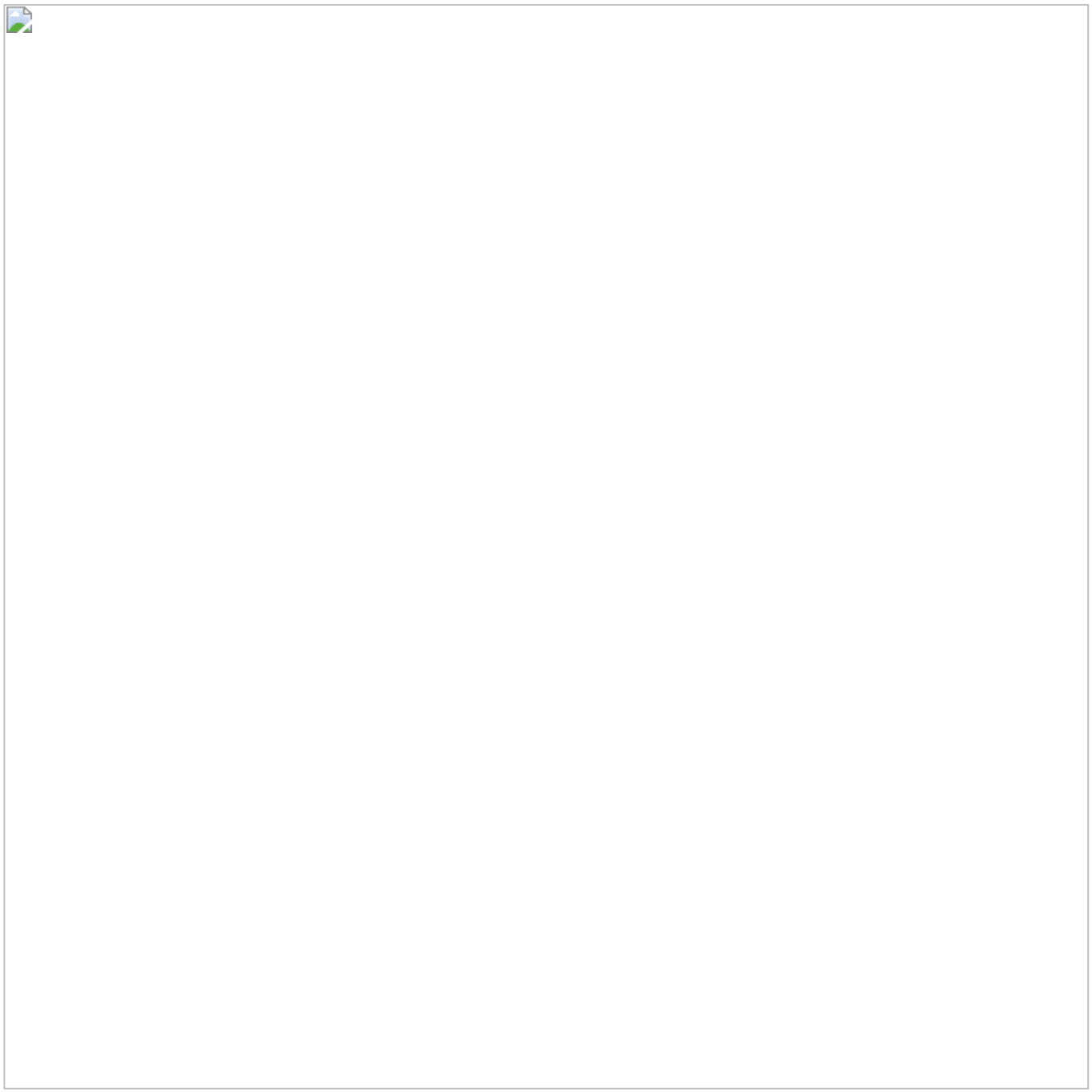

Figure 9: NtWriteVirtualMemory through syscall to inject shellcode to cmd.exe

At this point, the malware writes a file in the temporary folder with a random name, which contains the RemCos payload with other configuration data and additional modules for IDAT Loader.

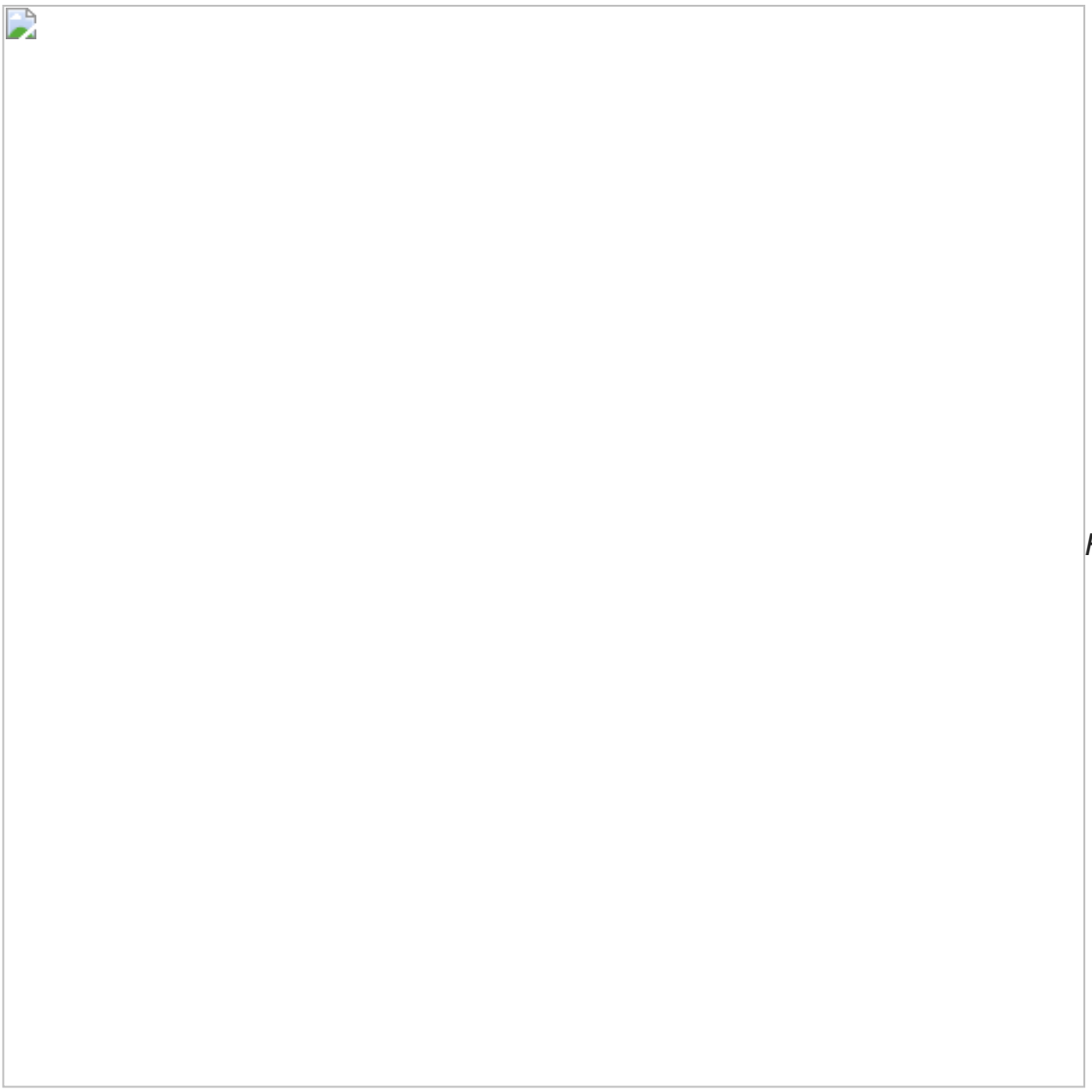

Figure 10: Writing a file in %temp%

This data is decrypted using XOR with a hardcoded key. For the analysis sample is **EC4837D0.**

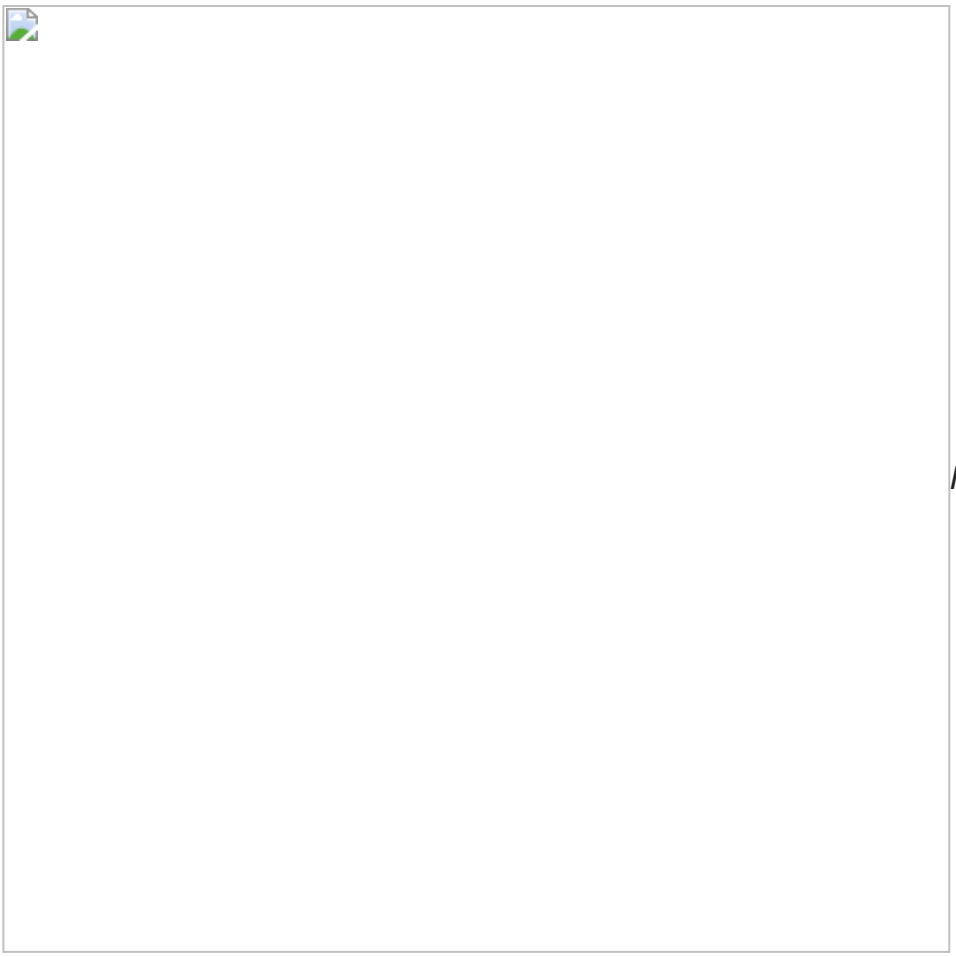

Figure 11: Decryption of the temporary file.

When the control passes to the cmd.exe process, the shellcode injected inside the **pla.dll** library.

At this point the shellcode sets the malware persistence through the creation of a LNK file pointing to the "%*appdata%\Roaming\DebugApp_v1\liveupdate.exe*" file. This technique is quite effective because all the security controls consider that kind of operation as legit because the liveupdate executable is legit.

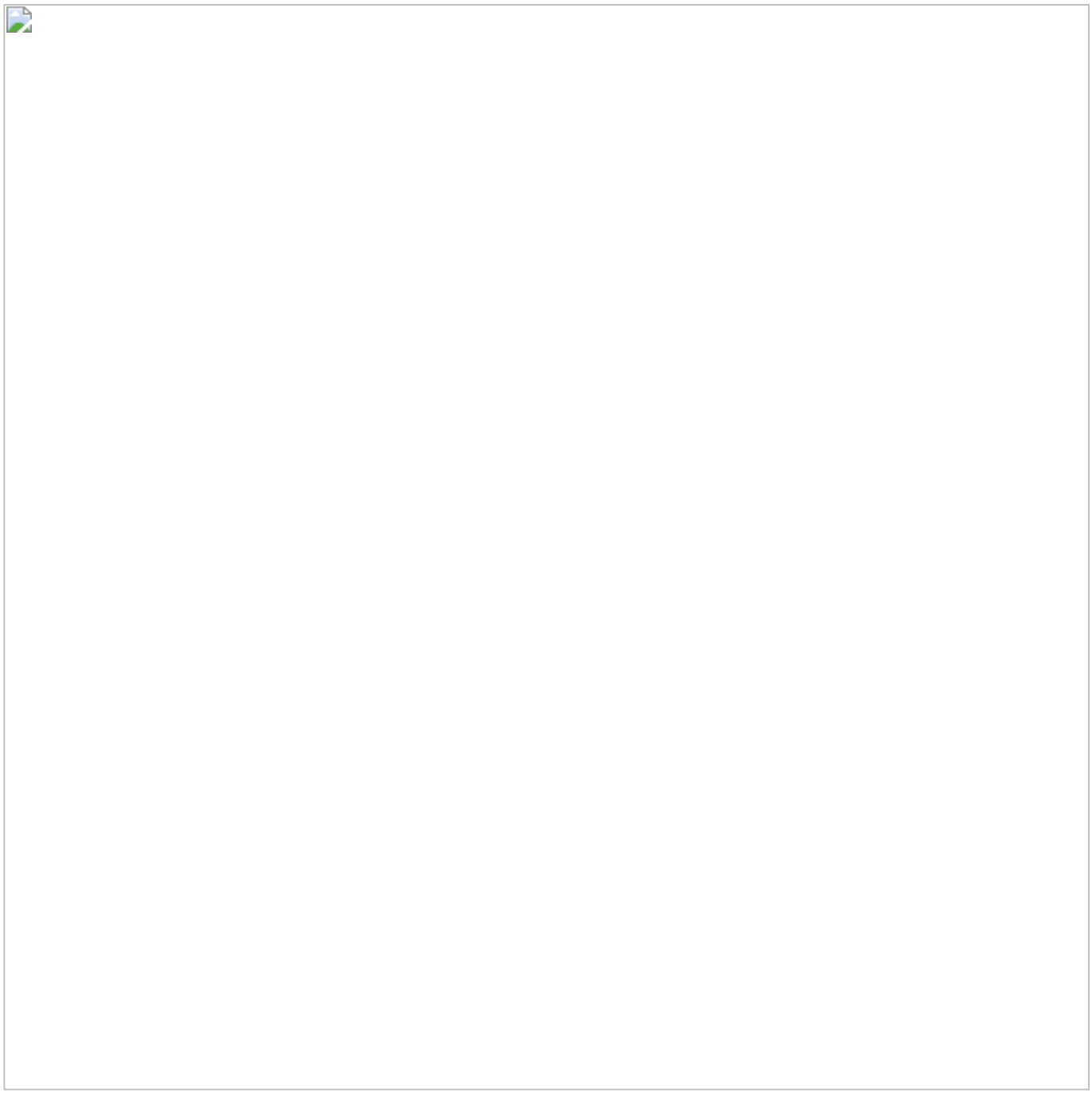

Figure 12: Setting up the persistence.

At this point, the malware goes on the infection chain by retrieving the temporary file written in the previous step and start the decryption of the Remcos payload contained inside that. The encryption is performed by using a XOR key 200 bytes-long, as shown in the following Figure.

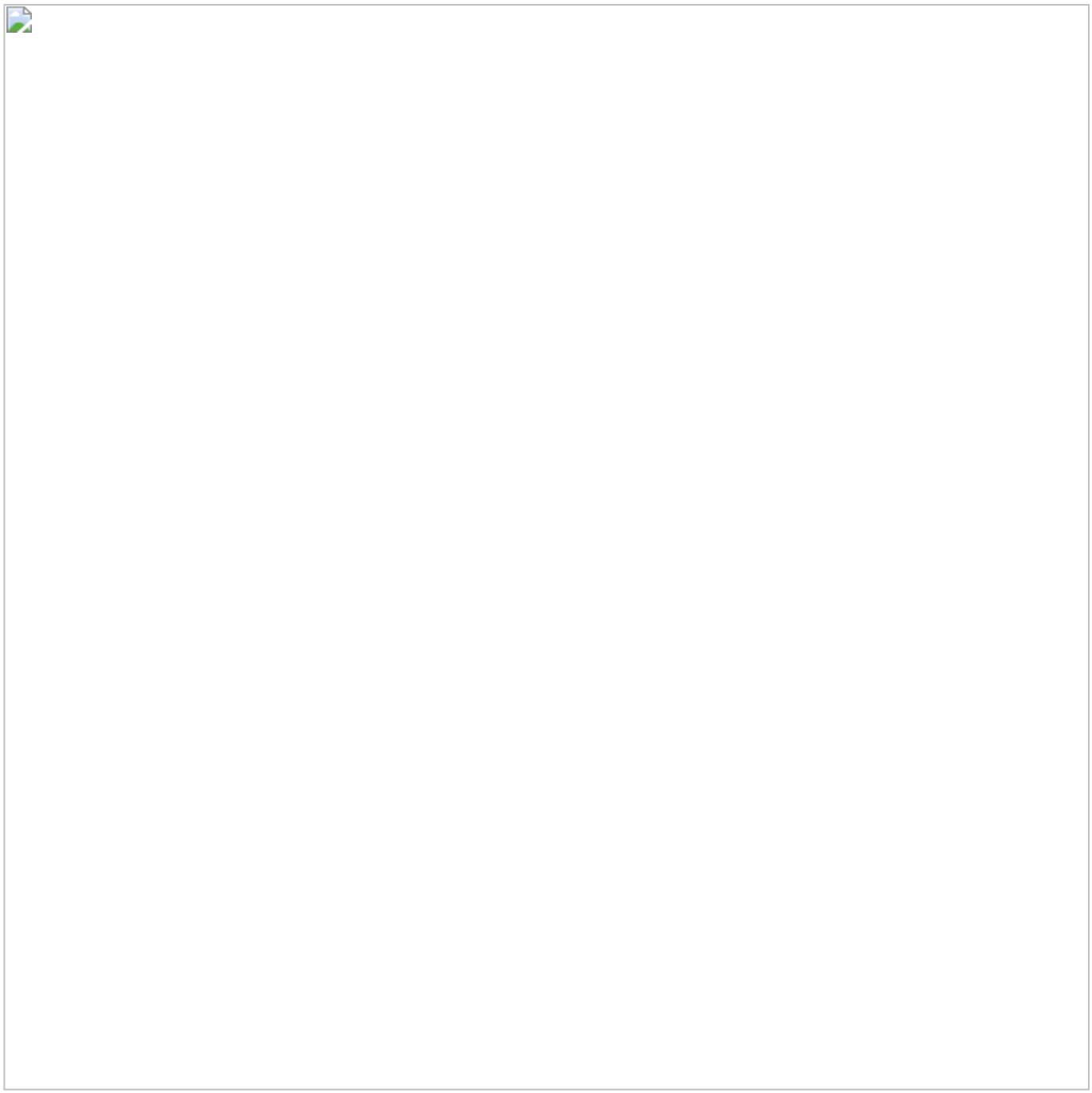

Figure 13: Decryption of the Remcos payload

Then the malicious CMD process calls the ***VirtualAlloc***, which allocates the memory to write the final shellcode. However, this long payload is injected inside another instance of the ***explorer.exe*** process created through the **CreateProcessInternalW** API call in suspended mode and injects that shellcode inside of it.

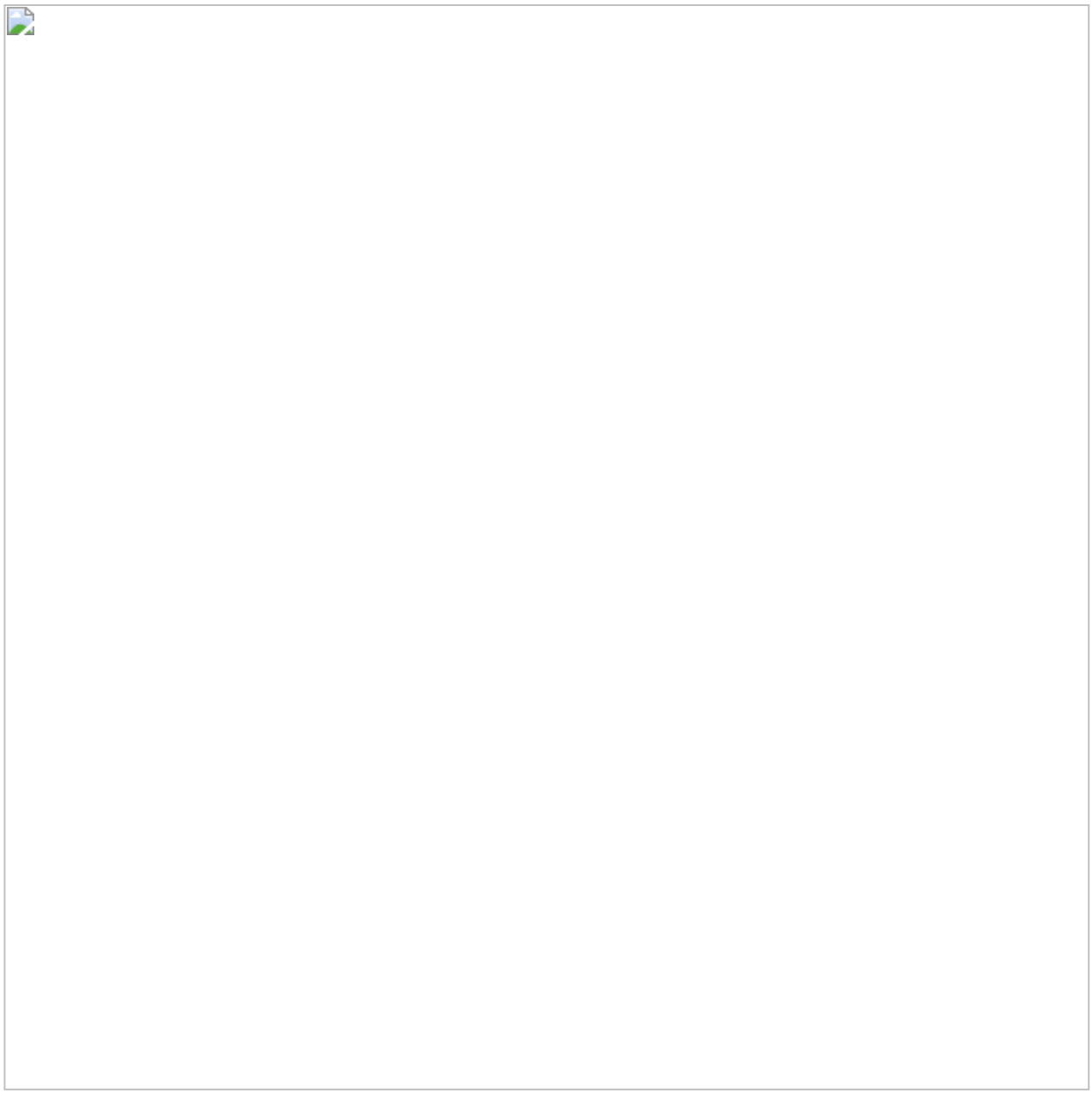

Figure 14: NtWriteVirtualMemory through syscall to inject shellcode to explorer.exe

Instead, for the injection of the Remcos payload, the malware uses the Heaven's Gate as mentioned in the previous stage. The routine is to create a new section inside the cmd.exe process through the NtCreateSection and the map it on the target process through the **NtMapViewOfSection** syscall, with the code **0x28**. This method works because the malware points to the handle to new **explorer.exe** process.

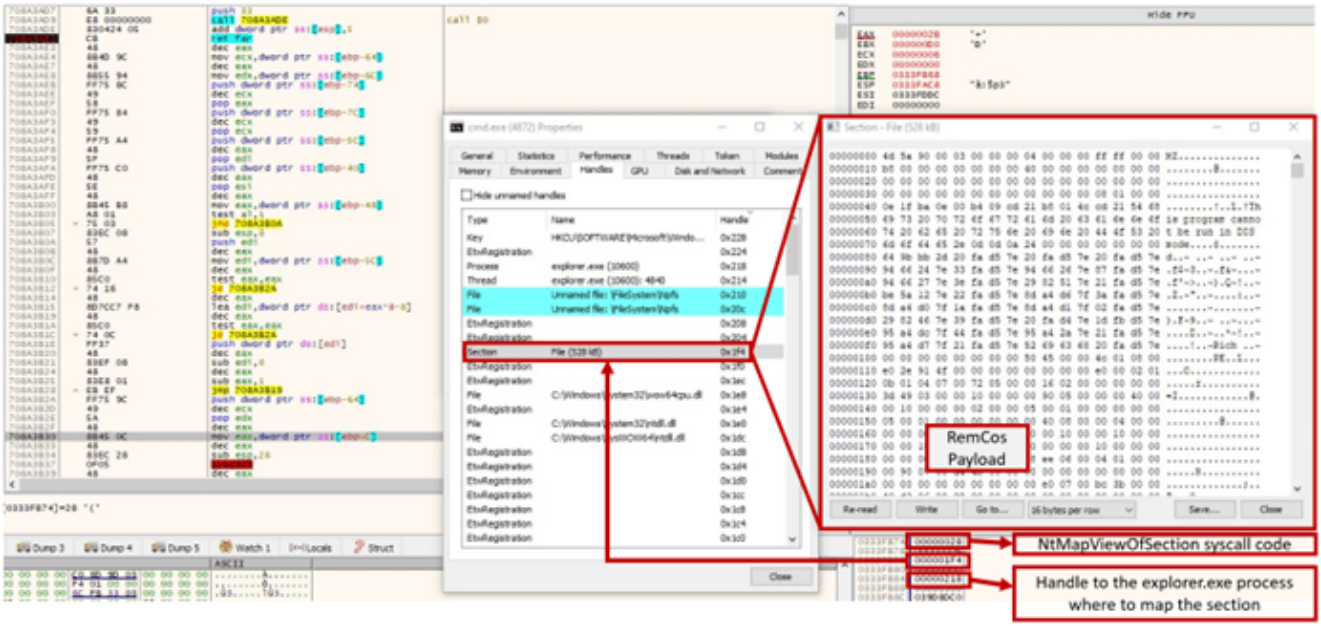

Figure 15: Mapping the Remcos Payload to exoplorer.exe

The last step of the analysis is to confirm that is Remcos malware. As report by many security firms, Remcos stores its configuration inside a resource, protecting it with a RC4 key long the first byte of that resource, and appended to the key there is the encrypted configuration:

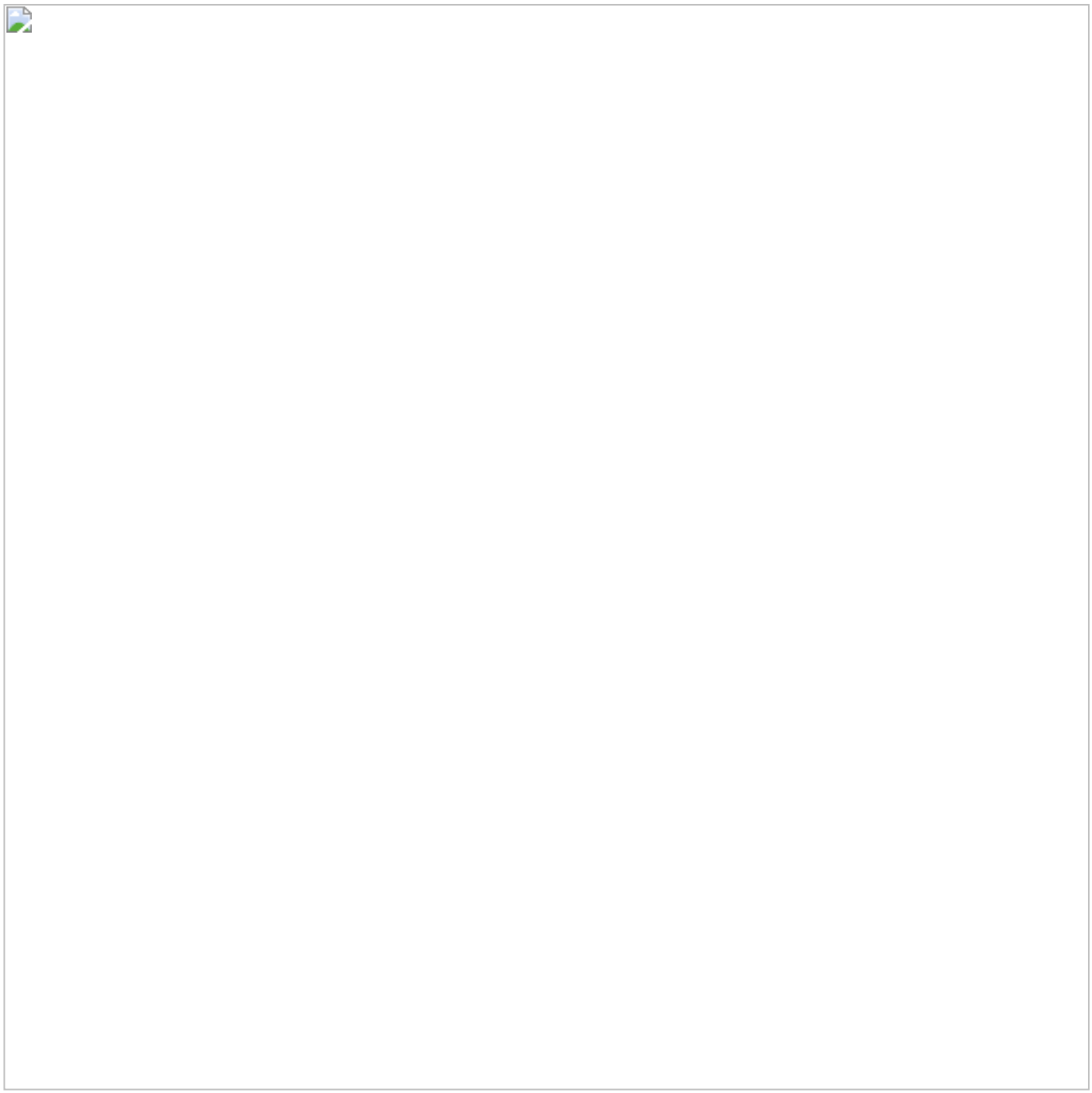

Figure 16: Remcos config

Conclusion

TA544 has been a constant threat in the past years to Italian organizations, in this report we wanted to highlight the importance to monitor the never-ending evolution of TTPs that occur to threat actors to elude defenses and be one step ahead. In the recent weeks after the longstanding wave of Ursnif spam, TA544 has switched to using IDAT Loader and Remcos, while also trying for a moment SystemBC as reported by the independent security researcher @JAMESWT_MHT

The evolution of the actor since 2017, when we started to monitor it is notable. This means that threat actors are realizing that they need to improve and innovate their TTPs in order to maintain their competitiveness high. Now, it is evident that TA544 is specializing in IAaaS (Initial Access as a Service). In fact, if we think about the Ursnif malware, we all know that has been designed to be as a Banking Trojan, but going on its evolution it is been evolved as backdoor for the Human Operated cyber intrusions as IAaaS and now with other RATs, like Remcos, SystemBC, etc.

Indicators of Compromis

- Hash

  2289f5e6c2e87cf4265ed7d05ef739d726ebd82614a1b856d4b5964834d307c9
  6e5db2efcad7fbacc72f1db53741d342a2524a481c4835885fe6c3a46e9036b3
  dd277db4beda582c70402c9163491da27fde7cba2906f15e5beb8b2a394c400b
  e02471f33d07a4f9046be6e7b15de68093bb72fdd15b61f3033aea57d9940108

- C2:

  listpoints.]online:6090
  retghrtgwtrgtg.bounceme.]net:3839
  listpoints.]click:7020
  datastream.myvnc.]com:5225
  gservicese.]com:2718
  center.onthewifi.]com:8118

Yara Rules

```
rule HijackLoader
 {
   meta:
      author = "Yoroi Malware ZLab"
      description = "Rule for IDAT Loader inital sample"
      last_updated = "2023-11-27"
      tlp = "WHITE"
      category = "informational"
  strings:
      $1 = {89 4D F4 C7 45 F8 00 00 00 00 C7 45 F? 00 00 00 00 8B 45 F? 8B 4D F4 0F B7
14 41 85 D2 74 ?? 8B 45 FC 8B 4D F4 0F B7 14 41 03 55 F8 89 55 F8 8B 45 FC 83 C0 01 89
45 FC}
      $2 = {C7 45 FC 00 00 00 00 C7 45 F? 00 00 00 00 8B 45 F? 8D 14 00 8B 45 08 01 D0
0F B7 00 66 85 C0 74 ?? 8B 45 F8 8D 14 00 8B 45 08 01 D0 0F B7 00 0F B7 C0 01 45 FC 83
45 F8 01}
  condition:
      any of them and uint16(0) == 0x5A4D
}
```

This Report has been authored by Luigi Martire, Carmelo Ragusa, Giovanni Pirozzi and Marco Giorgi

Download Report