TOFSEE

TECHNICAL ANALYSIS REPORT

ZAYOTEM

ZARARLI YAZILIM ÖNLEME VE TERSİNE MÜHENDİSLİK

Table of Contents

Overview

Tofsee is a powerful Trojan-type malware family that can be used as a botnet, causing serious damage including financial loss and computer infections.

Active since 2013, Tofsee malware spreads via spam emails advertising adult dating and drug sites. It is also known to originate mainly from Russia and Ukraine.

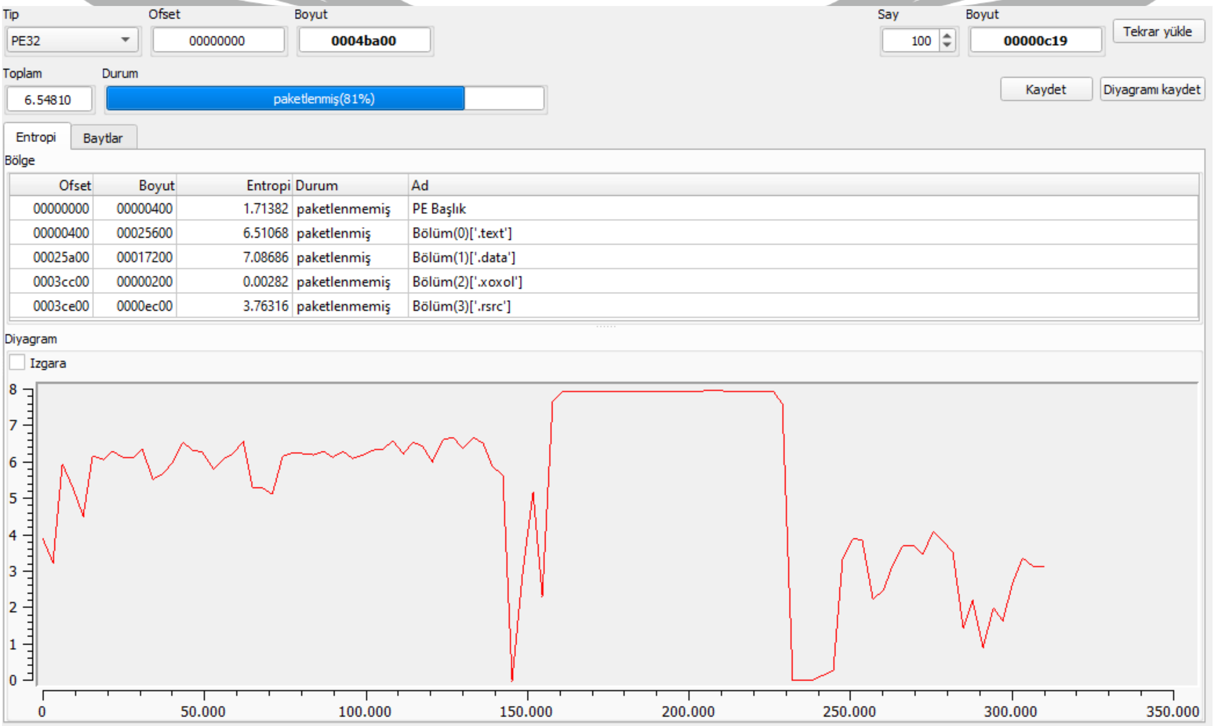
Despite being an email-oriented tool, having Tofsee installed can lead to many other problems. These problems include;

- Download malicious software,
- Sending spam emails,
- Conducting phishing attacks,
- Updating yourself,
- Steal various account credentials,
- Perform DDoS attacks,
- There may be methods such as forcing victims' computers to join other botnets.

Stage1 Analysis

Name	Fameborb.exe
MD5	9f9e5f55dc8cb3809e24b14fb8f9c27d
SHA256	b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad
File Type	PE32/EXE

Static Analysis



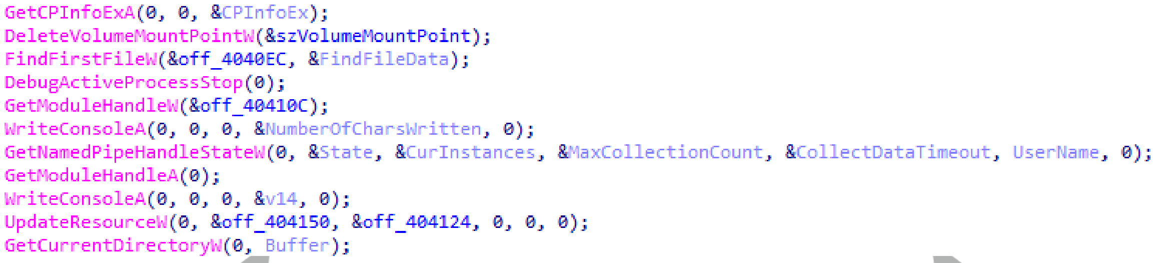
Tip: PE32
Ofset: 00000000
Boyut: 0004ba00
Say: 100
Boyut: 00000c19
Toplam: 6.54810
Durum: paketlenmiş(81%)

Bölge

Ofset	Boyut	Entropi	Durum	Ad
00000000	00000400	1.71382	paketlenmemiş	PE Başlık
00000400	00025600	6.51068	paketlenmiş	Bölüm(0)['.text']
00025a00	00017200	7.08686	paketlenmiş	Bölüm(1)['.data']
0003cc00	00000200	0.00282	paketlenmemiş	Bölüm(2)['.xoxol']
0003ce00	0000ec00	3.76316	paketlenmemiş	Bölüm(3)['.rsrc']

Figure 1 - Observing the Packing Process in the Malware

At first glance, the malware appears to be packed.

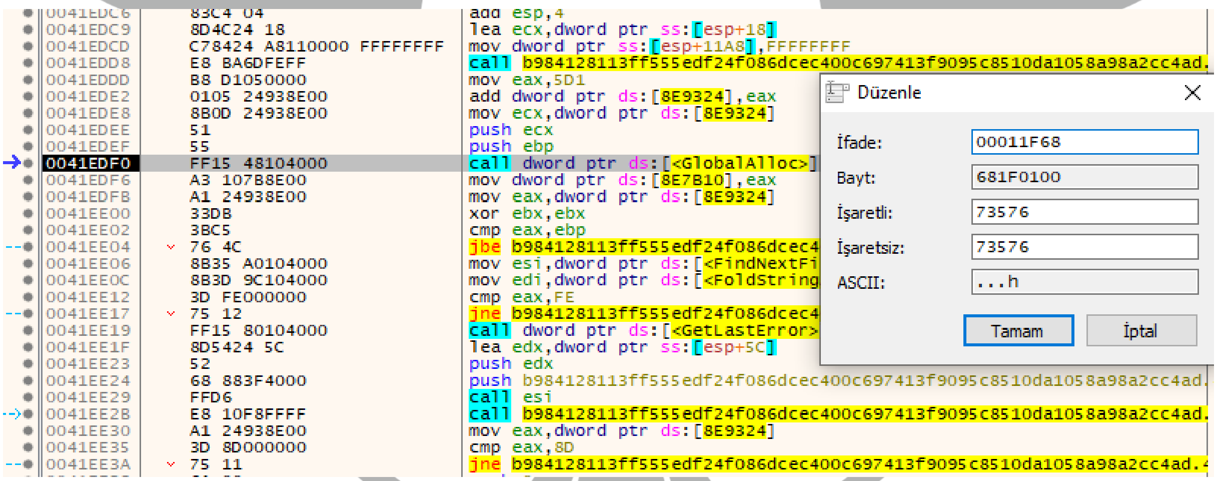


```
GetCPInfoExA(0, 0, &CPInfoEx);
DeleteVolumeMountPointW(&szVolumeMountPoint);
FindFirstFileW(&off_4040EC, &FindFileData);
DebugActiveProcessStop(0);
GetModuleHandleW(&off_40410C);
WriteConsoleA(0, 0, 0, &NumberOfCharsWritten, 0);
GetNamedPipeHandleStateW(0, &State, &CurInstances, &MaxCollectionCount, &CollectDataTimeout, UserName, 0);
GetModuleHandleA(0);
WriteConsoleA(0, 0, 0, &v14, 0);
UpdateResourceW(0, &off_404150, &off_404124, 0, 0, 0);
GetCurrentDirectoryW(0, Buffer);
```

Figure 2 - Obfuscate

It was observed that it was aimed to confuse the analyst by **obfuscating** the malware with empty and unnecessary API calls.

Dynamic Analysis



```
0041EDC6   83C4 04                    add esp,4
0041EDC9   8D4C24 18                  lea ecx,dword ptr ss:[esp+18]
0041EDCD   C78424 A8110000 FFFFFFFF   mov dword ptr ss:[esp+11A8],FFFFFFFF
0041EDD8   E8 BA6DFEFF                call b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.
0041EDDD   B8 D1050000                mov eax,5D1
0041EDE2   0105 24938E00              add dword ptr ds:[8E9324],eax
0041EDE8   8B0D 24938E00              mov ecx,dword ptr ds:[8E9324]
0041EDEE   51                         push ecx
0041EDEF   55                         push ebp
0041EDF0   FF15 48104000              call dword ptr ds:[<GlobalAlloc>]
0041EDF6   A3 107B8E00                mov dword ptr ds:[8E7B10],eax
0041EDFB   A1 24938E00                mov eax,dword ptr ds:[8E9324]
0041EE00   33DB                       xor ebx,ebx
0041EE02   3BC5                       cmp eax,ebp
0041EE04   76 4C                      jbe b984128113ff555edf24f086dcec4
0041EE06   8B35 A0104000              mov esi,dword ptr ds:[<FindNextFi
0041EE0C   8B3D 9C104000              mov edi,dword ptr ds:[<FoldString
0041EE12   3D FE000000                cmp eax,FE
0041EE17   75 12                      jne b984128113ff555edf24f086dcec4
0041EE19   FF15 80104000              call dword ptr ds:[<GetLastError>]
0041EE1F   8D5424 5C                  lea edx,dword ptr ss:[esp+5C]
0041EE23   52                         push edx
0041EE24   68 883F4000                push b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.
0041EE29   FFD6                       call esi
0041EE2B   E8 10F8FFFF                call b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.
0041EE30   A1 24938E00                mov eax,dword ptr ds:[8E9324]
0041EE35   3D 8D000000                cmp eax,8D
0041EE3A   75 11                      jne b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.
```

Düzenle

İfade: 00011F68

Bayt: 681F0100

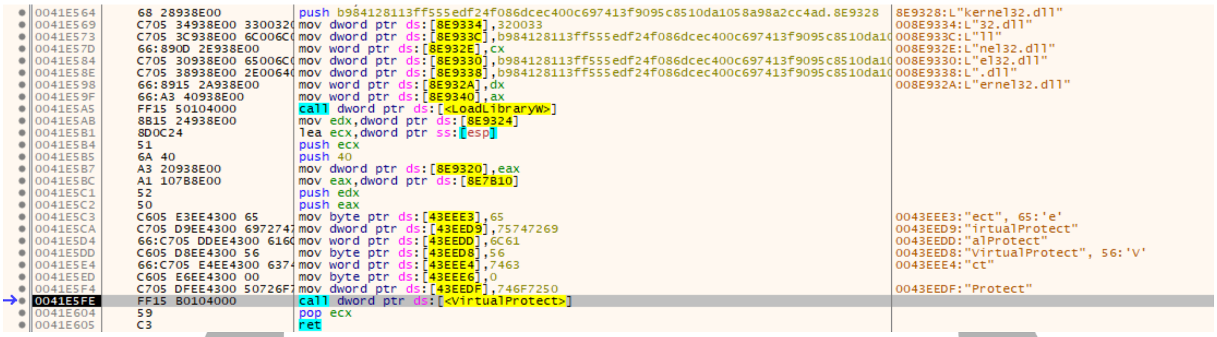
İşaretli: 73576

İşaretsiz: 73576

ASCII: ...h

Figure 3 – Allocating Memory Space

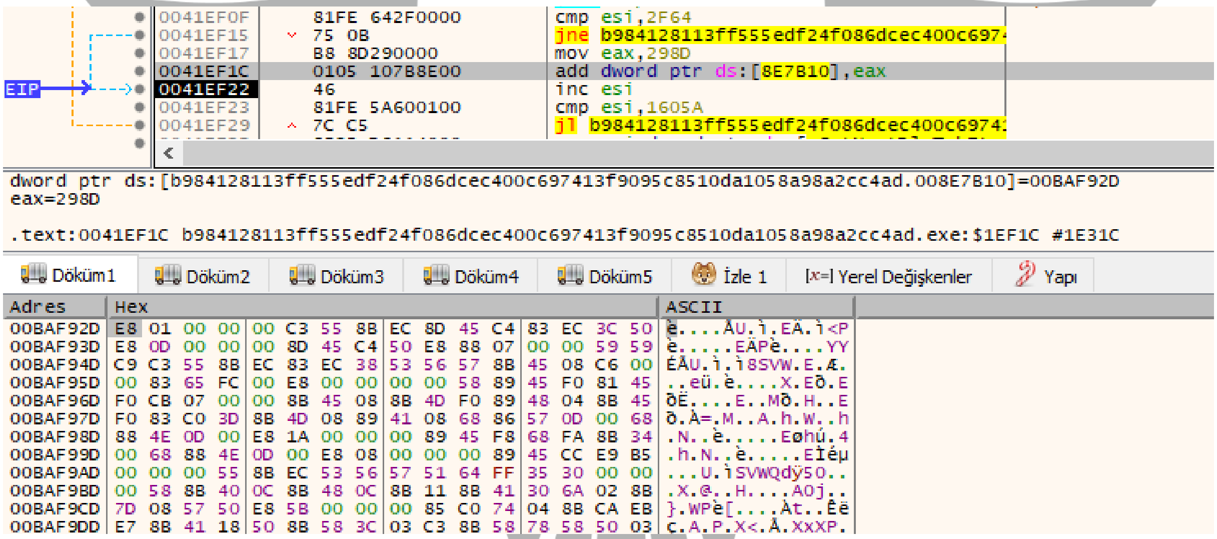
The malware was first observed to allocate **73576** bytes of memory with **GlobalAlloc**.



```
0041E564  68 28938E00            push b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.8E9328   8E9328:L"kernel32.dll"
0041E569  C705 34938E00 330032(  mov dword ptr ds:[8E9334],320033                                                008E9334:L"32.dll"
0041E573  C705 3C938E00 6C006C(  mov dword ptr ds:[8E933C],b984128113ff555edf24f086dcec400c697413f9095c8510da1  008E933C:L"ll"
0041E57D  66:890D 2E938E00       mov word ptr ds:[8E932E],cx                                                     008E932E:L"nel32.dll"
0041E584  C705 30938E00 65006C(  mov dword ptr ds:[8E9330],b984128113ff555edf24f086dcec400c697413f9095c8510da1  008E9330:L"el32.dll"
0041E58E  C705 38938E00 2E0064(  mov dword ptr ds:[8E9338],b984128113ff555edf24f086dcec400c697413f9095c8510da1  008E9338:L".dll"
0041E598  66:8915 2A938E00       mov word ptr ds:[8E932A],dx                                                     008E932A:L"ernel32.dll"
0041E59F  66:A3 40938E00         mov word ptr ds:[8E9340],ax
0041E5A5  FF15 50104000          call dword ptr ds:[<LoadLibraryW>]
0041E5AB  8B15 24938E00          mov edx,dword ptr ds:[8E9324]
0041E5B1  8D0C24                 lea ecx,dword ptr ss:[esp]
0041E5B4  51                     push ecx
0041E5B5  6A 40                  push 40
0041E5B7  A3 20938E00            mov dword ptr ds:[8E9320],eax
0041E5BC  A1 107B8E00            mov eax,dword ptr ds:[8E7B10]
0041E5C1  52                     push edx
0041E5C2  50                     push eax
0041E5C3  C605 E3EE4300 65       mov byte ptr ds:[43EEE3],65                                                     0043EEE3:"ect", 65:'e'
0041E5CA  C705 D9EE4300 697274   mov dword ptr ds:[43EED9],75747269                                              0043EED9:"irtualProtect"
0041E5D4  66:C705 DDEE4300 616C  mov word ptr ds:[43EEDD],6C61                                                   0043EEDD:"alProtect"
0041E5DD  C605 D8EE4300 56       mov byte ptr ds:[43EED8],56                                                     0043EED8:"VirtualProtect", 56:'V'
0041E5E4  66:C705 E4EE4300 6374  mov word ptr ds:[43EEE4],7463                                                   0043EEE4:"ct"
0041E5ED  C605 E6EE4300 00       mov byte ptr ds:[43EEE6],0
0041E5F4  C705 DFEE4300 50726F   mov dword ptr ds:[43EEDF],746F7250                                              0043EEDF:"Protect"
0041E5FE  FF15 B0104000          call dword ptr ds:[<VirtualProtect>]
0041E604  59                     pop ecx
0041E605  C3                     ret
```

Figure 4 – Permission to Write to Address in Memory

It was found that **Kernel32.dll** was loaded with **LoadLibraryW** and the location reserved with **GlobalAlloc** was then granted **PAGE_EXECUTE_READWRITE** permission using **VirtualProtect**.

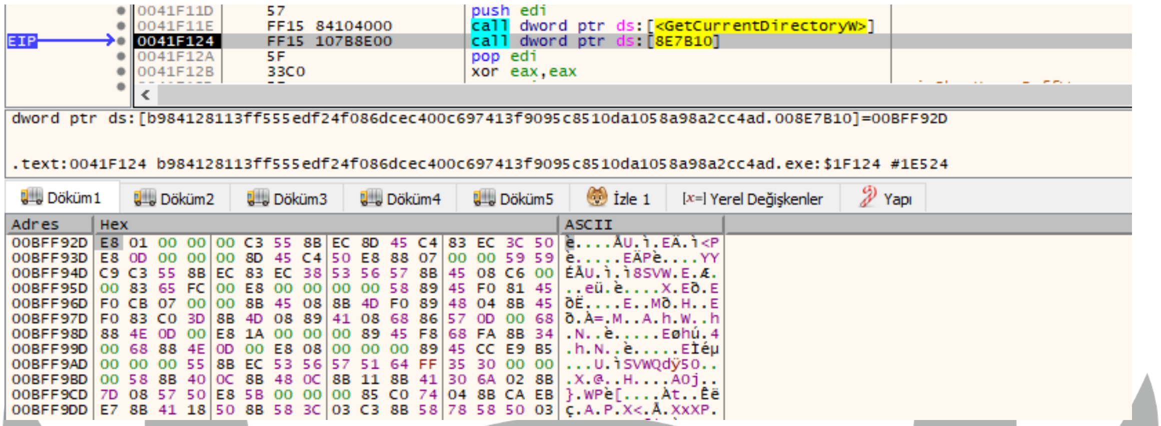


```
      0041EF0F     81FE 642F0000    cmp esi,2F64
      0041EF15  ⌄  75 0B            jne b984128113ff555edf24f086dcec400c697
      0041EF17     B8 8D290000      mov eax,298D
      0041EF1C     0105 107B8E00    add dword ptr ds:[8E7B10],eax
EIP → 0041EF22     46               inc esi
      0041EF23     81FE 5A600100    cmp esi,1605A
      0041EF29  ^  7C C5            jl b984128113ff555edf24f086dcec400c6974

dword ptr ds:[b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.008E7B10]=00BAF92D
eax=298D

.text:0041EF1C b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.exe:$1EF1C #1E31C
```

```
Adres     Hex                                              ASCII
00BAF92D  E8 01 00 00 00 C3 55 8B EC 8D 45 C4 83 EC 3C 50  è....ÃU.ì.EÄ.ì<P
00BAF93D  E8 0D 00 00 00 8D 45 C4 50 E8 88 07 00 00 59 59  è.....EÄPè....YY
00BAF94D  C9 C3 55 8B EC 83 EC 38 53 56 57 8B 45 08 C6 00  ÉÃU.ì.ì8SVW.E.Æ.
00BAF95D  00 83 65 FC 00 E8 00 00 00 00 58 89 45 F0 81 45  ..eü.è....X.Eð.E
00BAF96D  F0 CB 07 00 00 8B 45 08 8B 4D F0 89 48 04 8B 45  ðË....E..Mð.H..E
00BAF97D  F0 83 C0 3D 8B 4D 08 89 41 08 68 86 57 0D 00 68  ð.À=.M..A.h.W..h
00BAF98D  88 4E 0D 00 E8 1A 00 00 00 89 45 F8 68 FA 8B 34  .N..è.....Eøhú.4
00BAF99D  00 68 88 4E 0D 00 E8 08 00 00 00 89 45 CC E9 B5  .h.N..è.....EÌéµ
00BAF9AD  00 00 00 55 8B EC 53 56 57 51 64 FF 35 30 00 00  ...U.ìSVWQdÿ50..
00BAF9BD  00 58 8B 40 0C 8B 48 0C 8B 11 8B 41 30 6A 02 8B  .X.@..H....A0j..
00BAF9CD  7D 08 57 50 E8 5B 00 00 00 85 C0 74 04 8B CA EB  }.WPè[....Àt..Êë
00BAF9DD  E7 8B 41 18 50 8B 58 3C 03 C3 8B 58 78 58 50 03  ç.A.P.X<.Ã.XxXP.
```

Figure 5 – Setting Shellcode Address

It was observed that the **shellcode** address was set by shifting the value of the address whose permission was set **10637** bytes forward.

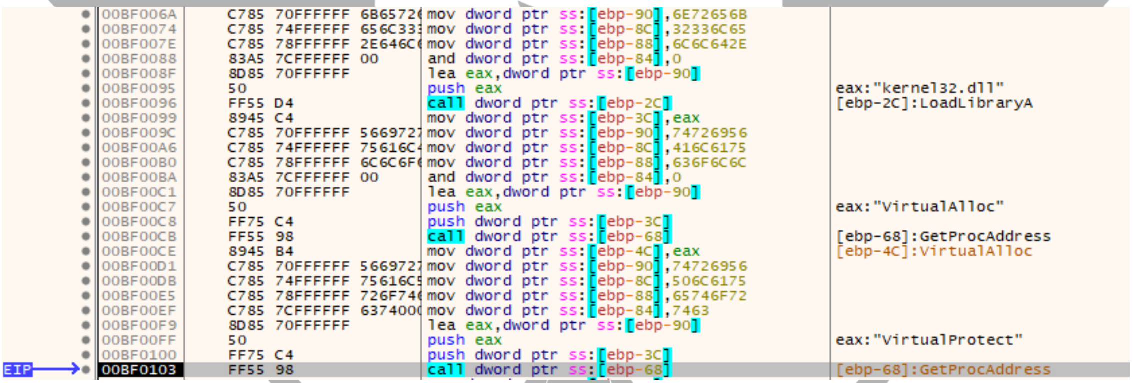


```
EIP →  0041F11D    57                push edi
       0041F11E    FF15 84104000     call dword ptr ds:[<GetCurrentDirectoryW>]
       0041F124    FF15 107B8E00     call dword ptr ds:[8E7B10]
       0041F12A    5F                pop edi
       0041F12B    33C0              xor eax,eax

dword ptr ds:[b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.008E7B10]=00BFF92D

.text:0041F124 b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.exe:$1F124 #1E524
```

```
Adres     Hex                                              ASCII
00BFF92D  E8 01 00 00 00 C3 55 8B EC 8D 45 C4 83 EC 3C 50  è....ÃU.ì.EÄ.ì<P
00BFF93D  E8 0D 00 00 00 8D 45 C4 50 E8 88 07 00 00 59 59  è.....EÄPè....YY
00BFF94D  C9 C3 55 8B EC 83 EC 38 53 56 57 8B 45 08 C6 00  ÉÃU.ì.ì8SVW.E.Æ.
00BFF95D  00 83 65 FC 00 E8 00 00 00 00 58 89 45 F0 81 45  ..eü.è....X.Eð.E
00BFF96D  F0 CB 07 00 00 8B 45 08 8B 4D F0 89 48 04 8B 45  ðË....E..Mð.H..E
00BFF97D  F0 83 C0 3D 8B 4D 08 89 41 08 68 86 57 0D 00 68  ð.À=.M..A.h.W..h
00BFF98D  88 4E 0D 00 E8 1A 00 00 00 89 45 F8 68 FA 8B 34  .N..è.....Eøhú.4
00BFF99D  00 68 88 4E 0D 00 E8 08 00 00 00 89 45 CC E9 B5  .h.N..è.....EÌéµ
00BFF9AD  00 00 00 55 8B EC 53 56 57 51 64 FF 35 30 00 00  ...U.ìSVWQdÿ50..
00BFF9BD  00 58 8B 40 0C 8B 48 0C 8B 11 8B 41 30 6A 02 8B  .X.@..H....A0j..
00BFF9CD  7D 08 57 50 E8 5B 00 00 00 85 C0 74 04 8B CA EB  }.WPè[....Àt..Êë
00BFF9DD  E7 8B 41 18 50 8B 58 3C 03 C3 8B 58 78 58 50 03  ç.A.P.X<.Ã.XxXP.
```

Figure 6 – Getting the Current Directory

Then the relevant **shellcode** is executed by jumping to this address.



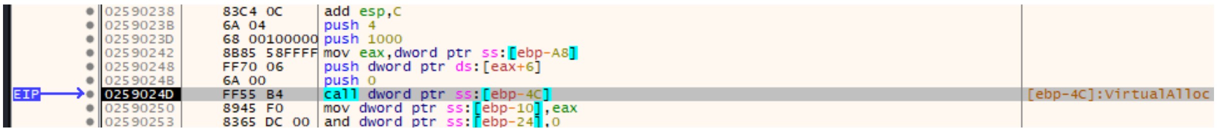
```
       00BF006A   C785 70FFFFFF 6B6572€ mov dword ptr ss:[ebp-90],6E72656B
       00BF0074   C785 74FFFFFF 656C33: mov dword ptr ss:[ebp-8C],32336C65
       00BF007E   C785 78FFFFFF 2E646C( mov dword ptr ss:[ebp-88],6C6C642E
       00BF0088   83A5 7CFFFFFF 00      and dword ptr ss:[ebp-84],0
       00BF008F   8D85 70FFFFFF         lea eax,dword ptr ss:[ebp-90]
       00BF0095   50                    push eax                          eax:"kernel32.dll"
       00BF0096   FF55 D4               call dword ptr ss:[ebp-2C]        [ebp-2C]:LoadLibraryA
       00BF0099   8945 C4               mov dword ptr ss:[ebp-3C],eax
       00BF009C   C785 70FFFFFF 566972: mov dword ptr ss:[ebp-90],74726956
       00BF00A6   C785 74FFFFFF 75616C· mov dword ptr ss:[ebp-8C],416C6175
       00BF00B0   C785 78FFFFFF 6C6C6F( mov dword ptr ss:[ebp-88],636F6C6C
       00BF00BA   83A5 7CFFFFFF 00      and dword ptr ss:[ebp-84],0
       00BF00C1   8D85 70FFFFFF         lea eax,dword ptr ss:[ebp-90]
       00BF00C7   50                    push eax                          eax:"VirtualAlloc"
       00BF00C8   FF75 C4               push dword ptr ss:[ebp-3C]
       00BF00CB   FF55 98               call dword ptr ss:[ebp-68]        [ebp-68]:GetProcAddress
       00BF00CE   8945 B4               mov dword ptr ss:[ebp-4C],eax     [ebp-4C]:VirtualAlloc
       00BF00D1   C785 70FFFFFF 566972: mov dword ptr ss:[ebp-90],74726956
       00BF00DB   C785 74FFFFFF 75616C! mov dword ptr ss:[ebp-8C],506C6175
       00BF00E5   C785 78FFFFFF 726F74( mov dword ptr ss:[ebp-88],65746F72
       00BF00EF   C785 7CFFFFFF 637400( mov dword ptr ss:[ebp-84],7463
       00BF00F9   8D85 70FFFFFF         lea eax,dword ptr ss:[ebp-90]
       00BF00FF   50                    push eax                          eax:"VirtualProtect"
       00BF0100   FF75 C4               push dword ptr ss:[ebp-3C]
EIP →  00BF0103   FF55 98               call dword ptr ss:[ebp-68]        [ebp-68]:GetProcAddress
```

Figure 7 - API Resolving

When the examination continues, it is observed that **API Resolving** operation is performed using **GetProcAddress**. Related API Calls are resolved at runtime to be used later.

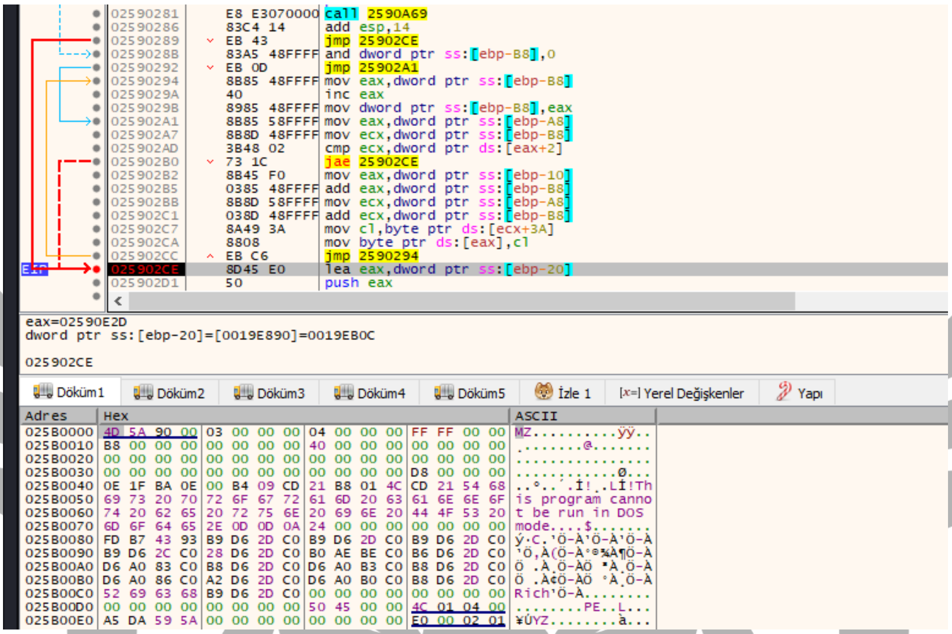
API calls resolved respectively with API Resolving:

- VirtualAlloc
- VirtualProtect
- VirtualFree
- GetVersionExA
- TerminateProcess
- ExitProcess
- SetErrorMode



```
EIP  02590238  83C4 0C       add esp,C
     0259023B  6A 04         push 4
     0259023D  68 00100000   push 1000
     02590242  8B85 58FFFF   mov eax,dword ptr ss:[ebp-A8]
     02590248  FF70 06       push dword ptr ds:[eax+6]
     0259024B  6A 00         push 0
EIP  0259024D  FF55 B4       call dword ptr ss:[ebp-4C]                [ebp-4C]:VirtualAlloc
     02590250  8945 F0       mov dword ptr ss:[ebp-10],eax
     02590253  8365 DC 00    and dword ptr ss:[ebp-24],0
```

Figure 8 – Allocating Memory Space

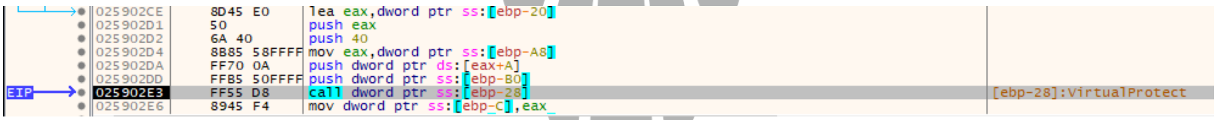


```
02590281    E8 E3070000  call 2590A69
02590286    83C4 14      add esp,14
02590289  ˅ EB 43        jmp 25902CE
0259028B    83A5 48FFFF  and dword ptr ss:[ebp-B8],0
02590292  ˅ EB 0D        jmp 25902A1
02590294    8B85 48FFFF  mov eax,dword ptr ss:[ebp-B8]
0259029A    40           inc eax
0259029B    8985 48FFFF  mov dword ptr ss:[ebp-B8],eax
025902A1    8B85 58FFFF  mov eax,dword ptr ss:[ebp-A8]
025902A7    8B8D 48FFFF  mov ecx,dword ptr ss:[ebp-B8]
025902AD    3B48 02      cmp ecx,dword ptr ds:[eax+2]
025902B0  ˅ 73 1C        jae 25902CE
025902B2    8B45 F0      mov eax,dword ptr ss:[ebp-10]
025902B5    0385 48FFFF  add eax,dword ptr ss:[ebp-B8]
025902BB    8B8D 58FFFF  mov ecx,dword ptr ss:[ebp-A8]
025902C1    038D 48FFFF  add ecx,dword ptr ss:[ebp-B8]
025902C7    8A49 3A      mov cl,byte ptr ds:[ecx+3A]
025902CA    8808         mov byte ptr ds:[eax],cl
025902CC  ^ EB C6        jmp 2590294
EIP 025902CE  8D45 E0    lea eax,dword ptr ss:[ebp-20]
025902D1    50           push eax

eax=02590E2D
dword ptr ss:[ebp-20]=[0019E890]=0019EB0C

025902CE
```

```
Adres     Hex                                              ASCII
025B0000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
025B0010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
025B0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
025B0030  00 00 00 00 00 00 00 00 00 00 00 00 D8 00 00 00  ............Ø...
025B0040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
025B0050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
025B0060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
025B0070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
025B0080  FD B7 43 93 B9 D6 2D C0 B9 D6 2D C0 B9 D6 2D C0  ý·C.¹Ö-À¹Ö-À¹Ö-À
025B0090  B9 D6 2C C0 28 D6 2D C0 B0 AE BE C0 B6 D6 2D C0  ¹Ö,À(Ö-À°®¾À¶Ö-À
025B00A0  D6 A0 83 C0 B8 D6 2D C0 D6 A0 B3 C0 B8 D6 2D C0  Ö .À¸Ö-ÀÖ ³À¸Ö-À
025B00B0  D6 A0 86 C0 A2 D6 2D C0 D6 A0 B0 C0 B8 D6 2D C0  Ö .À¢Ö-ÀÖ °À¸Ö-À
025B00C0  52 69 63 68 B9 D6 2D C0 00 00 00 00 00 00 00 00  Rich¹Ö-À........
025B00D0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
025B00E0  A5 DA 59 5A 00 00 00 00 00 00 00 00 E0 00 02 01  ¥ÚYZ........à...
```

Figure 9 – Extracting Executable File in Memory

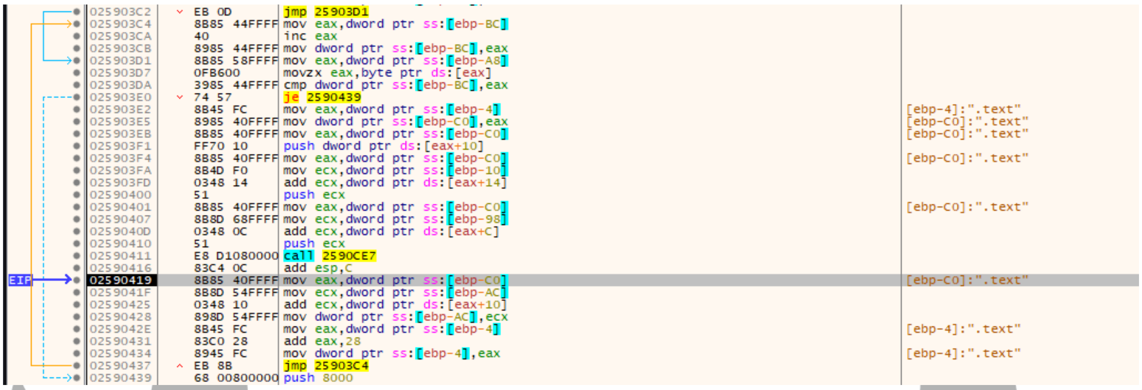
Continuing the analysis, it can be seen that **VirtualAlloc** is used to allocate memory space and an executable file is transferred to the allocated memory.



```
     025902CE  8D45 E0       lea eax,dword ptr ss:[ebp-20]
     025902D1  50            push eax
     025902D2  6A 40         push 40
     025902D4  8B85 58FFFF   mov eax,dword ptr ss:[ebp-A8]
     025902DA  FF70 0A       push dword ptr ds:[eax+A]
     025902DD  FFB5 50FFFF   push dword ptr ss:[ebp-B0]
EIP  025902E3  FF55 D8       call dword ptr ss:[ebp-28]                [ebp-28]:VirtualProtect
     025902E6  8945 F4       mov dword ptr ss:[ebp-C],eax
```

Figure 10 - Bellekteki Adrese Yazma İzni Veriliyor

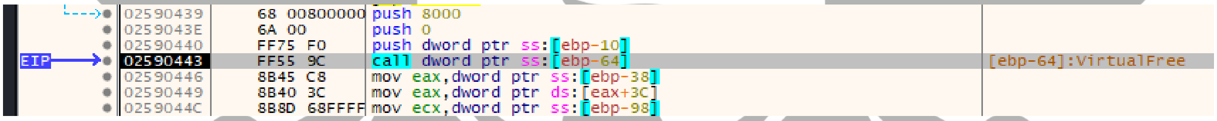
The malware grants **PAGE_EXECUTE_READWRITE** permission to the address set with **VirtualProtect**.



```
025903C2  EB 0D          jmp 25903D1
025903C4  8B85 44FFFF    mov eax,dword ptr ss:[ebp-BC]
025903CA  40             inc eax
025903CB  8985 44FFFF    mov dword ptr ss:[ebp-BC],eax
025903D1  8B85 58FFFF    mov eax,dword ptr ss:[ebp-A8]
025903D7  0FB600         movzx eax,byte ptr ds:[eax]
025903DA  3985 44FFFF    cmp dword ptr ss:[ebp-BC],eax
025903E0  74 57          je 2590439
025903E2  8B45 FC        mov eax,dword ptr ss:[ebp-4]               [ebp-4]:".text"
025903E5  8985 40FFFF    mov dword ptr ss:[ebp-C0],eax              [ebp-C0]:".text"
025903EB  8B85 40FFFF    mov eax,dword ptr ss:[ebp-C0]              [ebp-C0]:".text"
025903F1  FF70 10        push dword ptr ds:[eax+10]
025903F4  8B85 40FFFF    mov eax,dword ptr ss:[ebp-C0]              [ebp-C0]:".text"
025903FA  8B4D F0        mov ecx,dword ptr ss:[ebp-10]
025903FD  0348 14        add ecx,dword ptr ds:[eax+14]
02590400  51             push ecx
02590401  8B85 40FFFF    mov eax,dword ptr ss:[ebp-C0]              [ebp-C0]:".text"
02590407  8B8D 68FFFF    mov ecx,dword ptr ss:[ebp-98]
0259040D  0348 0C        add ecx,dword ptr ds:[eax+C]
02590410  51             push ecx
02590411  E8 D1080000    call 2590CE7
02590416  83C4 0C        add esp,C
EIP 02590419  8B85 40FFFF    mov eax,dword ptr ss:[ebp-C0]          [ebp-C0]:".text"
0259041F  8B8D 54FFFF    mov ecx,dword ptr ss:[ebp-AC]
02590425  0348 10        add ecx,dword ptr ds:[eax+10]
02590428  898D 54FFFF    mov dword ptr ss:[ebp-AC],ecx
0259042E  8B45 FC        mov eax,dword ptr ss:[ebp-4]               [ebp-4]:".text"
02590431  83C0 28        add eax,28
02590434  8945 FC        mov dword ptr ss:[ebp-4],eax               [ebp-4]:".text"
02590437  EB 8B          jmp 25903C4
02590439  68 00800000    push 8000
```

Figure 11 - Self Modifying Process

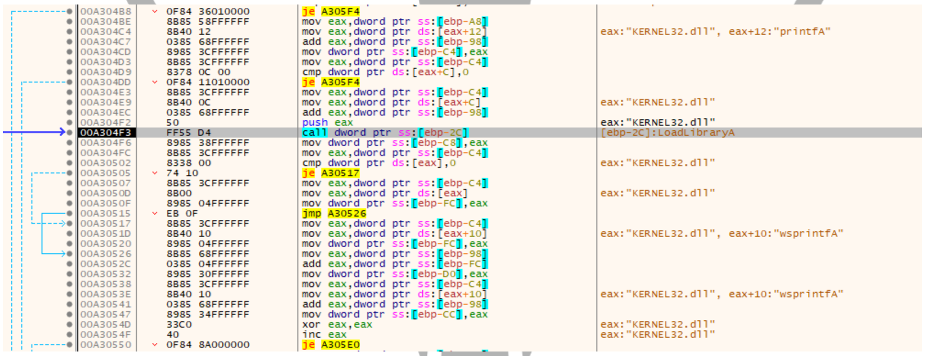
It was found that the malware performs **self-modifying** by setting its own sections as **text**, **rdata**, **data** and **reloc** sections of the relevant executable file, respectively.



```
02590439  68 00800000    push 8000
0259043E  6A 00          push 0
02590440  FF75 F0        push dword ptr ss:[ebp-10]
EIP 02590443  FF55 9C        call dword ptr ss:[ebp-64]             [ebp-64]:VirtualFree
02590446  8B45 C8        mov eax,dword ptr ss:[ebp-38]
02590449  8B40 3C        mov eax,dword ptr ds:[eax+3C]
0259044C  8B8D 68FFFF    mov ecx,dword ptr ss:[ebp-98]
```

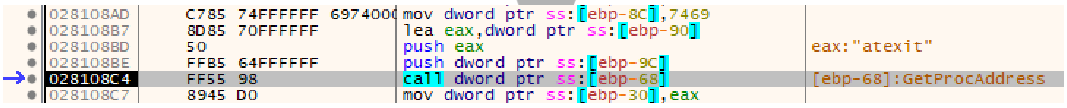
Figure 12 – Freeing Up Memory Space

After the partitions are set, the executable’s space in memory is freed.



```
00A304B8  0F84 36010000   je A305F4
00A304BE  8B85 58FFFFFF   mov eax,dword ptr ss:[ebp-A8]
00A304C4  8B40 12         mov eax,dword ptr ds:[eax+12]      eax:"KERNEL32.dll", eax+12:"printfA"
00A304C7  0385 68FFFFFF   add eax,dword ptr ss:[ebp-98]
00A304CD  8985 3CFFFFFF   mov dword ptr ss:[ebp-C4],eax
00A304D3  8B85 3CFFFFFF   mov eax,dword ptr ss:[ebp-C4]
00A304D9  8378 0C 00      cmp dword ptr ds:[eax+C],0
00A304DD  0F84 11010000   je A305F4
00A304E3  8B85 3CFFFFFF   mov eax,dword ptr ss:[ebp-C4]
00A304E9  8B40 0C         mov eax,dword ptr ds:[eax+C]       eax:"KERNEL32.dll"
00A304EC  0385 68FFFFFF   add eax,dword ptr ss:[ebp-98]
00A304F2  50              push eax                           eax:"KERNEL32.dll"
00A304F3  FF55 D4         call dword ptr ss:[ebp-2C]         [ebp-2C]:LoadLibraryA
00A304F6  8985 38FFFFFF   mov dword ptr ss:[ebp-C8],eax
00A304FC  8B85 3CFFFFFF   mov eax,dword ptr ss:[ebp-C4]
00A30502  8338 00         cmp dword ptr ds:[eax],0           eax:"KERNEL32.dll"
00A30505  74 10           je A30517
00A30507  8B85 3CFFFFFF   mov eax,dword ptr ss:[ebp-C4]
00A3050D  8B00            mov eax,dword ptr ds:[eax]         eax:"KERNEL32.dll"
00A3050F  8985 04FFFFFF   mov dword ptr ss:[ebp-FC],eax
00A30515  EB 0F           jmp A30526
00A30517  8B85 3CFFFFFF   mov eax,dword ptr ss:[ebp-C4]
00A3051D  8B40 10         mov eax,dword ptr ds:[eax+10]      eax:"KERNEL32.dll", eax+10:"wsprintfA"
00A30520  8985 04FFFFFF   mov dword ptr ss:[ebp-FC],eax
00A30526  8B85 68FFFFFF   mov eax,dword ptr ss:[ebp-98]
00A3052C  0385 04FFFFFF   add eax,dword ptr ss:[ebp-FC]
00A30532  8985 30FFFFFF   mov dword ptr ss:[ebp-D0],eax
00A30538  8B85 3CFFFFFF   mov eax,dword ptr ss:[ebp-C4]
00A3053E  8B40 10         mov eax,dword ptr ds:[eax+10]      eax:"KERNEL32.dll", eax+10:"wsprintfA"
00A30541  0385 68FFFFFF   add eax,dword ptr ss:[ebp-98]
00A30547  8985 34FFFFFF   mov dword ptr ss:[ebp-CC],eax
00A3054D  33C0            xor eax,eax                        eax:"KERNEL32.dll"
00A3054F  40              inc eax                            eax:"KERNEL32.dll"
00A30550  0F84 8A000000   je A305E0
```

Figure 13 – Importing DLL Files



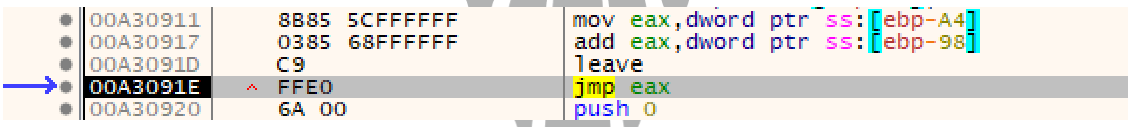
```
028108AD  C785 74FFFFFF 6974000  mov dword ptr ss:[ebp-8C],7469
028108B7  8D85 70FFFFFF          lea eax,dword ptr ss:[ebp-90]
028108BD  50                     push eax                          eax:"atexit"
028108BE  FFB5 64FFFFFF          push dword ptr ss:[ebp-9C]
028108C4  FF55 98                call dword ptr ss:[ebp-68]        [ebp-68]:GetProcAddress
028108C7  8945 D0                mov dword ptr ss:[ebp-30],eax
```

Figure 14 – Receiving API Calls

It is observed that the malware perform **Dynamic API Resolution** by retrieving certain API functions from the relevant DLL files with **GetProcAddress**.

Some important API Calls resolved with the Dynamic API Resolution technique:

WS2_32.dll	**Kernel32.dll**	**ADVAPI32.dll**	**SHELL32.dll**
ioctlsocket	GetCurrentProcess	CreateProcessWithLogonW	ShellExecuteA
send	WriteFile	RegCreatekeyExA	ShellExecuteExW
connect	ReadFile	StartServiceCtrlDispatcherA	
setsockopt	CreateFileA	RegisterServiceCtrlHandlerA	
bind	LoadLibraryA	SetServiceStatus	
accept	GetEnvironmentVariableA	RegDeleteValueA	
getsockname	DeleteFileA	RegSetValueExA	
htonl	WriteProcessMemory	ReqQueryValueExA	
gethostname	VirtualAlloc	RegEnumKeyA	
socket	VirtualAllocEx	RegOpenKeyExA	
select	GetProcAddress	RegEnumValueA	
recv	CreateProcessA	GetUserNameW	
htons	CreateFileW	LookupAccountNameW	
sendto	ResumeThread	LookupAccountNameA	
gethostbyaddr	SetThreadContext	GetUserNameA	
gethostbyname	CreateThread	RegCloseKey	



```
00A30911   8B85 5CFFFFFF   mov eax,dword ptr ss:[ebp-A4]
00A30917   0385 68FFFFFF   add eax,dword ptr ss:[ebp-98]
00A3091D   C9              leave
00A3091E ^ FFE0            jmp eax
00A30920   6A 00           push 0
```

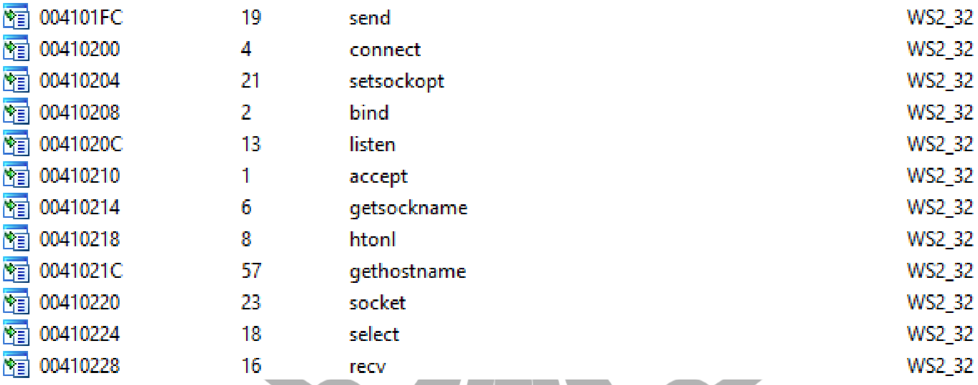
Figure 15 – Making a Jump to the Related Executable File

The malware then moves to the **text section** updated with the **jmp eax** instruction and executes the code of the corresponding executable.

Stage2 Analysis

Name	ce88300e4893d0317ee89dbddec08557537af9e8bd88989b51a962fcf1620da2.exe
MD5	95fc3460859b033780774fc0d5ec768d
SHA256	ce88300e4893d0317ee89dbddec08557537af9e8bd88989b51a962fcf1620da2
File Type	PE32/EXE

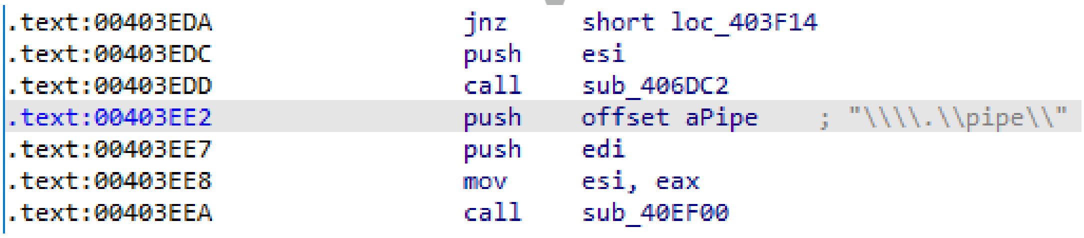
Static Analysis



004101FC	19	send	WS2_32
00410200	4	connect	WS2_32
00410204	21	setsockopt	WS2_32
00410208	2	bind	WS2_32
0041020C	13	listen	WS2_32
00410210	1	accept	WS2_32
00410214	6	getsockname	WS2_32
00410218	8	htonl	WS2_32
0041021C	57	gethostname	WS2_32
00410220	23	socket	WS2_32
00410224	18	select	WS2_32
00410228	16	recv	WS2_32

Figure 16 – API Calls Belonging to WS2_32.dll

The malware appears to import API calls belonging to **WS2_32.dll** such as **socket**, **recv**, **listen**. These APIs are known to be used to communicate with the C2 server over the **TCP/UDP** protocol.

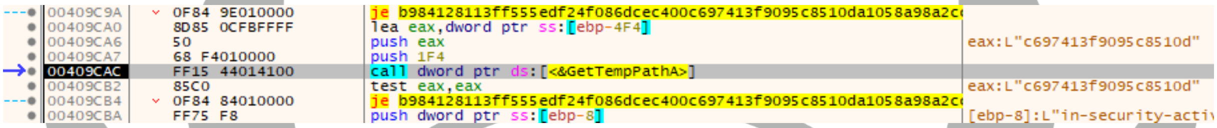


```
.text:00403EDA                 jnz     short loc_403F14
.text:00403EDC                 push    esi
.text:00403EDD                 call    sub_406DC2
.text:00403EE2                 push    offset aPipe    ; "\\\\.\\pipe\\"
.text:00403EE7                 push    edi
.text:00403EE8                 mov     esi, eax
.text:00403EEA                 call    sub_40EF00
```

Figure 17 - Pipeline Observed

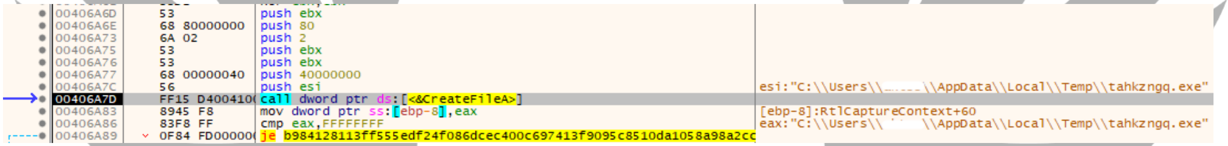
The **pipe** string is prominent in the malware. A pipe is a communication mechanism used to allow the output produced by one process to be received by another process.

Dynamic Analysis



```
00409C9A   0F84 9E010000   je b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2c
00409CA0   8D85 0CFBFFFF   lea eax,dword ptr ss:[ebp-4F4]
00409CA6   50              push eax                                                         eax:L"c697413f9095c8510d"
00409CA7   68 F4010000     push 1F4
00409CAC   FF15 44014100   call dword ptr ds:[<&GetTempPathA>]
00409CB2   85C0            test eax,eax                                                     eax:L"c697413f9095c8510d"
00409CB4   0F84 84010000   je b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2c
00409CBA   FF75 F8         push dword ptr ss:[ebp-8]                                        [ebp-8]:L"in-security-acti
```

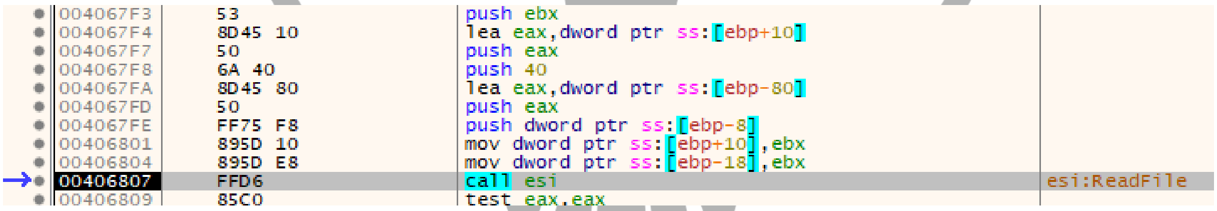
Figure 18 – Getting Temp Directory



```
00406A6D   53              push ebx
00406A6E   68 80000000     push 80
00406A73   6A 02           push 2
00406A75   53              push ebx
00406A76   53              push ebx
00406A77   68 00000040     push 40000000
00406A7C   56              push esi                  esi:"C:\\Users\\ \\AppData\\Local\\Temp\\tahkzngq.exe"
00406A7D   FF15 D400410    call dword ptr ds:[<&CreateFileA>]
00406A83   8945 F8         mov dword ptr ss:[ebp-8],eax    [ebp-8]:RtlCaptureContext+60
00406A86   83F8 FF         cmp eax,FFFFFFFF          eax:"C:\\Users\\ \\AppData\\Local\\Temp\\tahkzngq.exe"
00406A89   0F84 FD00000    je b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc
```

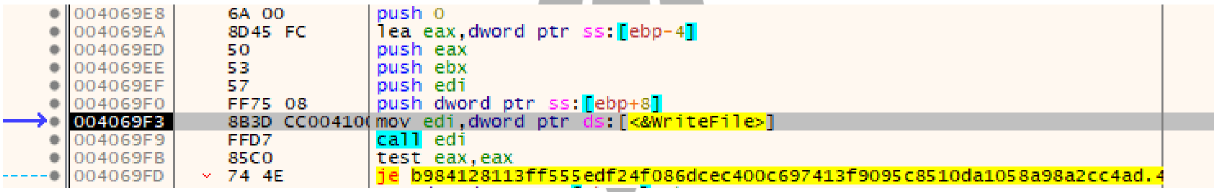
Figure 19 – Creating File in Temp Directory

The malware first takes the temp directory and creates a file.



```
004067F3   53          push ebx
004067F4   8D45 10     lea eax,dword ptr ss:[ebp+10]
004067F7   50          push eax
004067F8   6A 40       push 40
004067FA   8D45 80     lea eax,dword ptr ss:[ebp-80]
004067FD   50          push eax
004067FE   FF75 F8     push dword ptr ss:[ebp-8]
00406801   895D 10     mov dword ptr ss:[ebp+10],ebx
00406804   895D E8     mov dword ptr ss:[ebp-18],ebx
00406807   FFD6        call esi                              esi:ReadFile
00406809   85C0        test eax,eax
```

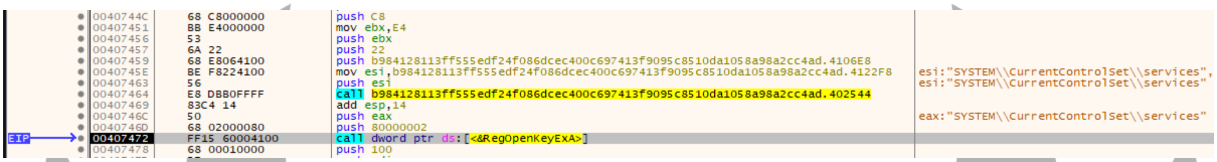
Figure 20 – Self-Reading Process



```
004069E8   6A 00          push 0
004069EA   8D45 FC        lea eax,dword ptr ss:[ebp-4]
004069ED   50             push eax
004069EE   53             push ebx
004069EF   57             push edi
004069F0   FF75 08        push dword ptr ss:[ebp+8]
004069F3   8B3D CC00410   mov edi,dword ptr ds:[<&WriteFile>]
004069F9   FFD7           call edi
004069FB   85C0           test eax,eax
004069FD   74 4E          je b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.4
```

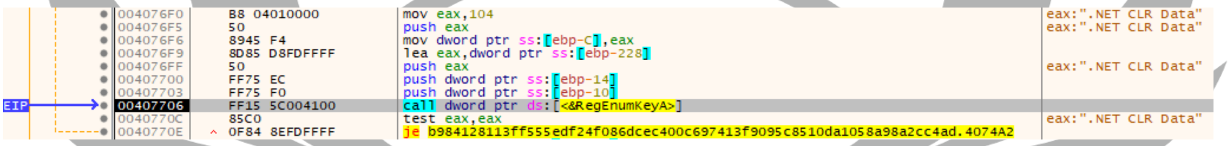
Figure 21 – Writes Itself to the Related File

The malware is then observed to write itself into this file.



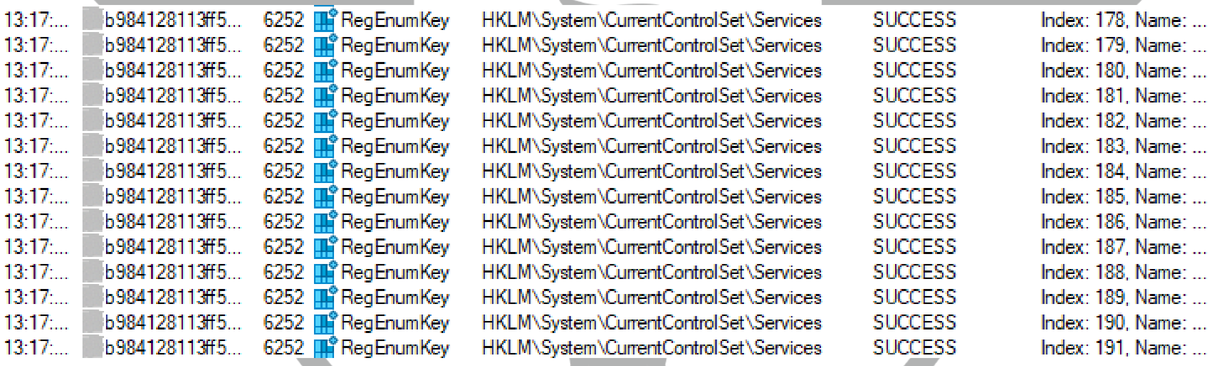
```
0040744C  68 C8000000  push C8
00407451  BB E4000000  mov ebx,E4
00407456  53           push ebx
00407457  6A 22        push 22
00407459  68 E8064100  push b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.4106E8
0040745E  BE F8224100  mov esi,b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.4122F8   esi:"SYSTEM\\CurrentControlSet\\services",
                                                                                                          esi:"SYSTEM\\CurrentControlSet\\services"
00407463  56           push esi
00407464  E8 DBB0FFFF  call b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.402544
00407469  83C4 14      add esp,14
0040746C  50           push eax                                                                           eax:"SYSTEM\\CurrentControlSet\\services"
0040746D  68 02000080  push 80000002
EIP → 00407472  FF15 60004100  call dword ptr ds:[<&RegOpenKeyExA>]
00407478  68 00010000  push 100
```

Figure 22 – Receiving Services



```
004076F0  B8 04010000  mov eax,104                          eax:".NET CLR Data"
004076F5  50           push eax                             eax:".NET CLR Data"
004076F6  8945 F4      mov dword ptr ss:[ebp-C],eax
004076F9  8D85 D8FDFFFF  lea eax,dword ptr ss:[ebp-228]
004076FF  50           push eax                             eax:".NET CLR Data"
00407700  FF75 EC      push dword ptr ss:[ebp-14]
00407703  FF75 F0      push dword ptr ss:[ebp-10]
EIP → 00407706  FF15 5C004100  call dword ptr ds:[<&RegEnumKeyA>]
0040770C  85C0         test eax,eax                         eax:".NET CLR Data"
0040770E  0F84 8EFDFFFF  je b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.4074A2
```

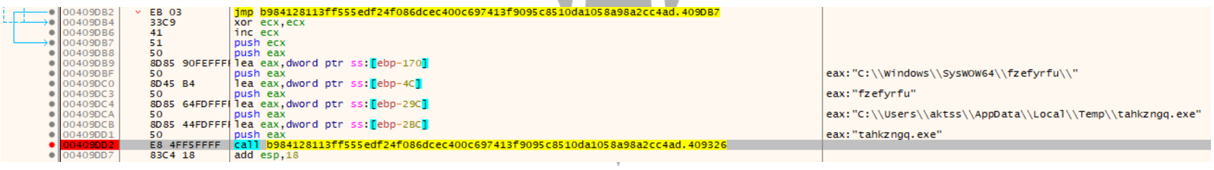
Figure 23 – Checking Services



```
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 178, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 179, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 180, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 181, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 182, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 183, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 184, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 185, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 186, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 187, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 188, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 189, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 190, Name: ...
13:17:...  b984128113ff5...  6252  RegEnumKey  HKLM\System\CurrentControlSet\Services  SUCCESS  Index: 191, Name: ...
```

Figure 24 – Procmon Display of Controlled Services

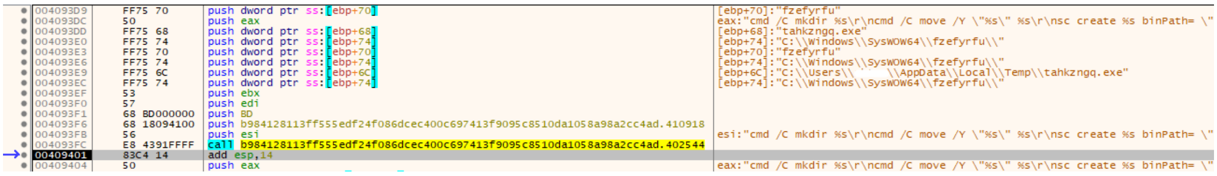
By enumerating services, the malware checks whether a specific service has been created before.



```
00409DB2  EB 03          jmp b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.409DB7
00409DB4  33C9           xor ecx,ecx
00409DB6  41             inc ecx
00409DB7  51             push ecx
00409DB8  50             push eax
00409DB9  8D85 90FEFFFF  lea eax,dword ptr ss:[ebp-170]
00409DBF  50             push eax                        eax:"C:\\Windows\\SysWOW64\\fzefyrfu\\"
00409DC0  8D45 B4        lea eax,dword ptr ss:[ebp-4C]
00409DC3  50             push eax                        eax:"fzefyrfu"
00409DC4  8D85 64FDFFFF  lea eax,dword ptr ss:[ebp-29C]
00409DCA  50             push eax                        eax:"C:\\Users\\aktss\\AppData\\Local\\Temp\\tahkzngq.exe"
00409DCB  8D85 44FDFFFF  lea eax,dword ptr ss:[ebp-2BC]
00409DD1  50             push eax                        eax:"tahkzngq.exe"
00409DD2  E8 4FF5FFFF    call b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.409326
00409DD7  83C4 18        add esp,18
```

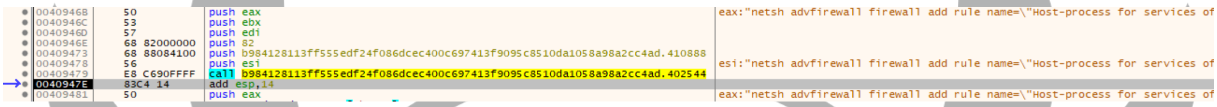
Figure 25 - İlgili Değerler Fonksiyona Veriliyor

The malware was observed to give the function the name and path of the file it had previously created and written itself in the temp directory, the **fzefyrfu** string and the **SysWOW64** directory.



```
004093D9  FF75 70      push dword ptr ss:[ebp+70]     [ebp+70]:"fzefyrfu"
004093DC  50           push eax                       eax:"cmd /C mkdir %s\r\ncmd /C move /Y \"%s\" %s\r\nsc create %s binPath= \"
004093DD  FF75 68      push dword ptr ss:[ebp+68]     [ebp+68]:"tahkzngq.exe"
004093E0  FF75 74      push dword ptr ss:[ebp+74]     [ebp+74]:"C:\\Windows\\SysWOW64\\fzefyrfu\\"
004093E3  FF75 70      push dword ptr ss:[ebp+70]     [ebp+70]:"fzefyrfu"
004093E6  FF75 74      push dword ptr ss:[ebp+74]     [ebp+74]:"C:\\Windows\\SysWOW64\\fzefyrfu\\"
004093E9  FF75 6C      push dword ptr ss:[ebp+6C]     [ebp+6C]:"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\tahkzngq.exe"
004093EC  FF75 74      push dword ptr ss:[ebp+74]     [ebp+74]:"C:\\Windows\\SysWOW64\\fzefyrfu\\"
004093EF  53           push ebx
004093F0  57           push edi
004093F1  68 BD000000  push BD
004093F6  68 18094100  push b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.410918
004093FB  56           push esi                       esi:"cmd /C mkdir %s\r\ncmd /C move /Y \"%s\" %s\r\nsc create %s binPath= \"
004093FC  E8 4391FFFF  call b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.402544
00409401  83C4 14      add esp,14
00409404  50           push eax                       eax:"cmd /C mkdir %s\r\ncmd /C move /Y \"%s\" %s\r\nsc create %s binPath= \"
```

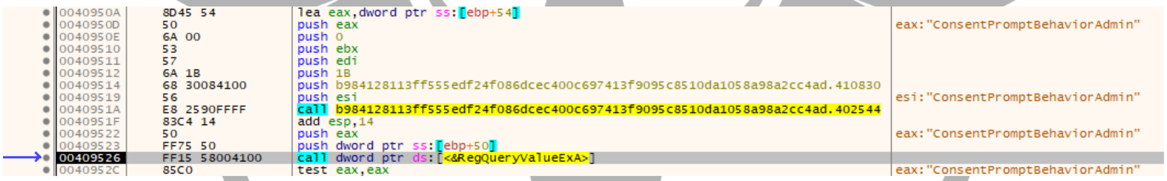
Figure 26 – Parse Function



```
0040946B  50           push eax      eax:"netsh advfirewall firewall add rule name=\"Host-process for services of
0040946C  53           push ebx
0040946D  57           push edi
0040946E  68 82000000  push 82
00409473  68 88084100  push b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.410888
00409478  56           push esi      esi:"netsh advfirewall firewall add rule name=\"Host-process for services of
00409479  E8 C690FFFF  call b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.402544
0040947E  83C4 14      add esp,14
00409481  50           push eax      eax:"netsh advfirewall firewall add rule name=\"Host-process for services of
```

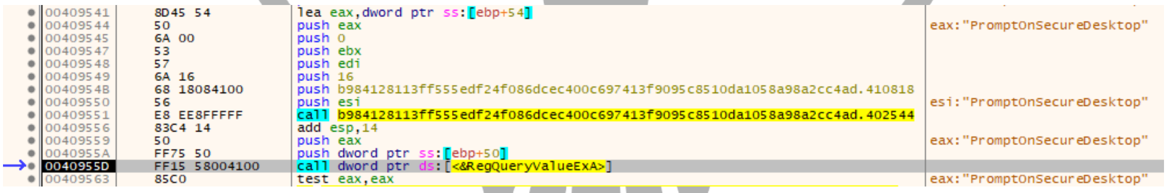
Figure 27 – Performing Parse Operation

In the related function, it is observed that terminal commands are combined with the given strings and character strings are set.



```
0040950A  8D45 54      lea eax,dword ptr ss:[ebp+54]
0040950D  50           push eax                      eax:"ConsentPromptBehaviorAdmin"
0040950E  6A 00        push 0
00409510  53           push ebx
00409511  57           push edi
00409512  6A 1B        push 1B
00409514  68 30084100  push b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.410830
00409519  56           push esi                      esi:"ConsentPromptBehaviorAdmin"
0040951A  E8 2590FFFF  call b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.402544
0040951F  83C4 14      add esp,14
00409522  50           push eax                      eax:"ConsentPromptBehaviorAdmin"
00409523  FF75 50      push dword ptr ss:[ebp+50]
00409526  FF15 58004100  call dword ptr ds:[<&RegQueryValueExA>]
0040952C  85C0         test eax,eax                  eax:"ConsentPromptBehaviorAdmin"
```

Figure 28 - ConsentPromptBehaviorAdmin Anahtarının Değiştirilmesi

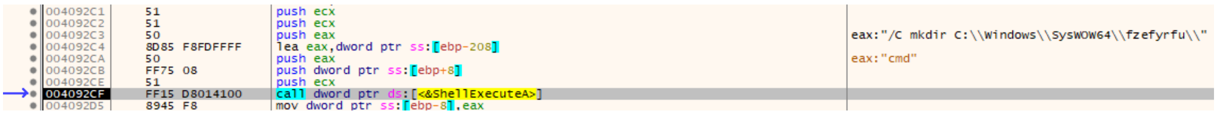


```
00409541  8D45 54      lea eax,dword ptr ss:[ebp+54]
00409544  50           push eax                      eax:"PromptOnSecureDesktop"
00409545  6A 00        push 0
00409547  53           push ebx
00409548  57           push edi
00409549  6A 16        push 16
0040954B  68 18084100  push b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.410818
00409550  56           push esi                      esi:"PromptOnSecureDesktop"
00409551  E8 EE8FFFFF  call b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.402544
00409556  83C4 14      add esp,14
00409559  50           push eax                      eax:"PromptOnSecureDesktop"
0040955A  FF75 50      push dword ptr ss:[ebp+50]
0040955D  FF15 58004100  call dword ptr ds:[<&RegQueryValueExA>]
00409563  85C0         test eax,eax                  eax:"PromptOnSecureDesktop"
```

Figure 29 - PromptOnSecureDesktop Anahtarının Değiştirilmesi

The malware was found to attempt to disable UAC control by modifying the **ConsentPromptBehaviorAdmin** and **PromptOnSecureDesktop** switches before executing commands.

User Account Control (UAC) is used to prevent unauthorized changes to the computer. **ConsentPromptBehaviorAdmin** determines whether to display a confirmation dialog to the user for operations that require administrator permission, while **PromptOnSecureDesktop** determines whether to display this dialog on a secure desktop.



```
004092C1   51              push ecx
004092C2   51              push ecx
004092C3   50              push eax                                eax:"/C mkdir C:\\Windows\\SysWOW64\\fzefyrfu\\"
004092C4   8D85 F8FDFFFF   lea eax,dword ptr ss:[ebp-208]
004092CA   50              push eax                                eax:"cmd"
004092CB   FF75 08         push dword ptr ss:[ebp+8]
004092CE   51              push ecx
004092CF   FF15 D8014100   call dword ptr ds:[<&ShellExecuteA>]
004092D5   8945 F8         mov dword ptr ss:[ebp-8],eax
```

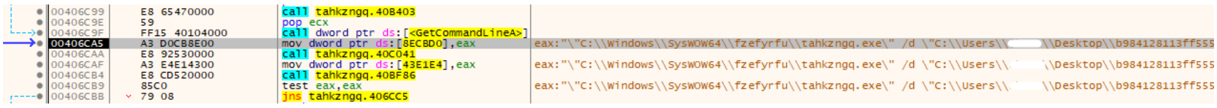
Figure 30 – Executing Commands

The relevant commands are executed respectively.

When the commands are examined, it is seen that it moves itself to the **fzefyrfu** folder in **SysWOW64** and creates a service names **fzefyrfu**. The service starts itself with the **/d** parameter and sets the start=auto parameter to start the service automatically.

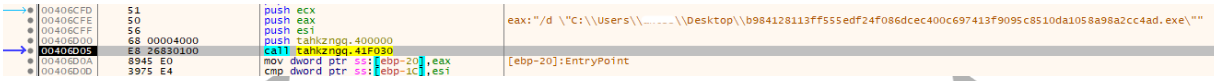
It is then seen that the malware allows **svchosts.exe** traffic by adding a firewall rule with the **netsh** command.

```
cmd /C mkdir C:\\Windows\\SysWOW64\\fzefyrfu\\\r

cmd /C move /Y \"C:\\Users\\aktss\\AppData\\Local\\Temp\\tahkzngq.exe\"
C:\\Windows\\SysWOW64\\fzefyrfu\\

sc create fzefyrfu binPath=
\"C:\\Windows\\SysWOW64\\fzefyrfu\\tahkzngq.exe
/d\\\"C:\\Users\\user\\Desktop\\b984128113ff555edf24f086dcec400c697413f90
95c8510da1058a98a2cc4ad.exe\\\"\" type= own start= auto DisplayName=
\"wifi support\"

sc description fzefyrfu \"wifi internet conection\"

sc start fzefyrfu

netsh advfirewall firewall add rule name=\"Host-process for services of
Windows\" dir=in action=allow
program=\"C:\\Windows\\SysWOW64\\svchost.exe\" enable=yes>nul
```



```
00406C99    E8 65470000    call tahkzngq.40B403
00406C9E    59             pop ecx
00406C9F    FF15 40104000  call dword ptr ds:[<GetCommandLineA>]
00406CA5    A3 D0CB8E00    mov dword ptr ds:[8ECBD0],eax    eax:"\"C:\\Windows\\SysWOW64\\fzefyrfu\\tahkzngq.exe\" /d \"C:\\Users\\[redacted]\\Desktop\\b984128113ff555
00406CAA    E8 92530000    call tahkzngq.40C041
00406CAF    A3 E4E14300    mov dword ptr ds:[43E1E4],eax    eax:"\"C:\\Windows\\SysWOW64\\fzefyrfu\\tahkzngq.exe\" /d \"C:\\Users\\[redacted]\\Desktop\\b984128113ff555
00406CB4    E8 CD520000    call tahkzngq.40BF86
00406CB9    85C0           test eax,eax                     eax:"\"C:\\Windows\\SysWOW64\\fzefyrfu\\tahkzngq.exe\" /d \"C:\\Users\\[redacted]\\Desktop\\b984128113ff555
00406CBB    79 08          jns tahkzngq.406CC5
```

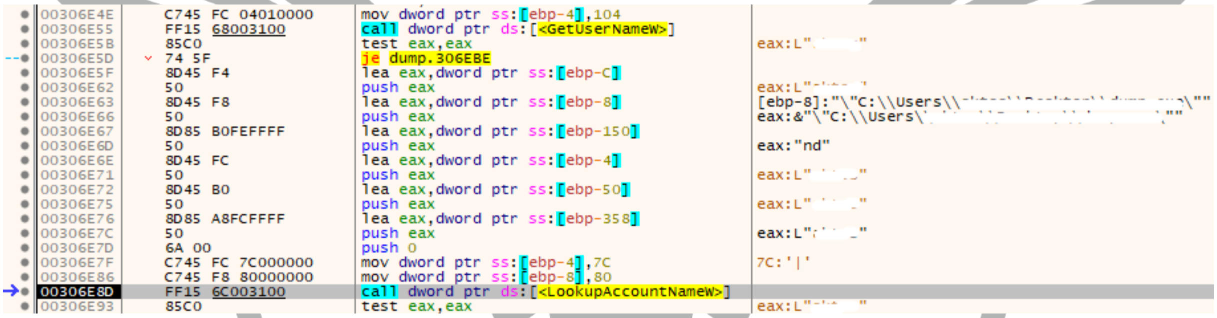
Figure 31 – Receiving Parameters



```
00406CFD    51             push ecx
00406CFE    50             push eax                         eax:"/d \"C:\\Users\\[redacted]\\Desktop\\b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.exe\""
00406CFF    56             push esi
00406D00    68 00004000    push tahkzngq.400000
00406D05    E8 26830100    call tahkzngq.41F030
00406D0A    8945 E0        mov dword ptr ss:[ebp-20],eax    [ebp-20]:EntryPoint
00406D0D    3975 E4        cmp dword ptr ss:[ebp-1C],esi
```

Figure 32 – Passing Related Paramater to Function

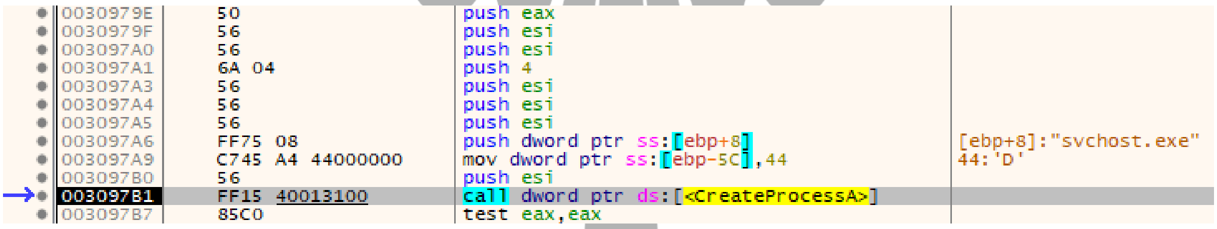
When the malware is run with the relevant parameters and the examination continues, it is observed that the malware receives the paramaters with **GetCommandLineA** and gives the values to the relevant function after performing **/d** parameter control.



```
00306E4E    C745 FC 04010000    mov dword ptr ss:[ebp-4],104
00306E55    FF15 68003100       call dword ptr ds:[<GetUserNameW>]
00306E5B    85C0                test eax,eax                      eax:L"[redacted]"
00306E5D    74 5F               je dump.306EBE
00306E5F    8D45 F4             lea eax,dword ptr ss:[ebp-C]
00306E62    50                  push eax                          eax:L"[redacted]"
00306E63    8D45 F8             lea eax,dword ptr ss:[ebp-8]      [ebp-8]:"\"C:\\Users\\[redacted]\""
00306E66    50                  push eax                          eax:&"\"C:\\Users\[redacted]""
00306E67    8D85 B0FEFFFF       lea eax,dword ptr ss:[ebp-150]
00306E6D    50                  push eax                          eax:"nd"
00306E6E    8D45 FC             lea eax,dword ptr ss:[ebp-4]
00306E71    50                  push eax                          eax:L"[redacted]"
00306E72    8D45 B0             lea eax,dword ptr ss:[ebp-50]
00306E75    50                  push eax                          eax:L"[redacted]"
00306E76    8D85 A8FCFFFF       lea eax,dword ptr ss:[ebp-358]
00306E7C    50                  push eax                          eax:L"[redacted]"
00306E7D    6A 00               push 0
00306E7F    C745 FC 7C000000    mov dword ptr ss:[ebp-4],7C       7C:'|'
00306E86    C745 F8 80000000    mov dword ptr ss:[ebp-8],80
00306E8D    FF15 6C003100       call dword ptr ds:[<LookupAccountNameW>]
00306E93    85C0                test eax,eax                      eax:L"[redacted]"
```

Figure 33 – Getting SID Value

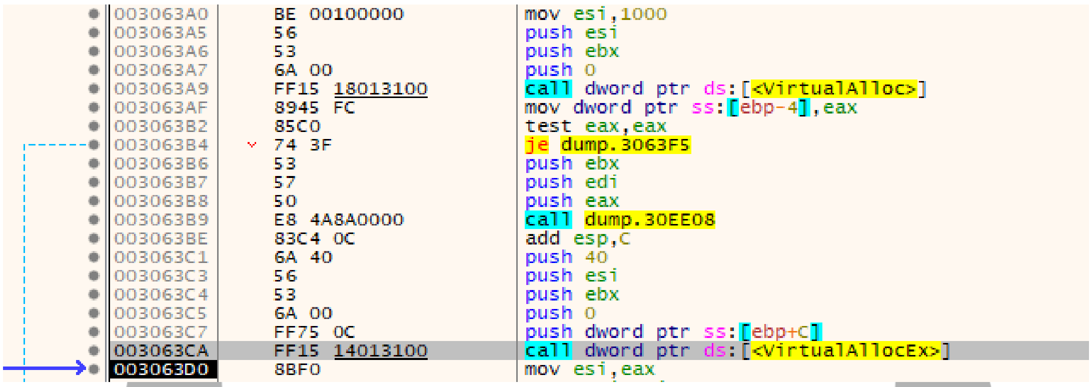
The username and **SID** of the user are retrieved.



```
0030979E    50                  push eax
0030979F    56                  push esi
003097A0    56                  push esi
003097A1    6A 04               push 4
003097A3    56                  push esi
003097A4    56                  push esi
003097A5    56                  push esi
003097A6    FF75 08             push dword ptr ss:[ebp+8]         [ebp+8]:"svchost.exe"
003097A9    C745 A4 44000000    mov dword ptr ss:[ebp-5C],44      44:'D'
003097B0    56                  push esi
003097B1    FF15 40013100       call dword ptr ds:[<CreateProcessA>]
003097B7    85C0                test eax,eax
```

Figure 34 – Running Svchost

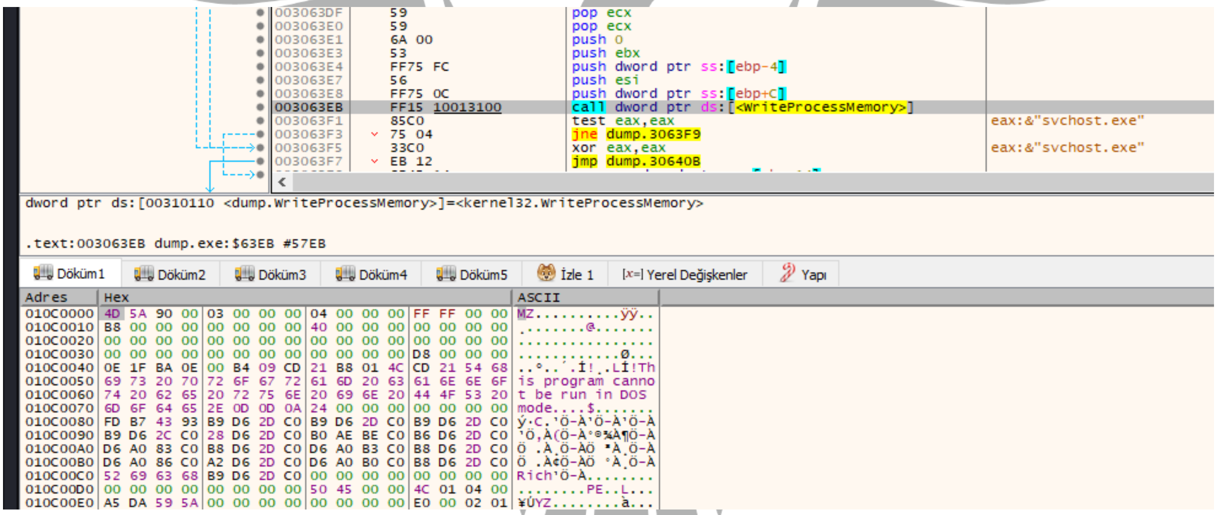
It has been observed that the malware runs the **svchost.exe** file.



```
003063A0    BE 00100000    mov esi,1000
003063A5    56             push esi
003063A6    53             push ebx
003063A7    6A 00          push 0
003063A9    FF15 18013100  call dword ptr ds:[<VirtualAlloc>]
003063AF    8945 FC        mov dword ptr ss:[ebp-4],eax
003063B2    85C0           test eax,eax
003063B4    74 3F          je dump.3063F5
003063B6    53             push ebx
003063B7    57             push edi
003063B8    50             push eax
003063B9    E8 4A8A0000    call dump.30EE08
003063BE    83C4 0C        add esp,C
003063C1    6A 40          push 40
003063C3    56             push esi
003063C4    53             push ebx
003063C5    6A 00          push 0
003063C7    FF75 0C        push dword ptr ss:[ebp+C]
003063CA    FF15 14013100  call dword ptr ds:[<VirtualAllocEx>]
003063D0    8BF0           mov esi,eax
```

Figure 35 – Allocating Memory Space

In this executed file, space is allocated using the **VirtualAlloc** and **VirtualAllocEx** API calls.

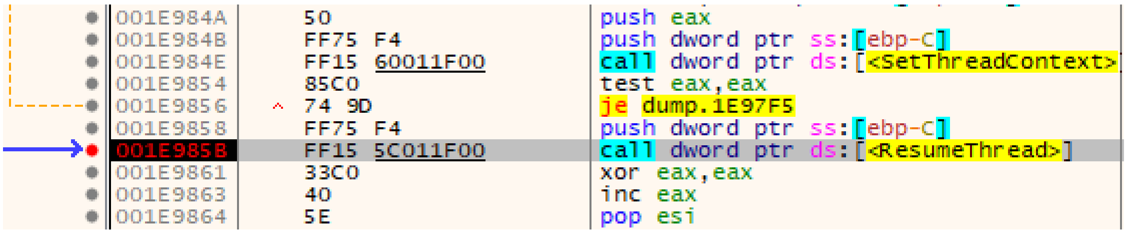


```
003063DF    59             pop ecx
003063E0    59             pop ecx
003063E1    6A 00          push 0
003063E3    53             push ebx
003063E4    FF75 FC        push dword ptr ss:[ebp-4]
003063E7    56             push esi
003063E8    FF75 0C        push dword ptr ss:[ebp+C]
003063EB    FF15 10013100  call dword ptr ds:[<WriteProcessMemory>]
003063F1    85C0           test eax,eax                                 eax:&"svchost.exe"
003063F3    75 04          jne dump.3063F9
003063F5    33C0           xor eax,eax                                  eax:&"svchost.exe"
003063F7    EB 12          jmp dump.30640B

dword ptr ds:[00310110 <dump.WriteProcessMemory>]=<kernel32.WriteProcessMemory>

.text:003063EB dump.exe:$63EB #57EB
```

```
Adres     Hex                                              ASCII
010C0000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
010C0010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
010C0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
010C0030  00 00 00 00 00 00 00 00 00 00 00 00 D8 00 00 00  ............Ø...
010C0040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
010C0050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
010C0060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
010C0070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
010C0080  FD B7 43 93 B9 D6 2D C0 B9 D6 2D C0 B9 D6 2D C0  ý·C.¹Ö-À¹Ö-À¹Ö-À
010C0090  B9 D6 2C C0 28 D6 2D C0 B0 AE BE C0 B6 D6 2D C0  ¹Ö,À(Ö-À°®¾À¶Ö-À
010C00A0  D6 A0 83 C0 B8 D6 2D C0 D6 A0 B3 C0 B8 D6 2D C0  Ö .À¸Ö-ÀÖ ³À¸Ö-À
010C00B0  D6 A0 86 C0 A2 D6 2D C0 D6 A0 B0 C0 B8 D6 2D C0  Ö .À¢Ö-ÀÖ °À¸Ö-À
010C00C0  52 69 63 68 B9 D6 2D C0 00 00 00 00 00 00 00 00  Rich¹Ö-À........
010C00D0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00  ........PE..L...
010C00E0  A5 DA 59 5A 00 00 00 00 00 00 00 00 E0 00 02 01  ¥ÚYZ........à...
```

Figure 36 – Printing Executable File to Svchost Memory

The malware was found to perform **Process Hallowing** by writing itself to this reserved location with **WriteProcessMemory**.



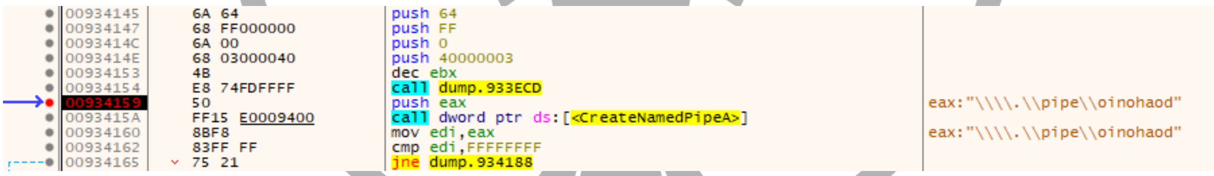
```
001E984A    50              push eax
001E984B    FF75 F4         push dword ptr ss:[ebp-C]
001E984E    FF15 60011F00   call dword ptr ds:[<SetThreadContext>
001E9854    85C0            test eax,eax
001E9856  ^ 74 9D           je dump.1E97F5
001E9858    FF75 F4         push dword ptr ss:[ebp-C]
001E985B    FF15 5C011F00   call dword ptr ds:[<ResumeThread>]
001E9861    33C0            xor eax,eax
001E9863    40              inc eax
001E9864    5E              pop esi
```

Figure 37 – Svchost Continues to Run with Updated Version

It is observed that the relevant process continues to run using **ResumeThread**.

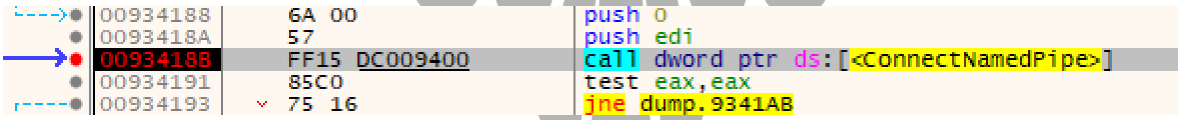
API functions called in the process of using the **Process Hallowing** technique:

- CreateProcessA
- VirtualAlloc
- VirtualAllocEx
- GetThreadContext
- SetThreadContext
- WriteProcessMemory
- ResumeThread



```
00934145    6A 64          push 64
00934147    68 FF000000    push FF
0093414C    6A 00          push 0
0093414E    68 03000040    push 40000003
00934153    4B             dec ebx
00934154    E8 74FDFFFF    call dump.933ECD
00934159    50             push eax                                   eax:"\\\\.\\pipe\\oinohaod"
0093415A    FF15 E0009400  call dword ptr ds:[<CreateNamedPipeA>]
00934160    8BF8           mov edi,eax                                eax:"\\\\.\\pipe\\oinohaod"
00934162    83FF FF        cmp edi,FFFFFFFF
00934165  v 75 21          jne dump.934188
```

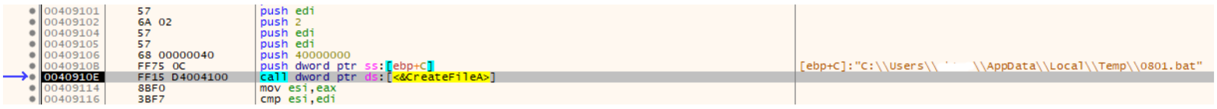
Figure 38 – Creating a Pipe



```
00934188    6A 00          push 0
0093418A    57             push edi
0093418B    FF15 DC009400  call dword ptr ds:[<ConnectNamedPipe>]
00934191    85C0           test eax,eax
00934193  v 75 16          jne dump.9341AB
```

Figure 39 – Pipe Connection Observed

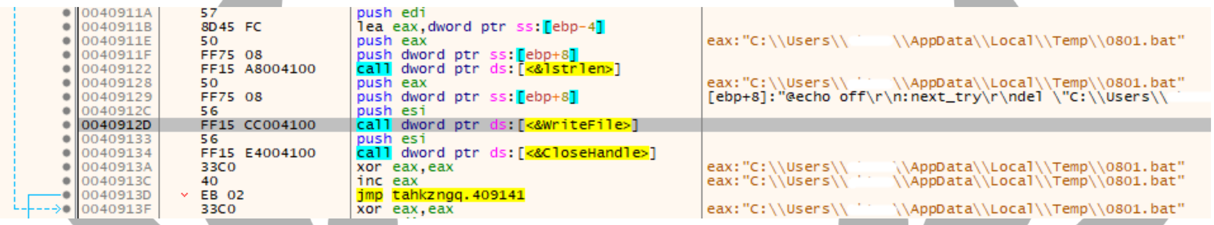
It has been observed that the malware forms pipeline.



```
00409101   57               push edi
00409102   6A 02            push 2
00409104   57               push edi
00409105   57               push edi
00409106   68 00000040      push 40000000
0040910B   FF75 0C          push dword ptr ss:[ebp+C]                [ebp+C]:"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\0801.bat"
0040910E   FF15 D4004100    call dword ptr ds:[<&CreateFileA>]
00409114   8BF0             mov esi,eax
00409116   3BF7             cmp esi,edi
```

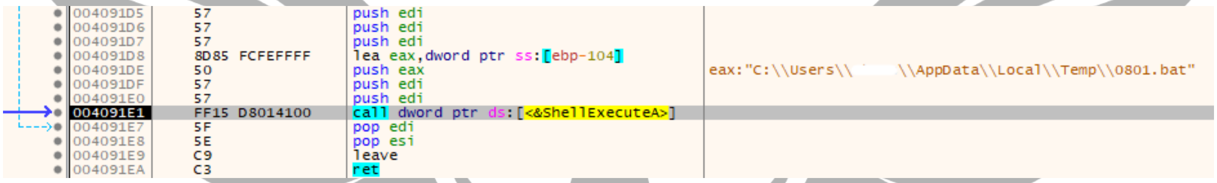
Figure 40 – Creating Bat File in Temp Directory

After all this, the malware creates a .bat file in the temp directory.



```
0040911A   57               push edi
0040911B   8D45 FC          lea eax,dword ptr ss:[ebp-4]
0040911E   50               push eax                                 eax:"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\0801.bat"
0040911F   FF75 08          push dword ptr ss:[ebp+8]
00409122   FF15 A8004100    call dword ptr ds:[<&lstrlen>]
00409128   50               push eax                                 eax:"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\0801.bat"
00409129   FF75 08          push dword ptr ss:[ebp+8]                [ebp+8]:"@echo off\r\n:next_try\r\ndel \"C:\\Users\\[redacted]
0040912C   56               push esi
0040912D   FF15 CC004100    call dword ptr ds:[<&WriteFile>]
00409133   56               push esi
00409134   FF15 E4004100    call dword ptr ds:[<&CloseHandle>]
0040913A   33C0             xor eax,eax                              eax:"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\0801.bat"
0040913C   40               inc eax                                  eax:"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\0801.bat"
0040913D   EB 02            jmp tahkzngq.409141
0040913F   33C0             xor eax,eax                              eax:"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\0801.bat"
```

Figure 41 – Updating Bat File Content



```
004091D5   57               push edi
004091D6   57               push edi
004091D7   57               push edi
004091D8   8D85 FCFEFFFF    lea eax,dword ptr ss:[ebp-104]
004091DE   50               push eax                                 eax:"C:\\Users\\[redacted]\\AppData\\Local\\Temp\\0801.bat"
004091DF   57               push edi
004091E0   57               push edi
004091E1   FF15 D8014100    call dword ptr ds:[<&ShellExecuteA>]
004091E7   5F               pop edi
004091E8   5E               pop esi
004091E9   C9               leave
004091EA   C3               ret
```

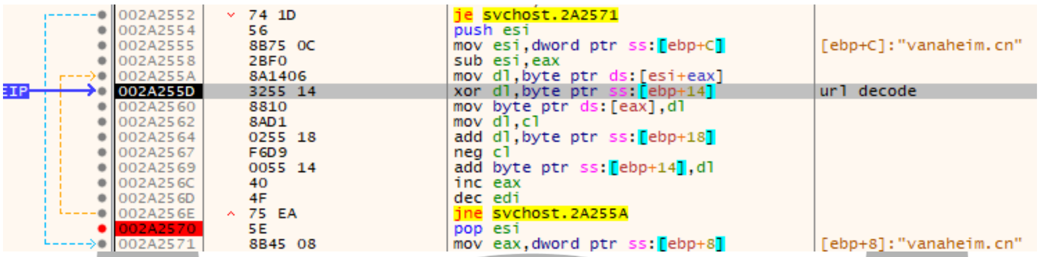
Figure 42 – Running Bat File

It was observed that it runs after writing to the relevant **.bat** file.

It appears that the bat file first deletes the main exe file specified with the /d parameter and deletes itself.

```
@echo off
:next_try
del "C:\Users\user\Desktop\
b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.exe
">nul
if exist "C:\Users\user\Desktop\
b984128113ff555edf24f086dcec400c697413f9095c8510da1058a98a2cc4ad.exe
" (
ping 127.0.0.1 >nul
goto next_try
)
del %0
```

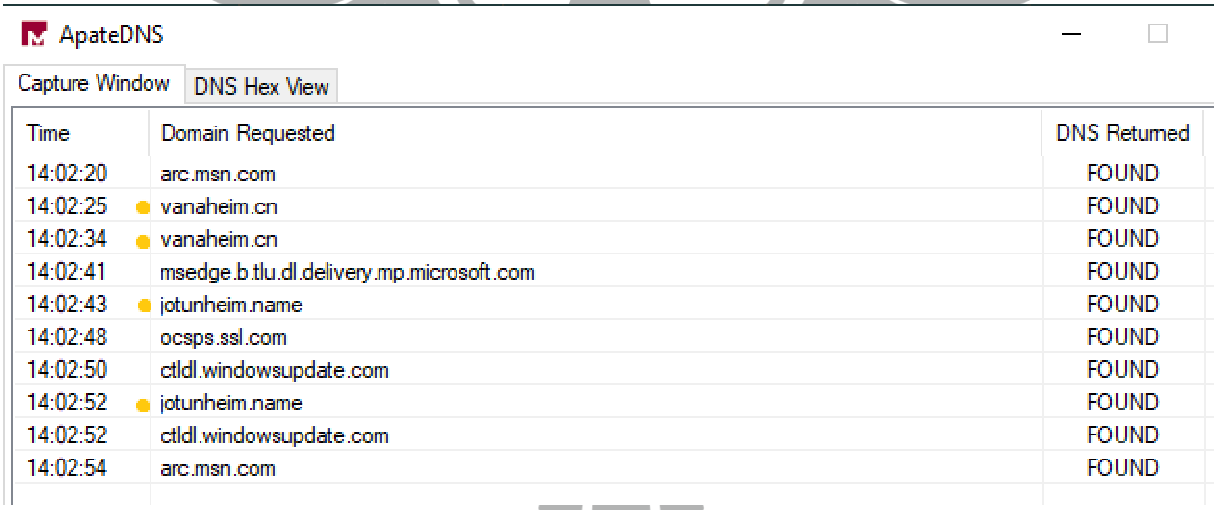
Network Analysis



```
EIP  002A2552  ⌄ 74 1D     je svchost.2A2571
     002A2554    56        push esi
     002A2555    8B75 0C   mov esi,dword ptr ss:[ebp+C]      [ebp+C]:"vanaheim.cn"
     002A2558    2BF0      sub esi,eax
     002A255A    8A1406    mov dl,byte ptr ds:[esi+eax]
     002A255D    3255 14   xor dl,byte ptr ss:[ebp+14]       url decode
     002A2560    8810      mov byte ptr ds:[eax],dl
     002A2562    8AD1      mov dl,cl
     002A2564    0255 18   add dl,byte ptr ss:[ebp+18]
     002A2567    F6D9      neg cl
     002A2569    0055 14   add byte ptr ss:[ebp+14],dl
     002A256C    40        inc eax
     002A256D    4F        dec edi
     002A256E  ^ 75 EA     jne svchost.2A255A
     002A2570    5E        pop esi
     002A2571    8B45 08   mov eax,dword ptr ss:[ebp+8]      [ebp+8]:"vanaheim.cn"
```

Figure 43 - Runtime Decryption

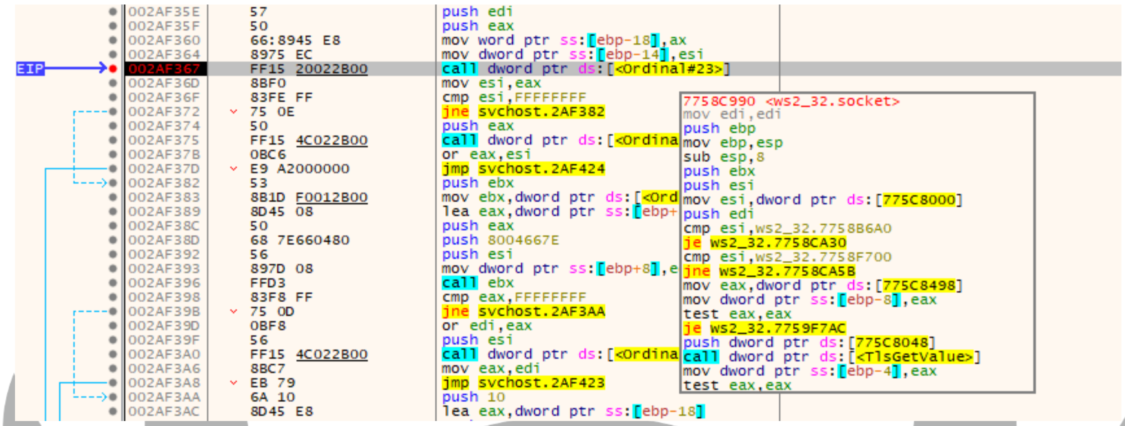
In the related **svchost** process, the malware decrypts the C2 Servers to be contacted with the **Runtime Decrypt** process.



Time		Domain Requested	DNS Returned
14:02:20		arc.msn.com	FOUND
14:02:25	●	vanaheim.cn	FOUND
14:02:34	●	vanaheim.cn	FOUND
14:02:41		msedge.b.tlu.dl.delivery.mp.microsoft.com	FOUND
14:02:43	●	jotunheim.name	FOUND
14:02:48		ocsps.ssl.com	FOUND
14:02:50		ctldl.windowsupdate.com	FOUND
14:02:52	●	jotunheim.name	FOUND
14:02:52		ctldl.windowsupdate.com	FOUND
14:02:54		arc.msn.com	FOUND

Figure 44 – Monitoring C2 Servers with ApateDNS

It was observed that attempts were made to connect to **vanaheim[.]cn** and **jotunheim[.]name** domain addresses resolved in the dynamics.

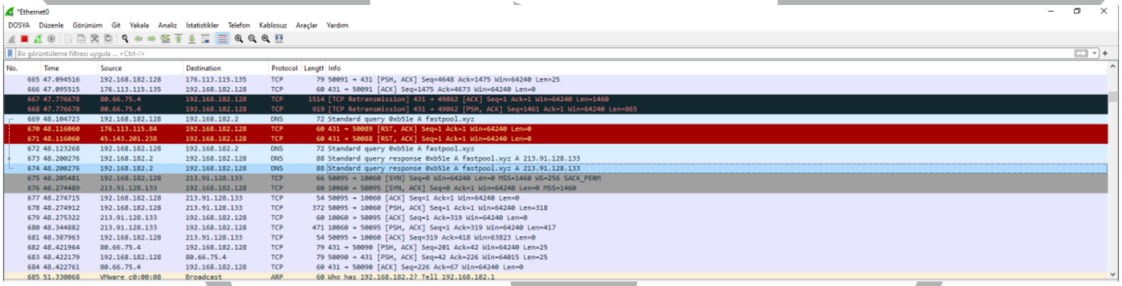


```
002AF35E   57              push edi
002AF35F   50              push eax
002AF360   66:8945 E8      mov word ptr ss:[ebp-18],ax
002AF364   8975 EC         mov dword ptr ss:[ebp-14],esi
002AF367   FF15 20022B00   call dword ptr ds:[<Ordinal#23>]
002AF36D   8BF0            mov esi,eax
002AF36F   83FE FF         cmp esi,FFFFFFFF
002AF372   75 0E           jne svchost.2AF382
002AF374   50              push eax
002AF375   FF15 4C022B00   call dword ptr ds:[<Ordina
002AF37B   0BC6            or eax,esi
002AF37D   E9 A2000000     jmp svchost.2AF424
002AF382   53              push ebx
002AF383   8B1D F0012B00   mov ebx,dword ptr ds:[<Ord
002AF389   8D45 08         lea eax,dword ptr ss:[ebp+
002AF38C   50              push eax
002AF38D   68 7E660480     push 8004667E
002AF392   56              push esi
002AF393   897D 08         mov dword ptr ss:[ebp+8],e
002AF396   FFD3            call ebx
002AF398   83F8 FF         cmp eax,FFFFFFFF
002AF39B   75 0D           jne svchost.2AF3AA
002AF39D   0BF8            or edi,eax
002AF39F   56              push esi
002AF3A0   FF15 4C022B00   call dword ptr ds:[<Ordina
002AF3A6   8BC7            mov eax,edi
002AF3A8   EB 79           jmp svchost.2AF423
002AF3AA   6A 10           push 10
002AF3AC   8D45 E8         lea eax,dword ptr ss:[ebp-18]
```

```
7758C990 <ws2_32.socket>
mov edi,edi
push ebp
mov ebp,esp
sub esp,8
push ebx
push esi
mov esi,dword ptr ds:[775C8000]
push edi
cmp esi,ws2_32.7758B6A0
je ws2_32.7758CA30
cmp esi,ws2_32.7758F700
jne ws2_32.7758CA5B
mov eax,dword ptr ds:[775C8498]
mov dword ptr ss:[ebp-8],eax
test eax,eax
je ws2_32.7759F7AC
push dword ptr ds:[775C8048]
call dword ptr ds:[<TlsGetValue>]
mov dword ptr ss:[ebp-4],eax
test eax,eax
```

Figure 45 – Establishing Connection with Socket

Relevant addresses are contacted using sockets.

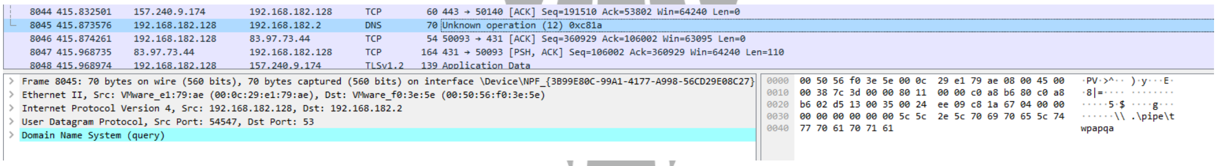


*Ethernet0
DOSYA Düzenle Görünüm Git Yakala Analiz İstatistikler Telefon Kablosuz Araçlar Yardım
Bir görüntüleme filtresi uygula ... <Ctrl-/>

```
No.  Time       Source           Destination      Protocol Length Info
665 47.094516  192.168.182.128  176.113.115.135  TCP      79 50091 → 431 [PSH, ACK] Seq=4648 Ack=1475 Win=64240 Len=25
666 47.095515  176.113.115.135  192.168.182.128  TCP      60 431 → 50091 [ACK] Seq=1475 Ack=4673 Win=64240 Len=0
667 47.776678  80.66.75.4       192.168.182.128  TCP    1514 [TCP Retransmission] 431 → 49862 [ACK] Seq=1 Ack=1 Win=64240 Len=1460
668 47.776678  80.66.75.4       192.168.182.128  TCP     919 [TCP Retransmission] 431 → 49862 [PSH, ACK] Seq=1461 Ack=1 Win=64240 Len=865
669 48.104723  192.168.182.128  192.168.182.2    DNS      72 Standard query 0xb51e A fastpool.xyz
670 48.116060  176.113.115.84   192.168.182.128  TCP      60 431 → 50089 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
671 48.116060  45.143.201.238   192.168.182.128  TCP      60 431 → 50088 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0
672 48.123268  192.168.182.128  192.168.182.2    DNS      72 Standard query 0xb51e A fastpool.xyz
673 48.200276  192.168.182.2    192.168.182.128  DNS      88 Standard query response 0xb51e A fastpool.xyz A 213.91.128.133
674 48.200276  192.168.182.2    192.168.182.128  DNS      88 Standard query response 0xb51e A fastpool.xyz A 213.91.128.133
675 48.205481  192.168.182.128  213.91.128.133   TCP      66 50095 → 10060 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
676 48.274489  213.91.128.133   192.168.182.128  TCP      60 10060 → 50095 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
677 48.274715  192.168.182.128  213.91.128.133   TCP      54 50095 → 10060 [ACK] Seq=1 Ack=1 Win=64240 Len=0
678 48.274912  192.168.182.128  213.91.128.133   TCP     372 50095 → 10060 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=318
679 48.275322  213.91.128.133   192.168.182.128  TCP      60 10060 → 50095 [ACK] Seq=1 Ack=319 Win=64240 Len=0
680 48.344882  213.91.128.133   192.168.182.128  TCP     471 10060 → 50095 [PSH, ACK] Seq=1 Ack=319 Win=64240 Len=417
681 48.387963  192.168.182.128  213.91.128.133   TCP      54 50095 → 10060 [ACK] Seq=319 Ack=418 Win=63823 Len=0
682 48.421964  80.66.75.4       192.168.182.128  TCP      79 431 → 50090 [PSH, ACK] Seq=201 Ack=42 Win=64240 Len=25
683 48.422179  192.168.182.128  80.66.75.4       TCP      79 50090 → 431 [PSH, ACK] Seq=42 Ack=226 Win=64015 Len=25
684 48.422761  80.66.75.4       192.168.182.128  TCP      60 431 → 50090 [ACK] Seq=226 Ack=67 Win=64240 Len=0
685 51.330068  VMware_c0:00:08  Broadcast        ARP      60 Who has 192.168.182.2? Tell 192.168.182.1
```

Figure 46 – Monitoring TCP Connection with Wireshark

Other domain services are contacted according to the commands received from the relevant C2 servers.

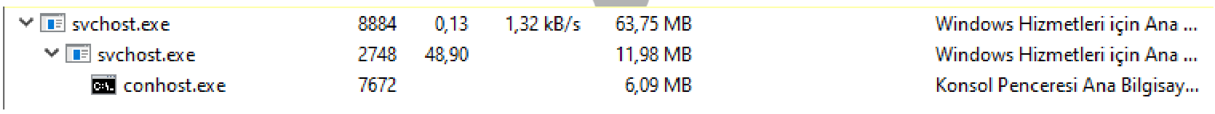


```
8044 415.832501  157.240.9.174    192.168.182.128  TCP      60 443 → 50140 [ACK] Seq=191510 Ack=53802 Win=64240 Len=0
8045 415.873576  192.168.182.128  192.168.182.2    DNS      70 Unknown operation (12) 0xc81a
8046 415.874261  192.168.182.128  83.97.73.44      TCP      54 50093 → 431 [ACK] Seq=360929 Ack=106002 Win=63095 Len=0
8047 415.968735  83.97.73.44      192.168.182.128  TCP     164 431 → 50093 [PSH, ACK] Seq=106002 Ack=360929 Win=64240 Len=110
8048 415.968974  192.168.182.128  157.240.9.174    TLSv1.2 139 Application Data
```

```
> Frame 8045: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{3B99E80C-99A1-4177-A998-56CD29E08C27}
> Ethernet II, Src: VMware_e1:79:ae (00:0c:29:e1:79:ae), Dst: VMware_f0:3e:5e (00:50:56:f0:3e:5e)
> Internet Protocol Version 4, Src: 192.168.182.128, Dst: 192.168.182.2
> User Datagram Protocol, Src Port: 54547, Dst Port: 53
> Domain Name System (query)
```

```
0000  00 50 56 f0 3e 5e 00 0c  29 e1 79 ae 08 00 45 00   ·PV·>^·· )·y···E·
0010  00 38 7c 3d 00 00 80 11  00 00 c0 a8 b6 80 c0 a8   ·8|=···· ········
0020  b6 02 d5 13 00 35 00 24  ee 09 c8 1a 67 04 00 00   ·····5·$ ····g···
0030  00 00 00 00 00 00 5c 5c  2e 5c 70 69 70 65 5c 74   ······\\ .\pipe\t
0040  77 70 61 70 71 61                                  wpapqa
```

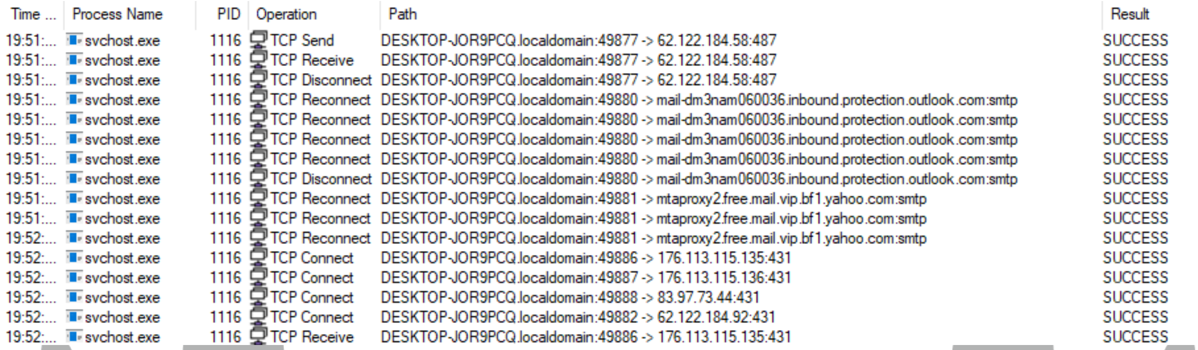
Figure 47 – Pipe Monitoring with Wireshark



svchost.exe	8884	0,13	1,32 kB/s	63,75 MB	Windows Hizmetleri için Ana ...
svchost.exe	2748	48,90		11,98 MB	Windows Hizmetleri için Ana ...
conhost.exe	7672			6,09 MB	Konsol Penceresi Ana Bilgisay...

Figure 48 – Viewing Child Process via Procmon

It was observed that the malware can start a child process according to the commands returned from the C2 server and communicate with the parent process using **pipes**. With this method, the malware aims to bypass the **EDR**.



Time ...	Process Name	PID	Operation	Path	Result
19:51:...	svchost.exe	1116	TCP Send	DESKTOP-JOR9PCQ.localdomain:49877 -> 62.122.184.58:487	SUCCESS
19:51:...	svchost.exe	1116	TCP Receive	DESKTOP-JOR9PCQ.localdomain:49877 -> 62.122.184.58:487	SUCCESS
19:51:...	svchost.exe	1116	TCP Disconnect	DESKTOP-JOR9PCQ.localdomain:49877 -> 62.122.184.58:487	SUCCESS
19:51:...	svchost.exe	1116	TCP Reconnect	DESKTOP-JOR9PCQ.localdomain:49880 -> mail-dm3nam060036.inbound.protection.outlook.com:smtp	SUCCESS
19:51:...	svchost.exe	1116	TCP Reconnect	DESKTOP-JOR9PCQ.localdomain:49880 -> mail-dm3nam060036.inbound.protection.outlook.com:smtp	SUCCESS
19:51:...	svchost.exe	1116	TCP Reconnect	DESKTOP-JOR9PCQ.localdomain:49880 -> mail-dm3nam060036.inbound.protection.outlook.com:smtp	SUCCESS
19:51:...	svchost.exe	1116	TCP Reconnect	DESKTOP-JOR9PCQ.localdomain:49880 -> mail-dm3nam060036.inbound.protection.outlook.com:smtp	SUCCESS
19:51:...	svchost.exe	1116	TCP Disconnect	DESKTOP-JOR9PCQ.localdomain:49880 -> mail-dm3nam060036.inbound.protection.outlook.com:smtp	SUCCESS
19:51:...	svchost.exe	1116	TCP Reconnect	DESKTOP-JOR9PCQ.localdomain:49881 -> mtaproxy2.free.mail.vip.bf1.yahoo.com:smtp	SUCCESS
19:51:...	svchost.exe	1116	TCP Reconnect	DESKTOP-JOR9PCQ.localdomain:49881 -> mtaproxy2.free.mail.vip.bf1.yahoo.com:smtp	SUCCESS
19:52:...	svchost.exe	1116	TCP Reconnect	DESKTOP-JOR9PCQ.localdomain:49881 -> mtaproxy2.free.mail.vip.bf1.yahoo.com:smtp	SUCCESS
19:52:...	svchost.exe	1116	TCP Connect	DESKTOP-JOR9PCQ.localdomain:49886 -> 176.113.115.135:431	SUCCESS
19:52:...	svchost.exe	1116	TCP Connect	DESKTOP-JOR9PCQ.localdomain:49887 -> 176.113.115.136:431	SUCCESS
19:52:...	svchost.exe	1116	TCP Connect	DESKTOP-JOR9PCQ.localdomain:49888 -> 83.97.73.44:431	SUCCESS
19:52:...	svchost.exe	1116	TCP Connect	DESKTOP-JOR9PCQ.localdomain:49882 -> 62.122.184.92:431	SUCCESS
19:52:...	svchost.exe	1116	TCP Receive	DESKTOP-JOR9PCQ.localdomain:49886 -> 176.113.115.135:431	SUCCESS

Figure 49 – Contacting SMTP Services

It has been determined that there is a continuous attempt to establish a connection with SMTP services.

STAGE1 YARA Rule

```
import "hash"

rule Tofsee

{

    meta:

        author="Alper Aktaş"

        description="tofsee"

        report_date="3.3.2024"

    strings:

        $str1 = ".?AVbad_alloc@std@@" ascii

        $str2= ".?AV?$basic_stringbuf@DU?$char_traits@D@std@@V?$” ascii

        $str3 = "`non-type-template-parameter" ascii

        $str4 = "cli::pin_ptr<" ascii

        $str5 = “GlobalAlloc” ascii

        $str5 = “VirtualProtect” ascii

        $technique = {E8 04 00 00 00 00 00 00 00}

    condition:

        hash.md5(0, filesize) == "9f9e5f55dc8cb3809e24b14fb8f9c27d" or all of them

}
```

STAGE2 YARA Rule

```
import "hash"

rule Tofsee

{

    meta:

        author="Alper Aktaş"

        description="tofsee"

        report_date="3.3.2024"


    strings:

        // encrypted vanaheim[.]cn

        $ip = {92 CC 1A 5C 6C A8 FD 30 0A 8E DA}

        // encrypted jotunheim[.]name

        $ip2 = {8E C2 00 48 6A A5 F1 34 49 C3 DA}


        $str1 = "%RND_NUM" ascii

        $str2 = "\\\\.\\pipe\\" ascii

        $str3 = "ret=%p" ascii


    condition:

        hash.md5(0, filesize) == "95fc3460859b033780774fc0d5ec768d" or all of them


}
```

MITRE ATTACK TABLE

Execution	Persistence	Defense Evasion	Credential Access	Discovery	Collection	Command and Control	Exfiltration
T1569 System Services	T1547 Boot or Login Autostart Discovery	T1055 Process Injection		T1012 Query Registry		T1105 Ingress Tool Transfer	
		T1112 Modify Registry		T1083 File and Directory Discovery		T1095 Non-Application Layer Protocol	
		T1562 Impair Defenses		T1082 System Information Discovery		T1571 Non-Standard Port	
		T1027 Obfuscated Files or Information					

Solution Suggestions

1. Attachments or links presented in emails from unknown, suspicious addresses should not be opened.
2. Keep software cracking tools and unreliable software download sources away from your computer.
3. To avoid exposure to malicious websites and downloads, use trusted websites and make downloads from trusted sources.
4. The applications used must be licensed and up-to-date.
5. By regularly updating your security software and operation system, you can strengthen its defenses against known attacks.

PREPARED BY

Alper AKTAŞ Linkedin