Vidar Stealer

TECHNICAL ANALYSIS REPORT

ZAYOTEM

ZARARLI YAZILIM ÖNLEME VE TERSİNE MÜHENDİSLİK

Index

Overview

The malware from the Vidar family appears as an EXE file. This malicious software can access personal information, cryptocurrency wallet data, and cookie information on infected computers. It was first discovered in late 2018 and has been regularly updated and improved since then. The malware can be spread through malicious links or attachments sent via email, downloaded fake files or applications, malvertising, and social engineering attacks.

Analysis of sample.exe

File Name	sample.exe
MD5	701477F861BDE9756D5FC3ACE9D2F019
SHA256	2E0F06DF176B574CD8F629F8E0D32FDEDC72DD20
File Type	PE32/EXE

The MD5, SHA-1, and SHA-256 hashes of the malware are listed in the table. The original name is 48aa1381548b2590a3ae1d740852fdefdf51c46666ee2d86e50aeae66afbda60.exe, but it has been renamed to sample.exe for ease of analysis.

Dynamic Analysis

```
mov     dword ptr [ebp+var_820+4], edx
push    42h ; 'B'
push    77Eh
lea     eax, [ebp+var_7B0]
push    eax
push    offset a3h8w2npbk4nrdu ; "3h8W2nPBk4nRDUrB6Y0h0HLpyqaFdsG77R2qmHs"...
call    sub_401080
```

Figure 2. The string used by the malware

When the malware is examined, the string named **3h8W2nOBk4nRDUrB6Y0HLpyqaFdsG77R2qmHs** draws attention. When the function it calls is examined, it is seen that it is a harmless Shellcode that performs jump operations in memory. It is understood that this string was inserted to confuse.

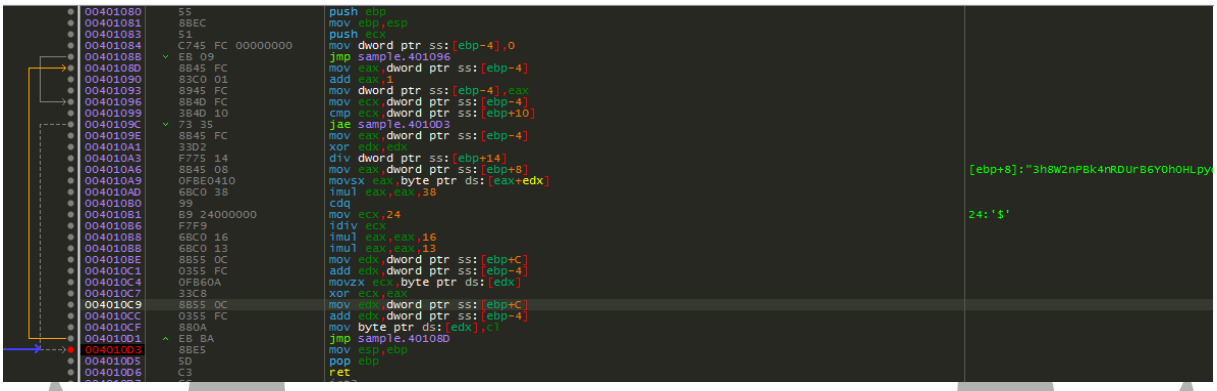


Figure 3. Jump operations in Memory

```
push    42h ; 'B'
push    5AE00h
push    offset unk_418008
push    offset aTsuyxh4r2bmpl6 ; "tsUYxh4R2BMPl6IVK7msKOJi8MeYnj3B4ogS6KP"...
call    sub_401000
```

Figure 4. The string used by the malware

When the malware is examined, another string named **tsUYxh4R2BMPl6IVK7msKOJi8MeYnj3B4ogS6KP** stands out. This string is assigned to the function named **sub_401000**.

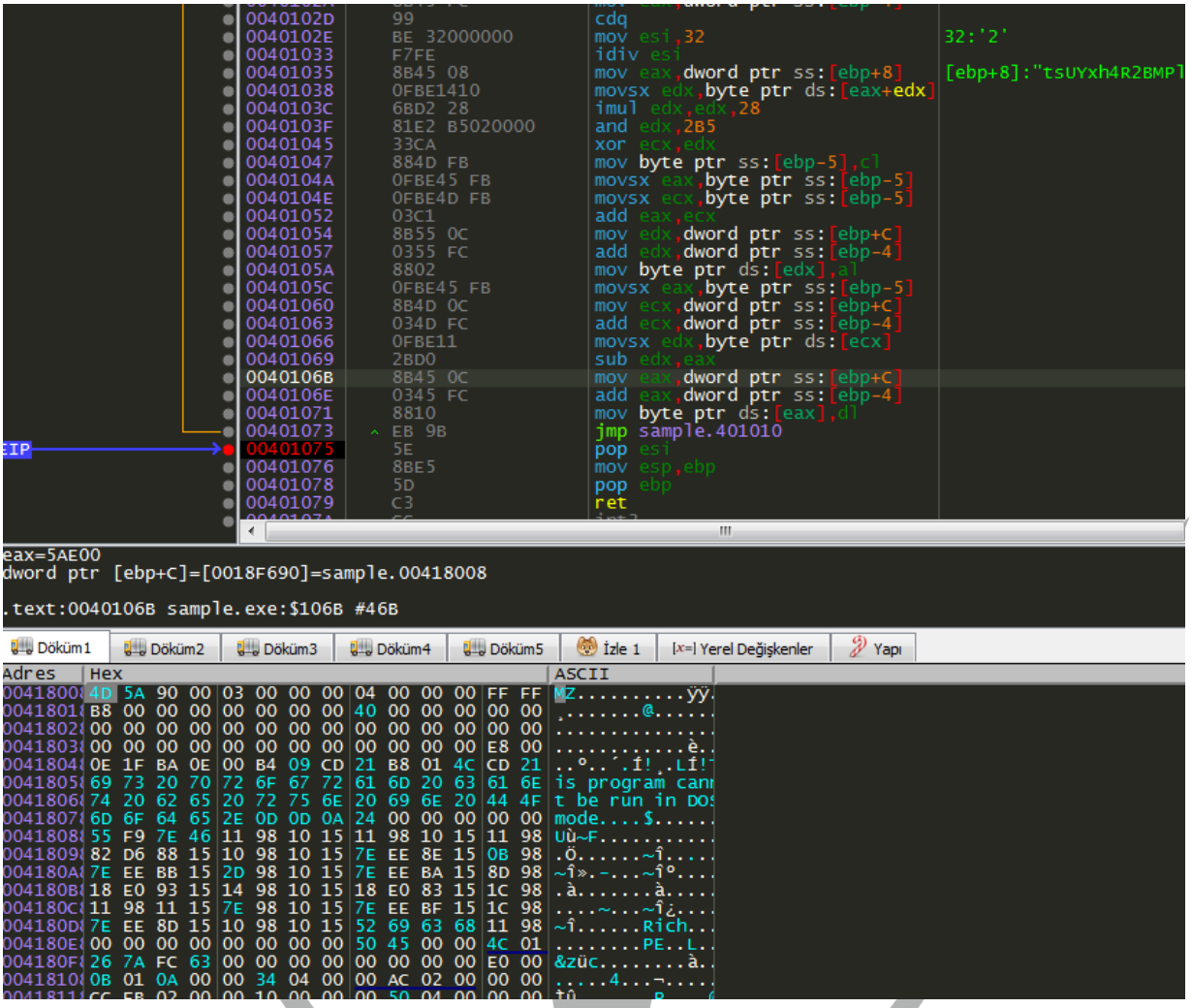


Figure 5. Injected PE file

Upon entering and examining the function **sub_401000**, it is understood that a PE file is injected into sample.exe with the pop esi instruction. The dump was followed in the memory map and then saved as **SAMPLE_00418000.EXE**. The .exe file that was recorded was analyzed statically and dynamically.

Analysis of sample_00418000.exe

File Name	sample_00418000.exe
MD5	35EBCE61CD83460135893269B991E740
SHA256	DA39750642B84880BD1E882E3EF53C7E72C42366
File Type	PE32/EXE

The MD5 and SHA-256 information of the dropped .exe file is given in the table. It is named sample_00418000.exe for ease of analysis.

Dynamic Analysis

```
0040DA73    FF15 BC404100    call dword ptr ds:[<&GetSystemTimeAsFileTime>]
0040DA79    8B75 FC          mov esi,dword ptr ss:[ebp-4]
0040DA7C    3375 F8          xor esi,dword ptr ss:[ebp-8]
0040DA7F    FF15 18404100    call dword ptr ds:[<&GetCurrentProcessId>]
0040DA85    33F0             xor esi,eax
0040DA87    FF15 68404100    call dword ptr ds:[<&GetCurrentThreadId>]
0040DA8D    33F0             xor esi,eax
0040DA8F    FF15 B8404100    call dword ptr ds:[<&GetTickCount>]
0040DA95    33F0             xor esi,eax
0040DA97    8D45 F0          lea eax,dword ptr ss:[ebp-10]
0040DA9A    50               push eax
0040DA9B    FF15 B4404100    call dword ptr ds:[<&QueryPerformanceCounter>]
```

Figure 6. APIs used by malware

In the first phase, the malware collects general information about the system. The APIs it uses are given in the table below.

GetSystemTimeAsFileTime	Gets the current system date and time. The information is in Coordinated Universal Time (UTC) format.
GetCurrentProccessId	Gets the process identifier of the calling process

GetCurrentThreadId	Gets the thread identifier of the calling thread
GetTickCount	Gets the number of milliseconds (up to 49.7 days) since system initialization.
QueryPerformanceCounter	Gets the current value of the performance counter with a high resolution (<1us) timestamp that can be used for time interval measurements.

Figure 7. Impossible Disassembly Technique

When the dropped .exe file is examined in the Disassembler program, it is understood that the Impossible Disassembly technique is used. This technique is an Anti-Disassembly technique that aims to make reverse engineering difficult. As shown in the figure, data bytes are added to the conditional skip directive. These data bytes are designed to prevent the disassembly algorithm from disassembling the actual instruction after the jump instruction. The B8 opcode in the figure is not used at all because it is skipped. Since the disassembler cannot make sense of these parts, it misinterprets them. This technique is often used in malware and other unsafe software.

```
.text:00414E60 74 03                                jz      short near ptr loc_414E64+1 ; Jump if Zero (ZF=1)
.text:00414E62 75 01                                jnz     short near ptr loc_414E64+1 ; Jump if Not Zero (ZF=0)
.text:00414E64
.text:00414E64                          loc_414E64:                                     ; CODE XREF: .text:WinMain(x,x,x,x)↑j
.text:00414E64                                                                          ; .text:00414E62↑j
.text:00414E64 B8 E8 26 C2 FE                       mov     eax, 0FEC226E8h
.text:00414E69 FF 74 03 75                          push    dword ptr [ebx+eax+75h]
.text:00414E6D 01 B8 E8 1C C2 FE                    add     [eax-13DE318h], edi ; Add
.text:00414E73 FF 74 03 75                          push    dword ptr [ebx+eax+75h]
.text:00414E77 01 B8 E8 12 C2 FE                    add     [eax-13DED18h], edi ; Add
```

Figure 8. Impossible Disassembly technique before decoding

We analyze the impossible disassembly technique by replacing the B8 opcode in the figure with 90, that is, by filling it with NOP.

```
.text:00414E60
.text:00414E60                          ; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
.text:00414E60                          _WinMain@16:                            ; CODE XREF: ___tmainCRTStartup+115↓p
.text:00414E60 74 03                                jz      short loc_414E65 ; Jump if Zero (ZF=1)
.text:00414E62 75 01                                jnz     short loc_414E65 ; Jump if Not Zero (ZF=0)
.text:00414E64 90                                   nop                      ; No Operation
.text:00414E65
.text:00414E65                          loc_414E65:                             ; CODE XREF: .text:WinMain(x,x,x,x)↑j
.text:00414E65                                                                  ; .text:00414E62↑j
.text:00414E65 E8 26 C2 FE FF                       call    sub_401090      ; Call Procedure
.text:00414E6A 74 03                                jz      short near ptr unk_414E6F ; Jump if Zero (ZF=1)
.text:00414E6C 75 01                                jnz     short near ptr unk_414E6F ; Jump if Not Zero (ZF=0)
.text:00414E6E 90                                   nop                      ; No Operation
```

Figure 9. After analyzing the Impossible Disassembly technique

In this way, the malware can be analyzed without getting stuck in the Disassembler.

To avoid malware that uses Impossible Disassembly tactics:

- Download software from trusted sources.
- Keep the software up to date.
- Use an antivirus program.
- Analyze your computer's software with special tools designed for reverse engineering.

```
00E423D5    50              push eax
00E423D6    E8 30B80100     call dropped.E5DC0B
00E423DB    83C4 04         add esp,4
00E423DE    8975 20         mov dword ptr ss:[ebp+20],esi
00E423E1    895D 1C         mov dword ptr ss:[ebp+1C],ebx
00E423E4    885D 0C         mov byte ptr ss:[ebp+C],bl
00E423E7    397D 3C         cmp dword ptr ss:[ebp+3C],edi
00E423EA    72 0C           jb dropped.E423F8
00E423EC    8B4D 28         mov ecx,dword ptr ss:[ebp+28]   [ebp+28]:"https://t.me/dionysus_tg"
00E423EF    51              push ecx
00E423F0    E8 16B80100     call dropped.E5DC0B             tg
00E423F5    83C4 04         add esp,4
00E423F8    8B4D F4         mov ecx,dword ptr ss:[ebp-C]
00E423FB    64:890D 00000000 mov dword ptr fs:[0],ecx
00E42402    59              pop ecx
00E42403    5F              pop edi
00E42404    5E              pop esi                         esi:"ERROR"
```

Figure 10. Telegram bot URL address

The dropped .exe file contains the URL address of the telegram bot used for the C&C server.

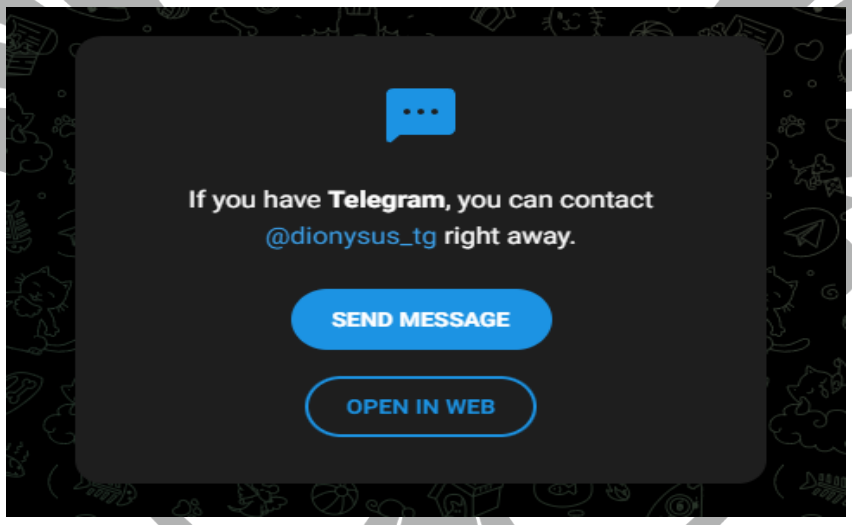


Figure 11. Telegram bot

```
00E3E7C0    55              push ebp
00E3E7C1    8BEC            mov ebp,esp
00E3E7C3    51              push ecx
00E3E7C4    33C0            xor eax,eax
00E3E7C6    6A 35           push 35
00E3E7C8    C746 14 0F000000 mov dword ptr ds:[esi+14],F
00E3E7CF    8946 10         mov dword ptr ds:[esi+10],eax
00E3E7D2    68 60FAE700     push dropped.E7FA60             E7FA60:"https://steamcommunity.com/profiles/76561199482248283"
00E3E7D7    8BCE            mov ecx,esi                     esi:"ERROR"
00E3E7D9    8945 FC         mov dword ptr ss:[ebp-4],eax
00E3E7DC    8806            mov byte ptr ds:[esi],al        esi:"ERROR"
00E3E7DE    E8 0DA9FFFF     call dropped.E390F0
00E3E7E3    8BC6            mov eax,esi                     esi:"ERROR"
00E3E7E5    8BE5            mov esp,ebp
00E3E7E7    5D              pop ebp
00E3E7E8    C3              ret
```

Figure 1. Steam URL address

The dropped .exe file contains the steamcommunity URL address.

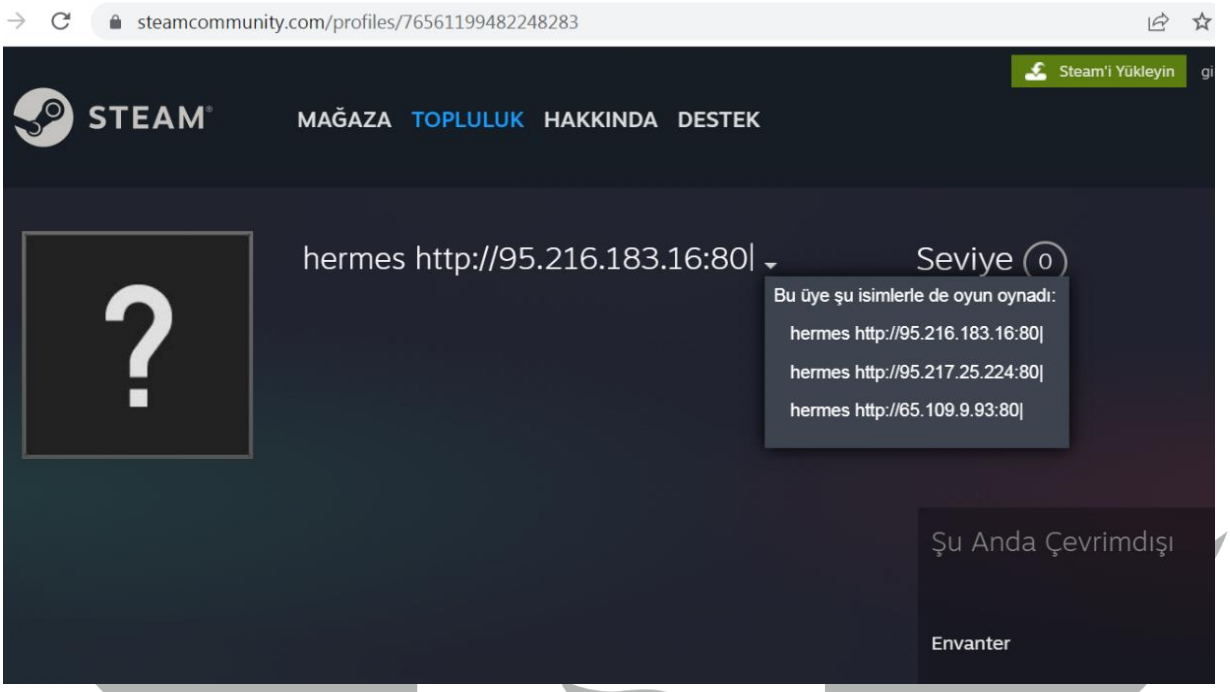


Figure 2. Steam profile page

Using the steamcommunity URL address, the malware sends an HTTP GET request to hermes http[:]//95[.]216[.]183[.]183[.]16[:]80, hermes http[:]//95[.]217[.]25[.]224[:]80, hermes http[:]//65[.]109[.]9[.]93[:]80. With this request,

the malware tries to pull a file named hera.zip from the server.

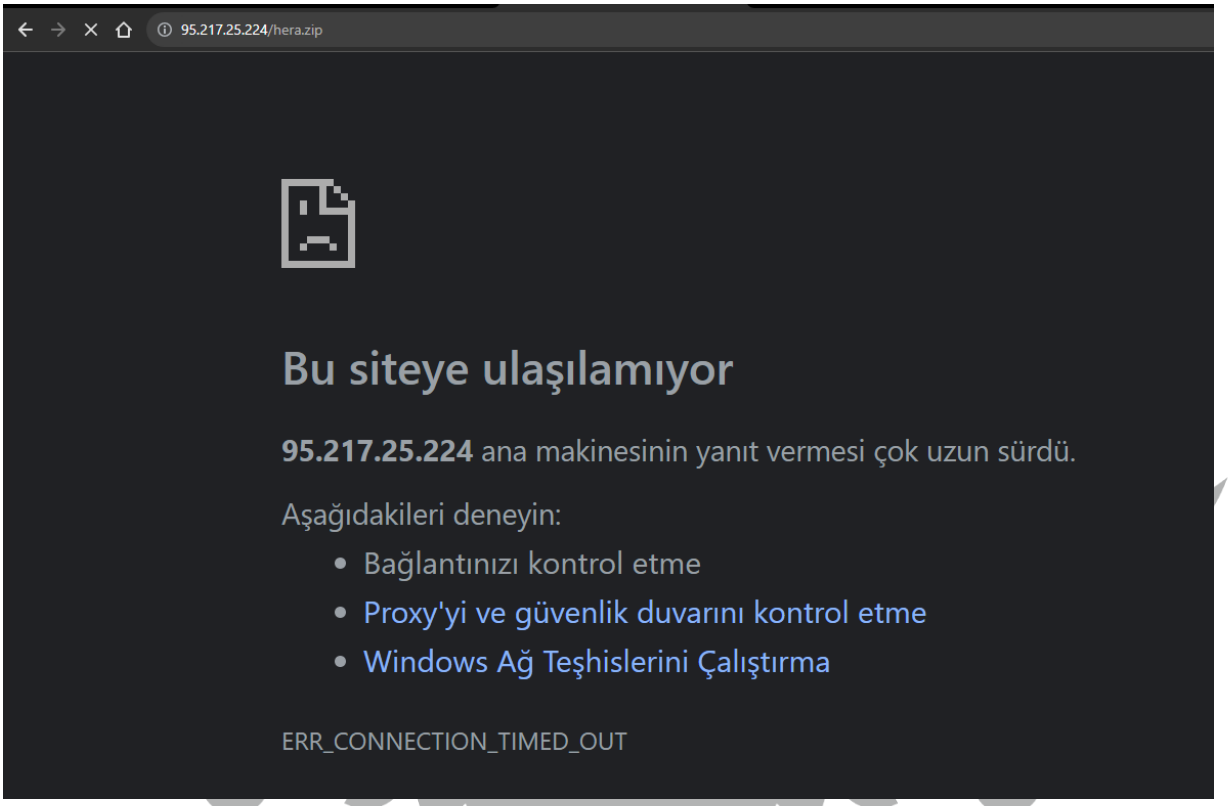


Figure 14. web server to pull hera.zip

It is seen that the server from which the file named hera.zip will be extracted is down.

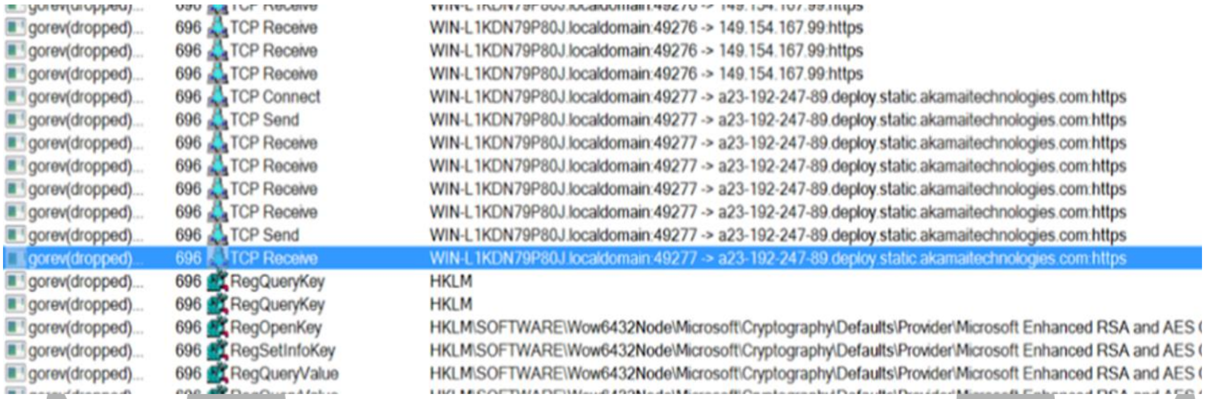


Figure 15. Request discarded from local machine

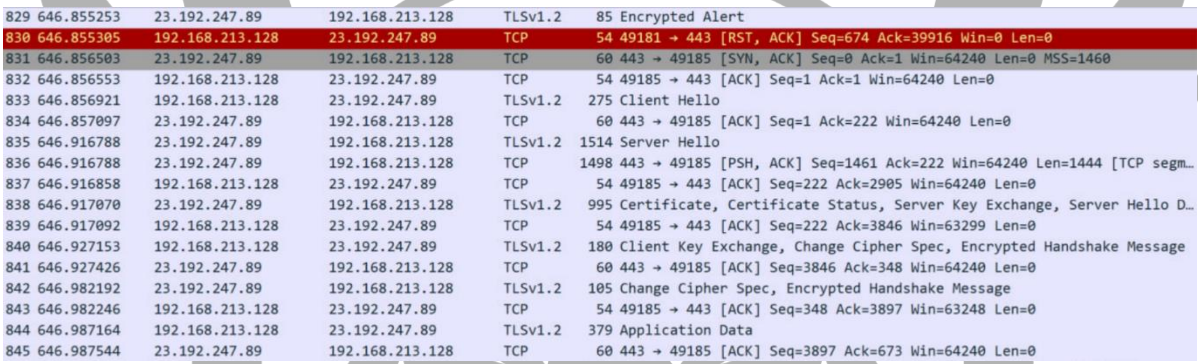


Figure 3. Wireshark screenshot

At the same time, it is seen that the malware also sends requests to the IP address **23[.]192[.]192[.]247[.]89** from the local machine.

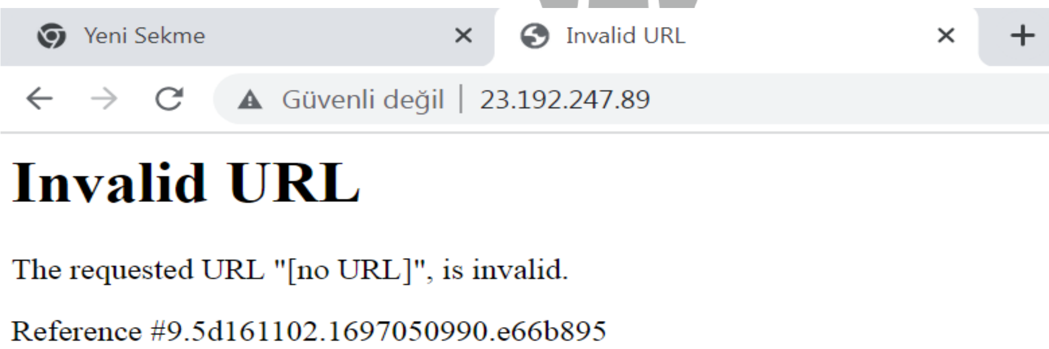
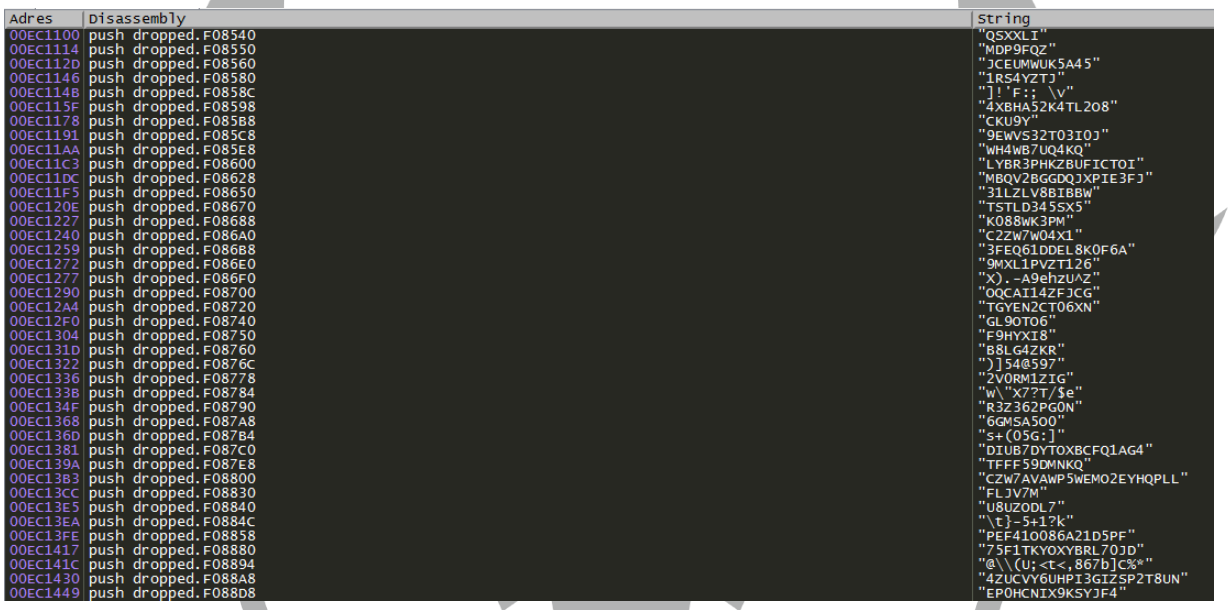


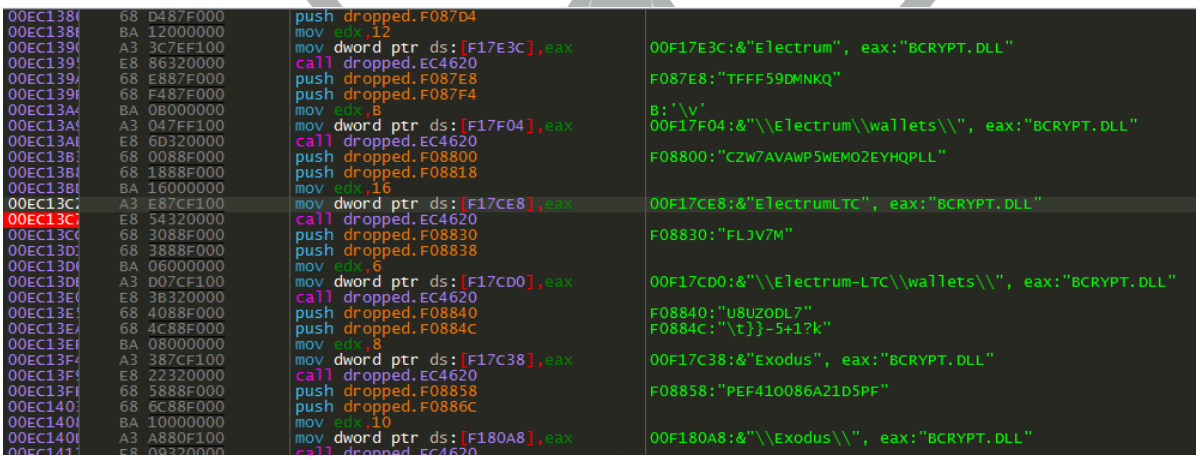
Figure 4. Invalid IP address

Looking at the IP address **23[.]192[.]192[.]247[.]89,** it is seen that it is invalid.



Adres	Disassembly	String
00EC1100	push dropped.F08540	"QSXXLI"
00EC1114	push dropped.F08550	"MDP9FQZ"
00EC112D	push dropped.F08560	"JCEUMWUK5A45"
00EC1146	push dropped.F08580	"1RS4YZTJ"
00EC114B	push dropped.F0858C	"]!'F:; \v"
00EC115F	push dropped.F08598	"4XBHA52K4TL2O8"
00EC1178	push dropped.F085B8	"CKU9Y"
00EC1191	push dropped.F085C8	"9EWVS32T03IOJ"
00EC11AA	push dropped.F085E8	"WH4WB7UQ4KQ"
00EC11C3	push dropped.F08600	"LYBR3PHKZBUFICTOI"
00EC11DC	push dropped.F08628	"MBQV2BGGDQJXPIE3FJ"
00EC11F5	push dropped.F08650	"31LZLV8BIBBW"
00EC120E	push dropped.F08670	"TSTLD3455X5"
00EC1227	push dropped.F08688	"KO88WK3PM"
00EC1240	push dropped.F086A0	"C2ZW7W04X1"
00EC1259	push dropped.F086B8	"3FEQ61DDEL8K0F6A"
00EC1272	push dropped.F086E0	"9MXL1PVZT126"
00EC1277	push dropped.F086F0	"X).-A9ehzUAZ"
00EC1290	push dropped.F08700	"OQCAI14ZFJCG"
00EC12A4	push dropped.F08720	"TGYEN2CTO6XN"
00EC12F0	push dropped.F08740	"GL9OTO6"
00EC1304	push dropped.F08750	"F9HYXI8"
00EC131D	push dropped.F08760	"B8LG4ZKR"
00EC1322	push dropped.F0876C	")]54@597"
00EC1336	push dropped.F08778	"2V0RM1ZIG"
00EC133B	push dropped.F08784	"W\"X7?T/$e"
00EC134F	push dropped.F08790	"R3Z362PG0N"
00EC1368	push dropped.F087A8	"6GMSA5OO"
00EC136D	push dropped.F087B4	"s+(O5G:]"
00EC1381	push dropped.F087C0	"DIUB7DYTOXBCFQ1AG4"
00EC139A	push dropped.F087E8	"TFFF59DMNKQ"
00EC13B3	push dropped.F08800	"CZW7AVAWP5WEMO2EYHQPLL"
00EC13CC	push dropped.F08830	"FLJV7M"
00EC13E5	push dropped.F08840	"U8UZODL7"
00EC13EA	push dropped.F0884C	"\t}-5+1?k"
00EC13FE	push dropped.F08858	"PEF41O086A21D5PF"
00EC1417	push dropped.F08880	"75F1TKYOXYBRL7OJD"
00EC141C	push dropped.F08894	"@\\(U;<t<,867b]C%*"
00EC1430	push dropped.F088A8	"4ZUCVY6UHPI3GIZSP2T8UN"
00EC1449	push dropped.F088D8	"EPOHCNIX9K5YJF4"

Figure 18. string references in the module



```
00EC138(    68 D487F000     push dropped.F087D4
00EC138E    BA 12000000     mov edx,12
00EC139(    A3 3C7EF100     mov dword ptr ds:[F17E3C],eax     00F17E3C:&"Electrum", eax:"BCRYPT.DLL"
00EC139'    E8 86320000     call dropped.EC4620
00EC139A    68 E887F000     push dropped.F087E8               F087E8:"TFFF59DMNKQ"
00EC139F    68 F487F000     push dropped.F087F4
00EC13A4    BA 0B000000     mov edx,B                         B:'\v'
00EC13A9    A3 047FF100     mov dword ptr ds:[F17F04],eax     00F17F04:&"\\Electrum\\wallets\\", eax:"BCRYPT.DLL"
00EC13AE    E8 6D320000     call dropped.EC4620
00EC13B3    68 0088F000     push dropped.F08800               F08800:"CZW7AVAWP5WEMO2EYHQPLL"
00EC13B8    68 1888F000     push dropped.F08818
00EC13BE    BA 16000000     mov edx,16
00EC13C2    A3 E87CF100     mov dword ptr ds:[F17CE8],eax     00F17CE8:&"ElectrumLTC", eax:"BCRYPT.DLL"
00EC13C7    E8 54320000     call dropped.EC4620
00EC13CC    68 3088F000     push dropped.F08830               F08830:"FLJV7M"
00EC13D1    68 3888F000     push dropped.F08838
00EC13D6    BA 06000000     mov edx,6
00EC13DB    A3 D07CF100     mov dword ptr ds:[F17CD0],eax     00F17CD0:&"\\Electrum-LTC\\wallets\\", eax:"BCRYPT.DLL"
00EC13E0    E8 3B320000     call dropped.EC4620
00EC13E5    68 4088F000     push dropped.F08840               F08840:"U8UZODL7"
00EC13EA    68 4C88F000     push dropped.F0884C               F0884C:"\t}-5+1?k"
00EC13EF    BA 08000000     mov edx,8
00EC13F4    A3 387CF100     mov dword ptr ds:[F17C38],eax     00F17C38:&"Exodus", eax:"BCRYPT.DLL"
00EC13F9    E8 22320000     call dropped.EC4620
00EC13FE    68 5888F000     push dropped.F08858               F08858:"PEF41O086A21D5PF"
00EC1403    68 6C88F000     push dropped.F0886C
00EC1408    BA 10000000     mov edx,10
00EC140D    A3 A880F100     mov dword ptr ds:[F180A8],eax     00F180A8:&"\\Exodus\\", eax:"BCRYPT.DLL"
00EC1411    E8 09320000     call dropped.EC4620
```

Figure 19. assembly view

The malware has a common function for encrypting strings and wallet addresses.

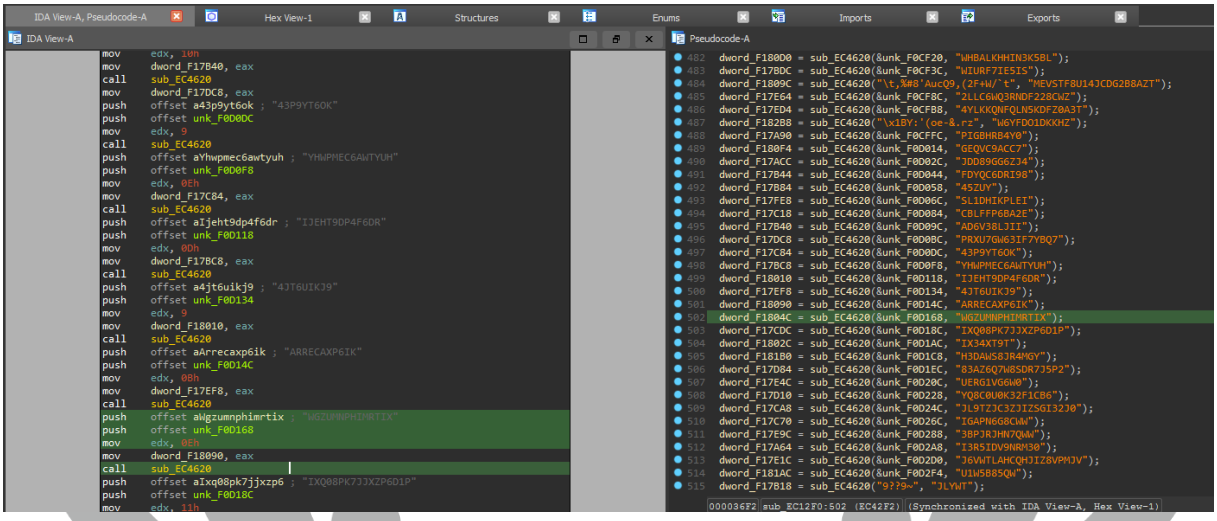


Figure 5. ida and pseudo code view

While many of these were strings used in wallet transactions, the decryption function (EC4620) was used in many parts of the malware.

```
_BYTE *__fastcall sub_EC4620(int a1, unsigned int a2, int a3, const char *a4)
{
  int v4; // ecx
  char *v5; // eax
  _WORD *v6; // edi
  __int16 v7; // ax
  _BYTE *v8; // ebx
  unsigned int v9; // eax
  unsigned int v10; // ecx
  unsigned int v11; // edx
  _BYTE *v12; // esi
  unsigned int v14; // [esp+4h] [ebp-210h] BYREF
  char v15; // [esp+8h] [ebp-20Ch] BYREF

  v14 = a2;
  v4 = 520;
  v5 = &v15;
  do
  {
    *v5++ = 0;
    --v4;
  }
  while ( v4 );
  v6 = (_WORD *)&v14 + 1;
  do
  {
    v7 = v6[1];
    ++v6;
  }
  while ( v7 );
  qmemcpy(
    v6,
    L"Nor again is there anyone who loves or pursues or desires to obtain pain of itself, because it
    0xCEu);
```

Figure 21. decryption function

The general structure of the function is as shown in the picture, but it also leaves a message in memory.

Browser plugins targeted by the malware:

Plug-in ID	Plug-in Name
gojhcdgcpbpfigcaejpfhfegekdgiblk	Opera Wallet
pnndplcbkakcplkjnolgbkdgjikjednm	Tronium
egjidjbpglichdcondbcbdnbeeppgdph	Trust Wallet
aholpfdialjgjfhomihkjbmgjidlcdno	Exodus Web3 Wallet
jnlgamecbpmbajjfhmmmlhejkemejdma	Braavos
kkpllkodjeloidieedojogacfhpaihoh	Enkrypt
mcohilncbfahbmgdjkbpemcciiolgcge	OKX Web3 Wallet
epapihdplajcdnnkdeiahlgigofloibg	Sender
gjagmgiddbbciopjhllkdnddhcglnemk	Hashpack
kmhcihpebfmpgmihbkipmjlmmioameka	Eternl
bgpipimickeadkjlklgciifhnalhdjhe	GeroWallet
phkbamefinggmakgklpkljjmgibohnba	Pontem Wallet
ejjladinnckdgjemekebdpeokbikhfci	Petra Wallet
efbglgofoippbgcjepnhiblaibcnclgk	Martian Wallet
cjmkndjhnagcfbpiemnkdpomccnjblmj	Finnie
aijcbedoijmgnlmjeegjaglmepbmpkpi	Leap Terra

YARA Rule

```
import "hash"

rule sample{

meta:

        author="Team3"

        description="Vidar Stealer"

        first date="16.09.2023"

        report date="16.10.2023"

        file name="sample.exe"

strings:

        $dnm_a="3h8W2nPBk4nRDUrB6Y0h0HLpyqaFdsG77R2qmHs6N8ItqBhW4SbYsSYEyutCXGUpq"

        $dnm_b="http://ocsp.entrust.net00"

        $dnm_c="tsUYxh4R2BMPl6IVK7msKOJi8MeYnj3B4ogS6KPyHbGwtYiJEr9efHvkNOaoLGqUp"

        $dnm_d="http://ocsp.digicert.com0X"

        $dnm_e="http://pki.eset.com/csp0"

Condition:

        Hash.md5(0,filesize)=="701477F861BDE9756D5FC3ACE9D2F019" or all of them

}
```

MITRE ATTACK TABLE

Execution	Persistence	Defense Evasion	Credential Access	C&C
Windows Management Instrumentation	Account Manipulation	Deobfuscate/Decode Files or Information	OS Credential Dumping	Encrypted Channel
Native API		Obfuscated Files or Information	Input Capture	Ingress Tool Transfer
		Virtualization/Sandbox Evasion	Credentials in Registry	Non-Application Layer Protocol
		Process Injection		Application Layer Protocol

Solution Suggestions

1. Up-to-date, reliable anti-virus software must be used on the systems,

2.Network packets should be filtered and monitored,

3.Make sure that the links clicked are correct and reliable,

4. More reliable cryptocurrency storage methods such as cold wallets should be preferred,

5. Your crypto accounts should use two-factor authentication,

6.In case of suspicion, the network should be monitored and intervened accordingly.

Prepared By

Elif ÇAĞLAR

https://www.linkedin.com/in/elif-%C3%A7a%C4%9Flar-27902b214/

Ömer Faruk SÖNMEZ

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

Melike TAŞDELEN

https://www.linkedin.com/in/melike-taşdelen-90345926a/