

Vidar Stealer

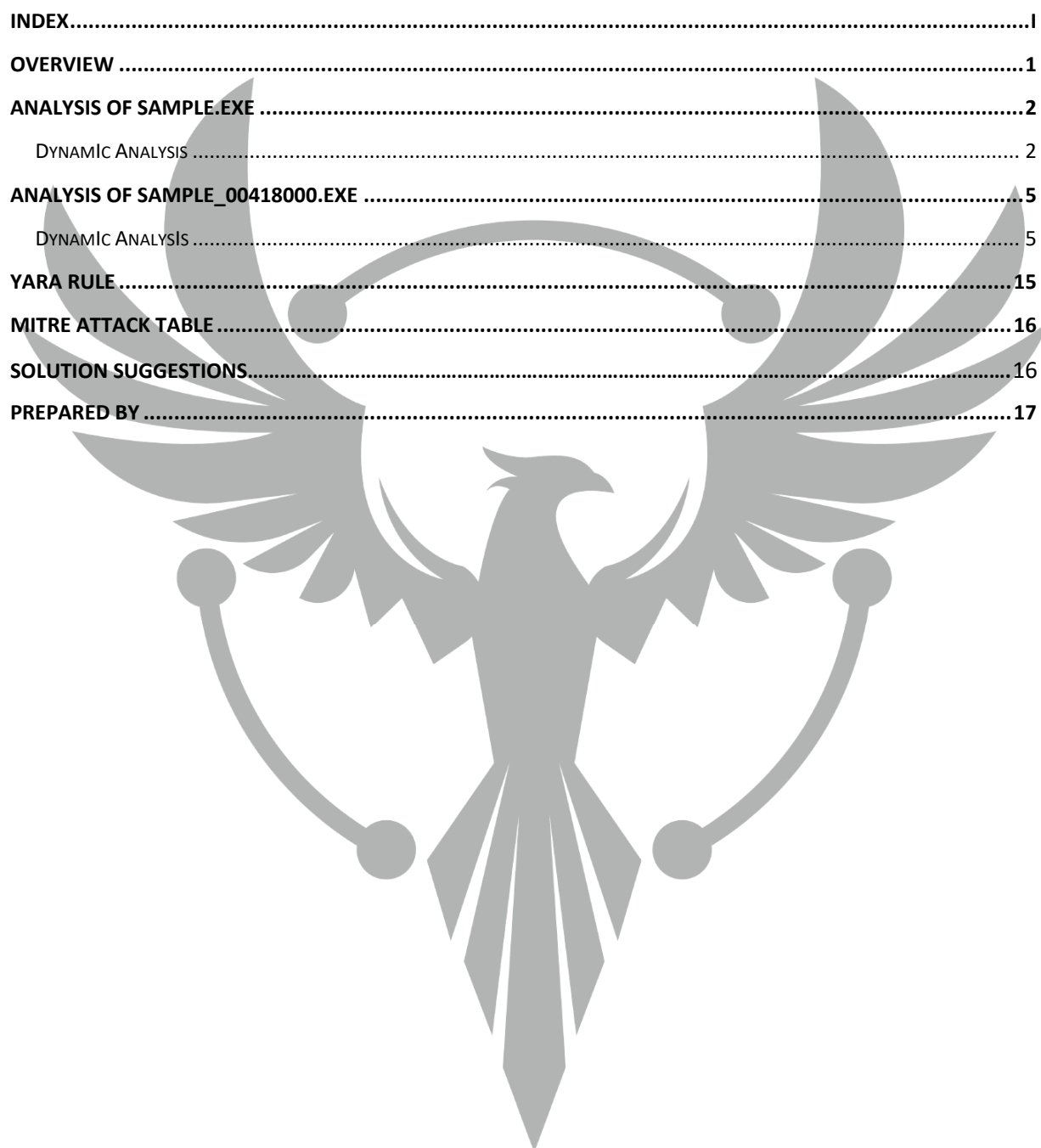
TECHNICAL ANALYSIS REPORT

ZAYOTEM

ZARARLI YAZILIM ÖNLEME VE TERSİNE MÜHENDİSLİK

Index

INDEX.....	I
OVERVIEW	1
ANALYSIS OF SAMPLE.EXE	2
DYNAMIC ANALYSIS	2
ANALYSIS OF SAMPLE_00418000.EXE	5
DYNAMIC ANALYSIS	5
YARA RULE	15
MITRE ATTACK TABLE	16
SOLUTION SUGGESTIONS.....	16
PREPARED BY	17





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.

00401080	55	push ebp	
00401081	8BEC	mov ebp,esp	
00401083	51	push esi	
00401084	C745 FC 00000000	mov dword ptr ss:[ebp-4],0	
00401088	EB 09	jmp sample.401096	
0040108D	8B45 FC	mov dword ptr ss:[ebp-4],eax	
00401090	83C0 01	add eax,1	
00401093	8945 FC	mov dword ptr ss:[ebp-4],eax	
00401096	8B4D FC	mov dword ptr ss:[ebp-4],edx	
00401099	3B4D 10	cmp dword ptr ss:[ebp+10],edx	
0040109C	73 35	jae sample.4010D3	
0040109E	8B45 FC	mov dword ptr ss:[ebp-4],edx	
004010A1	33D2	xor ebx,ebx	
004010A3	F775 14	div dword ptr ss:[ebp+14],ebx	
004010A6	8B45 08	mov dword ptr ss:[ebp+8],eax	[ebp+8]: "3h8w2nPBk4rRDurB6Y0h0Hlpy
004010A9	0FBED410	movsx eax,byte ptr ds:[eax+edx]	
004010AD	68C0 3B	imul eax,edx,3B	
004010B0	99	cdq	
004010B1	B9 24000000	mov ecx,24	24: '\$'
004010B6	F7F9	idiv eax	
004010B8	68C0 16	imul eax,edx,16	
004010BB	68C0 13	imul eax,edx,13	
004010BE	8B55 0C	mov dword ptr ss:[ebp+C],eax	
004010C1	0355 FC	add edi,dword ptr ss:[ebp-4]	
004010C4	0FB60A	movzx edi,byte ptr ds:[edx]	
004010C7	33C8	xor ecx,ecx	
004010C9	8B55 0C	mov dword ptr ss:[ebp+C],ecx	
004010CC	0355 FC	add edi,dword ptr ss:[ebp-4]	
004010CF	8B0A	mov byte ptr ds:[edx],al	
004010D1	E9 BA	jmp sample.4010D0	
004010D3	8BE5	mov esi,ebp	
004010D5	5D	pop ebp	
004010D6	C3	ret	

Figure 3. Jump operations in Memory

```

push 42h ; 'B'
push 5AE00h
push offset unk_418008
push offset aTsuyxh4r2bmp16 ; "tsUYxh4R2BMP16IVK7msKOJi8MeYnj3B4ogS6KP"...
call sub_401000

```

Figure 4. The string used by the malware

When the malware is examined, another string named **tsUYxh4R2BMP16IVK7msKOJi8MeYnj3B4ogS6KP** stands out. This string is assigned to the function named **sub_401000**.

```

0040102D 99          cdq
0040102E BE 32000000 mov esi,32
00401033 F7FE       idiv esi
00401035 8B45 08     mov eax,dword ptr ss:[ebp+8]
00401038 0FBE1410   movsx edx,byte ptr ds:[eax+edx]
0040103C 6BD2 28     imul edx,edx,28
0040103F 81E2 B5020000 and edx,2B5
00401045 33CA       xor ecx,edx
00401047 884D FB     mov byte ptr ss:[ebp-5],cl
0040104A 0FBE45 FB   movsx eax,byte ptr ss:[ebp-5]
0040104E 0FBE4D FB   movsx ecx,byte ptr ss:[ebp-5]
00401052 03C1       add eax,ecx
00401054 8B55 0C     mov edx,dword ptr ss:[ebp+C]
00401057 0355 FC     add edx,dword ptr ss:[ebp-4]
0040105A 8802       mov byte ptr ds:[edx],al
0040105C 0FBE45 FB   movsx eax,byte ptr ss:[ebp-5]
00401060 8B4D 0C     mov ecx,dword ptr ss:[ebp+C]
00401063 034D FC     add ecx,dword ptr ss:[ebp-4]
00401066 0FBE11     movsx edx,byte ptr ds:[ecx]
00401069 2BD0       sub edx,eax
0040106B 8B45 0C     mov eax,dword ptr ss:[ebp+C]
0040106E 0345 FC     add eax,dword ptr ss:[ebp-4]
00401071 8810       mov byte ptr ds:[eax],dl
00401073 EB 9B       jmp sample.401010
00401075 5E         pop esi
00401076 8BE5       mov esp,ebp
00401078 5D         pop ebp
00401079 C3         ret

```

eax=5AE00
dword ptr [ebp+c]=[0018F690]=sample.00418008
.text:0040106B sample.exe:\$106B #46B

Adres	Hex	ASCII
00418000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF	KZ.....yy
00418014	B8 00 00 00 00 00 00 00 40 00 00 00 00 00@.....
00418020	00 00 00 00 00 00 00 00 00 00 00 00 00 00
00418034	00 00 00 00 00 00 00 00 00 00 00 00 E8 00E8..
00418040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21	..o...i!..Li!
00418054	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E	is program can
00418068	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F	t be run in Dos
0041807C	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00	mode...\$.
00418088	55 F9 7E 46 11 98 10 15 11 98 10 15 11 98	U~F.....
0041809C	82 D6 88 15 10 98 10 15 7E EE 8E 15 0B 98	..Ö.....~i...
004180A8	7E EE 8B 15 2D 98 10 15 7E EE BA 15 8D 98	~i».....~i°...
004180BC	18 E0 93 15 14 98 10 15 18 E0 83 15 1C 98	..à.....~i...
004180C8	11 98 11 15 7E 98 10 15 7E EE BF 15 1C 98	..~i.....~i...
004180D4	7E EE 8D 15 10 98 10 15 52 69 63 68 11 98	..~i.....Rich...
004180E8	00 00 00 00 00 00 00 00 50 45 00 00 4C 01PE..L...
004180F4	26 7A FC 63 00 00 00 00 00 00 00 00 E0 00	&züc.....à...
00418100	0B 01 0A 00 00 34 04 00 00 AC 02 00 00 00	...4...~....
0041810C	CC FB 02 00 00 10 00 00 00 50 04 00 00 00	...P...f...

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.
GetCurrentProcessId	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.

```

.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 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	
00E423D8	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],al	
00E423E7	397D 3C	cmp dword ptr ss:[ebp+3C],edi	
00E423EA	72 0C	jb dropped.E423F8	
00E423EC	884D 28	mov eax,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"
00E42405	5B	pop ebx	

Figure 10. Telegram bot URL address

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

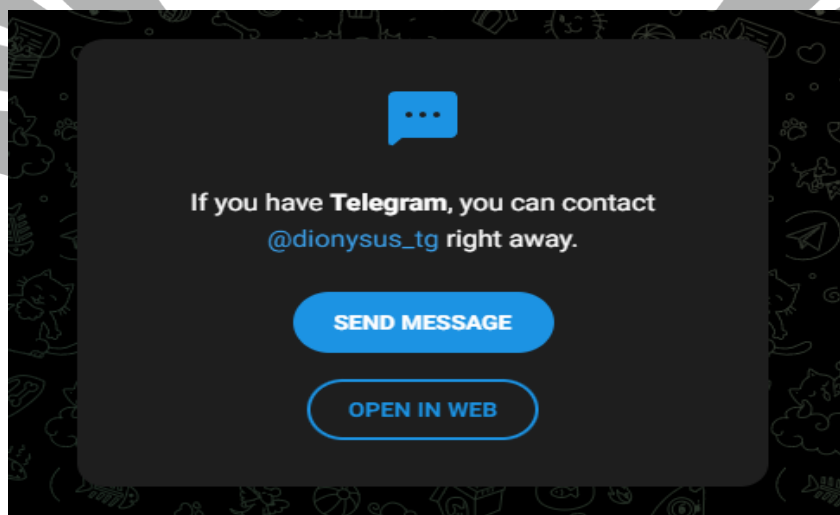


Figure 11. Telegram bot

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	8B06	mov byte ptr ds:[esi],al	esi: "ERROR"
00E3E7DE	E8 0DA9FFFF	call dropped.E390F0	esi: "ERROR"
00E3E7E3	8BC6	mov eax,esi	
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.

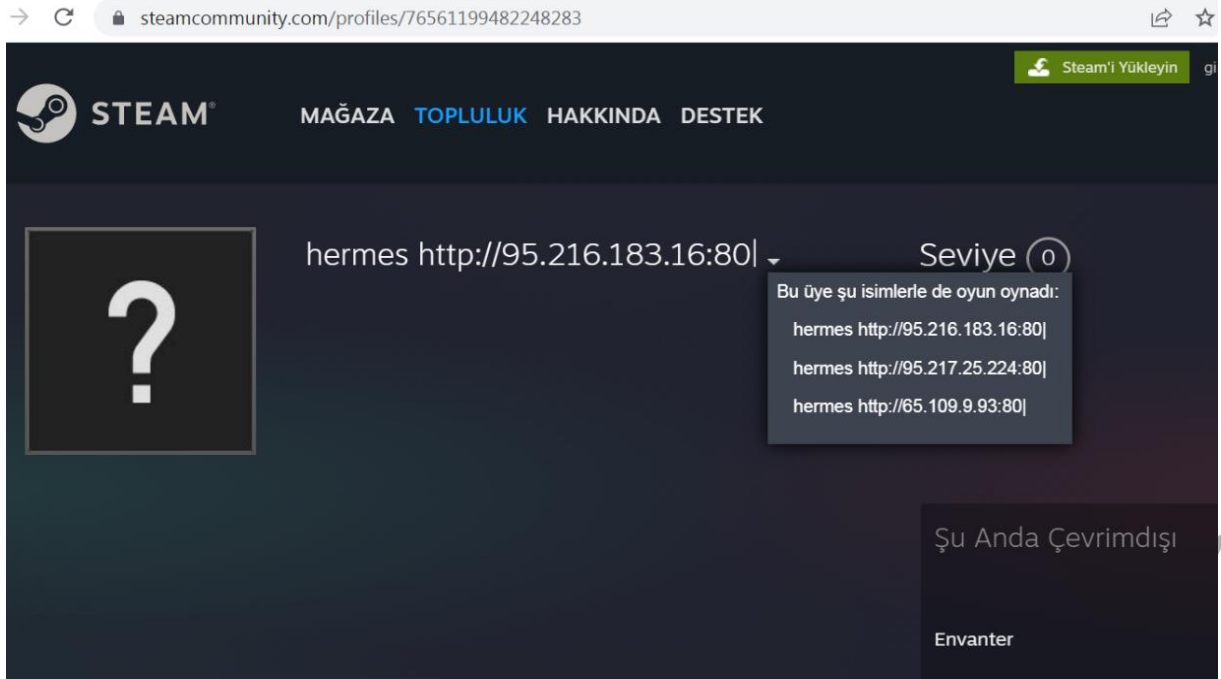


Figure 2. Steam profile page

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.

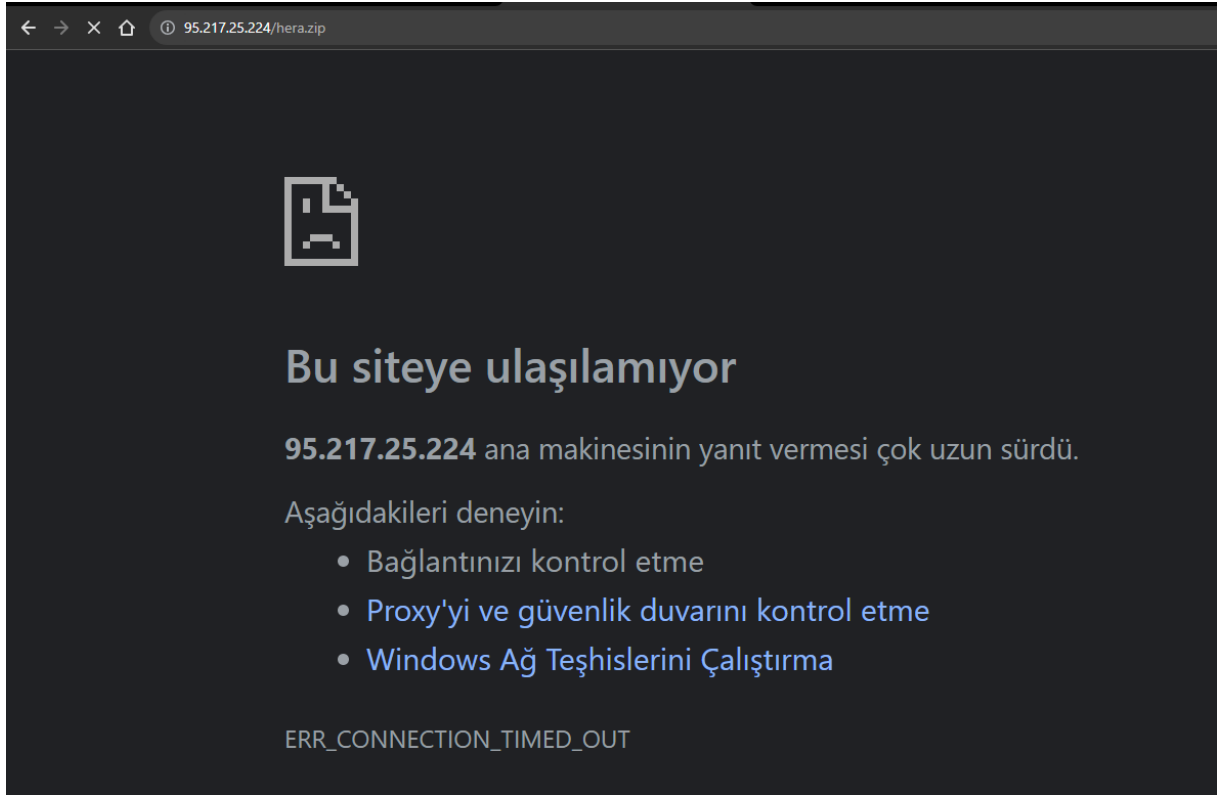


Figure 14. web server to pull hera.zip

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

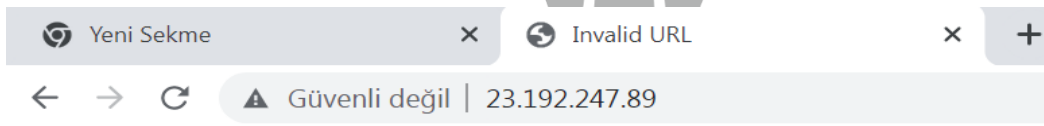
gorev(dropped)...	696	TCP Receive	WIN-L1KDN79P80J.localdomain:49276 -> 149.154.167.99:https
gorev(dropped)...	696	TCP Receive	WIN-L1KDN79P80J.localdomain:49276 -> 149.154.167.99:https
gorev(dropped)...	696	TCP Receive	WIN-L1KDN79P80J.localdomain:49276 -> 149.154.167.99:https
gorev(dropped)...	696	TCP Connect	WIN-L1KDN79P80J.localdomain:49277 -> a23-192-247-89:deploy.static.akamaitechnologies.com:https
gorev(dropped)...	696	TCP Send	WIN-L1KDN79P80J.localdomain:49277 -> a23-192-247-89:deploy.static.akamaitechnologies.com:https
gorev(dropped)...	696	TCP Receive	WIN-L1KDN79P80J.localdomain:49277 -> a23-192-247-89:deploy.static.akamaitechnologies.com:https
gorev(dropped)...	696	TCP Receive	WIN-L1KDN79P80J.localdomain:49277 -> a23-192-247-89:deploy.static.akamaitechnologies.com:https
gorev(dropped)...	696	TCP Receive	WIN-L1KDN79P80J.localdomain:49277 -> a23-192-247-89:deploy.static.akamaitechnologies.com:https
gorev(dropped)...	696	TCP Receive	WIN-L1KDN79P80J.localdomain:49277 -> a23-192-247-89:deploy.static.akamaitechnologies.com:https
gorev(dropped)...	696	TCP Send	WIN-L1KDN79P80J.localdomain:49277 -> a23-192-247-89:deploy.static.akamaitechnologies.com:https
gorev(dropped)...	696	TCP Receive	WIN-L1KDN79P80J.localdomain:49277 -> a23-192-247-89:deploy.static.akamaitechnologies.com:https
gorev(dropped)...	696	RegQueryKey	HKLM
gorev(dropped)...	696	RegQueryKey	HKLM
gorev(dropped)...	696	RegOpenKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Cryptography\Defaults\Provider\Microsoft Enhanced RSA and AES (
gorev(dropped)...	696	RegSetInfoKey	HKLM\SOFTWARE\Wow6432Node\Microsoft\Cryptography\Defaults\Provider\Microsoft Enhanced RSA and AES (
gorev(dropped)...	696	RegQueryValue	HKLM\SOFTWARE\Wow6432Node\Microsoft\Cryptography\Defaults\Provider\Microsoft Enhanced RSA and AES (

Figure 15. Request discarded from local machine

829	646.855253	23.192.247.89	192.168.213.128	TLSv1.2	85 Encrypted Alert
830	646.855305	192.168.213.128	23.192.247.89	TCP	54 49181 -> 443 [RST, ACK] Seq=674 Ack=39916 Win=0 Len=0
831	646.856503	23.192.247.89	192.168.213.128	TCP	60 443 -> 49185 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
832	646.856553	192.168.213.128	23.192.247.89	TCP	54 49185 -> 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
833	646.856921	192.168.213.128	23.192.247.89	TLSv1.2	275 Client Hello
834	646.857097	23.192.247.89	192.168.213.128	TCP	60 443 -> 49185 [ACK] Seq=1 Ack=222 Win=64240 Len=0
835	646.916788	23.192.247.89	192.168.213.128	TLSv1.2	1514 Server Hello
836	646.916788	23.192.247.89	192.168.213.128	TCP	1498 443 -> 49185 [PSH, ACK] Seq=1461 Ack=222 Win=64240 Len=1444 [TCP segm...
837	646.916858	192.168.213.128	23.192.247.89	TCP	54 49185 -> 443 [ACK] Seq=222 Ack=2905 Win=64240 Len=0
838	646.917070	23.192.247.89	192.168.213.128	TLSv1.2	995 Certificate, Certificate Status, Server Key Exchange, Server Hello D...
839	646.917092	192.168.213.128	23.192.247.89	TCP	54 49185 -> 443 [ACK] Seq=222 Ack=3846 Win=63299 Len=0
840	646.927153	192.168.213.128	23.192.247.89	TLSv1.2	180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
841	646.927426	23.192.247.89	192.168.213.128	TCP	60 443 -> 49185 [ACK] Seq=3846 Ack=348 Win=64240 Len=0
842	646.982192	23.192.247.89	192.168.213.128	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
843	646.982246	192.168.213.128	23.192.247.89	TCP	54 49185 -> 443 [ACK] Seq=348 Ack=3897 Win=63248 Len=0
844	646.987164	192.168.213.128	23.192.247.89	TLSv1.2	379 Application Data
845	646.987544	23.192.247.89	192.168.213.128	TCP	60 443 -> 49185 [ACK] Seq=3897 Ack=673 Win=64240 Len=0

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.



Invalid URL

The requested URL "[no URL]", is invalid.

Reference #9.5d161102.1697050990.e66b895

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"
00EC1120	push dropped.F08560	"3CEUMWUK5A45"
00EC1146	push dropped.F08580	"1R54YZTJ"
00EC1148	push dropped.F0858C	"J! 'F:: \v"
00EC115F	push dropped.F08598	"4XBHA52K4TL208"
00EC1178	push dropped.F085B8	"CKU9Y"
00EC1191	push dropped.F085C8	"9EWVS32T03I0J"
00EC11AA	push dropped.F085E8	"WH4WB7UQ4KQ"
00EC11C3	push dropped.F08600	"LYBR3PHKZBUFICTOI"
00EC11DC	push dropped.F08628	"MBQV28GGDQJXPIE3FJ"
00EC11F5	push dropped.F08650	"31LZLV8BIB8W"
00EC120E	push dropped.F08670	"TSTLD345SX5"
00EC1227	push dropped.F08688	"K088WK3PM"
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	"TGYEN2CT06XN"
00EC12F0	push dropped.F08740	"GL90T06"
00EC1304	push dropped.F08750	"F9HVXI8"
00EC131D	push dropped.F08760	"B8LG4ZKR"
00EC1322	push dropped.F0876C	"J]540597"
00EC1336	push dropped.F08778	"2V0RM1ZIG"
00EC133B	push dropped.F08784	"w\ 'x7?T/\$e"
00EC134F	push dropped.F08790	"R3Z362PG0N"
00EC1368	push dropped.F087A8	"6GMSA500"
00EC136D	push dropped.F087B4	"s+ (05G:J"
00EC1381	push dropped.F087C0	"DIUB7DYTOXBCFQ1AG4"
00EC139A	push dropped.F087E8	"TFF59DMNKQ"
00EC13B3	push dropped.F08800	"CZW7AVAWP5WEMO2EYHQPLL"
00EC13CC	push dropped.F08830	"FLJ3V7M"
00EC13E5	push dropped.F08840	"U8UZODL7"
00EC13EA	push dropped.F0884C	"\t}-5+1?k"
00EC13FE	push dropped.F08858	"PEF410086A21D5PF"
00EC1417	push dropped.F08880	"75F1TKYXYBRL70JD"
00EC141C	push dropped.F08894	"@\\(U; < < , 867b]C%*"
00EC1430	push dropped.F088A8	"4ZUCVY6UHP13GIZSP2T8UN"
00EC1449	push dropped.F088D8	"EP0HCNIX9KSYJF4"

Figure 18. string references in the module

00EC1381	68 D487F000	push dropped.F087D4	
00EC1381	BA 12000000	mov edx,12	
00EC1391	A3 3C7EF100	mov dword ptr ds:[F17E3C],eax	00F17E3C:"Electrum", eax:"BCRYPT.DLL"
00EC1391	E8 86320000	call dropped.EC4620	
00EC1391	68 E887F000	push dropped.F087E8	F087E8:"TFF59DMNKQ"
00EC1391	68 F487F000	push dropped.F087F4	
00EC13A1	BA 0B000000	mov edx,B	B:"\v"
00EC13A1	A3 047FF100	mov dword ptr ds:[F17F04],eax	00F17F04:"\\Electrum\\wallets\\", eax:"BCRYPT.DLL"
00EC13A1	E8 6D320000	call dropped.EC4620	
00EC13B1	68 0088F000	push dropped.F08800	F08800:"CZW7AVAWP5WEMO2EYHQPLL"
00EC13B1	68 1888F000	push dropped.F08818	
00EC13B1	BA 16000000	mov edx,16	
00EC13C1	A3 E87CF100	mov dword ptr ds:[F17CE8],eax	00F17CE8:"ElectrumLTC", eax:"BCRYPT.DLL"
00EC13C1	E8 54320000	call dropped.EC4620	
00EC13C1	68 3088F000	push dropped.F08830	F08830:"FLJ3V7M"
00EC13D1	68 3888F000	push dropped.F08838	
00EC13D1	BA 06000000	mov edx,6	
00EC13D1	A3 D07CF100	mov dword ptr ds:[F17CD0],eax	00F17CD0:"\\Electrum-LTC\\wallets\\", eax:"BCRYPT.DLL"
00EC13E1	E8 3B320000	call dropped.EC4620	
00EC13E1	68 4088F000	push dropped.F08840	F08840:"U8UZODL7"
00EC13E1	68 4C88F000	push dropped.F0884C	F0884C:"\t}-5+1?k"
00EC13F1	BA 08000000	mov edx,8	
00EC13F1	A3 387CF100	mov dword ptr ds:[F17C38],eax	00F17C38:"Exodus", eax:"BCRYPT.DLL"
00EC13F1	E8 22320000	call dropped.EC4620	
00EC13F1	68 5888F000	push dropped.F08858	F08858:"PEF410086A21D5PF"
00EC1401	68 6C88F000	push dropped.F0886C	
00EC1401	BA 10000000	mov edx,10	
00EC1401	A3 A880F100	mov dword ptr ds:[F180A8],eax	00F180A8:"\\Exodus\\", eax:"BCRYPT.DLL"
00EC1411	68 08320000	call dropped.EC4620	

Figure 19. assembly view

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

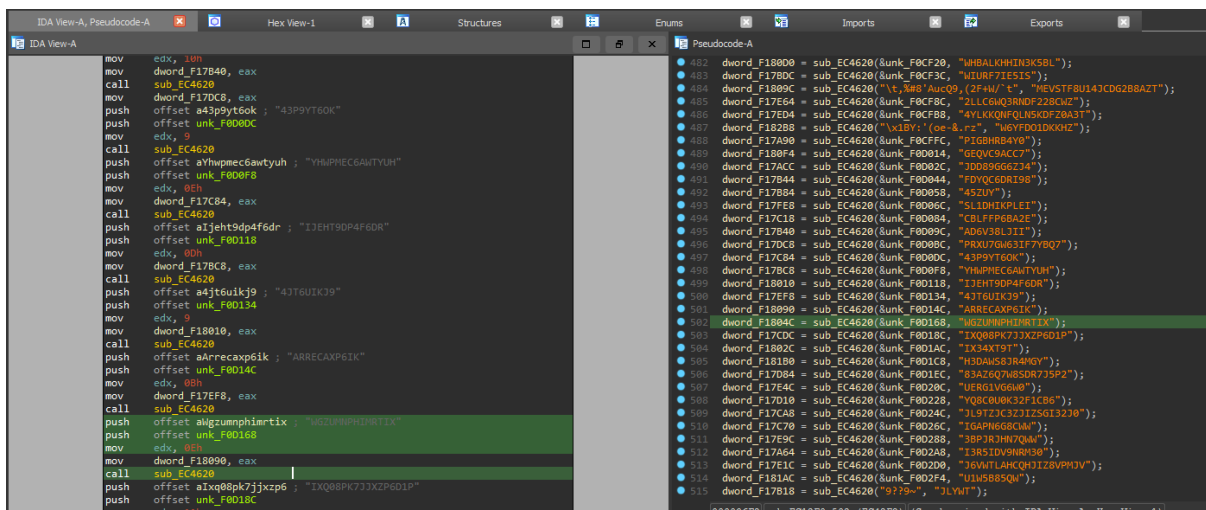


Figure 5. ida and pseudo code view

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

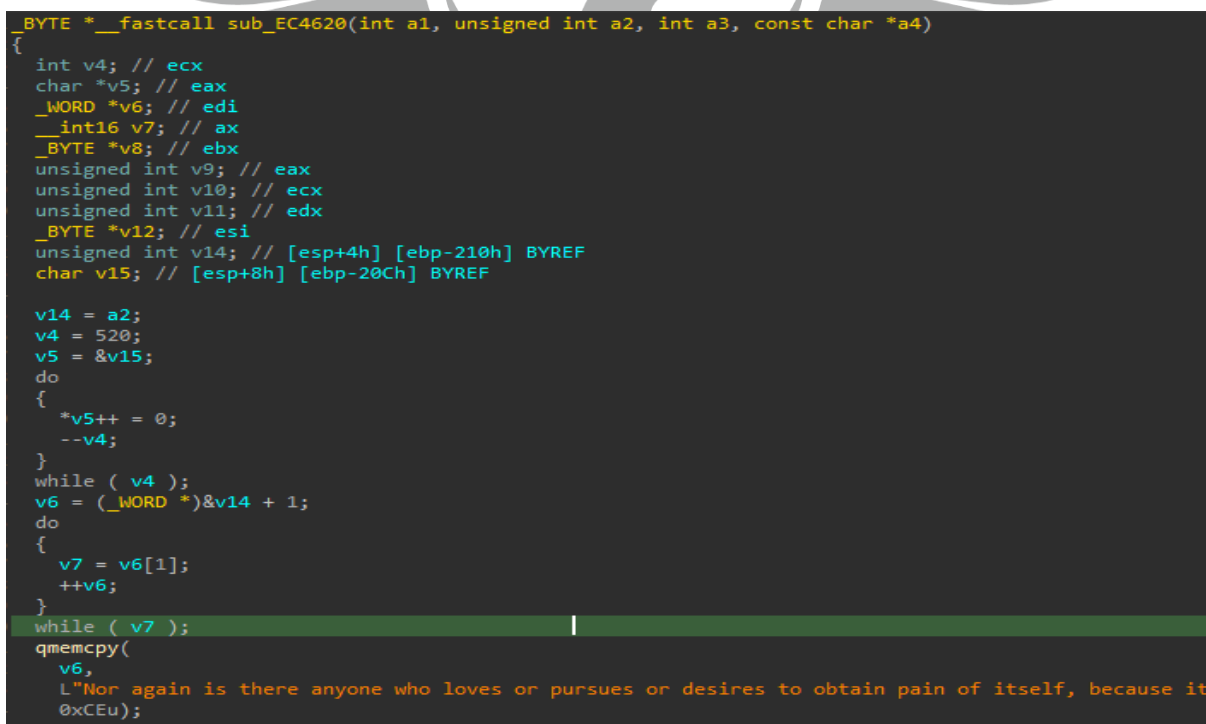


Figure 21. decryption function

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
gojhcdgcpbpfigcaeipfhfegekdgiblk	Opera Wallet
pnndplcbkakcplkjnlgbkdgjikjednm	Tronium
egjidjbpglichdcondbcdbnbeepgdph	Trust Wallet
aholpfdialjgjfhomihkjbmgiidlcdno	Exodus Web3 Wallet
jnlgamecbpmbajjfhmmmlhejkemejdma	Braavos
kkpllkdjeloidieedojogacfhpaihoh	Enkrypt
mcohilncbfahbmgdjkbpemcciolgcge	OKX Web3 Wallet
epapihdplajcdnnkdeiahlgigofloibg	Sender
gjagmgiddbbciopjhllkdnddhcglnemk	Hashpack
kmhcihpebfmpgmihbkipmjlmioameka	Eternl
bgpipimickeadkjlklgciifhnalhdjhe	GeroWallet
phkbamefinggmakgklpklljmgibohnba	Pontem Wallet
eijladinnckdgjemekebdpeokbikhfci	Petra Wallet
efbglgofoippbgcjepnhiblaibcncgk	Martian Wallet
cjmkndjhnagcfbpiemnkdpomccnjbmlj	Finnie
aijcbedoijmgnlmjeegjaglmepbmkpi	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="3h8W2nPBk4nRDUrB6Y0h0HLpyqaFdsG77R2qmHs6N8ltqBhW4SbYsSYEyutCXGUpq"

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

    $dnm_c="tsUYxh4R2BMPI6IVK7msKOJi8MeYnj3B4ogS6KPyHbGwtYiJEr9efHvkNOaoLGqUp"

    $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 AĐLAR

<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/>