



CYBER THREAT INTELLIGENCE



RACCOONSTEALER V2.0

TECHNICAL ANALYSIS REPORT

Contents

Executive Summary	2
File.exe Analysis	3
General Review	3
Stage 2 Analysis	4
DLL Detection	4
Process Detection.....	5
Computer Name Detection	6
Username Detection	7
InstallUtil.exe Analysis.....	8
General Review	8
Dynamic Analysis.....	8
Getting API Function Address.....	8
String Resolving Algorithm	9
Process Access Detection	9
Creation of Request Contents	12
Network Analysis.....	13
Request Analysis	13
After Response	15
Gathering Device Information.....	15
DLL Loading	19
Database Operations	19
File Traversal Algorithm	20
Additional Analysis	21
SQL Queries	21
YARA Rule	22
MITRE ATTACK TABLE	23
Mitigations	23

Executive Summary

Raccoon Stealer V2.0 is an advanced version of the malicious software Raccoon Stealer. This malware is used by hackers to steal sensitive information from target computers and use personal information for malicious purposes.

Features of Raccoon Stealer V2.0 include the ability to steal users' browser histories, cookies, logins, and other sensitive information. The effects of Raccoon Stealer V2.0 can pose a serious threat to organisations. This malware can seriously compromise organisations' data security, damage their reputation and cause financial losses.

Organisations can protect against threats such as Raccoon Stealer V2.0 by adopting a strong security strategy and raising user awareness. Keeping security software up to date, using strong encryption methods and conducting regular security audits can help minimise the impact of this type of malware.

To detect and mitigate Raccoon Stealer V2.0, organisations should seek support from security experts and security software, and be prepared to deal with threats through continuous monitoring and updates.

File.exe Analysis

General Review

SHA 256	1976859574585aac13a24b6696cec26479029a92334c721ec71492094a7edec3
Name	file.exe
File Type	PE32-EXE

Table 1 file.exe file information

```

    push r11,55C4B0
    mov edx,dword ptr ds:[55C8E0]
    push edx
    call dword ptr ds:[<&GetProcAddress>]
    mov dword ptr ds:[<&VirtualProtect>],eax
    lea eax,dword ptr ss:[ebp-4]
    push eax
    push 40
    mov ecx,dword ptr ss:[ebp+C]
    push ecx
    mov edx,dword ptr ss:[ebp+8]
    push edx
    call dword ptr ds:[<&VirtualProtect>]
    mov esp,ebp
    pop ebp
    ret

```

dword ptr [0055C498 <file.&VirtualProtect>]=<kernel32.virtualProtect>

.text:0054BA62 file.exe:\$16BA62 #16AE62

Döküm1	Döküm2	Döküm3	Döküm4	Döküm5	İzle 1	[x] Yerel Değişkenler	Yapı
Adres	Hex	ASCII					
025E0020	4D 5A 45 52 E8 00 00 00 00 58 83 E8 09 50 05 00	MZERè...X.è.P..					
025E0030	E0 15 00 FF D0 C3 00 00 40 00 00 00 00 00 00 00	à.yDÀ..@.....					
025E0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
025E0050	00 00 00 00 00 00 00 00 00 00 00 78 00 00 00 00	..o...!i..L!tH					
025E0060	0E 1F BA OE 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..o...!i..L!tH					
025E0070	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno					
025E0080	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS					
025E0090	60 6F 64 65 2E 24 00 00 50 45 00 00 4C 01 03 00	mode.\$..PE.L...					
025E00A0	4E 1E 2D 63 00 00 00 00 00 00 00 00 E0 00 02 01	Ná-C.....à...					
025E00B0	08 01 0E 00 00 5E 12 00 00 50 03 00 00 00 40 00^..P.....ü..					
025E00D0	00 10 00 00 00 10 00 00 06 00 00 00 00 00 00 00					
025E00E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					

Figure 1 Obtaining executable permission for the area where the analysed code is written

```

    stc
    push ds
    jmp file.549D83
    pushfd
    les edx,fword ptr ss:[ebp]
    mov dword ptr ss:[ebp-4],eax
    call dword ptr ss:[ebp-4]
    mov esp,ebp
    pop ebp
    ret

```

dword ptr [ebp-4]=[001AEF48]=025E0020

.text:00549DE9 file.exe:\$169DE9 #169E9

Döküm1	Döküm2	Döküm3	Döküm4	Döküm5	İzle 1	[x] Yerel Değişkenler	Yapı
Adres	Hex	ASCII					
025E0020	4D 5A 45 52 E8 00 00 00 00 58 83 E8 09 50 05 00	MZERè...X.è.P..					
025E0030	E0 15 00 FF D0 C3 00 00 40 00 00 00 00 00 00 00	à.yDÀ..@.....					
025E0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00					
025E0050	00 00 00 00 00 00 00 00 00 00 00 78 00 00 00 00	..o...!i..L!tH					
025E0060	0E 1F BA OE 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..o...!i..L!tH					
025E0070	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno					
025E0080	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS					
025E0090	60 6F 64 65 2E 24 00 00 50 45 00 00 4C 01 03 00	mode.\$..PE.L...					
025E00A0	4E 1E 2D 63 00 00 00 00 00 00 00 00 E0 00 02 01	Ná-C.....à...					
025E00B0	08 01 0E 00 00 5E 12 00 00 50 03 00 00 00 40 00^..P.....ü..					
025E00D0	00 10 00 00 00 10 00 00 06 00 00 00 00 00 00 00					

Figure 2 Call to the starting address of the decoded code

Malware has been detected and unpacked.

Stage 2 Analysis

At this stage, it was determined that malware applied analysis detection techniques.

DLL Detection

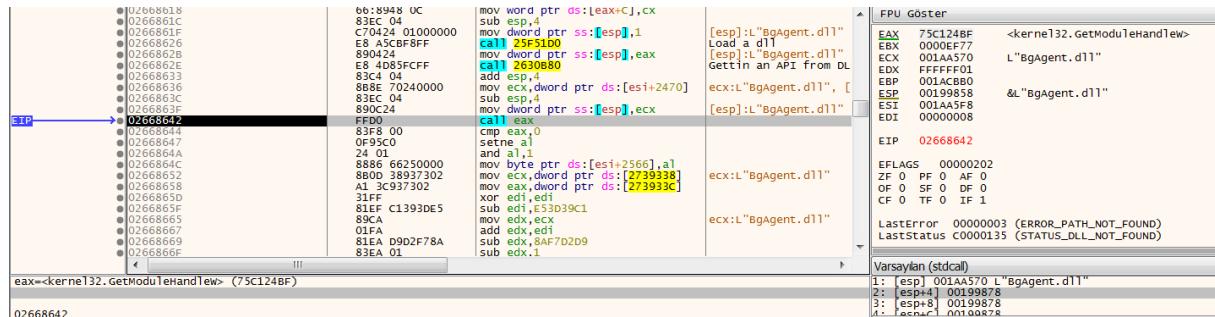


Figure 3 Determination of the existence of the dynamic library names analysed at runtime on the computer

It has been observed that the malware tries to detect DLL files belonging to some security products and systems. The DLLs it tries to detect and the systems they belong to are as follows:

CWSandbox	api_log.dll
	dir_watch.dll
	pstorec.dll
Sandboxie	sbieDII.dll
ThreatExpert	dbghelp.dll
Comodo	cmdvrt32.dll /cmdvrt64.dll
BullGuard	BgAgent.dll

Table 2 Names of the checked dynamic library files and the systems they belong to

Process Detection

The screenshot shows the assembly view of a debugger. The CPU pane displays assembly code for the `kernel32.dll`. A blue arrow points to the instruction at address `0x265981F`, which is `FFD0`. The Registers pane shows various registers with their current values.

Figure 4 Taking a snapshot of the processes running in the background

0x2654A00	08c2	or dl,al		▲ FPU Göster
0x2654A11	B8 B2E0F160	mov eax,6001E8B2		EAX 75CD62D <kernel32.1strcmpiW>
0x2654A16	B9 E5213FB	mov ecx,FB13523C		EBX C47E000
0x2654A18	B8 01	test dl,1		ECX 00000000
0x2654A19	0005C0	cmpl dl,ecx		EDX 001995C4 L["BullGuardCore.exe"]
0x2654A21	8945 D0	mov dword ptr ss:[ebp-30],eax		EBP 00199850 L["System Process"]
0x2654A24	E9 00160000	jmp 2658A29		ESP 00199594 &L["System Process"]
0x2654A25	8945 00	mov dword ptr ss:[esp+1],eax		ESI 00199801
0x2654A2C	C70424 01000000	[esp]:L["[System Process]"]		EDI FFFFFFFF
0x2654A33	E8 98A0F9F6	call 25F5100		EIP 0265A458
0x2654A34	8945 00	mov dword ptr ss:[esp+1],eax		EFLAGS 00000020
0x2654A38	E8 401B0000	[esp]:L["[System Process]"]		ZF 0 PF 0 AF 0
0x2654A40	83C4 04	add esp,4		OF 0 SF 0 DF 0
0x2654A41	89D9 DC	mov edx,dword ptr ss:[bp-24]		CF 0 TF 0 IF 1
0x2654A42	8809	mov edx,dword ptr ds:[ecx]		LastError 0000007E (ERROR_MOD_NOT_FOUND)
0x2654A48	8B55 E0	mov edx,dword ptr ss:[ebp-20]		LastStatus C0000135 (STATUS_DLL_NOT_FOUND)
0x2654A49	83C2 24	add edx,24		
0x2654A4B	8945 08	sub edx,8		
0x2654A4C	891424	mov dword ptr ss:[esp+1],edx		
0x2654A51	894C24 04	mov dword ptr ss:[esp+4],ecx		
0x2654A55	8945 00	[esp]:L["[System Process]"]		
0x2654A56	83F8 00	cmp eax,x0		
0x2654A59	0F94C0	sete al		
0x2654A5D	4			Varsayılan (stdcall)

Figure 5 Process Blacklist Control

Figure 6. Process Blacklist Control

It was observed that the malware compares the processes running in the background with its own blacklist. The process list compared is as follows:

- fmon.exe
 - WRSA.exe
 - PSUAService.exe
 - BullGuardCore.exe

ECHO

Computer Name Detection

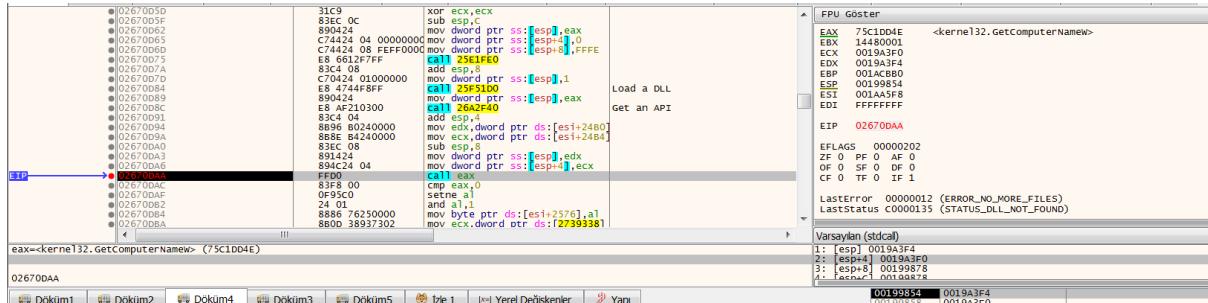


Figure 7 Withdrawing the victim's computer name

0267782A		Döküm1	Döküm2	Döküm4	Döküm3	Döküm5	İzle 1	İx=l Yerel Değişken
Adres	Hex	ASCII						
0019A2A4	B4 A2 19 00	B4	A2	19	00			
0019A2B4	37 00 53 00	49	00	4C	00	56	00	49
0019A2C4	6B 00 6C 00	6F	00	6E	00	65	00	5F
0019A2D4	34 00 2D 00	70	00	63	00	00	00	00
0019A2E4	73 00 69 00	64	00	65	00	54	00	6D
0019A2F4	54 00 55 00	2D	00	34	00	4E	00	48
0019A304	53 00 4D 00	43	00	47	00	31	00	48
0019A314	54 00 45 00	51	00	55	00	49	00	4C
0019A324	4F 00 4F 00	4D	00	42	00	4F	00	4F
0019A334	46 00 4F 00	52	00	54	00	49	00	4E
0019A344	00 00 00 00	57	00	49	00	4E	00	37
0019A354	52 00 41 00	50	00	53	00	00	00	00
0019A364	45 00 4C 00	4C	00	45	00	52	00	2D
0019A374	00 00 00 00	48	00	41	00	4E	00	53
0019A384	54 00 45 00	52	00	2D	00	50	00	43
0019A394	1A 00 4F 00	48	00	4E	00	2D	00	50
0019A3A4	53 00 41 00	4E	00	44	00	42	00	4F
0019A3B4	74 00 7A 00	00	00	00	00	4E	00	66
0019A3C4	46 00 62 00	50	00	66	00	48	00	00
0019A3D4	76 00 64 00	68	00	78	00	00	00	45
0019A3E4	49 00 43 00	5A	00	00	00	FO	A3	19
0019A3F4	57 00 49 00	4E	00	2D	00	4C	00	31

Figure 8 Black list of computer names after analyses

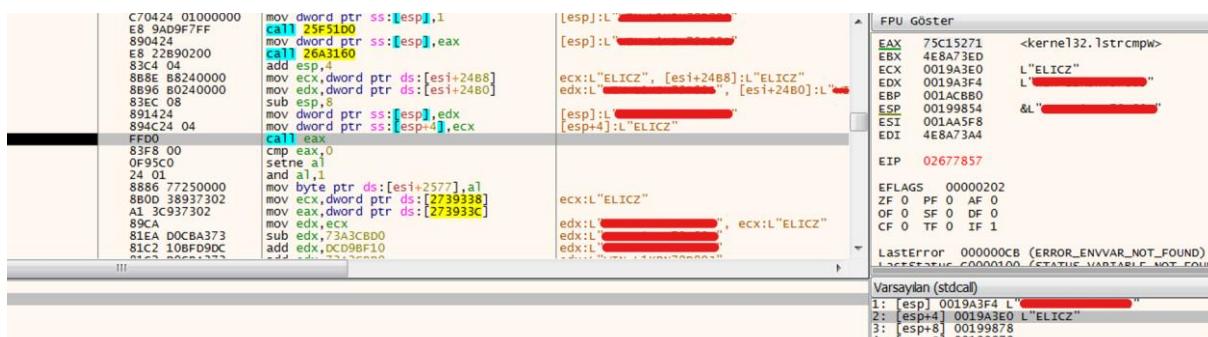


Figure 9 Comparing the computer names in the list with the victim computer name in order

It was observed that malware analyses a list of names and compares it with the name of the computer on which it was found. Analysed computer names:

SANDBOX	JOHN-PC	HANSPETER-PC	MUELLER-PC
WIN7-TRAPS	FORTINET	TEQUILABOOMBOOM	TU-4NH09SMCG1HC
InsideTm	klone_x64-pc	7SILVIA	tz
NfZt	FbPfH	hfvdhx	ELICZ

Table 3 Analysed computer names

ECHO

Username Detection

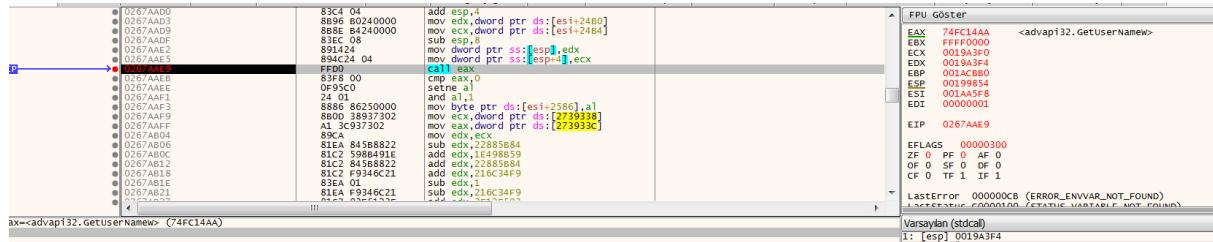


Figure 10 User name retrieval process

0019A0D8	E4 A0 19 00	E4 A0 19 00	E4 A0 19 00	54 00 45 00	ā . . ā . . T. E.
0019A0E8	51 00 55 00	49 00 4C 00	41 00 42 00	4F 00 4F 00	Q. U. I. L. A. B. O. O.
0019A0F8	4D 00 42 00	4F 00 4F 00	4D 00 00 00	73 00 61 00	M. B. O. M. S. A.
0019A108	6E 00 64 00	62 00 6F 00	78 00 00 00	74 00 69 00	n. d. b. o. x. t. i.
0019A118	5D 00 6D 00	79 00 00 00	4A 00 6F 00	68 00 6E 00	m. m. y. . J. o. h. n.
0019A128	20 00 44 00	6F 00 65 00	00 00 00 00	77 00 69 00	. D. o. e. . . . w. i.
0019A138	6C 00 62 00	65 00 72 00	74 00 00 00	76 00 69 00	T. b. e. r. t. . . v. i.
0019A148	72 00 75 00	73 00 63 00	6C 00 6F 00	6E 00 65 00	r. u. s. c. l. o. n. e.
0019A158	00 00 00 00	73 00 6E 00	6F 00 72 00	74 00 00 00	. . . s. n. o. r. t.
0019A168	41 00 6E 00	64 00 79 00	00 00 00 00	76 00 69 00	A. n. d. y. . . . v. i.
0019A178	72 00 75 00	73 00 65 00	74 00 65 00	73 00 74 00	r. u. s. t. e. s. t.
0019A188	20 00 75 00	73 00 65 00	72 00 00 00	6D 00 61 00	u. s. e. r. . . . m. a.
0019A198	6C 00 74 00	65 00 73 00	74 00 00 00	6D 00 61 00	l. t. e. s. t. . . m. a.
0019A1A8	6C 00 77 00	61 00 72 00	65 00 00 00	73 00 61 00	l. w. a. r. e. . . s. a.
0019A1B8	5E 00 64 00	20 00 62 00	6F 00 78 00	00 00 00 00	n. d. . b. o. x. . . .
0019A1C8	50 00 65 00	74 00 65 00	72 00 20 00	57 00 69 00	P. e. t. e. r. . . w. i.
0019A1D8	6C 00 73 00	6F 00 6E 00	00 00 00 00	6D 00 69 00	l. s. o. n. . . . m. i.
0019A1E8	6C 00 6F 00	7A 00 73 00	00 00 00 00	4D 00 69 00	l. o. z. s. . . . M. i.
0019A1F8	6C 00 6C 00	65 00 72 00	00 00 00 00	4A 00 6F 00	l. l. e. r. . . . J. o.
0019A208	68 00 6E 00	73 00 6F 00	6E 00 00 00	49 00 54 00	h. n. s. o. n. . I. T.
0019A218	2D 00 41 00	44 00 4D 00	49 00 4E 00	00 00 00 00	- A. D. M. I. N. . . .
0019A228	48 00 6F 00	6E 00 67 00	20 00 4C 00	65 00 65 00	H. o. n. g. . . L. e. e.
0019A238	00 00 00 00	48 00 41 00	50 00 55 00	42 00 57 00	. . . H. A. P. U. B. W.
0019A248	53 00 00 00	45 00 6D 00	69 00 6C 00	79 00 00 00	S. . . E. m. i. l. y. . .
0019A258	43 00 75 00	72 00 72 00	65 00 6E 00	24 00 55 00	C. u. r. r. e. n. t. U.
0019A268	73 00 65 00	72 00 00 00	B4 A2 19 00	B4 A2 19 00	s. e. r. . . . C. . .
0019A278	B4 A2 19 00	B4 A2 19 00	B4 A2 19 00	B4 A2 19 00	C. . . . C. . . .

Figure 11 Post-analysis username blacklist

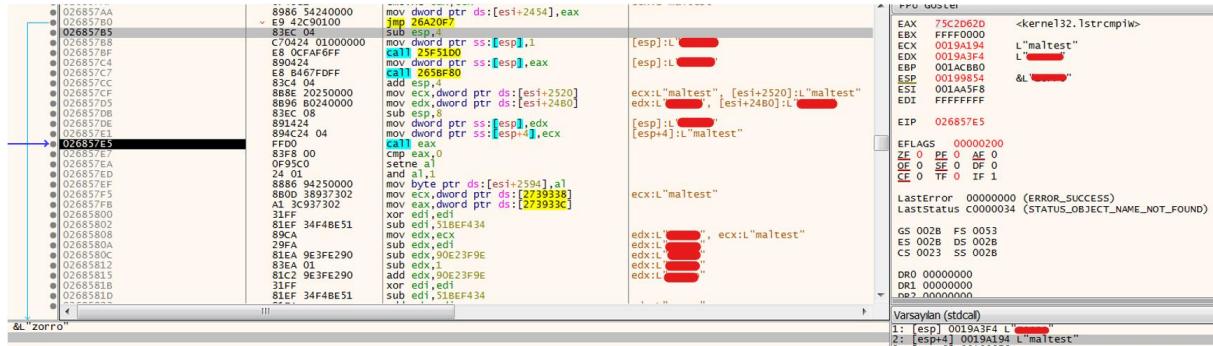


Figure 12 The process of comparing the user names in the list

The malware was also found to analyse the username list and compare it with the username of the computer it was found on. Compared usernames:

CurrentUser	sandbox	Emily	HAPUBWS
Hong Lee	IT-ADMIN	Johnson	Miller
TEQUILABOOMBOOM	milozs	Peter Wilson	sand box
malware	maltest	test user	virus
Andy	snort	virusclone	wilbert
virusClone	John Doe	timmy	

Table 4 Checked user names

It was detected that the malware executes malicious code with the ProcessHollowing technique in the executable file "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe".

InstallUtil.exe Analysis

General Review

SHA256	6052F7D7832F6EDDF1BA8309F189FCCCB9917128D216FC1C181327B3DEBDEDAC
Name	InstallUtil.exe
File Type	PE32-EXE

Table 5 InstallUtil.exe file information

Dynamic Analysis

Getting API Function Address

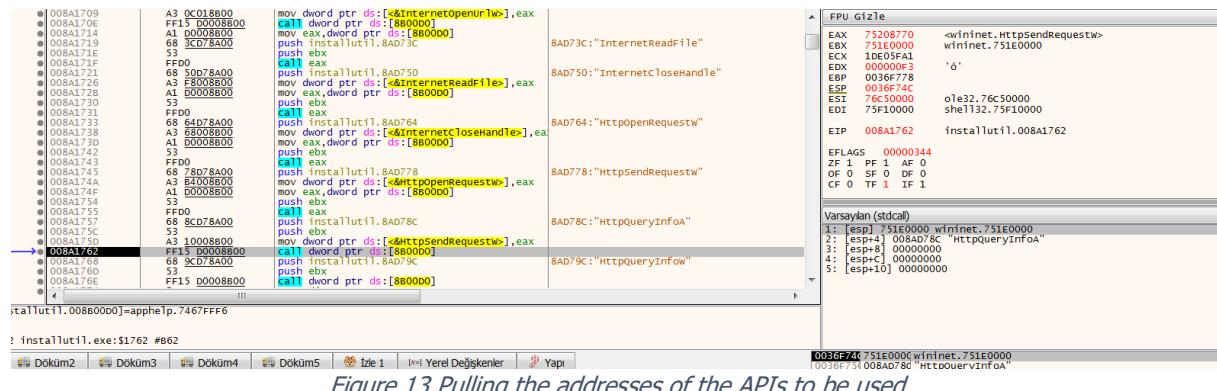


Figure 13 Pulling the addresses of the APIs to be used

It has been determined that it has received the addresses of the API Functions it will use. The functions whose addresses he received are as follows:

GetFileSize	GetDriveType	GetFileSize	GetDriveType
GetModuleFileNameW	GetSystemInfo	GetModuleFileNameW	GetSystemInfo
wideCharToMultiByte	ShellExecuteW	wideCharToMultiByte	ShellExecuteW
PathMatchSpecW	InternetReadFile	PathMatchSpecW	InternetReadFile
HttpSendRequestW	HttpQueryInfoA	HttpSendRequestW	HttpQueryInfoA

Table 6 APIs whose addresses were retrieved

ECHO



Figure 14 Mutex Creation

It was also detected that malware created a mutex named "**264782971_qJ5tS2bD5fD1nZ5kD2kV**".

String Resolving Algorithm

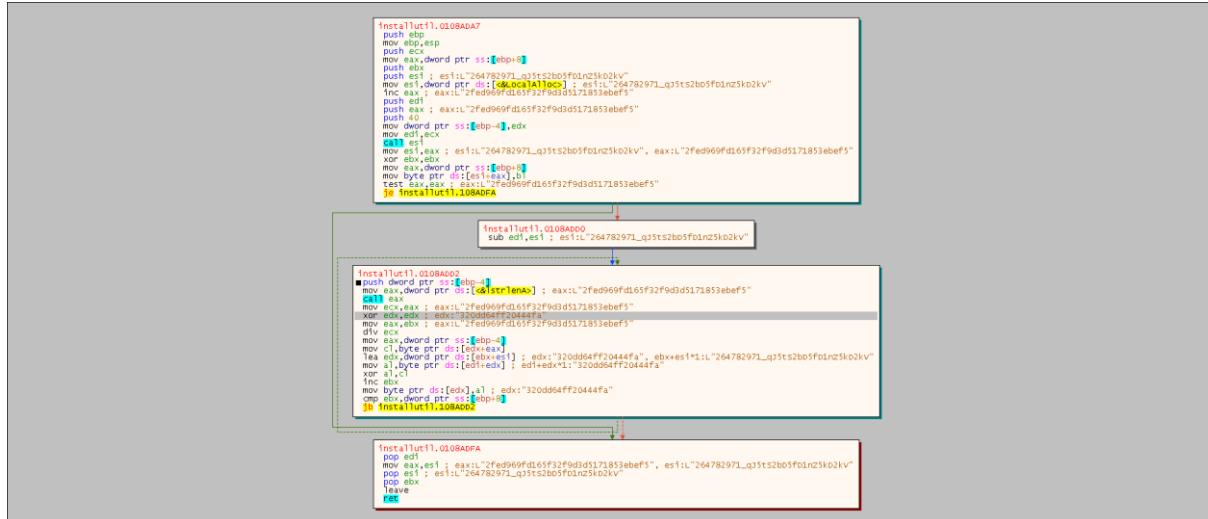


Figure 15 Algorithm for analysing String expressions

Process Access Detection



Figure 16 Access token information of the current process

ECHO

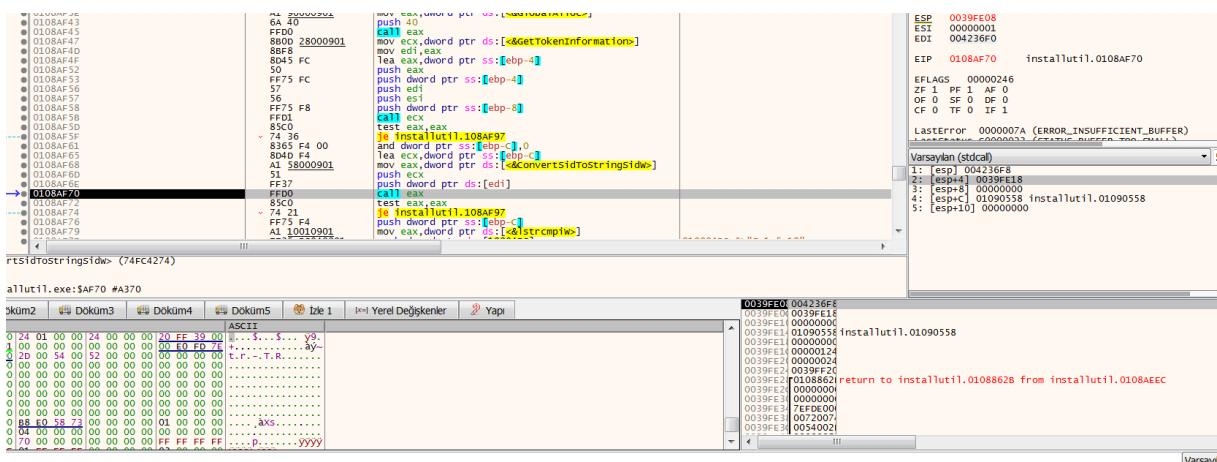


Figure 17 Retrieval of SID information for comparison

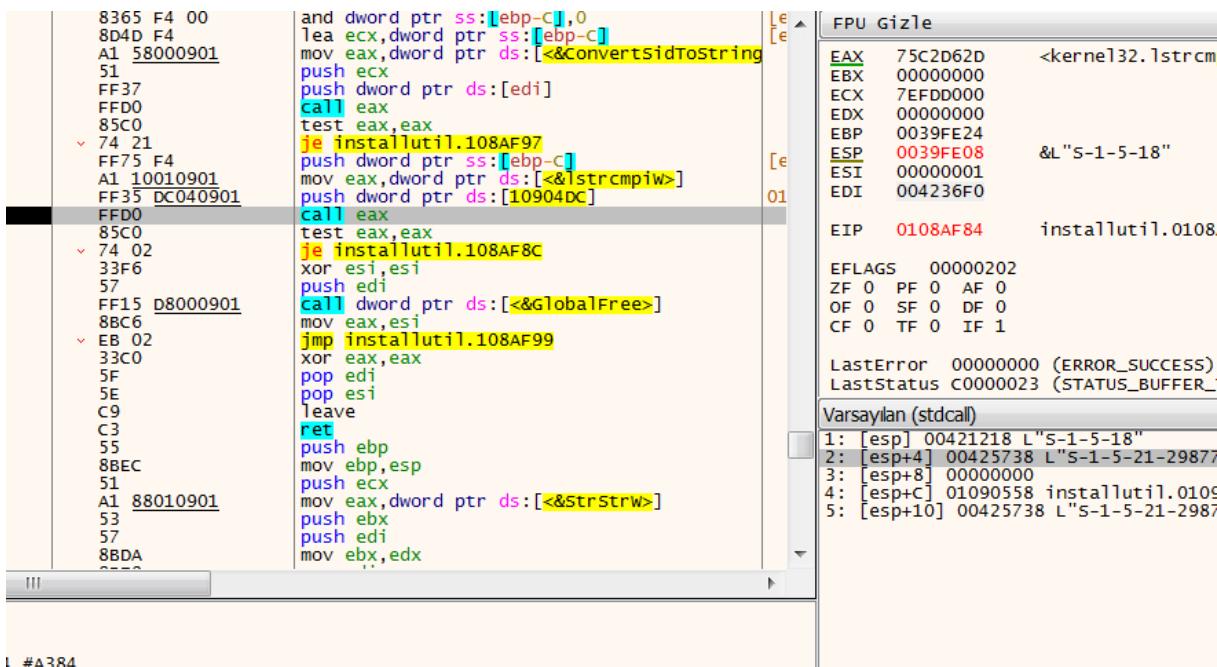


Figure 18 Admin authorisation comparison with SID information

It has been determined that the malware checks whether it has Admin authorisation. If it does not have Admin authorisation, it copies the Access Token of explorer.exe and restarts itself.

The Duplication Algorithm of Access Token of explorer.exe

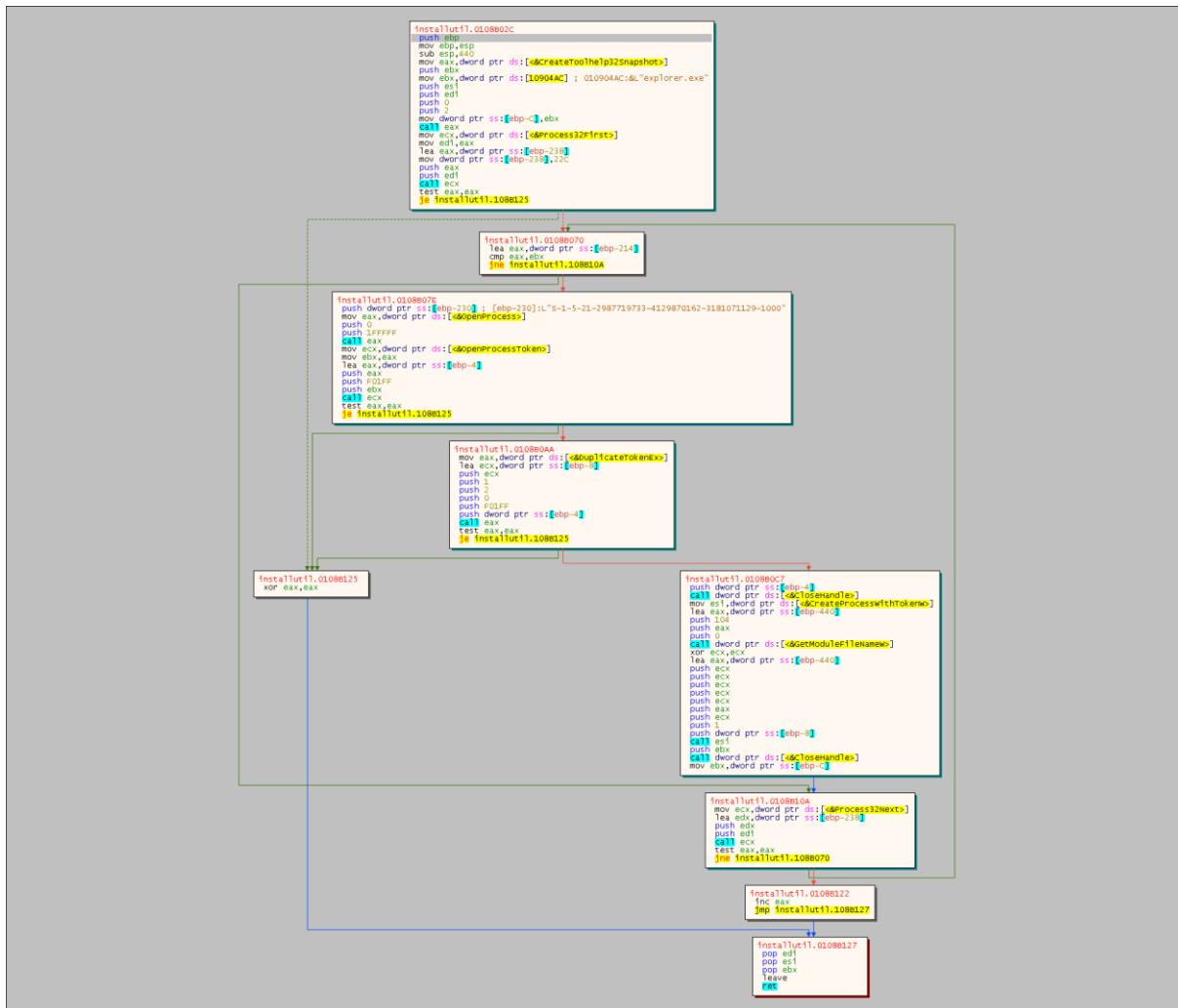


Figure 19 Algorithm to be applied if the process detects that it does not have admin authorisation

Creation of Request Contents



Figure 20 Obtaining Machine GuID information

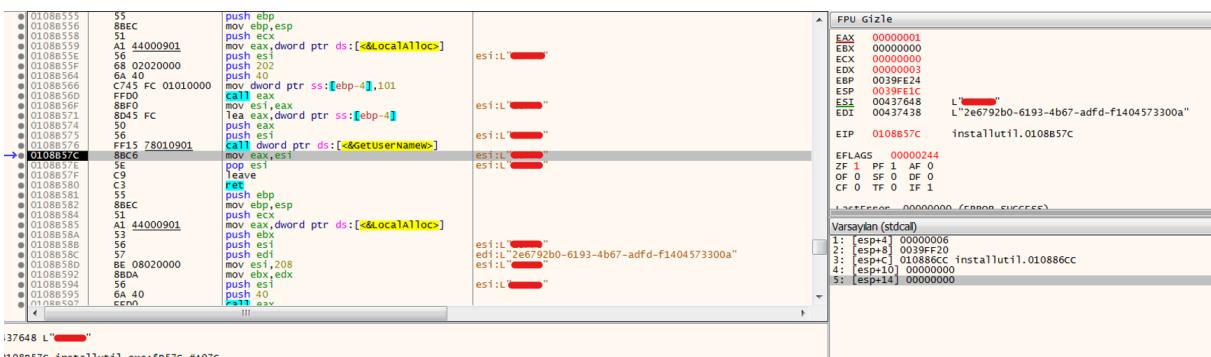


Figure 21 Pulling username information to generate victim ID information

It has been detected that it tries to create a uniq machine ID with MachineGuID and username. The malware will use this machineID and the configID variables it analyses in the http request it will send in the future. The created machineID is as follows:

machineId= <MachineGuID> |<username>&configId=2fed969fd165f32f9d3d5171853ebef5

Network Analysis

Request Analysis

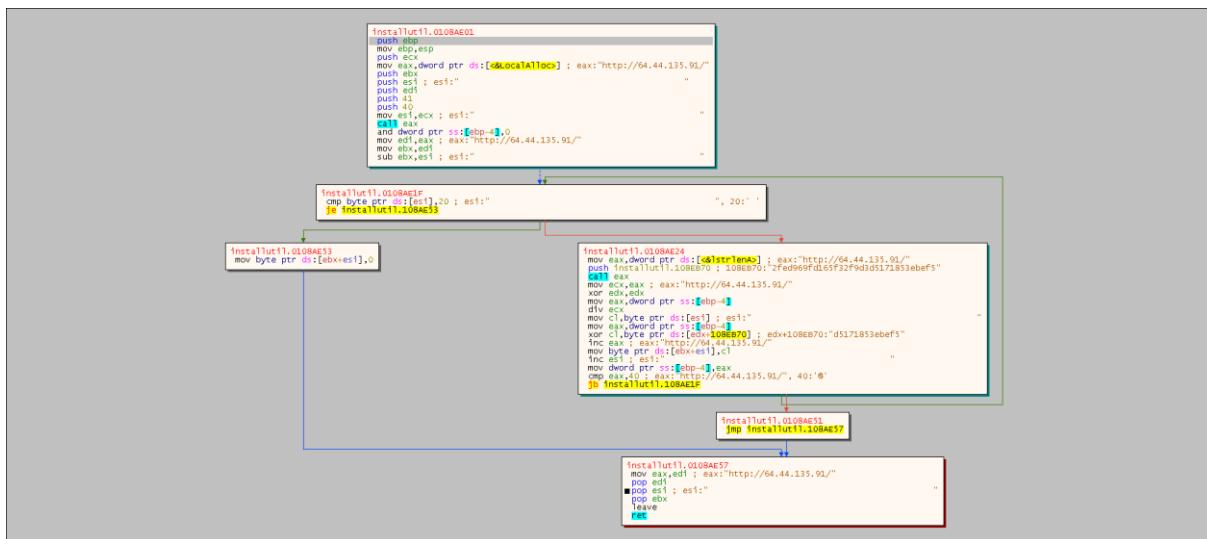


Figure 22 IP resolving

As a result of the analysis, it was determined that the IP "<http://64.144.135.91>/" appeared.

Request İçeriği

The screenshot shows a debugger interface with two panes. The left pane displays assembly code for a exploit development process, with several instructions highlighted in yellow. The right pane shows the CPU register dump, specifically the EIP register which contains the address 01088B9A, corresponding to the highlighted instruction.

Address	Instruction
01088B6A	mov eax,dword ptr ds:[<&wideCharToMultiByte>]
01088B6F	xor edx,edx
01088B71	push edx
01088B72	push edx
01088B73	push esi
01088B74	push ecx
01088B75	push FFFFFFFF
01088B77	push dword ptr ss:[ebp+8]
01088B7A	push edx
01088B7B	push FDE9
01088B80	call eax
01088B82	test eax,eax
01088B84	je installutil.1088CFF
01088B8A	mov eax,dword ptr ds:[<&InternetOpenW>]
01088B8F	xor ecx,ecx
01088B91	push ecx
01088B92	push ecx
01088B93	push ecx
01088B94	push ecx
01088B95	push installutil.108ECFC
01088B9A	call eax
01088B9C	mov esi,esi
01088B9E	mov dword ptr ss:[ebp-14],esi
01088B9F	test esi,esi
01088BA1	je installutil.1088C7D
01088BA3	push 1
01088BA9	xor eax,eax
01088BAB	mov ecx,dword ptr ds:[<&InternetConnectW>]
01088BAD	push eax
01088BB3	push 3
01088BB4	6A 03

Figure 23 Detection of Agent information used in C2 communication

ECHO

Figure 24 shows the assembly code for the installutil.exe process. The code is comparing memory at [ebp-C] with 0x00000000. If it's not zero, it calls ECX (which is 0x004259F0). The assembly code includes instructions like push eax, mov edx, cmp word ptr, and call ecx.

Figure 24

Figure 25 shows the assembly code for net.HttpOpenRequestW. The code is comparing memory at [ebp-C] with 0x00000000. If it's not zero, it calls ECX (which is 0x75205AB0). The assembly code includes instructions like push eax, mov edx, cmp word ptr, and call ecx.

Figure 25 Determining the method of the request to be sent

Figure 26 shows the assembly code for installutil.exe. The code is comparing memory at [ebp-C] with 0x00000000. If it's not zero, it calls ECX (which is 0x75208770). The assembly code includes instructions like push eax, mov edx, cmp word ptr, and call ecx.

Figure 26

Figure 27 shows the assembly code for installutil.exe. The code is comparing memory at [ebp-C] with 0x00000000. If it's not zero, it calls ECX (which is 0x75208770). The assembly code includes instructions like push eax, mov edx, cmp word ptr, and call ecx.

Figure 27 Sending the edited request to the C2 server

It has been determined that the agent information is "**TakeMyPainBack**" and the Request method is POST.

Http request content:

- Content-Type: application/x-www-form-urlencoded; charset=utf-8\r\n\r\n\r\n\r\n\r\n
- machineId=2e6792b0-6193-4b67-adfd-f1404573300a|zorro&configId=2fed969fd165f32f9d3d5171853ebef5

Expected Variables in Response

token
wlts_
grbr_
tlgrm_
dr_

Table 7

When the behaviour of malware after the response was examined, it was determined that it tried to use some data from the responses. The detected data variables are shown in Table 7.

After Response

Gathering Device Information

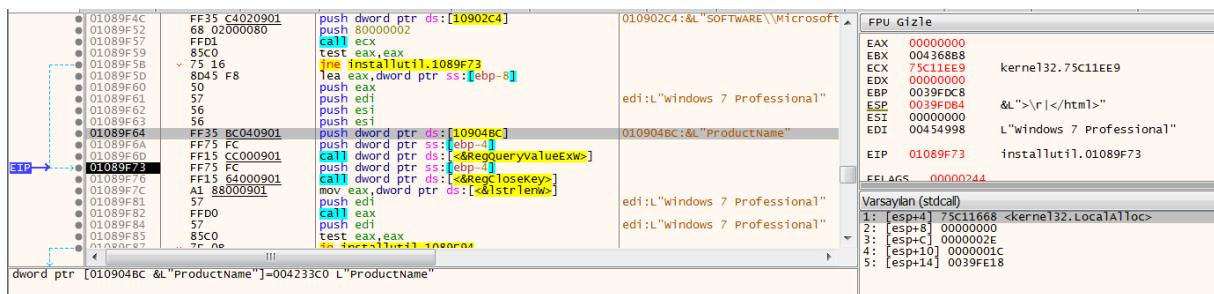


Figure 28 Detection of collecting information through the Registry

It has been found that the malware collects some information from the victim device. This information includes:

Locale	Time Zone	OS	Architecture	System Informations
Memory Information	Display Size	Display Devices	Application Information	

Table 1 3 Registry üzerinden aldığı bazı bilgiler

It has been found that malware collects this information in the following format:

- Locale: %s \n\t- Time zone: %c%ld minutes from GMT \n\t- OS: %s\n\t- CPU: %s (%d cores)) - \n\t- Architecture: x%d\n\t- RAM: %d MB \n\t- Display size: %dx%d\n\t- Display Devices: %s

It generates a random file name for the generated collection of information and is transmitted to the server with the POST method along with the file name.

Request içeriği: "Content-Type: multipart/form-data; boundary=bF0xB2TEnUcQ7DfR\r\n\r\n\r\n\r\n"

bF0xB2TEnUcQ7DfR = <verilerin toplandığı dosyanın adı>

Figure 29 Sending the collected information back to the C2 server with the token information returned by the server

It was found that the token information received from the server was also used in sending requests to the server.

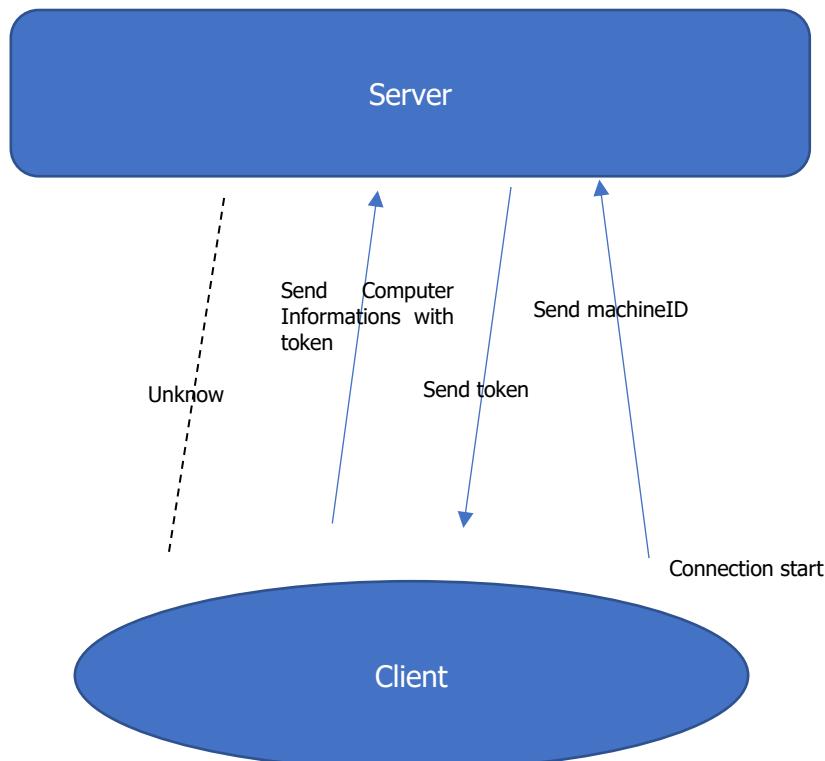


Figure 30 C2 Server and victim device communication

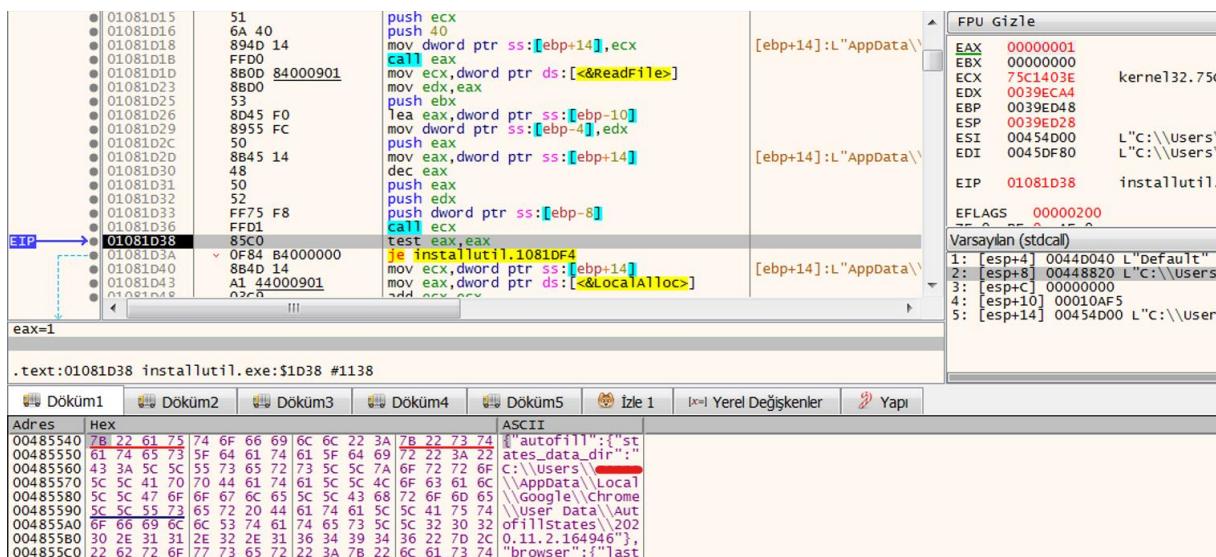


Figure 31



Figure 32 Detection of encrypted_key information encrypted with base64 algorithm and used as byte

It was found that the malware extracted the information in the "**AppData\Local\Chrome\User Data\Default\Login**" file and the "**AppData\Local\Google\Chrome\User Data\AutofillStates**" folder.

It was observed that encrypted_key data was encrypted with base64 encryption algorithm.

ECHO

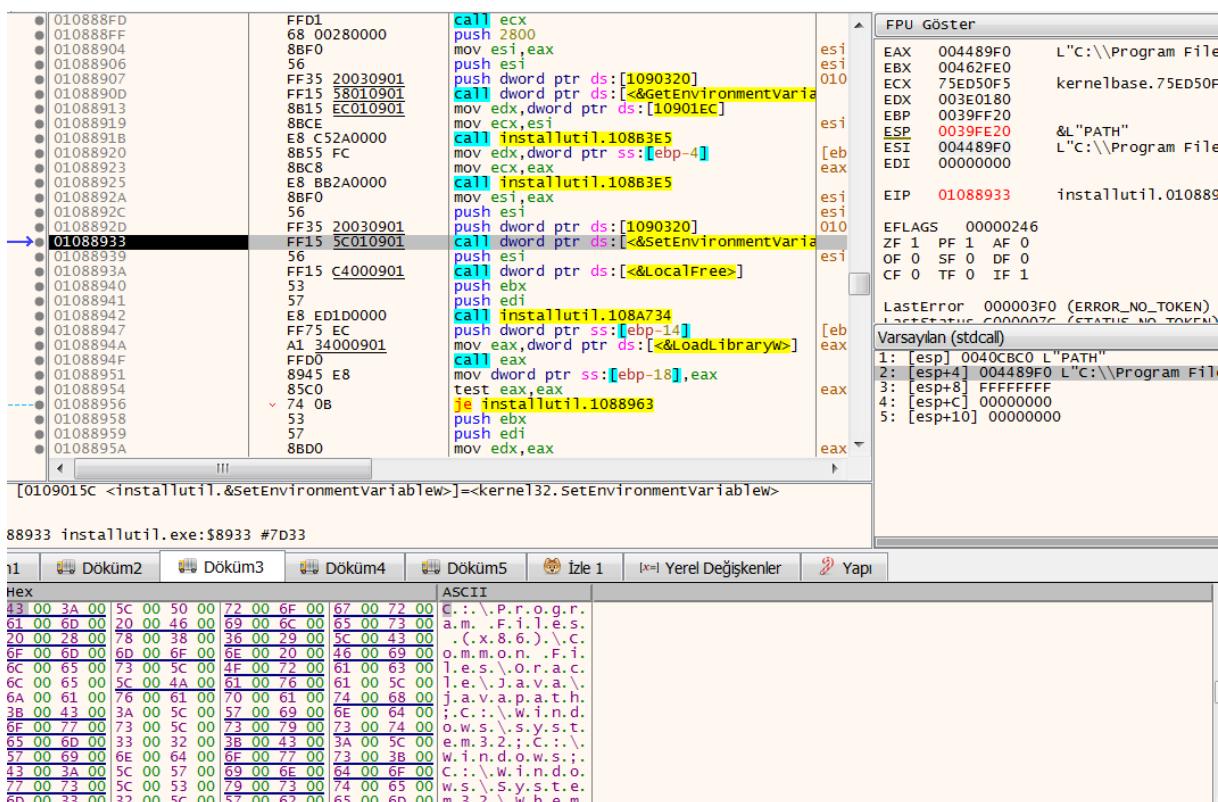


Figure 33 Change detection on environment variables

It was determined that malware made additions on the PATH variable among the environment variables.

```
C:\\Program Files (x86)\\Common
Files\\Oracle\\Java\\javapath;C:\\Wi
ndows\\system32;C:\\Windows;C:\\
Windows\\System32\\Wbem;C:\\Win
dows\\System32\\WindowsPowerShe
ll\\v1.0\\;C:\\Users\\<user>
\\AppData\\Local\\Programs\\Python
\\Python37\\Scripts\\;C:\\Users\\<us
er>\\AppData\\Local\\Programs\\Pyt
hon\\Python37\\;C:\\Users\\<user>\\
AppData\\LocalLow
```

ECHO

DLL Loading

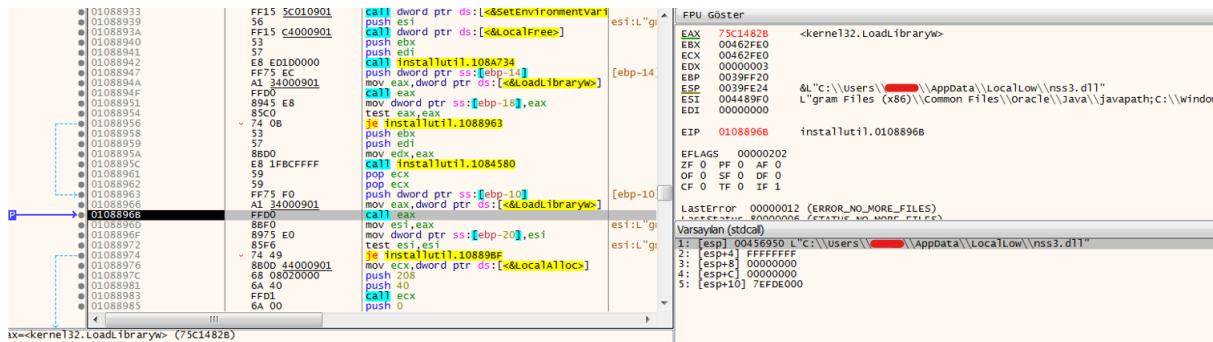


Figure 34 nss3.dll installation process

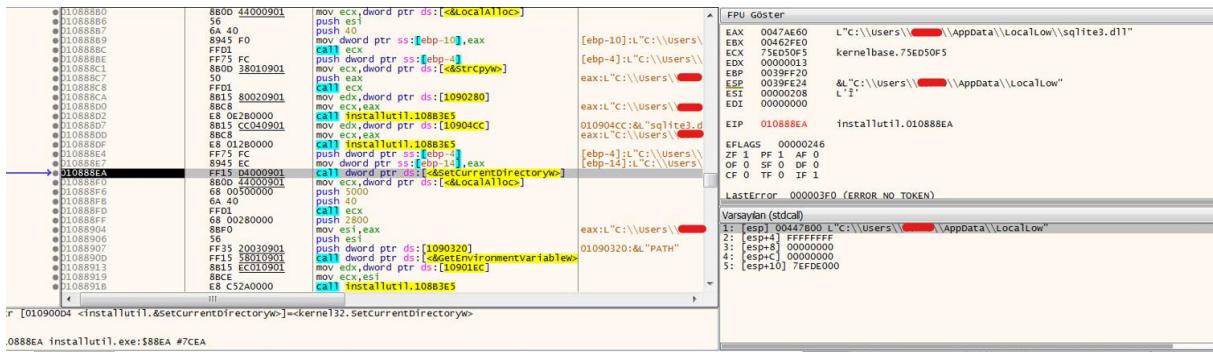


Figure 35 sqlite3.dll installation process

It has been detected that the malware installs sqlite3.dll and nss3.dll created in the LocalLow directory.

Database Operations

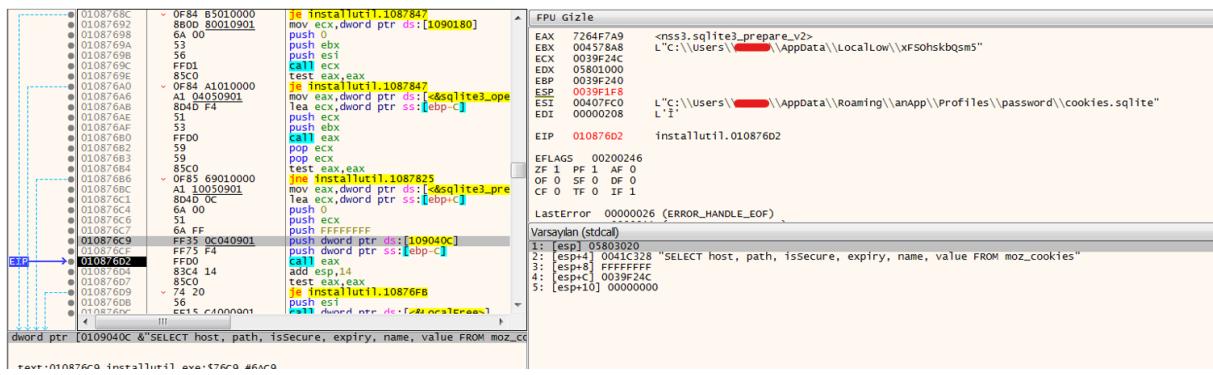


Figure 36

It has been detected that the malware creates a database file with a random name in the AppData\LocalLow\ directory. It has been determined that it pulls data from the following files with SQL queries to the created database.

coocies.sqlite
formhistory.sqlite
password.txt
storage\default
wallet.dat
logins.json

Table 8 Some targeted file and directory names

File Traversal Algorithm

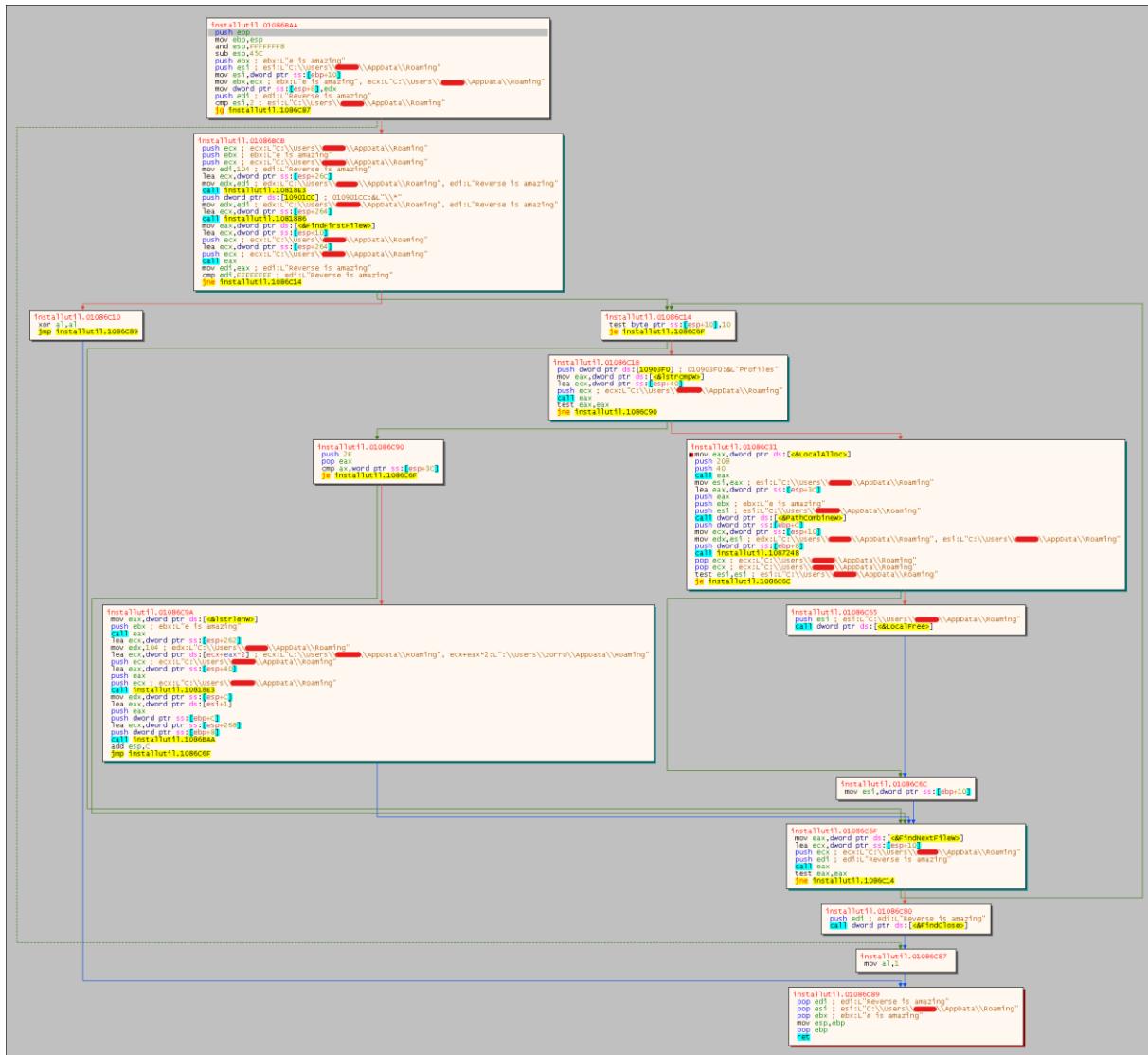


Figure 37 Index scanning algorithm

It was observed that it scanned the directory until it found the Profiles and User Data folder.

ECHO

Additional Analysis

It was found that malware takes a ScreenShot and saves it as a recording.

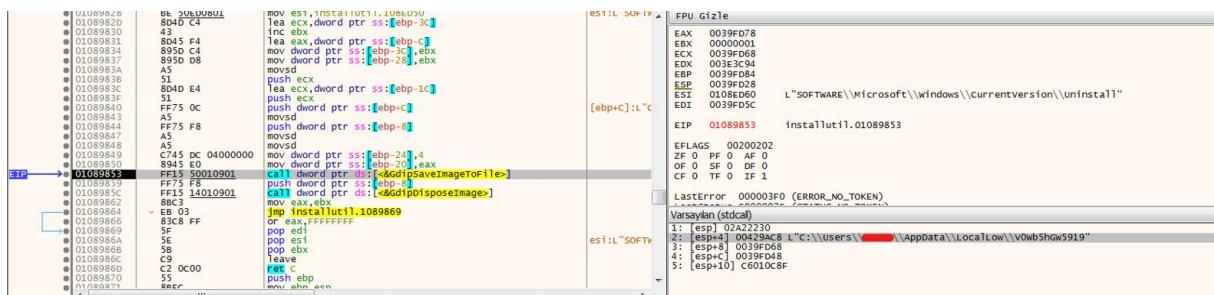


Figure 38 Saving the screen photo taken by Malware

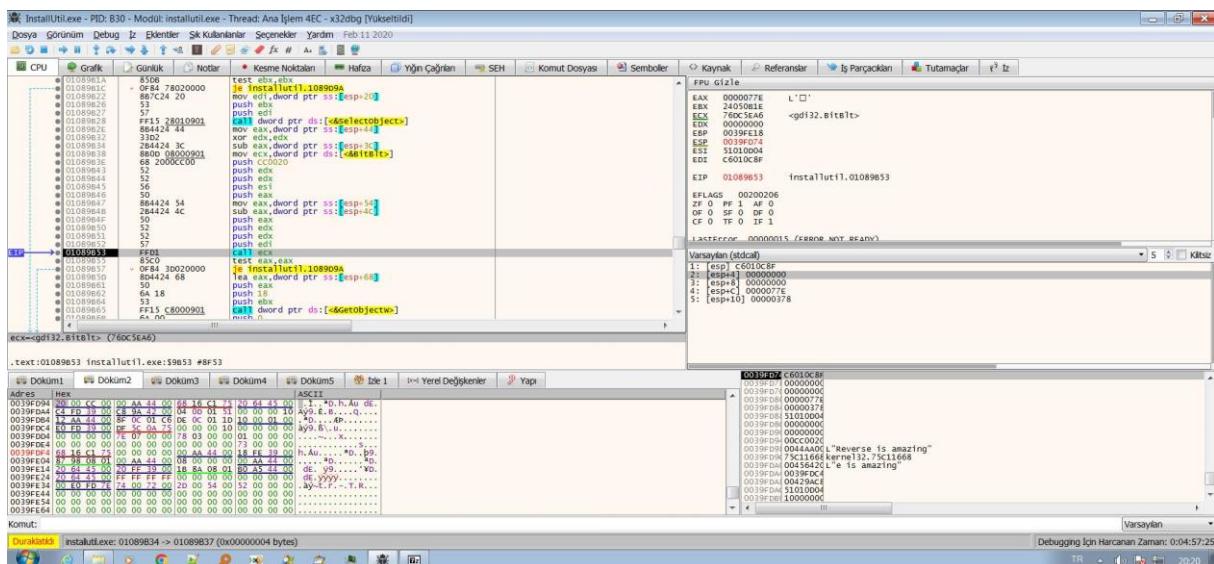


Figure 39 Screen Photo taken by Malware

SQL Queries

SELECT origin_url, username_value, password_value FROM logins
SELECT origin_url, username_value, password_value FROM logins
SELECT host_key, path, is_secure , expires_utc, name, encrypted_value FROM cookies
SELECT name, value FROM autofill
SELECT host, path, isSecure, expiry, name, value FROM moz_cookies
SELECT fieldname, value FROM moz_formhistory
SELECT name_on_card, card_number_encrypted, expiration_month, expiration_year FROM credit_cards

Table 9 Some of the enquiries made

YARA Rule

```

rule Rule_InstallUtil
{
meta:
    author = "Bilal BAKARTEPE (EchoCTI Team)"
    site = "https://github.com/bixploit"
    description = "RaccoonStealler v2.0 second stage PE file"
    hash= "d69ee30203430d1404a2890268bb04e9"
strings:
    $sql1 = "SELECT origin_url, username_value, password_value FROM logins"
    $sql2 = "SELECT host_key, path, is_secure , expires_utc, name, encrypted_value FROM cookies"
    $sql3 = "SELECT name, value FROM autofill"
    $sql4 = "SELECT host, path, isSecure, expiry, name, value FROM moz_cookies"
    $sql5 = "SELECT fieldname, value FROM moz_formhistory"
    $sql6 = "SELECT name_on_card, card_number_encrypted, expiration_month, expiration_year FROM credit_cards"

    $dir_name1 = "profiles"
    $dir_name2 = "712006f6e7da2882" //User Data
    $dir_name3 = "Default"
    $dir_name4 = "Login Data"
    $file_name1= "password.txt"
    $file_name2= "cookies.sqlite"
    $file_name3= "Cookies"

    $agent="TakeMyPainBack"

    $ip_clear="http://64.44.135.91"
    $ip_enc="d5171853ebef5"

    $enc_str1="aa0bb6f89e4fc28e"
    $enc_str2="ba0c5f9d6a984fdd" //encrypted_values
    $enc_str3="587a51bde849292f"
    $enc_str4="ca82e1c9d5793376"
    $configurationID="2fed969fd165f32f9d3d5171853ebef5"
    $mutex_name="264782971_qJ5tS2bD5fD1nZ5kD2kV"

    $respons_variable1="320dd64ff20444fa" //tlgrm
    $respons_variable2="d286b66a2753e1b1" //wlts
    $respons_variable3="bee04f3449ba713e" //sstmnfo
    $respons_variable4="6ef9561122a8649a" //token
    $respons_variable5="877e12dc4d066d8c" //nss3.dll
    $respons_variable6="274f2fd9bfa77a7c" //sqlite.dll

    $opc1 = {53 56 57 6A 41 6A 40 8B F1 FF D0 83 65 FC 00 8B F8 8B DF 2B DE 80 3E 20 74 2F A1
             94 01 41 00 68 70 EB 40 00 FF D0 8B C8 33 D2 8B 45 FC F7 F1 8A 0E 8B 45 FC 32 8A 70 EB 40
             00 40 88 0C 33 46 89 45 FC 83 F8 40 72 CE}// allocation and deobfuscation
    $opc2 = {55 8B EC 51 53 56 57 8B 3D 88 00 41 00 8B DA 53 89 4D FC FF D7 FF 75 FC 8B F0 FF D7
             8B 0D 44 00 41 00 8D B8 80 00 00 00 03 FE 8D
             04 3F 50 6A 40 FF D1 FF 75 FC 8B F0 8B D7 8B CE E8 34 64 FF FF 53 8B D7 8B CE E8 57 64 FF
             FF FF 75 FC FF 15 D8 00 41 00 5F 8B C6 5E 5B C9 C3}//deobfuscation and ascii to unicode transition

condition:
    (any of ($opc*)) and (2 of ($sql*, $dir_name*, $file_name*, $enc_str*, $respons_variable*) or any
    of ($ip_clear, $ip_enc, $agent, $configurationID, $mutex_name))
}

```

MITRE ATTACK TABLE

Reconnaissance	Execution	Discovery	Collection	Defense Evasion	Credential Access	Command and Control	Exfiltration
T1592 Gather Victim Host Information: <u>Hardware</u>	T1559 Inter-Process Communication: <u>Component Object Model</u>	T1012 Query Registry	T1005 Data from Local System	T1070 Indicator Removal on Host: File Deletion	T1539 Steal Web Session Cookie	T1071 Application Layer Protocol: <u>Web Protocols</u>	T1041 Exfiltration Over C2 Channel
T1589 Gather Victim Identity Information: <u>Credentials</u>		T1082 System Information Discovery	T1113 Screen Capture	T1140 Deobfuscate/Decode Files or Information		T1105 Ingress Tool Transfer	T1020 Automated Exfiltration
T1592 Gather Victim Host Information: <u>Software</u>		T1614 System Location Discovery: <u>System Language Discovery</u>					

Mitigations

1. Emails and senders should be carefully checked to ensure they are genuine before opening any attachments.
2. Do not download any resources from unsafe websites.
3. Use a reliable, high quality and always updated antivirus software.
4. Keep your operating system and applications up to date with the latest security patches.
5. End user training is vital for your organisation, make sure you inform your employees about the dos and don'ts of online security.



ECHO

CYBER THREAT INTELLIGENCE