Revisiting Heaven's Gate with Lumma Stealer

Prepared by: Vlad Pasca, Senior Malware & Threat Analyst

SecurityScorecard

SecurityScorecard.com
info@securityscorecard.com

Tower 49
12 E 49th Street
Suite 15-001
New York, NY 10017
1.800.682.1707

Table of contents

Executive summary

Lumma is a stealer that has been sold on hacker forums since August 2022. The malware steals information from web browsers, cryptocurrency wallets, 2FA extensions, and applications such as AnyDesk, FileZilla, KeePass, Steam, and Telegram. The process also gathers data about the infected machine, such as the installed applications, the username and computer name, the current hardware profile, the system default language, the screen resolution, the RAM amount, and the processor name and type. The malware employs the Heaven's Gate technique that enables a 32-bit process to execute 64-bit code by performing a call using segment selector 0x33. The stolen information is exfiltrated to the C2 server using multiple HTTP POST requests.

Analysis and findings

SHA256: 199de8b727ceae96afb7c7560092c1d7a4dbe5a005c07ae20cffd9871da52b82

Using the "Detect It Easy" tool, we determined that the sample is packed with MPRESS:

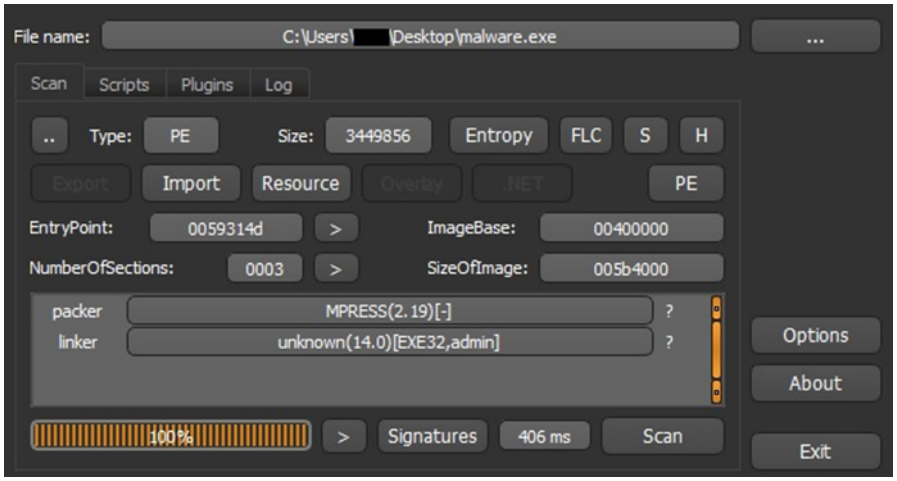

Figure 1

As we can see in figure 2, the code is decoded at runtime, and we need to find the point where to dump the unpacked executable:

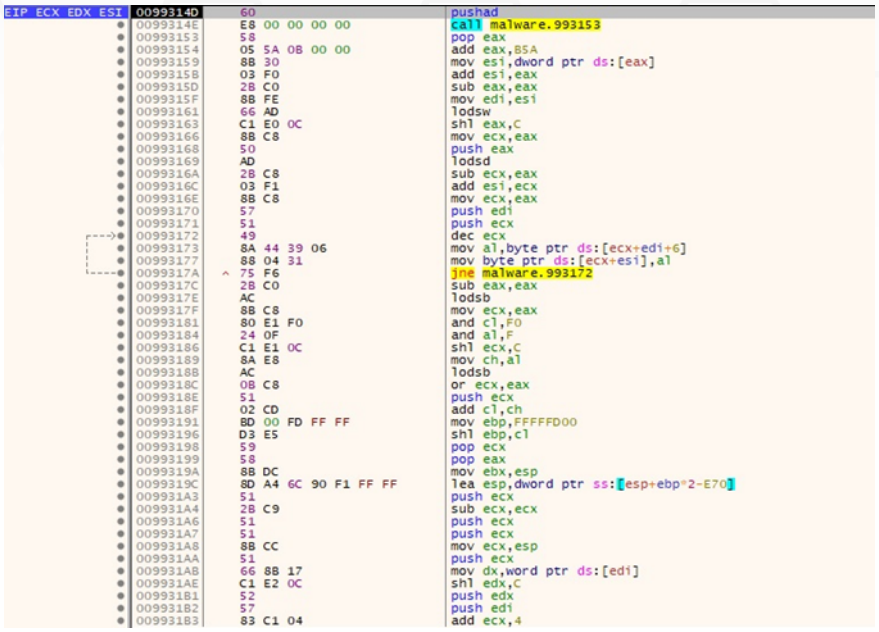

Figure 2

The malware implements the API hashing technique that resolves the necessary APIs based on 4-byte hashes:

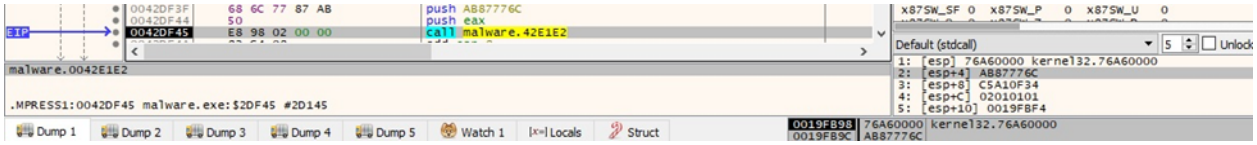

Figure 3

Figure 4

The ExpandEnvironmentStringsW API is utilized to obtain the path of the Google Chrome User Data directory, as shown in figure 5.

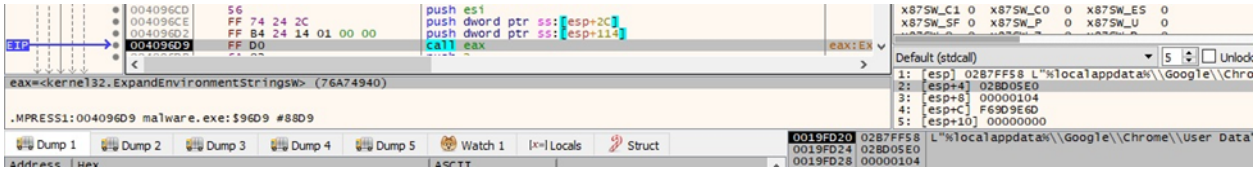

Figure 5

The malicious binary determines if the current process is running under WOW64 via a function call to IsWow64Process2:

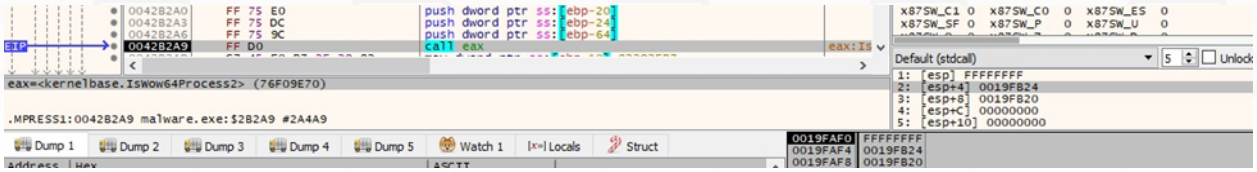

Figure 6

The malicious process opens the NTDLL.dll file in order to extract the syscall numbers that will be used at runtime (0x80000000 = **GENERIC_READ**, 0x1 = **FILE_SHARE_READ**, 0x3 = **OPEN_EXISTING**, 0x80 = **FILE_ATTRIBUTE_NORMAL**):

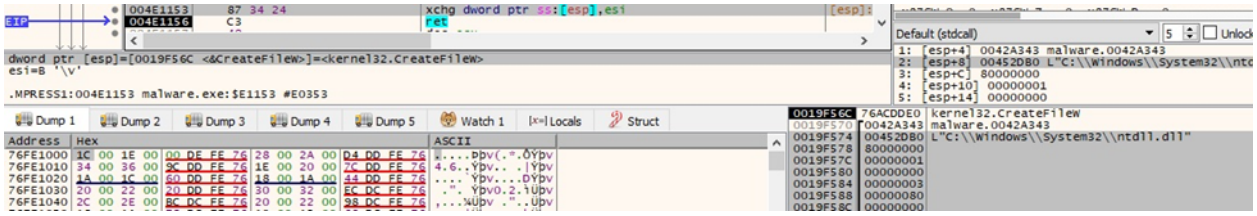

Figure 7

It retrieves the size of the specified DLL using GetFileSize:

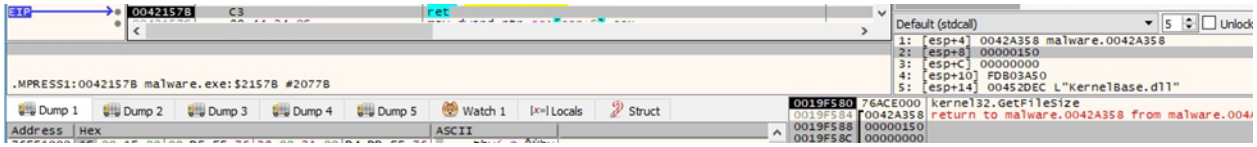

Figure 8

The ReadFile method is used to read data from the DLL:

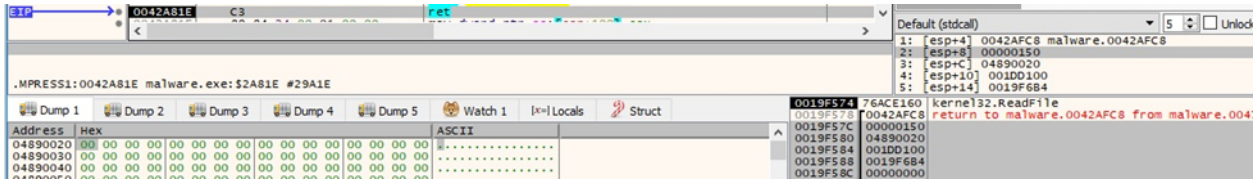

Figure 9

The malicious activity is implemented using the Heaven's Gate technique. The segment selector 0x33 is utilized to transition to x64 mode and execute 64-bit code. As we can see below, the syscall 0x55 (NtCreateFile function) is used to open the Local State file:

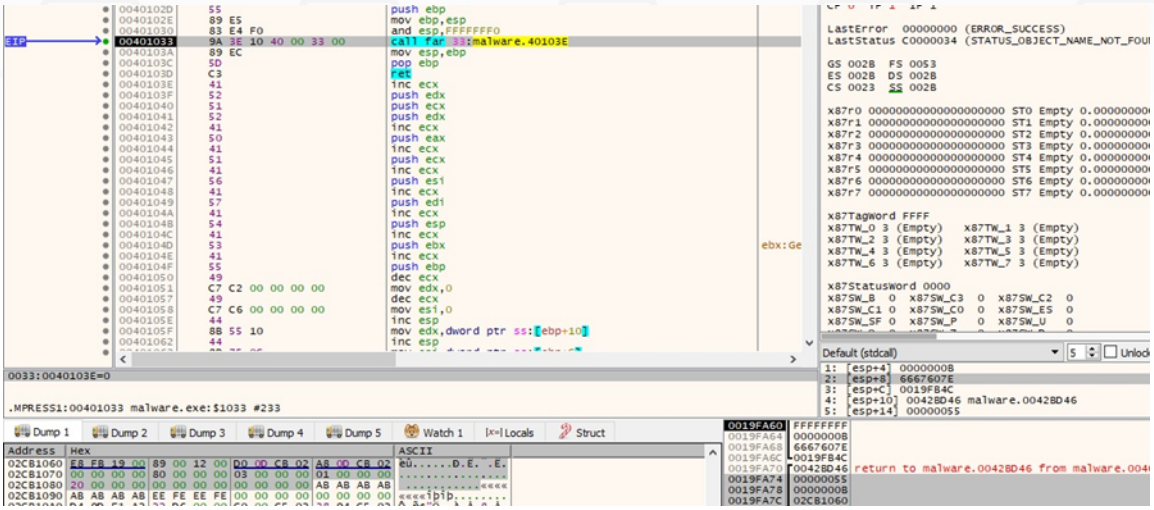

Figure 10

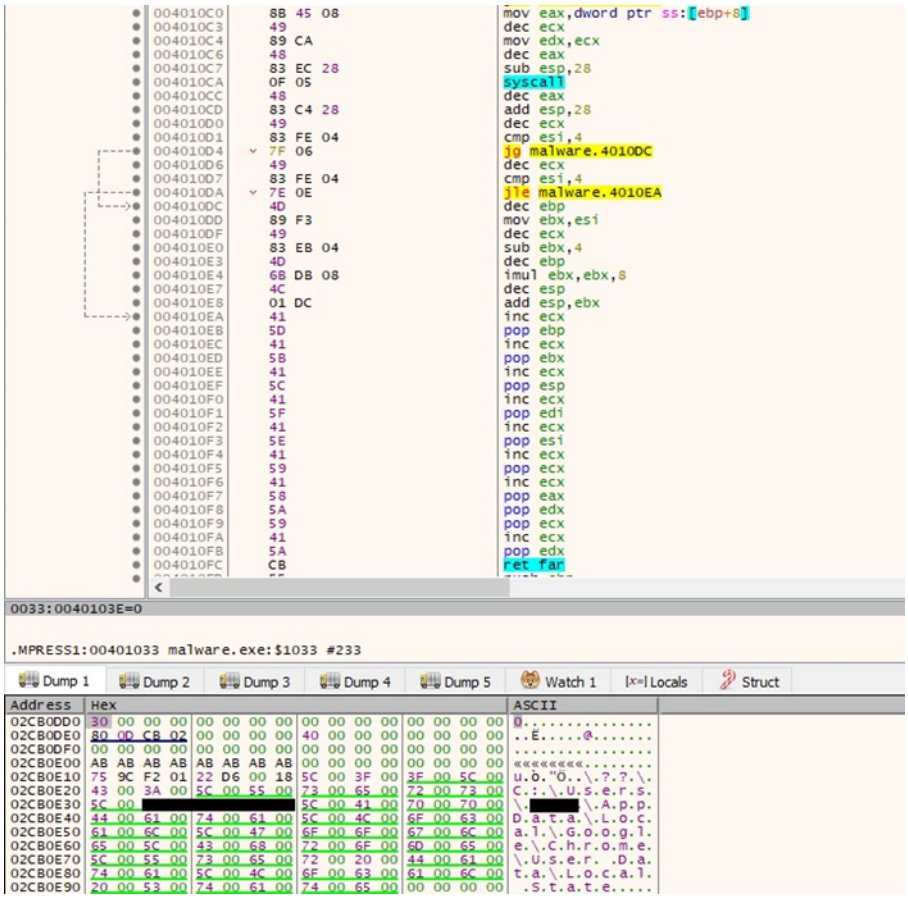

Figure 11

NtQueryInformationFile is used to obtain the above file size:

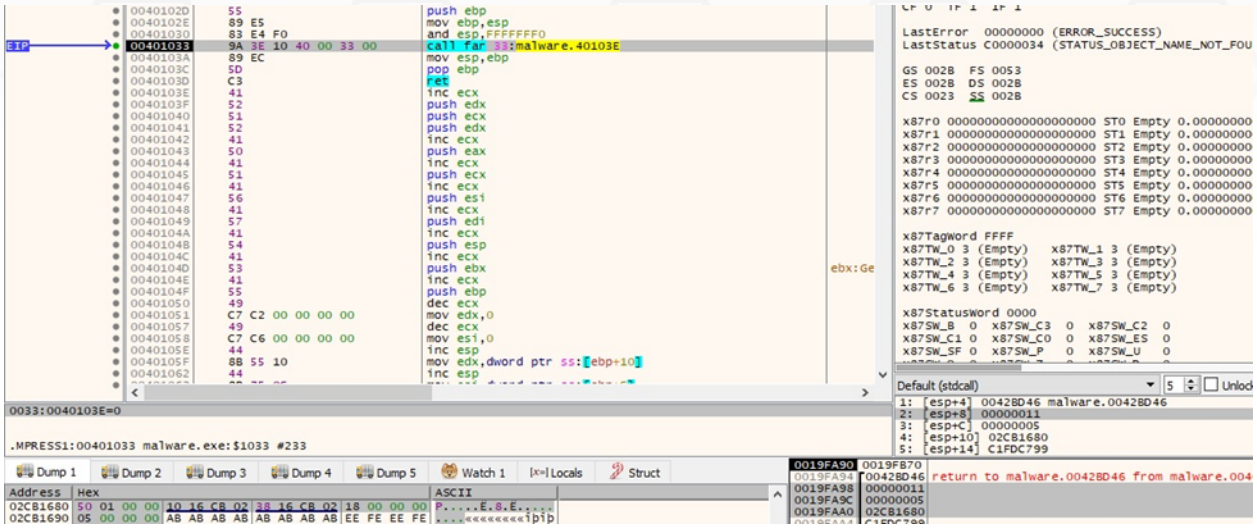

Figure 12

The malicious binary reads the file content by calling the NtReadFile method:

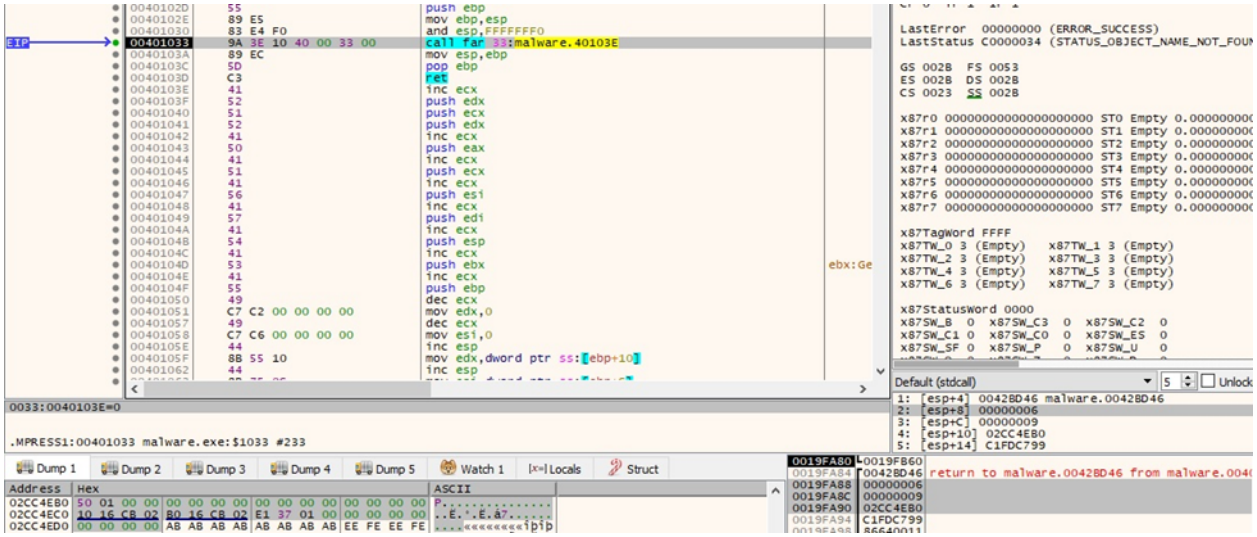

Figure 13

Finally, the file is closed via a function call to NtCloseFile:

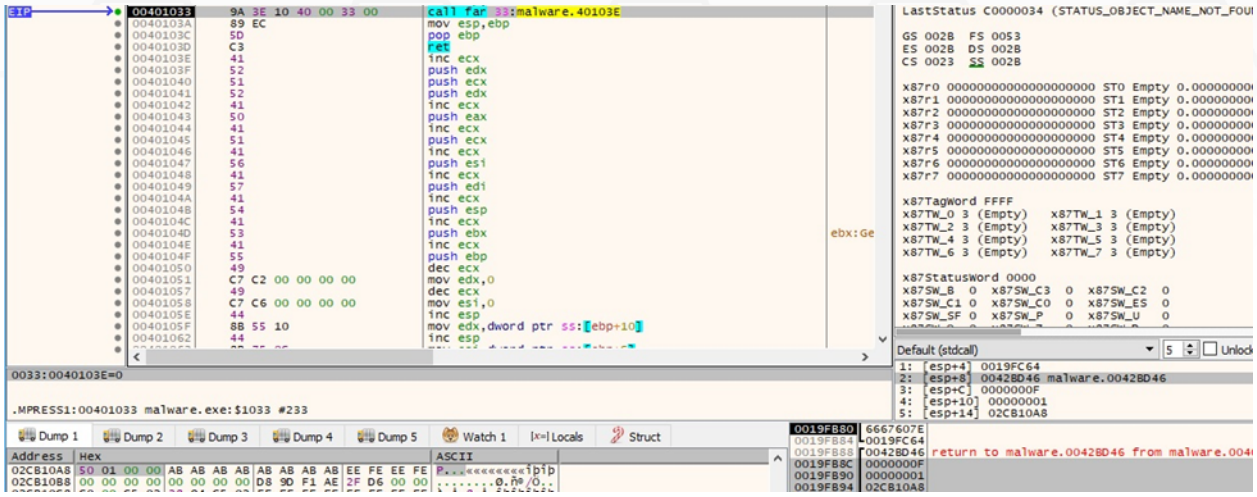

Figure 14

The “encrypted_key” extracted from the Local State file is Base64-decoded using the CryptStringToBinaryA API (0x1 = **CRYPT_STRING_BASE64**):

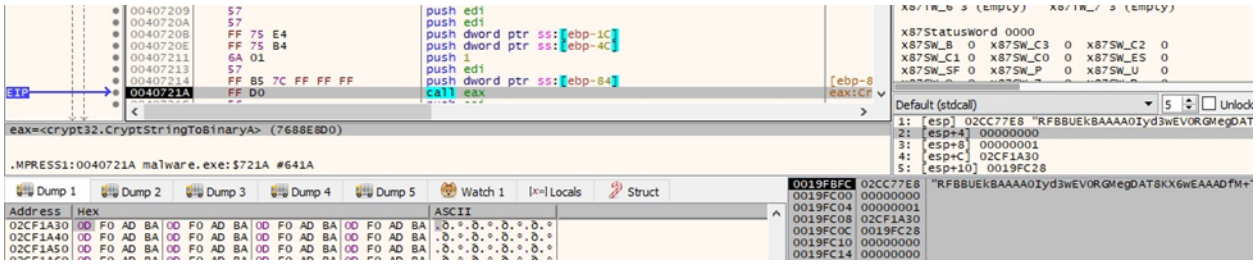

Figure 15

The binary decrypts the resulting buffer and obtains a 32-byte AES key using CryptUnprotectData:

Figure 16

Lumma Stealer targets multiple Browser wallets, which are located in the “Google\Chrome\User Data\Default\Local Extension Settings\” directory. An example is shown in figure 17, which is looking for MetaMask:

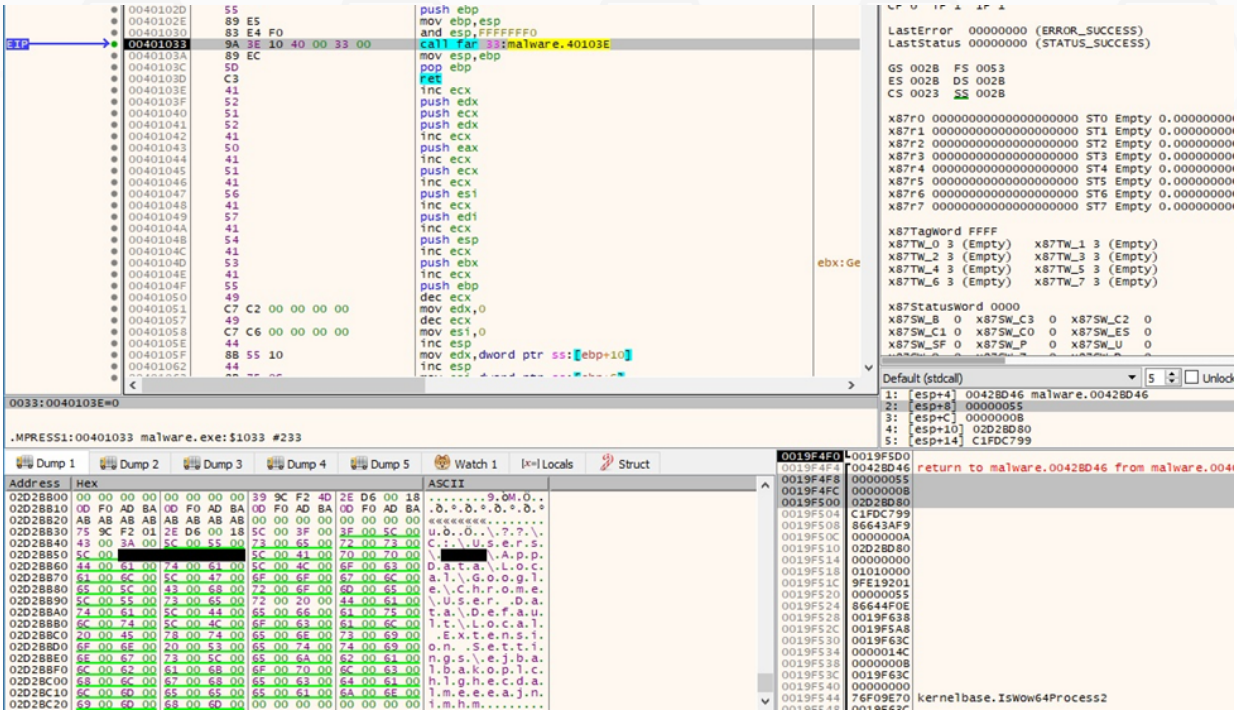

Figure 17

The following Browser wallets and 2FA applications are targeted:

- aeachknmefphepccionboohckonoeemg (**Coin98 Wallet**)

- afbcbjpbpfadlkmhmclhkeeodmamcflc (**Math Wallet**)

- aiifbnbfobpmeekipheeijimdpnlpgpp (**Station Wallet**)

- amkmjjmmflddogmhpjloimipbofnfjih (**Wombat**)

- bcopgchhojmggmffilplmbdicgaihlkp (**Hycon Lite Wallet**)

- bhghoamapcdpbohphigoooaddinpkbai (**Google Authenticator extension**)

- blnieiiffboillknjnepogjhkgnoapac (**EQUAL Wallet**)

- cihmoadaighcejopammfbmddcmdekcje (**Leaf Wallet**)

- cjelfplplebdjjenllpjcblmjkfcffne (**Jaxx Liberty**)

- cnmamaachppnkjgnildpdmkaakejnhae (**Auro Wallet**)

- cphhlgmgameodnhkjdmkpanlelnlohao (**NeoLine Wallet**)

- dkdedlpgdmmkkfjabffeganieamfklkm (**Cyano Wallet**)

- dmkamcknogkgcdfhhbddcghachkejeap (**Keplr Wallet**)

- ejbalbakoplchlghecdalmeeeajnimhm (**MetaMask**)

- ffnbelfdoeiohenkjibnmadjiehjhajb (**Yoroi Wallet**)

- fhbohimaelbohpjbbldcngcnapndodjp (**Binance Wallet**)

- fhmfendgdocmcbmfikdcogofphimnkno (**Sollet**)
- fihkakfobkmkjojpchpfgcmhfjnmnfpi (**BitApp Wallet**)
- flpiciilemghbmfalicajoolhkkenfel (**ICONex**)
- fnjhmkhhmkbjkkabndcnnogagogbneec (**Ronin Wallet**)
- gaedmjdfmmahhbjefcbgaolhhanlaolb (**Authy 2FA Authentication**)
- hcflpincpppdclinealmandijcmnkbgn (**KHC Wallet**)
- hnfanknocfeofbddgcijnmhnfnkdnaad (**Coinbase Wallet**)
- hpglfhgfnhbgpjdenjgmdgoeiappafln (**Guarda**)
- ibnejdfjmmkpcnlpebklmnkoeoihofec (**TronLink**)
- ijmpgkjfkbfhoebgogflfebnmejmfbml (**BitClip**)
- ilgcnhelpchnceeipipijaljkblbcobl (**GAuth Authenticator**)
- imloifkgjagghnncjkhggdhalmcnfklk (**Trezor Password Manager**)
- infeboajgfhgbjpjbeppbkgnabfdkdaf (**OneKey Legacy**)
- jbdaocneiiinmjbjlgalhcelgbejmnid (**Nifty Wallet**)
- jojhfeoedkpkglbfimdfabpdfjaoolaf (**Polymesh Wallet**)
- kkpllkodjeloidieedojogacfhpaihoh (**Enkrypt**)
- klnaejjgbibmhlephnhpmaofohgkpgkd (**ZilPay**)
- kncchdigobghenbbaddojjnnaogfppfj (**iWallet**)
- kpfopkelmapcoipemfendmdcghnegimn (**Liquality Wallet**)
- lkcjlnjfpbikmcmbachjpdbijejflpcm (**Steem Keychain**)
- lodccjjbdhfakaekdiahmedfbieldgik (**DappPlay**)
- mnfifefkajgofkcjkemidiaecocnkjeh (**TezBox**)
- nanjmdknhkinifnkgdcggcfnhdaammmj (**GuildWallet**)
- nhnkbkgjikgcigadomkphalanndcapjk (**CLV Wallet**)
- nkbihfbeogaeaoehlefnkodbefgpgknn (**MetaMask**)
- nkddgncdjgjfcddamfgcmfnlhccnimig (**Saturn Wallet**)
- nknhiehlklippafakaeklbeglecifhad (**Nabox Wallet**)
- nlbmnnijcnlegkjjpcfjclmcfggfefdm (**MEW CX**)
- nlgbhdfgdhgbiamfdfmbikcdghidoadd (**Byone**)
- oeljdldpnmdbchonielidgobddfffla l (**EOS Authenticator**)
- onofpnbbkehpmmoabgpcpmigafmmnjhl (**Nash Extension**)
- ookjlbkiijinhpmnjffcofjonbfbgaoc (**Temple**)

The malicious process opens and reads the Chrome browser history file:

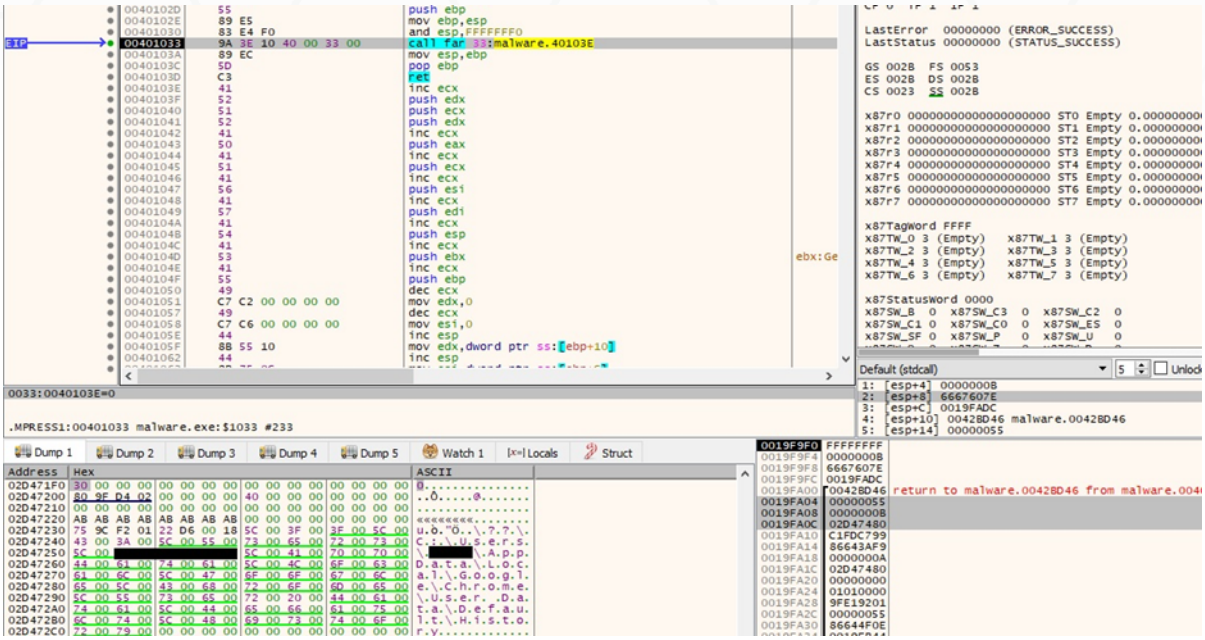

Figure 18

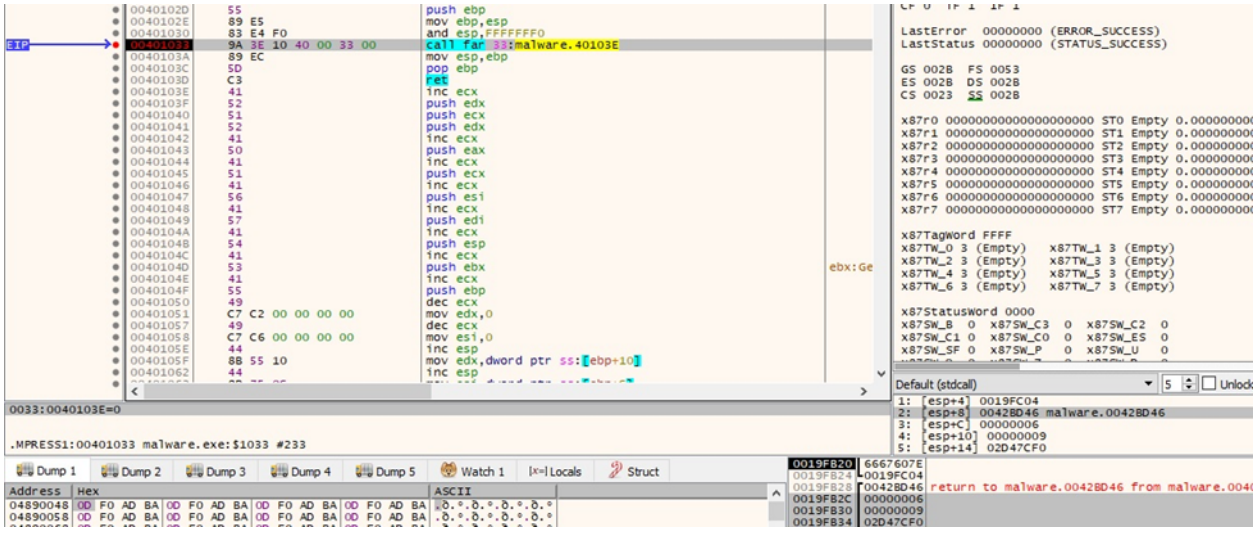

Figure 19

The “Login Data” database containing login data such as usernames and passwords will also be exfiltrated by the stealer:

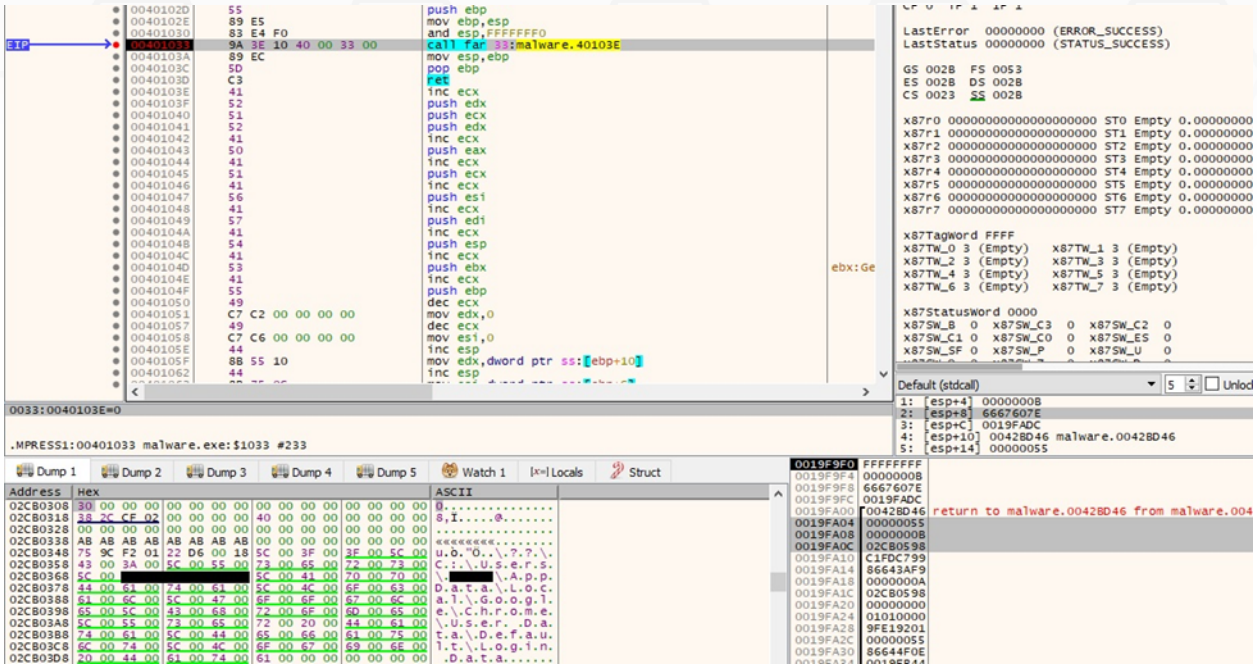

Figure 20

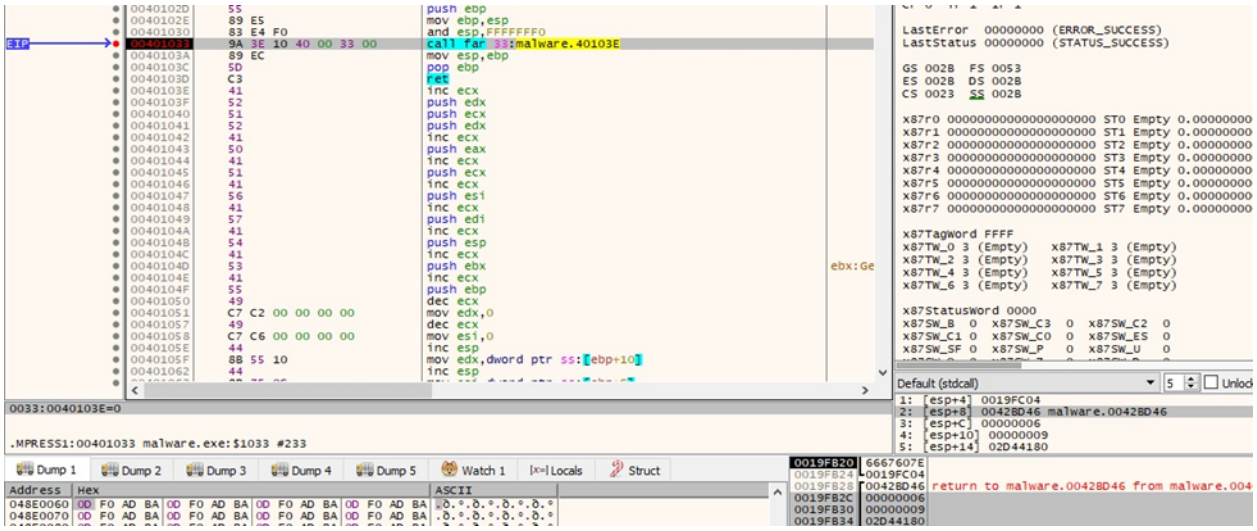

Figure 21

The malware organizes the database based on its name and the SQLite format:

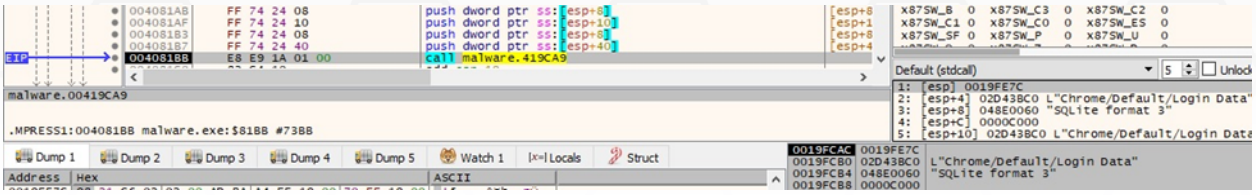

Figure 22

Multiple relevant strings are obfuscated by inserting the “edx765” string. An example of the decoding operation is displayed in figure 23.

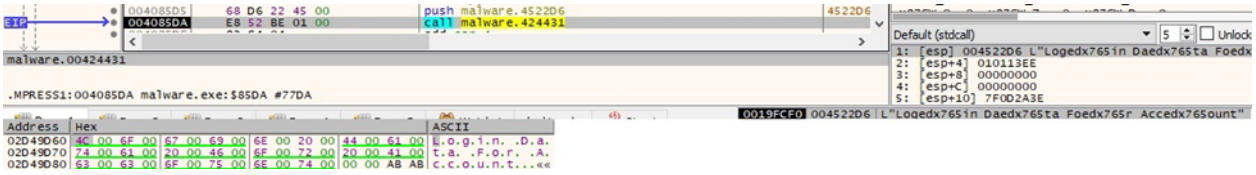

Figure 23

Another database called “Login Data For Account” is also targeted by the malware (see figure 24).

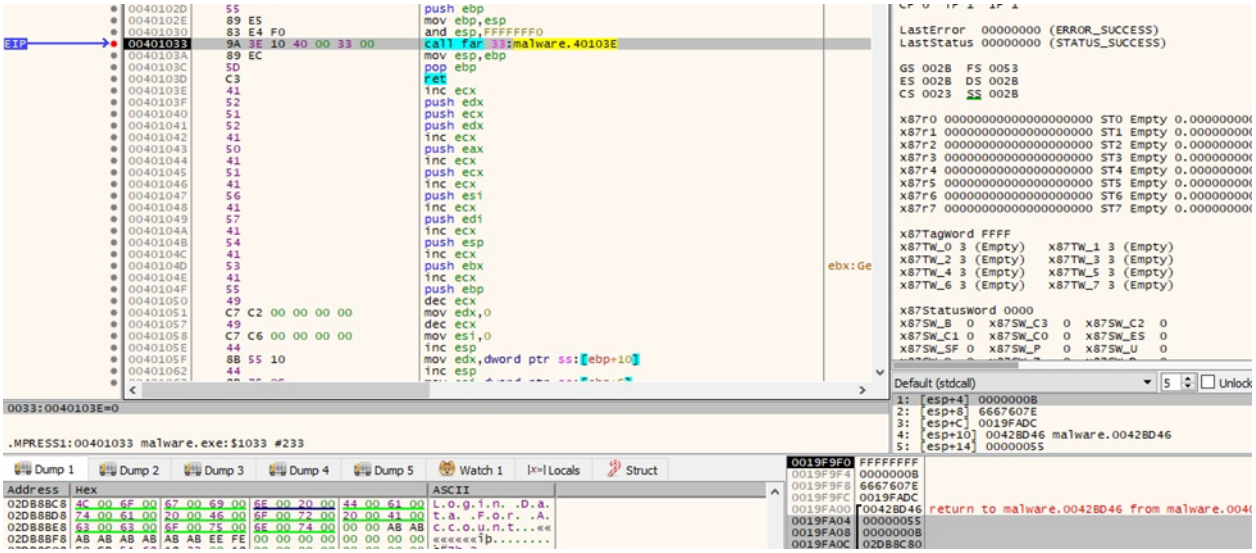

Figure 24

A Google Chrome database called “Web Data” will be exfiltrated as well:

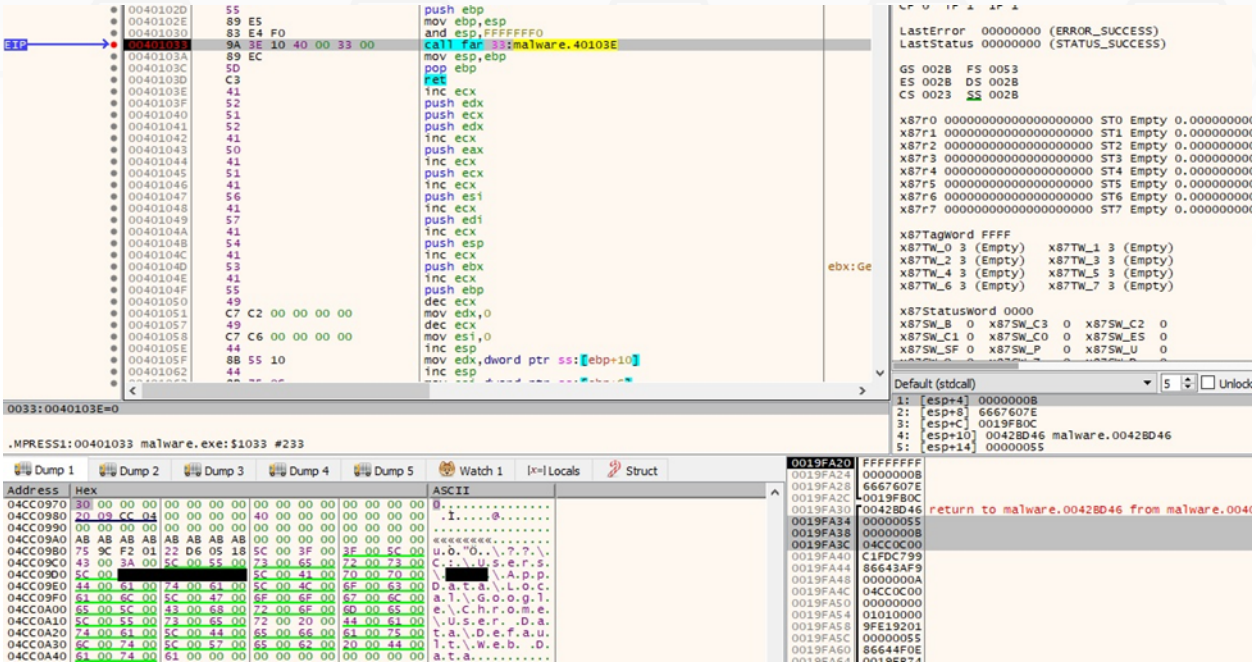

Figure 25

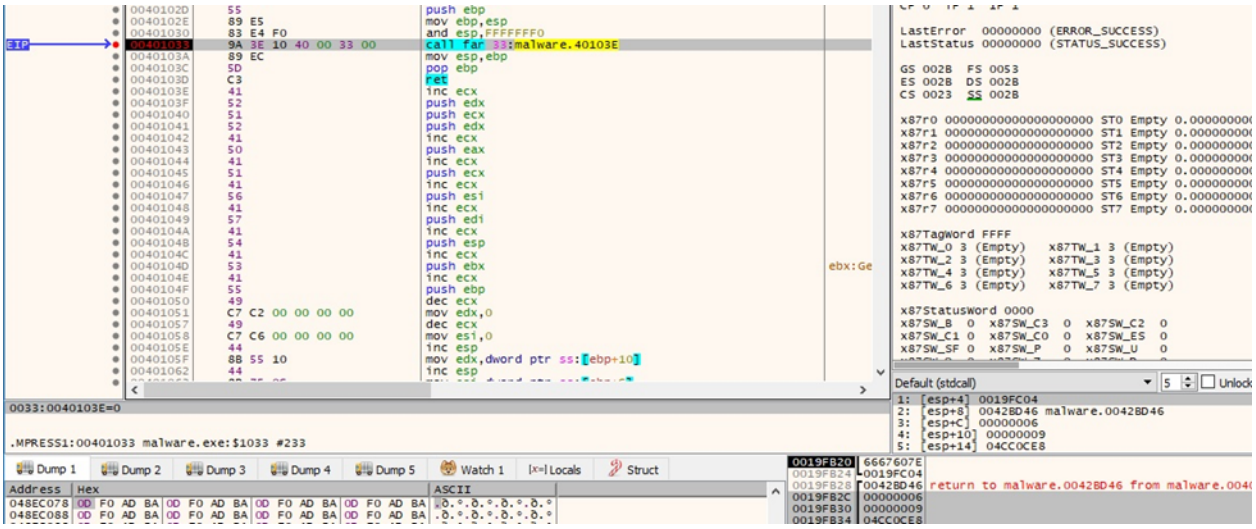

Figure 26

The process opens and reads the Google Chrome cookies database, as highlighted below:

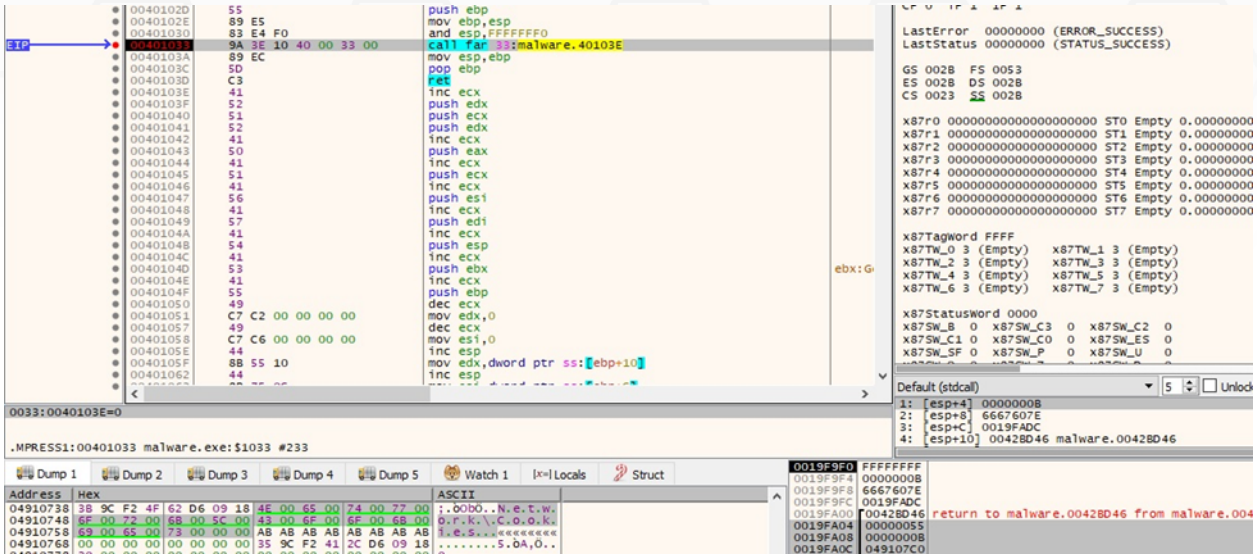

Figure 27

The local storage data can be found in the “Google\Chrome\User Data\Default\Local Storage\leveldb\” directory:

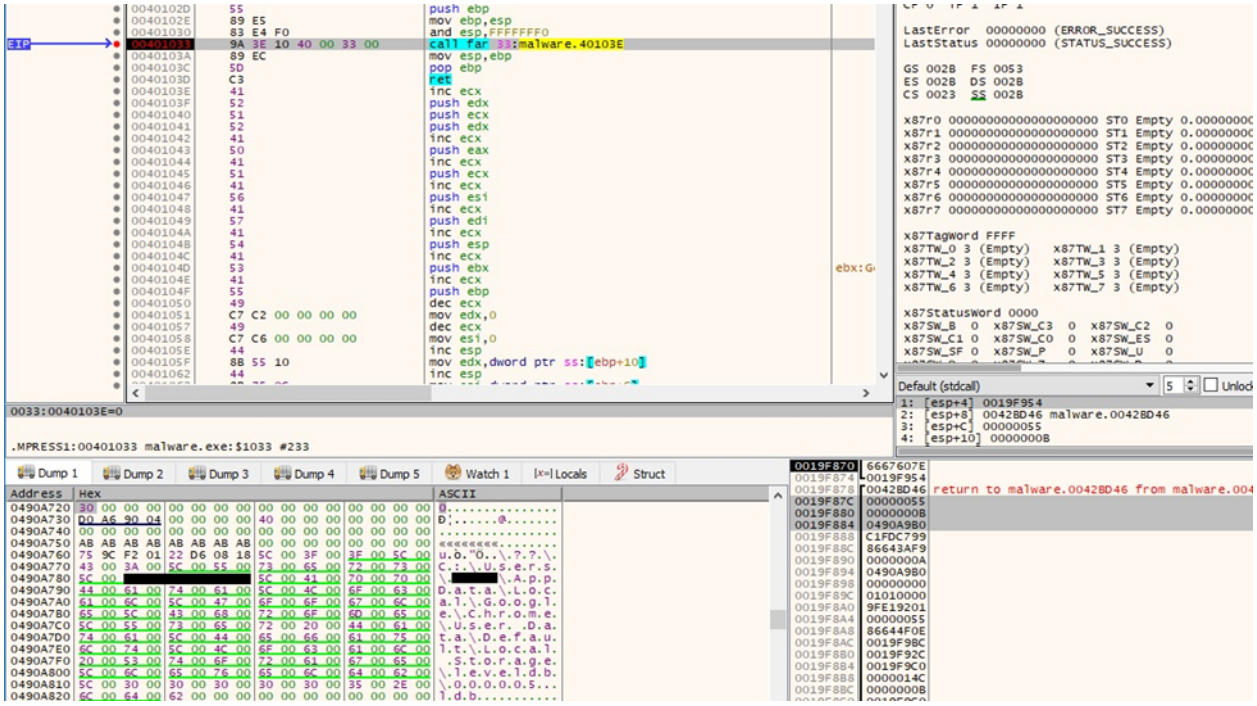

Figure 28

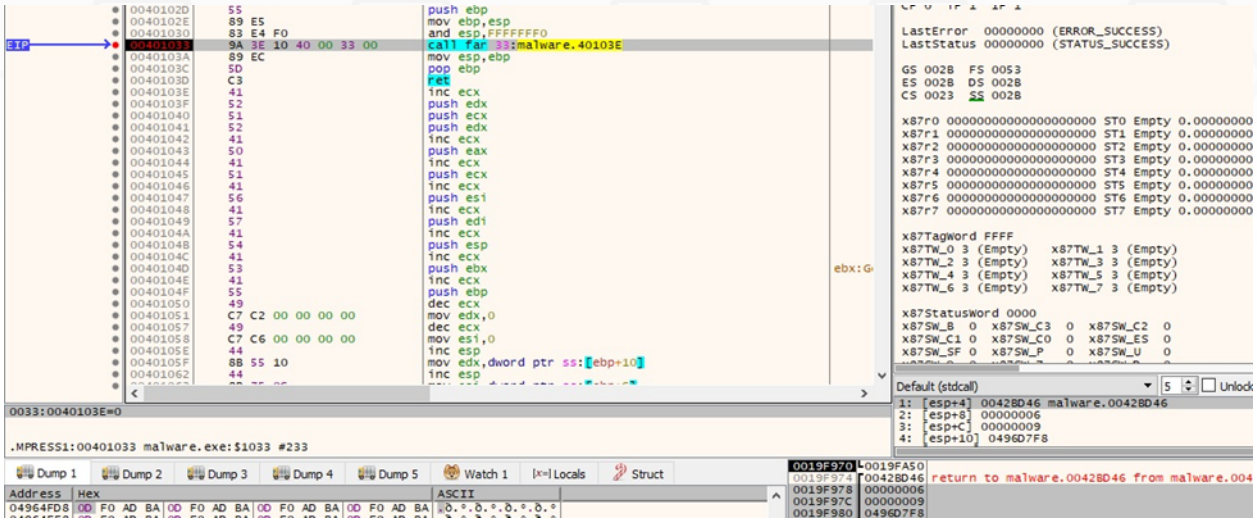

Figure 29

GetCurrentHwProfileA is utilized to obtain information about the current hardware profile (figure 30).

Figure 30

The data that will be exfiltrated contains the GUID string corresponding to the hardware profile, the “TRNGVa—stream” Lumma ID, and the targeted databases that were compressed:

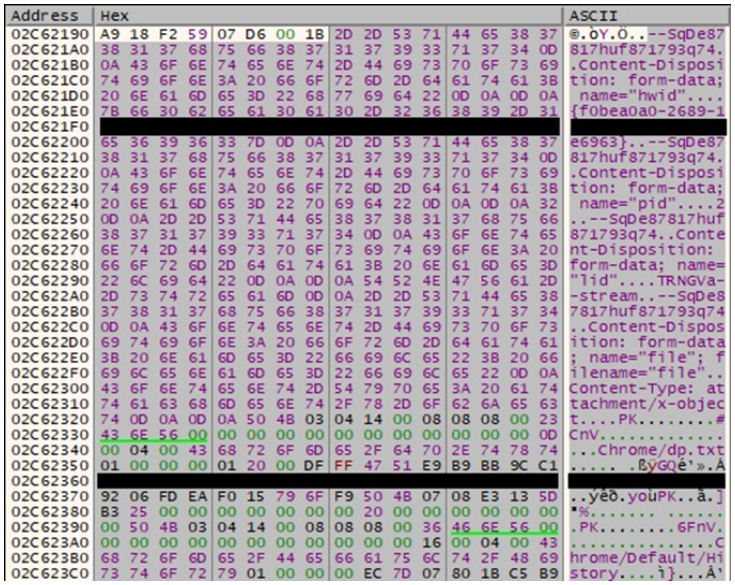

Figure 31

WinHttpOpen is used to initialize the use of WinHTTP functions with the “TeslaBrowser/5.5” user agent (see figure 32).

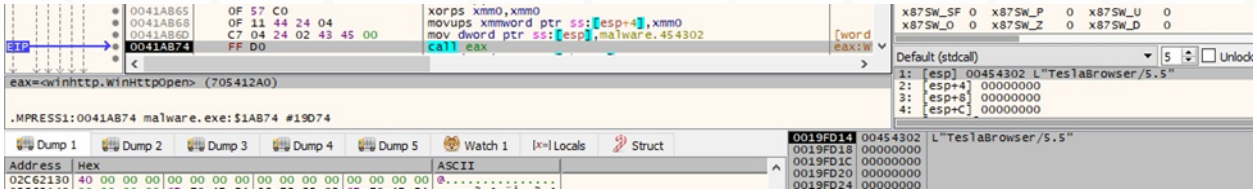

Figure 32

The malicious binary performs a network connection to the “45.9.74.78” C2 server on port 80:

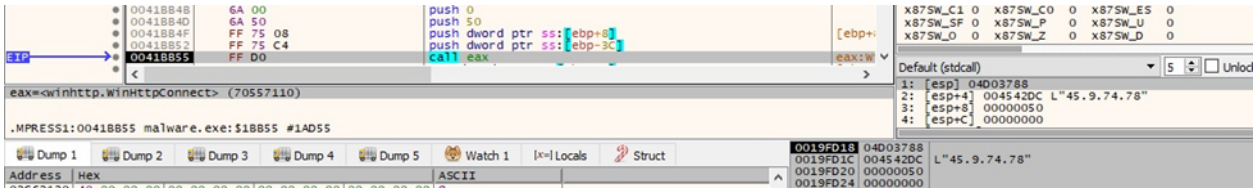

Figure 33

The WinHttpOpenRequest API is utilized to create an HTTP POST request to the “/c2sock” URI:

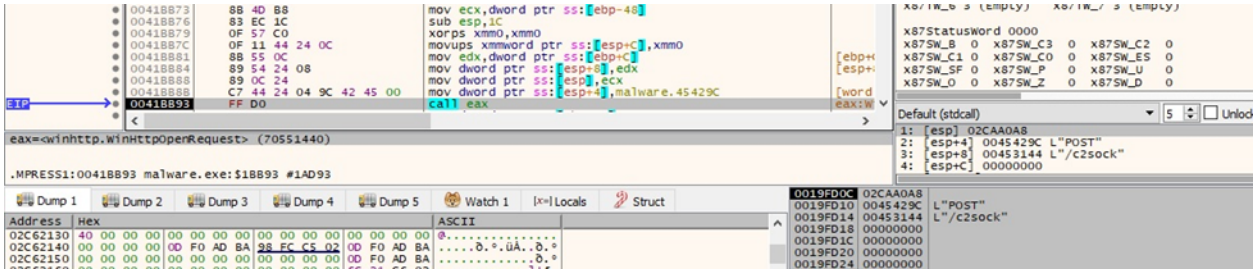

Figure 34

The stealer sets the time outs involved in the HTTP connections using WinHttpSetTimeouts (figure 35).

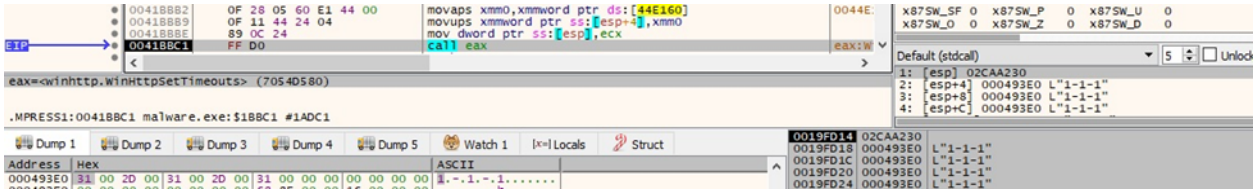

Figure 35

The malware adds an HTTP request header via a function call to WinHttpAddRequestHeaders (0x20000000 = **WINHTTP_ADDREQ_FLAG_ADD**):

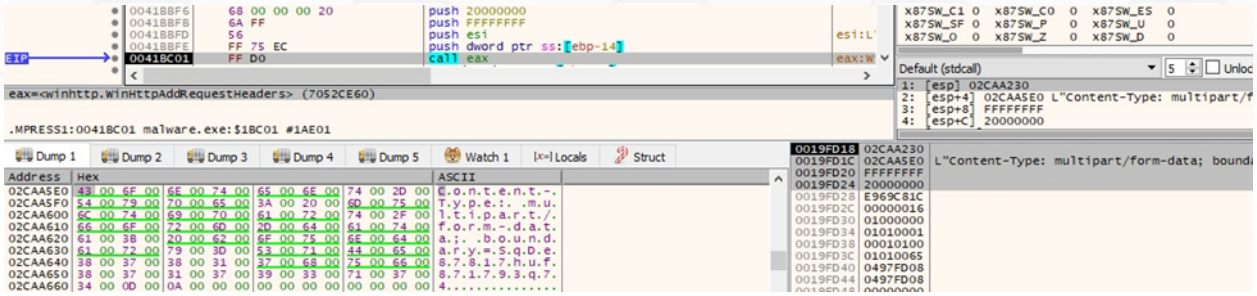

Figure 36

WinHttpSendRequest is used to send the HTTP request to the C2 server, as highlighted in figure 37.

Figure 37

The stolen information is exfiltrated by calling the WinHttpWriteData method:

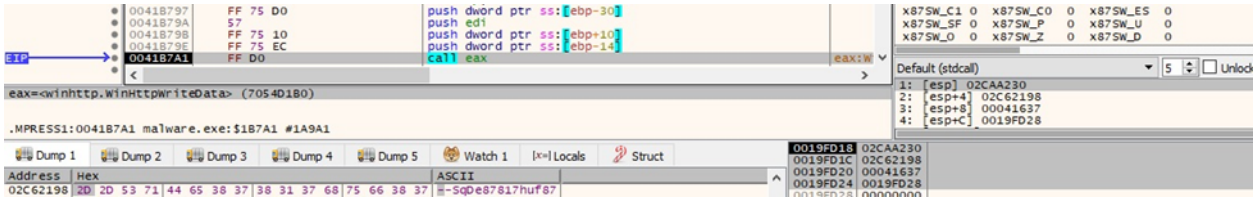

Figure 38

The process waits to receive the response from the C2 server using the WinHttpReceiveResponse function:

Figure 39

Finally, the binary closes the HINTERNET handle using WinHttpCloseHandle:

Figure 40

Other browsers are also targeted (figure 41):

- Chromium

- Microsoft Edge

- Kometa

- Opera Stable

- Opera GX Stable

- Opera Neon

- Brave

- Comodo Dragon

- CocCoc Browser

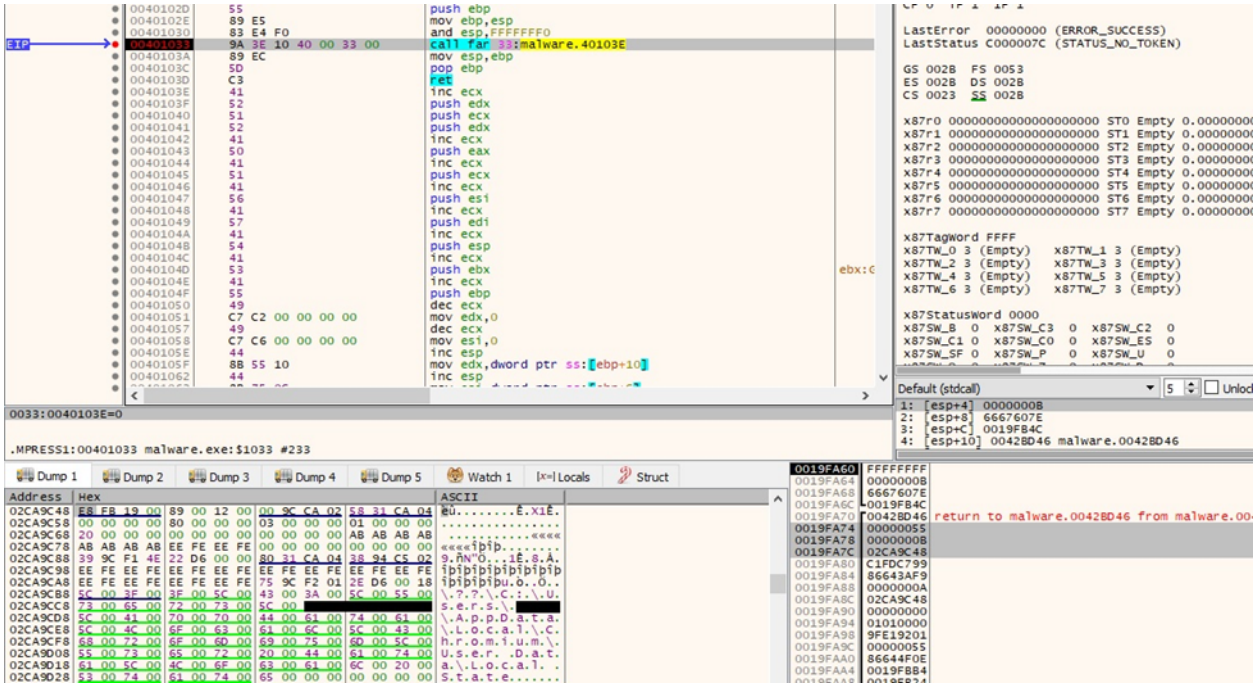

Figure 41

The stealer is looking for “.txt” files in the “C:\Users\<User>\” folder recursively:

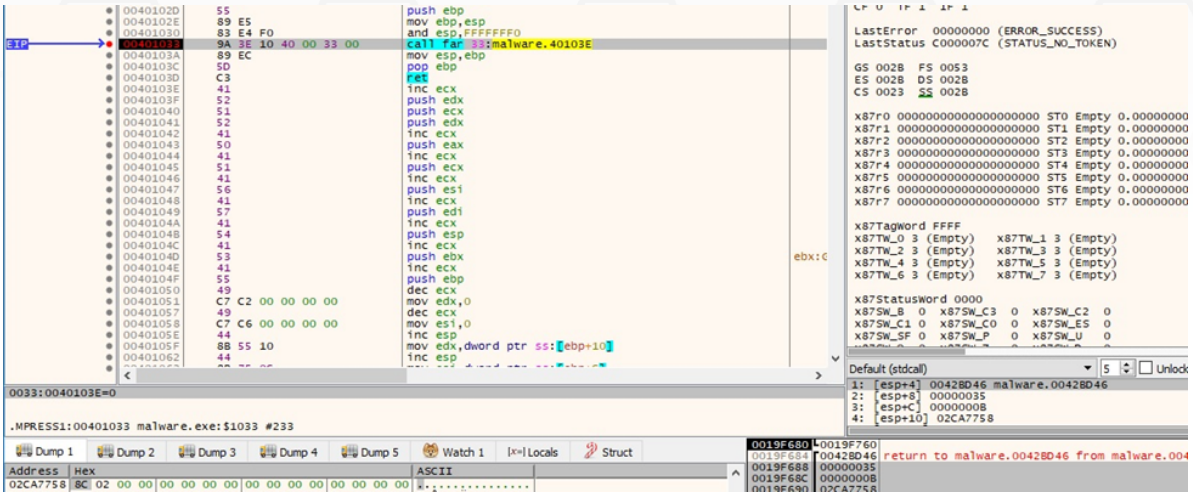

Figure 42

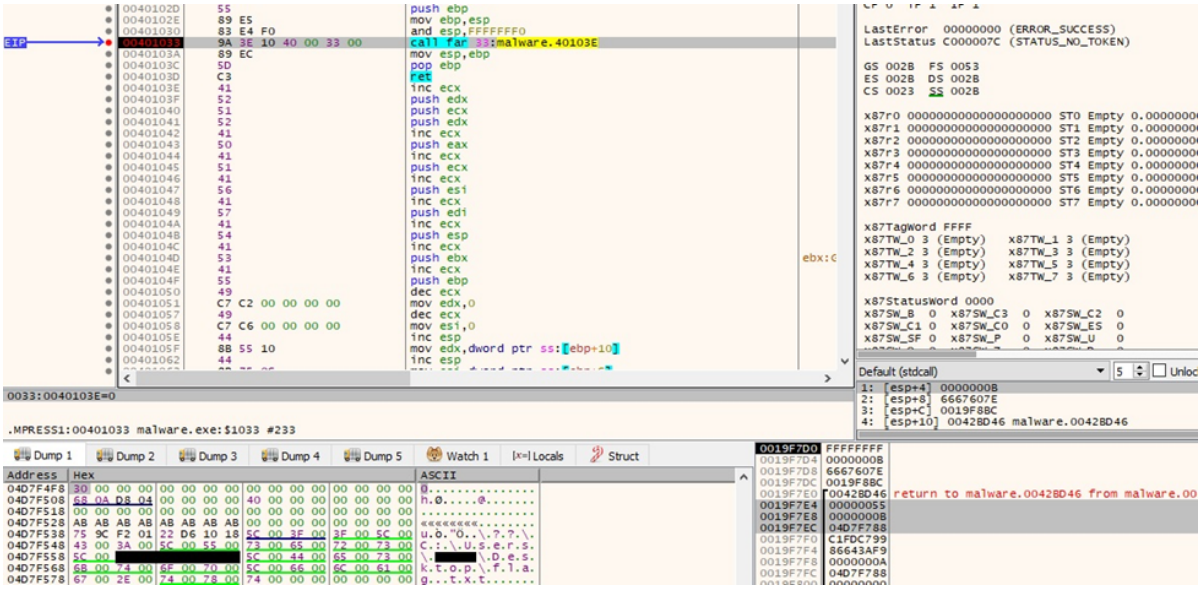

Figure 43

The malware steals data from the following crypto wallets (see figure 44):

- Binance
- Electrum
- Ethereum
- Exodus
- Ledger Live
- Atomic
- Coinomi

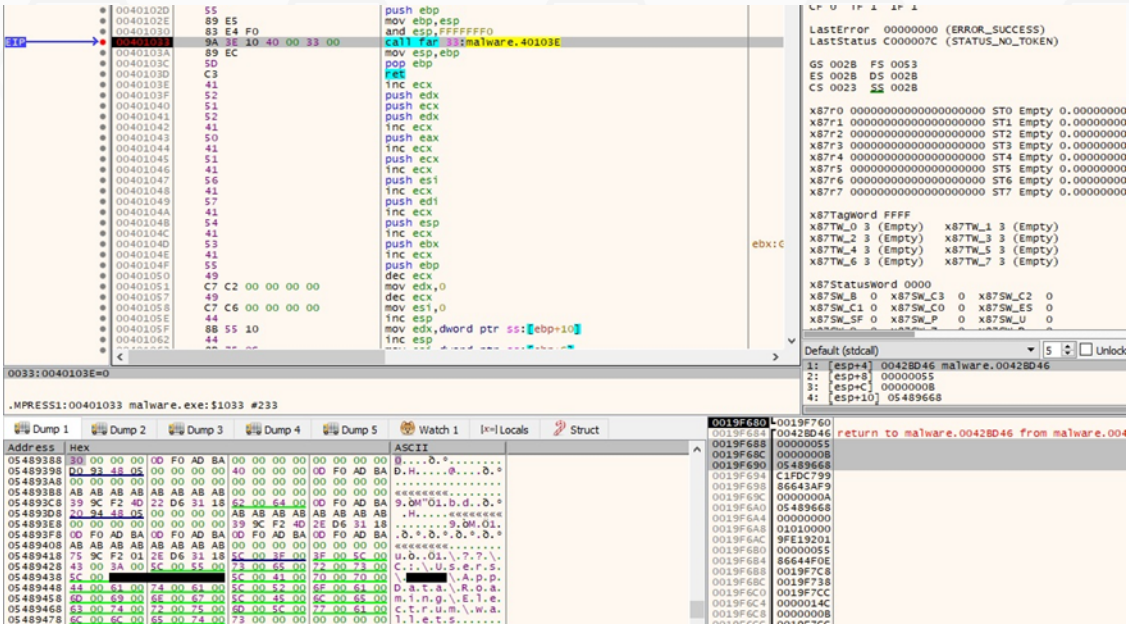

Figure 44

The text files are compressed and exfiltrated using the same approach as before:

Address	Hex	ASCII
055840F0	34 0D 0A 43 6F 6E 74 65 6E 74 2D 44 69 73 70 6F	4..Content-Dispo
05584100	73 69 74 69 6F 6E 3A 20 66 6F 72 6D 2D 64 61 74	sition: form-dat
05584110	61 3B 20 6E 61 6D 65 3D 22 6C 69 64 22 0D 0A 0D	a; name="lid"...
05584120	0A 54 52 4E 47 56 61 2D 2D 73 74 72 65 61 6D 0D	.TRNGVa--stream.
05584130	0A 2D 2D 53 71 44 65 38 37 38 31 37 68 75 66 38	.--SqDe87817huf8
05584140	37 31 37 39 33 71 37 34 0D 0A 43 6F 6E 74 65 6E	71793q74..Conten
05584150	74 2D 44 69 73 70 6F 73 69 74 69 6F 6E 3A 20 66	t-Disposition: f
05584160	6F 72 6D 2D 64 61 74 61 3B 20 6E 61 6D 65 3D 22	orm-data; name="
05584170	66 69 6C 65 22 3B 20 66 69 6C 65 6E 61 6D 65 3D	file"; filename=
05584180	22 66 69 6C 65 22 0D 0A 43 6F 6E 74 65 6E 74 2D	"file"..Content-
05584190	54 79 70 65 3A 20 61 74 74 61 63 68 6D 65 6E 74	Type: attachment
055841A0	2F 78 2D 6F 62 6A 65 63 74 0D 0A 0D 0A 50 4B 03	/x-object....PK.
055841B0	04 14 00 08 08 08 00 C5 3B 70 56 00 00 00 00 00	.......Å;pV.....
055841C0	00 00 00 00 00 00 00 27 00 04 00 49 6D 70 6F 72	.......'...Impor
055841D0	74 61 6E 74 20 46 69 6C 65 73 2F 50 72 6F 66 69	tant Files/Profi
055841E0	6C 65 2F 44 65 73 6B 74 6F 70 2F 62 6C 61 2E 74	le/Desktop/bla.t
055841F0	78 74 01 00 00 00 55 93 39 8E 1D 31 0C 44 73 03	xt....U.9..1.Ds.
05584200	BE C3 60 52 27 6A AD AD C0 37 F9 89 D6 1B 18 F8	¾Ã`R'j..À7ù.Ö..ø
05584210	C7 F7 AB 61 0F 06 0A 88 A6 48 16 97 22 BB FF DB	C÷«a....¦H..”»ÿÛ

Figure 45

The stealer searches for “.conf” files in the AnyDesk directory:

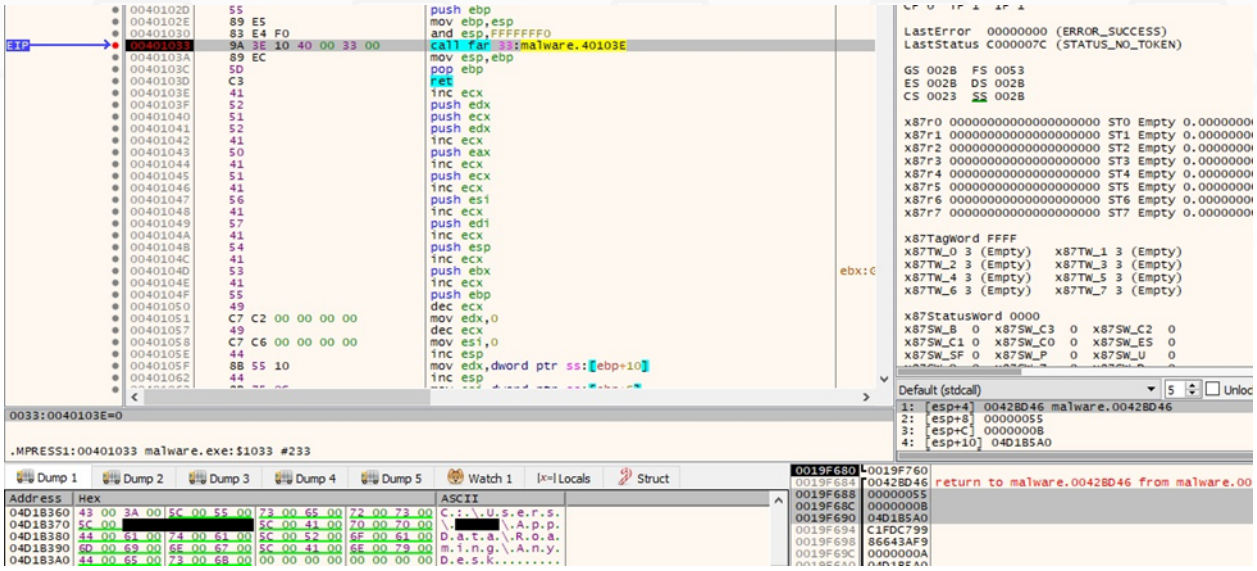

Figure 46

The “recentservers.xml” and “sitemanager.xml” files found in the FileZilla directory will also be exfiltrated:

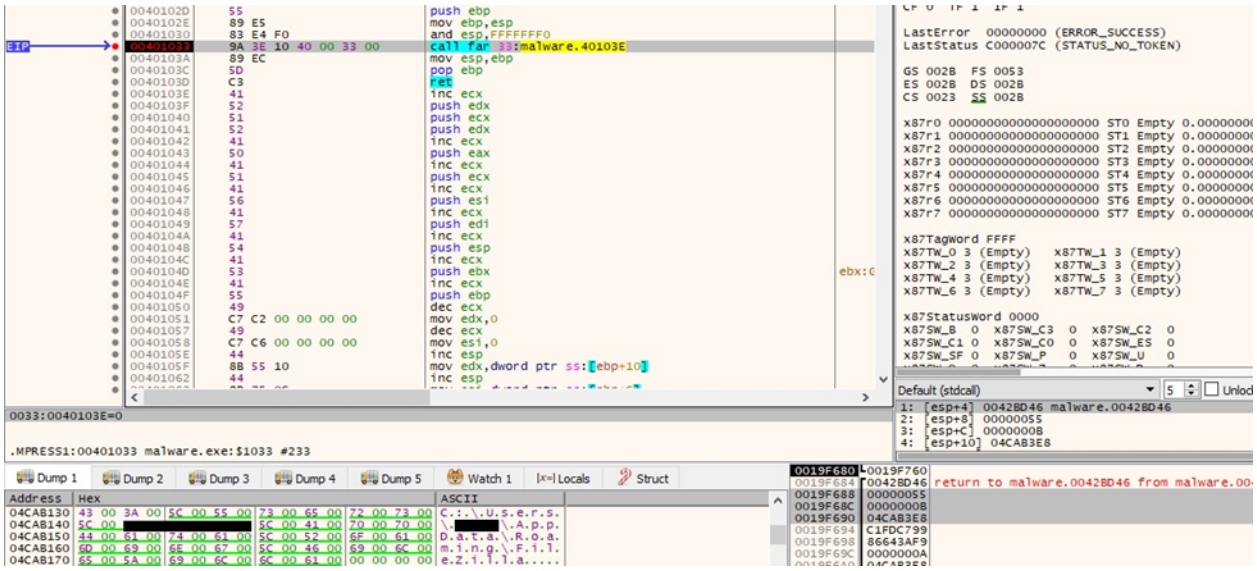

Figure 47

The malicious binary searches for KeePass files (*.kbdx) in the user profile directory. It locates the Steam folder and detects the “ssfn*” files, as shown in the figure below.

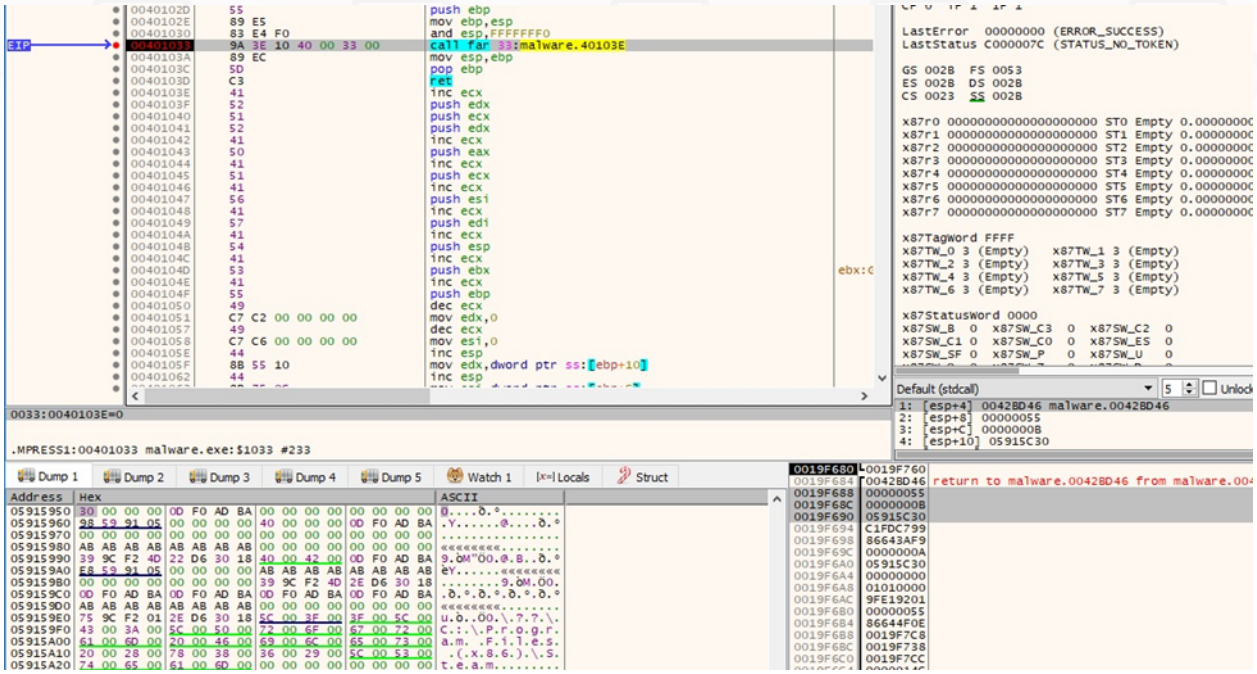

Figure 48

The process steals "*s" files from Telegram Desktop:

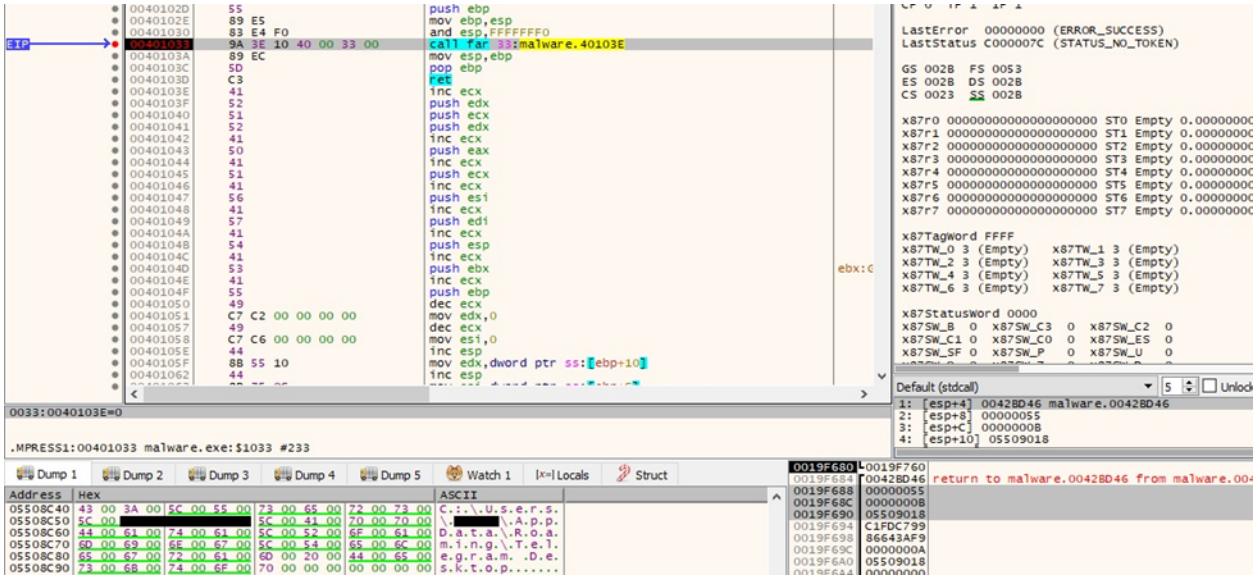

Figure 49

The binary steals Mozilla Firefox passwords, and the "key4.db" database found in the Profiles directory:

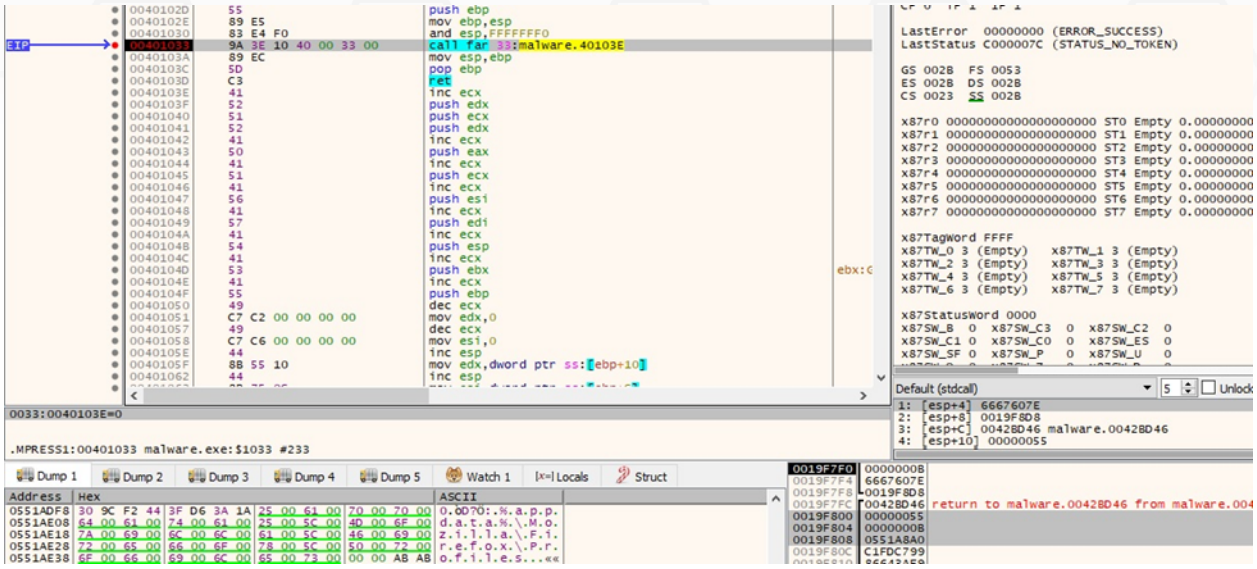

Figure 50

The width and height of the screen of the primary display monitor are retrieved using GetSystemMetrics (0x0 = **SM_CXSCREEN**, 0x1 = **SM_CYSCREEN**):

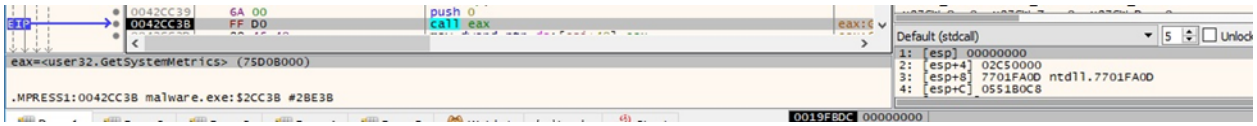

Figure 51

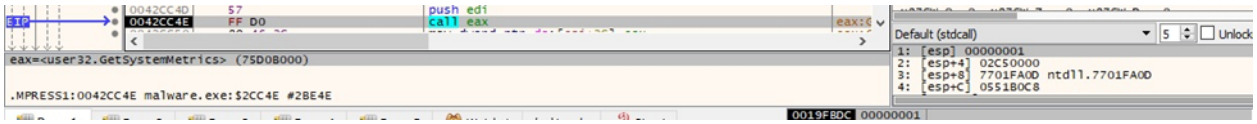

Figure 52

GetComputerNameA is utilized to extract the NetBIOS name of the local machine (see figure 53).

Figure 53

The sample also obtains the username associated with the current thread:

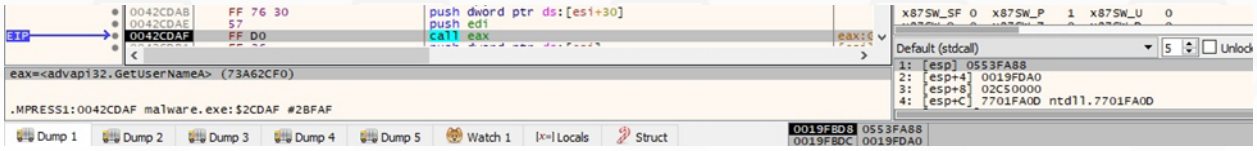

Figure 54

The malware retrieves the system default language by calling the GetSystemDefaultLocaleName API:

Figure 55

The cpuid instruction is used to extract the processor name and type, as displayed in figure 56.

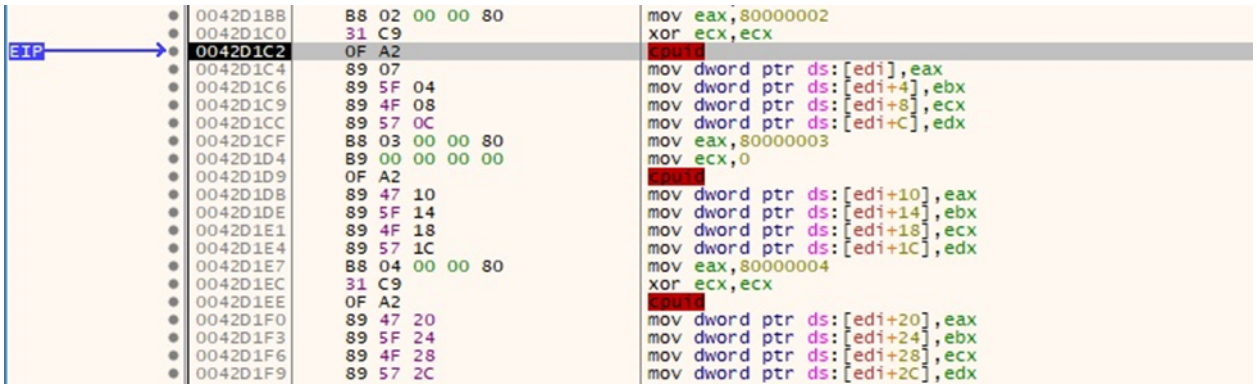

Figure 56

GetPhysicallyInstalledSystemMemory is utilized to obtain the RAM amount that is installed on the computer:

Figure 57

Finally, the stealer exfiltrates an archive containing a file called “System.txt” that is shown in figure 58, and another one called “Software.txt” that contains the installed software.

```
LummaC2, Build 20233101
LID(Lumma ID): TRNGVa--stream

- PC: [redacted]
- User: [redacted]
- OS Version: Windows 10 (10.0.16299)
- HWID: [redacted]
- Screen Resoluton: 1920x1080
- Language: en-US
- CPU Name: [redacted]
- GPU: VMware SVGA 3D
- Physical Installed Memory: 2048MB
```

Figure 58

The process takes a screenshot and exfiltrates it as “Screen.png”:

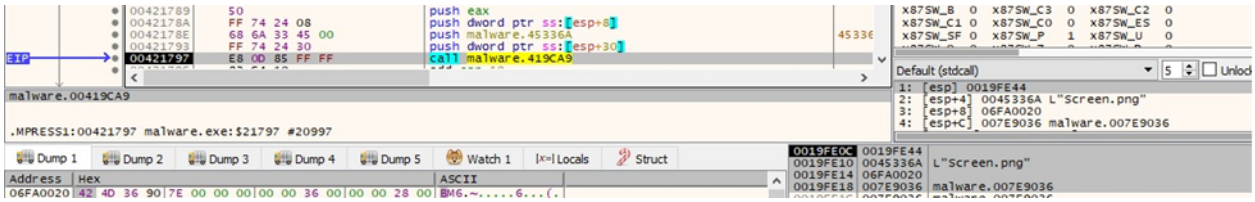

Figure 59

Indicators of Compromise

SHA256

199de8b727ceae96afb7c7560092c1d7a4dbe5a005c07ae20cffd9871da52b82

C2 server

45.9.74.78

User agent

TeslaBrowser/5.5