Arkanix Stealer targets a variety of data, offers a MaaS referral program

SL securelist.com/arkanix-stealer/119006

Kirill Korchemny

February 19, 2026

Introduction

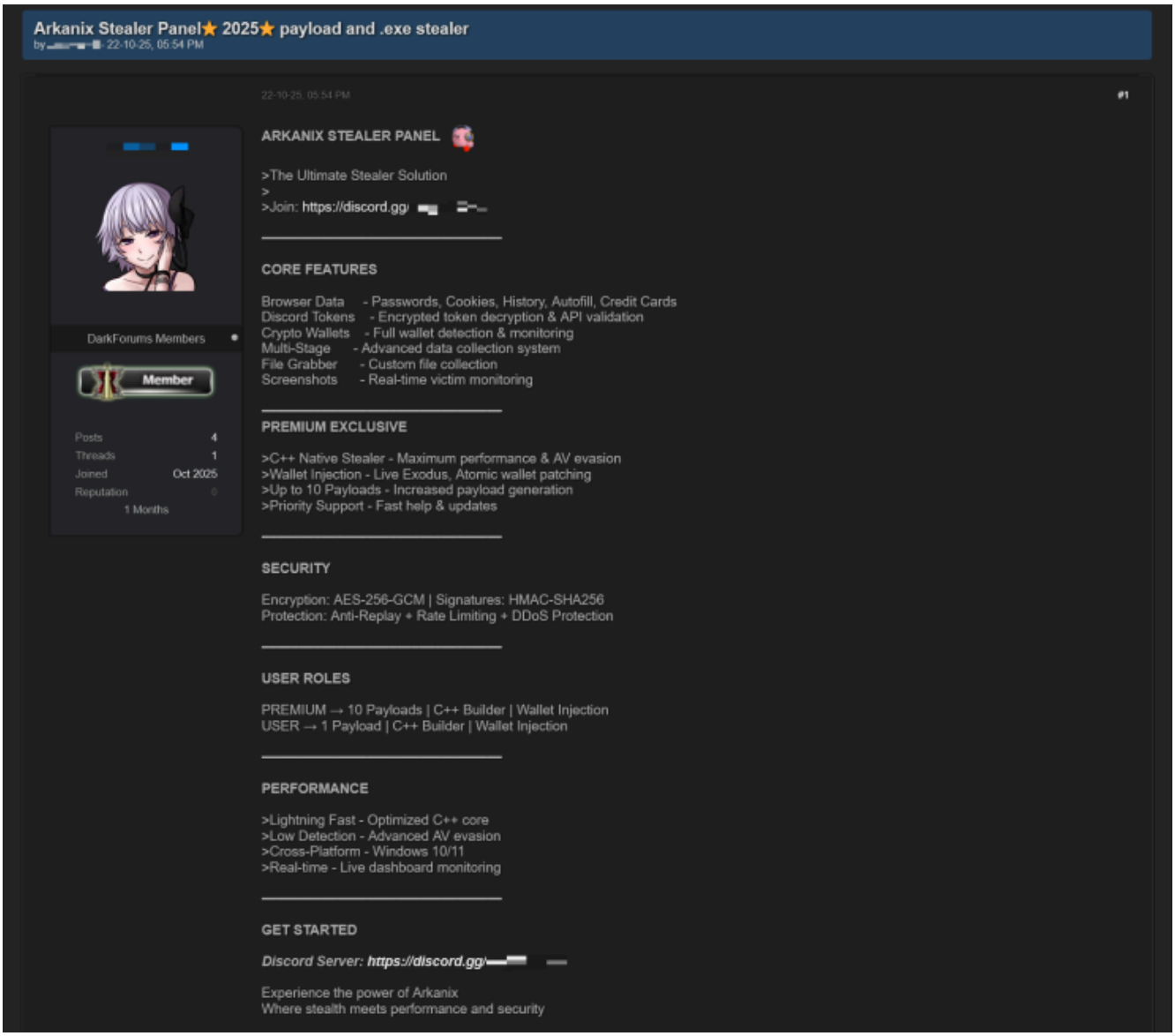
In October 2025, we discovered a series of forum posts advertising a previously unknown stealer, dubbed “Arkanix Stealer” by its authors. It operated under a MaaS (malware-as-a-service) model, providing users not only with the implant but also with access to a control panel featuring configurable payloads and statistics. The set of implants included a publicly available browser post-exploitation tool known as ChromElevator, which was delivered by a native C++ version of the stealer. This version featured a wide range of capabilities, from collecting system information to stealing cryptocurrency wallet data. Alongside that, we have also discovered Python implementation of the stealer capable of dynamically modifying its configuration. The Python version was often packed, thus giving the adversary multiple methods for distributing their malware. It is also worth noting that Arkanix was rather a one-shot malicious campaign: at the time of writing this article, the affiliate program appears to be already taken down.

Kaspersky products detect this threat as `Trojan-PSW.Win64.Coins.*`, `HEUR:Trojan-PSW.Multi.Disco.gen`, `Trojan.Python.Agent.*`.

Technical details

Background

In October 2025, a series of posts was discovered on various dark web forums, advertising a stealer referred to by its author as “Arkanix Stealer”. These posts detail the features of the stealer and include a link to a Discord server, which serves as the primary communication channel between the author and the users of the stealer.



Example of an Arkanix Stealer advertisement

Upon further research utilizing public resources, we identified a set of implants associated with this stealer.

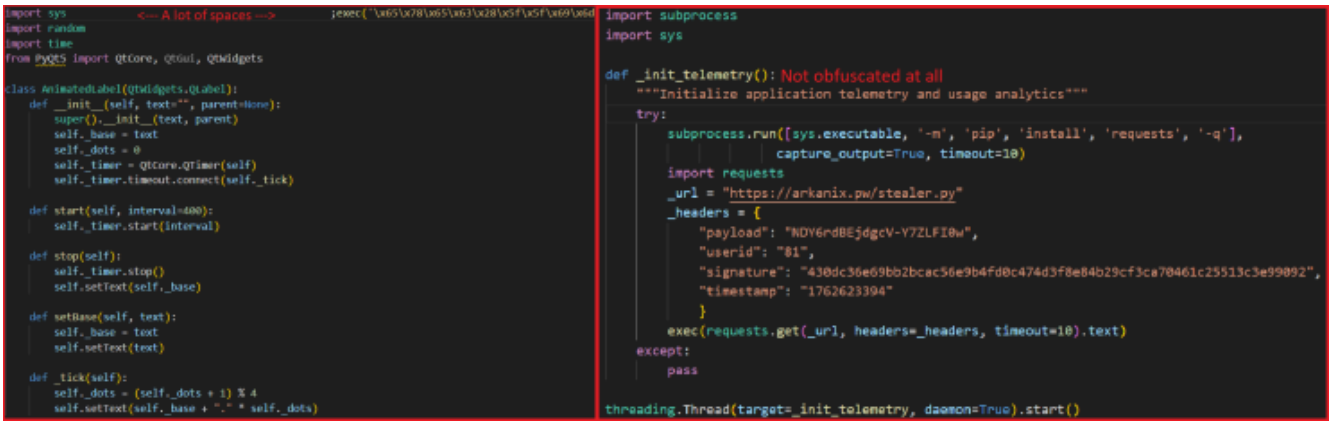
Initial infection or spreading

The initial infection vector remains unknown. However, based on some of the file names (such as `steam_account_checker_pro_v1.py`, `discord_nitro_checker.py`, and `TikTokAccountBotter.exe`) of the loader scripts we obtained, it can be concluded with high confidence that the initial infection vector involved phishing.

Python loader

MD5	208fa7e01f72a50334f3d7607f6b82bf
File name	discord_nitro_code_validator_right_aligned.py

The Python loader is the script responsible for downloading and executing the Python-based version of the Arkanix infostealer. We have observed both plaintext Python scripts and those bundled using PyInstaller or Nuitka, all of which share a common execution vector and are slightly obfuscated. These scripts often serve as decoys, initially appearing to contain legitimate code. Some of them do have useful functionality, and others do nothing apart from loading the stealer. Additionally, we have encountered samples that employ no obfuscation at all, in which the infostealer is launched in a separate thread via Python's built-in `threading` module.



Variants of Python loaders executing the next stage

Upon execution, the loader first installs the required packages — namely, `requests`, `pycryptodome`, and `psutil` — via the `pip` package manager, utilizing the `subprocess` module. On Microsoft Windows systems, the loader also installs `pywin32`. In some of the analyzed samples, this process is carried out twice. Since the loader does not perform any output validation of the module installation command, it proceeds to make a POST request to `hxxps://arkanix[.]pw/api/session/create` to register the current compromised machine on the panel with a predefined set of parameters even if the installation failed. After that, the stealer makes a GET request to `hxxps://arkanix[.]pw/stealer.py` and executes the downloaded payload.

Python stealer version

MD5	af8fd03c1ec81811acf16d4182f3b5e1
File name	–

During our research, we obtained a sample of the Python implementation of the Arkanix stealer, which was downloaded from the endpoint `hxxps://arkanix[.]pw/stealer.py` by the previous stage.

The stealer's capabilities — or features, as referred to by the author — in this version are configurable, with the default configuration predefined within the script file. To dynamically update the feature list, the stealer makes a GET request to `hxxps://arkanix[.]pw/api/features/{payload_id}`, indicating that these capabilities can be modified on the panel side. The feature list is identical to the one that was described [in the GDATA report](#).

```
FEATURE_FLAGS = {
    'collect_system_info': True,
    'collect_browser_history': True,
    'collect_autofill_data': True,
    'collect_filezilla': True,
    'collect_vpn_data': True,
    'collect_steam': True,
    'collect_screenshots': False,
    'telegram_optimized': True,
    'discord_self_spread': False,
    'discord_spam_config': {'message': 'Check this out!', 'max_dms': 10, 'delay': 5}
}
```

Configurable options

Prior to executing the information retrieval-related functions, the stealer makes a request to `hxxps://arkanix[.]pw/upload_dropper.py`, saves the response to `%TEMP%\upd_{random 8-byte name}.py`, and executes it. We do not have access to the contents of this script, which is referred to as the "dropper" by the attackers.

During its main information retrieval routine, at the end of each processing stage, the collected information is serialized into JSON format and saved to a predefined path, such as `%LOCALAPPDATA\Arkanix_lol\%info_class%.json`.

In the following, we will provide a more detailed description of the Python version's data collection features.

System info collection

Arkanix Stealer is capable of collecting a set of info about the compromised system. This info includes:

- OS version
- CPU and GPU info
- RAM size
- Screen resolution
- Keyboard layout
- Time zone
- Installed software
- Antivirus software
- VPN

Information collection is performed using standard shell commands with the exception of the VPN check. The latter is implemented by querying the endpoint `hxxps://ipapi[.]co/json/` and verifying whether the associated IP address belongs to a known set of VPNs, proxies, or Tor exit nodes.

Browser features

This stealer is capable of extracting various types of data from supported browsers (22 in total, ranging from the widely popular Google Chrome to the Tor Browser). The list of supported browsers is hardcoded, and unlike other parameters, it cannot be modified during execution. In addition to a separate Chrome grabber module (which we'll discuss later), the stealer itself supports the extraction of diverse information, such as:

- Browser history (URLs, visit count and last visit)
- Autofill information (email, phone, addresses and payment cards details)
- Saved passwords
- Cookies
- In case of Chromium-based browsers, 0Auth2 data is also extracted

All information is decrypted using either the Windows DPAPI or AES, where applicable, and searched for relevant keywords. In the case of browser information collection, the stealer searches exclusively for keywords related to banking (e.g., "revolut", "stripe", "bank") and cryptocurrencies (e.g., "binance", "metamask", "wallet"). In addition to this, the stealer is capable of extracting extension data from a hardcoded list of extensions associated with cryptocurrencies.

```
BROWSER_EXTENSIONS = [
    {"name": "Authenticator", "path": "\\Local Extension Settings\\bhghoamapcdpbohphigoooaddinpkbai"},
    {"name": "Binance", "path": "\\Local Extension Settings\\fhbohimaelbohpjbbldcngcnapndodjp"},
    {"name": "Bitapp", "path": "\\Local Extension Settings\\fihkakfobkmkjojpchpfgcmhfjnmnfpi"},
    {"name": "BoltX", "path": "\\Local Extension Settings\\aodkkagnadcbobfpggfnjeongemjbjca"},
    {"name": "Coin98", "path": "\\Local Extension Settings\\aeachknmefphepccionboohckonoeemg"},
    {"name": "Coinbase", "path": "\\Local Extension Settings\\hnfanknocfeofbddgcijnmhnfnkdnaad"},
    {"name": "Core", "path": "\\Local Extension Settings\\agoakfejjabomempkjlepdflaleeobhb"},
    {"name": "Crocobit", "path": "\\Local Extension Settings\\pnlfjmlcjdjgkddecgincndfgegkecke"},
    {"name": "Equal", "path": "\\Local Extension Settings\\blnieiiffboillknjnepogjhkgnoapac"},
    {"name": "Ever", "path": "\\Local Extension Settings\\cgeeodpfagjceefieflmdfphplkenlfk"},
    {"name": "ExodusWeb3", "path": "\\Local Extension Settings\\aholpfdialjgjfhomihkjbmgjidlcdno"},
    {"name": "Fewcha", "path": "\\Local Extension Settings\\ebfidpplhabeedpnhjnobghokpiioolj"},
    {"name": "Finnie", "path": "\\Local Extension Settings\\cjmkndjhnagcfbpiemnkdpomccnjblmj"},
    {"name": "Guarda", "path": "\\Local Extension Settings\\hpglfhgfnhbgpjdenjgmdgoeiappafln"},
    {"name": "Guild", "path": "\\Local Extension Settings\\nanjmdknhkinifnkgdcggcfnhdaammmj"},
    {"name": "HarmonyOutdated", "path": "\\Local Extension Settings\\fnnegphlobjdpkhecapkijjdkgcjhkib"},
    {"name": "Iconex", "path": "\\Local Extension Settings\\flpiciilemghbmfalicajoolhkkenfel"},
    {"name": "Jaxx Liberty", "path": "\\Local Extension Settings\\cjelfplplebdjjenllpjcblmjkfcffne"},
    {"name": "Kaikas", "path": "\\Local Extension Settings\\jblndlipeogpafnldhgmapagcccfchpi"},
    {"name": "KardiaChain", "path": "\\Local Extension Settings\\pdadjkfkgcafgbceimcpbkalnfnepbnk"},
```

Part of the extension list which the stealer utilizes to extract data from

Telegram info collection

Telegram data collection begins with terminating the Telegram.exe process using the `taskkill` command. Subsequently, if the `telegram_optimized` feature is set to False, the malware zips the entire tdata directory (typically located at `%APPDATA%\Roaming\Telegram Desktop\tdata`) and transmits it to the attacker. Otherwise, it selectively copies and zips only the subdirectories containing valuable info, such as message log. The generated archive is sent to the endpoint `/delivery` with the filename `tdata_session.zip`.

Discord capabilities

The stealer includes two features connected with Discord: credentials stealing and self-spreading. The first one can be utilized to acquire credentials both from the standard client and custom clients. If the client is Chromium-based, the stealer employs the same data exfiltration mechanism as during browser credentials stealing.

The self-spreading feature is configurable (meaning it can be disabled in the config). The stealer acquires the list of user's friends and channels via the Discord API and sends a message provided by the attacker. This stealer does not support attaching files to such messages.

VPN data collection

The VPN collector is searching for a set of known VPN software to extract account credentials from the credentials file with a known path that gets parsed with a regular expression. The extraction occurs from the following set of applications:

- Mullvad VPN

- NordVPN
- ExpressVPN
- ProtonVPN

File retrieval

File retrieval is performed regardless of the configuration. The script relies on a predefined set of paths associated with the current user (such as Desktop, Download, etc.) and file extensions mainly connected with documents and media. The script also has a predefined list of filenames to exfiltrate. The extracted files are packed into a ZIP archive which is later sent to the C2 asynchronously. An interesting aspect is that the filename list includes several French words, such as “motdepasse” (French for “password”), “banque” (French for “bank”), “secret” (French for “secret”), and “compte” (French for “account”).

Other payloads

We were able to identify additional modules that are downloaded from the C2 rather than embedded into the stealer script; however, we weren’t able to obtain them. These modules can be described by the following table, with the “Details” column referring to the information that could be extracted from the main stealer code.

Module name	Endpoint to download	Details
Chrome grabber	/api/chrome-grabber-template/{payload_id}	–
Wallet patcher	/api/wallet-patcher/{payload_id}	Checks whether “Exodus” and “Atomic” cryptocurrency wallets are installed
Extra collector	/api/extra-collector/{payload_id}	Uses a set of options from the config, such as collect_filezilla, collect_vpn_data, collect_steam, and collect_screenshots
HVNC	/hvnc	Is saved to the Startup directory (%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\hvnc.py) to execute upon system boot

The Wallet patcher and Extra collector scripts are received in an encrypted form from the C2 server. To decrypt them, the attackers utilize the AES-GCM algorithm in conjunction with PBKDF2 (HMAC and SHA256). After decryption, the additional payload has its template placeholders replaced and is stored under a partially randomized name within a temporary folder.

```
def decrypt_with_payload_key(encrypted_b64, key_suffix=''):
    try:
        import hashlib
        secret_salt = b'arkanix_panel_secret_2025'
        key = hashlib.pbkdf2_hmac(
            'sha256',
            (payload + key_suffix).encode('utf-8'),
            secret_salt,
            100000,
            dklen=32
        ) key_suffix can be either "_patcher" or "_collector"
        encrypted = base64.b64decode(encrypted_b64)
        nonce = encrypted[:16]
        tag = encrypted[16:32]
        ciphertext = encrypted[32:]
        from Crypto.Cipher import AES
        cipher = AES.new(key, AES.MODE_GCM, nonce=nonce)
        plaintext = cipher.decrypt_and_verify(ciphertext, tag)
        return plaintext.decode('utf-8')
    except:
        return None
```

```
encrypted_collector = response.read().decode('utf-8')
collector_code = decrypt_with_payload_key(encrypted_collector, '_collector')
if not collector_code:
    return
collector_code = collector_code.replace('{{PANEL_URL}}', BASE_URL)
collector_code = collector_code.replace('{{PAYLOAD_ID}}', payload)
collector_code = collector_code.replace('{{USERID}}', userid)
collector_code = collector_code.replace('{{SESSION_TOKEN}}', session_token or '')
collector_code = collector_code.replace('{{DESKTOP_NAME}}', desktop_name or '')
collector_code = collector_code.replace('{{HWID}}', hwid or '')
collector_code = collector_code.replace('{{FEATURES}}', ','.join(extra_features))
collector_code = collector_code.replace('{{REAL_IP}}', real_ip or 'unknown')
temp_dir = os.getenv('TEMP', '/tmp')
import random
random_name = ''.join([chr(random.randint(97, 122)) for _ in range(8)])
collector_path = os.path.join(temp_dir, f'ec_{random_name}.py')
```

Decryption routine and template substitution

Once all operations are completed, the stealer removes itself from the drive, along with the artifacts folder (`Arkanix_lol` in this case).

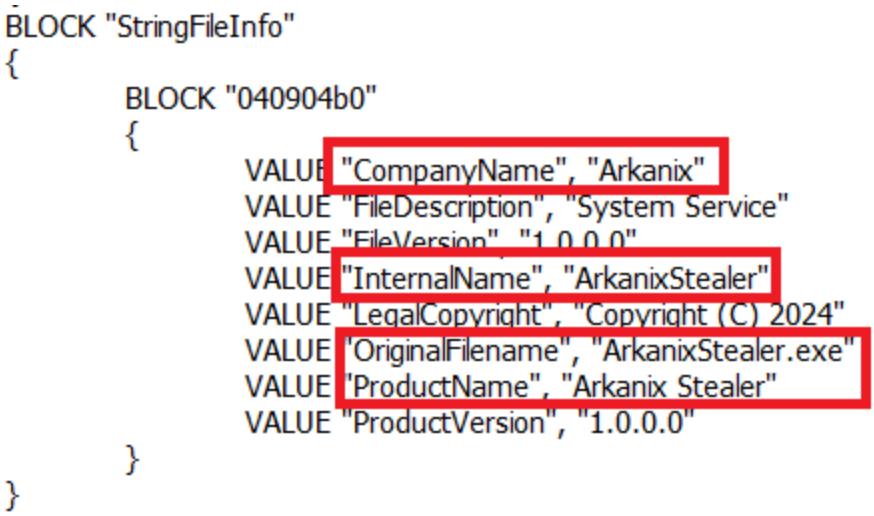
Native version of stealer

MD5	a3fc46332dcd0a95e336f6927bae8bb7
File name	ArkanixStealer.exe

During our analysis, we were able to obtain both the release and debug versions of the native implementation, as both were uploaded to publicly available resources. The following are the key differences between the two:

- The release version employs VMProtect, but does not utilize code virtualization.
- The debug version communicates with a Discord bot for command and control (C2), whereas the release version uses the previously mentioned C2 domain `arkanix[.]pw`.
- The debug version includes extensive logging, presumably for the authors' debugging purposes.

Notably, the native implementation explicitly references the name of the stealer in the `VersionInfo` resources. This naming convention is consistent across both the debug version and certain samples containing the release version of the implant.



Version info

After launching, the stealer implements a series of analysis countermeasures to verify that the application is not being executed within a sandboxed environment or run under a debugger. Following these checks, the sample patches `AmsiScanBuffer` and `EtwEventWrite` to prevent the triggering of any unwanted events by the system.

Once the preliminary checks are completed, the sample proceeds to gather information about the system. The list of capabilities is hardcoded and cannot be modified from the server side, in contrast to the Python version. What is more, the feature list is quite similar to the Python version except a few ones.

RDP connections

The stealer is capable of collecting information about known RDP connections that the compromised user has. To achieve this, it searches for .rdp files in `%USERPROFILE%\Documents` and extracts the full server address, password, username and server port.

Gaming files

The stealer also targets gamers and is capable to steal credentials from the popular gaming platform clients, including:

- Steam
- Epic Games Launcher
- net
- Riot
- Origin
- Unreal Engine
- Ubisoft Connect
- GOG

Screenshots

The native version, unlike its Python counterpart, is capable of capturing screenshots for each monitor via `capCreateCaptureWindowA WinAPI`.
In conclusion, this sample communicates with the C2 server through the same endpoints as the Python version. However, in this instance, all data is encrypted using the same AES-GCM + PBKDF2 (HMAC and SHA256) scheme as partially employed in the Python variant. In some observed samples, the key used was `arkanix_secret_key_v20_2024`. Alongside that, the C++ sample explicitly sets the User-Agent to `ArkanixStealer/1.0`.

Post-exploitation browser data extractor

MD5	3283f8c54a3ddf0bc0d4111cc1f950c0
File name	–

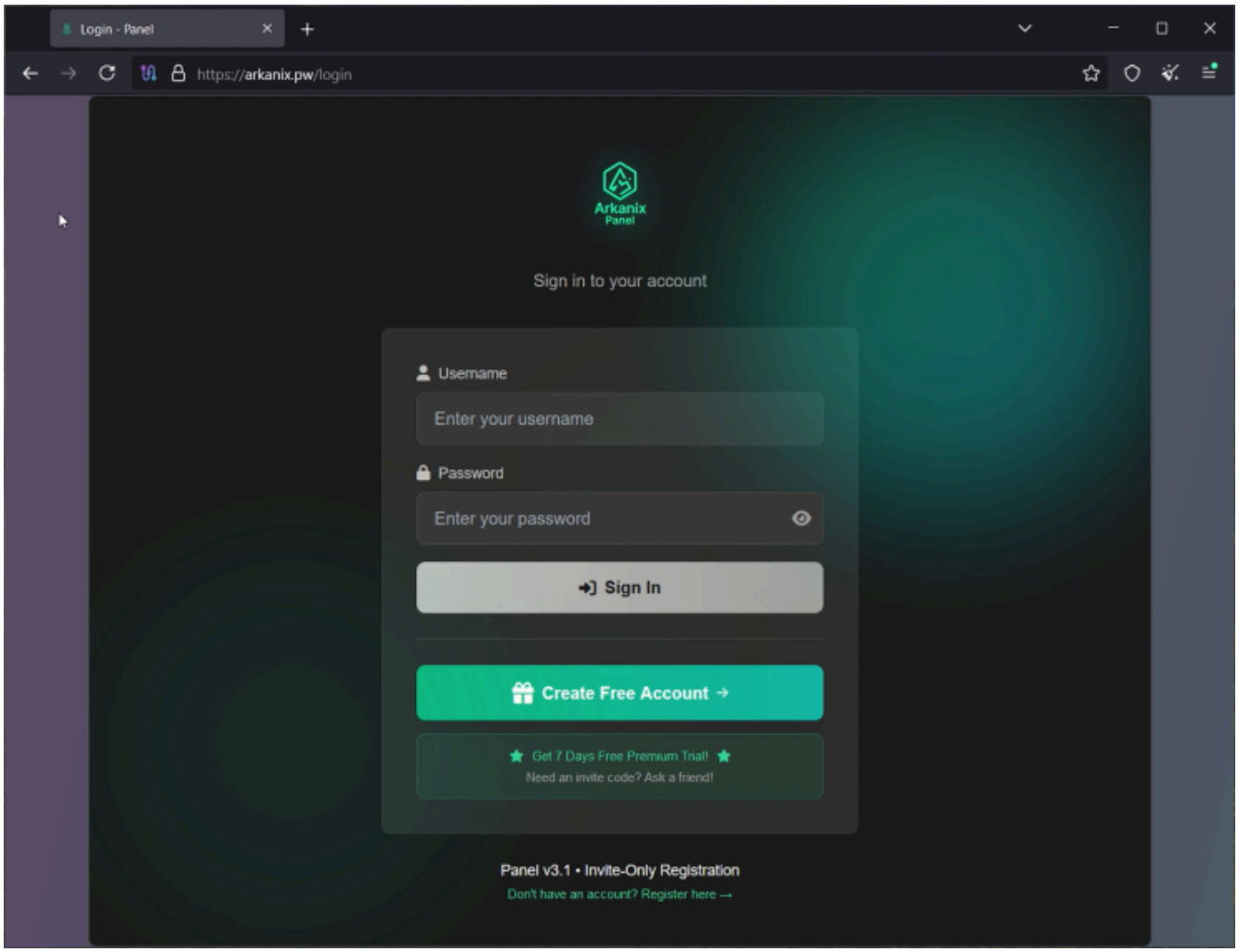
This is an implant embedded within the resources of the C++ implementation. The author incorporated it into the resource section without applying any obfuscation or encryption. Subsequently, the stealer extracts the payload to a temporary folder with a randomly generated name composed of hexadecimal digits (0-9 and A-F) and executes it using the `CreateProcess WinAPI`. The payload itself is the unaltered publicly available project known as “ChromElevator”. To summarize, this tool consists of two components: an injector and the main payload. The injector initializes a direct syscall engine, spawns a suspended target browser process, and injects the decrypted code into it via Nt syscalls. The injected payload then decrypts the browser master key and exfiltrates data such as cookies, login information, web data, and so on.

Infrastructure

During the Arkanix campaign, two domains used in the attacks were identified. Although these domains were routed through Cloudflare, a real IP address was successfully discovered for one of them, namely, `arkanix[.]pw`. For the second one we only obtained a Cloudflare IP address.

Domain	IP	First seen	ASN
arkanix[.]pw	195.246.231[.]60	Oct 09, 2025	–
arkanix[.]ru	172.67.186[.]193	Oct 19, 2025	–

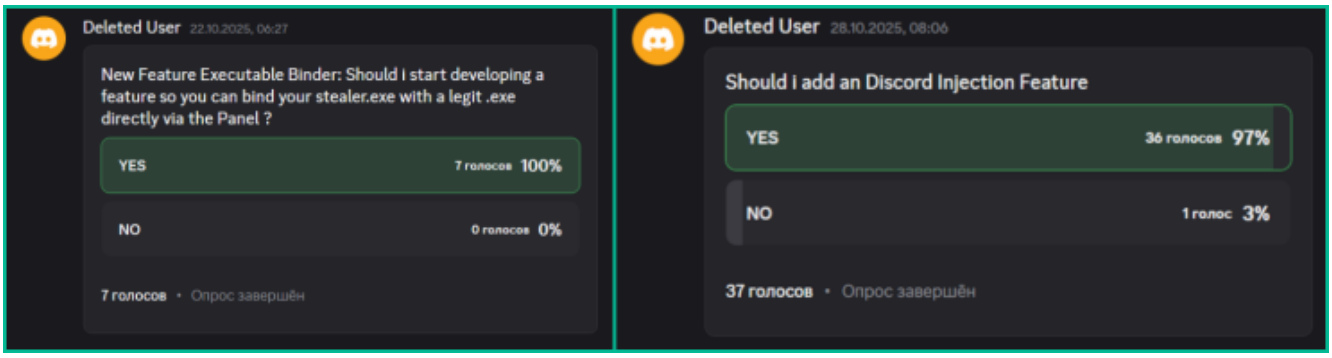
Both servers were also utilized to host the stealer panel, which allows attackers to monitor their victims. The contents of the panel are secured behind a sign-in page. Closer to the end of our research, the panel was seemingly taken down with no message or notice.



Stealer panel sign-in page

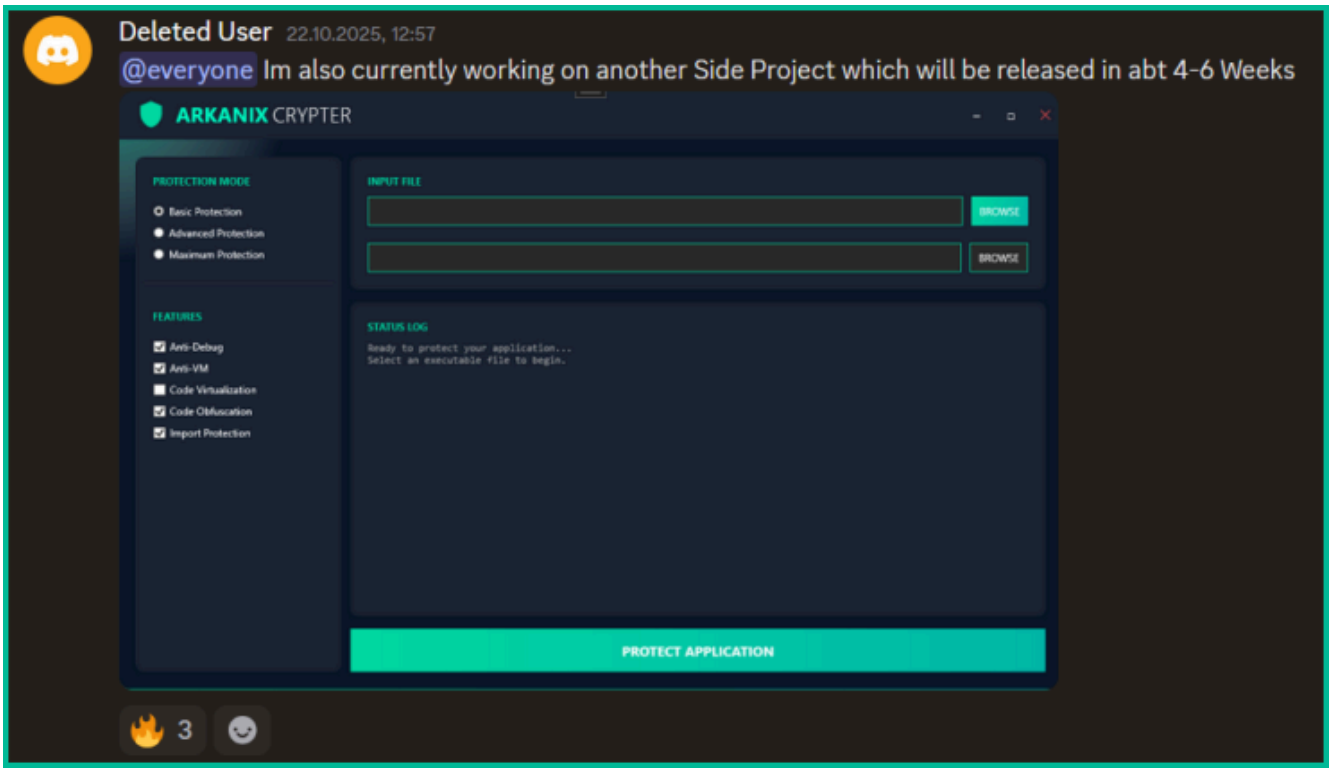
Stealer promotion

During the research of this campaign, we noticed that the forum posts advertising the stealer contained a link leading to a Discord server dubbed “Arkanix” by the authors. The server posed as a forum where authors posted various content and clients could ask various questions regarding this malicious software. While users mainly thank and ask about when the feature promised by the authors will be released and added into the stealer, the content made by the authors is broader. The adversary builds up the communication with potential buyers using the same marketing and communication methods real companies employ. To begin with, they warm up the audience by posting surveys about whether they should implement specific features, such as Discord injection and binding with a legitimate application (sic!).



Feature votes

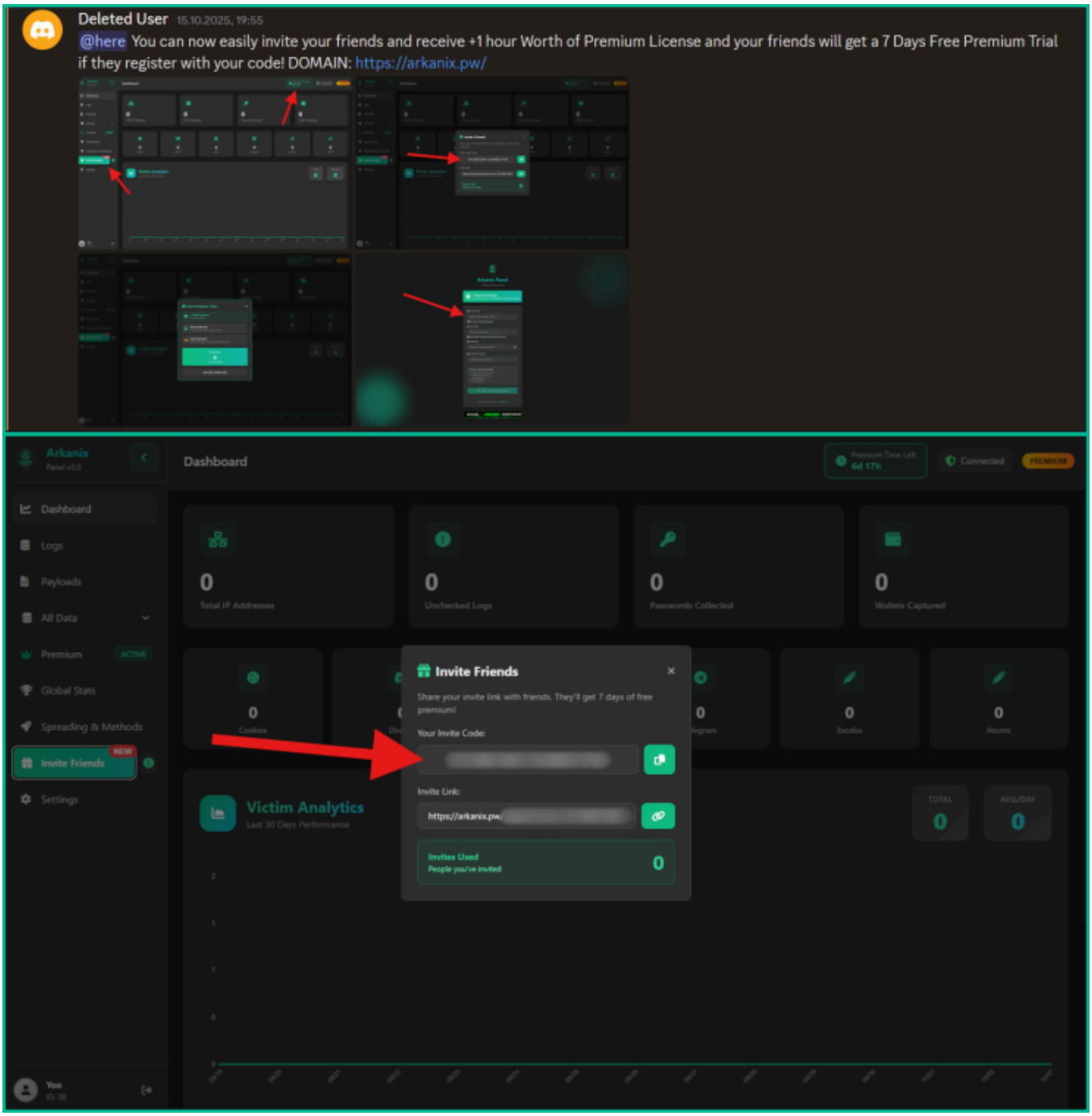
Additionally, the author promised to release a crypter as a side project in four to six weeks, at the end of October. As of now, the stealer seems to have been taken down without any notice while the crypter was never released.



Arkanix Crypter

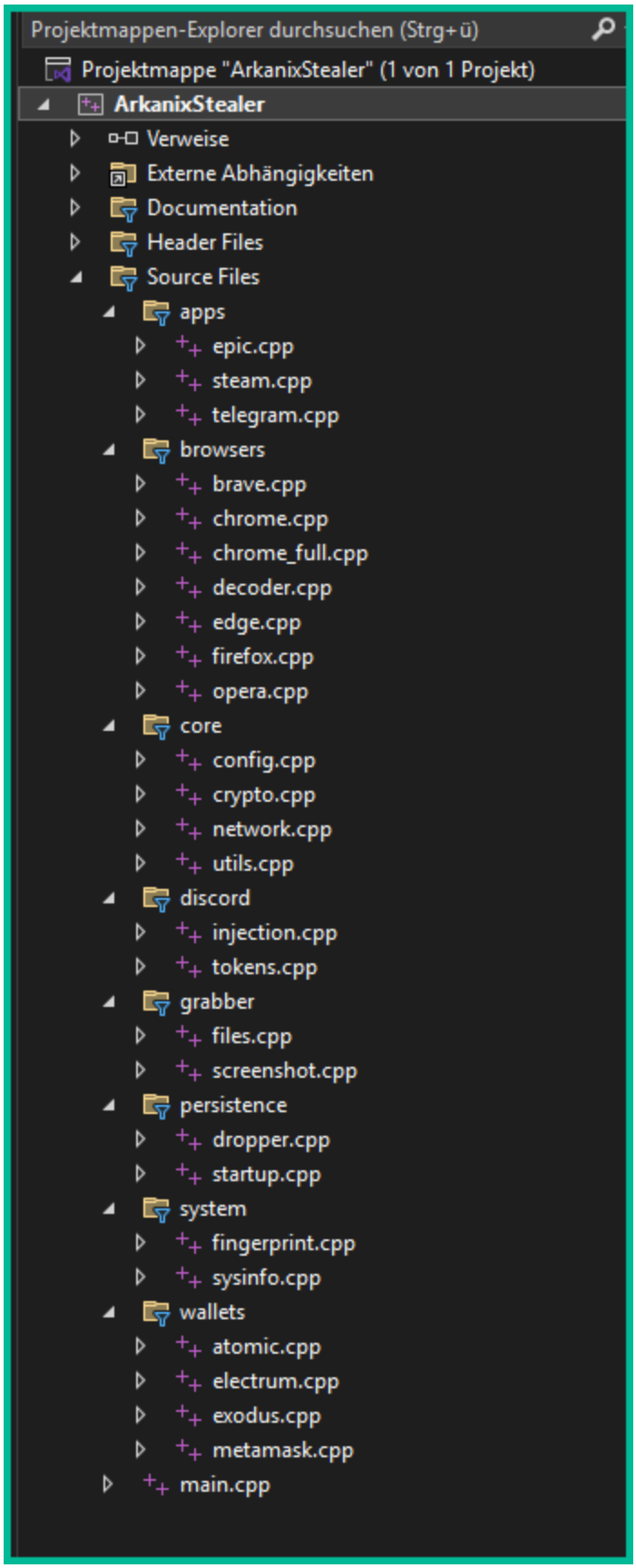
Furthermore, the Arkanix Stealer authors decided to implement a referral program to attract new customers. Referrers were promised an additional free hour to their premium license, while invited customers received seven days of free “premium” trial use. As stated in forum posts, the premium plan included the following features:

- C++ native stealer
- Exodus and Atomic cryptocurrency wallets injection
- Increased payload generation, up to 10 payloads
- Priority support

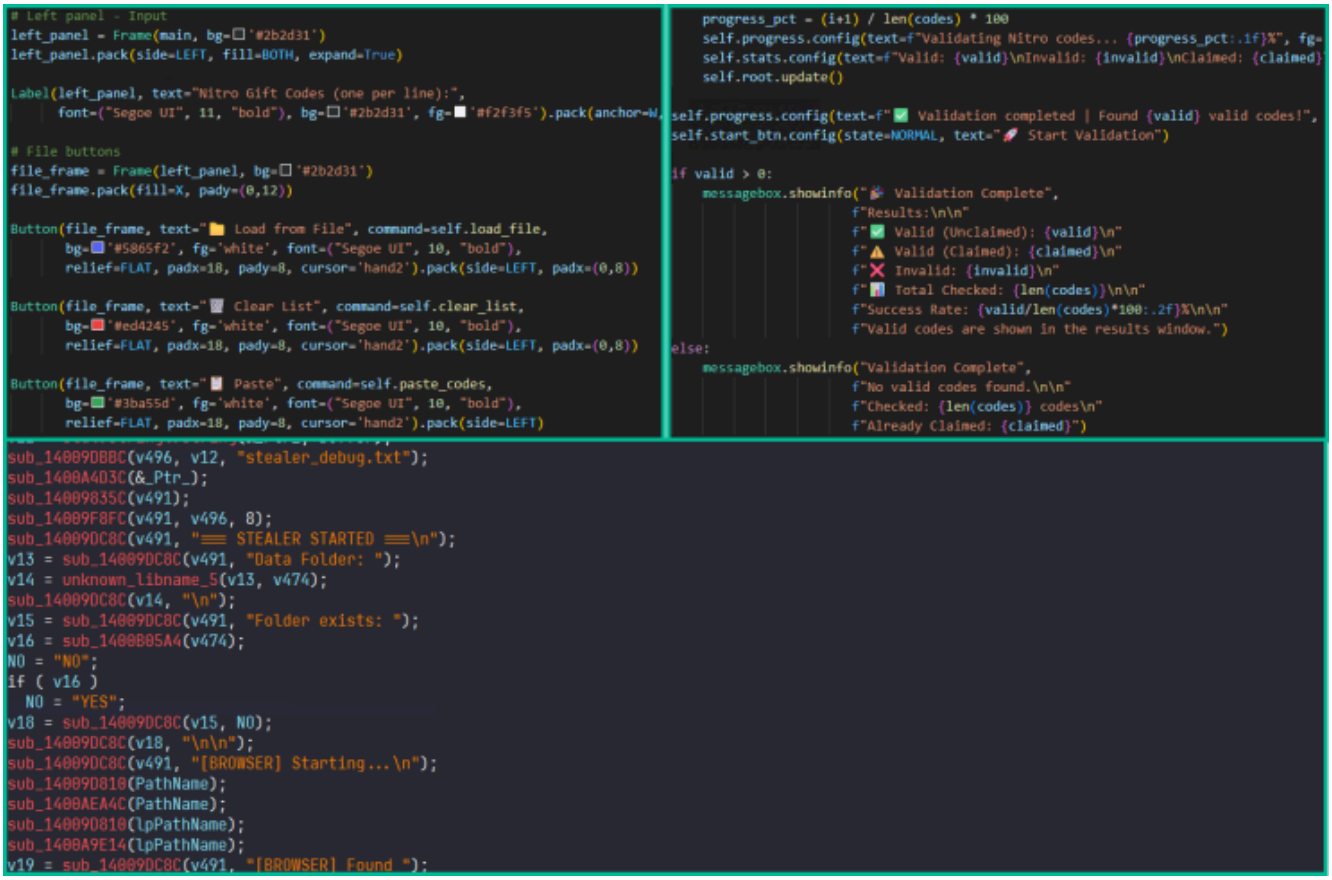


Referral program ad and corresponding panel interface

Speaking of technical details, based on the screenshot of the Visual Studio stealer project that was sent to the Discord server, we can conclude that the author is German-speaking.


Projektmappen-Explorer durchsuchen (Strg+ü)
Projektmappe "ArkanixStealer" (1 von 1 Projekt)
ArkanixStealer
Verweise
Externe Abhängigkeiten
Documentation
Header Files
Source Files
apps
epic.cpp
steam.cpp
telegram.cpp
browsers
brave.cpp
chrome.cpp
chrome_full.cpp
decoder.cpp
edge.cpp
firefox.cpp
opera.cpp
core
config.cpp
crypto.cpp
network.cpp
utils.cpp
discord
injection.cpp
tokens.cpp
grabber
files.cpp
screenshot.cpp
persistence
dropper.cpp
startup.cpp
system
fingerprint.cpp
sysinfo.cpp
wallets
atomic.cpp
electrum.cpp
exodus.cpp
metamask.cpp
main.cpp

This same screenshot also serves as a probable indicator of AI-assisted development as it shares the common patterns of such assistants, e.g. the presence of the utils.cpp file. What provides even more confidence is the overall code structure, the presence of comments and extensive debugging log output.



```
# Left panel - Input
left_panel = Frame(main, bg='#2b2d31')
left_panel.pack(side=LEFT, fill=BOTH, expand=True)

Label(left_panel, text="Nitro Gift Codes (one per line):",
      font=("Segoe UI", 11, "bold"), bg='#2b2d31', fg='#f2f3f5').pack(anchor=W,

# File buttons
file_frame = Frame(left_panel, bg='#2b2d31')
file_frame.pack(fill=X, pady=(0,12))

Button(file_frame, text="📁 Load from File", command=self.load_file,
       bg='#5865f2', fg='white', font=("Segoe UI", 10, "bold"),
       relief=FLAT, padx=18, pady=8, cursor='hand2').pack(side=LEFT, padx=(0,8))

Button(file_frame, text="🗑 Clear List", command=self.clear_list,
       bg='#ed4245', fg='white', font=("Segoe UI", 10, "bold"),
       relief=FLAT, padx=18, pady=8, cursor='hand2').pack(side=LEFT, padx=(0,8))

Button(file_frame, text="📋 Paste", command=self.paste_codes,
       bg='#3ba55d', fg='white', font=("Segoe UI", 10, "bold"),
       relief=FLAT, padx=18, pady=8, cursor='hand2').pack(side=LEFT)
```

```
    progress_pct = (i+1) / len(codes) * 100
    self.progress.config(text=f"Validating Nitro codes... {progress_pct:.1f}%", fg=
    self.stats.config(text=f"Valid: {valid}\nInvalid: {invalid}\nClaimed: {claimed}
    self.root.update()

self.progress.config(text=f"✅ Validation completed | Found {valid} valid codes!",
self.start_btn.config(state=NORMAL, text="🚀 Start Validation")

if valid > 0:
    messagebox.showinfo("🎉 Validation Complete",
                        f"Results:\n\n"
                        f"✅ Valid (Unclaimed): {valid}\n"
                        f"⚠️ Valid (Claimed): {claimed}\n"
                        f"❌ Invalid: {invalid}\n"
                        f"📊 Total Checked: {len(codes)}\n\n"
                        f"Success Rate: {valid/len(codes)*100:.2f}%\n\n"
                        f"Valid codes are shown in the results window.")
else:
    messagebox.showinfo("Validation Complete",
                        f"No valid codes found.\n\n"
                        f"Checked: {len(codes)} codes\n"
                        f"Already Claimed: {claimed}")
```

```
sub_14009DBBC(v496, v12, "stealer_debug.txt");
sub_1400A4D3C(&_Ptr_);
sub_14009835C(v491);
sub_14009F8FC(v491, v496, 8);
sub_14009DC8C(v491, "=== STEALER STARTED ===\n");
v13 = sub_14009DC8C(v491, "Data Folder: ");
v14 = unknown_libname_5(v13, v474);
sub_14009DC8C(v14, "\n");
v15 = sub_14009DC8C(v491, "Folder exists: ");
v16 = sub_1400B05A4(v474);
NO = "NO";
if ( v16 )
  NO = "YES";
v18 = sub_14009DC8C(v15, NO);
sub_14009DC8C(v18, "\n\n");
sub_14009DC8C(v491, "[BROWSER] Starting...\n");
sub_14009D810(PathName);
sub_1400AEA4C(PathName);
sub_14009D810(lpPathName);
sub_1400A9E14(lpPathName);
v19 = sub_14009DC8C(v491, "[BROWSER] Found ");
```

Example of LLM-specific patterns

Conclusions

Information stealers have always posed as a serious threat to users' data. Arkanix is no exception as it targets a wide range of users, from those interested in cryptocurrencies and gaming to those using online banking. It collects a vast amount of information including highly sensitive personal data. While being quite functional, it contains probable traces of LLM-assisted development which suggests that such assistance might have drastically reduced development time and costs. Hence it follows that this campaign tends to be more of a one-shot campaign for quick financial gains rather than a long-running infection. The panel and the Discord chat were taken down around December 2025, leaving no message or traces of further development or a resurgence.

In addition, the developers behind the Arkanix Stealer decided to address the public, implementing a forum where they posted development insights, conducted surveys and even ran a referral program where you could get bonuses for "bringing a friend". This behavior makes Arkanix more of a public software product than a shady stealer.

Indicators of Compromise

Additional IoCs are available to customers of our Threat Intelligence Reporting service. For more details, contact us at crimewareintel@kaspersky.com.

File hashes

752e3eb5a9c295ee285205fb39b67fc4
c1e4be64f80bc019651f84ef852dfa6c
a8eeda4ae7db3357ed2ee0d94b963eff
c0c04df98b7d1ca9e8c08dd1ffbdd16b
88487ab7a666081721e1dd1999fb9fb2
d42ba771541893eb047a0e835bd4f84e
5f71b83ca752cb128b67dbb1832205a4
208fa7e01f72a50334f3d7607f6b82bf
e27edcdeb44522a9036f5e4cd23f1f0c
ea50282fa1269836a7e87eddb10f95f7
643696a052ea1963e24cfb0531169477
f5765930205719c2ac9d2e26c3b03d8d
576de7a075637122f47d02d4288e3dd6
7888eb4f51413d9382e2b992b667d9f5
3283f8c54a3ddf0bc0d4111cc1f950c0

Domains and IPs

arkanix[.]pw
arkanix[.]ru

- Malware Technologies
- Malware Descriptions
- Malware
- Trojan
- Data theft
- Trojan-stealer
- Python
- Malware-as-a-Service
- Infostealers
- Arkanix Stealer

Arkanix Stealer: a C++ & Python infostealer

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

Name *

Email *

This site uses Akismet to reduce spam. Learn how your comment data is processed.