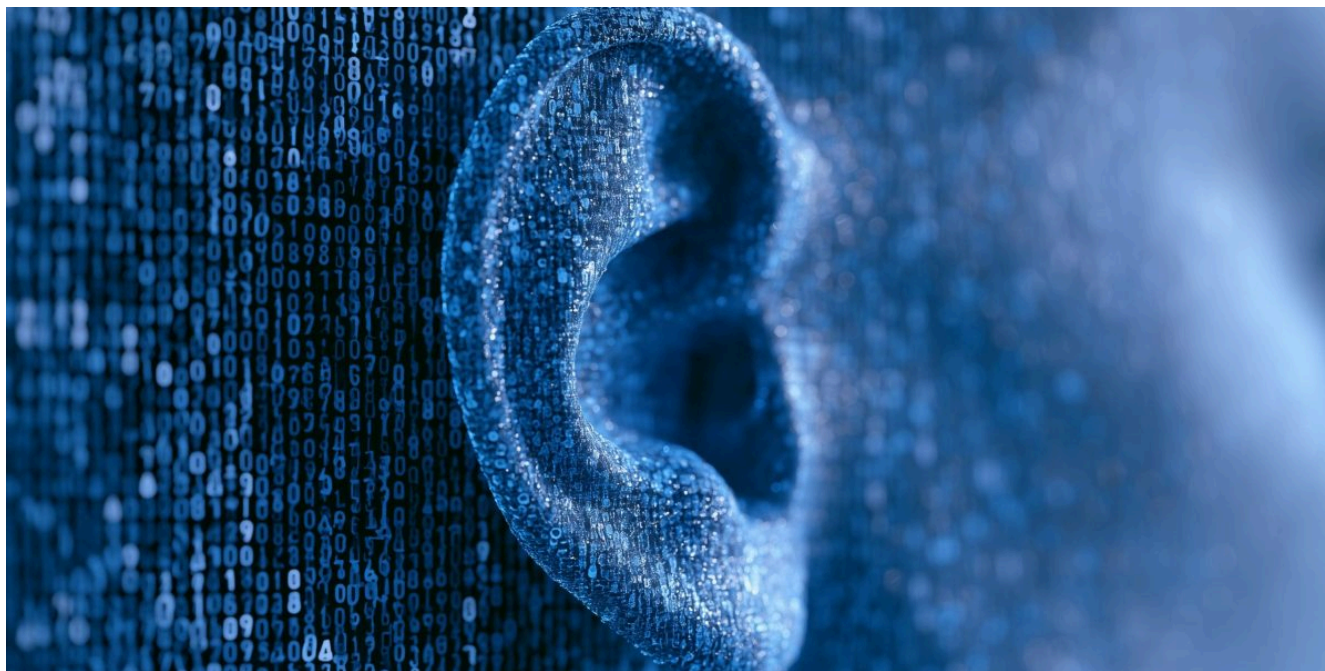


New Armored Likho tools target Telegram and eavesdropping

SL securelist.com/armored-likho-still-toolkit/121033

Konstantin Isakov

August 13, 2026



In May 2026, we discovered a new cyber-espionage campaign by the Armored Likho group, also known as Eagle Werewolf, that targets private individuals and organizations across various industries in Russia, including major corporations, the public sector, IT, and education. The attackers used a fake app as bait that mimics a service for donations. However, the most interesting part of this campaign isn't the initial infection method – it's the malicious implants the attackers use for cyber-espionage.

We've [written](#) previously about recent Armored Likho attacks, but our analysis shows that the campaign discussed below has more in common with [the group's activity from February](#). That said, the attackers have significantly expanded their arsenal.

During our research, we found a new cyber-espionage toolkit written in Rust: the **Still Toolkit**. One of its components, **Still Sync**, steals Telegram session data to gain ongoing access to the victim's account. With this stolen data, attackers can leverage the Telegram API to automatically pull chat logs, media files, and other information from the account.

The second component, **Still Audio**, is an implant for covert audio surveillance. It analyzes the incoming audio stream, automatically detects speech, records conversations, and sends the recordings to a command-and-control server.

In this article, we'll look at the initial infection method, how the new Still Toolkit components are built, and the technical details of how they operate.

Kaspersky products detect this threat as Trojan.Win64.Agent.* and HEUR:Backdoor.Win32.Generic.

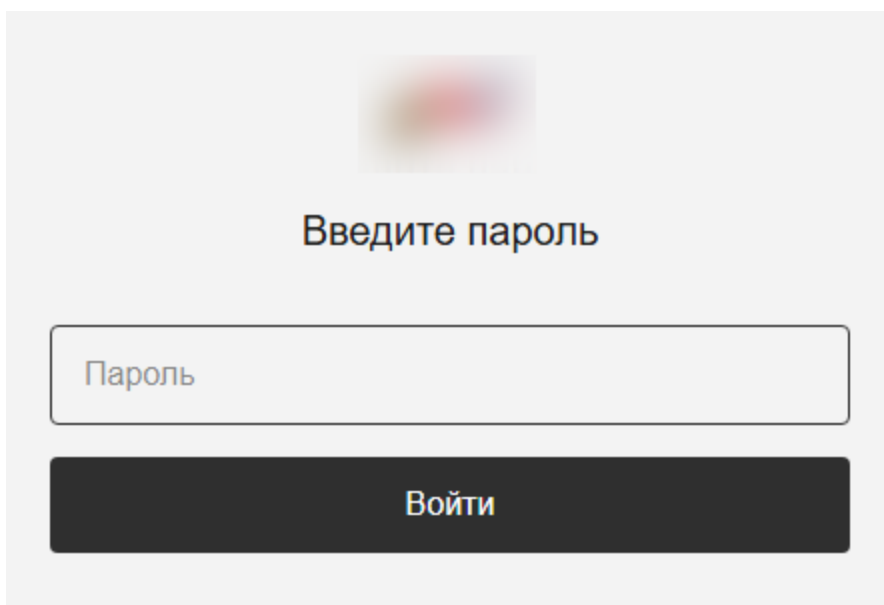
Background

Armored Likho's malicious activity has been documented several times before: in [November 2024](#), and in [February](#) and [July 2026](#). The current campaign shows significant overlap with the November and February campaigns, which used malicious droppers disguised as documents and applications related to Starlink activation or fundraising efforts as the initial infection vector. This campaign also uses fundraising as its lure. At the same time, our research uncovered a number of new tools that point to the attackers expanding their capabilities.

Initial infection

The infection chain starts with an app that mimics a donation service. As of this writing, the app distribution method remains unknown. During our research, however, we obtained several samples posing as apps from different Russian foundations.

In reality, the app is a dropper. Its developers wrote it in Rust on top of the popular Tauri framework, and it has a graphical interface designed to deceive the user. After launch, it displays a login form that asks for a password, presumably one the attackers supplied.



The image shows a login form with a light gray background. At the top center is a blurred logo. Below it is the text "Введите пароль" in a bold, black font. Underneath is a white rectangular input field with a thin gray border, containing the placeholder text "Пароль". At the bottom is a dark gray rectangular button with the white text "Войти".

The login form

After the user enters a valid password, they see a catalog of donatable items. The app pulls item and category information from `orderapiserver[.]info` through the `public/categories` and `public/products` endpoints. A clickable catalog makes the app look legitimate. While the user browses the items, the dropper quietly decrypts and launches the payload for the next stage in the background.

Our analysis shows that the mechanism for decrypting the payload and launching subsequent stages hasn't changed since the February campaign. However, we found a new cyber-espionage toolkit – the Still Toolkit – made up of two components: Still Sync and Still Audio.

Still Sync

Still Sync is a stealer written in Rust that steals Telegram session data. However, its capabilities don't stop there. With this stolen data, Sync can log in to the victim's account and pull messages and media files through the Telegram API.

Architecturally, Sync is an asynchronous application based on the [Tokio](#) library. It talks to the server over [gRPC](#) and serializes messages with [FlatBuffers](#). It supports both HTTP and HTTPS as transport protocols; the URL of the command-and-control server determines which one it uses.

How it works

When Sync launches, the attackers set several environment variables. Before starting any malicious activity, the implant pulls configuration parameters from these:

- **STILL_SYNC_ADDR**: the address of the command-and-control server. By default, this is `https://tg4service[.]com:443`.
- **STILL_SEND_PATH**: the path to the `tdata`
- **STILL_TELEGRAM_PASSCODE**: the password for decrypting the `tdata` folder, if Telegram data encryption is enabled on the victim's device.

Sync also supports several command-line arguments:

- `--console`: runs as a console application. If this parameter is absent, the implant creates a **TReload** service to keep running in the background.
- `--version`: prints version information and exits.
- `--firefly`: launches a trace thread that monitors the program's operation. It writes error messages to a hidden file, `bin`, located in the same folder as the main executable.
- `--db`: turns on debug mode with detailed logging.

```

C:\Users\user\Desktop>winapi.exe --db --console
2026-06-21T12:03:15.470647Z INFO TReload: service starting; sync_addr='https://tg4service.com:443'
2026-06-21T12:03:15.470847Z DEBUG still_core::system_info: generate_unique_id_v22
2026-06-21T12:03:15.471005Z DEBUG still_core::system_info: get_computer_name
2026-06-21T12:03:15.471061Z DEBUG still_core::system_info: Computer name: 
2026-06-21T12:03:15.471104Z DEBUG still_core::system_info: get_bios_serial
2026-06-21T12:03:17.340735Z DEBUG still_core::system_info: get_system_uuid
2026-06-21T12:03:18.467354Z DEBUG still_core::system_info: get_processor_id
2026-06-21T12:03:19.587875Z DEBUG still_core::system_info: get_motherboard_serial
2026-06-21T12:03:20.673387Z DEBUG still_core::system_info: compute_unique_id_v22
2026-06-21T12:03:20.673549Z DEBUG still_core::system_info: Unique machine id: 
2026-06-21T12:03:20.674330Z DEBUG still_core::system_info: get_bios_serial
2026-06-21T12:03:21.745474Z DEBUG still_core::system_info: get_system_uuid
2026-06-21T12:03:22.824845Z DEBUG still_core::system_info: get_processor_id
2026-06-21T12:03:23.942715Z DEBUG still_core::system_info: get_motherboard_serial
2026-06-21T12:03:25.090449Z WARN TReload: initial register_machine failed: at still-sc/src/rpc.rs line 33 column 35

Caused by:
  0: transport error
  1: dns error
  2: dns error
  3: No such host is known. (os error 11001)
2026-06-21T12:03:25.092464Z DEBUG TReload: register loop sleeping for 569s
2026-06-21T12:03:25.092540Z INFO TReload: send loop sleeping for 696s
2026-06-21T12:03:25.092633Z DEBUG still_core::system_info: generate_unique_id_v22

```

Example Still Sync logs

Once it launches, the malware begins registering the device with the C2 server. To do this, Sync collects the following information about the victim's system:

- Motherboard serial number
- CPU ID
- System UUID
- BIOS serial number
- Computer domain name

The malware combines the collected data into a single string with a colon as the separator. It then hashes that string with SHA-256 and stores the resulting hash under the key `sysmarker`. Worth noting: other Armored Likho tools, [AquilaRAT](#) included, use this same hashing algorithm.

Sync then serializes a package containing all the collected information and the agent version, and sends it in a POST request to `/still.rpc.Sync/RegisterMachine`. The response contains a `machine_id` value, which Sync uses to identify itself in subsequent requests.

Once registration succeeds, Sync sends a POST request with the `machine_id` parameter to `/still.rpc.Sync/GetMachineSettings`. The server responds with the following settings:

- **enabled:** triggers malicious activity on the infected device.
- **scan_portable:** turns on extended scanning when searching for the tdata We'll cover this feature in more detail below.
- **fetch_telegram:** if this parameter is on, Sync attempts to log in to Telegram and extract data. We'll cover this feature in more detail below.
- **download_channels:** if this parameter is off, Sync skips channel dialogs when exfiltrating Telegram data.

These parameters have no default values, so Sync doesn't perform any malicious actions until the registration and settings-retrieval processes both complete successfully.

Telegram data collection

Before stealing a Telegram session, Sync searches for the `tdata` folder, unless the **STILL_SEND_PATH** variable is already set. The list of search paths includes both standard and nonstandard directories, if the **scan_portable** option is turned on:

- `C:\Users\<username>\AppData\Roaming\Telegram Desktop\`: the standard Telegram Desktop installation directory.
- `C:\Users\<username>\AppData\Local\Packages\<package_folder>\LocalCache\Roaming\`: the installation directory for the Microsoft Store version. Sync identifies the package folder by a name that contains the string `TelegramMesse`.
- `C:\`: used for the extended search (if the **scan_portable** option is on).

Sync then sends a POST request with a list of files from the `tdata` folder to the `/still.rpc.Sync/CheckFiles` endpoint. The server responds with the following values:

- **snapshot_id**: an identifier the server assigns to the current data snapshot.
- **present**: a list of file paths that are already present on the server.

This lets the C2 server avoid re-receiving files it already has. In addition, if Sync can't access files on disk through standard methods, it falls back on three mechanisms that abuse the **SeBackupPrivilege** privilege:

- Opening files with the `CreateFileW` function using the `FILE_FLAG_BACKUP_SEMANTICS` parameter
- Creating a backup copy through the Shadow Copy service and reading files from there
- If the previous methods all fail, attempting to copy the file using the Robocopy utility in backup mode

Beyond stealing Telegram session data, Sync can carry out full-scale collection of user information from the messaging app. When the **fetch_telegram** option is on, it launches a separate thread that authenticates to the chat app using the previously obtained `tdata`. Once authentication succeeds, Sync gains access to the account data and sends the following collected information to the server:

- User details, such as username, phone number, first and last name
- Information about private chats, groups, or channels, such as chat name and ID, the member list, and so on

- Dialogs from private chats, groups, and channels (if the **download_channels** option is on)
- Media files under 250MB: photos, documents, stickers, and contacts

Still Audio

Still Audio is an audio surveillance implant written in Rust. Its main job is to analyze the incoming audio stream and start recording voice when certain conditions are met – we'll cover those in the next section. Architecturally, Still Audio largely mirrors Sync and uses the same mechanisms for communicating with the C2 server.

On launch, **Still Audio** performs a sequence of actions:

- It extracts `libmp3lame.dll`, a file stored inside the executable. This is a library used to encode audio data.
- If the `--console` command-line argument is absent, the implant creates a service named `auxhost`, connects to it, and continues running in the background.
- While running in the background, it creates a file, `logfile.log`, to write logs to.

Next, Still Audio retrieves the C2 server address. As with Sync, it stores the URL in an environment variable – in this case, **STILL_AUDIO_SYNC_ADDR**. If that variable isn't set, it falls back to **STILL_SYNC_ADDR**, which shows the two modules are compatible with each other. If neither variable is set, it uses the default URL, `https://srwinservice[.]com`.

Still Audio also uses the [Dead Drop Resolver](#) technique as a fallback mechanism for obtaining the C2 address. If the current server stays unreachable for three days, the tool tries to pull the current C2 URL from a GitHub repository. In the sample under analysis, we found the following URL for the page containing C2 information:

```
hxxps://raw.githubusercontent[.]com/mmarln/pi-  
mono/refs/heads/main/packages/pods/src/array12.json
```

A screenshot of a GitHub repository file view. The breadcrumb path at the top is 'pi-mono / packages / pods / src / array12.json'. Below the path is a commit by user 'mmarln' with the message 'Create array12.json'. There are two tabs: 'Code' (selected) and 'Blame'. The file statistics show '3 lines (3 loc) · 41 Bytes'. The code content is a JSON array with three lines: line 1 is '[', line 2 is ' "7XyqWEYGjegzc1nKSQAFZqSpRRFuLL0r"', and line 3 is ']'.

Encrypted C2 address inside the GitHub repository

The repository, a fork of a popular project, contains the server URL Base64-encoded and encrypted with the Blowfish algorithm in ECB mode, using the key 5c8e153228edd3c6cbf75684 (lowercase string). Older AquilaRAT samples use this exact same algorithm and key.

Once it obtains the current C2 address, the Audio module starts a registration process similar to Sync's, but through a different endpoint:

/still.rpc.Audio/RegisterAudioMachine. Also, unlike Sync, Audio sends a list of available audio input devices along with the system information.

The server responds with settings for the implant:

- **machine_id**: a unique identifier for the current device.
- **vad_threshold**: the threshold value for the VAD (Voice Activity Detection) algorithm. Expressed as a decimal fraction, it represents a proportion of the maximum sound level the input device can pick up. Sound above this threshold counts as voice activity. The default **vad_threshold** is **02**.
- **max_silence_duration**: the number of audio samples with a VAD value below the set threshold after which the implant considers the recording finished.
- **max_buffer_size**: the maximum buffer size for recorded audio data.
- **active_device**: the name of the input device selected for recording, from the list of available devices.

The eavesdropping process

Still Audio works with raw audio samples it captures directly from the input device. To detect voice activity, it implements an algorithm based on Root Mean Square (RMS), a lightweight signal-processing method that distinguishes speech from silence by measuring the audio signal's average power over time. The implant doesn't rely on any third-party libraries here; it implements all the calculations itself.

The implant compares the calculated RMS value against the **vad_threshold** parameter. If RMS meets or exceeds this threshold, recording starts. To avoid losing the beginning of the recording, Still Audio uses a pre-buffer, a size-limited buffer that stores samples from just before the current recording moment. A sequence of **max_silence_duration** samples (**320** by default) with RMS values below the threshold signals the end of the recording. For example, with a standard headset running at a **44.1kHz** sampling rate, recording stops after roughly **7ms** of silence.

Interestingly, the Audio module makes no attempt to hide its use of the microphone: its name shows up in Windows settings. In the sample we examined, the file was saved to disk as `IntAudio.exe`, and it appeared in the list of apps using the microphone as "Intel Audio":

Allow desktop apps to access your microphone

Some apps and Windows features need to access your microphone to work as intended. Turning off this setting here might limit what desktop apps and Windows can do.



Some desktop apps might not appear in the following list or are not affected by this setting. [Find out why](#)

Sort by: Name ▾



Intel Audio
Currently in use

The malicious module in the list of apps using the microphone

Before sending recordings to the server, the implant uses the **libmp3lame** library to encode the raw audio samples. It sends the recording files via a POST request to `/tgfrg`, adding a

Client-Id header containing the **machine_id** obtained during registration to identify the device.

Infrastructure

This campaign draws on a broad set of hosting providers and domains registered at different points in time, which suggests the attackers are trying to make their infrastructure harder to detect. We found no direct overlap in domains or IP addresses with the February campaign. Even so, the two infrastructures share some similarities:

- They use the same hosting providers, with the ASNs **149440**, **202448**, and **215311**.
- Their domain names follow similar naming patterns that mimic Windows system services and update mechanisms.

Domain	IP address	Registration date	ASN
orderapiserver[.]info	187.127.153[.]38	April 18, 2026	47583
tg4service[.]com	159.198.37[.]74	October 4, 2025	22612
srwinservice[.]com	213.252.244[.]123	March 19, 2026	61272
screenserv[.]com	23.26.237[.]250	February 13, 2026	149440
windowsserv[.]net	23.27.24[.]30	February 10, 2026	149440
managementapiservice[.]com	188.212.124[.]178	May 1, 2026	202448
service8date[.]com	145.223.69[.]143	January 13, 2026	215311
updateservs[.]com	145.223.68[.]66	December 23, 2025	215311

Victims

In this campaign, we’ve determined that the attackers’ primary targets are users in Russia. Most victims are private individuals, though the corporate sector, government organizations, IT companies, and educational institutions are also affected.

Attribution

This campaign has been using both new tools and malware families documented in BI.ZONE’s February report. While some components turned up for the first time, they show

significant code-level overlap with malicious tools seen in earlier Armored Likho campaigns. Based on these overlaps, along with additional technical artifacts, we're highly confident the Armored Likho group is behind the campaign. The overlaps we identified include:

- **Identical dropper architecture** in the February and current campaigns, which includes the use of the Tauri library to build the graphical interface, a similar user-input handler, a payload with the ICRYPTMP header, and the same multi-part encryption format.
- **The same encryption algorithm and key** used in AquilaRAT from the previous campaign and in the Still Audio module from the current campaign, both implementing the Dead Drop Resolver technique.
- **Identical logic for generating** the sysmarker value in older AquilaRAT samples and in the Still toolkit from the current campaign. The algorithms match down to the PowerShell commands used to collect system information.
- **Substantial infrastructure overlap**, which includes the hosting providers and domain-naming patterns described in the Infrastructure section.

Takeaways

The campaign described in this post shows Armored Likho's toolkit evolving, with the group steadily expanding its cyber-espionage capabilities. Beyond the components we already knew about, the attackers rolled out new modules that let them not only access Telegram data but also conduct audio surveillance on victims. Together, these capabilities significantly widen the range of information attackers can collect in a single compromise.

One point deserves particular attention: the new tools form a cohesive set, sharing similar architecture, C2 communication mechanisms, and common implementation elements. This points to the group building out its own tool ecosystem, designed for long-term use and further expansion.

The emergence of new, specialized modules shows the attackers aren't just trying to preserve their existing capabilities – they're working to make intelligence-gathering more effective by controlling multiple communication channels at once.

Indicators of compromise

Additional information about this threat, indicators of compromise included, is available to customers of [Kaspersky Threat Intelligence Reporting](#). Contact intelreports@kaspersky.com for more details.

File hashes

Droppers

[C1D1EE16B92E6A138FFA048855F75D7D](#)
[17674B250D8B422A50A86C9FF207186D](#)
[62801F6223E860A7CCA271522E303B2D](#)

Still Sync

[68F0365D2FA8C828D012D8859E52A773](#)
[4BD7C352AE277B0E38D07BEEDD4DD507](#)
[D4BC09FB10EA2A5DC0BCBEEDA5E5AFDD](#)

Still Audio

[2CA8ADBAB98EBE305EACF272CF48F5A0](#)
[3AC41B097236A7723821848AE31EF141](#)
[439255736797BC88BD19F282449E0436](#)

Domains

[orderapiserver\[.\]info](#)
[tg4service\[.\]com](#)
[srwinservice\[.\]com](#)
[screenserv\[.\]com](#)
[window serv\[.\]net](#)
[managementapiservice\[.\]com](#)
[service8date\[.\]com](#)
[updateservs\[.\]com](#)

- [Malware Technologies](#)
- [Targeted attacks](#)
- [Malware Descriptions](#)
- [Spyware](#)
- [Malware](#)
- [APT](#)
- [Trojan](#)
- [Trojan-stealer](#)
- [Telegram](#)
- [Armored Likho](#)
- [Still toolkit](#)
- [Audio surveillance](#)