Sturnus: Mobile Banking Malware bypassing WhatsApp, Telegram and Signal Encryption

Published: 2024-10-01 · Archived: 2026-04-06 00:42:39 UTC

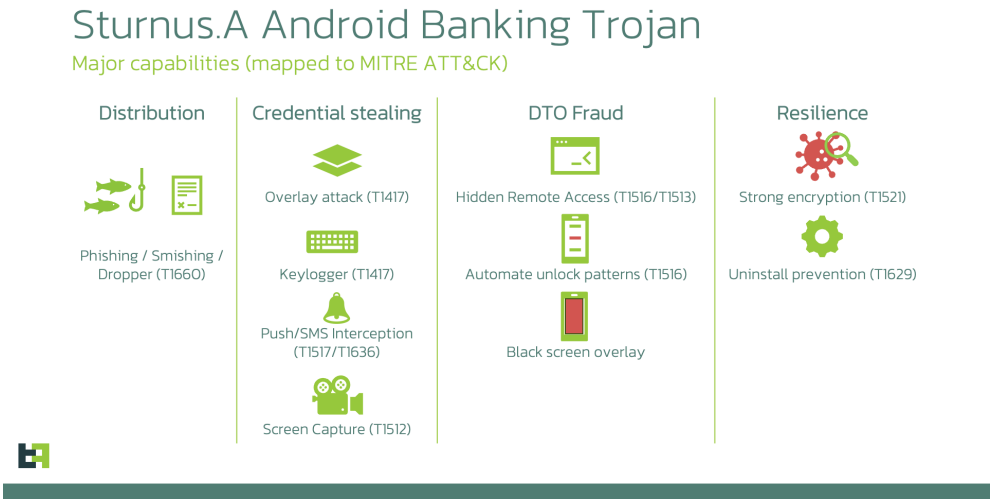
MTI Security researchers have identified Sturnus, a privately operated Android banking trojan. This malware supports a broad range of fraud-related capabilities, including full device takeover. A key differentiator is its ability to **bypass encrypted messaging**. By capturing content directly from the device screen after decryption, Sturnus can monitor communications via WhatsApp, Telegram, and Signal.

The trojan can harvest banking credentials through convincing fake login screens that replicate legitimate banking apps. In addition, it provides attackers with extensive remote control, enabling them to observe all user activity, inject text without physical interaction, and even black out the device screen while executing fraudulent transactions in the background—without the victim's knowledge.

An advanced threat in its early stages

While analysis indicates this operation is currently in a development or limited testing phase, Sturnus has already been configured with targeted attacks against financial institutions across Southern and Central Europe, suggesting preparations for a broader campaign. While we emphasize that the malware is likely in its pre-deployment state, it is also currently fully functional, and in aspects such as its communication protocol and device support, it is more advanced than current and more established malware families.

ThreatFabric mapped the capabilities of this new malware family according to the MITRE ATT&CK matrix, and you can find the corresponding techniques used:



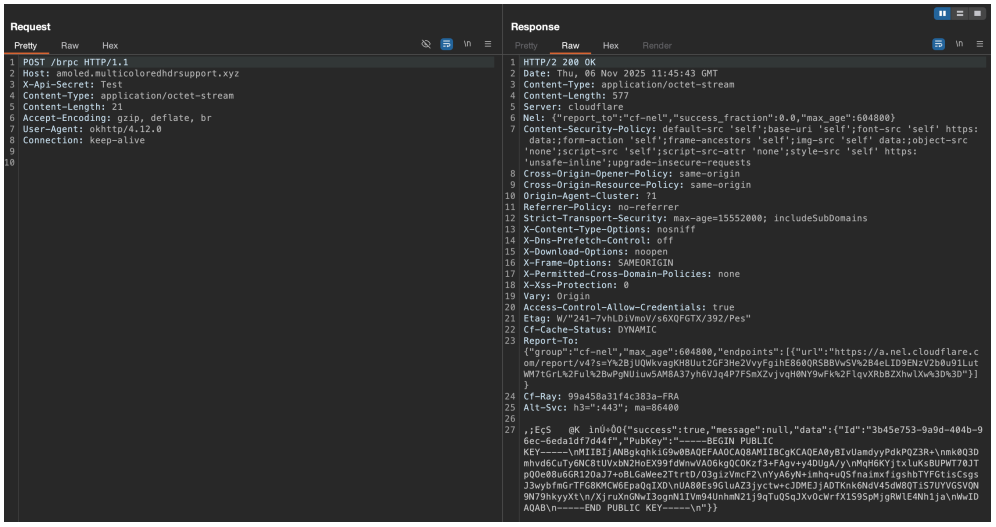
Target and Victimology

Current evidence indicates that Sturnus.A is still in an evaluation and tuning phase, with relatively few samples and short, intermittent campaigns rather than sustained large-scale activity. The victimology so far points to targets primarily located in Southern and Central Europe, where we have observed region-specific overlay templates. In parallel, the malware's behavior shows a clear focus on compromising widely used secure messaging platforms such as WhatsApp, Telegram, and Signal, suggesting that the operators are testing its ability to capture sensitive communications across different environments. Although the spread remains limited at this stage, the combination of targeted geography and high-value application focus implies that the attackers are refining their tooling ahead of broader or more coordinated operations.

The complex call of the Songbird

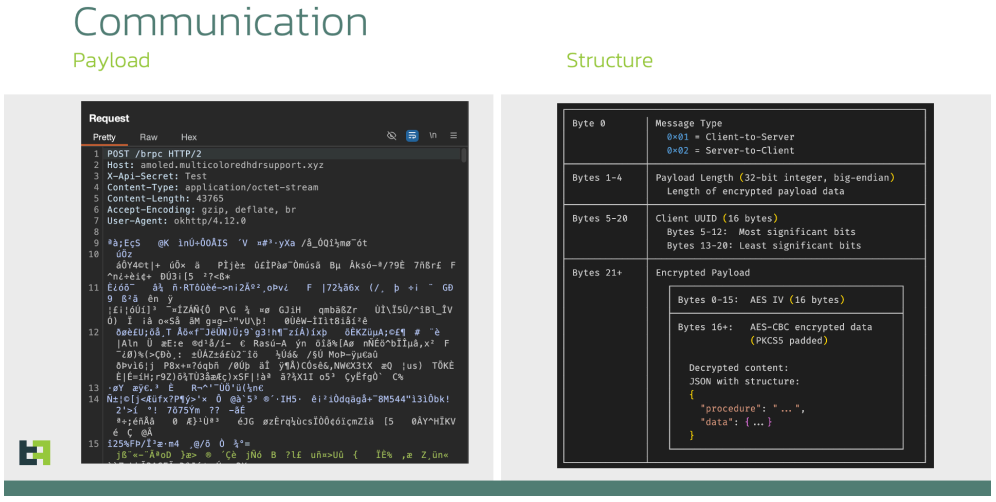
The malware's layered and slightly chaotic mix of plaintext, RSA, and AES communications—switching unpredictably between simple and complex messages—reminds us of the *Sturnus vulgaris*, whose rapid, irregular chatter jumps between whistles, clicks, and imitations. This noisy and intricate pattern inspired the malware's name.

The technical flow reflects this same structure: the malware communicates with its command-and-control server through both WebSocket (WSS) and HTTP channels, sending a combination of encrypted and plaintext data, with most plaintext appearing over WebSocket. Its protocol begins with an HTTP POST request to the endpoint to register the device using a placeholder payload. The server replies with a UUID (client ID) and an RSA public key. The malware then generates a 256-bit AES key locally, encrypts it using RSA/ECB/OAEPWithSHA-1AndMGF1Padding, and sends the encrypted key back, while storing the plaintext AES key on the device in Base64 form.



Once the key exchange is complete, all further communication is protected with AES encryption using **AES/CBC/PKCS5Padding** and a 256-bit key. For every message, the malware generates a fresh 16-byte initialization vector (IV), prepends it to the encrypted payload, and wraps the result in a custom binary protocol that includes a header with the message type, message length, and the client UUID. In parallel, the malware also establishes a WebSocket channel, which it uses to communicate directly with the bot during VNC sessions.

Here is the message structure, when encrypted:

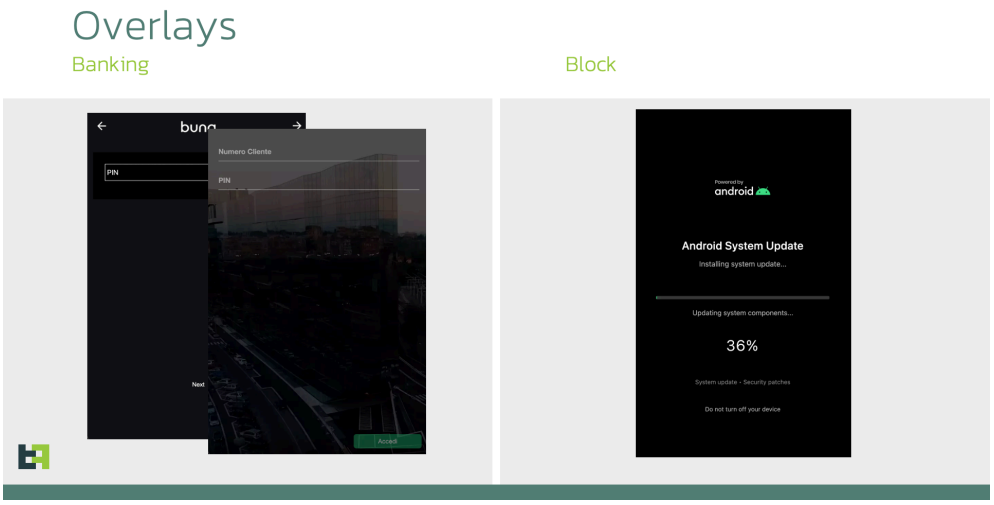


Capabilities

Data Exfiltration

The malware relies on two tightly integrated mechanisms—HTML overlays and accessibility-based keylogging—to capture and exfiltrate user credentials and other sensitive data. Its overlay engine maintains a persistent repository of phishing templates under `/data/user/0/<malware_package>/files/overlays/`, where each HTML file corresponds to a targeted banking application. When an overlay is triggered, the malware launches an activity containing a WebView configured with JavaScript, DOM storage, and a JavaScript bridge that intercepts and forwards any data the victim enters directly to the C2 server. Once the information is exfiltrated, the overlay for that specific target is disabled to reduce repetition and avoid suspicion.

The system also supports a full-screen “block overlay”, that allows attackers to hide their malicious activities from victims by displaying a full screen black overlay that blocks all visual feedback while the malware operates in the background .



Alongside overlays, the malware implements a comprehensive keylogging pipeline through the Android Accessibility Service. It processes events such as:

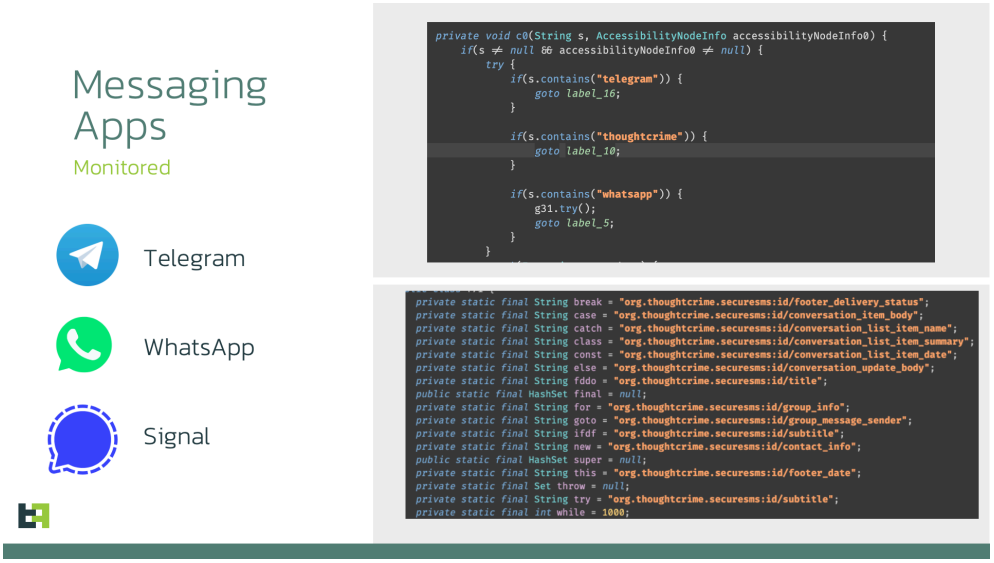
- TYPE_VIEW_TEXT_CHANGED
- TYPE_VIEW_FOCUSED
- TYPE_VIEW_CLICKED
- TYPE_WINDOW_CONTENT_CHANGED

to capture text as it is typed, track focus changes for context, and record UI interactions. Beyond simple keystroke logging, it continuously monitors the device’s UI tree and sends structured logs describing what is displayed on screen. This enables attackers to reconstruct full user activity even when screen capture is blocked by FLAG_SECURE or when network conditions prevent live video transmission. Together, these mechanisms give the operator a detailed, real-time picture of the victim’s actions while providing multiple redundant paths for data theft.

These capabilities are also used by the malware to programmatically steal PINs and Passwords to easily unlock the device.

Messaging Apps control

Sturnus, in addition to banking applications, also monitors the foreground app and automatically activates its UI-tree collection whenever the victim opens encrypted messaging services such as WhatsApp, Signal, or Telegram.



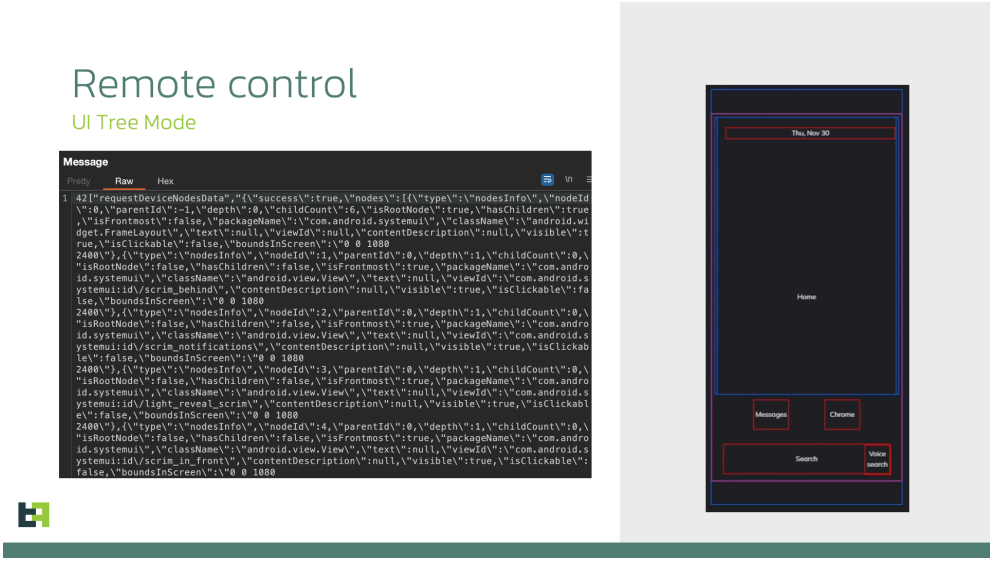
Because it relies on Accessibility Service logging rather than network interception, the malware can read everything that appears on screen—including contacts, full conversation threads, and the content of incoming and outgoing messages—in real time. This makes the capability particularly dangerous: it completely sidesteps end-to-end encryption by accessing messages after they are decrypted by the legitimate app, giving the attacker a direct view into supposedly private conversations. The user sees a secure interface, but from the moment the device is compromised, every sensitive exchange becomes visible to the operator, with no cryptographic protection left to rely on.

Remote Control

Sturnus supports full remote sessions, letting operators interact with the victim's device using two complementary screen-capture techniques that provide redundancy across Android versions and permission states. Its primary method relies on the system's display-capture framework to mirror the device screen in real time, while a fallback mechanism uses Accessibility-based screenshots when standard capture is blocked. Both approaches produce continuous visual streams that can be scaled, throttled, or reconfigured remotely depending on bandwidth and operational needs. Screen data is converted into a native framebuffer and encoded for transmission, enabling responsive remote interaction. Then, the management of the session is handed to a native library, which implements the VNC RFB protocol, encoding frames for transmission to connected clients.

In parallel with pixel-based streaming, Sturnus exposes a second, highly efficient control layer based entirely on Accessibility-derived UI information. Instead of sending images, it transmits structured descriptions of every visible interface element, allowing attackers to map the entire screen, understand its layout, and issue precise actions such as clicks, text input, scrolling, app launches, or permission confirmations. This method consumes minimal bandwidth, works without triggering screen-capture indicators that would normally appear, and remains fully operational when elements are off-screen or protected by security flags.

Together, the visual stream and the UI-tree control channel give attackers persistent, covert, and fine-grained control over an infected device. Here is a screenshot of data exfiltrated by Sturnus, and an example of how this data could look on the control panel of the criminals (example taken from another banking malware with the same feature):



Here is the full list of supported actions, which often accept parameters such as coordinates or node identifiers:

Value	Description
buttonAction	System navigation control
clickNode	Click specific UI element
textSender	Inject text into focused field
gensureScroll	Execute swipe/scroll gesture
lastClickedNode	Click multiple nodes in sequence
nodesInfo	Enumerate all UI elements
enableBlackScreen	Hide screen from victim
disableBlackScreen	Remove black screen overlay

Environment monitoring

Sturnus reinforces its persistence by securing Android Device Administrator privileges and actively defending them. Once granted, these privileges allow the malware to monitor password changes, unlock attempts, and lock-screen activity, sending each event to the command-and-control server with precise timestamps. The same privileges let it lock the device remotely and, more importantly, make itself significantly harder to remove. Whenever the user navigates to settings screens that could disable its administrator status, the malware detects the attempt through Accessibility monitoring, identifies relevant controls, and automatically navigates away from the page to interrupt the user. Until its administrator rights are manually revoked, both ordinary uninstallation and removal through tools like ADB are blocked, giving the malware strong protection against cleanup attempts.

```
➜  ~ adb shell pm list packages | grep com.hoquvxut.janoschae
package:com.hoquvxut.janoschae
➜  ~ adb uninstall com.hoquvxut.janoschae
Failure [DELETE_FAILED_DEVICE_POLICY_MANAGER]
➜  ~ adb shell pm disable-user com.hoquvxut.janoschae
Package com.hoquvxut.janoschae new state: disabled-user
➜  ~ adb uninstall com.hoquvxut.janoschae
Success
```

Alongside its administrator defenses, Sturnus maintains extensive situational awareness through a broad environmental monitoring subsystem designed to ensure long-term resilience on the device. Twelve internal broadcast receivers and a dedicated security-checking thread continuously track system activity, connectivity changes, power and battery states, SIM transitions, app installation events, USB behavior, and signs of forensic probing or rooting. It also monitors security-relevant settings such as developer mode, ADB debugging, SELinux state, and the device's patch level, reporting any change immediately to the operators.

By collecting sensor information, network conditions, hardware data, and installed-app inventories, the malware builds a detailed device profile that helps attackers assess risk, adapt their tactics, and detect analysis environments or emulators. This continuous feedback loop allows Sturnus to persist on the device, avoid exposure, and remain operational even as conditions change around it.

Conclusions

Sturnus represents a sophisticated and comprehensive threat, implementing multiple attack vectors that provide attackers with near-complete control over infected devices. The combination of overlay-based credential theft, message monitoring, extensive keylogging, real-time screen streaming, remote control, device administrator abuse, and comprehensive environmental monitoring creates a dangerous threat to victims' financial security and privacy.

The malware's architecture demonstrates advanced evasion capabilities through code obfuscation, complex and solid communication encryption, and active interference with security controls. The persistent C2 connection via WebSocket enables real-time command and control, allowing attackers to adapt their tactics dynamically based on device state and victim behavior.

Appendix

Bot Commands

Commands	Description
APP_HIDE	Hides the specified app from the user.

APP_UNHIDE	Makes a previously hidden app visible again.
HIDE_ICON	Removes or hides the app’s launcher icon.
APP_SUSPEND	Temporarily disables an app so it cannot run.
APP_UNSUSPEND	Re-enables an app that was previously suspended.
APP_UNINSTALL	Uninstalls the specified app from the device.
SEND_NOTIFICATION	Displays a notification on the device.
PING	Sends a heartbeat to confirm the malware is active.
LOCK_SCREEN	Forces the device screen to lock.
UNLOCK_SCREEN	Attempts to unlock or bypass the screen lock.
OPEN_URL	Opens a specified URL on the device.
KILL_SELF	Forces the malware to remove or terminate itself.
APP_INSTALL	Installs an app.
REFRESH_DELAY	Updates internal timing or command polling intervals.
LAUNCH_APP	Opens or launches a specified application.
FORCE_STOP_APP	Forces an app to stop running.
CALL_PHONE	Initiates a phone call to a given number.
SEND_SMS	Sends an SMS message to a specified number.
DELETE_ALL_SMS	Deletes all SMS messages on the device.
DELETE_SMS	Deletes a specific SMS message.
DELETE_ALL_CALLS	Deletes the entire call log.
DELETE_CALL	Deletes a specific call entry.

ADD_NEW_CONTACT	Adds a new contact to the address book.
DELETE_ALL_CONTACTS	Removes all contacts from the device.
DELETE_CONTACT	Removes a specific contact.
RING_BUZZ_DEVICE	Makes the device ring or vibrate.
REFRESH_ALL_DATA	Forces complete reloading or syncing of malware data.
START_VNC	Starts a remote-control VNC session.
STOP_VNC	Stops an active VNC remote-control session.
ENABLE_BLACK_OVERLAY	Shows a black screen overlay to hide activity.
DISABLE_BLACK_OVERLAY	Removes the black screen overlay.
ENABLE_UPDATE_OVERLAY	Shows an overlay indicating an update to mask actions.
DISABLE_UPDATE_OVERLAY	Removes the update-style overlay.
START_HVNC	Starts a hidden VNC session (invisible remote control).
STOP_HVNC	Stops the hidden VNC session.
RELOAD_INJECTS	Reloads phishing/overlay inject templates.
ENABLE_INJECT	Activates a specific phishing/overlay injection.
PIN_SOLVER	Attempts to bypass or solve the device PIN.
REQUEST_PERMISSION	Prompts the user for a required permission.
DISABLE_INJECT	Deactivates a specific phishing/overlay injection.
HIDE_SMS	Hides SMS messages from the inbox.
UNHIDE_SMS	Restores previously hidden SMS messages.
PIN_SOLVER2	Alternate or updated method for bypassing device PIN.

Indicators of Compromise

SHA-256	Package name	Application name	C2
045a15df1121ec2a6387ba15ae72f8e658c52af852405890d989623cf7f6b0e5	com.klivkfbky.izaybebnx	Google Chrome	amoled[.]multicolore
0cf970d2ee94c44408ab6cbcaabfee468ac202346b9980f240c2feb9f6eb246d	com.uvxuthoq.noscjahae	Preemix Box	walnut[.]almondcolle

---

Source: https://www.threatfabric.com/blogs/sturnus-banking-trojan-bypassing-whatsapp-telegram-and-signal