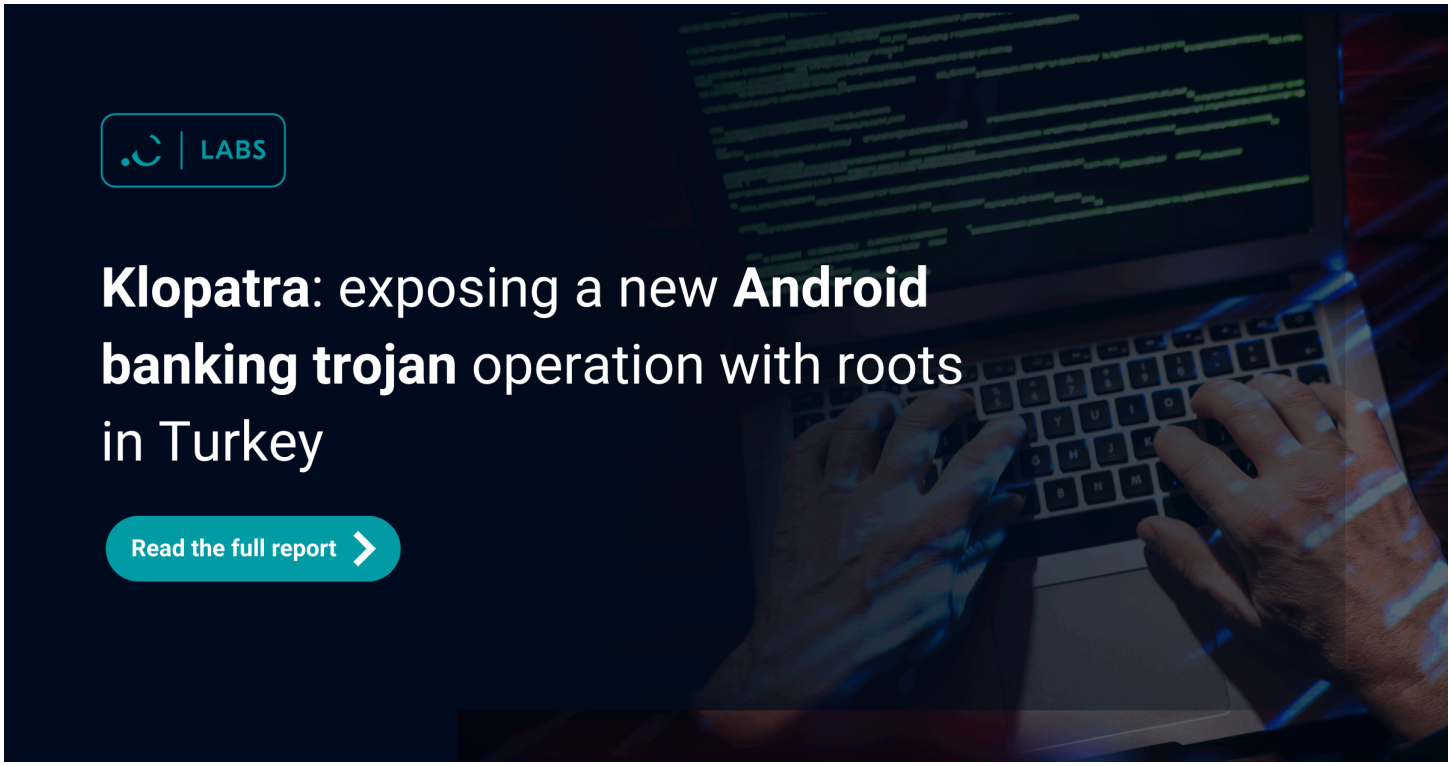
Klopatra: exposing a new Android banking trojan operation with roots in Turkey

Simone Mattia, Michele Roviello ⋮



Download your PDF guide to TeaBot

Get your free copy to your inbox now

Download PDF Version

Key points

- **New Sophisticated Threat Emerges:** In late August 2025, the Cleafy team discovered and analyzed **Klopatra**, a new Android Remote Access Trojan (RAT) that was previously unknown and had no apparent connections to known malware families.
- **Next-Generation Evasion:** Klopatra represents a significant evolution in mobile malware sophistication. It combines extensive use of **native libraries** with the integration of **Virbox**, a commercial-grade code protection suite, making it exceptionally difficult to detect and analyze.
- **Powerful Banking Trojan Capabilities:** At its core, Klopatra is a modern banking trojan that executes fraud through a powerful combination of **Hidden VNC** for complete remote device control and dynamic **Overlay attacks** for credential theft.

- **Active Campaigns in Europe:** At the time of analysis, two main botnets were identified, actively compromising **over 3,000 devices**, with campaigns heavily focused on financial targets in **Spain and Italy**.
- **Turkish-Speaking Threat Actor:** Extensive evidence from malware artifacts, C2 infrastructure data, and direct operator notes strongly indicates that Klopatra is developed and operated by a **Turkish-speaking criminal group**.
- **Agile Development and Private Operation:** Tracking over 40 distinct builds since March 2025 suggests a rapid development cycle. It's plausible that Klopatra is operating as a private botnet with limited affiliates and no public Malware-as-a-Service (MaaS) offering.

Executive Summary

In late August 2025, Cleafy's Threat Intelligence team discovered **Klopatra**, a new, highly sophisticated Android malware currently used in active campaigns against financial institutions and their customers. The analysis identified two major botnets targeting users primarily in **Spain and Italy**, with the number of compromised devices already exceeding 3,000. Klopatra operates as a powerful banking trojan and Remote Access Trojan (RAT), allowing its operators to gain complete control over infected devices, steal sensitive credentials, and execute fraudulent transactions.

What elevates Klopatra above the typical mobile threat is its advanced architecture, built for stealth and resilience. The malware authors have integrated **Virbox**, a commercial-grade code protection tool rarely seen in the Android threat landscape. This, combined with a strategic shift of core functionalities from Java to **native libraries**, creates a formidable defensive layer. This design choice drastically reduces its visibility to traditional analysis frameworks and security solutions, applying extensive code obfuscation, anti-debugging mechanisms, and runtime integrity checks to hinder analysis.

This technical sophistication provides a clear footprint of the Threat Actor (TAs). Linguistic clues within the malware's code and intelligence gathered from the Command and Control (C2) infrastructure point decisively to a **Turkish-speaking origin**. This assessment is corroborated by operational notes left by the attackers themselves, revealing a cohesive and disciplined group managing the entire attack lifecycle, from development to monetization. Klopatra marks a significant step in the professionalization of mobile malware, demonstrating a clear trend of TAs adopting commercial-grade protections to maximize the lifespan and profitability of their operations.

Introduction: The Rise of Klopatra

The mobile threat landscape is constantly evolving, with Android banking trojans becoming increasingly bold and technically advanced. However, most of these threats tend to follow recognizable patterns, reusing known code or techniques. Occasionally, a threat emerges that **deviates from the norm**, signaling a shift in adversary tactics and capabilities.

Klopatra is one such threat. Initially discovered by **Cleafy's automated detection systems** due to behavioral anomalies indicative of sophisticated activity, it quickly stood out as a previously undocumented malware. The analysis revealed no connections to existing malware families, prompting the team to classify it as a new

family. The name "Klopatra" was derived from the name of a certificate embedded in one of the first analyzed samples (March 2025), an artifact left by the developers that provided the first clue to its identity.

Distinguished Name	C:TR, CN:KLOPATRA, L:ANKARA, O:AHMET TOLONOY, ST:İSTANBUL, OU:KLOPATRA
Common Name	KLOPATRA
Organization	AHMET TOLONOY
Organizational Unit	KLOPATRA
Country Code	TR
State	İSTANBUL
Locality	ANKARA

Figure 1 - Certificate details

This report provides a comprehensive technical analysis of Klopatra. It explores its unique, **evasion-focused architecture**, its powerful capabilities as a financial fraud tool, the operational infrastructure supporting active campaigns, and the evidence that conclusively links the operation to a Turkish-speaking group. Through this analysis, Klopatra emerges as an **immediate threat to financial institutions** and a harbinger of **a new wave of mobile malware** adopting professional-grade obfuscation techniques to evade defenses, which has already been seen on other modern Android banking trojans.

Anatomy of an Attack: From Lure to Full Control

Klopatra's effectiveness lies in a carefully orchestrated **infection chain**, which begins with social engineering and culminates in the complete takeover of the victim's device. Each stage is designed to overcome the defenses of the user and the Android operating system.

The Dropper: A Trojan Horse Disguised as IPTV

The initial phase of the attack relies on a dropper application, a lure designed to appear legitimate and desirable. In this case, the dropper masquerades as an IPTV application called "Mobdro Pro IP TV + VPN," promising access to **high-quality television channels**. This choice is not accidental; pirated streaming applications are very popular, and users are often willing to install them from unofficial sources, bypassing the protections of the Google Play Store.

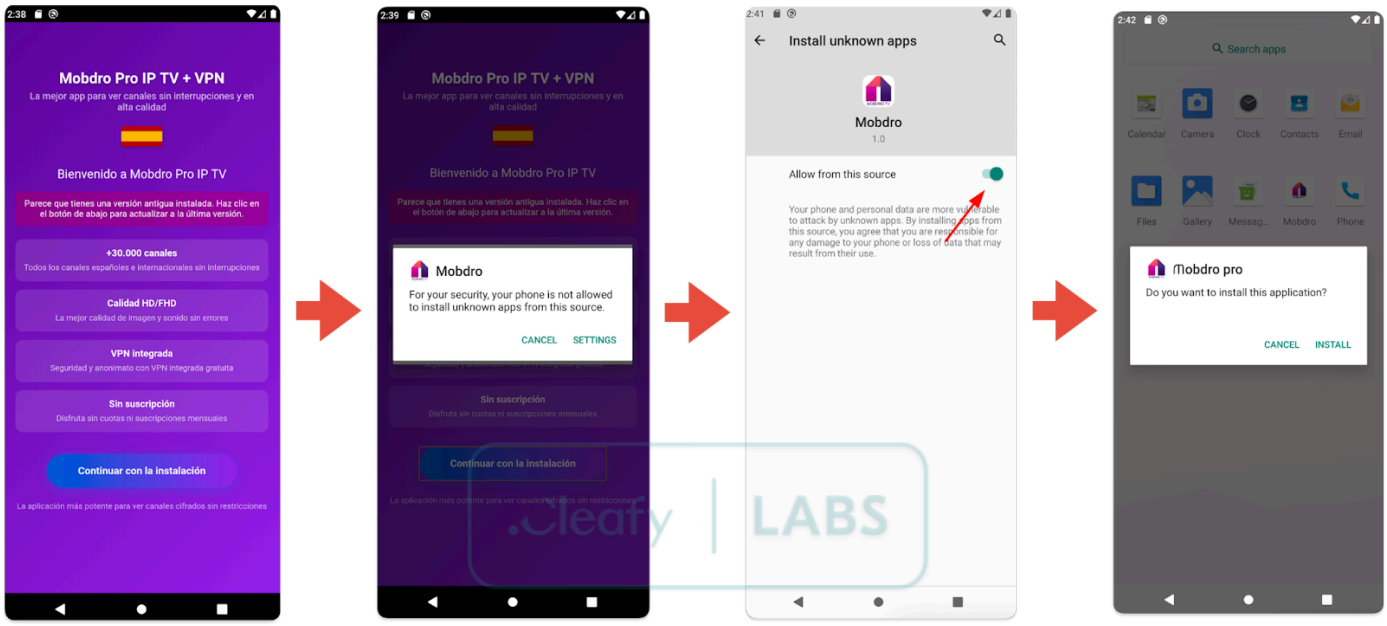


Figure 2 - Installation process

Once installed, the dropper application has a single purpose: to convince the user to grant critical permission, which is the ability to install packages from unknown sources (**REQUEST_INSTALL_PACKAGES**). To achieve this, the app presents a **simple user interface** with a button inviting users to "continue with the installation." Tapping this button redirects the user to Android's system settings and instructs them to grant the permission.

Technically, the dropper uses a technique known as "JSON Packer" to hide the real malicious payload within its resources. This serves as a first layer of evasion, making it more difficult for static analysis tools to identify the malicious nature of the initial application immediately. Once permission is obtained, the dropper **silently extracts and installs the main Klopatra payload**, completing the initial installation process.

Abusing Accessibility Services for Full Control

Once the main Klopatra payload is installed, the real threat manifests. The malware immediately requests a wide range of permissions, but one is crucial for its success: the Android Accessibility Services permission.

Accessibility Services are a powerful framework designed to assist users with disabilities. They allow applications to read screen content and perform actions on behalf of the user. In the hands of malware, this functionality becomes a weapon.

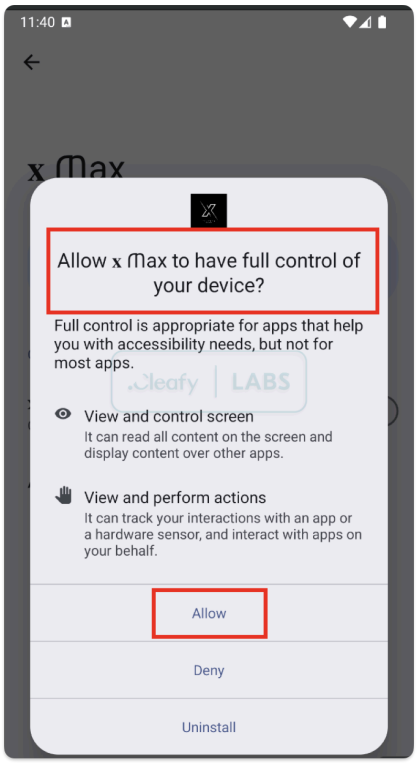


Figure 3 - Enabling Accessibility Services

This single permission grants the malware almost unlimited powers over the device :

- **Screen Monitoring:** It can read any text displayed, including bank balances, private messages, and typed passwords (which are briefly visible).
- **Input Capturing:** It can record every typed tap and character, acting as a comprehensive keylogger.
- **Action Execution:** It can simulate taps and gestures, allowing it to navigate apps, click buttons ("Allow," "Transfer"), enter text, and ultimately, perform fraudulent transactions autonomously.

The abuse of Accessibility Services is the cornerstone of modern banking malware fraud. The technical mechanism turns a malware infection into a direct financial loss, allowing Klopatra to operate with the same level of authority as the legitimate user, but completely invisibly.

Technical Analysis

Klopatra's internal code structure reveals a level of sophistication that sets it apart from most Android malware. Its developers have made **deliberate design choices** to maximize evasion, resilience, and operational effectiveness.

The Evasion Engine: Virbox and Native Code

The most innovative feature of Klopatra is its **multi-layered approach to evading analysis**. At the heart of this strategy is the integration of **Virbox**, a Chinese commercial software protection solution. While using commercial packers and protectors like VMProtect is standard practice in **Windows malware**, its adoption in

the mobile landscape, particularly on Android, is still rare. This choice signals a significant escalation, indicating that operators invest financial resources to protect their malicious "assets."

Adopting Virbox is not a purely technical decision, but a strategic one. It drastically increases the time and expertise required for security researchers to reverse-engineer the malware. Virbox is combined with a firm reliance on native libraries (C/C++ code). Unlike typical Android malware that implements most of its logic in Java/Kotlin, Klopatra shifts its core functionalities to the native layer. This strategy offers two main advantages:

1. **Powerful Layer of Protection**: Moving the core functionality into native libraries drastically reduces its visibility to traditional detection tools and analysis frameworks.
2. **Static Analysis Evasion:** It further integrates robust anti-debugging mechanisms, runtime integrity checks, and emulator detection routines designed to thwart analysis environments and prevent execution under controlled conditions.

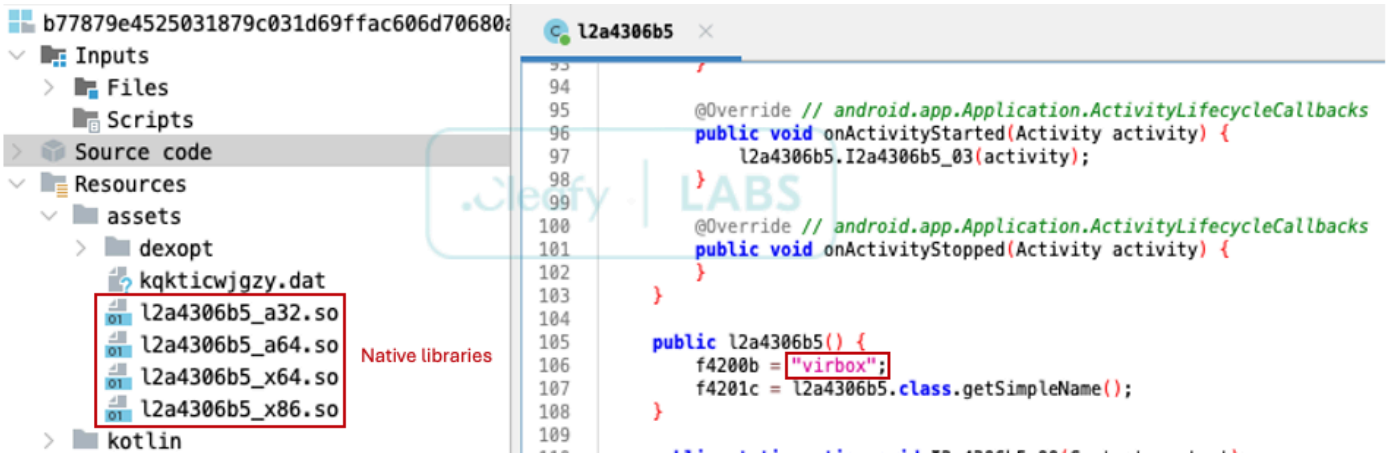


Figure 4 - ‘virbox’ reference and its related native libraries

The RAT Framework: Remote Control and Espionage

At its core, Klopatra is a powerful Remote Access Trojan (RAT). It provides operators with **granular, real-time control** over the infected device. This capability is implemented through two main VNC (Virtual Network Computing) features:

- **Standard VNC (start_vnc):** The operator can view and interact with the victim's device screen remotely. The operator sees exactly what the user sees.
- **Hidden VNC (start_hvnc):** This is the more dangerous mode. When activated, the malware displays a black screen on the victim's device (via the **action_blackscreen command**), making it seem like the phone is off or locked. Behind this screen, the operator has full control and can navigate through apps, enter PINs and passwords, and execute banking transactions without the victim's knowledge.

The level of control is highly detailed, as demonstrated by the list of C2 commands (see Appendix A). Operators can perform a wide range of actions, including simulating taps at specific coordinates (**click_coord**), system button presses like "back" (**global_action_back**) and "home" (**global_action_home**), and even executing complex gestures like swipes (**swipe_up, swipe_down**) and long-presses.

```
public final void doCommand(JSONObject command) {
    AbstractC2873o.m3371f(command, "command");
    String string = command.getString("command");
    Log.d("doCommand", string);
    if (AbstractC2873o.m3366a(string, "open_app")) {
        JSONObject jSONObject = command.getJSONObject("data");
        Context applicationContext = getApplicationContext();
        AbstractC2873o.m3370e(applicationContext, "getApplicationContext(...)");
        String string2 = jSONObject.getString("paket");
        AbstractC2873o.m3370e(string2, "getString(...)");
        openApp(applicationContext, string2);
        return;
    }
    if (AbstractC2873o.m3366a(string, "action_click")) {
        JSONObject jSONObject2 = command.getJSONObject("data");
        if (jSONObject2.getInt("Node") != -1) {
            this.totalNodesEncountered = 0;
            AccessibilityNodeInfo obtain = AccessibilityNodeInfo.obtain(this.lastestevent);
            traverseAndClick(obtain, jSONObject2.getInt("Node"), 16);
            obtain.recycle();
            return;
        }
```


Figure 5 - Parsing commands received from the C2 infrastructure

The Financial Fraud Module: Overlays and Data Exfiltration

In addition to direct device control, Klopatra employs the classic technique of **overlay attacks** for **large-scale credential theft**. The malware maintains a list of target financial and cryptocurrency applications. When it detects the user opening one of these applications, Klopatra sends a request to the C2 server.

The server responds with custom HTML content, which the malware displays as a dialog box over the legitimate app (via the **open_enj** command). This dialog box perfectly mimics the banking app's login screen, tricking users into entering their username and password. These credentials are captured and immediately sent to the C2 server.

In parallel, Klopatra performs comprehensive data collection. It gathers device information (model, manufacturer, battery level), a list of all installed applications (**paketleri_al**), and captures sensitive data such as keystrokes and clipboard content. All collected data is structured into a JSON object, Base64-encoded, and sent to the C2, providing operators with a detailed profile of each victim.

```
f_dtpvv = anazinDosyamiz.f_dtpvv();
JSONArray jSONArray = new JSONArray();
for (C0667e c0667e : anazinDosyamiz.getVeriYoneticisi()) {
    JSONObject jSONObject = new JSONObject();
    jSONObject.put("paket", c0667e.f2843a);
    jSONObject.put("icerik", c0667e.f2844b);
    jSONArray.put(jSONObject);
}
anazinDosyamiz.getVeriYoneticisi().clear();
JSONObject jSONObject2 = new JSONObject();
str = anazinDosyamiz.v_vslom;
f_sdllu = anazinDosyamiz.f_sdllu(String.valueOf(str));
jSONObject2.put("botid", f_sdllu);
f_sdllu2 = anazinDosyamiz.f_sdllu("0.0.0.0");
jSONObject2.put("ip", f_sdllu2);
f_sdllu3 = anazinDosyamiz.f_sdllu(anazinDosyamiz.f_mbgky());
jSONObject2.put("ulke_kodu", f_sdllu3);
jSONObject2.put("kaynak", this.f2813b);
f_sdllu4 = anazinDosyamiz.f_sdllu("27.08-1");
```

Figure 6 - Collected data formatted and sent to the C2

Self-Preservation and Defense

Klopatra is designed to survive on the device for as long as possible. It uses **Accessibility Services privileges** to self-grant additional permissions, programmatically clicking "Allow" or "OK" buttons in system dialogs. It autonomously navigates to the battery optimization settings (command **pil_opt**) to add itself to the exception list, preventing the operating system from terminating its background processes.

Furthermore, Klopatra adopts active defensive measures. It contains a **hardcoded list of package names** belonging to popular security and AV solutions. If it detects one of these apps installed on the device, it may attempt to uninstall it to eliminate threats to its operation. Finally, it can simulate the "back" button press to prevent the user from easily accessing the settings screens where they might attempt to uninstall the malware.

```
    private final Set<String> v_tpzah = AbstractC1138B.m2402I("com.miui.packageinstaller",
"com.avast.android.mobilesecurity", "com.antivirus", "com.bitdefender.android.antivirus",
"com.kms.free", "com.mcafee.android.app", "com.symantec.mobilesecurity", "com.sophos.smsec",
"com.trendmicro.android.avm", "com.eset.ems.android.antivirus", "com.qustodio.qustodioapp",
"com.comodo.mobile.security", "org.malwarebytes.antimalware",
"com.pandasecurity.mobilesecurity", "com.drweb.pro", "com.ahnlab.v3mobile", "com.avira.android"
, "com.mcafee.android.app", "com.fsecure.key", "cz.zoner.android.security",
"com.iobit.mobile.security", "com.secureteen.android", "com.gdata.mobile.android.security");
```

Figure 7 - Hardcoded list of security and AV solutions

Attribution and Operator Analysis: Unveiling the Turkish Connection

Malware analysis goes beyond its code; **investigating who is behind it** is crucial to understanding the threat. In the case of Klopatra, multiple lines of evidence converge, painting a clear picture of a Turkish-speaking criminal group.

Linguistic Fingerprints in Code and Infrastructure

The first clues emerged directly from the **malware's code**. Function and variable names, left by the developers, were in Turkish. For example, a central function for handling remote commands was named **ArkaUcKomutIsleyicisi**, which roughly translates to "Backend Command Handler."

```
public final void ArkaUcKomutIsleyicisi(JSONObject jSONObject) {
    AbstractC0838i.m2077e("command", jSONObject);
    String string = jSONObject.getString("command");
    Log.i("doCommand", string);
    if (string != null) {
        switch (string.hashCode()) {
            case -2128954546:
                if (string.equals("start_vnc")) {
                    f_nwusm();
                    this.v_nxbtr = true;
                    return;
                }
                break;
            case -2121957709:
                if (string.equals("diger_uyg")) {
                    new Handler(Looper.getMainLooper()).postDelayed(new RunnableC0638d(this, 7), 500L);
                    return;
                }
                break;
            case -2046190510:
                if (string.equals("enj_switch")) {
                    String string2 = jSONObject.getJSONObject("data").getString("paket");
                    AbstractC0838i.m2074b(string2);
                    Boolean f_ysejs = f_ysejs(string2);
                    if (AbstractC0838i.m2073a(f_ysejs, Boolean.FALSE)) {
```

Figure 8 - Backend Command Handler function (ArkaUcKomutIsleyicisi).

This linguistic footprint extended to the **Command and Control infrastructure**. Analysis of the JSON responses sent by the C2 servers revealed that many field names were Turkish words. This is a strong indicator, as the control panel's structure reflects the developers' native language. Key fields included :

- **etiket**: "label" or "tag," operators use to version malware builds and segment campaigns.
- **favori_durumu**: "favorite status," a boolean flag used to mark high-value victims for manual analysis.
- **bot_notu**: "bot note," a free-text field for operators to add comments about specific victims.

These fields confirm the Turkish origin and offer insight into the operators' TTPs (Tactics, Techniques, and Procedures), showing how they manage and prioritize compromised devices.

Retrieve Intelligence from Operator Notes

The **bot_notu** field proved to be a goldmine for **intelligence**, providing a direct window into **human interactions** with the control panel. Analysis of a data dump from one of the C2 servers uncovered a particularly revealing note associated with a specific bot: *"7k atılan piç şifre z"*.

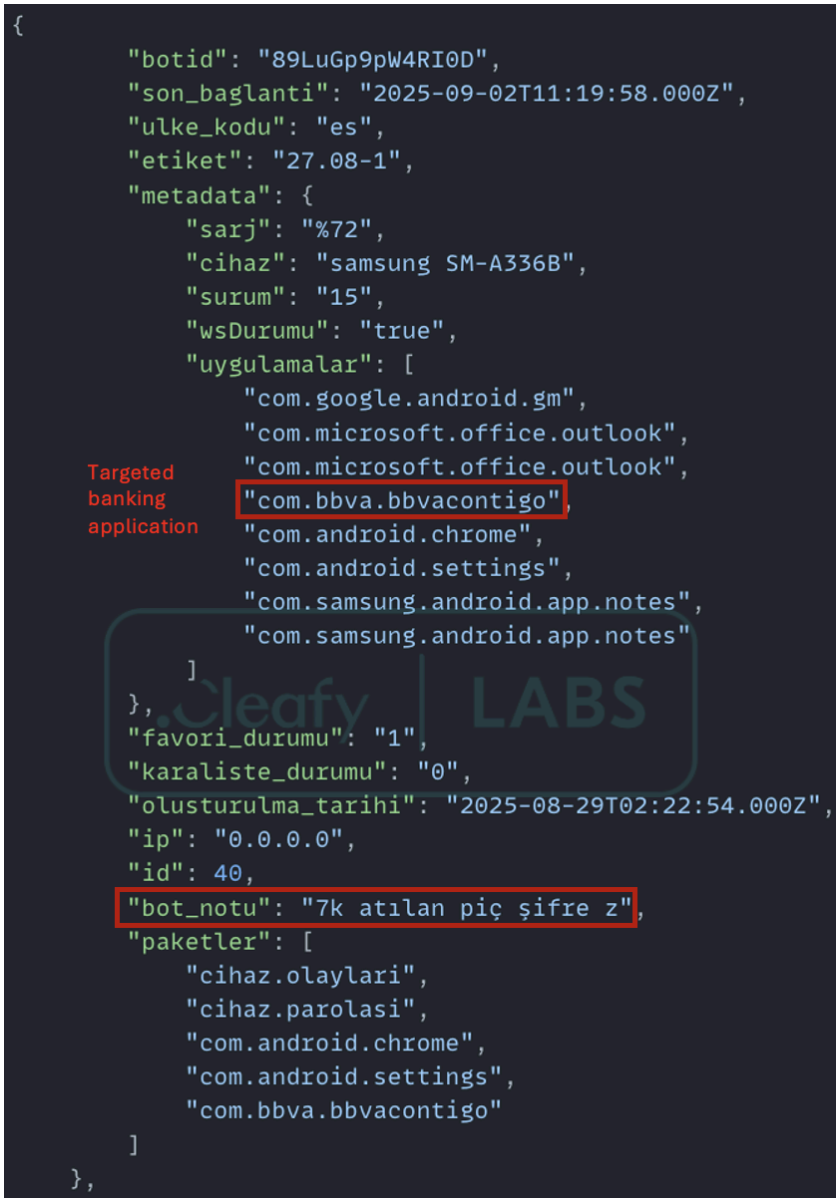


Figure 9 - Operator's note written in Turkish

This short phrase, written in colloquial and vulgar Turkish, is extremely significant. An analysis of the note suggests the following:

- *"7k atılan"*: Likely refers to a fraudulent transaction attempt of EUR 7,000. "Atılan" means "thrown" or "sent."
- *"piç"*: Is a common insult in Turkish, expressing frustration. This suggests the transaction may have failed.
- *"şifre z"*: "Şifre" means "password," and "z" most likely refers to the device's 'Z'-shaped unlock pattern.

This note alone provides overwhelming evidence. It confirms the language and shows **a human operator actively engaged in fraud attempts**, documenting victim device details (like the unlock pattern) for future access. Further examples reinforced this TTP, with notes like *"desen M"* ("desen" is Turkish for "pattern") and numeric PINs like *"4758"* and *"12369"* found in other bot records.

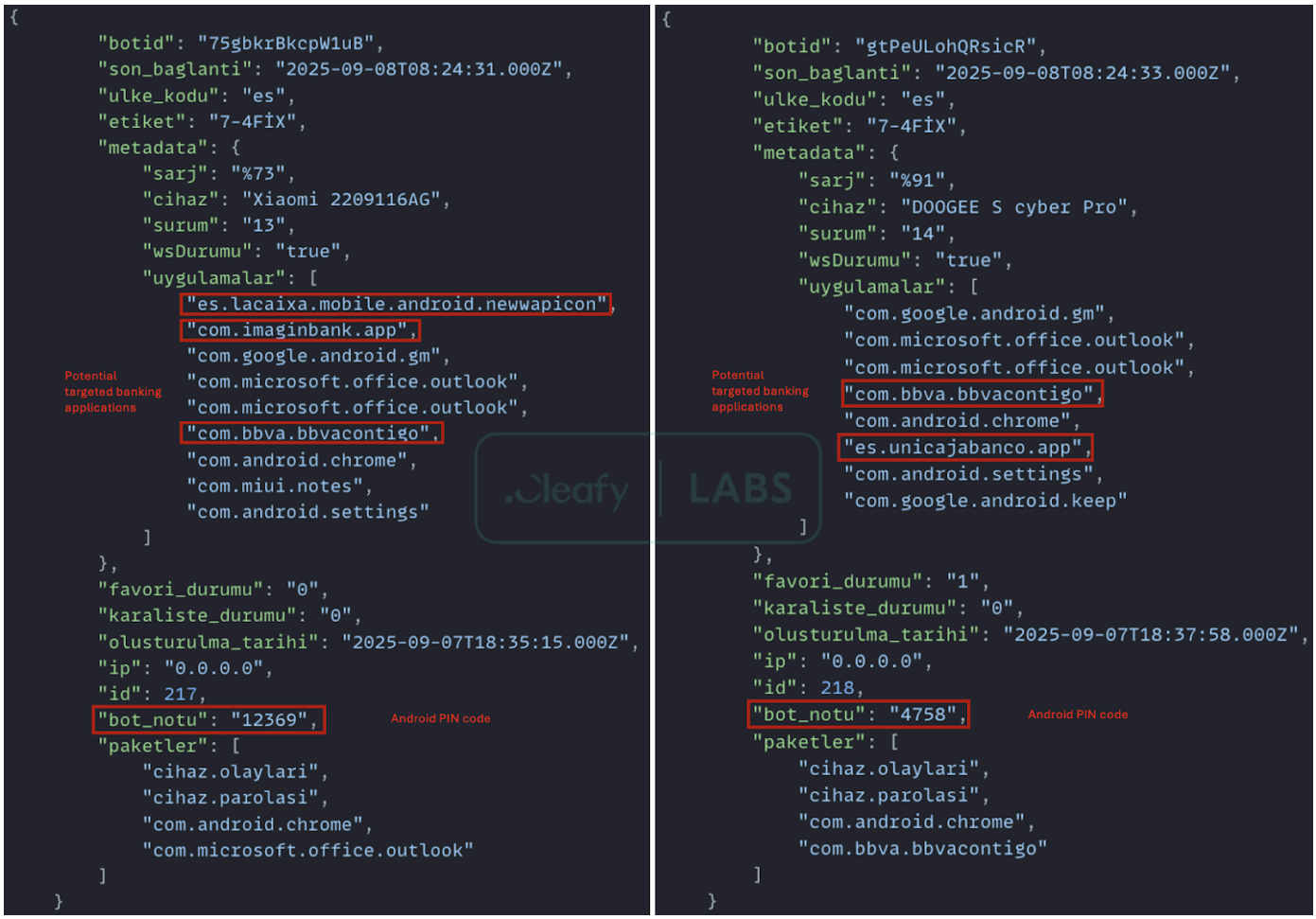


Figure 10 - PIN device written in the note field

The **linguistic consistency** between the malware's code, the C2 panel interface, and the colloquial operational notes from the operators themselves strongly suggests a vertically integrated criminal operation. This is not a **Malware-as-a-Service model**, where developers of one nationality sell their tools to customers of another. Instead, it indicates a cohesive Turkish-speaking group managing the entire attack lifecycle, from code development to victim monetization.

Mapping the Operation: Klopatra's Live Infrastructure

The investigation into the Command and Control (C2) infrastructure between August and September 2025 provided direct visibility into Klopatra's active campaigns. Although this visibility may be partial, the data collected from the three main servers offers valuable insights into the group's operations and TTPs.

The Italian and Spanish Botnets

The analysis revealed two main campaigns managed by dedicated C2 servers with a clear geographical focus.

- **adsservices[.]uk**: This server commands the largest observed botnet, with nearly 1,000 active infections. The campaign is overwhelmingly focused on Spain, which accounts for most of its victims. Consequently, the targeted applications are those of major Spanish banks.

- **adsservice2[.]org**: This server orchestrates the Italian campaign, managing a botnet of approximately 450 infected devices. The data confirms a clear specialization, with the vast majority of victims in Italy and the malware targeting major Italian banking applications.
- **141.98.11[.]227**: A third C2 server was identified solely by its origin IP, without an associated Cloudflare FQDN. This server also primarily targets Spain and controls a smaller botnet of around 175 devices. It is hosted by the same Lithuanian provider as **adsservice2[.]org**, strongly suggesting a connection between the two Spanish-focused operations.

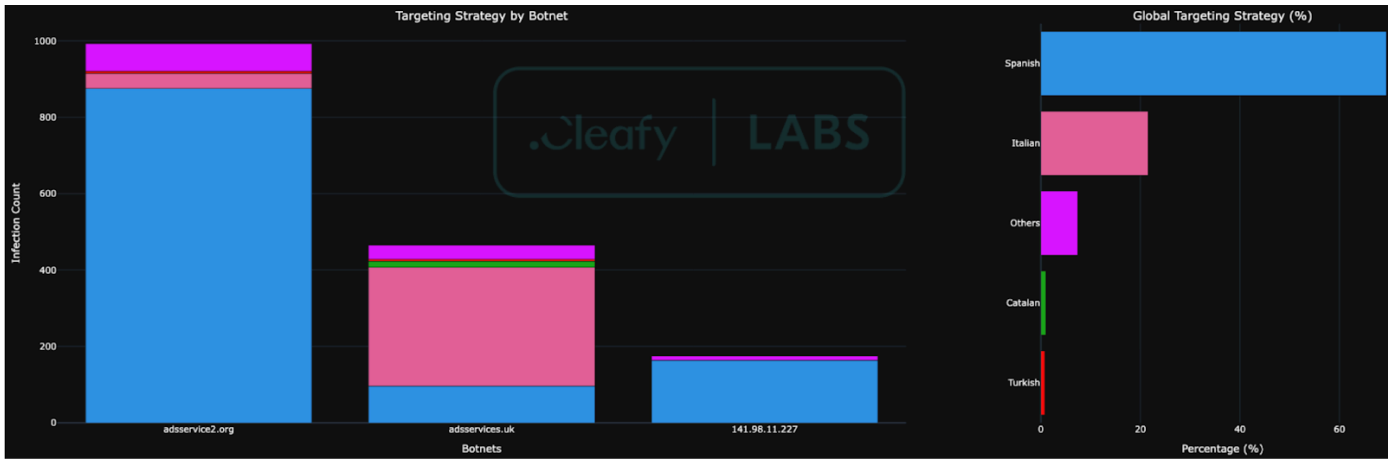


Figure 11 - Victims' statistics from C2 infrastructure (September 2025)

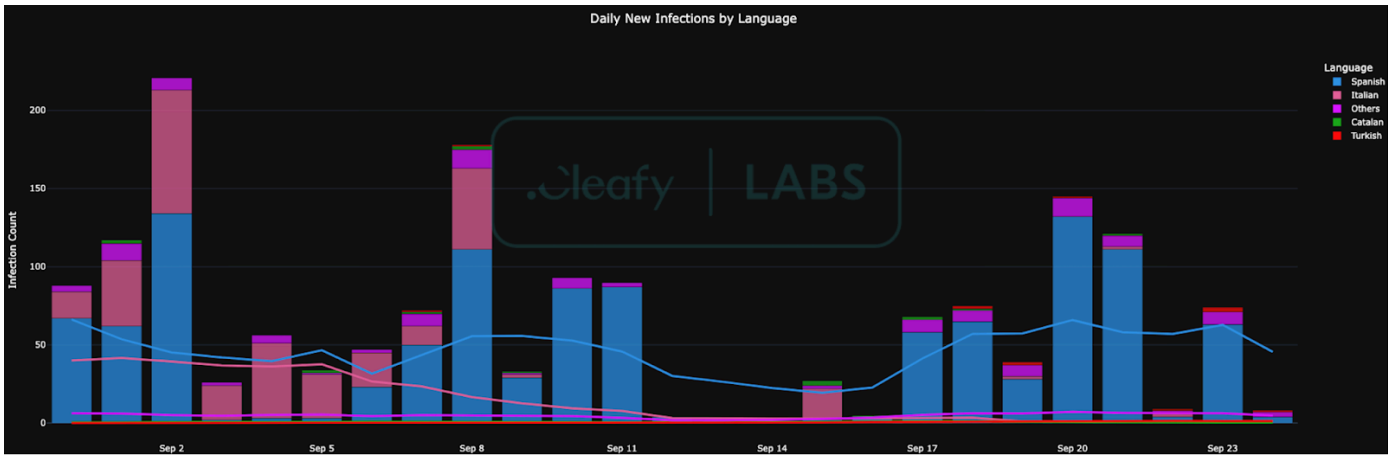


Figure 12 - Daily new infections (September 2025)

A Staging Server?

The third identified server, **guncel-tv-player-lnat[.]com**, had a very different profile. It hosted only 9 bots scattered across various locations (Georgia, Kuwait, Qatar) and with an inconsistent set of targets. All evidence suggests that this server is not part of a large-scale campaign but rather serves as a staging or testing environment for the malware developers. This provides insight into the actor's development lifecycle, showing where they might test new features before deploying them to the main botnets.

Behind the Cloudflare Veil

As is common practice for TAs, Klopatra's operators used Cloudflare to mask the true IP addresses of their C2 servers. Whois records confirmed that Cloudflare was used as both the registrar and the nameserver, making

direct identification via standard DNS lookups difficult.

Despite this layer of obfuscation, it was possible to identify the origin IP addresses. This de-anonymization was likely the result of an operational security (OPSEC) mistake by the actors. By leveraging network asset intelligence platforms, it was possible to uncover misconfigurations. The most likely scenario is that the operators pointed the domains directly to their server IPs during the initial setup phase, before correctly configuring the Cloudflare proxy. This error permanently exposed the true server location to historical analysis tools.

This discovery allowed for the localization of the main infrastructure and the linking of the two main campaigns :

- The campaign servers, **adsservice2[.]org** and **adsservices[.]uk**, resolve to **141[.]98.11.69** and **141[.]98.11.80**, respectively. Both IPs are in Lithuania and belong to the same provider, UAB Host Baltic (ASN 206805).
- The test server, **guncel-tv-player-lnat[.]com**, resolves to **37[.]114.60.174**, hosted in Germany by the provider Vtion Group (ASN 39367).

The definitive proof of this link was obtained by accessing the C2 endpoint through the Cloudflare-protected domain and directly via the discovered origin IP, obtaining identical responses in both cases.

FQDN	Origin IP Address	Provider & Location	Primary Geographic Target	Last Observed Bot Count*
adsservice2[.]org	141[.]98.11.69	UAB Host Baltic (Lithuania)	Spain	495
adsservices[.]uk	141[.]98.11.80	UAB Host Baltic (Lithuania)	Italy	2433
guncel-tv-player-lnat[.]com	37[.]114.60.174	Vtion Group (Germany)	N/A	194
N/A	141[.]98.11.227	UAB Host Baltic (Lithuania)	Spain	214

The "Bot Count**" metrics were derived from a specific C2 endpoint that lists all connected bots. This data includes a progressive'id'field, which increments with each new infection. The numbers presented in the table reflect the highest 'id' value observed for each server during our last data extraction on September 22, 2025.*

The Evolution of a Threat

Klopatra is not a static threat. The analysis shows it is under an active and rapid development cycle, with the authors constantly refining its capabilities and evasion techniques.

Targeted threat hunting operations, using unique artifacts from the initial sample as pivot points (package name, signing certificate), identified nearly 40 related samples. Among these, one discovery proved crucial: a development or test build of Klopatra that was not protected by Virbox. This unobfuscated sample acted as a

"Rosetta Stone," allowing for a complete reverse-engineering of the malware's core logic, understanding its C2 mechanisms, and discovering the full range of its capabilities.

This collection of samples has made it possible to outline at least three distinct evolutionary phases:

1. **Initial Builds (circa March 2025):** The earliest identified versions were more limited. They lacked key features essential for a modern banking trojan, such as overlay attacks and advanced permission management. They likely represented a prototype or an alpha version.
2. **Intermediate Builds:** This phase introduced the commercial protector Virbox as the main defense mechanism. This marks the time when the developers decided to invest in professional evasion techniques to increase the malware's longevity.
3. **Recent Builds:** The latest observed version demonstrates a further leap in quality. The authors have adopted a multi-layered defensive approach, combining Virbox with a custom string encryption mechanism. Analysis identified that this obfuscation is performed via the string encryption routines of NP Manager, another publicly available tool.

Moreover, recent builds include more commands related to the **UI Overlays and Injections** and advanced **permission management**, which are crucial for effective banking trojans.

This progression from a basic version to a feature-rich and heavily obfuscated one highlights the malware authors' rapid iteration and adaptation. This agile development cycle indicates a dedicated and responsive adversary.

Fraud in Action: Reconstructing a Nocturnal Attempt

Klopatra's technical capabilities are not just theoretical. Through incident response activities and the real-time visibility provided by the Cleafy platform, our team was able to reconstruct the precise TTPs used by the operators in a live fraud scenario. A clear picture of a sophisticated, hands-on keyboard attack emerged by correlating data from victims' devices, our platform's telemetry, and the malware's code.

The operators show a clear preference for conducting their attacks during the night. This timing is strategic: the victim is likely asleep, and their device is often left charging, ensuring it remains powered on and connected. This provides the perfect window for the attacker to operate undetected.

The attack unfolds in a carefully orchestrated sequence:

1. **Environment Check:** The malware first confirms the ideal conditions. It uses **BatteryManager** to detect if the device is charging and **PowerManager.isInteractive()** to verify that the screen is off and the user is inactive.
2. **Stealth Mode Activation:** The operator initiates the stealth takeover once the conditions are met. The **action_blackscreen** command is sent, which programmatically reduces the screen brightness to zero (**screenBrightness = 0.0f**) and displays a black overlay (**R.drawable.blsc**), making the device appear to be turned off.
3. **Device Unlock:** Using the PIN or pattern previously stolen and recorded in the C2 panel's **bot_notu** field, the operator sends the **unlock_pin** command to gain access to the device.

4. **Application Launch:** The operator then issues the **open_app** command to launch the target banking application.
5. **Fraudulent Transaction:** With full control via Hidden VNC (**start_hvnc**), the operator manually navigates the banking app's interface, fills in the details for a new payee, and initiates multiple instant bank transfers, draining the victim's account while they sleep.

The following screenshot from the Cleafy console shows the detection of the first nocturnal fraud attempt (patient zero). The platform identified a high-risk session from a compromised device in the middle of the night.

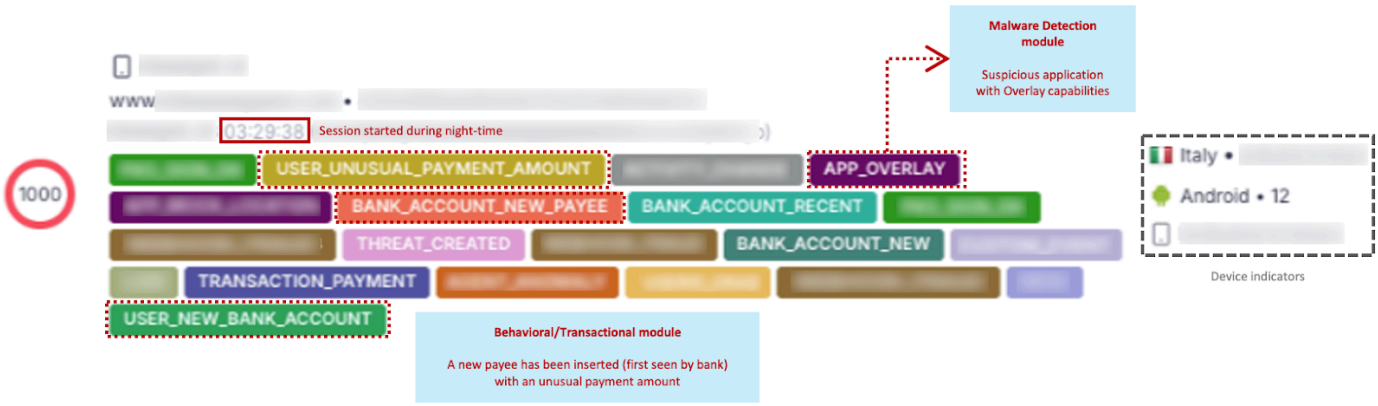


Figure 13 -Cleafy console (Patient zero)

A detailed view of the session reveals the operator's actions: a rapid succession of instant payment attempts to a new bank account.

Figure 14 - Fraud attempts details

This use case demonstrates the seamless integration of Klopatra's automated capabilities with direct, manual intervention by a human operator. It transforms the malware from a simple data-stealer into a direct, real-time financial theft tool, highlighting the critical need for **combining malware detection, behavioral capabilities, and transactional data** to detect such sophisticated attacks.

Conclusions

Klopatra represents a **significant and sophisticated threat to the financial sector and mobile device users**, particularly in Europe. The analysis conducted by the Cleafy team revealed malware that is not only technically advanced but is also managed by a cohesive and disciplined Turkish-speaking criminal group, controlling operations from A to Z.

Its main innovation—adopting **commercial-grade protections** like Virbox, combined with an **architecture based on native code**—marks a turning point in the Android malware landscape. This approach, once the domain of desktop malware, is now a reality in the mobile world and serves as a harbinger of future trends. It is likely that other criminal groups will follow suit, making detection and analysis increasingly complex and resource-intensive.

The active campaigns in **Italy** and **Spain**, with **over 1,000 confirmed victims**, demonstrate that Klopatra is not an experiment but a fully operational and successful fraud tool. The agility shown by its rapid development cycle suggests that the operators will continue to refine their TTPs, expand their target list, and integrate new evasion techniques to stay one step ahead of the security community.

For financial institutions and anti-fraud teams, the emergence of Klopatra underscores the need for threat detection solutions that go **beyond static analysis** and focus on **device-level behavioral monitoring**. For the threat intelligence community, continuous monitoring of this group and its infrastructure will be essential to anticipate their next moves and protect users from this evolving threat.

Appendix A - Command List

Command Category	Initial Builds	Intermediate Builds	Recent Builds	Description
User Interaction	action_click	action_click	click_by_id_press_long	Simulates a single click on a UI element (Node).
	action_long_click	action_long_click	click_by_id_click	Simulates a long press on a UI element (Node).
	click_coord	click_coord	swipe	Taps at specific screen coordinates.
	action_edit_text	action_edit_text	click_by_id_and_text	Enters a specified string of text into an editable UI element.
	action_custom_gesture	action_custom_gesture	swipe_with_list	Performs a complex gesture (e.g., a series of taps or a multi-point swipe).
	unlock_pin	unlock_pin	type_and_enter	Enter a PIN or password and confirm it to unlock the device.
	(none)	yazi_gonder	send_text_to_view	Pastes a text string from the clipboard into a specific UI element.
Navigation	global_action_back	global_action_back	global_action_back	Simulates the back button.

Command Category	Initial Builds	Intermediate Builds	Recent Builds	Description
	global_action_home	global_action_home	global_action_home	Simulates the home button.
	global_action_recents	global_action_recents	open_recents	Simulates the recent apps button.
	swipe_up	swipe_up	(none)	Performs an upward swipe.
	swipe_down	swipe_down	(none)	Performs a downward swipe.
	swipe_left	swipe_left	(none)	Performs a leftward swipe.
	swipe_right	swipe_right	(none)	Performs a rightward swipe.
Device & System	action_screen_on	action_screen_on	device_lock_screen	Locks the screen or wakes it from sleep.
	start_hvnc	start_hvnc	screen_control	Starts the remote screen control session.
	stop_hvnc	stop_hvnc	screen_off	Stops the remote screen control session.
	(none)	ses_kapat	unlock	Mutes all audio on the device.
Data & Information	sendDeviceInfoToApi (implied)	veri_gonder	get_device_info	Gathers and sends device-specific information to the attacker.
	(none)	paketleri_al	kill_activity	Collects a list of all installed applications.
	start_record_gesture, stop_record_gesture	start_record_gesture, stop_record_gesture	(none)	Starts and stops recording user gestures.
	(none)	start_vnc, stop_vnc	screen_dump, screen_on	Starts and stops streaming the device's screen contents.

Command Category	Initial Builds	Intermediate Builds	Recent Builds	Description
	(none)	(none)	take_screenshot	Captures a screenshot of the device.
UI Overlays & Injections	action_blackscreen	action_blackscreen	send_sms_by_name	Displays a black overlay on the screen, often with a fake message.
	(none)	action_blackscreen2	(none)	Displays a black overlay without any message.
	(none)	open_enj	open_url	Opens a WebView to display a web page or injected HTML.
	(none)	enj_switch	get_permission	Toggles the active status of a specific app for screen injection.
Permission Management	(none)	bildirim	open_app	Creates and displays a fake notification.
	(none)	bildirim_izin	open_app_by_name	Redirects to notification settings for the app.
	(none)	izin_switch	change_permission	Toggles the malware's own accessibility service permission.
	(none)	diger_uyg	open_settings	Navigates to a specific app's system settings page.
	(none)	pil_opt	open_notifications	Redirects to battery optimization settings.
	(none)	(none)	logcat	Collects logs from the device.

Appendix B - Indicators of Compromise (IOCs)

Klopatra samples:

Hash (SHA256)
ecf0a143df1c42b3395a6654aa025945378868442e6e0ea3819d943ab42f37d8
d6cd1f369e89692feb2593a4f49bf67b2386df533a551915d8949fd709a0f35a
9893bec37c8c85e36f9a26766ae63c3cf52d2efe6946f02e16a7b0d68b11011e
b77879e4525031879c031d69ffac606d70680a3f06613dd418ee7c656046fbe5
e2b9aa6a5febde1bba371dbed3b04909a222b765cf04ccb12547a9efb5a0dd50
455910bf83c07cb9e3d3fd1a8c3b4d9ceec96a0c81e9d2804ecebfb177c8c97f
79f40993850ebed38a4091f771cdbf327a2143bb17f66af89e6fcb717f3ec0bf
6769da8d5d2c5e57e018234b7dcbc9514a1f2b9ac7cfb42f7c3f55af9f23bad8
e07392df7484aeafad61af6d0a75a9af874183434181bfb8aab843c8dea87e63
0fd9c2e09e05ce59e90f5d373edb7f1d8199a965fbad989651b9ddaa95900dcf
85b05cb1268c6002a6bedb855ad5983eee7b1f084728943bd18d19fcfd9c7d85
02ac34edfacbb3b1dd731ff305846b5bedfdbcaaf11a00f446efd8d0c009788d
3f972448cf4fdf8938b56c0627a2e274e3c9968b0212975eadda8e4de7ab782e
683746b5da267b80795746d5e14b3fbf33d19fb4f39181fb63d28715edcb5d36
e27ba9efe44ef0d08128f80b010854670e77761b32160036465ac25b75723dce
1377664fea7ce3745feeddd30c1498ca98081ad7399f36b7c00e91d733f5211f
6ddb8562e7f1371c3436b983b96930ce75e218e73680f332d9589da96ac2063d
2b8206be0f4e62d2d3036687757587847 87d6674c408a1ce706181ab96362d3e
37119c69ce4f9d68e0ebeb6de99393738144966b130518ed1679c7203f8999ab
451a82fe018dd72cc145e5ad6c21ad955844fe8b33fd1cb23bbf76766ec6a34c
d5fe92a103f643735d42e6070dc3fcc28f15e2cef488dae42ca235a061bc836a
0add6a2853c526843de97234ba1f4db4ab1136ce898189a068486d1e5d1f33e0
57d3198b481f8096a3aac8d3a6da15636369c1a5021a775c2a25fceaa5bbc075
4c6939f639873cdc706144a263b7f11087283ff27289f2131c6f45654b00c1bb
073014cfda7bc8cf54219ef7c05eefe8f8c49c063c2857790a89e34dcdfdb455
0ae8c721bf5240966b7275875c42f789dc922cc1a7e4cef7af2827a2aa171b8b
72a3541a456f7ec3d9de6656c5b9640c8ffd28c5571c5cdfb17d65613c643f84
a045d8b62bbf4cdcfbd449a994958c1e051d06c0d888e0936838fff4be47aefc
1d580642b5bfe53bef632cce7d2db7fe141351e0c02b11a2589c2482a5f63ed2
3dd5b08c0b942912d2fc65c070894466a8fa356a22a768869af14ae90e8f2ad0

C2 Infrastructure:

FQDN	Origin IP
adsservice2.]org	141[.]98.11.69
adsservices.]uk	141[.]98.11.80
guncel-tv-player-lnat.]com	37[.]114.50.174
N/A	141[.]98.11.227