A new TrickMo saga: from banking trojan to victim's data leak

By Michele Roviello, Alessandro Strino

Archived: 2026-04-05 15:32:59 UTC

Key Points

- In June, the Cleafy Threat Intelligence team identified an unclassified Android banking Trojan. Subsequent analyses revealed that the malware was a variant of **TrickMo**, albeit with newly incorporated anti-analysis mechanisms.
- The mechanisms include using **malformed ZIP** files in combination with **JSONPacker**. In addition, the application is installed through a Dropper app that shares the same anti-analysis mechanisms. These features are designed to evade detection and hinder cybersecurity professionals' efforts to analyse and mitigate the malware.
- The sample analysis allowed us to trace the structure of the command-and-control (C2) server and the organisation of **exfiltrated data**, highlighting critical endpoints used to store and manage stolen information. By gaining access to these endpoints, we uncovered **sensitive files**, including credentials and pictures, exfiltrated from infected devices.
- The new findings underscore an enhancement in the Threat Actor’s capabilities. Although TrickMo retains the typical functionalities of an Android banking Trojan, the data collected from infected devices could enable the attacker to undertake additional actions, compromising the victim on multiple levels.

Introduction

Cleafy’s Threat Intelligence team observed an interesting Android malware sample in early June, initially classified as unknown. Further analyses revealed that the malware was a variant of the banking Trojan **TrickMo**, but with newly integrated anti-analysis features that complicated its classification.

TrickMo has a well-documented history of targeting Android devices. It emerged as part of TrickBot’s evolution, enabling TAs (Threat Actors) to expand the infection to the Android environment. The introduced anti-analysis mechanisms, which consist of a combination of different techniques known as **malformed ZIP, JSONPacker**, and **dropper** apps, highlight the malware's ever-evolving nature. The malware's purpose is to evade detection and hinder the efforts of cybersecurity professionals to analyse and mitigate this threat.

Nevertheless, the sample analysis also allowed us to trace the structure of the **command-and-control (C2)** server and the management of exfiltrated data, highlighting critical endpoints used to store the stolen information. We uncovered sensitive files, including credentials and pictures, exfiltrated from infected devices by gaining access to these endpoints.

What makes this discovery particularly noteworthy is the potential impact of the compromise on a victim user. A TA's actions with the exfiltrated data extend beyond banking fraud, potentially triggering identity theft scenarios.

Moreover, as highlighted in this document, the exfiltrated data could be accessible to third parties without authentication, exposing the user to multiple attackers.

Ultimately, in the following sections, we will delve deeper into how a TrickMo infection can expose the user to multiple levels of risk, including banking fraud, identity theft, and data leakage.

Historical Overview

CERT-Bund identified TrickMo for the first time in 2019. The malware was designed to facilitate financial fraud by intercepting **one-time passwords (OTPs)** and other **two-factor authentication (2FA)** mechanisms crucial for secure banking transactions. Its primary targets were banking applications across Europe, particularly in Germany.

The malware represents an evolution of the **TrickBot** group's malicious activities for the mobile domain. TrickBot, originally designed to target Windows systems, quickly became famous for its ability to steal banking credentials and other sensitive information. As cybersecurity defences improved, the TrickBot group developed TrickMo to target Android devices, leveraging and adapting the sophisticated techniques that made TrickBot successful. Over time, TrickMo has continually evolved, incorporating advanced obfuscation techniques and anti-analysis mechanisms to thwart detection and analysis efforts by cybersecurity professionals.

The malware's key features include:

1. **Interception of One-Time Passwords (OTPs)**: it can intercept OTPs sent via SMS or generated by authenticator apps, allowing cybercriminals to bypass 2FA and authorise fraudulent transactions.
2. **Screen Recording and Keylogging:** The malware can record the victim's screen and capture keystrokes, providing attackers with sensitive information such as login credentials and PINs.
3. **Remote Control Capabilities:** TrickMo enables remote control of the infected device, allowing attackers to perform various actions, including initiating transactions and modifying account settings without the user's knowledge. With these capabilities, TrickMo can enable TAs to perform the On-Device Fraud (ODF) scenario, one of the most dangerous types of banking fraud.
4. **Accessibility Service Abuse:** By exploiting Android's accessibility services, TrickMo can grant itself elevated permissions, manipulate user inputs, and capture data from other apps, making it particularly effective in targeting banking applications.
5. **Advanced Obfuscation Techniques:** TrickMo continually evolves its obfuscation methods to avoid detection. It uses sophisticated code-hiding techniques to make it difficult for security researchers to analyse the malware.
6. **Anti-Analysis Mechanisms:** TrickMo incorporates sophisticated methods to evade detection, including various techniques to detect and thwart virtualised environments and analysis tools.

The recent discovery of TrickMo's new anti-analysis mechanisms highlights the malware's continuous evolution. The following sections will discuss these techniques in detail, providing deeper insights into how TrickMo operates. Moreover, the analysed sample revealed characteristics of the malware that extend beyond the typical functionalities of a banking Trojan, possibly aligning with those of an infostealer.

Malicious App Overview

The malicious app is distributed via a dropper disguised as the **Google Chrome** browser. Upon installation, the app displays a warning message prompting users to update Google Play services.

If the user confirms the update, another APK containing the TrickMo malware will be installed. The new app is deceptively named **"Google Services"** and poses as a legitimate instance of Google Play Services. Upon launching, the app displays a window to ask the user to enable **Accessibility services** for the app. It guides the users through the process by instructing them to navigate to "*Settings*" and "*Downloaded Services*". This social engineering tactic exploits the user's trust in familiar names and interfaces, thereby granting the malware the elevated permissions to carry out its malicious activities undetected.

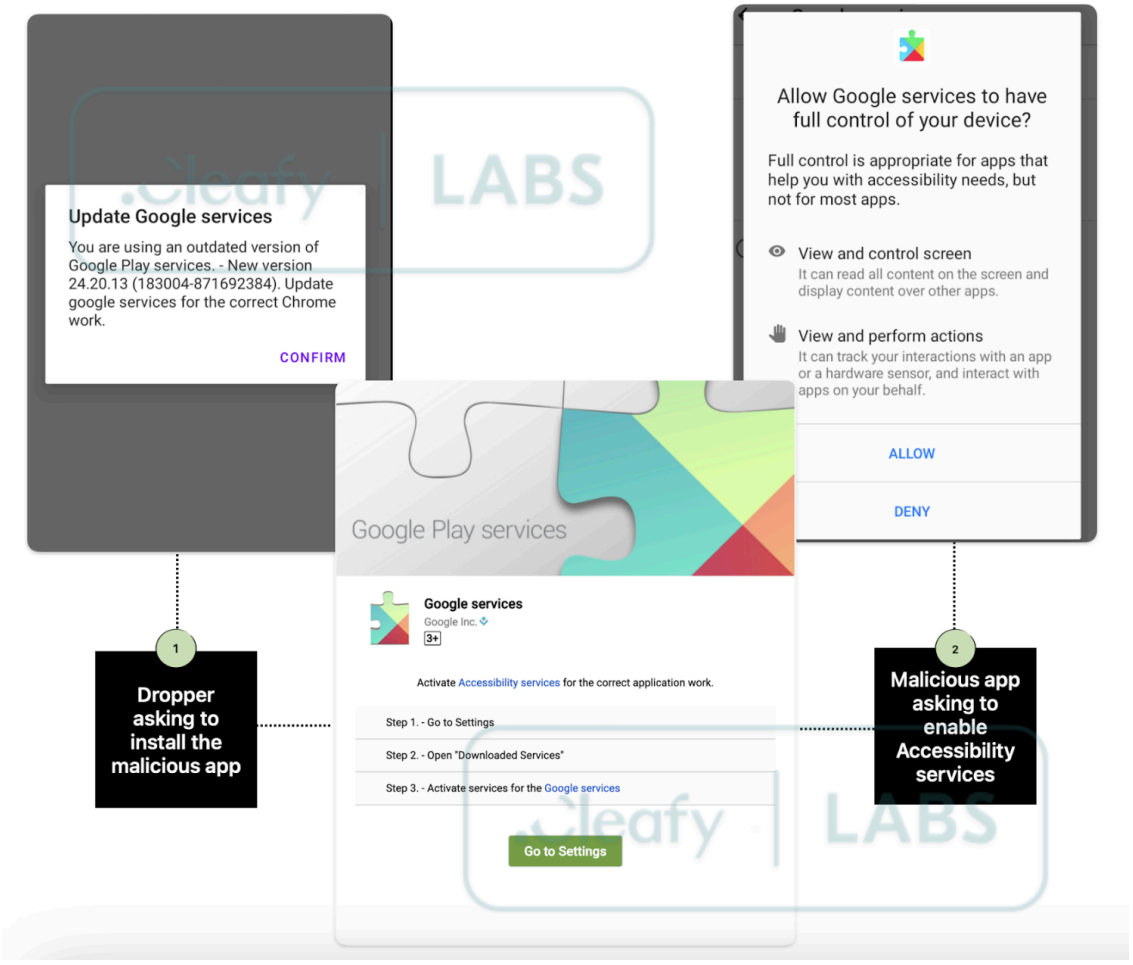


Figure 1 - Malicious app behaviour

Accessibility services are designed to assist users with disabilities by providing alternative ways to interact with their devices. These services can perform various actions, such as reading text aloud, automating repetitive tasks, and simplifying navigation. However, when exploited by malicious apps like TrickMo, these services can grant extensive control over the device.

This elevated permission allows TrickMo to perform various **malicious actions**, such as intercepting SMS messages, handling notifications to intercept or hide authentication codes, and executing HTML overlay attacks to steal user credentials. Additionally, the malware can dismiss keyguards and auto-accept permissions, enabling it to

integrate seamlessly into the device's operations. These capabilities allow TrickMo to conduct financial fraud, making it extremely difficult to detect and remove from the infected device.

Anti-analysis Mechanisms Dropper

Android droppers are applications used to install additional apps on the device. This type of application contains only one functionality, i.e. the update functionality, that may be abused by TAs. Through this functionality, TAs can deceive the user by proposing an update for the application, allowing the installation of a second, possibly malicious, application.
As noted in the previous paragraph, in this case, the dropper is disguised as a bogus Chrome Browser APK, which includes another APK infected with the TrickMo malware. This malicious APK is stored in one of its resources, specifically the file *assets/base.apk*.

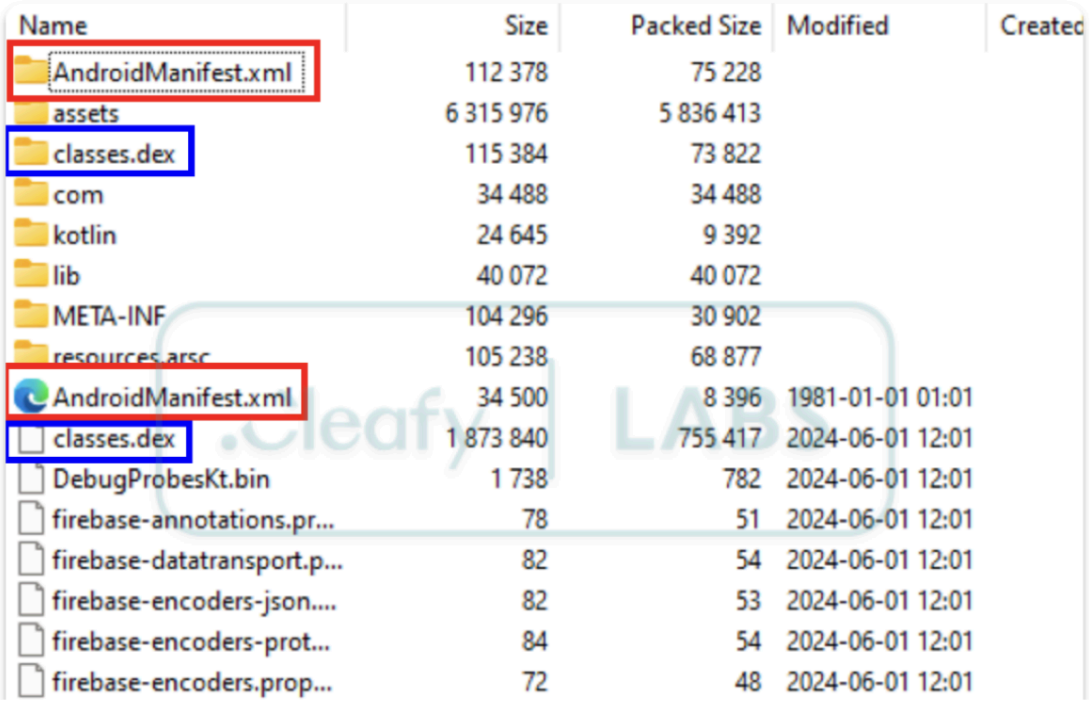
```
InstallDropSession.this.addApkToInstallSession("base.apk", session);
Intent intent = new Intent(InstallDropSession.this, (Class<?>) InstallDropSession.class);
intent.setAction(InstallDropSession.PACKAGE_INSTALLED_ACTION);
session.commit(PendingIntent.getActivity(InstallDropSession.this, 0, intent, 33554432).
```

Figure 2 - Dropper code installing malicious APK

Droppers can be highly effective in perpetrating attacks. Not only do they disguise themselves as legitimate applications, but they are also developed with minimal fingerprinting (even in terms of permissions requested during installation) to minimise their detection. Moreover, in this specific case, the analysed dropper employs the same anti-analysis protections discussed in the following paragraphs to further minimise detection.

Malformed ZIP

One of the recently introduced anti-analysis mechanisms employed by the latest TrickMo variant involves using **malformed ZIP files.** In this tactic, the APK file is manipulated by adding directories with the same names as crucial files, such as AndroidManifest.xml and classes.dex.



Name	Size	Packed Size	Modified	Created
AndroidManifest.xml	112 378	75 228		
assets	6 315 976	5 836 413		
classes.dex	115 384	73 822		
com	34 488	34 488		
kotlin	24 645	9 392		
lib	40 072	40 072		
META-INF	104 296	30 902		
resources.arsc	105 238	68 877		
AndroidManifest.xml	34 500	8 396	1981-01-01 01:01	
classes.dex	1 873 840	755 417	2024-06-01 12:01	
DebugProbesKt.bin	1 738	782	2024-06-01 12:01	
firebase-annotations.pr...	78	51	2024-06-01 12:01	
firebase-datatransport.p...	82	54	2024-06-01 12:01	
firebase-encoders-json....	82	53	2024-06-01 12:01	
firebase-encoders-prot...	84	54	2024-06-01 12:01	
firebase-encoders.prop...	72	48	2024-06-01 12:01	

Figure 3 - Malformed ZIP file

This clever obfuscation strategy can cause an unzip operation to overwrite these critical files, potentially hindering subsequent analysis. When security researchers or automated analysis tools attempt to extract and examine the contents of the APK, the malformed structure can lead to errors or incomplete extractions, significantly complicating the analysis process and providing TrickMo with an additional layer of evasion.

```
checkdir error:  resources.arsc exists but is not directory
                 unable to process resources.arsc/drawable/notification_bg_low.xml.
  inflating: classes.dex/drawable/animated_vector_autorenew.xml
  inflating: AndroidManifest.xml/drawable/abc_textfield_search_material.xml
checkdir error:  resources.arsc exists but is not directory
                 unable to process resources.arsc/drawable/btn_checkbox_checked_to_unchecked_mtrl_animation.xml.
 extracting: classes.dex/mipmap-xhdpi/ic_launcher.png
 extracting: AndroidManifest.xml/mipmap-xhdpi/ic_launcher2.png
replace AndroidManifest.xml? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

Figure 4 - Malformed ZIP file extraction

Despite this hindrance, the AndroidManifest.xml file can still be retrieved using apktool, even though the malformed ZIP technique can obstruct analysis performed by some of the most common tools, such as JADX. Apktool's ability to decompile and decode APK files allows researchers to bypass this obfuscation method and gain critical insights into the malware's structure and behaviour, especially after retrieving the Android Manifest File. The latter reveals a set of suspicious permissions commonly associated with Android banking trojans. These permissions enable the malware to perform various malicious activities, including intercepting communications, accessing sensitive data, and manipulating device settings. Moreover, the file contains several suspicious activities, services, and receivers related to the package dreammes.ross431.in.

```
<receiver android:name="dreammes.ross431.in.cnEdB0ghNnGaog">
    <intent-filter android:priority="1000">
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
        <action android:name="android.intent.action.USER_PRESENT" />
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.PHONE_STATE" />
        <action android:name="android.intent.action.NEW_OUTGOING_CALL" />
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED" />
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED" />
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
        <action android:name="android.provider.action.DEFAULT_SMS_PACKAGE_CHANGED" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.PACKAGE_ADDED" />
        <data android:scheme="package" />
    </intent-filter>
</receiver>
```

Figure 5 - Receiver with suspicious permissions

Unpacking

Despite multiple references within the application Manifest File to the package dreammes.ross431.in, this package is not present in the original APK. This absence suggests that the application is packed, a tactic commonly employed to conceal malicious components. Specifically, the packer used in this instance is identified as **JSONPacker**, which effectively hides the malicious DEX file containing critical code for various actions, services, and receivers.

When the app is launched on the device, the payload is unpacked through the class *com.turkey.inner.Uactortrust*, where the malicious DEX file is retrieved, decrypted, and loaded from the path */data/user/0/dreammes.ross431.in/app_inflict/wF.json*.

Figure 6 - Unpacking code

Malware Features

The unpacked DEX file reveals the app's malicious functionalities. The malware can respond to a series of commands issued by the C2 server, allowing it to perform all the operations necessary to carry out financial fraud and other malicious activities, such as:

- Intercepting SMS messages by changing the default SMS application.
- Retrieves all photos stored on the device.
- Record screen activity and enable remote access and control.
- Perform clicks and gestures on the device.
- The complete list of commands is reported in the Appendix of this document.

Particularly interesting is the functionality of performing HTML **Overlay Attacks**. The malware retrieves the list of the applications installed on the infected device; then, the C2 server, after receiving the list, will send a command labelled "**SaveHtml**" accompanied by the package name and an overlay URL. The URL will point to an HTML file later used as an HTML Overlay Injection page.

```
public JSONArray getInstalledApps(boolean onlyNotSystem) {
    JSONArray array = new JSONArray();
    try {
        List<PackageInfo> list = this.packageManager.getInstalledPackages(0);
        for (int i = 0; i < list.size(); i++) {
            PackageInfo item = list.get(i);
            if (onlyNotSystem) {
                if ((item.applicationInfo.flags & 1) == 0 && !item.packageName.equals(this.context.getPackageName()))

                    array.put(item.packageName);
                }
            } else if (!item.packageName.equals(this.context.getPackageName())) {
                array.put(item.packageName);
            }
        }
    } catch (Exception e) {
        Mixer.m133d(String.valueOf(System.currentTimeMillis()));
    }
    return array;
}
```

Figure 7 - Method to retrieve the list of installed apps

Command-and-Control (C2) Server Communication

The analysed TrickMo Android banking trojan communicates with its command-and-control (C2) server using the HTTP protocol, specifically through a domain extracted from the malware configuration. This communication channel relays commands from the attackers to the infected device and exfiltrates sensitive data. Through this C2 server, the attackers can manage the malware's activities, receive stolen information, and issue new instructions to the infected devices, ensuring continuous and dynamic control over the compromised systems.

```
package dreammes.ross431.in;

/* loaded from: classes3.dex */
public class Constants {
    public static final String KEY =
"MIIBVgIBADANBgkqhkiG9w0BAQEFAASCAUAwggE8AgEAAkEA0Tx+KHjj2bh9WENk/XlN4IdIchJC27maO+ENOfkAAeUQ2a32
FDv3oC3KHcXq/sllnpjeJcmfGNobewYotWLIwIDAQABAkBu8CAD3XURrxvmfHVDX9nGvnPO55fDYHlCXmTg5AD4/0Gjg5eG5m
NLDFTORTDcrrxah0V1kH14ViM33+8mm1BAiEA6/EG2mwPq/wq90Xzh4rSAhnF9T85yAUthyQx1tBWMrkCIQDjBkKnoIefoAjY
vpxPeWi4cbcJR/HlOuwFly3jStuuwIhAJToj9y2qsVu52ccnPCEqrpsrcV02/DjY7KRI2tiHaTxAiEAuiQ87MJj0U7fUduzLr
Il6e1bHWH/BWvtb5wIRO/w9cCIQC+yXRH0iUet2gLq4NqZxj155wMsjbBo+zEP0FOLUSb3g==";
    public static final String LAUNCH_APP = "com.android.chrome";
    public static String SERVER = "[redacted]";
    public static long INTERVAL = Long.parseLong("60");
    public static boolean SHOW_FIRST_DIALOG = true;
    public static boolean USE_ACCESSIBILITY = true;
}
```

Figure 8 - Constants class with C2 information

The initial message sent to the C2 server consists of an **HTTP POST** request to the endpoint */c*. The body of this request contains a **JSON file** with detailed information about the infected device, including the phone number, model, and a comprehensive list of apps installed on the device. This information enables the attackers to tailor their malicious activities to the specific characteristics of each infected device, enhancing the effectiveness of their campaigns and ensuring continuous and dynamic control over the compromised systems.

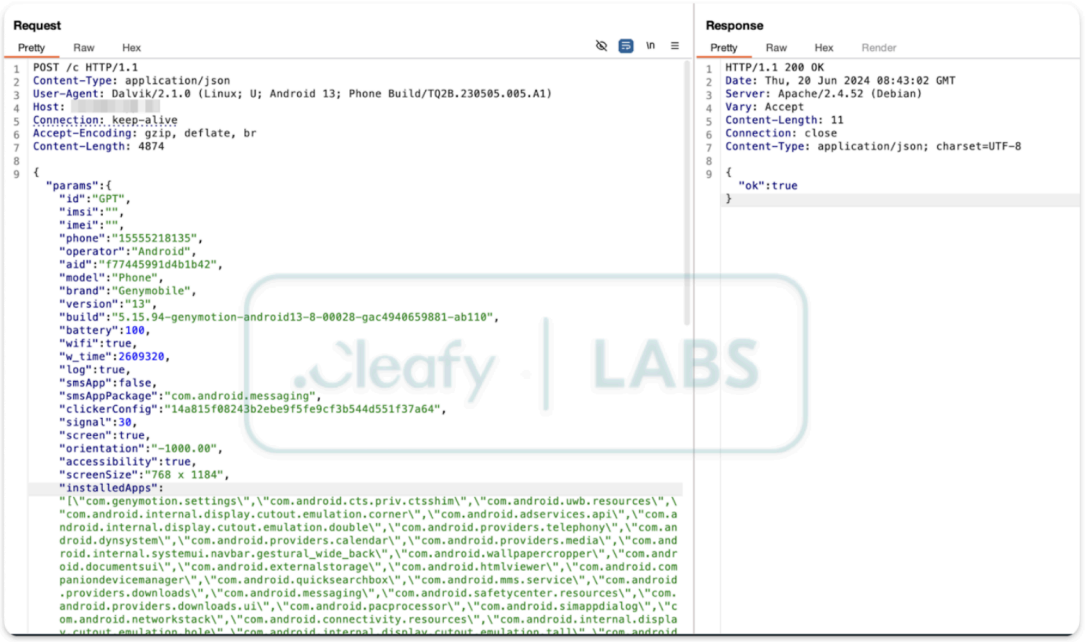


Figure 9 - HTTP request to the /c endpoint

Subsequently, the malware transmits another message to the C2 server to request the **Clicker configuration**.

Clicker configuration

The Clicker configuration showcases a sophisticated method of controlling an infected device via the Accessibility Service. To exploit it, the malware combines a class named Clicker with a clicker.json file saved in the assets directory. The file can also be updated by downloading new versions from the C2 server.

Figure 10 - HTTP request to the /config endpoint

The JSON file contains the package names on which the Accessibility service operates an “auto-click” functionality. For example, one operation that may be performed is automatically accepting permissions for the malware on the device.

Figure 11 - Auto acceptance of permissions through the Accessibility Service

This configuration is crucial as it enables the execution of operations that exploit the Accessibility Service, allowing the attackers to perform automated actions on the device. The Clicker configuration targets a mix of system applications and utility services, including settings, package installers, and system managers. This indicates TrickMo's intent to gain profound control over device configurations by disabling security features and enabling permissions without user consent.

The actions defined in the *Clicker.json* file can be categorised into several types of payloads:

- Blocking system updates (e.g., Samsung Update).
- Disabling security features.
- Preventing the uninstallation of certain apps.

The languages and text filters in the JSON configuration suggest a focus on German and English-speaking users. Phrases like "Aktivieren," "App-info," and "Deinstall" indicate that the malware is designed to interact with devices set to German, hinting at potential victim geolocations in Germany, Austria, or Switzerland. Similarly, English phrases such as "Activate," "App info," and "Uninstall" suggest that the malware also targets devices set to English, indicating potential victims in the United Kingdom and the United States.

Figure 12 - German language found in the Clicker configuration

Moreover, the same infrastructure leveraged for controlling the botnet and retrieving additional configurations is also employed to store the data exfiltrated from the victim’s device, including logs, credentials, and photos. The following section discusses the specifics of the information that can be extracted in depth. The analysis revealed that the C2 server does not provide an authentication mechanism to access the obtained data, meaning third parties

could also access it. Although a login page is provided, the data is accessible by knowing specific endpoints on the server, which are easily guessable, thereby further exposing the victims.

Data leak from C2 Server

This chapter delves into the numerous misconfigurations in the Command and Control (C2) infrastructure leveraged by TrickMo. These misconfigurations, if exploited, provide access to a significant portion of the data exfiltrated from the infected devices of victims targeted by TrickMo distribution campaigns. The data stored within the C2 server encompasses a wide range of sensitive information, including personal photos, documents, connection logs, credentials, and more, **totalling 12 GB of files.** This analysis underscores the critical security lapses in the C2 setup, which not only jeopardise the privacy of victims but also highlight the operational deficiencies of the TAs behind TrickMo.

Our in-depth analysis of the command-and-control (C2) server unveiled several critical endpoints that play pivotal roles in the malware's operation.

One of the folders contains a list of IP addresses presumably associated with the compromised devices. This list enables the attackers to keep track of infected devices and manage their operations more effectively.

Name	Last modified	Size	Description
Parent Directory		-	
4/	2023-06-03 09:22	-	
11/	2023-06-02 12:32	-	
16/	2023-06-03 09:31	-	
20/	2023-06-03 14:52	-	
27/	2023-06-03 15:02	-	
39/	2023-06-03 15:08	-	
41/	2023-06-03 15:33	-	
45/	2023-06-03 15:43	-	
49/	2023-06-03 16:02	-	
51/	2023-06-15 09:49	-	
53/	2023-06-15 09:53	-	
67/	2023-06-15 09:56	-	
72/	2023-06-15 10:02	-	
76/	2023-06-15 10:22	-	
81/	2023-06-15 10:25	-	
85/	2023-06-15 10:28	-	
137/	2023-06-15 10:31	-	
141/	2023-06-15 10:44	-	
151/	2023-06-15 10:53	-	

Figure 13 - Directory listing of the logs stored on the C2 server

Another folder reveals a list of subdirectories identified by numerical filenames. Each subdirectory contains detailed logs of operations performed on the compromised devices, providing the attackers with a comprehensive record of their malicious activities. These logs can include information on intercepted communications, executed commands, and other actions carried out by the malware, further highlighting the sophisticated level of control the attackers maintain over the infected devices.

Журнал для бота 11

22.05.2023, 08:14:58 Accessibility: Yes
22.05.2023, 08:14:58 Интересных приложений: **13** шт.
22.05.2023, 08:14:58 Список приложений сохранил в комментариях
22.05.2023, 08:14:58 Подходящих инжектов: **7** шт.
22.05.2023, 08:14:58 Overlays: No
22.05.2023, 08:14:58 Locale: de_DE
22.05.2023, 08:14:58 Отправили SaveHtml для com.amazon.mShop.android.shopping
22.05.2023, 08:14:58 Отправили SaveHtml для com.paypal.android.p2pmobile
22.05.2023, 08:14:58 Отправили SaveHtml для com.microsoft.office.outlook
22.05.2023, 08:14:58 Отправили SaveHtml для com.netflix.mediaclient
22.05.2023, 08:14:58 Отправили SaveHtml для com.ubercab
22.05.2023, 08:14:58 Отправили SaveHtml для de.ingdiba.bankingapp
22.05.2023, 08:14:58 Отправили SaveHtml для com.ebay.mobile
24.05.2023, 14:01:29 saveHtml id: com.amazon.mShop.android.shopping, http code: 200, bytes: 125313
24.05.2023, 14:01:29 saveHtml id: com.paypal.android.p2pmobile, http code: 200, bytes: 10050
24.05.2023, 14:01:29 saveHtml id: com.microsoft.office.outlook, http code: 200, bytes: 53550
24.05.2023, 14:01:29 saveHtml id: com.netflix.mediaclient, http code: 200, bytes: 74377
24.05.2023, 14:01:29 saveHtml id: com.ubercab, http code: 200, bytes: 49644
24.05.2023, 14:01:29 saveHtml id: de.ingdiba.bankingapp, http code: 200, bytes: 36956
24.05.2023, 14:01:29 saveHtml id: com.ebay.mobile, http code: 200, bytes: 149145
26.05.2023, 13:59:25 Отправили боту конфиг с хешем 0c0cc06bee1be245ccc4677ee0ad91788e7dafbc
27.05.2023, 09:13:14 Необходимо включить Overlays!
27.05.2023, 09:13:14 Конфиг с хешем 0c0cc06bee1be245ccc4677ee0ad91788e7dafbc: команда отправлена 27.05.2023, 07:49:54, выполнена 27.05.2023, 07:54:35
27.05.2023, 13:21:15 Конфиг с хешем 0c0cc06bee1be245ccc4677ee0ad91788e7dafbc: команда отправлена 27.05.2023, 13:15:58, выполнена 27.05.2023, 13:32:02
27.05.2023, 13:21:15 Пока нет смысла чекать инжекты
28.05.2023, 12:48:15 Отправили боту конфиг с хешем 33fac1717682fbf8230dd4faf450cf72b6e79529

Figure 14 - Log file on the C2 Server

Furthermore, the C2 provides HTML files used in HTML overlay attacks. These files include deceptive login pages for various services, including bank accounts such as ATB Mobile and Alpha Bank and cryptocurrency platforms like Binance. By displaying these overlays, TrickMo can effectively phish for user credentials, enabling attackers to gain unauthorised access to sensitive accounts.

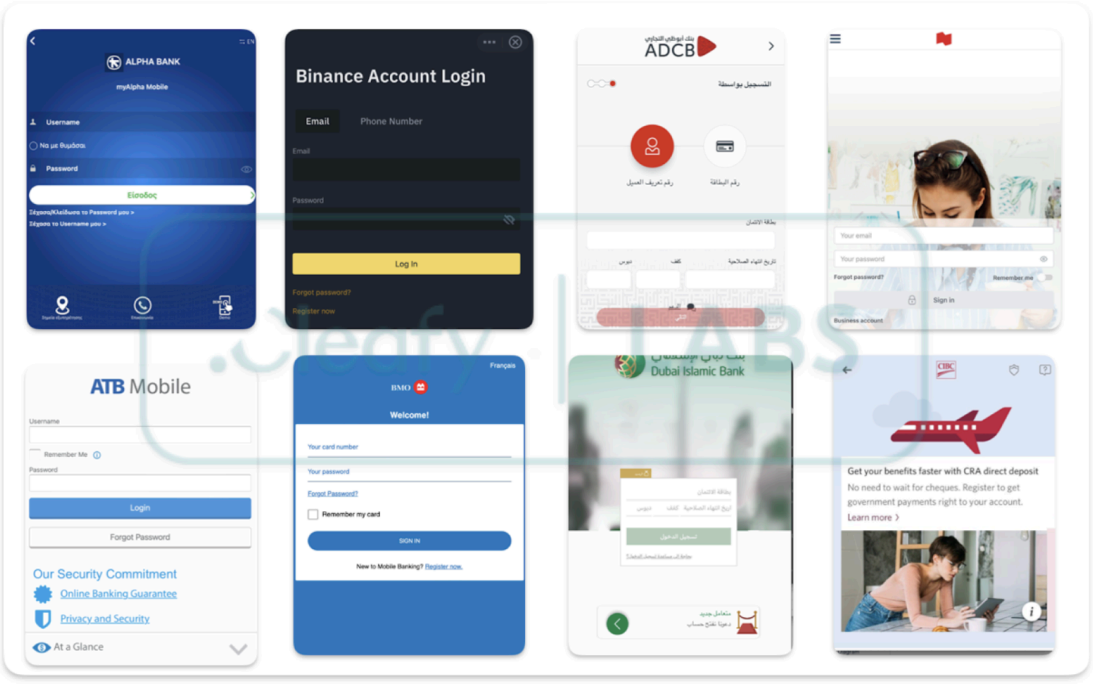


Figure 15 - Overlaid HTML pages

A third endpoint is responsible for storing CSV files that contain captured username and password combinations. This endpoint serves as a centralised repository for the stolen credentials, allowing attackers to access a wealth of sensitive login information. This organised storage facilitates using these credentials in further fraudulent activities, such as unauthorised access to financial accounts and identity theft.

2309test

name	username	password	note
amazon.com	jo[redacted]org	/M[redacted]	
amazon.com	m[redacted]om	ch[redacted]	
bitcoinera-app.com	m[redacted]om	Jjh[redacted]	
discord.com	m[redacted]om	ch[redacted]	
Facebook	m[redacted]om	ch[redacted]	
google.com	kk[redacted]	/kk[redacted]	
live.com	ch[redacted]	qN[redacted]	
live.com	m[redacted]om	/Cl[redacted]	
netflix.com	m[redacted]om	/S[redacted]	
nike.com	m[redacted]om	Ch[redacted]	
PayPal	m[redacted]om	/S[redacted]	
snapchat.com	m[redacted]om	/S[redacted]	
Spotify: música y podcasts	m[redacted]om	ch[redacted]	
spotify.com	m[redacted]	ch[redacted]	
spotify.com	m[redacted]om	ch[redacted]	
Tinder - citas	ar[redacted]XDFCSUh	mc[redacted]com	
twitch.tv	ch[redacted]	ch[redacted]	
twitter.com	m[redacted]om	ch[redacted]	
Wish: compra y ahorra	m[redacted]om	ch[redacted]	

Figure 16 - Stolen credentials

Finally, a specific folder is dedicated to storing ZIP files containing all the images extracted from the compromised devices. These images include potentially sensitive personal pictures, identification documents, and financial information. Attackers can use this data to gather further intelligence about the victims, significantly compromising their privacy and increasing security risks for the affected individuals.

Figure 17 - Stolen pictures

Hence, the data leak from TrickMo's Command and Control (C2) infrastructure exposes a wealth of sensitive information. This breach amplifies the threat of further exploitation by TAs or third parties. When a TA gains access to credentials and sensitive photos, the range of potential malicious activities expands significantly. With user credentials, including usernames and passwords, the TA can easily **infiltrate various online accounts**, such as banking, email, social media, and other personal services. This access enables **direct financial theft, unauthorised money transfers,** and **fraudulent purchases**. Compromised email accounts can also be leveraged to **reset passwords** for other services, further extending the attacker's reach.

Sensitive photos, such as images of passports, credit cards, and personal identification documents, can be used to commit **identity theft**. The TA can create fake identities or verify stolen accounts to bypass security checks. These photos can also be exploited for **Social Engineering attacks**, blackmail, or extortion. For instance, personal or compromising pictures can be used to coerce victims into paying ransom or performing actions beneficial to the attacker.

Furthermore, combining credentials and sensitive photos enhances the TA's ability to perform highly targeted **phishing attacks**. Using personal information and images, the attacker can craft convincing messages that trick victims into divulging even more information or executing malicious actions. Exploiting such comprehensive personal data results in immediate financial and reputational damage and long-term consequences for the victims, making recovery a complex and prolonged process.

Conclusions

The analysed TrickMo sample and its ability to intercept communications, manipulate device settings, and access sensitive data underscore the advanced nature of this sample and the persistent efforts of its creators to evade detection (e.g., malformed ZIP and JSONPacker) and enhance its capabilities. However, this analysis also gave us

more insights about the C2 server, which revealed the structure and organisation of exfiltrated data, highlighting critical endpoints that store and manage stolen information.

Moreover, the TAs behind TrickMo made multiple OPSEC mistakes, publicly exposing part of the exfiltrated data (including pictures exfiltrated from infected devices by gaining access to these endpoints). These images contained valuable information such as passports, credit card details, and other personal documents, demonstrating the extent of data compromise and the level of control the attackers have over the infected devices.

This shallowness, leaving leaked data publicly available, should not be undertaken because it exposes victims to multiple threats that span from identity theft, fraud, extortion, etc. However, it's worth mentioning that the impact of that information is not limited to cyberspace but could also involve a physical threat, with even more risks. Moreover, it's crucial to highlight that the exposed information can be exploited by various TAs who may access the leaked data for purposes entirely different from the original intent (banking fraud in the case of TrickMo). These bad actors can leverage the information for various malicious activities, each potentially more damaging than the last.

Under these circumstances, it is crucial to re-evaluate the risks associated with such threats, emphasising the consequences of these actions in the medium and long term rather than focusing on the near future. This comprehensive assessment underscores the importance of robust data protection and systems with enhanced predictive capabilities to prevent malicious actions arising from such threats.

Appendix 1: Malware Commands

ID	Command	Description
1	Server	Configures the C&C (Command and Control) server details.
2	Interval	Sets the frequency at which the malware communicates with the C&C server.
3	DeleteAll	Deletes all data or traces of the malware from the infected device.
4	SelfDestroy	Triggers the self-destruction of the malware to remove itself from the device.
6	SetSmsApp	Changes the default SMS application on the infected device.
7	SetPhone	Modifies phone settings or configurations.
8	SendSms	Sends SMS messages to specified numbers.
9	ShowPopup	Displays a popup message on the infected device.
10	ActiveInterval	Sets the active time interval for the malware's operations.

ID	Command	Description
11	RequestInfo	Requests specific information from the infected device, such as contacts or messages.
12	GetAllPhotos	Retrieves all photos stored on the infected device.
13	GetPhoto	Retrieves a specific photo from the infected device.
14	VNC	Enables remote access and control of the infected device using Virtual Network Computing (VNC) technology.
15	ScreenRecord	Records the screen activity of the infected device.
16	LoadModule	Downloads and executes additional modules or payloads.
17	StartOrInstall	Starts a specific application or installs a new one on the infected device.
18	SetClickerConfig	Configures the clicker settings for automated clicking actions.
19	ShowDialog	Displays a dialog box on the infected device.
20	ShowNotification	Shows a notification on the infected device.
21	SetVars	Sets variables or parameters for the malware's operation.
22	ReadSms	Reads SMS messages from the infected device.
23	RequestIgnoreBatteryOptimizations	Requests the device to ignore battery optimization settings for the malware.
24	ShowCover	Displays a full-screen cover to hide malicious activities.
25	UnlockScreen	Attempts to unlock the device's screen.
26	DisableNotifications	Disables notifications to prevent the user from seeing alerts about the malware.
27	PressHome	Simulates pressing the home button.
28	PressBack	Simulates pressing the back button.
29	OpenSetNewPasswordSettings	Opens the settings to set a new password on the device.
30	SaveHtml	Saves HTML content to the device.
31	PressRecents	Simulates pressing the recents button to show recent apps.
32	OpenPowerDialog	Opens the power dialog for options like shutdown or restart.

ID	Command	Description
33	KillBackgroundProcesses	Terminates background processes running on the device.
34	RequestOverlayPermission	Requests permission to draw over other apps.
35	RequestPermissions	Requests additional permissions from the user.
36	OpenGoogleProtectSettings	Opens the settings for Google Play Protect.
37	TakeScreenshot	Captures a screenshot of the device's display.
38	Update	Updates the malware to a newer version.
39	OpenAccessibilitySettings	Opens the accessibility settings menu.
40	GetAllVideos	Retrieves all video files stored on the infected device.
41	GetVideo	Retrieves a specific video from the infected device.
42	OpenNotificationSettings	Opens the notification settings menu.
43	OpenAppSettings	Opens the settings for a specific application.
44	SendUssd	Sends a USSD code (Unstructured Supplementary Service Data) to perform actions like balance checks or top-ups.
45	ReadCalls	Reads the call logs from the infected device.
46	ChangeIcon	Changes the icon of the malware to disguise its presence on the device.

Appendix 2: Indicator of Compromise (IOCs) / TLP-AMBER

We have identified several Indicators of Compromise (IOCs) that provide critical insights into the TA infrastructure and behaviour. However, we have decided to keep these IOCs confidential due to significant misconfigurations within the Command and Control (C2) infrastructure leveraged by TrickMo. This decision stems from the potential risk that additional malicious actors could exploit these vulnerabilities to harvest a substantial amount of sensitive information from the compromised victims within the botnet.

While we acknowledge that withholding IOCs might not be well-received within the cybersecurity community, we prioritise the privacy and security of affected individuals. Consequently, we will disseminate these IOCs under a TLP-AMBER protocol exclusively to trusted entities such as law enforcement agencies, reputable CERTs, and banking institutions.

Researchers and analysts recognised as trusted entities within the community can request access to the IOCs by contacting us at labs@cleafy.com. By adopting this cautious approach, we aim to mitigate further exploitation

risks while ensuring that critical threat intelligence is shared responsibly with those who can act upon it effectively.
