BingoMod: The new android RAT that steals money and wipes data

cleafy.com/cleafy-labs/bingomod-the-new-android-rat-that-steals-money-and-wipes-data

Alessandro Strino, Simone Mattia

Key Points

- At the end of May 2024, the Cleafy TIR team discovered and analysed a new Android RAT. Since we didn't find references to any known families, we decided to dub this new family **BingoMod**.
- The main goal of **BingoMod** is to initiate money transfers from the compromised devices via **Account Takeover** (ATO) using a well-known technique, called **On Device Fraud** (ODF). It aims to bypass bank countermeasures used to enforce users' identity verification and authentication, combined with behavioural detection techniques applied by banks to identify suspicious money transfers.
- After installation on the victim's device, BingoMod leverages various permissions, including **Accessibility Services**, to quietly steal sensitive information, including credentials, SMS messages, and current account balances. In addition, the malware is equipped with active features that allow it to conduct overlay attacks and remotely access the compromised device using VNC-like functionality.
- After a successful fraudulent transfer, the **infected device is typically wiped**, removing any traces of BingoMod activity to hinder forensic investigations.
  Another interesting element that emerged during the BingoMod investigation is related to target devices, which include three languages: **English, Romanian**, and **Italian**.
- At the time of writing, BingoMod is in a development phase, where developers are experimenting with obfuscation techniques to **lower its detection rate** against AV solutions. From the whole sample collected, what has emerged is the will to try multiple anti-analysis configurations rather than making the malware more complex in terms of functionalities.
- According to the comments identified within the malware code, developers may be **Romanian** speakers.

Executive Summary

At the end of May 2024, a new Android RAT appeared in Cleafy's telemetries.
Due to the lack of information and the absence of a proper nomenclature for this malware family, we decided to dub it **BingoMod** to track it inside our Threat Intelligence taxonomy. This nomenclature is based on the malware's core component, known at an early stage as "ChrUpdate" but later renamed "**BingoMod**".

BingoMod belongs to the modern RAT generation of mobile malware, as its remote access capabilities allow Threat Actors (TAs) to conduct **Account Takeover** (ATO) directly from the infected device, thus exploiting the **On Device Fraud** (ODF) technique. This consolidation of this technique has already been seen recently by other banking trojans, such as Medusa, Copybara, and Teabot.

These techniques have several advantages: they require less skilled developers, expand the malware's target base to any bank, and bypass various behavioural detection countermeasures put in place by multiple banks and financial services. However, this advantage does not come for free, one of the drawbacks of this technique relies on a live operator that is required to insert and authorise a money transfer, which implicitly means lowering its scale factor.

BingoMod is similar to the Brata's operation model in using device wiping after a successful fraudulent transfer. This self-destruction mechanism is designed to eradicate any trace of BingoMod's activity on the infected device, effectively hindering forensic analysis and making it more challenging for researchers to identify and attribute incidents. This tactic is relatively rare in the Android landscape, suggesting that the developers of BingoMod could be aware of Brata's methods and have incorporated them into their methodology.

Moreover, it's also worth mentioning that this sample is in its early stage of development, and it's still hard to predict which direction will be taken. However, the developers’ commitment to attempting obfuscation techniques underlines their intention to pursue a more opportunistic approach than a tailored one already seen in malware like SharkBot or Gustuff.

The following table represents a summary of the TTP behind BingoMod campaigns:

First Evidence	**May 2024**
State	Active (July 2024)
Affected Entities	Retail banking
Target OSs	Android Devices
Target Countries	IT
Infected Chain	Social Engineering (smishing) -> Side-loading
Fraud Scenario	On-Device Fraud (ODF)
Preferred Cash-Out	Instant/SEPA transfer
Amount handled (per transfer)	Up to 15K EUR

All detected samples provided in the Appendix are **disguised** as legitimate **security tools to protect the device**.

Figure 1 - Common decoy used by BingoMod

Figure 1 - Common decoy used by BingoMod

Technical Analysis

As previously mentioned, the malicious app is distributed via smishing and often masquerades as a legitimate antivirus application. After installation on the victim's device, BingoMod prompts the user to activate **Accessibility Service**s, disguising the request as necessary for the app to function correctly. If the user grants the requested permissions, the APK begins to **unpack** itself, executing its malicious payload. Once the operation is completed, the apps still lock out the user from the main screen to collect device information and set up the C2 communication channel.

Figure 2 - Starting phase of BingoMod

Figure 2 - Starting phase of BingoMod

After activation, BingoMod's **background** functions act, aiming to provide sensitive data to the actors behind the malware. In detail, two features typical of banking Trojans are used:

- **Key-logging**: This function exploits the Accessibility Services to steal sensitive information displayed on the device screen or entered by the user, such as login credentials or account balances.
- **SMS interception**: This function starts monitoring SMS messages, often used by financial institutions for transaction authentication numbers (TANs).

Figure 3 - Keylogging routine

Figure 3 - Keylogging routine

As mentioned earlier, BingoMod's main objective is to initiate money transfers directly from compromised devices (ODF). Therefore, the malware implements several **remote control functionalities**. To do this, BingoMod establishes a **socket-based** connection with the command and control infrastructure (C2) to receive commands that TAs want to perform on the compromised device.

The malware provides around **40 remote control functions**, among the most relevant are indeed related to the real-time screen control that is implemented in the following way:

- **VNC-like routine**: Leveraging Android's Media Projection API, TAs capture screenshots of the victim's device screen at regular intervals, giving them a complete overview of what is happening on the screen.
- **Screen interaction**: Leveraging Accessibility Service, BingoMod provides several commands to remotely control the infected device screen, allowing TAs to operate the device as if they were physically in front of it. These functionalities include clicking buttons, filling in forms, and navigating between apps.

For this purpose, BingoMod uses **two separate communication channels**: a socket-based channel for command transmission (in the case of VNC start/stop) and an **HTTP-based** channel for **image transfer**.

Figure 4 - C2 communication scheme during VNC routine

Figure 4 - C2 communication scheme during VNC routine

To better illustrate this process, we created a **simulated C2 infrastructure**. This setup includes a socket-based C2 server for remote control and an HTTP-based "VNC" server to capture and display real-time screenshots the infected device sends.

Figure 5 - VNC in action (TAs' point of view)

On the malware side, the VNC routine abuses Android's Media Projection API to obtain real-time screen content. Once received, this is transformed into a suitable format and transmitted via HTTP to the TAs' infrastructure. An exciting feature of the routine is leveraging Accessibility Services to impersonate the user and enable the screen-casting request, exposed by the Media Projection API.

Figure 6 - VNC routine

Figure 6 - VNC routine

Once the VNC-like routine is activated, TAs can interact with the device using **dedicated commands**. These include, for example, opening a specific application (<*LAUNCH*>), moving to a particular area on the screen (<*MOVEAT*>), clicking a particular area (<*CLICKAT*>) or writing in a particular text box (<*SETTEXT*>).

In addition to real-time screen control, the malware shows **phishing capabilities** through **Overlay Attacks** and fake notifications. Unusually, overlay attacks are not triggered when specific target apps are opened but are initiated directly by the malware operator. Still, in the context of phishing, TAs can also send SMS messages from the compromised device; this functionality can be used to spread the malware further.

Figure 7 - Overlay in action

Figure 7 - Overlay in action

Finally, the malware implements some **security measures** to improve its **resilience**: it hinders editing system settings, especially one regarding the malware itself, and can **block** the activity of specified applications set by the actors via a dedicated command. If this is not enough, BingoMod can also **uninstall** arbitrary applications. For instance, this feature can be used to prevent security apps from detecting the presence of the malware itself.

As mentioned earlier, BingoMod's most notable security measure is its ability to **wipe the device** remotelywith a dedicated command. This feature can be implemented by BingoMod when it is a **device administrator** and is typically executed after a successful fraud. However, this functionality is **limited to the device's external storage only**, so we speculate that the complete wipe is performed by TAs directly from the device's system settings, leveraging BingoMod's remote access capabilities.

Figure 8 - Wipe Routine

Figure 8 - Wipe Routine

The entire BingoMod command set is provided in the appendix.

Malware Evolution

It’s worth considering that from the first sample until now, it has been possible to observe that developers are in an “experimental” phase, mainly on app obfuscation and packing process that aims to reduce its detection against AV solutions instead of equipping their code with advanced capabilities. This change is immediately observable using detection engines from VirusTotal, showing that early campaigns were easily detected, whereas the recent one dropped their detection rate.

Figure 9 - Detection rate dropped after obfuscation layers

Figure 9 - Detection rate dropped after obfuscation layers

As mentioned, functions and classes stayed mostly the same over time. However, the obfuscation employed lowers the overall detection rate. The figure below gives a code example that refers to the hiddenVNC procedure, where the malware is sending a screenshot to the C2, simulating a live stream.

Figure 10 - Code comparison between early and newer versions of BingoMod

Figure 10 - Code comparison between early and newer versions of BingoMod

In the upper part of this image, we can see an early version of the sample. Aside from some variable renaming, the overall code is understandable. However, the same function appears to be heavily obfuscated using **code-flattening** and **string obfuscation** techniques. After tweaking the code to resolve the switches, the overall structures are the same, strengthening the hypothesis that TAs rely on obfuscation over malware complexity.

Moreover, it’s worth comparing different code versions to analyse their changes when discussing malware evolution. As shown in Figure 11, there were no significant changes in the overall structure and functionality. However, an asynchronous callback mechanism has been introduced in the PingUtil class to send "alive" signals to the C2 server, giving information about the bot's status.

Figure 11 - Class name comparison between the earlier malicious BingoMod packages and newer versions

Figure 11 - Class name comparison between the earlier malicious BingoMod packages and newer versions

Another evidence confirming its developmental stage is the amount of log information left within the code. It is worth mentioning that although some comments have been deleted in future releases, some still refer to older package names. For instance, in release **1.4.3b**, references to package names from the first version (**1.0**) are still visible.

Another interesting element that emerged during the BingoMod investigation is related to target devices, which include three languages: **English**, **Romanian**, and **Italian**, here disguised as **com**.

Figure 12 - Switches to select the proper language device

Figure 12 - Switches to select the proper language device

However, it is possible to observe that the Romanian language is mixed with English words. For the Italian language, it has also been observed with a few typos. Those strings and the general code could be another sign that those versions are still in their “debug” phase.

Attribution

Details retrieved during the investigation led us to several speculations about the developers. For instance, some comments are in Romanian, but these comments have been removed in more recent versions. This suggests that the TAs have evolved, possibly incorporating developers from different countries, or they are trying to reduce indicators that can lead further LEA investigation to a specific region.

Figure 13 - Romanian comments left in the code

Figure 13 - Romanian comments left in the code

Moreover, a recent campaign involving version 1.4.3b was uploaded to VirusTotal from an IP address geolocating to the Romanian region. While this behaviour is plausible, it strengthens suspicions about active campaigns and areas involved in their attack.

Figure 14 - Suspicious upload of BingoMod from Romanian country

Figure 14 - Suspicious upload of BingoMod from Romanian country

Conclusion

BingoMod shows relatively straightforward functionalities commonly found in most contemporary RAT, such as HiddenVNC for remote control and SMS suppression to intercept and manipulate communication and logging user interactions to steal sensitive data. The emphasis on obfuscation and unpacking techniques suggests that the developers may lack the sophistication or experience of more advanced malware authors. This is evident in their reliance on basic obfuscation methods and the minimalistic approach to developing the malware’s functionalities. It is plausible that the developers aimed to quickly produce a functional piece of malware, prioritising speed over complexity.

One notable aspect of this malware is its device-wiping capability, triggered after a fraudulent transaction. This behaviour is reminiscent of the Brata malware, which also employed device-wiping to cover its tracks and hinder forensic analysis. However, the simplicity and rudimentary nature of the code suggests that this feature is more of an easy exit strategy rather than an indication of any direct lineage or connection to Brata. This action further supports the theory that the developers opted for straightforward and efficient methods to achieve their goals without delving into more complex development practices. Compared to more sophisticated threats such as Sharkbot, which incorporates an Automatic Transfer System (ATS) to automate fraudulent transactions, this malware needs to improve in terms of advanced capabilities.

The lack of automated components implies that scaling this approach would be challenging, as it requires direct operator interaction to perform Account Takeover (ATO) activities. This direct interaction often involves manipulating the victim or their device in real time, limiting the scale of potential attacks and exposing the operators to a higher risk of detection. The reliance on manual intervention underscores the limited technical capabilities behind this malware and indicates that it operates within a more traditional fraud paradigm. Interestingly, the current trend among mobile banking trojans, exemplified by threats like Copybara and Medusa, focuses on On-Device Fraud (ODF) through ATO. This shift highlights a broader trend within the malware landscape where the emphasis is placed on exploiting the device to conduct fraudulent activities rather than developing highly automated systems. The analysed malware aligns with this trend, focusing on direct interaction with the victim's device to carry out its malicious objectives. This approach, while less sophisticated, still poses significant risks to end-users and financial institutions due to the potential for substantial economic loss and the disruption of personal data security.

Appendix 1: Indicator of Compromise (IoC)

MD5	Version	App Name	Package Name
8b173081ea73ee0ed223d5703bb5fcd1	1	APP Protection	com.djokovic.chromeupdate
bb8a2e045fdc2017b2171ff57286b05c		Antivirus Cleanup	
3f6dfc31e152d39d52388ec7673f64d5		Chrome Update	
bd1f1a2e8ff984ce6d795d025bbccdb1			
41d1d5e16df294a24e36fd735076ef93			
38dc0f70fa3c76b28ba5ad06d84a3e08			
bdbec1c7c816b61b4ef9c76804d18f47			
7574c1cc849108f911652571a73e2447		InfoWeb	com.coffeestainstudios.goatsimulator
03b486cc13618d806a79d794ba138b43		SicurezzaWeb	com.ccandroid.suite
e9a58a77a042986ea5fdfdc6b2a396c0		WebSecurity	
5bf85b009e29f0af6218991942f32329		WebsInfo	com.bimiboo.coloring
802624f4d0169e949bf40b613824d967			com.halfbrick.joyride
1e850e735c649b1f80ba36c7b07a198a	1.1	WebInfo	com.tocaboca.tocahairsalon4
60d2350b8f5bd08e05612ed8c894af20	1.4.1b	APKAPPSCUDO	com.danza.perfectarcher
b4156ef9761f51dbac2d1104946dd3a8			com.kanko.negruzzi
a29c774dc6dc5f29d603f1b52fcdf241	1.4.3b		com.vonation.hitenhit
2788e87b8760ebdec67bce21899893d2	1.5.1		com.pescado.hitenhit
3534af6660e5ac844167fc3eef00bcc5	1.6.7		com.primo.eternalache
08878948f69846d2217290614b70c151	1.7.1		com.bleuinc.xperinz
75bee41937b00ab466d31bd9e7193b02	1.8.2		com.pelosi.polskaball

MD5	Version	App Name	Package Name
516ab57114f204eb24e690f56b9699c1	1.9.4		com.deco.canta

IP	Description
101.99.92[.]10	C2 Server
103.155.92[.]11	C2 Server

Appendix 2: Command Sets

Version 1	Version 1.4.1b	Version 1.4.3b	Version 1.5.1
<ACTIVITY>	<ACTIVITY>	<ACTIVITY>	<ACTIVITY>
<BLOCKAPP>	<BLOCKAPP>	<BLOCKAPP>	<BLOCKAPP>
-	-	-	<BLOCKCALL>
<BRIGHTNESS>	<BRIGHTNESS>	<BRIGHTNESS>	<BRIGHTNESS>
<CALLNO>	-	-	-
-	<CHECKPERM>	<CHECKPERM>	<CHECKPERM>
-	<CLEARNOT>	<CLEARNOT>	<CLEARNOT>
<CLICKAT>	<CLICKAT>	<CLICKAT>	<CLICKAT>
<CLICKNODE>	<CLICKNODE>	<CLICKNODE>	<CLICKNODE>
<DRAWVIEW>	-	-	-
<FAKESMS>	<FAKESMS>	<FAKESMS>	<FAKESMS>
<GETADMIN>	<GETADMIN>	<GETADMIN>	<GETADMIN>
<GETNODES>	<GETNODES>	<GETNODES>	<GETNODES>
<GETNOTIFYPERM>	<GETNOTIFYPERM>	<GETNOTIFYPERM>	<GETNOTIFYPERM>
<GETSMSPERM>	<GETSMSPERM>	<GETSMSPERM>	<GETSMSPERM>
<GETWRITEPERM>	<GETWRITEPERM>	<GETWRITEPERM>	<GETWRITEPERM>
<GRANT>	<GRANT>	<GRANT>	<GRANT>
-	<INPUFOCUS>	<INPUFOCUS>	<INPUFOCUS>
<INPUT>	<INPUT>	<INPUT>	<INPUT>
<LAUNCH>	<LAUNCH>	<LAUNCH>	<LAUNCH>
<LAUNCHA>	<LAUNCHA>	<LAUNCHA>	<LAUNCHA>

Version 1	Version 1.4.1b	Version 1.4.3b	Version 1.5.1
<MOVEAT>	<MOVEAT>	<MOVEAT>	<MOVEAT>
<MUTEDEV>	<MUTEDEV>	<MUTEDEV>	<MUTEDEV>
<NOTIFY>	<NOTIFY>	<NOTIFY>	<NOTIFY>
<OPTIMISATIONPERM>	<OPTIMISATIONPERM>	<OPTIMISATIONPERM>	<OPTIMISATIONPERM>
<OV>	<OV>	<OV>	<OV>
<PM>	<PM>	<PM>	<PM>
-	<REFRESHSMS>	<REFRESHSMS>	<REFRESHSMS>
<RGTGETWRITEPERM>	<RGTGETWRITEPERM>	<RGTGETWRITEPERM>	<RGTGETWRITEPERM>
-	<SELFUNINSTALL>	<SELFUNINSTALL>	<SELFUNINSTALL>
<SETDEFAULT>	<SETDEFAULT>	<SETDEFAULT>	<SETDEFAULT>
<SETTEXT>	<SETTEXT>	<SETTEXT>	<SETTEXT>
<STARTVNC>	<STARTVNC>	<STARTVNC>	<STARTVNC>
<STOP>	<STOP>	<STOP>	<STOP>
<SUPRESSMS>	<SUPRESSMS>	<SUPRESSMS>	<SUPRESSMS>
<UNBLOCKAPP>	<UNBLOCKAPP>	<UNBLOCKAPP>	<UNBLOCKAPP>
<UNINSTALL>	<UNINSTALL>	<UNINSTALL>	<UNINSTALL>
<UNMUTEDEV>	<UNMUTEDEV>	<UNMUTEDEV>	<UNMUTEDEV>
<VIBRATE>	<VIBRATE>	<VIBRATE>	<VIBRATE>
<WIPE>	<WIPE>	<WIPE>	<WIPE>