


# BlankBot - a new Android banking trojan with screen...

 [intel471.com/blog/blankbot-a-new-android-banking-trojan-with-screen-recording-keylogging-and-remote-control-capabilities](https://intel471.com/blog/blankbot-a-new-android-banking-trojan-with-screen-recording-keylogging-and-remote-control-capabilities)

```
/* loaded from: classes.dex */
public final class Workeras extends Worker {
    /* JADX WARN: 'super' call moved to the top of the method (can break code semantics) */
    public Workeras(Context context, WorkerParameters workerParameters) {
        super(context, workerParameters);
        d.f(context, "appContext");
        d.f(workerParameters, "workerParams");
    }

    @Override // androidx.work.Worker
    public final p f() {
        String str;
        Object obj = this.f5369k.b.f5361a.get("calledFrom");
        if (obj instanceof String) {
            str = (String) obj;
        } else {
            str = null;
        }
        if (str == null) {
            str = "Default Value";
        }
        Context context = this.f5368j;
        d.e(context, "applicationContext");
        i0.J(context, str);
        return new p(h.f5360c);
    }
}
```

```
/* JADX WARN: 'super' call moved to the top of the method (can break code semantics) */
public final p f() {
    super(context, workerParameters);
    d.f(context, "appContext");
    d.f(workerParameters, "workerParams");
}

/* JADX WARN: Code restructure failed: missing block: B:120:0x00e8, code
    return new z1.p(z1.h.f5360c);
*/
@Override // androidx.work.Worker
/*
    Code decompiled incorrectly, please refer to instructions dump.
*/
public final p f() {
    String str = null;
    Context context = null;
    String str2 = null;
    String str3 = null;
    String str4 = null;
    String str5 = null;
    Object obj = null;
    while (true) {
        switch (true) {
            case -1825698284:
                str = null;
                break;
            case -1751689151:
                str = null;
                break;
            case -1803827709:
                str = null;
                break;
            case -1599511640:
                str = null;
                str5 = (String) obj;
                break;
            case -1246295915:
                String str6 = null;
                while (true) {
                    switch (str6.hashCode() ^ 1572456310) {
                        case -1685905080:
                            str = null;
                            continue;
                    }
                }
            default:
                break;
        }
    }
}
```

## Key Points:

- In July 2024, Intel 471 Malware Intelligence researchers discovered the new **BlankBot** Android banking trojan.
- Based on the application names and certain strings found within the application, it is highly likely BlankBot's primary targets are Turkish users.
- BlankBot features a range of malicious capabilities, which include customer injections, keylogging, screen recording and it communicates with a control server over a WebSocket connection.
- At the time of this report, most samples remain largely undetected by the majority of antivirus software, according to VirusTotal service.
- The malware apparently still is under development, as indicated by the presence of logs and code variants.

## Overview

On July 24, 2024, Intel 471 Malware Intelligence researchers discovered malicious Android samples that impersonated utility applications which could not be attributed to any known existing malware family (see: Figure 1). We named it **BlankBot** since there was no reference via open sources at the time of this report.

The first BlankBot samples were from the end of June 2024 and almost all were undetected by most antivirus software.

Figure 1: The image depicts a screenshot of Android package kit (APK) icons BlankBot malware used, which we captured July 29, 2024.

Like many other Android banking trojans, BlankBot also abuses accessibility services to obtain complete control of infected devices. In particular, the malware is able to log everything that appears on the infected device, including short message service (SMS) text, sensitive information and a list of applications used. The malware also is able to conduct custom injections used to steal bank information, such as payment card data and a lock pattern for the device.

Communications between BlankBot and a controller start with a “GET” request where the hypertext transfer protocol (HTTP) headers include information about the infected device, such as the battery level, screen size, model, manufacturer and operating system (OS) version. The malware uses port 8080 via a WebSocket connection for subsequent controller communication.

At this early stage, application names and certain strings found within the application suggest it is likely the primary BlankBot targets are Turkish users. However, no specific financial institutions were identified as targets during our analysis, therefore, this malware could be distributed in campaigns against users in different countries.

## Technical analysis

---

The malicious app is installed, the icon is not displayed on the device launcher and the user is prompted to grant accessibility permissions accompanied by an explanation message (see: Figure 2):

*“Welcome! App needs Accessibility permission to run properly. Please give accessibility permission!”*

Figure 2: The image depicts a screenshot of the BlankBot installation process, which we captured July 29, 2024.

The malware subsequently initiates communication with the control server by sending an HTTP “GET” request to the controller and switches to the WebSocket network connection protocol.

Once the accessibility service access is granted, BlankBot displays a black screen to the user, which indicates that the app is updating. However, the malware automatically obtains all necessary permissions in the background (see: Figure 3).

Figure 3: The image depicts a screenshot of a fake update screen and permissions BlankBot obtained automatically, which we captured July 29, 2024.

If the malware is installed on a device with Android 13 or newer, BlankBot implements a session-based package installer to bypass the restricted settings feature implemented in Android 13. The bot asks the victim to allow installing applications from the third-party sources, then it retrieves the Android package kit (APK) file stored inside the application assets directory with no encryption and proceeds with the package installation process (see: Figure 4).

Figure 4: The image depicts a screenshot of the BlankBot payload installation phase via Android 13, which we captured July 29, 2024.

## Capabilities

---

### Screen recording

---

The malware can perform screen recording using Android's MediaProjection and MediaRecorder application programming interfaces (APIs). BlankBot is able to capture a video of the device screen via the MediaProjection API and the content is saved as a moving pictures experts group (MP4) file within the application's internal storage. However, this feature apparently still is under development since the implementation changes in different samples.

BlankBot uses the MediaRecorder API to capture screen images after specifying the height and width of the infected device, the image format and the maximum number of images to be acquired. The joint photographic experts group (JPEG) images captured are Base64 encoded and sent to the remote server (see: Figure 5).

Figure 5: The image depicts a screenshot of malware code used to exfiltrate captured screen images, which we captured July 30, 2024.

### Keylogging

---

Like many other Android banking trojans, the use of accessibility services plays a key role in intercepting and stealing confidential information. BlankBot also abuses accessibility to retrieve data from the infected devices, such as a list of applications used, notifications, text the user types and other sensitive information that appears on the screen or the victim copies and pastes.

However, BlankBot uses a unique custom virtual keyboard implemented via the “InputMethodService” class that Android provides, unlike most other malware. The primary purpose of this code is to intercept and send keys the user presses on the keyboard.

Figure 7: The image depicts a screenshot of a malware code snippet used to send intercepted keyboard keystrokes to the controller, which we captured July 29, 2024.

## Injections

---

Upon a specific command received from the command-and-control (C2) server, the bot is able to create a customizable overlay based on the threat actors’ needs. The overlay could be abused to ask for banking credentials, personal information, payment card data or to steal the lock pattern. The malware developers included two external, open source libraries to implement the custom injection templates, specifically:

- The CompactCreditInput library at <https://github.com/10bis/CompactCreditInput> is used to create a view to steal payment card data, automatically validate data and manage the card type logo based on the number entered.
- The Pattern Locker View library at <https://github.com/l7naive/pattern-lock> is used to create a pattern lock view.

We simulated the control server functionality to trigger and test a wide variety of BlankBot capabilities, which include overlays. We issued a command to create three different views customized with the ING bank logo and user interface (UI) text elements displayed (see: Figure 8). Any user input was logged and promptly exfiltrated to the control server.

Figure 8: The image depicts a screenshot of customized overlays that our Malware Intelligence Team generated, which we captured July 30, 2024.

## Commands

---

BlankBot communicates with the C2 server over a WebSocket connection to exfiltrate data and receive a wide range of bot commands. Specifically, the botmaster is able to start and stop screen recording to receive a live view of the infected device display. For applications that implement a “FLAG\_SECURE” security measure to prevent sensitive data leak, the botmaster can leverage a hidden virtual network (HVNC) module to exfiltrate the layout of UI elements by abusing the accessibility services. Threat actors are able to perform on-device fraud (ODF) by waking up and controlling the device remotely with different types of supported gestures, such as clicks or swipes. Additionally, BlankBot is capable of creating overlays, as described in the previous section, as well as collecting contacts, SMS text and a list of installed applications. The commands supported by the bot were described in the table below.

<b>Command ID</b>	<b>Description</b>
-3	Stop HVNC module
-1	Stop screen recording
1	Start screen recording
2	Perform a gesture
3	Start HVNC module
11	Create an overlay with message
12	Request a permission or change device settings
13	Collect SMS text
15	Collect contacts
16	Send text message
18	Wake up device
20	Collect installed applications
22	Create an overlay with edit text or pattern
24	Delete an SMS text
25	Uninstall an application
26	Launch an application
27	Create an overlay for payment card data

## Defense evasion

---

The malware is installed on a device and checks to determine whether it is an emulator. If the infected device is considered legitimate, it attempts to maintain persistence by preventing the user from performing a variety of actions, such as accessing the settings or antivirus applications. This is achieved using the accessibility services which monitor all events on the infected device and certain words that appear on the device screen (see: Figure 9).

Figure 9: The image depicts malware code BlankBot uses to maintain persistence on an infected device, which we captured July 29, 2024.

Recent BlankBot samples were partially obfuscated and junk code was added to slow down the reverse-code engineering process, which makes it significantly more challenging for security researchers to analyze the code and understand the malware's behavior.

Figure 10: The image depicts a screenshot of the same malware code BlankBot implemented in two different variants, which we captured July 29, 2024.

## Conclusions

---

BlankBot is a new Android banking trojan still under development, as evidenced by the multiple code variants observed in different applications. Regardless, the malware can perform malicious actions once it infects an Android device, which include conducting custom injection attacks, ODF or stealing sensitive data such as credentials, contacts, notifications and SMS messages.

This research aims to demonstrate how the mobile threat landscape continually evolves and how cybercriminals continue to create new types of malware to stay under the radar until receiving media attention or interest from most antivirus companies.

## Indicators of compromise (IoCs)

---

### BlankBot APK SHA-256

- 7d5b6bcc9b93aedc540e76059ee27841a96acb9ea74a51545dfef18b0fcf5b57
- 6fc672288e68146930b86c7a3d490f551c8d7a7e8ba3229d64a6280118095bea
- ad9044d9762453e2813be8ab96b9011efb2f42ab72a0cb26d7f98b9bd1d65965
- b4b4b195e14e9fda5a6d890ddb57f93ef81d6d9a976078354450ee45d18c89e3
- 8d6ca64e4c3c19587405e19d53d0e2f4d52b77f927621d4178a3f7c2bf50c2ea
- d163cc15a39fb36391bd67f6eaada6691f0c7bc75fc80282a4a258244163e12a

- 6681b0613fc6d5a3e1132f7499380eb9db52b03ab429f0c2109a641c9a2ea4d3
- 11751c6aa3e5c44c92765876bc9cd46da90f466b9924b9b1993fa1c91157681d
- fc5099e5be818f8268327aaf190cd07b4b4ebb04e9d63eefa5a04ea504f93d62

### BlankBot control servers

- 79.133.41.52
- 185.255.92.185

## MITRE ATT&CK

---

### MITRE ATT&CK techniques

---

TECHNIQUE TITLE	ID	USE
<b>Collection [TA0035]</b>		
Clipboard Data	T1414	Writes data in the user's clipboard when a specific command is received from the control server
Keylogging	T1417.001	Logs keystrokes by abusing the accessibility API
Screen Capture	T1513	Records screen content using the MediaProjection and MediaRecorder APIs
Contact List	T1636.003	Collects and exfiltrates the device's contact list
SMS Messages	T1636.004	Collects and exfiltrates SMS text from the device
<b>Credential Access [TA0031]</b>		
GUI Input Capture	T1417.002	Creates overlays to steal payment card data and pattern lock
<b>Discovery [TA0032]</b>		

---

Software Discovery	T1418	Retrieves a list of installed applications and exfiltrates it to the controller
System Information Discovery	T1426	Collects device information such as manufacturer, model, operating system version, battery level and screen size
<b>Command and Control [TA0037]</b>		
Non-Standard Port	T1509	Communicates with the control server over a WebSocket connection on port number 8080
<b>Impact [TA0034]</b>		
Input Injection	T1516	Abuses accessibility services to perform arbitrary actions on behalf of the user
SMS Control	T1582	Sends an SMS text to a specified phone number
<b>Persistence [TA0028]</b>		
Broadcast Receivers	T1624.001	Registers the "BOOT_COMPLETED" broadcast intent to run when the device boots
<b>Privilege Escalation [TA0029]</b>		
Device Administrator Permissions	T1626.001	Asks for device administrator privileges following a specific command
<b>Defense Evasion [TA0030]</b>		
Suppress Application Icon	T1628.001	Hides the icon from the application launcher



---

Prevent Application Removal	T1629.001	Abuses accessibility services to prevent the user from uninstalling the malware application
Uninstall Malicious Application	T1630.001	Uninstalls the malware application from the infected device after a specific command
Virtualization/Sandbox Evasion	T1633	Performs anti-emulation checks to avoid running in a sandbox environment

---

### **Exfiltration [TA0036]**

---

Exfiltration Over Alternative Protocol	T1639	Exfiltrates collected data via a WebSocket network communication protocol
--	-------	---