1 Attack Classification

（1）**Phishing Hijacking of the Stack:** an Activity of a third-party app is opened above the top of the stack, mimicking a sensitive page to trick users into entering private information.

（2）**Hardware Resource Hijacking:** hijacking the data provided by hardware resources or providing false data to deceive users when normal applications invoke hardware resources of mobile devices.

（3）**Obfuscation Attack:** useful information is maliciously obscured, and must be completely obscured to affect user judgment, causing user confusion leading to an attack.

2 Monitor

Monitor means to get the name of currently running application through background Service or foreground Activity. Since applications run between different sandboxes in Android OS, to get the running foreground application requires a method via **side channel**.

According to our experiments, four methods are designed: reading the system log from the I (Info) class information through adb logcat command to read the running package name, getting the process information through adb top command to read the package name of the foremost process, getting the process information through the pid file under /proc folder, and getting the application running data to determine the latest used application.

Among them, method one and method two are restricted to read inside the sandbox, that is, the malicious application can only read its own log and process information, and cannot realize listening. Method three is designed based on the underlying Linux system of Android, but currently reading the /proc folder has been banned, and can only be accessed with root privileges, which is too high a privilege level, so this method is also abandoned.

We found that in method 4, we only need to apply for android.permission.PACKAGE_USAGE_STATS permission to get the runtime data of all applications. Then, we can read the current system time, determine the application with the latest runtime, and listen to the application that is running in the foreground.

3 Phishing Hijacking of the Stack

3.1 Classification

(1) Hijacking of PIP Stack

When listening to the target App is running in PIP mode, the malware can enter PIP mode and overwrite/base the target App on the PIP stack with malicious Activity to achieve the purpose of hijacking.

(2) Hijacking of FF Stack

The malware runs in the main interface and the target software runs in the FF window. When the target Activity is detected to be open, the malware directly disguises the target software in the main interface and prompts the user with Toast, "For security reasons, the application has been switched to run in full screen" to achieve the purpose of fraud hijacking. (It would be better if the FF window application can be turned off: perhaps find if there is a pause multi-window API)

(3) Hijacking of the Main Stack

When listening to the target App running, it can overwrite the malicious Activity to the upper layer of the full-screen stack to seize the user's information.

(4) Hijacking of the Exit FF Stack

1. Direct hijacking: The malware runs in FF stack, when listening to the sensitive Activity, it will call out a browser page through the event of "download", which will run in the main stack, change the browser link and transfer it to the phishing page, in order to cover the sensitive Activity.

2. Conversion hijacking: The malware runs in FF stack, when listening to the sensitive Activity open, it will call out a browser page through the event "download", and at the same time the malware disguises itself as the sensitive Activity in the original main stack, and pops up the Toast message "For account For account security reasons, multi-window mode has been temporarily closed for you", giving the user the illusion that the application is switching between two windows, in order to achieve the purpose of fraud.

The current difficulty: need to determine how the "download" event opens the browser page?

3.2 Technical Path of Hijacking Implementation

(1) Hijacking of PIP Stack

Use Activity's startActivity or Service to open a HijackingActivity using the Intent after adding NEW_TASK, call enterPictureInPictureMode in the Activity to enter PIP mode, and squeeze the original PIP mode Activity in the original PIP mode will be squeezed out of the PIP stack.

(2) Hijacking of FF Stack

The malicious application calls startActivity to jump to the disguised interface, and sends a prompt message to instruct the user via Toast.makeText.

(3) Hijacking of the Main Stack

The malicious application creates a Service when running in full screen and keeps its TaskAffinity in FS mode; then the Service calls out HijackingActivity into PIP mode when it listens to the target application; then it calls out another VideoActivity into PIP mode to make HijackingActivity is squeezed into the FS stack.

(4) Exit the FF Stack Hijacking

Not yet, we need to find the source of the event that can enter the main stack from FF stack by "downloading".

4 Hardware Resource Hijacking

Information Leakage Attack

Description: There may be full-screen stack calls to system tools (e.g. camera, GPS) when the FF stack of applications read or block the information provided by the system tools through intent hijacking, which can cause information leakage or inconvenience to users. The information leakage attack will exploit multiple recovery mechanisms of multi-view, which can keep the malicious application's Activity running, thus bypassing a large number of protection states against background Services. The specific schematic is shown in Figure 1.

5 Obfuscation Attack

(1) Privilege Extraction Attack

Description: When the malicious app can listen to the system application open, it can open a false/deceptive description of the permission request panel/instructions ("Camera" detects that the application "ExtractPrivilege" is running Please open the camera permission for normal use), which can use the trustworthiness of the system application to deceive users into giving malware permission.

(2) Advertising Obfuscation Attack

Description: Some applications will open the adView during the opening/use process, and by clicking the adView, we can jump to the hyperlink set in the program. If we can open the adView through the FF mode app to cover the adView of the full-screen stack app, we can achieve the effect of ad obfuscation attack and present the content we need to the user.

(3) Notification Obfuscation Attack

Description: When the malware and the target software are running at the same time, the system notification popped up in the interface will be on top of the main screen, and the user cannot distinguish the source of the notification, which can achieve the purpose of obfuscation and fraud.