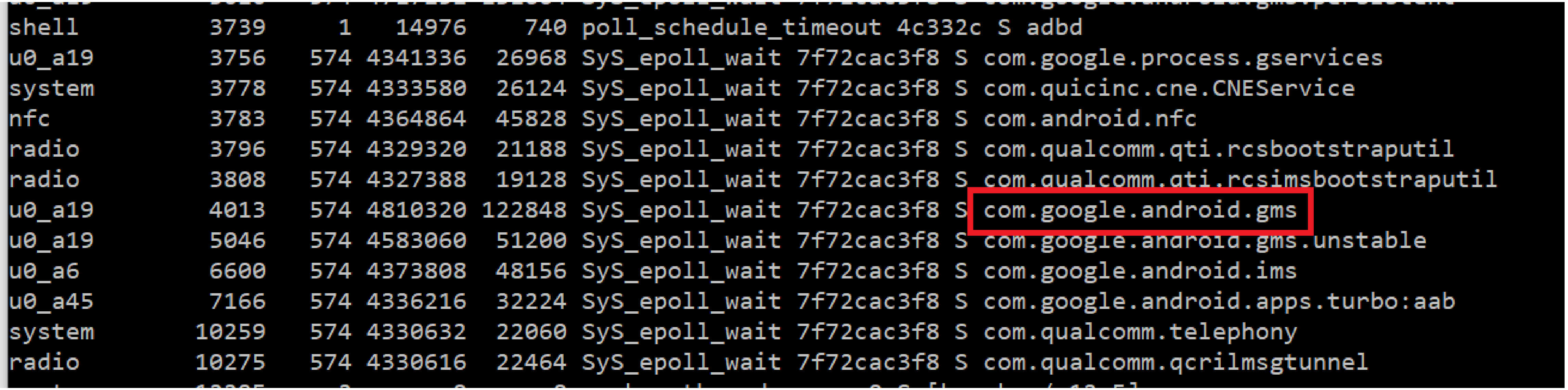
ANDROID SECURITY MODEL

Bitdefender

Sandboxing

- applications run in their own sandbox, isolated from each other. Each application:
  - has a unique user ID (UID, also referred to as AID) and unique string identifier ***(package name)***
  - runs it in its own process (process name is package name)
  - Containment/sandboxing is both at *__process level and file system access level__*

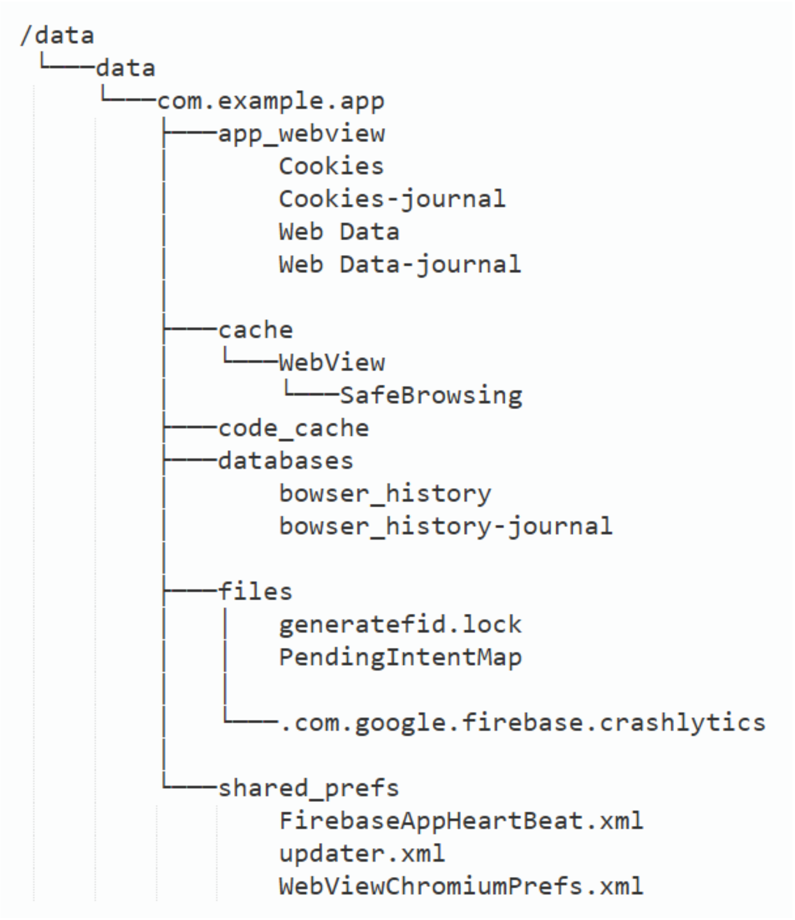
Example: Google Play services (com.google.android.gms) is currently running



```
shell           3739     1   14976    740 poll_schedule_timeout 4c332c S adbd
u0_a19          3756   574 4341336  26968 SyS_epoll_wait 7f72cac3f8 S com.google.process.gservices
system          3778   574 4333580  26124 SyS_epoll_wait 7f72cac3f8 S com.quicinc.cne.CNEService
nfc             3783   574 4364864  45828 SyS_epoll_wait 7f72cac3f8 S com.android.nfc
radio           3796   574 4329320  21188 SyS_epoll_wait 7f72cac3f8 S com.qualcomm.qti.rcsbootstraputil
radio           3808   574 4327388  19128 SyS_epoll_wait 7f72cac3f8 S com.qualcomm.qti.rcsimsbootstraputil
u0_a19          4013   574 4810320 122848 SyS_epoll_wait 7f72cac3f8 S com.google.android.gms
u0_a19          5046   574 4583060  51200 SyS_epoll_wait 7f72cac3f8 S com.google.android.gms.unstable
u0_a6           6600   574 4373808  48156 SyS_epoll_wait 7f72cac3f8 S com.google.android.ims
u0_a45          7166   574 4336216  32224 SyS_epoll_wait 7f72cac3f8 S com.google.android.apps.turbo:aab
system         10259   574 4330632  22060 SyS_epoll_wait 7f72cac3f8 S com.qualcomm.telephony
radio          10275   574 4330616  22464 SyS_epoll_wait 7f72cac3f8 S com.qualcomm.qcrilmsgtunnel
```

Application file system access

- Application specific directories can be both on external storages and internal, private storages.
- **External storage directories**
  - Can be accessed by other applications with the permission to read the storage
- **Private internal storage directories**
  - There are situated in: /data/data/<package_name>/



```
/data
 └───data
     └───com.example.app
         ├───app_webview
         │       Cookies
         │       Cookies-journal
         │       Web Data
         │       Web Data-journal
         │
         ├───cache
         │   └───WebView
         │       └───SafeBrowsing
         ├───code_cache
         ├───databases
         │       bowser_history
         │       bowser_history-journal
         │
         ├───files
         │   │   generatefid.lock
         │   │   PendingIntentMap
         │   │
         │   └───.com.google.firebase.crashlytics
         │
         └───shared_prefs
                 FirebaseAppHeartBeat.xml
                 updater.xml
                 WebViewChromiumPrefs.xml
```

Android Permission model

- model restricts for an application access to:
  - **data** such as a user's contact information or call log
  - **actions** such as connecting to a paired device, recording audio, or sending a SMS

- Permissions can be categorized into several different types based on:
  - the scope of restricted data that your app can access or actions that your app can perform (how risky you application is)
  - when the system grants your app that permission

Permission protection levels

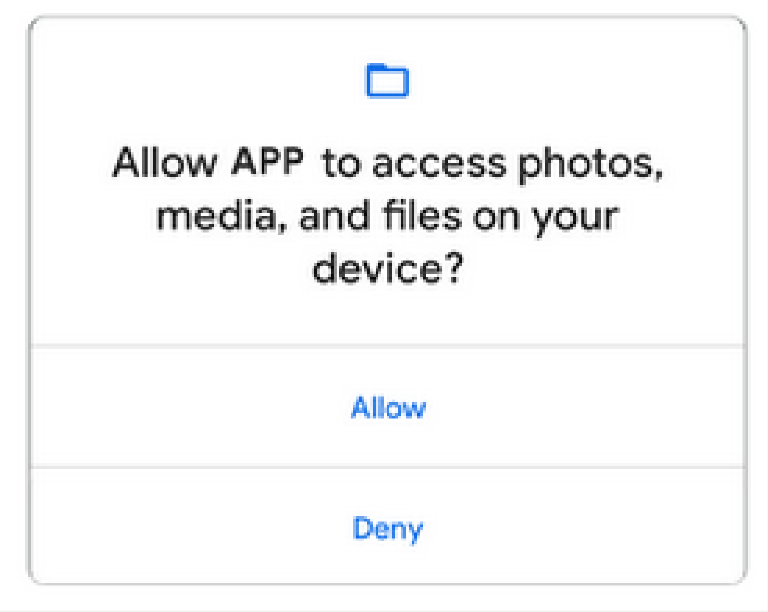
What risks giving a permission brings is divided in 4 protection levels and an extension (**appop**). The following are more relevant in malware analysis

- **Normal**
  - low-risk permission; system automatically grants access at installation (**Install-time permissions)**
- **Dangerous**
  - higher-risk permission – gives access to private user data or control over the device
  - User confirmation is required before granting (**Runtime permissions)**
- **Special permissions (appop)**
  - Extension, permissions with this marker are considered special.
  - Provide access to powerful actions: *e.g. drawing over other apps*
  - explicitly granted by user, usually through a specific permission management screen
- Permissions description can be found here

Permission granting

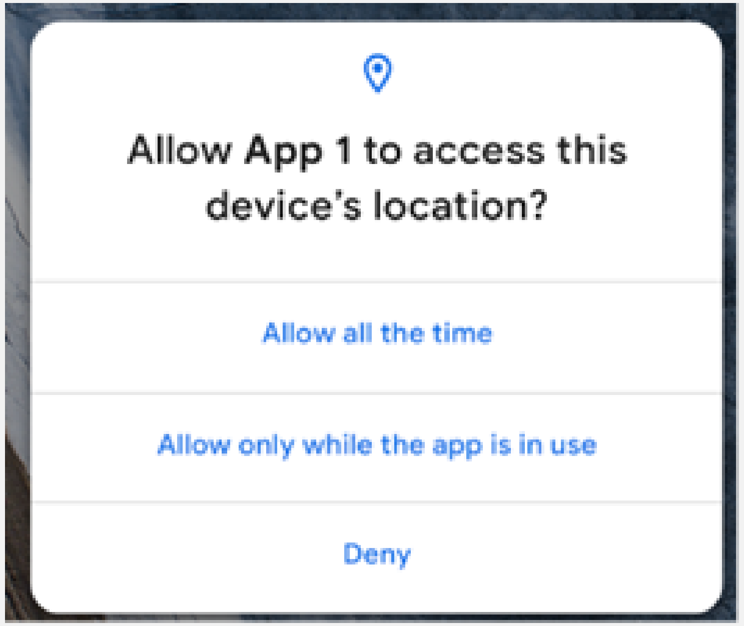
- **Install-time permissions**
  - Granted at application install time
- **Runtime permissions**
  - requested by app after install
  - Must be accepted by user.
  - When requested, system presents a standard runtime permission prompt

From Android 6.0 to Android 9 you had only the option to allow or deny

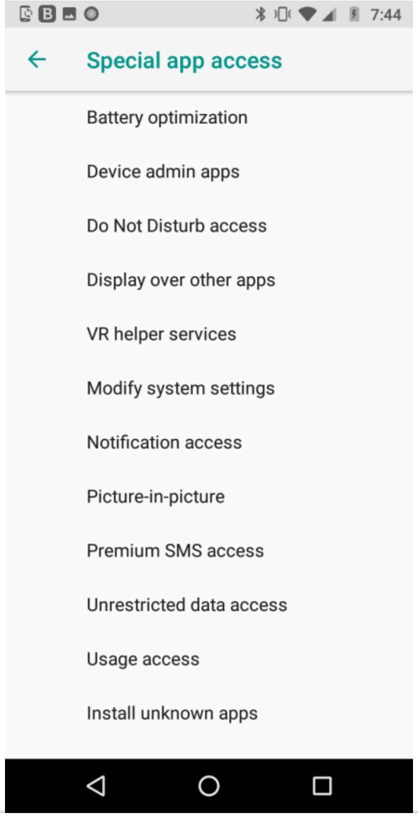


Allow APP to access photos, media, and files on your device?

Starting with Android 10, for some permissions, a third option was added to allow only while the app is in use

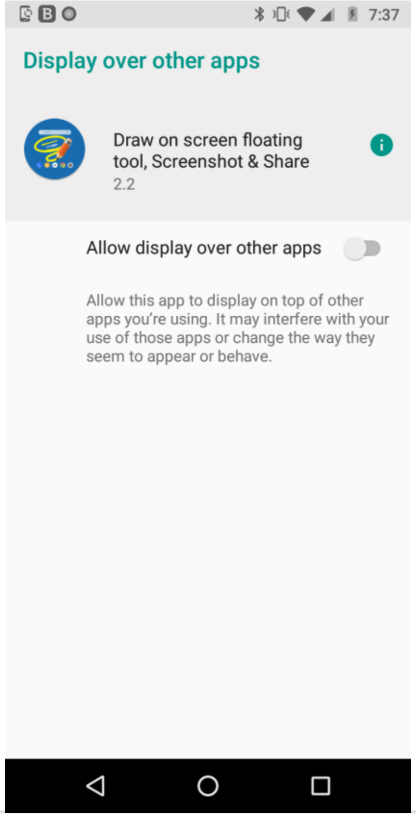


Allow App 1 to access this device's location?



Special app access

- Battery optimization
- Device admin apps
- Do Not Disturb access
- Display over other apps
- VR helper services
- Modify system settings
- Notification access
- Picture-in-picture
- Premium SMS access
- Unrestricted data access
- Usage access
- Install unknown apps

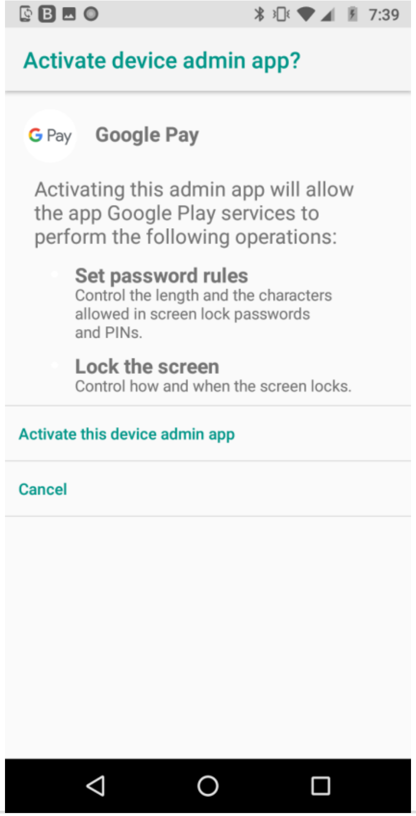


Display over other apps

Draw on screen floating tool, Screenshot & Share
2.2

Allow display over other apps

Allow this app to display on top of other apps you're using. It may interfere with your use of those apps or change the way they seem to appear or behave.



Activate device admin app?

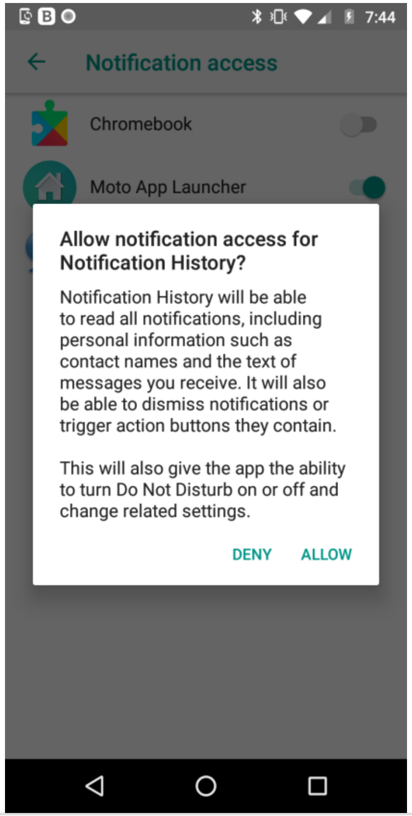
Google Pay

Activating this admin app will allow the app Google Play services to perform the following operations:

- **Set password rules**
  Control the length and the characters allowed in screen lock passwords and PINs.
- **Lock the screen**
  Control how and when the screen locks.

Activate this device admin app

Cancel



Notification access

Chromebook

Moto App Launcher

Allow notification access for Notification History?

Notification History will be able to read all notifications, including personal information such as contact names and the text of messages you receive. It will also be able to dismiss notifications or trigger action buttons they contain.

This will also give the app the ability to turn Do Not Disturb on or off and change related settings.

DENY ALLOW

Permission granting (continued)

- **Special permissions**
  - give access to potentially dangerous actions
  - Requested at runtime
  - user must explicitly grant them
  - Custom permission management prompt for each

Certificate validations

- applications running on Android OS must be digitally signed
- on installation, the Package Manager verifies the APK
  - if not signed – rejected: **INSTALL_PARSE_FAILED_NO_CERTIFICATES**
  - if integrity check of signing algorithm fails – rejected: **INSTALL_PARSE_FAILED_UNEXPECTED_EXCEPTION**

- Apps can be signed using self-signed certificates generated by anyone
  - APK can be signed using **apksigner**
  - *Attribution to one entity can be done by correlating samples signed with the same certificate*

Certificate validations (continued)

Several APK Signature and validation Schemes are supported:

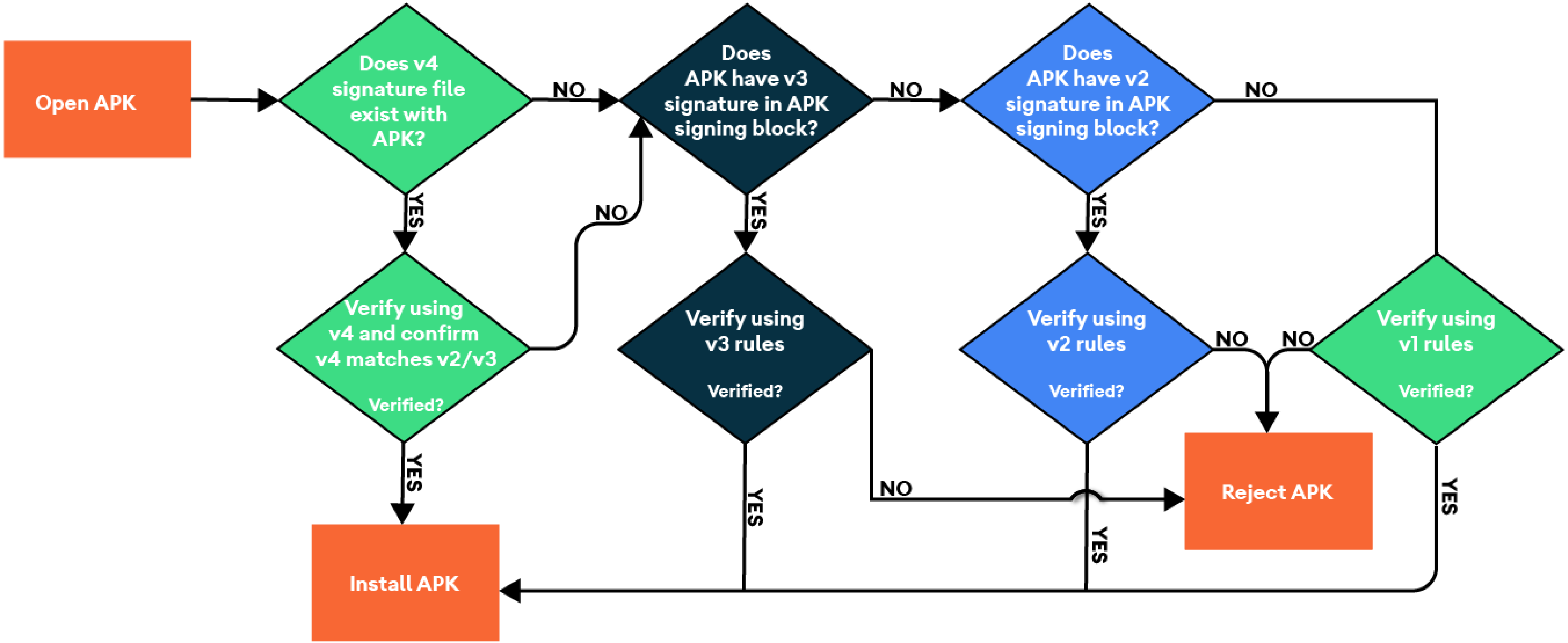
- v1 scheme: based on JAR signing
- v2 scheme introduced in Android 7.0
- v3 in Android 9
- v4 in Android 11.

V1 schema validates the integrity of each file in the APK, treating them as zip entries

- This leaves an attack surface where threat actors ca modify the APK file without modifying the zip entries themselves (e.g. Janus CVE-2017-13156)

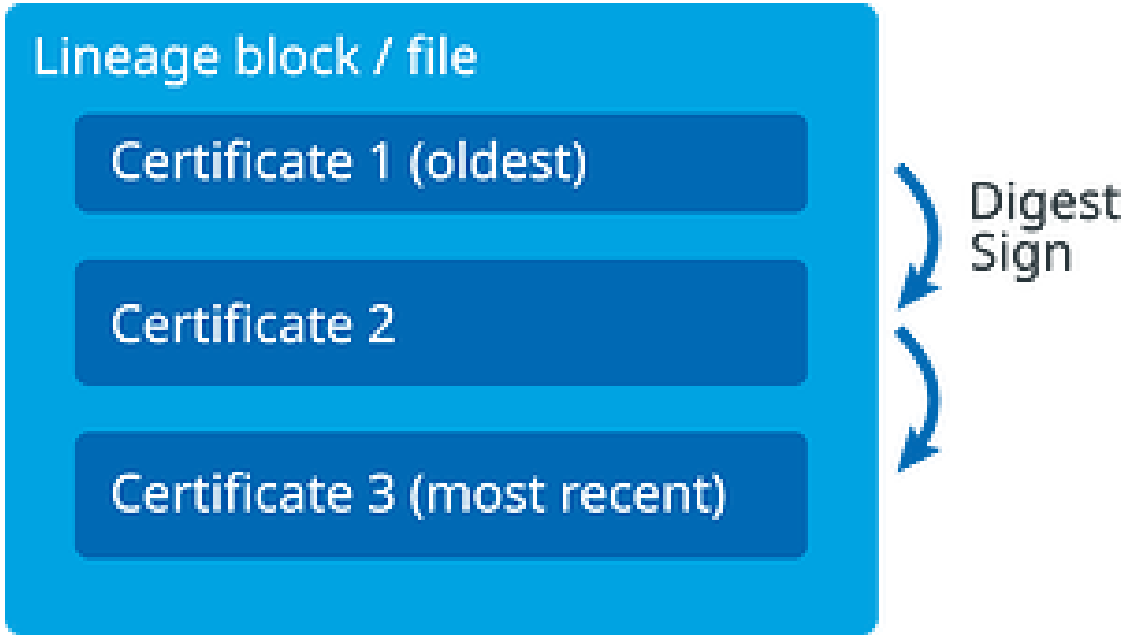
V2 schema and newer also validate the integrity of the APK file as a whole.

Certificate validations (continued)


Open APK
Does v4 signature file exist with APK?
NO
YES
Does APK have v3 signature in APK signing block?
NO
YES
Does APK have v2 signature in APK signing block?
NO
YES
Verify using v4 and confirm v4 matches v2/v3
Verified?
NO
YES
Verify using v3 rules
Verified?
YES
NO
Verify using v2 rules
Verified?
NO
YES
Verify using v1 rules
Verified?
NO
YES
Reject APK
Install APK

Certificate Proof of Rotation

- Introduced with v3, allows apps to change their certificate on updates.
- Linked list with the oldest certificate as the root, each certificate should sign the next.
- APK Signature scheme v3.1 - added target api metadata for a certificate section


Lineage block / file
Certificate 1 (oldest)
Certificate 2
Certificate 3 (most recent)
Digest
Sign