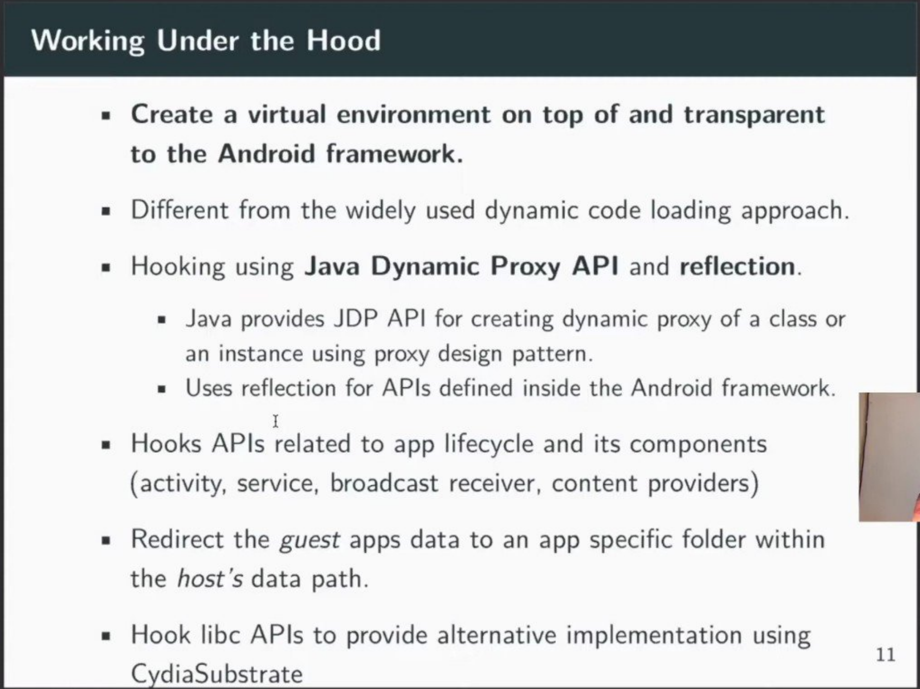
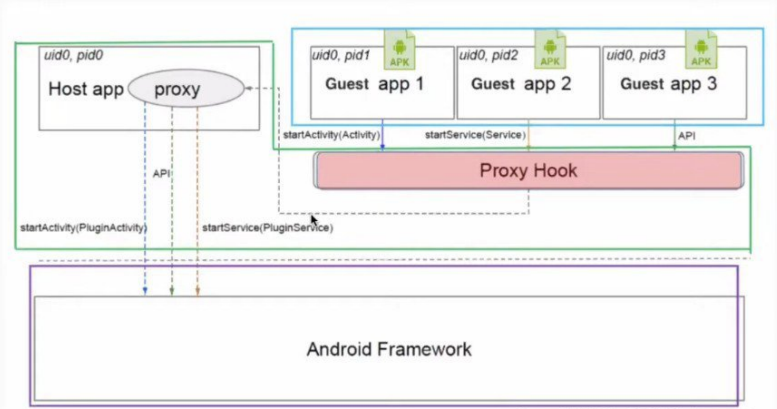


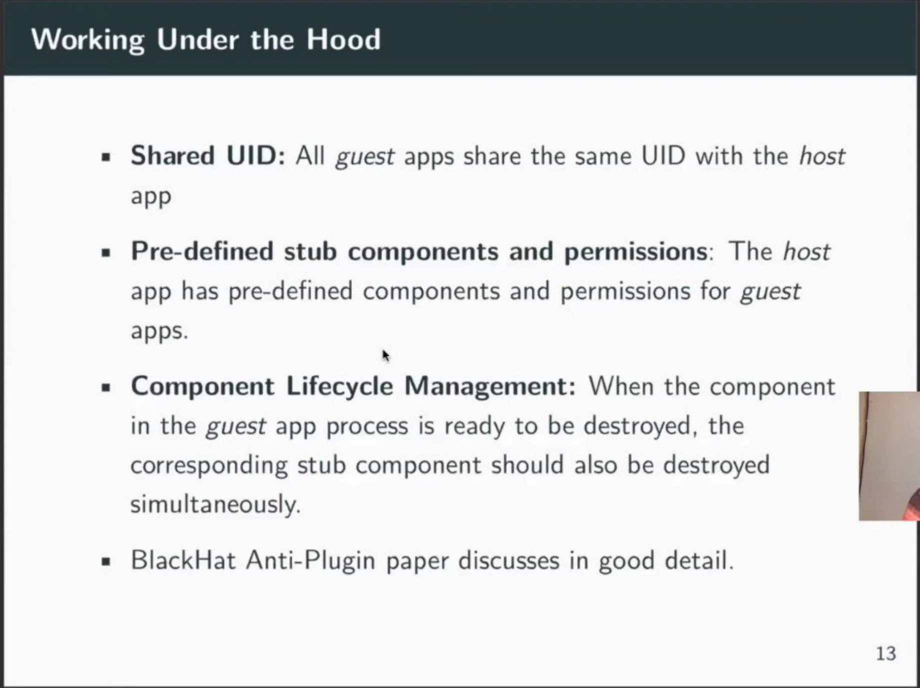
## Working Under the Hood

- **Create a virtual environment on top of and transparent to the Android framework.**
- Different from the widely used dynamic code loading approach.
- Hooking using **Java Dynamic Proxy API** and **reflection**.
  - Java provides JDP API for creating dynamic proxy of a class or an instance using proxy design pattern.
  - Uses reflection for APIs defined inside the Android framework.
- Hooks APIs related to app lifecycle and its components (activity, service, broadcast receiver, content providers)
- Redirect the *guest* apps data to an app specific folder within the *host's* data path.
- Hook libc APIs to provide alternative implementation using CydiaSubstrate

## Working Under the Hood

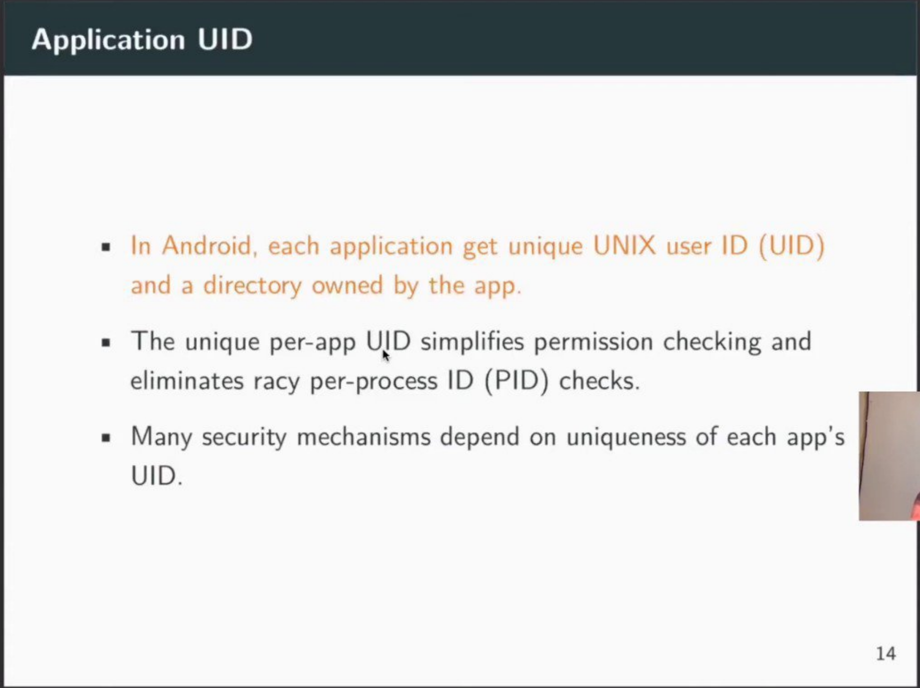


*Source: BlackHat Asia 2017 Anti-Plugin presentation



## Working Under the Hood

- **Shared UID:** All *guest* apps share the same UID with the *host* app
- **Pre-defined stub components and permissions**: The *host* app has pre-defined components and permissions for *guest* apps.
- **Component Lifecycle Management:** When the component in the *guest* app process is ready to be destroyed, the corresponding stub component should also be destroyed simultaneously.
- BlackHat Anti-Plugin paper discusses in good detail.

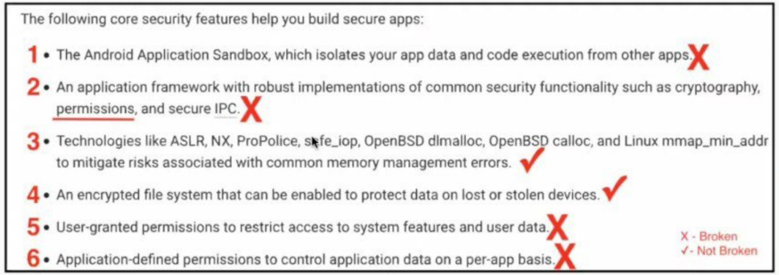


## Application UID

- In Android, each application get unique UNIX user ID (UID) and a directory owned by the app.
- The unique per-app UID simplifies permission checking and eliminates racy per-process ID (PID) checks.
- Many security mechanisms depend on uniqueness of each app's UID.

## Android Built-In Security

- From Android documentation, most of the Android core security features are broken for a *guest* app.



The following core security features help you build secure apps:

1 • The Android Application Sandbox, which isolates your app data and code execution from other apps. X

2 • An application framework with robust implementations of common security functionality such as cryptography, permissions, and secure IPC. X

3 • Technologies like ASLR, NX, ProPolice, safe_iop, OpenBSD dlmalloc, OpenBSD calloc, and Linux mmap_min_addr to mitigate risks associated with common memory management errors. ✓

4 • An encrypted file system that can be enabled to protect data on lost or stolen devices. ✓

5 • User-granted permissions to restrict access to system features and user data. X

6 • Application-defined permissions to control application data on a per-app basis. X

X - Broken
✓ - Not Broken

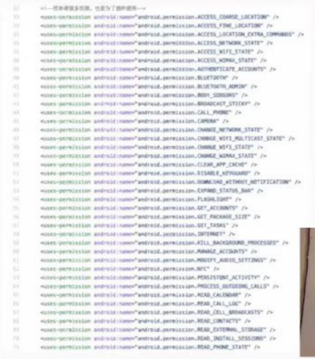
- 3 & 4 are lower in abstraction layer than at which virtual container operates, and thus not affected.

## Breaking Android Security Model

- Many Android security and privacy features depend upon UID assigned to an app:
  - **Application Permissions**
  - **Android Keystore**
  - **Android ID**
- Unauthorized access to other *guest* app's **sandbox data**.
- A *guest* app can get list of other running *guest* apps.

## Android Manifest - Permissions

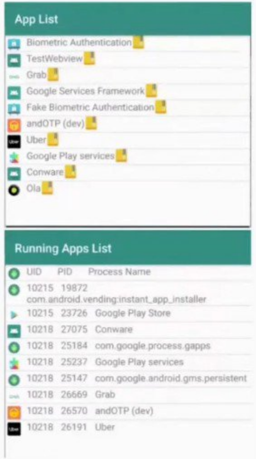
- **All or none**: For one granted permission, all *guest* apps get access for that permission.
- **DroidPlugin declares 141 permissions** in manifest file.
- In a virtual container, an app is never installed, thus manifest data is not really processed.
- Granted permissions persist even if the *guest* app that requested is uninstalled.
- A major **privacy concern**.
- On manually disabling a permission may break some other *guest* app.



```
<!--[illegible]-->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIMAX_STATE" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BODY_SENSORS" />
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIMAX_STATE" />
<uses-permission android:name="android.permission.CLEAR_APP_CACHE" />
<uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
<uses-permission android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION" />
<uses-permission android:name="android.permission.EXPAND_STATUS_BAR" />
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.GET_PACKAGE_SIZE" />
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES" />
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.NFC" />
<uses-permission android:name="android.permission.PERSISTENT_ACTIVITY" />
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
<uses-permission android:name="android.permission.READ_CALENDAR" />
<uses-permission android:name="android.permission.READ_CALL_LOG" />
<uses-permission android:name="android.permission.READ_CELL_BROADCASTS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_INSTALL_SESSIONS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

## List Other Guest Apps

- Can get list of other **installed** and **running** *guest* apps in the virtual container.
- List of installed apps by iterating the storage directory.
- From API 22 (Android 5.1.1) it is deprecated to list running apps.
- A **privacy** concern.



App List

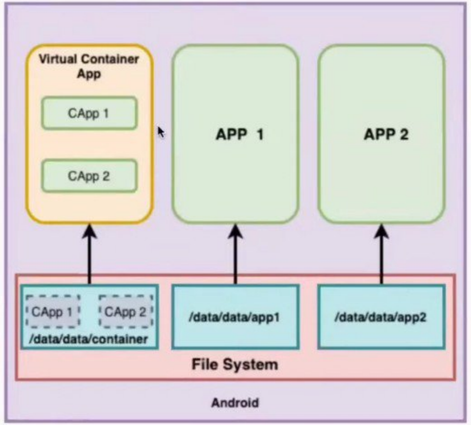
Biometric Authentication
TestWebview
Grab
Google Services Framework
Fake Biometric Authentication
andOTP (dev)
Uber
Google Play services
Conware
Ola

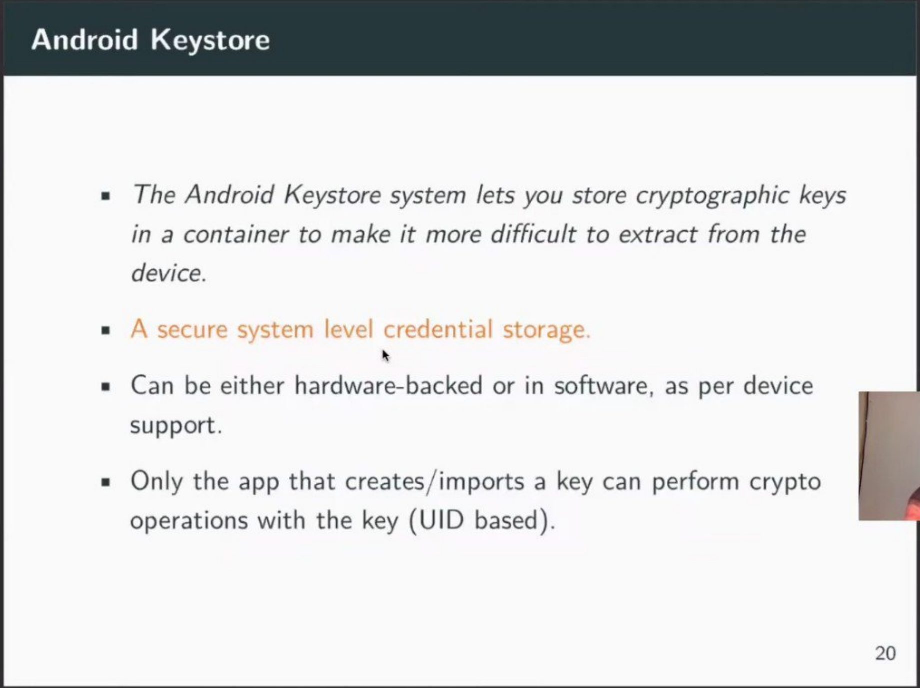
Running Apps List

| UID | PID | Process Name |
|---|---|---|
| 10215 | 19872 | com.android.vending:instant_app_installer |
| 10215 | 23726 | Google Play Store |
| 10218 | 27075 | Conware |
| 10218 | 25184 | com.google.process.gapps |
| 10218 | 25237 | Google Play services |
| 10218 | 25147 | com.google.android.gms.persistent |
| 10218 | 26669 | Grab |
| 10218 | 26570 | andOTP (dev) |
| 10218 | 26191 | Uber |

## Android Filesystem Sandbox

- Data sandboxing: One app cannot access data from another app.
    - Implemented and enforced at the kernel level.
- No data sandboxing between *guest* apps in a virtual container.



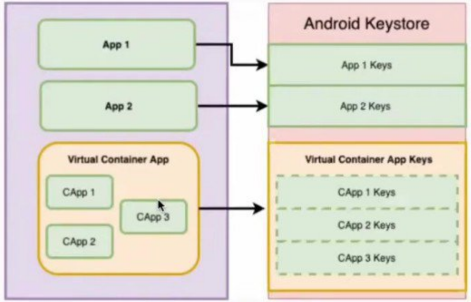


## Android Keystore

- *The Android Keystore system lets you store cryptographic keys in a container to make it more difficult to extract from the device.*
- A secure system level credential storage.
- Can be either hardware-backed or in software, as per device support.
- Only the app that creates/imports a key can perform crypto operations with the key (UID based).

## Android Keystore

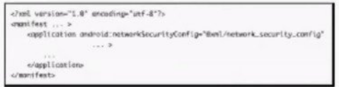
- In a virtual container, all apps have same UID!!



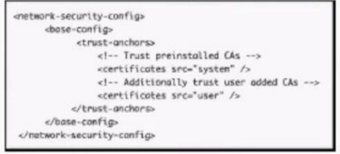
- *Host* and *Guest* apps have access to all the keys generated by other *guest* apps.
- Keys remain in keystore even if the *guest* app that generated it is removed from the container.
- Brings back an old bug - key leakage between security domains

## Android Manifest - Network Security Config

- Since Android Nougat (7.0), apps can customize their network security settings.
- User installed TLS CA certs are not trusted by default, requires explicit declaration in manifest file.
- By setting attribute *android:networkSecurityConfig*.
- Network Security Configuration of *host* will be inherited by all the *guest* apps.
- If *host* trusts self-signed certs, a *guest* app will trust too.



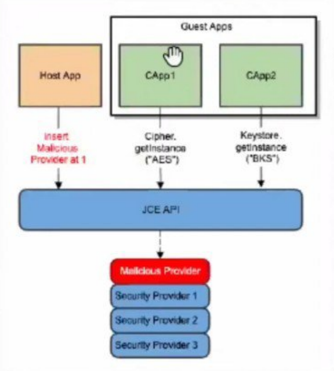
```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:networkSecurityConfig="@xml/network_security_config"
                    ... >
        ...
    </application>
</manifest>
```



```
<network-security-config>
    <base-config>
        <trust-anchors>
            <!-- Trust preinstalled CAs -->
            <certificates src="system" />
            <!-- Additionally trust user added CAs -->
            <certificates src="user" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

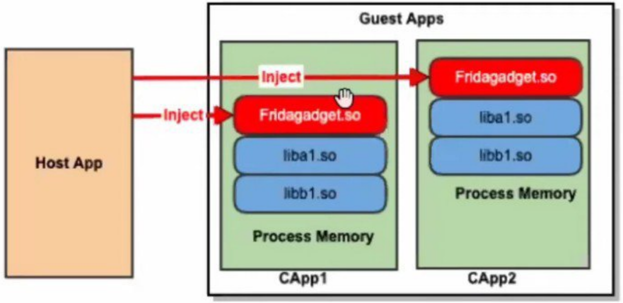
## Java Security Provider

- A provider for Java Security API
  - Cryptographic engines
  - Keystore
- *Host* app can override the security provider used in the *guest* apps, if not explicitly specified in *guest* apps.
- *Guest* apps relying on system default security provider are at risk.



## Dynamic Instrumentation

- Pre-load native libraries
  - *Fridagadget* can be pre-loaded when a *guest* app is invoked.
  - Easy to perform dynamic instrumentation on *guest* apps


Guest Apps
Host App
Inject
Inject
Fridagadget.so
liba1.so
libb1.so
Process Memory
Fridagadget.so
liba1.so
libb1.so
Process Memory
CApp1
CApp2