

Application Insecurity in Android Virtual Containers

Gautam Arvind Pandian Vikas Gupta

January 2, 2021

Thales DIS

SINCON2020, Singapore Virtual

Intro

Overview

- **Introduction** to Android virtual containers
- **Use case** and current market status of virtual container apps
- **Attacking** applications running inside virtual containers
- How to **detect** virtual containers
- **Recommendation** for developers
- **Discussion**
- **Conclusion**

About Us

- **Gautam Arvind Pandian**

- Security Researcher and Architect at *Thales DIS, Singapore*
- CTF (R2Pay) contributor in R2CON 2020 conference
- Speaker at Android Security Symposium 2020
- Interests: Crypto, Hardening Mobile Apps

- **Vikas Gupta**

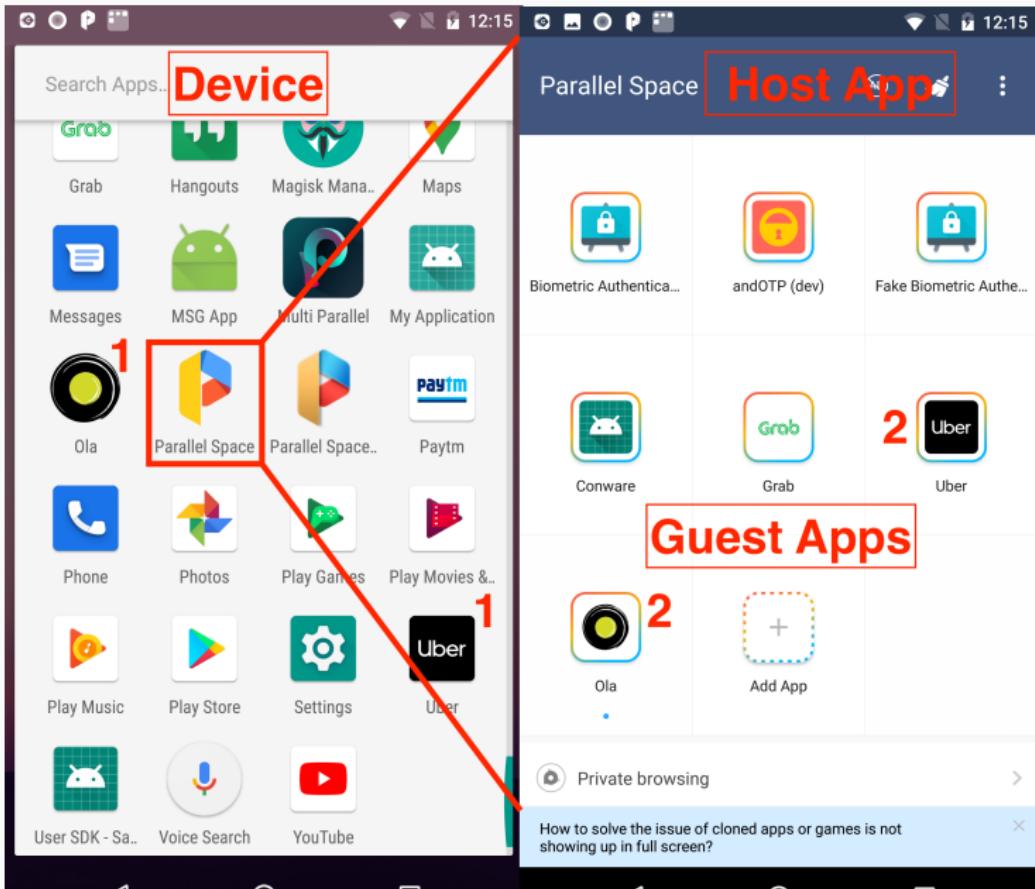
- Security Researcher and Pentester at *Thales DIS, India*
- Reverse Engineering, (De)Obfuscation, Crypto
- Co-Author contributor for *OWASP MSTG* (Mobile Security Testing Guide)
- M.Sc in Computer Security, OSCP certified

Android Virtual Containers

What are Virtual Containers?

- Innovative application-level virtualisation framework.
- Consists of an **Android application** (*host*) running multiple *guest* Android apps.
- *Guest* app does not require any installation.
- *Guest* app can be a second instance of an existing application on the actual device.
- **Virtual Containers reduces the security of *guest* apps.**

Android Virtual Containers



Android Virtual Containers



- *Android-Plugin: Don't Let Your App Play as an Android Plugin* in BlackHat Asia 2017.
- Detailed description on the working.
- Discussed from a malware-centric point of view.
- Proposed ideas for detecting virtual containers.
- **Note:** We find the term *Android Plugin* confusing and not descriptive enough, and instead we use **Virtual Container**.

Where Can I Get One?



VirtualApp



- **Virtual App - GitHub**
 - Many virtual container apps derived from this project
- **Droid Plugin - GitHub**
- **Parallel Space** - 100M+ downloads
- **Dual Space** - 100M+ downloads
- **2 Accounts, Multi Parallel, Dual App** - 5M+ downloads
- **Dr.Clone, Clone App, Super Clone** - 1M+ downloads
- Download numbers suggest used by **millions of users**.

Parallel Apps on OnePlus Devices

- A hardcoded list of apps allowed to have multiple instances - around 56 apps.
- **Does not use virtual container approach.**
- **Creates a new Android user** internally for running second instance of an app.
- Messaging apps: Whatsapp, WeChat
- Social Media: FB, Instagram, Twitter, Pinterest
- Ride-Hailing: Uber, Ola
- Digital Payment: Paytm, AliPay
- And many other Chinese apps.
- **Xiaomi** supports similar functionality with name *Dual Apps*.



Is it Really Needed?

- With many **dual sim devices**, users need two accounts for each number.
- Run multiple instances of chat messengers, ride-hailing apps, social media accounts, e-commerce apps etc.
- Segregate personal and business apps on one device.
- Run games - enables to tinker with games without being root.

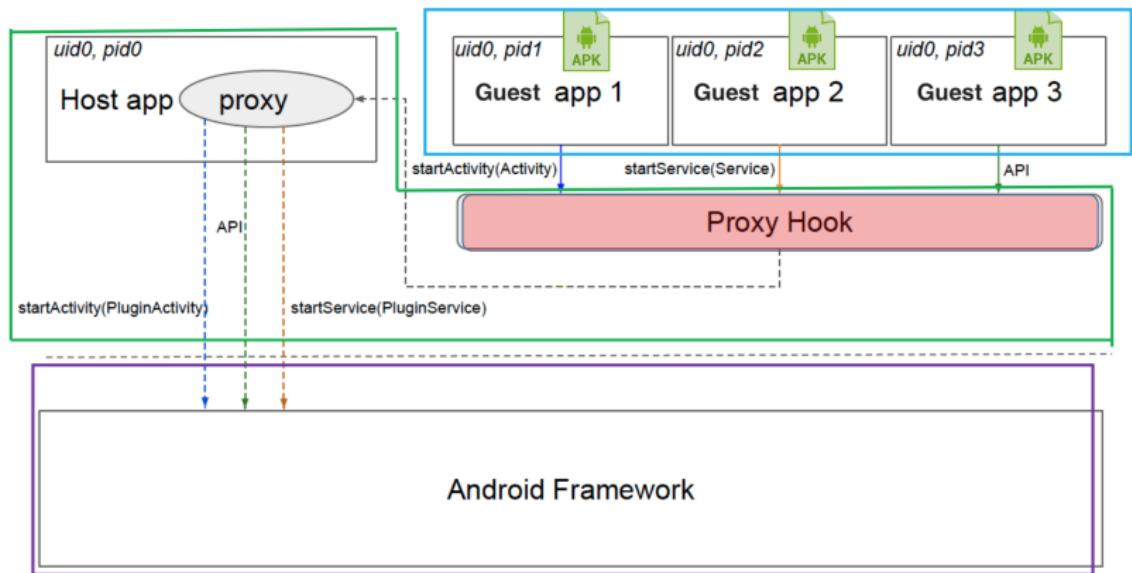


Working of Virtual Containers

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.

App Security in Virtual Containers

Application UID

- In Android, each application gets a 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's core security features are broken for a *guest* app.

The following core security features help you build secure apps:

- The Android Application Sandbox, which isolates your app data and code execution from other apps. X
- An application framework with robust implementations of common security functionality such as cryptography, permissions, and secure IPC. X
- 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. ✓
- An encrypted file system that can be enabled to protect data on lost or stolen devices. ✓
- User-granted permissions to restrict access to system features and user data. X
- 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.

```
32 <!--请求很多权限。也是为了框架使用-->
33 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
34 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
35 <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
36 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
37 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
38 <uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
39 <uses-permission android:name="android.permission.BLUETOOTH" />
40 <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
41 <uses-permission android:name="android.permission.BODY_SENSORS" />
42 <uses-permission android:name="android.permission.BROADCAST_STICKY" />
43 <uses-permission android:name="android.permission.CALL_PHONE" />
44 <uses-permission android:name="android.permission.CAMERA" />
45 <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
46 <uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
47 <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
48 <uses-permission android:name="android.permission.CLEAR_APP_CACHE" />
49 <uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
50 <uses-permission android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION" />
51 <uses-permission android:name="android.permission.DRAW_STATUS_BAR" />
52 <uses-permission android:name="android.permission.GET_ACCOUNTS" />
53 <uses-permission android:name="android.permission.GET_ANIMATION_RESOURCES" />
54 <uses-permission android:name="android.permission.GET_ACCOUNTS" />
55 <uses-permission android:name="android.permission.GET_PACKAGE_SIZE" />
56 <uses-permission android:name="android.permission.GET_TASKS" />
57 <uses-permission android:name="android.permission.INTERNET" />
58 <uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES" />
59 <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
60 <uses-permission android:name="android.permission.NFC" />
61 <uses-permission android:name="android.permission.PERSISTENT_ACTIVITY" />
62 <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
63 <uses-permission android:name="android.permission.READ_CALENDAR" />
64 <uses-permission android:name="android.permission.READ_CALL_LOG" />
65 <uses-permission android:name="android.permission.READ_CELL_BROADCASTS" />
66 <uses-permission android:name="android.permission.READ_CONTACTS" />
67 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
68 <uses-permission android:name="android.permission.READ_INSTALL_SESSIONS" />
69 <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**.

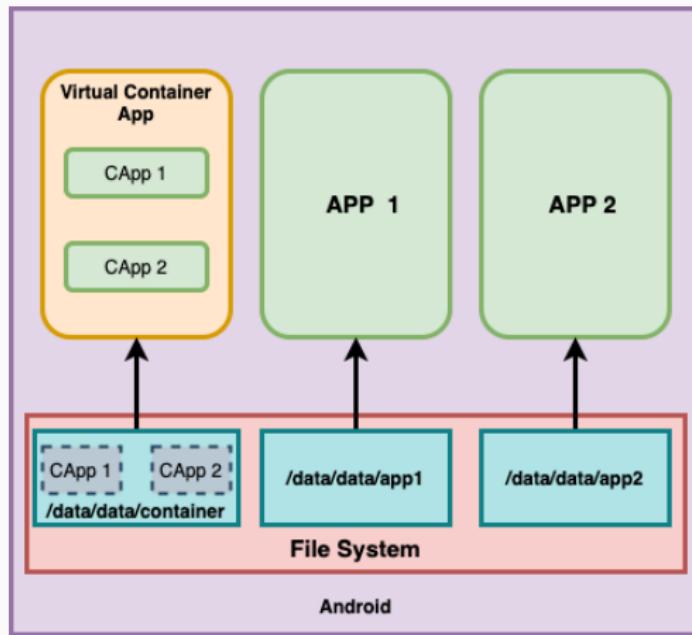
The screenshot shows two tables side-by-side. The left table is titled 'App List' and lists various installed applications. The right table is titled 'Running Apps List' and lists applications currently running on the device. Both tables include columns for the app icon, name, and process ID (PID).

App List		
Biometric Authentication	10218	TestWebview
Grab	10218	Grab
Google Services Framework	10218	Fake Biometric Authentication
andOTP (dev)	10218	Uber
Conware	10218	Google Play services
Ola	10218	

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
Grab	10218	Grab
andOTP (dev)	10218	26570
Uber	10218	26191

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.

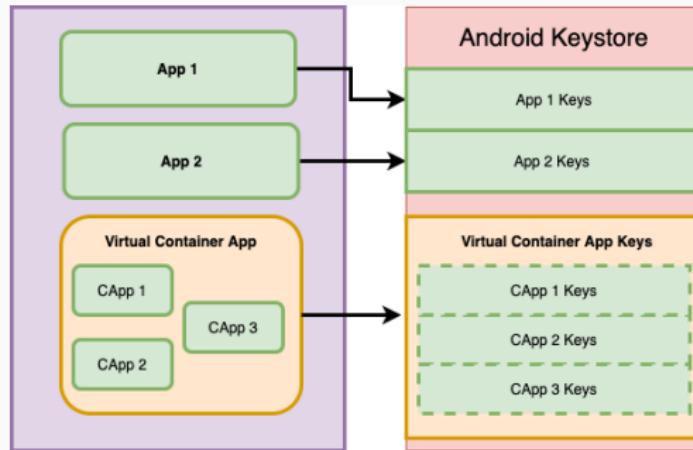


Android Keystore

- *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).

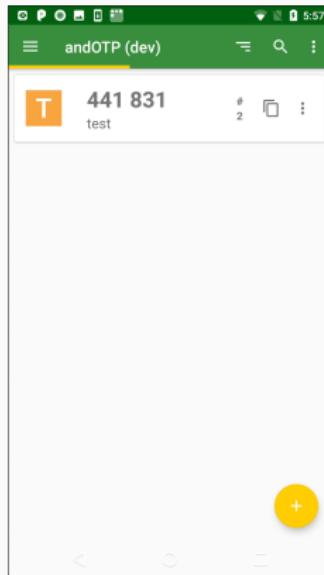
Android Keystore

- 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**

DEMO: Attacking Filesystem and Keystore



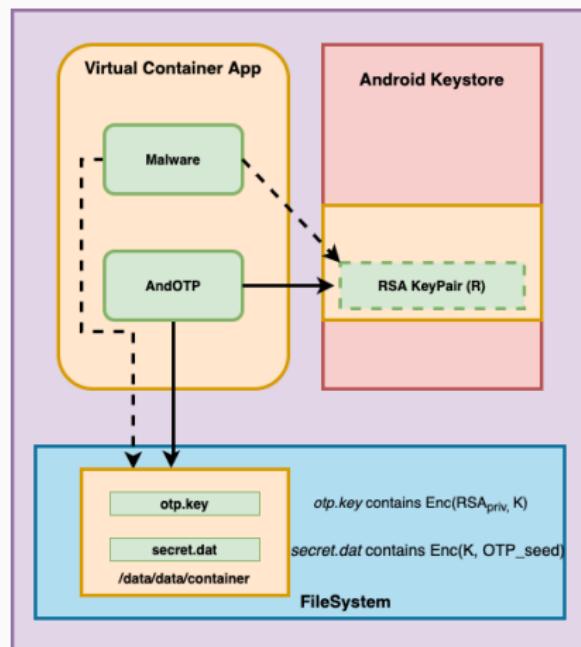
- **andOTP** - App for generating 2FA TOTP and HOTP tokens.
- **Version:** 0.7.0
- <https://github.com/andOTP>

DEMO: Attacking Filesystem and Keystore

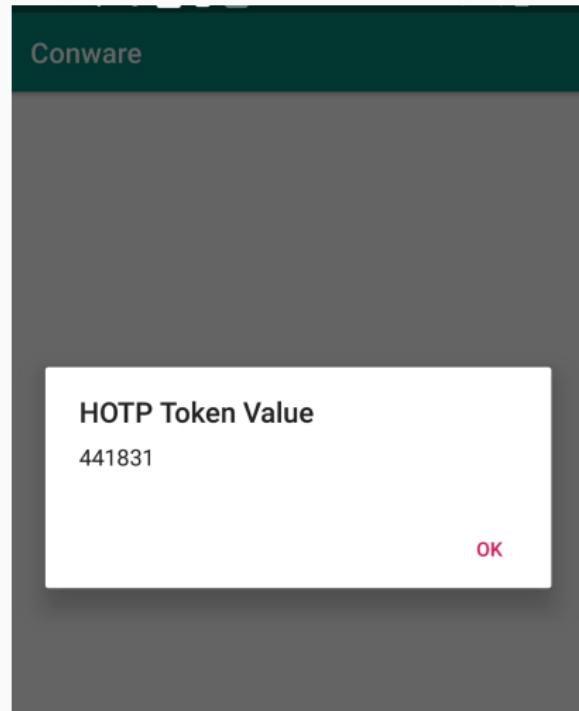
- Option to have **encrypted data backup** using a hardware-backed keystore.
- OTP seeds and related data is encrypted using AES key K and stored in **secret.dat** file.
- AES key K is generated using the user provided password.
- Key K, in turn, is encrypted using a RSA keypair generated in a hardware-backed keystore and stored in **otp.key** file.
- In this attack, a *guest* app will access sandbox data and keystore keys of *guest* andOTP app.

DEMO: Attacking Filesystem and Keystore

- Using a malware *guest* app, we can decrypt the data, steal the OTP seeds and generate OTPs.



DEMO: Video



Pentesting Environment

Virtual Containers as a Pentesting Environment

- A virtual container can provide a basic pentesting environment.
- No need to always **patch, re-compile and re-sign** a *guest* app while pentesting.
- Many *runtime application self-protection* (RASP) mechanisms can be bypassed in such environment.

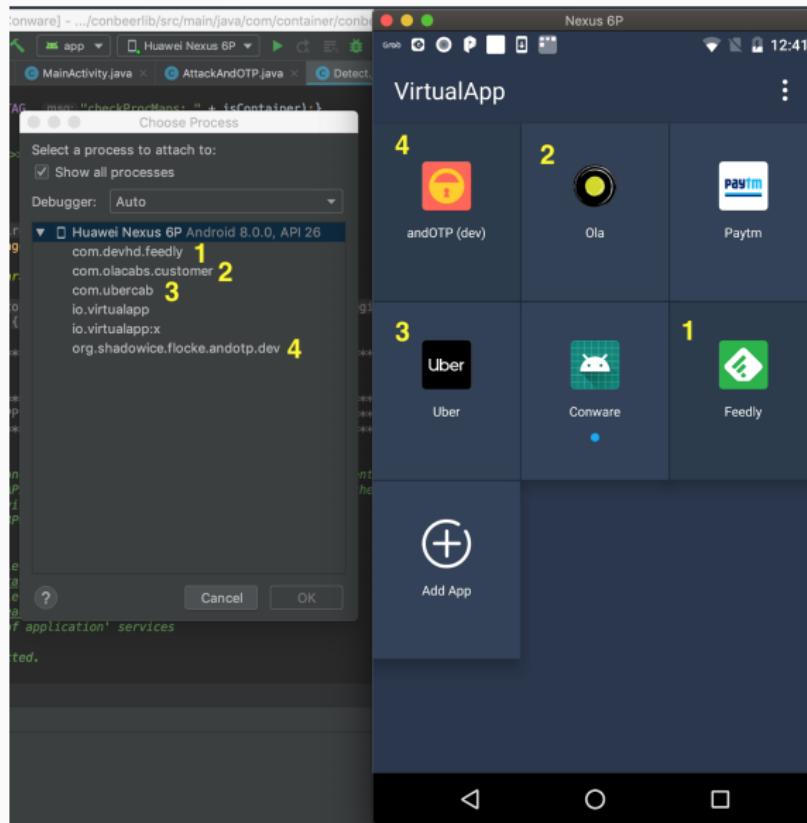
```
└ apktool d test.apk
I: Using Apktool 2.4.1 on test.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /Users/lostboy/Library/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
Exception in thread "main" java.lang.NullPointerException
    at brut.androlib.res.data.value.ResEnumAttr.serializeBody(ResEnumAttr.java:56)
    at brut.androlib.res.data.value.ResAttr.serializeToResValuesXml(ResAttr.java:64)
    at brut.androlib.res.AndrolibResources.generateValuesFile(AndrolibResources.java:742)
    at brut.androlib.res.AndrolibResources.decode(AndrolibResources.java:263)
    at brut.androlib.Androlib.decodeResourcesFull(Androlib.java:129)
    at brut.androlib.ApkDecoder.decode(ApkDecoder.java:124)
    at brut.apktool.Main.cmdDecode(Main.java:170)
    at brut.apktool.Main.main(Main.java:76)
```

Apktool can fail sometimes

Android Manifest - Debuggable

- `android:debuggable='true'` is needed to debug an Android application.
- By default an application is **not** debuggable.
- For a 3rd party app from Play Store, requires recompiling and resigning an app using apktool
- If the *host* is debuggable, then all the *guest* apps become debuggable.
 - Possible to attach Android debugger to *guest* apps
 - *ptrace* can be attached between *guest* apps and from host app to *guest* app and vice versa
- With container being debuggable, no additional steps for recompiling and resigning needed.

Android Manifest - Debuggable



Android Manifest - Network Security Config

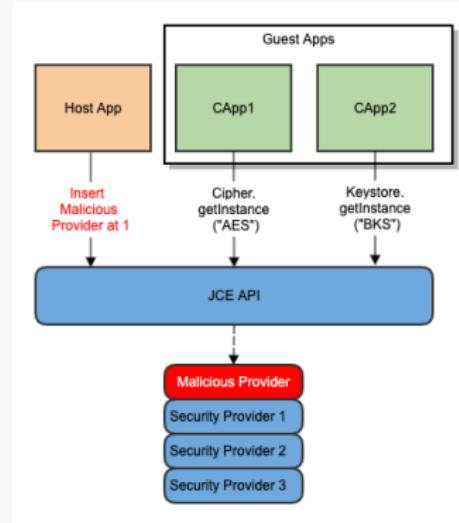
- Since Android Nougat (7.0), apps can customize their network security settings.
- User installed TLS CA certs are not trusted by default, requires explicit declaration in manifest file.
- By setting attribute *android:networkSecurityConfig*.
- Network Security Configuration of *host* will be inherited 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>
```

Binary Injection - Insert fake 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.*

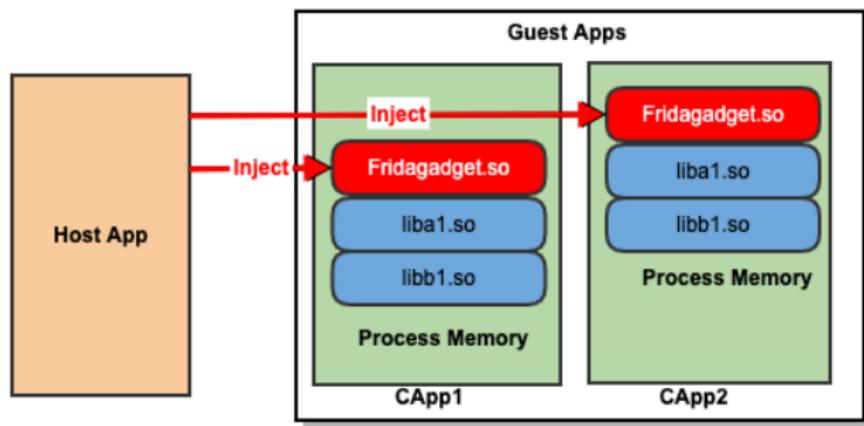


Java Security Provider: andOTP



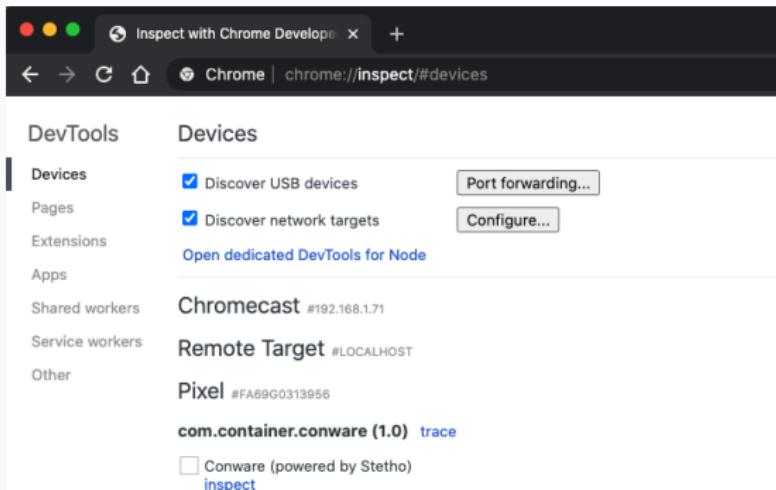
Binary Injection - Inject Frida for Dynamic Instrumentation

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



Binary Injection - Inject Stetho for App Sandbox Inspection

- **Stetho** is a sophisticated developer friendly tool by Facebook.
- Exposes functionality via Chrome Developer Tools or command-line interface.
- Stetho is useful for reverse engineers too.
 - *Stetho can be initialized in the context of a guest app.*
 - Easily inspect sandbox data of a *guest app* from desktop Chrome browser.



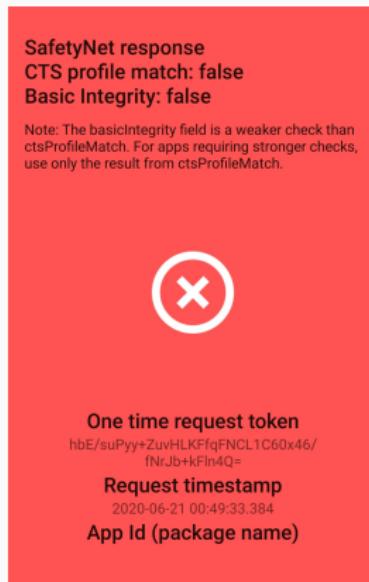
Summary of Attacks

- **Android Manifest**
 - Permissions
 - Debuggable
 - Network Security Config
- List all running *guest* apps
- Application's **Filesystem Sandbox**
- **Android Keystore**
- Manipulating **Java Security Providers**
- **Dynamic instrumentation** of a *guest* app
- Inspection of *guest* app sandbox data
- Not an exhaustive list, many other attacks may be possible.
 - **Android ID:** Used for anti-cloning purposes but it is same for all *guest* apps.

Detecting Virtual Containers

SafetyNet

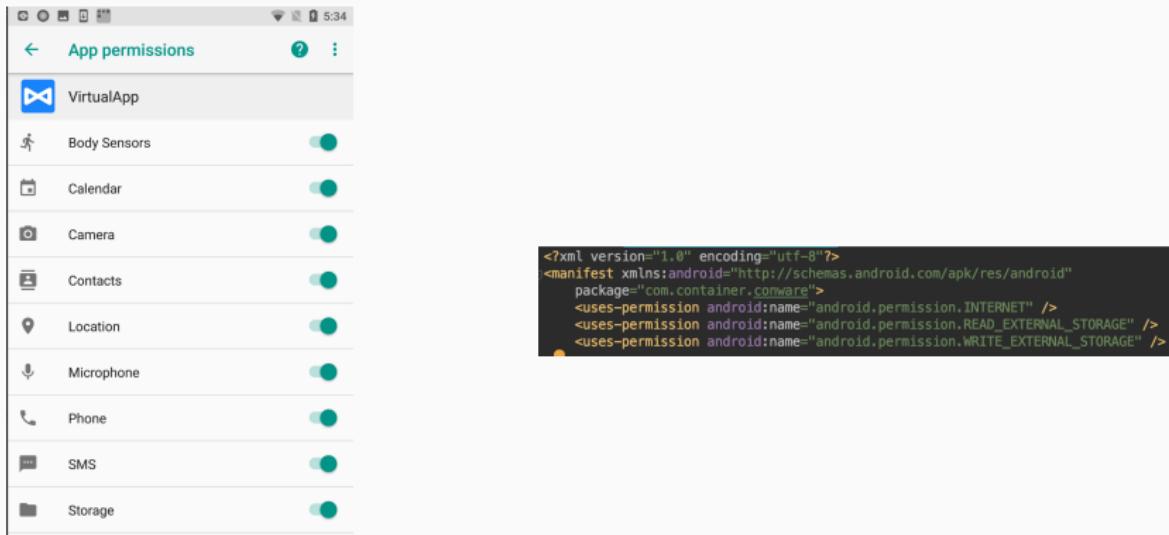
- A set of services and APIs that help protect app against security threats:
 - Device tampering
 - Bad URLs
 - Potentially harmful apps
 - Fake users
- SafetyNet check **fails** for a *guest* app.
- **The *basic integrity & CTS profile match* checks fail.**
- Not all virtual containers support Google Play Services.



Detection Using Heuristics

1. Manifest Permissions

- In a container, a *guest* app may have more permissions than declared in manifest.



Detection Using Heuristics

2. Storage directory

- Data stored inside virtual container app's directory.
- On device - /data/data/app_package_name
- In Parallel Space -

/data/data/com.lbe.parallel.intl.arm64/...

```
angler:/data/data/org.shadowice.flocke.andotp.dev # pwd  
/data/data/org.shadowice.flocke.andotp.dev andOTP storage dir on device  
angler:/data/data/org.shadowice.flocke.andotp.dev # cd /data/data/com.lbe  
com.lbe.parallel.intl.arm64/ com.lbe.parallel.intl/ andOTP storage dir in Parallel Space  
lbe.parallel.intl.arm64/parallel_intl/0/org.shadowice.flocke.andotp.dev/  
angler:/data/data/com.lbe.parallel.intl.arm64/parallel_intl/0/org.shadowice.flocke.andotp.dev # pwd  
/data/data/com.lbe.parallel.intl.arm64/parallel_intl/0/org.shadowice.flocke.andotp.dev  
angler:/data/data/com.lbe.parallel.intl.arm64/parallel_intl/0/org.shadowice.flocke.andotp.dev # █
```

Detection Using Heuristics

3. Process Memory

- Check for files in memory not belonging to the app's filepath.
- In /proc/self/maps artifacts should be only from either paths:
 - /data/app/app_package_name
 - /data/data/app_package_name

```
7153e8f000-7153e1000 r-xp 00000000 fd:00 1884753  
7153eb1000-7153eb2000 r-p 00012000 fd:00 1884753  
7153eb2000-7153eb3000 rw-p 00013000 fd:00 1884753  
7153ef3000-7154745000 r-p 00000000 00:04 547009  
0$54f4cw==/base.apk (deleted) Artifacts with path not belonging to the application
```

Expected Path

```
/data/app/com.lbe.parallel.intl-Na00PMy0c1ws0v8s4fcw==/lib/arm/libdaciclient_64.so  
/data/app/com.lbe.parallel.intl-Na00PMy0c1ws0v8s4fcw==/lib/arm/libdaciclient_64.so  
/data/app/com.lbe.parallel.intl-Na00PMy0c1ws0v8s4fcw==/lib/arm/libdaciclient_64.so  
/dev/ashmem/davix-k-classes.dex extracted in memory from /data/app/com.lbe.parallel.intl-Na00PMy0c1ws0v8s4fcw==/base.apk  
/data/app/com.lbe.parallel.intl-arm64-qrXnhnCnxJxI_Sz-b1TnsnQ==/oat/arm64/base.odex  
/data/app/com.lbe.parallel.intl-arm64-qrXnhnCnxJxI_Sz-b1TnsnQ==/oat/arm64/base.odex  
/data/app/com.lbe.parallel.intl-arm64-qrXnhnCnxJxI_Sz-b1TnsnQ==/oat/arm64/base.odex  
/data/app/com.lbe.parallel.intl-arm64-qrXnhnCnxJxI_Sz-b1TnsnQ==/oat/arm64/base.odex  
/data/app/com.lbe.parallel.intl-arm64-qrXnhnCnxJxI_Sz-b1TnsnQ==/oat/arm64/base.vdex  
/data/app/com.lbe.parallel.intl-arm64-qrXnhnCnxJxI_Sz-b1TnsnQ==/base.apk  
/data/app/org.shadowice.flocke.andotp.dev-4YKKj-y10Vycopstw_g_idog==/base.apk  
/data/app/org.shadowice.flocke.andotp.dev-4YKKj-y10Vycopstw_g_idog==/base.apk  
/data/app/com.lbe.parallel.intl-arm64-qrXnhnCnxJxI_Sz-b1TnsnQ==/base.apk  
/data/app/com.lbe.parallel.intl-arm64-qrXnhnCnxJxI_Sz-b1TnsnQ==/base.apk  
/data/app/com.lbe.parallel.intl-arm64-qrXnhnCnxJxI_Sz-b1TnsnQ==/oat/arm64/base.art  
/16f7609000-716f7a0000 r-p 00000000 fd:00 1884579
```

Detection Using Heuristics

4. Environment Variables

- Check declared environment variables like *LD_PRELOAD* and many other custom variables.
- Example: Parallel Space declare variables like *WHITELIST_SRC22*

Command Execution

Parallel Space Environment Variables
WHITELIST_SRC22=/sdcard/Podcasts/ WHITELIST_SRC22=/sdcard/Alarms/ WHITELIST_SRC23=/sdcard/Movies/ WHITELIST_SRC24=/sdcard/DCIM/ WHITELIST_SRC25=/sdcard/Notifications/ WHITELIST_SRC26=/sdcard/Download/ WHITELIST_SRC27=/sdcard/Ringtones/ WHITELIST_SRC28=/storage/emulated/0/Pictures/ WHITELIST_SRC29=/storage/emulated/0/.magic/ WHITELIST_SRC30=/storage/emulated/0/Android/obb/ WHITELIST_SRC31=/storage/emulated/0/Documents/ WHITELIST_SRC32=/storage/emulated/0/Podcasts/ WHITELIST_SRC33=/storage/emulated/0/Alarms/ WHITELIST_SRC34=/storage/emulated/0/Movies/ WHITELIST_SRC35=/storage/emulated/0/DCIM/ WHITELIST_SRC36=/storage/emulated/0/Notifications/ WHITELIST_SRC37=/storage/emulated/0/Download/ WHITELIST_SRC38=/storage/emulated/0/Ringtones/ FORBID_SRC1=/data/data/com.container.conware/ FORBID_SRC2=/data/user/0/com.container.conware/ DA_HOSTROOT=parallel_intl DA_HOSTDIRECT=/data/data/com.lbe.parallel.intl/ lib/libbdclient.so DA_API_LEVEL=26 LD_PRELOAD=/data/data/com.lbe.parallel.intl/lib/ libbdclient_64.so PATH=/sbin:/system/sbin:/system/bin:/system/xbin:/ vendor/bin:/vendor/xbin DOWNLOAD_CACHE=/data/cache ANDROID_BOOTLOGO=1 ANDROID_ROOT=/system ANDROID_ASSETS=/system/app ANDROID_DATA=/data

Many Env variable set

LD_PRELOAD set

Detection Using Heuristics

5. Running Services

- Many containers have their own services running.
- Check for the presence of services not belonging to the app.
- e.g: com.lbe.parallel.service.KeyguardService

```
CONBEER D >>>>> APP SERVICE NAMES <<<<<<<<
CONBEER D com.lbe.parallel.service.KeyguardService
CONBEER D cn.thinkingdata.android.TDQuitSafelyService$TDKeepAliveService
CONBEER D com.lbe.parallel.install.AppInstallService
CONBEER D com.lbe.doubleagent.service.proxy.KeepAliveService
CONBEER D checkAppServiceName: true
CONBEER D >>>>> APP SERVICE NAMES: DONE <<<<<<<
```

Running services for Parallel Space

Detection Using Heuristics

6. App Components

- Declare **disabled** app components in the Manifest and **enable** them dynamically.
- **Enabling** such components will **not work** in a virtual container.

```
<application>                                         Manifest - Component Disabled
    <receiver
        android:name=".utils.FakeBroadcastReceiver"
        android:enabled="false"
        android:exported="false"
        android:process=":fake_receiver">
        <intent-filter>
            <action
                android:name="com.container.conbeer.intent.TEST" />
        </intent-filter>
    </receiver>
ComponentName componentName = new ComponentName(this.getApplicationContext(), FakeBroadcastReceiver.class);
this.getPackageManager().setComponentEnabledSetting(componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);
```

Dynamically Enabled

Detection Using Heuristics

7. Stack Trace

- Check call stack trace for presence of unexpected classes.
- Host* apps may inject code to virtualize *Guest* app.

```
CONBEER D >>>>>>>> STACKTRACE AT RUNTIME: START<<<<<<<<<<
CONBEER D com.container.conbeer.ConBeer
chatty I uid=10235(com.ludashi.superboost) com.container.conware identical 1 line
CONBEER D com.container.conbeer.ConBeer
CONBEER D com.container.conware.Activities.DetectContainer
CONBEER D android.app.Activity
CONBEER D android.app.Activity
CONBEER D android.app.Instrumentation
CONBEER D com.lody.virtual.client.hook.delegate.InstrumentationDelegate
CONBEER D com.lody.virtual.client.hook.delegate.a
CONBEER D android.app.ActivityThread
CONBEER D android.app.ActivityThread
CONBEER D android.app.servertransaction.LaunchActivityItem
CONBEER D android.app.servertransaction.TransactionExecutor
CONBEER D android.app.servertransaction.TransactionExecutor
CONBEER D android.app.ActivityThread$H
CONBEER D android.os.Handler
CONBEER D android.os.Looper
CONBEER D android.app.ActivityThread
CONBEER D java.lang.reflect.Method
CONBEER D com.android.internal.os.RuntimeInit$MethodAndArgsCaller
CONBEER D com.android.internal.os.ZygoteInit
CONBEER D >>>>>>>> STACKTRACE AT RUNTIME: DONE<<<<<<<<<
```

Unexpected Classes

conbeerlib

- Releasing **conbeerlib**
 - In spirit of *rootbeerlib* :)
 - Open source
 - <https://github.com/su-vikas/conbeerlib>
- Detects presence of virtual containers using various heuristic techniques discussed.
- *Caution:* The library is not tested for all real world Android device madness, and may have rough edges.

conbeerlib

Checks	Apps	Parallel Space	Virtual App	Dual Space	Clone App	2Accounts	Dual Apps	Multi Parallel	Dr.Clone
Manifest Permissions	✓	✓	✓	✗	✓	✓	✓	✓	✓
Process Memory	✓	✓	✓	✓	✓	✓	✓	✓	✓
Environment	✓	✓	✓	✗	✓	✓	✓	✗	✗
Storage Directory	✓	✓	✓	✓	✓	✓	✓	✓	✓
Running Services	✓	✗	✗	✓	✓	✗	✗	✗	✓
App Components	✓	✓	✓	✗	✗	✗	✗	✓	✓
StackTrace	✓	✓	✓	✓	✓	✓	✓	✓	✓

Recommendation for Developers

Recommendations

- Do not let your app run inside a virtual container, detect and take action.
 - Also increases the cost of re-packing an app as a malware (using virtual containers).
- Do not **only** depend on security offered by OS.
 - Encrypt data at rest.
- **Network:**
 - Prevent MITM
 - Certificate Pinning
 - Another layer of encryption or signing network traffic to avoid sniffing or tampering of data in transit.

Recommendations

- **Crypto:**
 - Use combination of hardware and software components for crypto key derivation.
 - Do not rely **only** on default platform Crypto and Keystore providers
 - Use crypto APIs directly, e.g. from BouncyCastle
 - Add and use explicit security providers, for crypto and keystore both
 - `Cipher.getInstance("AES", "BC")`
 - `KeyStore.getInstance("BKS", "BC")`
- **Recommendations applicable as per threat model.**

Discussion & Conclusion

Discussion

- Implications for end users
 - Users should avoid using virtual containers for sensitive apps, at least.
- More OEMs should support running multiple instances of an app - like OnePlus and Xiaomi.
- Support implemented in AOSP itself?
- Why is virtual containers supported by Android?
 - Leaving security decision in hands of an end user is not a good idea.
- Given the security implications, we are not able to reason why it is allowed.

Conclusion

- There is a need to **run multiple instances** of an app on a device.
- Virtual containers are **convenient** to use, but **breaks Android built-in security** model.
- Both security and privacy is a concern for *guest* apps.
- Virtual containers can be used as a basic **pentesting environment**.
- SafetyNet and heuristics can be used to detect virtual containers.
- **Our contribution:**
 - Documenting and demonstrating the attacks
 - New detection techniques and releasing conbeerlib

Thank You

