

Unpack Android apps on non-rooted and factory ROM devices using virtualization technique

Jason Wang | 王羿廷

© Fourdesire. All Rights Reserved.

About Me



Jason Wang(王羿廷)

Fourdesire Senior Security Engineer

- Android Security
- Reverse Engineering
- Trying hard to build Taiwan's first independent R&D product for app security

Education & Experience

- Speaker of CYBERSEC 2019
- Speaker of CSA Taiwan Summit 2017
- Security Consultant of Digicentre
- Security Engineer of Gamania
- Bachelor Degree from Department of Applied Mathematics, NCTU

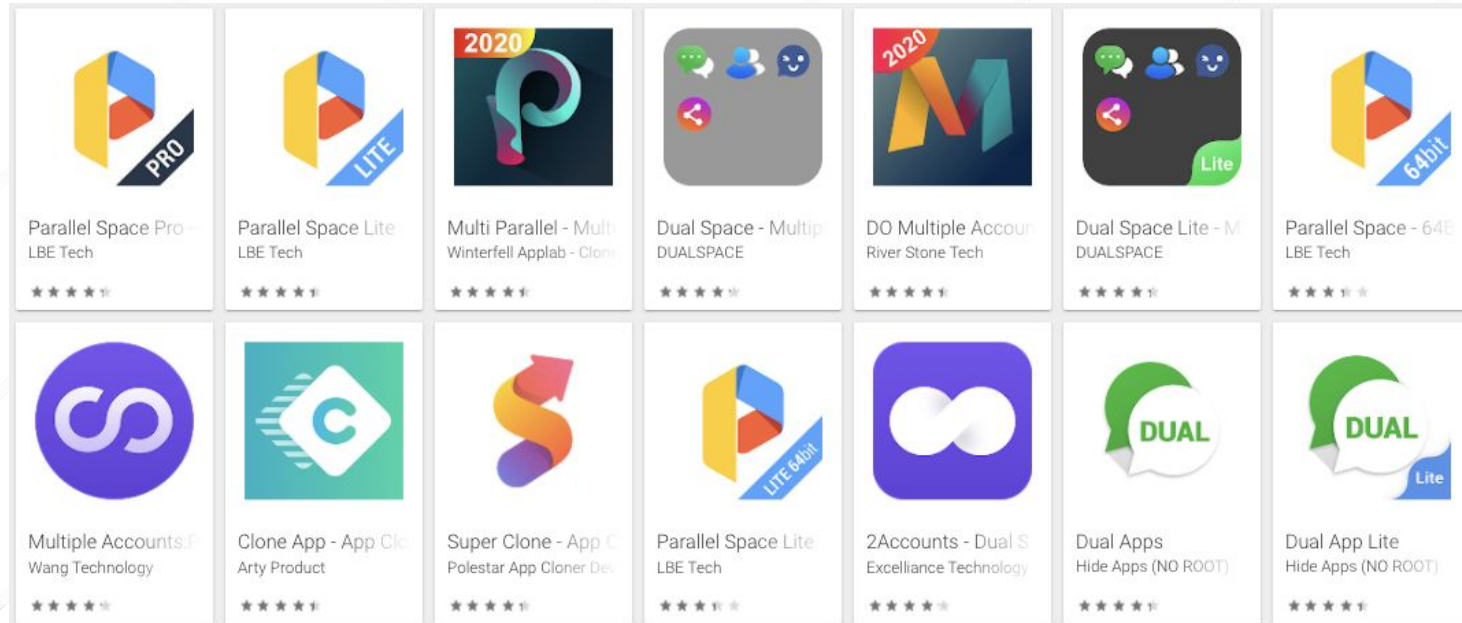
Outline

- What is virtual space on Android?
- What is Android packer?
- How to unpack an app packed by a packer?
- Implement a virtual space with unpacking functions
- Demo

What is virtual space on Android?

Virtual Space

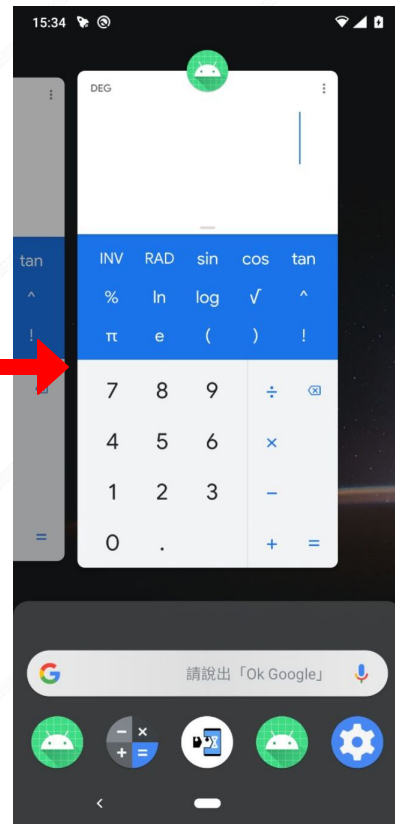
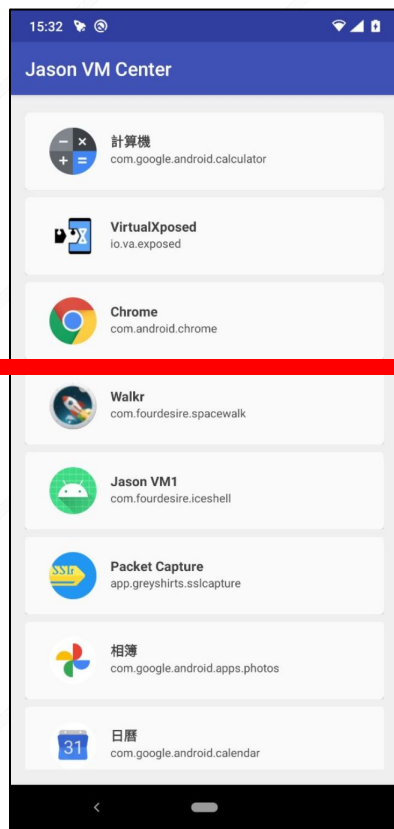
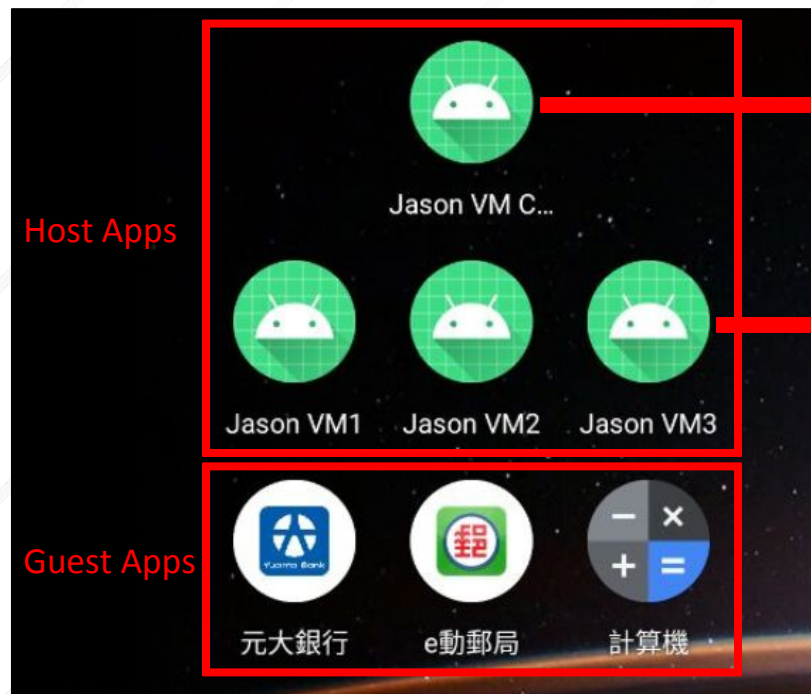
Virtual space is a virtualization technique that allows apps to run in an isolated space, just like there is an virtual machine on your Android device. By this technique, you can run multiple instances of a specific app on a Android device. For example, you can simultaneously launch two LINE apps on a device for personal and business use.



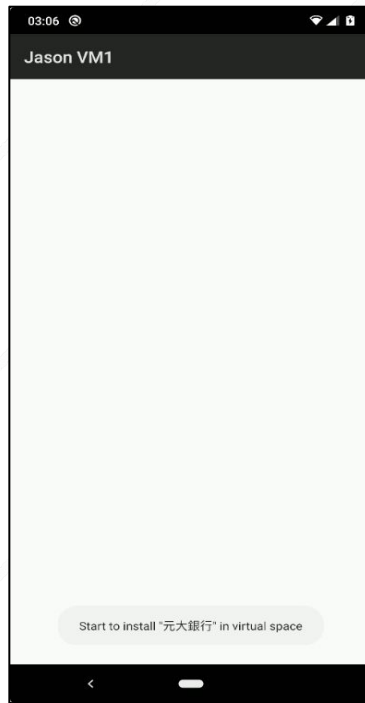
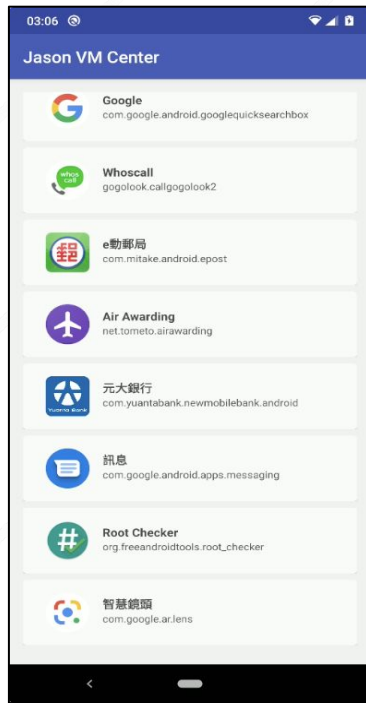
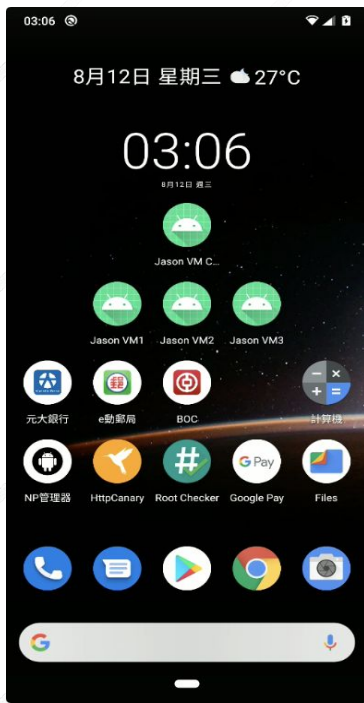
Key Features

- A host-guest structure
- Host app is an Android app installed on system
- Guest app is just an APK file stored on system
- No need to install guest app on system
- Host app dynamically load and launch guest app without installation
- This technique can be implemented on non-rooted devices
- Visit [VirtualApp](#) and [VirtualAPK](#) for more detail information

Overview Figures

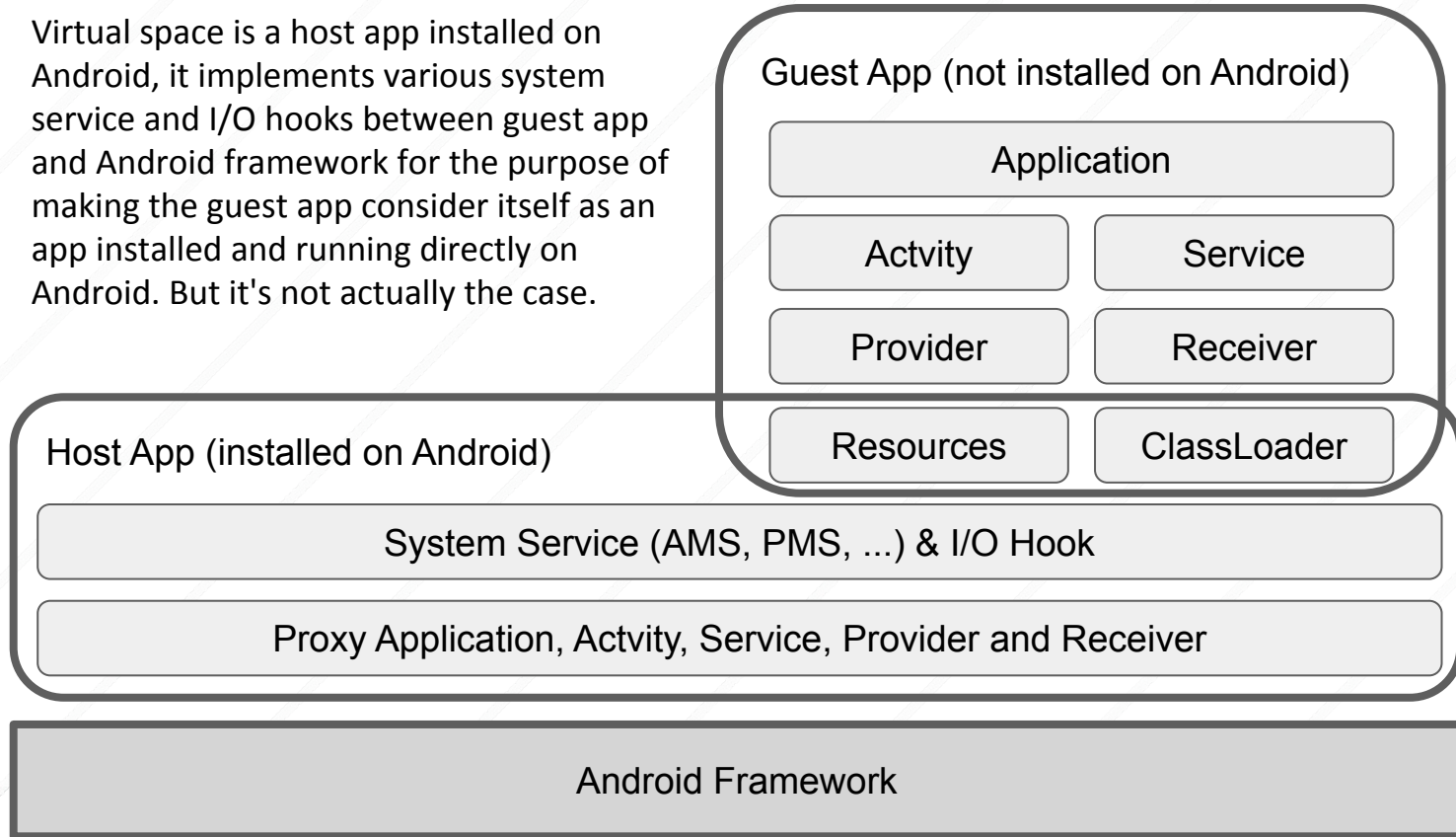


Demo



Mechanisms

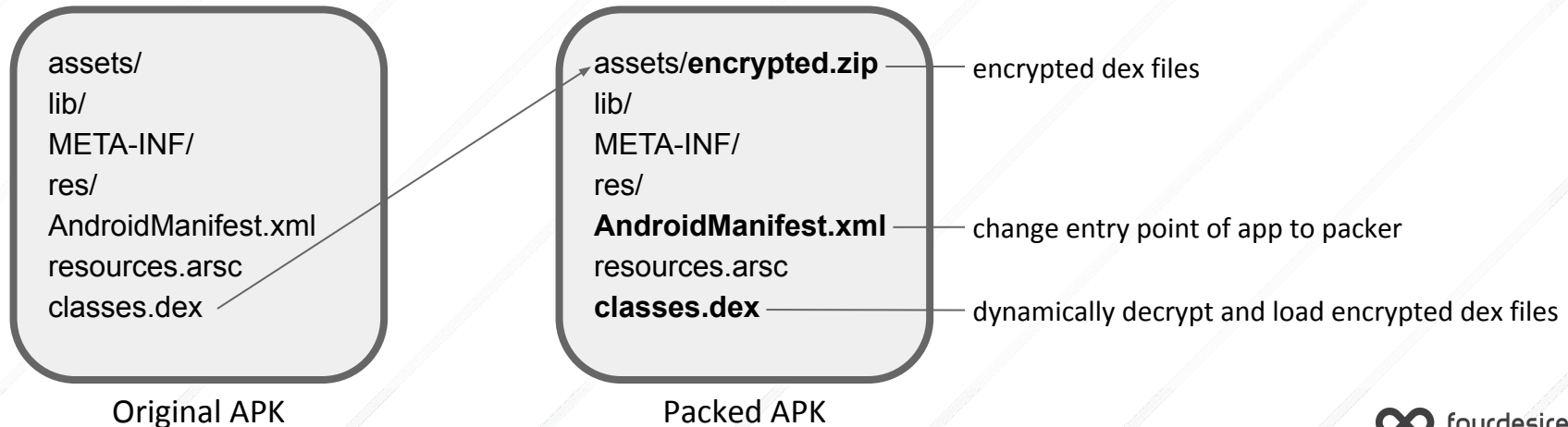
Virtual space is a host app installed on Android, it implements various system service and I/O hooks between guest app and Android framework for the purpose of making the guest app consider itself as an app installed and running directly on Android. But it's not actually the case.



What is Android packer?

Android Packer

- Encrypt dex files and stored in some special files
- Dynamically decrypt and load dex files into memory during runtime
- Please refer to [CyberSec 2019](#) for more detail information



How to unpack an app packed by a packer?

Some approaches to unpack an Android app

1. Find out the algorithm and key of encryption (very difficult)
2. Extract dex file from compiled oat file

Reference: <https://github.com/testwhat/SmaliEx>

3. Build a custom Android ROM

Reference: <https://github.com/hanbinglengyue/FART>

4. Dump dex file from process memory

Reference: <https://github.com/hluwa/FRIDA-DEXDump>

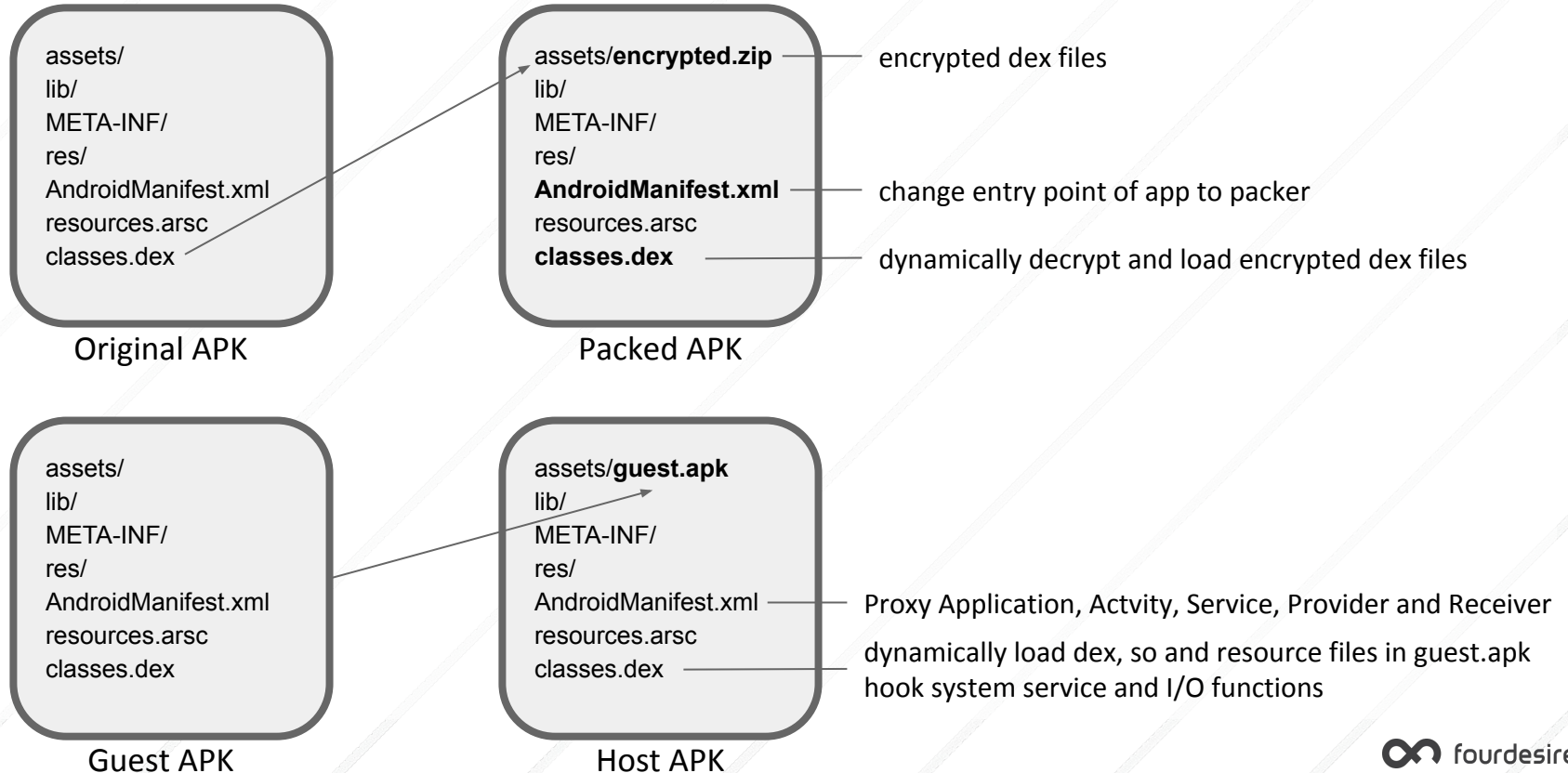
5. Hook functions in libart.so

Reference: <https://github.com/dstmath/frida-unpack>

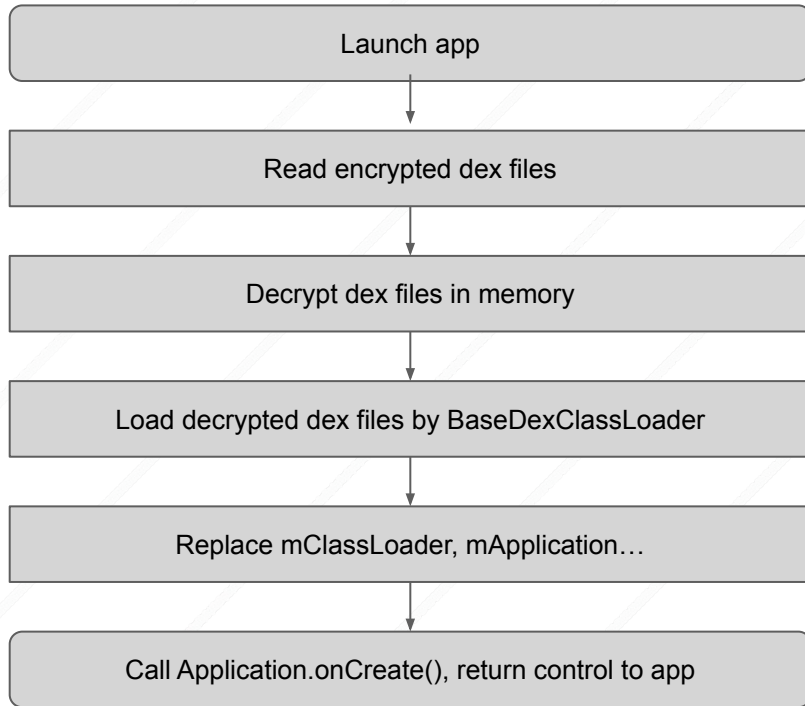
Almost all approaches require root privileges or custom ROM

Implement a virtual space with unpacking functions

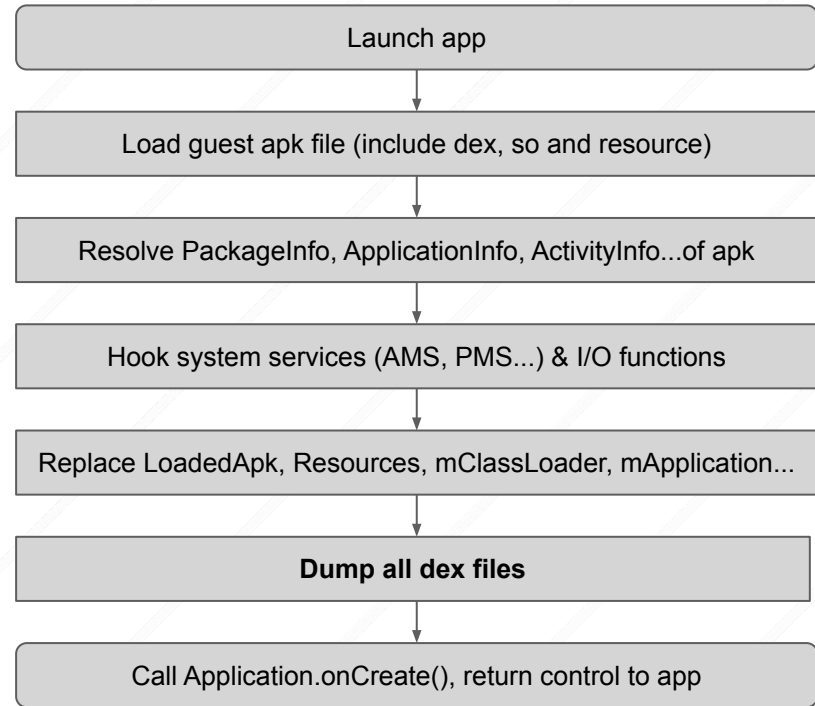
Packer & Virtual Space



Launch Process Comparison

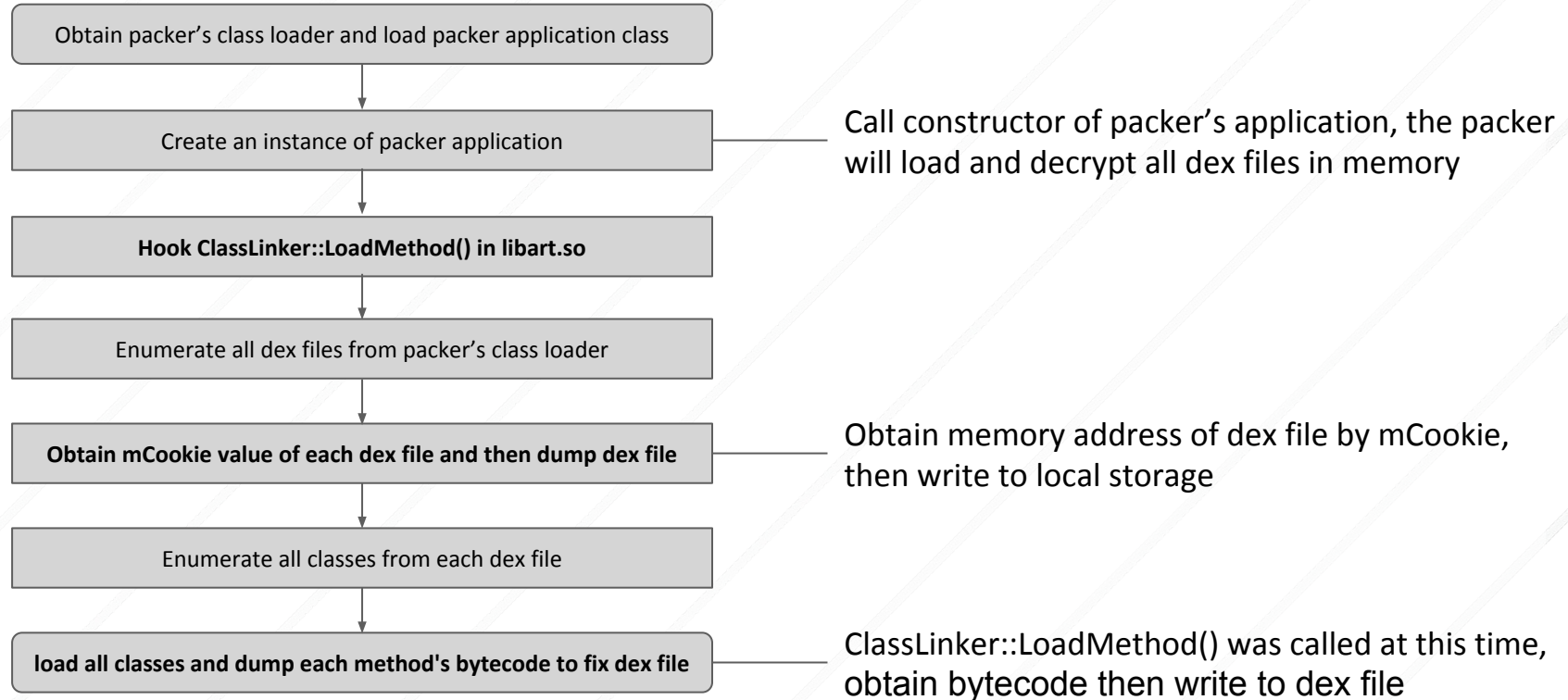


Packed App



Virtual Space Host App

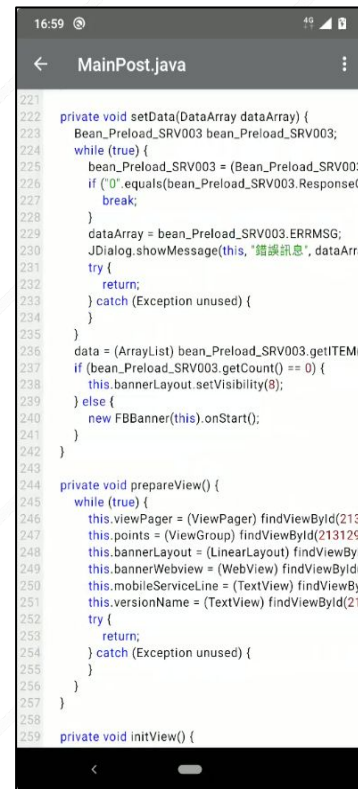
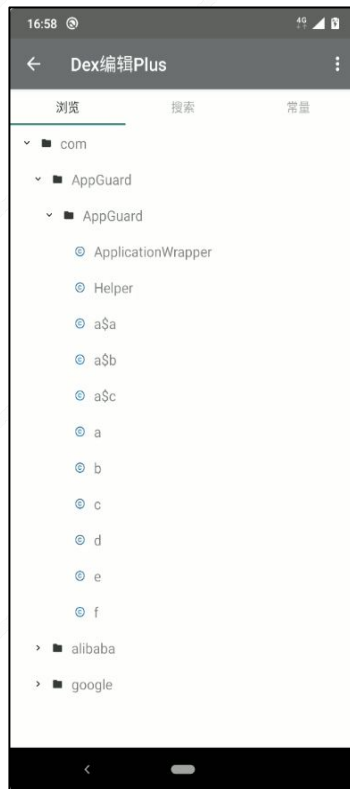
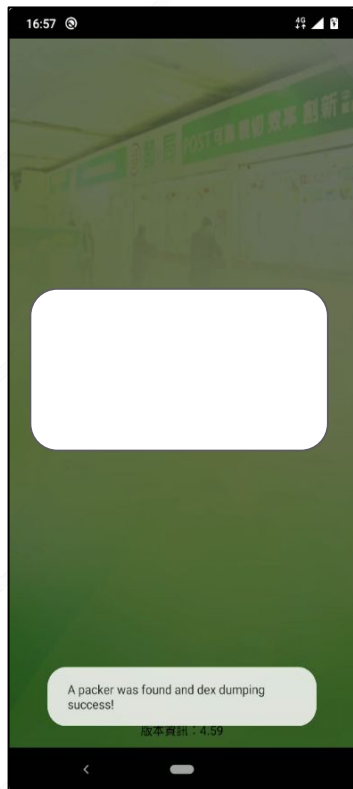
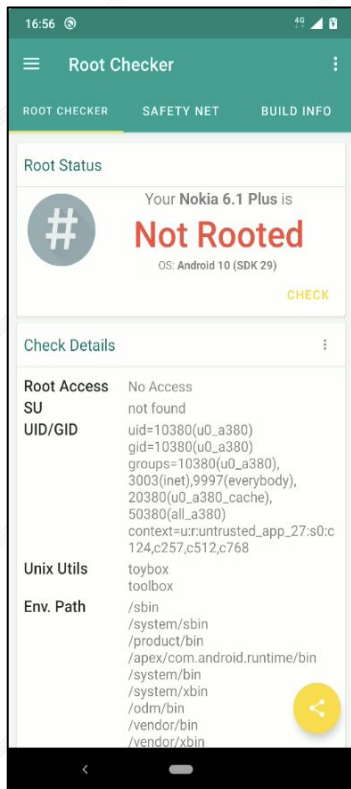
How to dump dex files



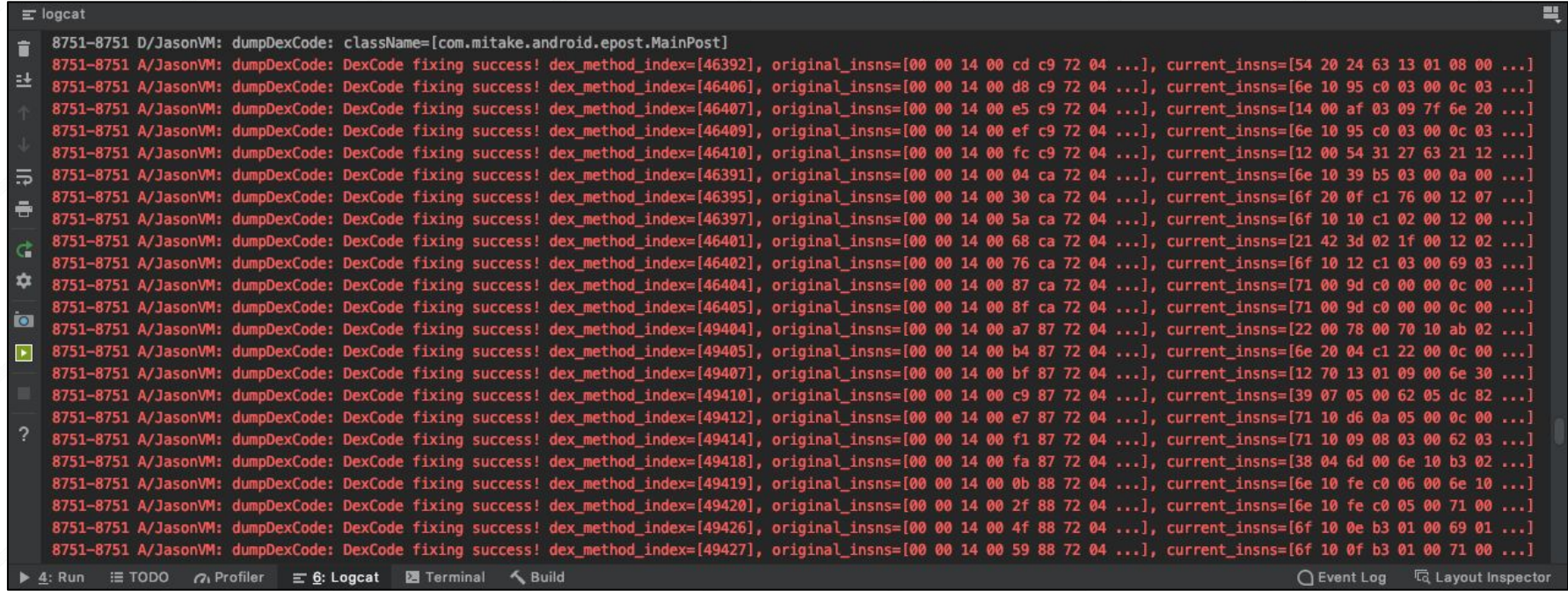
Implementation

```
/* An Android unpacking technique implemented by Jason Wang(王羿廷) for CYBERSEC 2020*/
// Obtain packer's class loader
ClassLoader packerClassLoader = getClassLoader();
// Load guest Application class and create an instance of guest application
packerClassLoader.loadClass("com.xxx.xxx.ApplicationWrapper").newInstance();
// Enumerate all dex files from packer's class loader
Object dexPathList = ReflectionHelper.getObjectField("dalvik.system.BaseDexClassLoader", packerClassLoader, "pathList");
Object[] dexElements = (Object[]) ReflectionHelper.getObjectField(dexPathList, "dexElements");
for (Object aDexElement : dexElements)
{
    // Obtain mCookie value of each dex file
    DexFile dexFile = (DexFile) ReflectionHelper.getObjectField("dalvik.system.DexPathList$Element", aDexElement, "dexFile");
    long[] mCookie = (long[]) ReflectionHelper.getObjectField(dexFile, "mCookie");
    // 1. dump entire dex file by mCookie
    dumpDexFile(mCookie);
    // Enumerate all classes from the dex file
    Enumeration<String> dexEntries = dexFile.entries();
    while (dexEntries.hasMoreElements())
    {
        // Obtain class name
        String className = dexEntries.nextElement();
        Log.d(TAG, "dumpDexCode: className=[" + className + "]");
        // 2. load this class and dump each method's bytecode to fix the dex file
        packerClassLoader.loadClass(className);
    }
}
```

Demo



Dump bytecode of each method



```
logcat
8751-8751 D/JasonVM: dumpDexCode: className=[com.mitake.android.epost.MainPost]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46392], original_insns=[00 00 14 00 cd c9 72 04 ...], current_insns=[54 20 24 63 13 01 08 00 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46406], original_insns=[00 00 14 00 d8 c9 72 04 ...], current_insns=[6e 10 95 c0 03 00 0c 03 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46407], original_insns=[00 00 14 00 e5 c9 72 04 ...], current_insns=[14 00 af 03 09 7f 6e 20 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46409], original_insns=[00 00 14 00 ef c9 72 04 ...], current_insns=[6e 10 95 c0 03 00 0c 03 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46410], original_insns=[00 00 14 00 fc c9 72 04 ...], current_insns=[12 00 54 31 27 63 21 12 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46391], original_insns=[00 00 14 00 04 ca 72 04 ...], current_insns=[6e 10 39 b5 03 00 0a 00 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46395], original_insns=[00 00 14 00 30 ca 72 04 ...], current_insns=[6f 20 0f c1 76 00 12 07 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46397], original_insns=[00 00 14 00 5a ca 72 04 ...], current_insns=[6f 10 10 c1 02 00 12 00 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46401], original_insns=[00 00 14 00 68 ca 72 04 ...], current_insns=[21 42 3d 02 1f 00 12 02 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46402], original_insns=[00 00 14 00 76 ca 72 04 ...], current_insns=[6f 10 12 c1 03 00 69 03 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46404], original_insns=[00 00 14 00 87 ca 72 04 ...], current_insns=[71 00 9d c0 00 00 0c 00 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[46405], original_insns=[00 00 14 00 8f ca 72 04 ...], current_insns=[71 00 9d c0 00 00 0c 00 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49404], original_insns=[00 00 14 00 a7 87 72 04 ...], current_insns=[22 00 78 00 70 10 ab 02 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49405], original_insns=[00 00 14 00 b4 87 72 04 ...], current_insns=[6e 20 04 c1 22 00 0c 00 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49407], original_insns=[00 00 14 00 bf 87 72 04 ...], current_insns=[12 70 13 01 09 00 6e 30 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49410], original_insns=[00 00 14 00 c9 87 72 04 ...], current_insns=[39 07 05 00 62 05 dc 82 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49412], original_insns=[00 00 14 00 e7 87 72 04 ...], current_insns=[71 10 d6 0a 05 00 0c 00 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49414], original_insns=[00 00 14 00 f1 87 72 04 ...], current_insns=[71 10 09 08 03 00 62 03 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49418], original_insns=[00 00 14 00 fa 87 72 04 ...], current_insns=[38 04 6d 00 6e 10 b3 02 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49419], original_insns=[00 00 14 00 0b 88 72 04 ...], current_insns=[6e 10 fe c0 06 00 6e 10 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49420], original_insns=[00 00 14 00 2f 88 72 04 ...], current_insns=[6e 10 fe c0 05 00 71 00 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49426], original_insns=[00 00 14 00 4f 88 72 04 ...], current_insns=[6f 10 0e b3 01 00 69 01 ...]
8751-8751 A/JasonVM: dumpDexCode: DexCode fixing success! dex_method_index=[49427], original_insns=[00 00 14 00 59 88 72 04 ...], current_insns=[6f 10 0f b3 01 00 71 00 ...]
```

Recover bytecode of each method

```
dumpDexCode: className=[com.mitake.android.epost.MainPost]
dumpDexCode: DexCode fixing success! dex_method_index=[46392], original_insns=[00 00 14 00 cd c9 72 04 ...], current_insns=[54 20 24 63 13 01 08 00 ...]
```

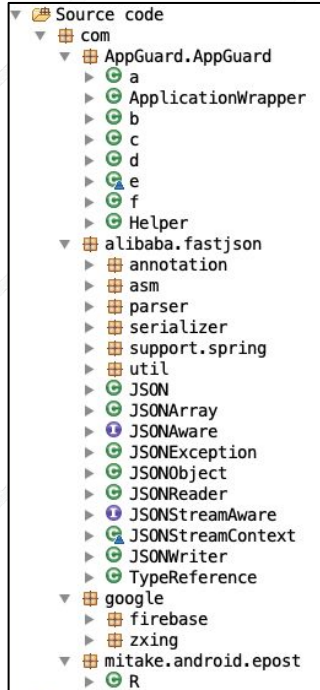
```
004A3EB8      # .....
004A3EB8      # Method 46392 (0xb538)
004A3EB8      word_4A3EB8: .short 3          # DATA XREF: CODE:001AF913+1
004A3EB8      # Number of registers : 0x3
004A3EBA      .short 1        # Size of input args (in words) : 0x1
004A3EBC      .short 2        # Size of output args (in words) : 0x2
004A3EBE      .short 1        # Number of try_items : 0x1
004A3EC0      C9CD 0472      .int 0x472C9CD    # Debug info
004A3EC4      0040 0000      .int 0x40         # Size of bytecode (in 16-bit units): 0x40
004A3EC8      # Source file: MainPost.java
004A3EC8      # catch Exception:
004A3EC8      private void com.mitake.android.epost.MainPost.initView()
004A3EC8      this = v2      # DATA XREF: CODE:004A3F54+1
004A3EC8      0000          nop
004A3ECA      0014 C9CD 0472  nop
004A3ED0      0000          const v0, 0x472C9CD
004A3ED2      0000          nop
004A3ED4      0000          nop
004A3ED6      0000          nop
```

First Dump:
dump entire dex file

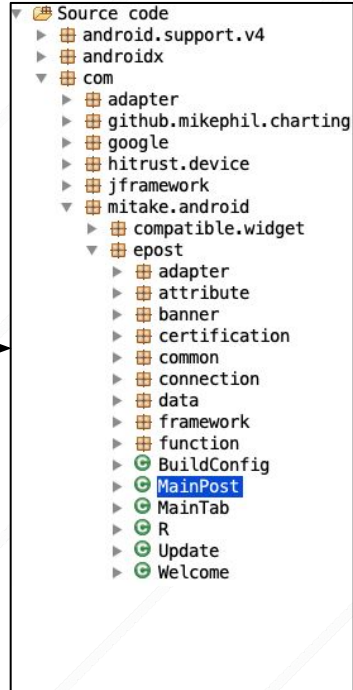
```
004A3EB8      # .....
004A3EB8      # Method 46392 (0xb538)
004A3EB8      word_4A3EB8: .short 3          # DATA XREF: CODE:001AF913+1
004A3EB8      # Number of registers : 0x3
004A3EBA      .short 1        # Size of input args (in words) : 0x1
004A3EBC      .short 2        # Size of output args (in words) : 0x2
004A3EBE      .short 1        # Number of try_items : 0x1
004A3EC0      C9CD 0072      .int word_72C9CD  # Debug info
004A3EC4      0040 0000      .int 0x40         # Size of bytecode (in 16-bit units): 0x40
004A3EC8      # Source file: MainPost.java
004A3EC8      # catch Exception:
004A3EC8      private void com.mitake.android.epost.MainPost.initView()
004A3EC8      this = v2      # CODE XREF: MainPost onCreate@VL+52+p
004A3EC8      # DATA XREF: CODE:004A3F54+1
004A3EC8      .line 230
004A3EC8      2054 6324      iget-object v0, this, MainPost_bannerWebView
004A3ECC      0113 0008      const/16 v1, 8
004A3ED0      206E 0ED6 0010  invoke-virtual {v0, v1}, <void WebView.setVisibility(
004A3ED6      .line 231
004A3ED6      2054 6329      iget-object v0, this, MainPost_mobileServiceLine
```

Second Dump:
dump bytecode of each method

Unpacking Result



Packed APK



Unpacked APK

```
private void initView() {  
}
```

```
private void parseTwp01(DataArray dataArray) {  
}
```

First Dump

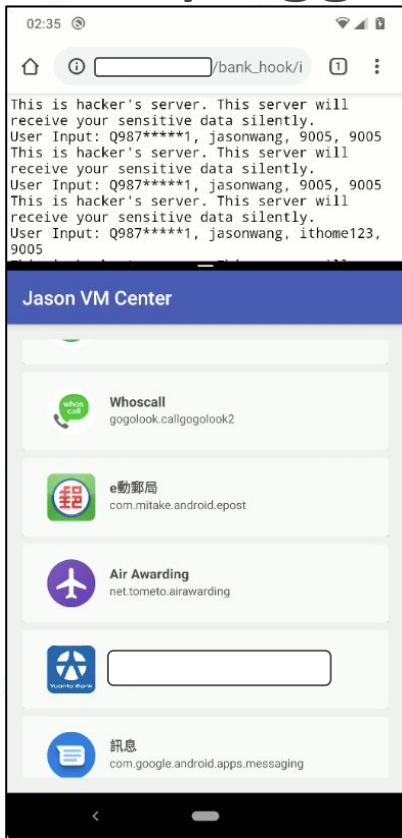
```
private void initView() {  
    this.bannerWebView.setVisibility(8);  
    this.mobileServiceLine.setText(Html.fromHtml(getString(R.string.mobile_service_line)));  
    this.mobileServiceLine.setOnClickListener(this.setOnClickListener);  
    this.versionName.setText("版本資訊: " + PhoneInfoHelper.getAppVersion(this) + GlobalProperties.getDebugVersion());  
}
```

```
/* JADX WARNING: type inference failed for: r2v0, types: [android.content.Context, com.mitake.android.epost.MainPost]  
private void parseTwp01(DataArray dataArray) {  
    Bean_Banking_TWP0000_1 bean_Banking_TWP0000_1 = (Bean_Banking_TWP0000_1) dataArray.get().get(0);  
    if (bean_Banking_TWP0000_1.getResponseCode() != 0) {  
        GlobalProperties.twPayUrl = "";  
        JDIALOG.showMessages(this, getString(R.string.error_message), bean_Banking_TWP0000_1.ERRMSG, new 4());  
        return;  
    }  
    int result = bean_Banking_TWP0000_1.getResult();  
    if (result == 1) {  
        gridViewClick(9);  
    } else if (result != 2) {  
        JDIALOG.showMessages(this, getString(R.string.error_message), "QRCode錯誤，請重新掃描");  
    } else {  
        JDIALOG.showMessages(this, getString(R.string.error_message), "請使用郵保鑄交易");  
    }  
}
```

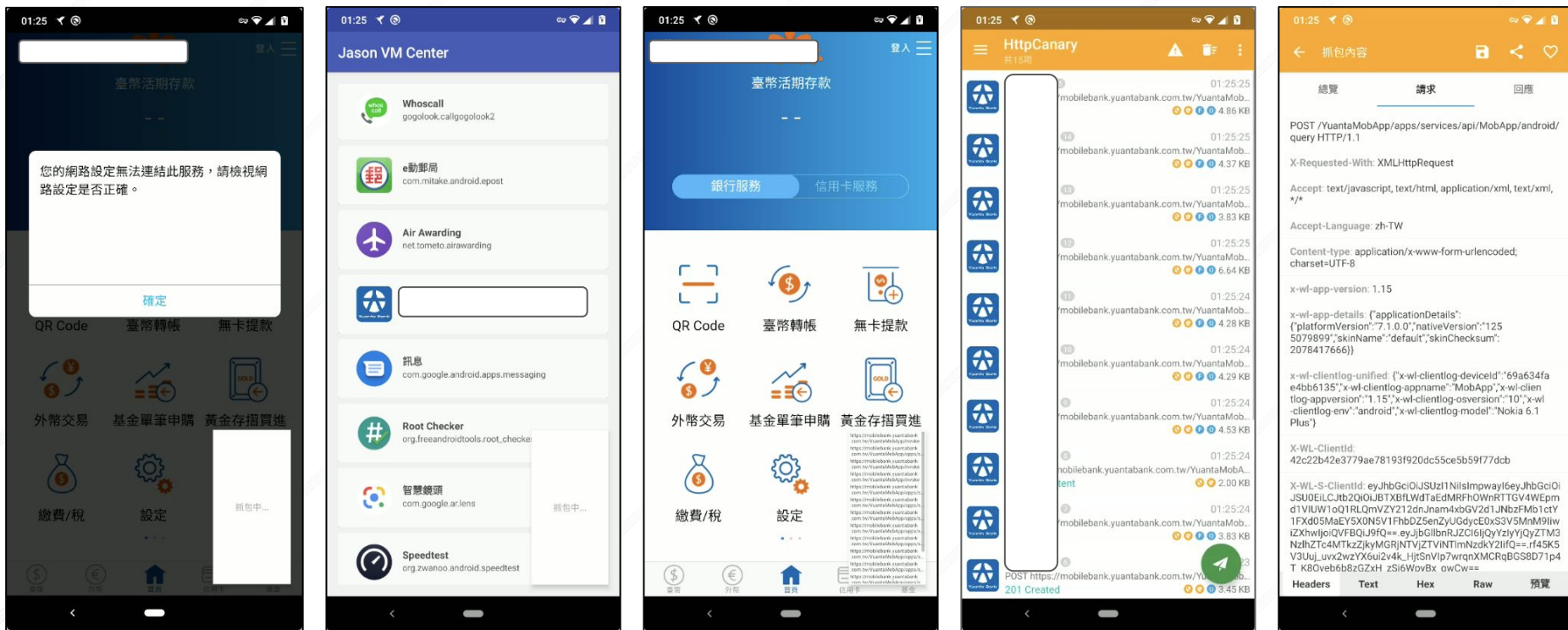
Second Dump

Something More

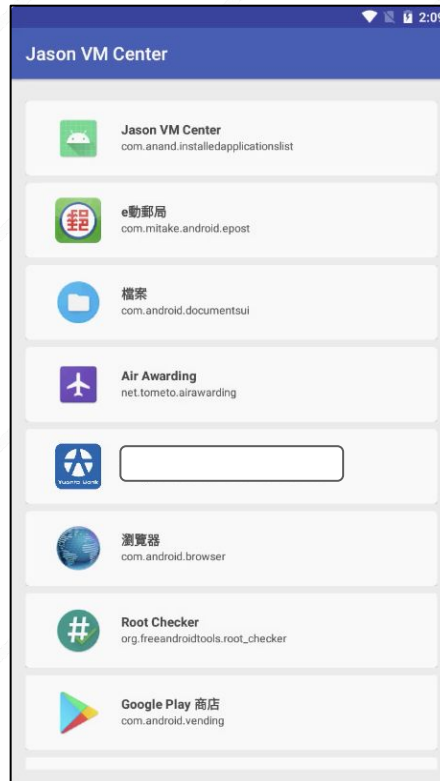
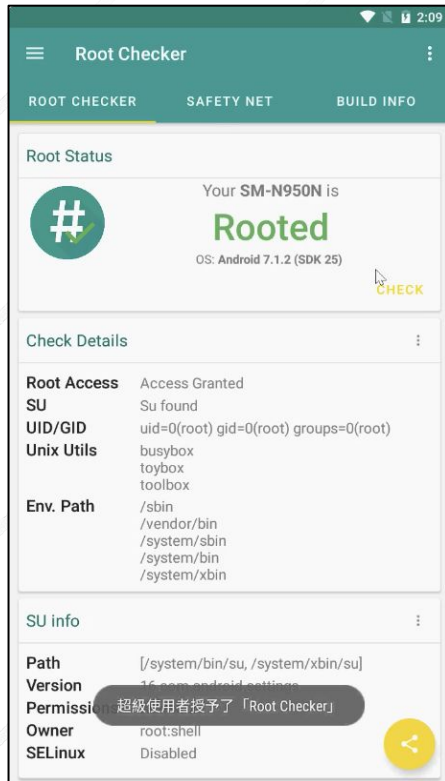
Keylogger



Bypass SSL Certificate Pinning



Bypass Root Detection



Thank you :)

王羿廷 | Jason Wang

jason.wang@fourdesire.com