Reversing Android Apps

Tools, Techniques, and Practices

Why Reverse Android Apps?

- Secret data that shouldn’t be stored in the app
  - Encryption Keys
  - “Test” account credentials
- Undocumented features or methods of authentication bypass
- Hard to trigger HTTP routes and their parameters
- Usage of HTTP for sensitive data
- “Debug” Functionality
- Anything else that might be useful when you go to hit the server :-)

Process

- Download App on Android Device
- Find App APK file path
- Pull APK at file path
- Use APKTool to extract APK to resources and bytecode
- Use Dex2Jar to convert from Smali ByteCode to Jar file
- Use JD-GUI (or Fernflower) to decompile back to Java

Acquiring APKs

- Best route is to download to an android device!
- BEST ROUTE IS TO DOWNLOAD TO AN ANDROID DEVICE!
- ...some third party sites allow downloading through a browser
  - However, who knows what they’re doing to the APK file, so don’t install it or run it!

Roundtrip from APK to Compiler (and back?)

Decompiled Java Source generally cannot be recompiled to recreate the app.

This is because of the conversion from Dalvik (smali) bytecode to Java Bytecode

Decompiled Java Source is generally only useful for figuring out what an app is doing.

However

APKTool output can be rebuilt into an app. So if you need to flip resource file flags, or edit smali bytecode by hand, APKtool output can make roundtrips from app to output and output to app.

But if you make any modifications, you will not be able to resign the app. The app will run, it’ll just be invalidly signed :-(

Script to make a production app debuggable

https://gist.github.com/nstarke/615ca3603fdded8aee47fab6f4917826

Tools used

- IntelliJ - (https://www.jetbrains.com/idea/download/)
- Android Studio - (https://developer.android.com/studio/index.html)
- ADB (provided by Android Framework - comes bundled with Android Studio)
- APKTool - Extra resources and Smali Bytecode (https://github.com/iBotPeaches/Apktool)
- Dex2Jar - Extracting Apktool to a Jar file (https://github.com/pxb1988/dex2jar)
- JD (Java Decompiler) - (https://github.com/java-decompiler/jd-gui)
- A script to automate the last three: (https://gist.github.com/nstarke/d42e138a4338708174e08923cd2a4eb8)
- Fernflower - JetBrains Java Decompiler (Comes with IntelliJ)
- Javap - Viewing Dex2Jar class files disassembled bytecode

IntelliJ (https://www.jetbrains.com/idea/download/)

- Java IDE
- Useful for navigating decompiled java code
- Comes bundled with “Fernflower” Java Decompiler

Android Studio (https://developer.android.com/studio/index.html)

- Contains a bunch of platform tools that might be useful to android developers
- Contains ADB, which we will definitely need
- Useful for navigating Decompiled Java Code (Based on IntelliJ)

ADB (Android Debug Bridge)

- Useful for executing commands on an android device
- `adb shell pm list packages -f -3` to see all third party packages
- `adb pull /data/packge/base.apk` to pull APK from android device

APKTool (https://github.com/iBotPeaches/Apktool)

- Extract APK archive
- Useful if you want to make manual modifications to app
- Output can be fed into Dex2Jar to create .class files from .smali files

Dex2Jar - (https://github.com/pxb1988/dex2jar)

- Converts .smali files to .class files
- Useful if you want to use another tool to move from bytecode to .java files
- Output can be fed into a Java Decompiler like JD or Fernflower for the trip to Java Code

JD-GUI (and JD-CMD) (https://github.com/java-decompiler/jd-gui)

Java Decompiler that can be run from the command line via JD-CMD:

https://github.com/kwart/jd-cmd

FernFlower

FernFlower is JetBrains Java Decompiler, and it produces superior results compared to JD

- `find $INTELLIJ_PATH -name “java-decompiler.jar”`
- Then cd into the folder that contains `java-decompiler.jar`
- `java -cp java-decompiler.jar org.jetbrains.java.decompiler.main.decompiler.ConsoleDecompiler $ARGS`

Javap

- Useful for disassembly .class files (Java Bytecode - output from Dex2Jar)
- `javap -c $FILE`

I wrote a thing to automate all this!

https://gist.github.com/nstarke/d42e138a4338708174e08923cd2a4eb8

Demo Time!

Slides will be available at:

https://secdsm.org

Thank you