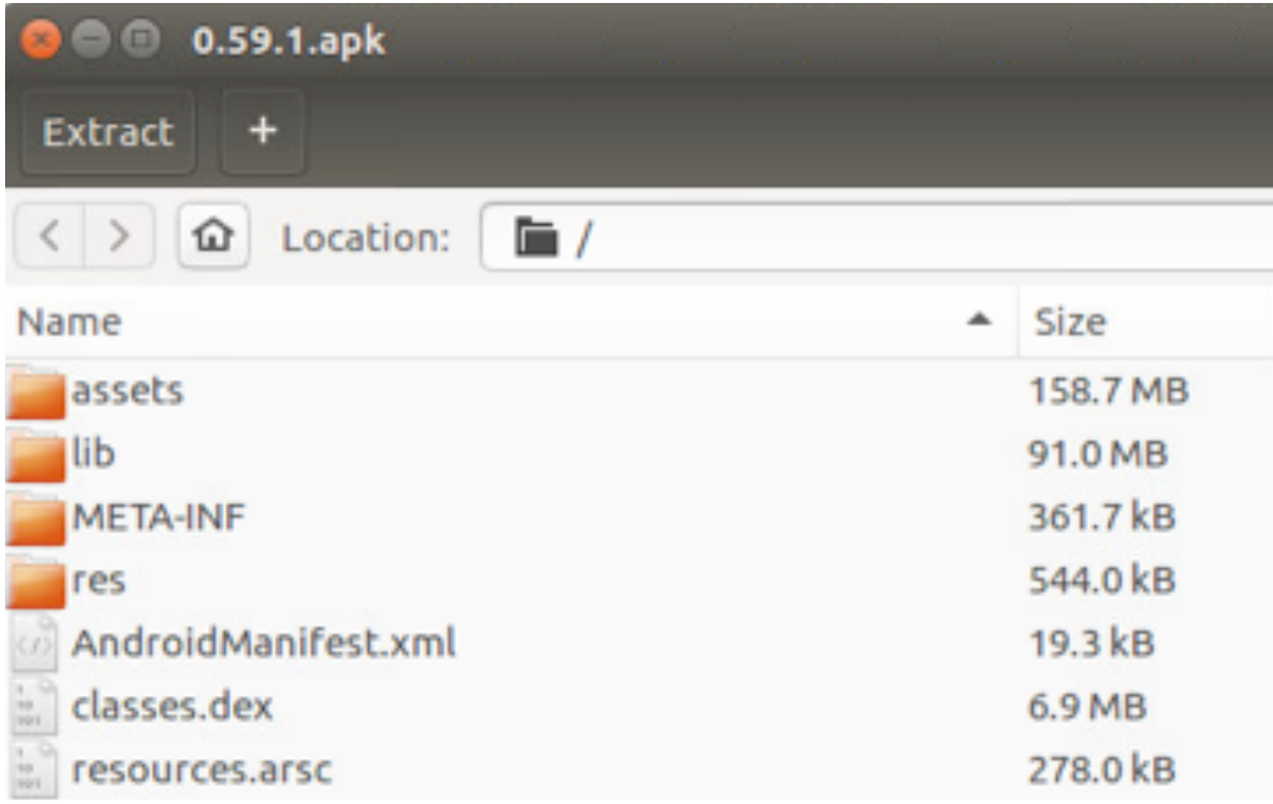
Exploring the Android Application

connor tumbleson

sourcetoad
DEVELOPMENT STUDIO

The final step: APK

- After compilation
  - APK - **A**ndroid **P**ackage **K**it
- Basically a ZIP file
  - Unpack easily
  - Readable? No



Understanding some files

- **classes.dex**
  - Assembled Java code
- **resources.arsc**
  - Assets (layouts, strings)

Android XML - AXML

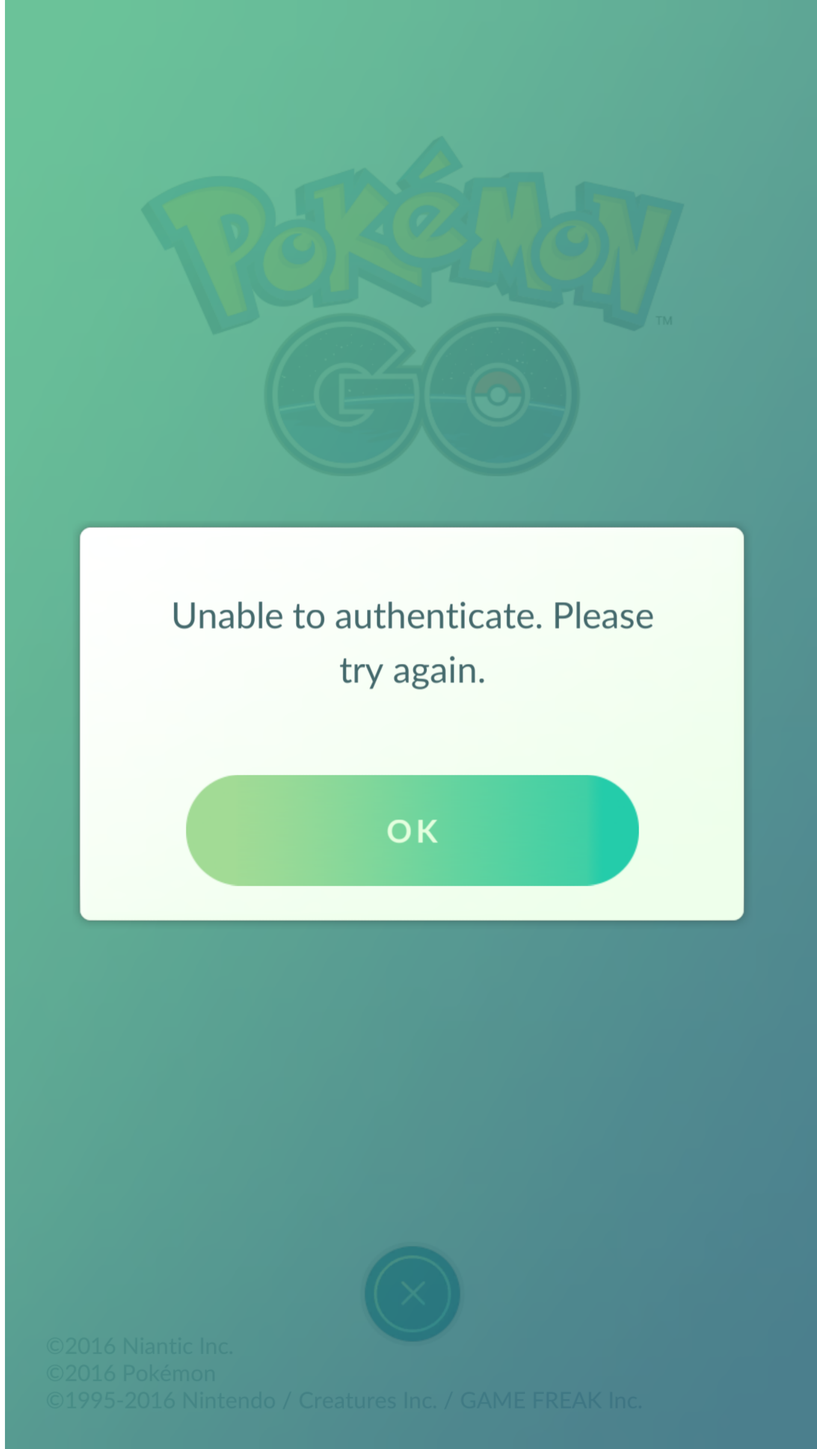
- **XML files are AXML (Android XML)**
  - Binary XML
  - Easier for machines
  - Hard to read for humans

Libraries (.so files)

- **Libraries (Shared Objects)**
  - Game Engines
  - Android NDK
  - Native Code

Example Protection: SSL Pinning

- **Uh oh**
  - No more MITM
- **We have APK**
  - Decode
  - Modify
  - Rebuild



A tool for reverse engineering Android apk files

```
$ apktool d test.apk
I: Using Apktool 2.2.2 on test.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: 1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
$ apktool b test
I: Using Apktool 2.2.2 on test
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir...
```

Apktool

A tool for reverse engineering 3rd party, closed, binary Android apps. It can decode resources to nearly original form and rebuild them after making some modifications. It also makes working with an app easier because of the project like file structure and automation of some repetitive tasks like building apk, etc.

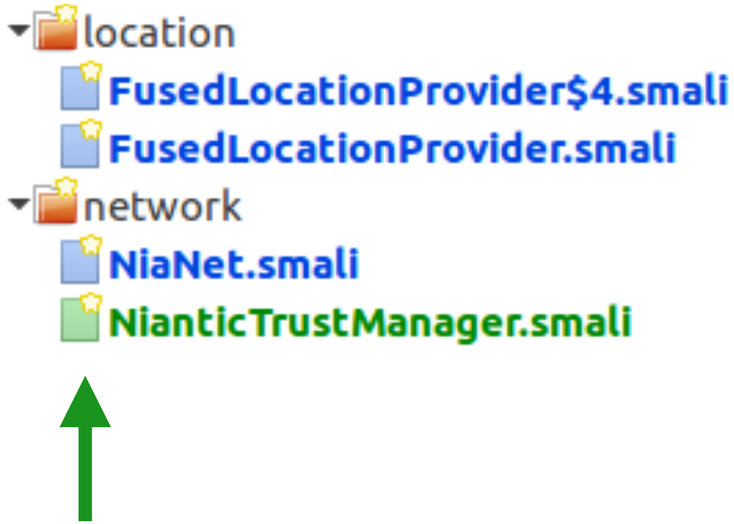
It is **NOT** intended for piracy and other non-legal uses. It could be used for localizing, adding some features or support for custom platforms, analyzing applications and much more.

Example Attack: Decoding Pokemon Go

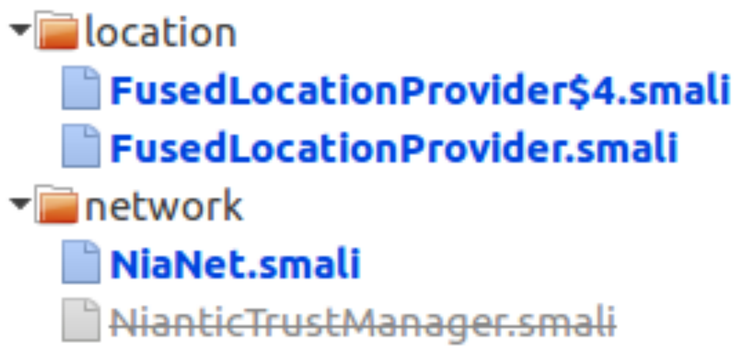
```
➜ PokemonGo apktool d pokemon_go.apk -o decoded_apktool
I: Using Apktool 2.2.1-4c93cb-SNAPSHOT on pokemon_go.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/ibotpeaches/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
➜ PokemonGo
```

Example Protection: SSL Pinning

New



Old



- NianticTrustManager.smali
  - hmmm

Example Attack: Removing SSL Pinning

```
# virtual methods
.method public checkClientTrusted([Ljava/security/cert/X509Certificate;Ljava/lang/String;)V
    .locals 2
    .param p1, "chain"    # [Ljava/security/cert/X509Certificate;
    .param p2, "authType"    # Ljava/lang/String;
    .annotation system Ldalvik/annotation/Throws;
        value = {
            Ljava/security/cert/CertificateException;
        }
    .end annotation

    .prologue
    .line 30
    iget-object v1, p0, Lcom/nianticlabs/nia/network/NianticTrustManager;->callbackLock:Ljava/lang/Object;
```

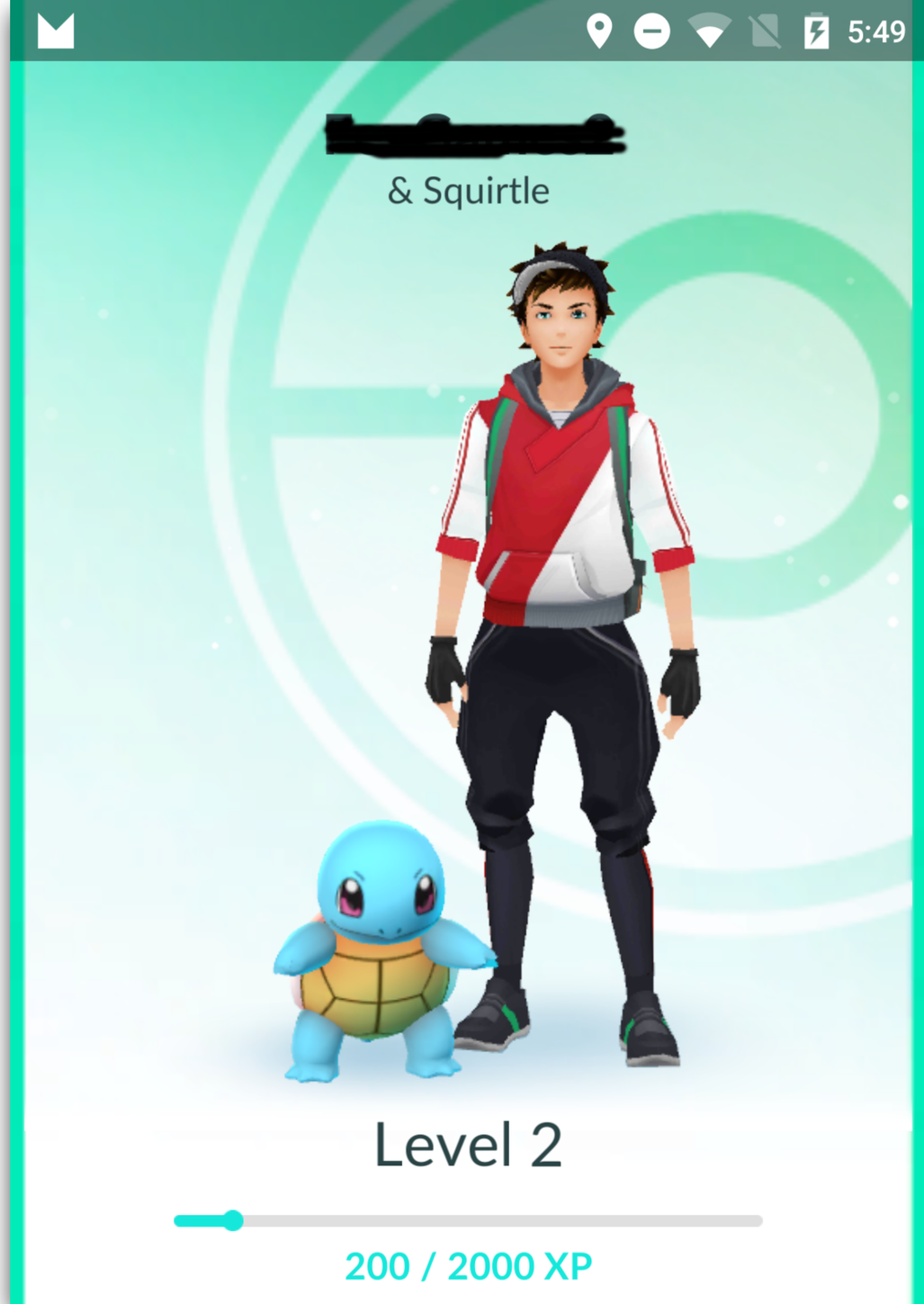
Example Attack: Removing SSL Pinning

```
# virtual methods
.method public checkClientTrusted([Ljava/security/cert/X509Certificate;Ljava/lang/String;)V
    return-void
    .locals 2
    .param p1, "chain"    # [Ljava/security/cert/X509Certificate;
    .param p2, "authType"    # Ljava/lang/String;
    .annotation system Ldalvik/annotation/Throws;
        value = {
            Ljava/security/cert/CertificateException;
        }
    .end annotation

    .prologue
    .line 30
    iget-object v1, p0, Lcom/nianticlabs/nia/network/NianticTrustManager;->callbackLock:Ljava/lang/Object;
```

Example Protection: SSL Pinning

- We are back
- Caveat: Google Auth



Context: Android 7.0 Release - Nougat

- Direct Boot
- Key Attestation
- Network Security Config
- Certificate Authority
- Signature Scheme v2
- Directory Access