Writing and Breaking Android Tamper Detection Techniques

James Stevenson

Ensure you have APKTool, Jadx, and Android Studio installed.




@_JamesStevenson

Github.com/user1342

James Stevenson (He/ Him)

JamesStevenson.me

Previous Roles:

🕵️ **SOC Analyst -** Alert Logic

📱 **Android Internals Software Engineer** - BT

⚙️ **Offensive Security Research** - F-Secure

💻 **Vulnerability Research** - Interrupt Labs

Side Projects:

✍️ **Published Author** - Android Software Internals by Apress Publishing

📚 **Self-Published Author** - Mobile Offensive Security Pocket Guide

🎓 **Part-time PhD Student** - Bristol University

🎤 **Occasional Conference** Speaker

Agenda

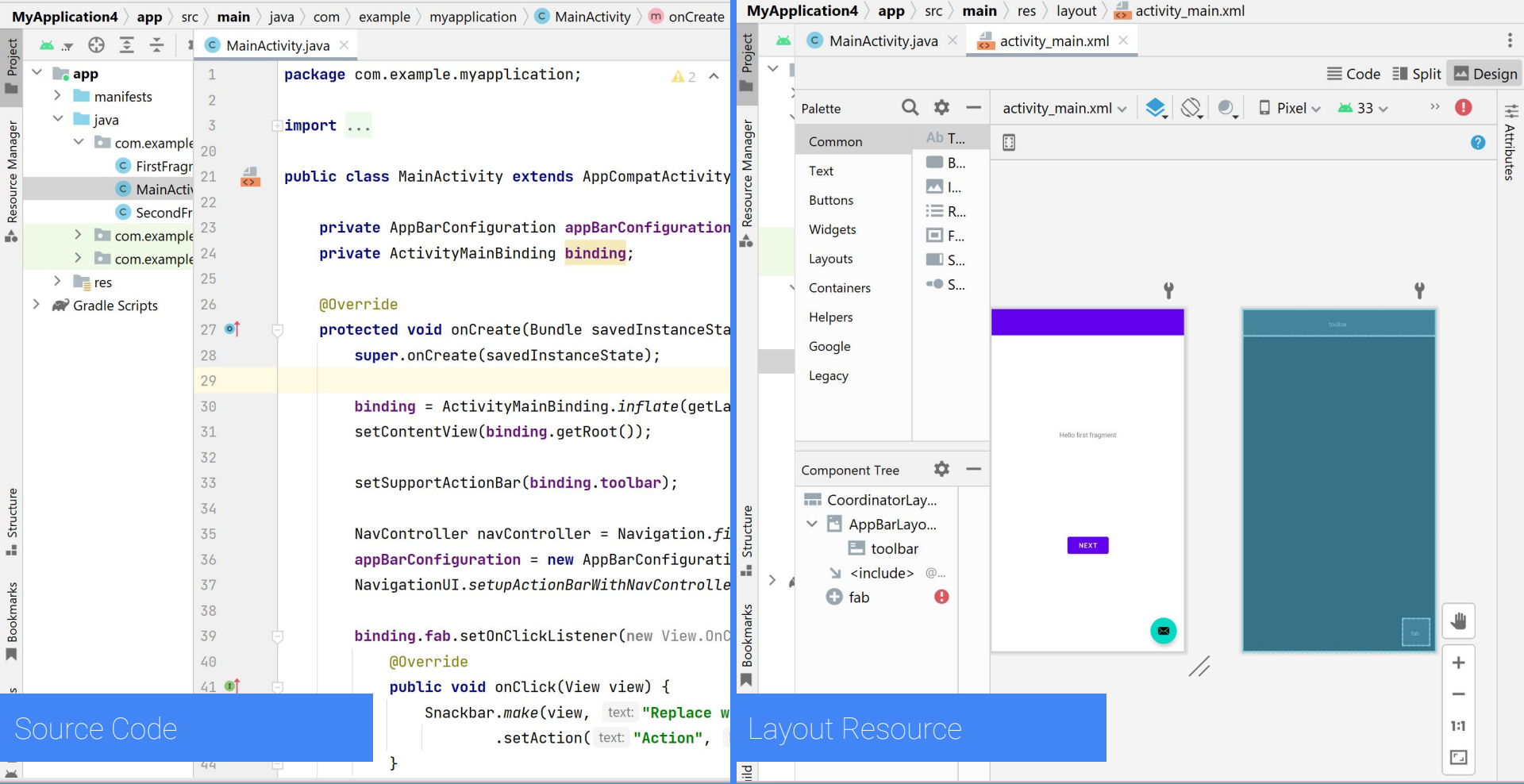
🔎 APK Static Analysis

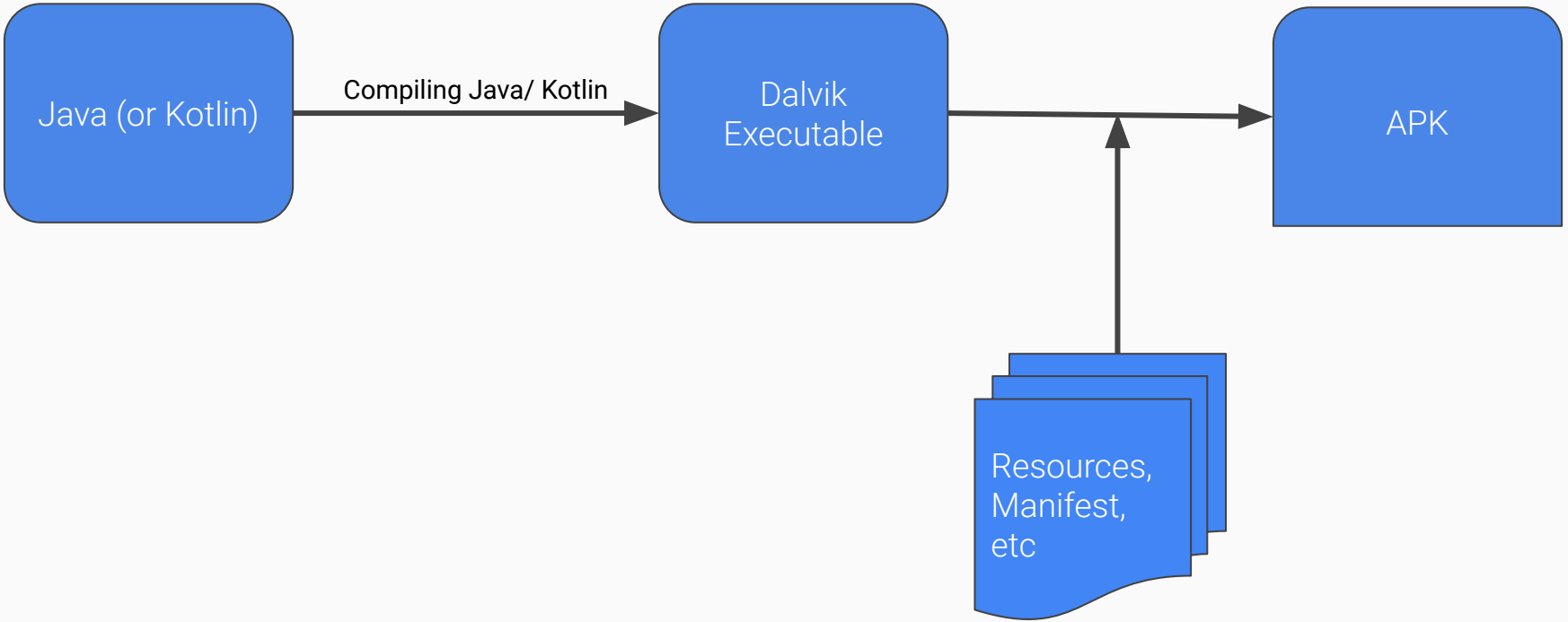
📱 Patching Android Applications

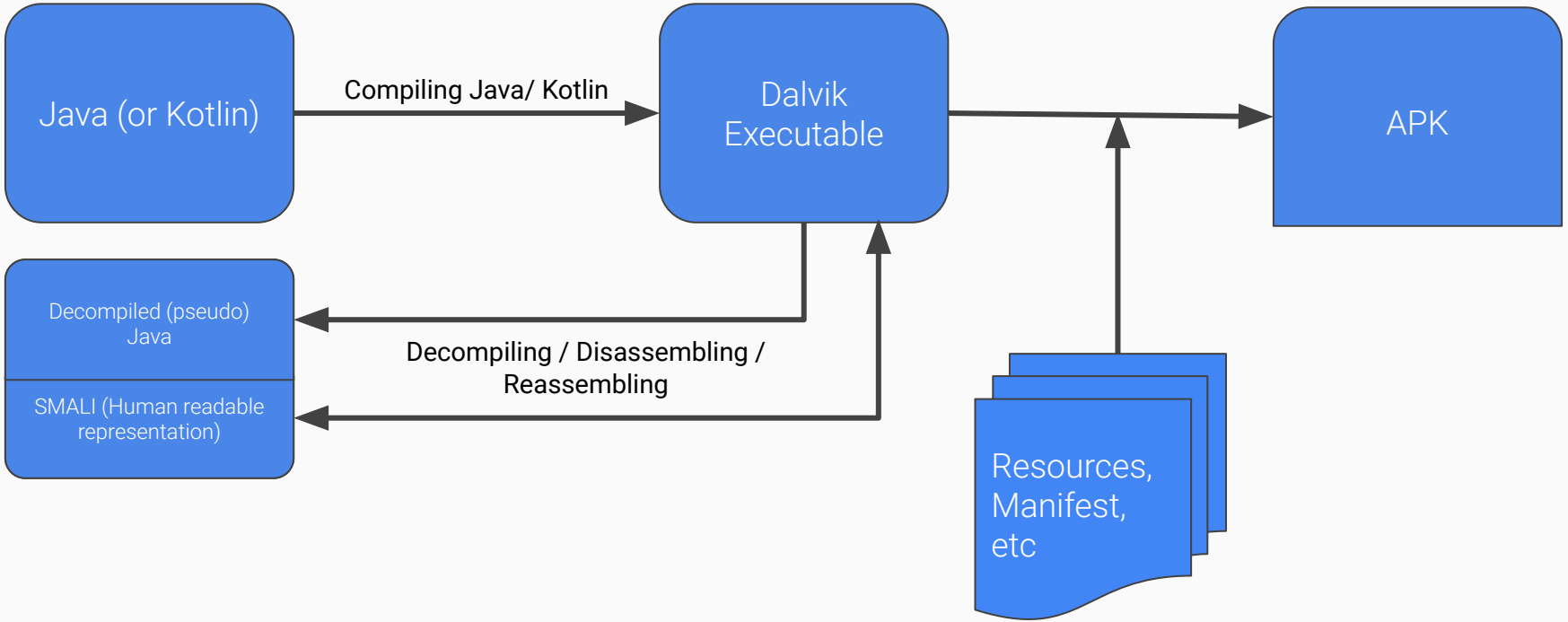
🕵️ Android Tamper Detection

Introduction to Android app RE *[Slides]*

- Building a simple Android app *[Exercise]* - 10 Minutes
- Developing a tamper detection app *[Exercise]* - 30 Minutes
- Reverse engineering eachothers apps *[Exercise]* - 30 Minutes

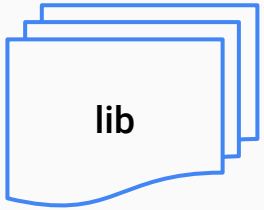

MyApplication4 app src main java com example myapplication MainActivity onCreate
MainActivity.java
app
manifests
java
com.example
FirstFragr
MainActi
SecondFr
com.example
com.example
res
Gradle Scripts
1 package com.example.myapplication;
2
3 import ...
20
21 public class MainActivity extends AppCompatActivity
22
23 private AppBarConfiguration appBarConfiguration
24 private ActivityMainBinding binding;
25
26 @Override
27 protected void onCreate(Bundle savedInstanceSta
28 super.onCreate(savedInstanceState);
29
30 binding = ActivityMainBinding.inflate(getLa
31 setContentView(binding.getRoot());
32
33 setSupportActionBar(binding.toolbar);
34
35 NavController navController = Navigation.fi
36 appBarConfiguration = new AppBarConfiguratio
37 NavigationUI.setupActionBarWithNavControlle
38
39 binding.fab.setOnClickListener(new View.OnC
40 @Override
41 public void onClick(View view) {
Snackbar.make(view, text: "Replace w
.setAction(text: "Action",
}
Source Code
MyApplication4 app src main res layout activity_main.xml
MainActivity.java
activity_main.xml
Code
Split
Design
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
activity_main.xml
Pixel
33
Component Tree
CoordinatorLay...
AppBarLayo...
toolbar
<include>
fab
Hello first fragment
NEXT
toolbar
Attributes
Layout Resource
Components Of Developing An APK


Java (or Kotlin)
Compiling Java/ Kotlin
Dalvik Executable
APK
Resources, Manifest, etc

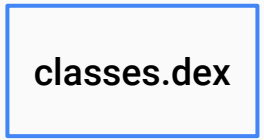

Java (or Kotlin)
Compiling Java/ Kotlin
Dalvik Executable
APK
Decompiled (pseudo) Java
SMALI (Human readable representation)
Decompiling / Disassembling / Reassembling
Resources, Manifest, etc

App Components

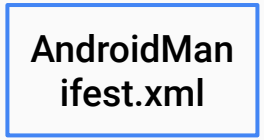
Arbitrary storage



Native libraries



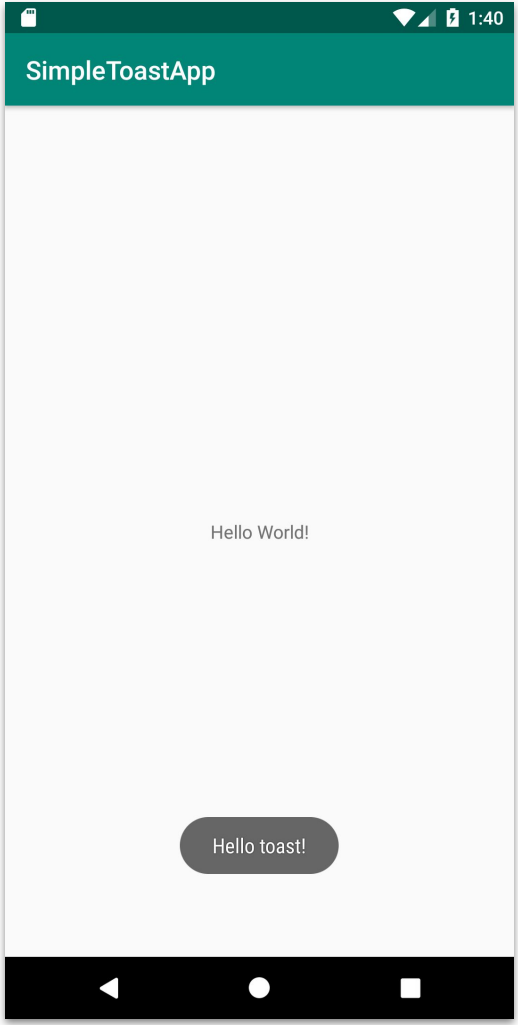
Compiled dalvik executable



Application config

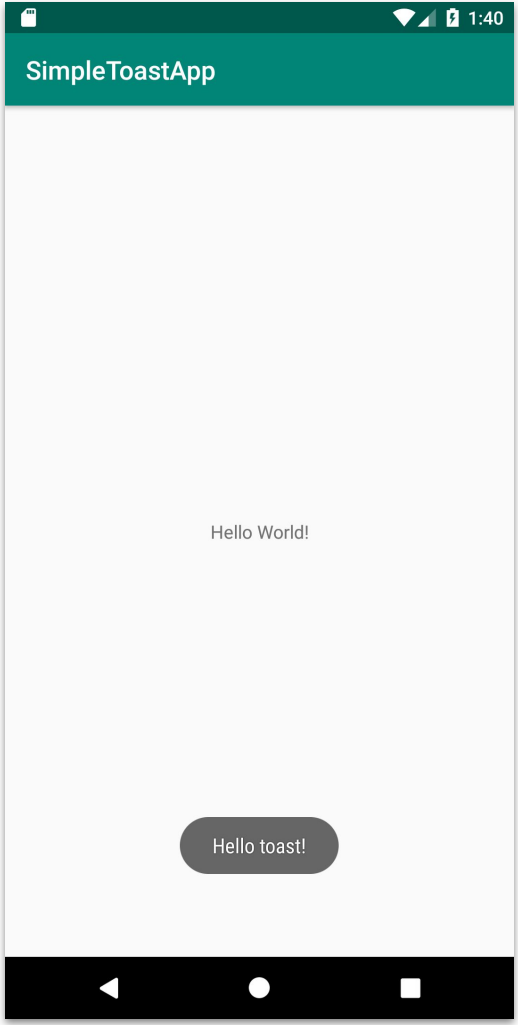
...

```
val text = "Hello toast!"
val duration = Toast.LENGTH_LONG

val toast =
Toast.makeText(applicationContext, text,
duration)
toast.show()
```

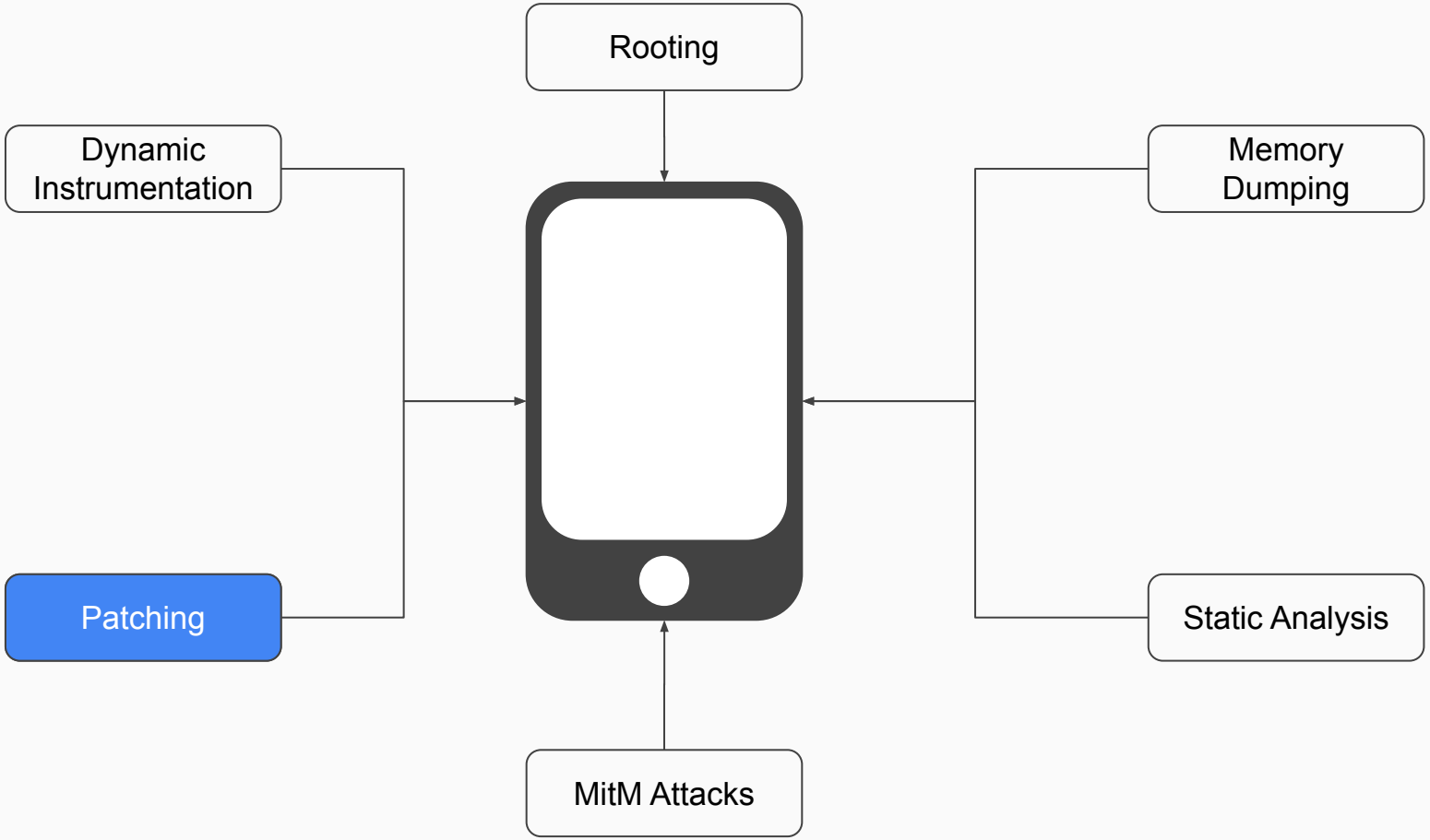


Toast in Kotlin

```
.line 13
const-string v0, "Hello Toast!"

.line 14
.local v0, "text":Ljava/lang/String;
const/4 v1, 0x1

.line 16
.local v1, "duration":I
invoke-virtual {p0},
Lcom/example/simpletoastapp/MainActivity;->getApplicationContext()L
android/content/Context;

move-result-object v2

move-object v3, v0

check-cast v3, Ljava/lang/CharSequence;

invoke-static {v2, v3, v1},
Landroid/widget/Toast;->makeText(Landroid/content/Context;Ljava/lan
g/CharSequence;I)Landroid/widget/Toast;

move-result-object v2

.line 17
.local v2, "toast":Landroid/widget/Toast;
invoke-virtual {v2}, Landroid/widget/Toast;->show()V
```



Toast in Smali (Dalvik Bytecode)


Rooting
Dynamic Instrumentation
Memory Dumping
Patching
Static Analysis
MitM Attacks

Normal Candy Crush

Patched Candy Crush

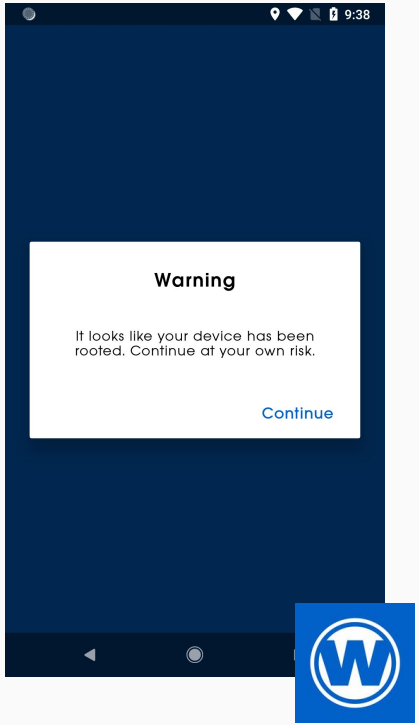
Setup A Simple ‘*Hello World*’ Android App

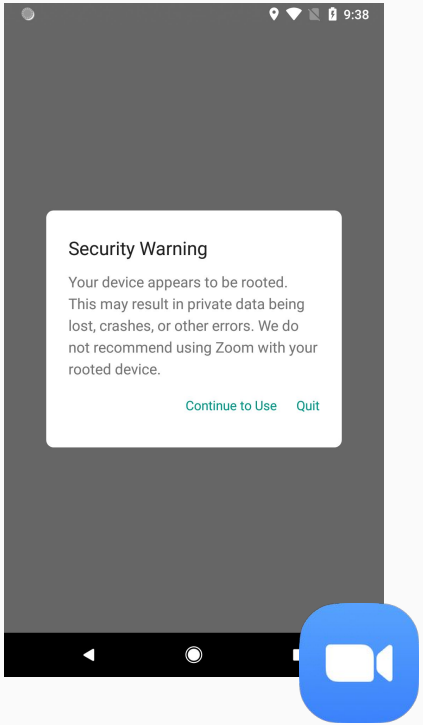
Breakout One

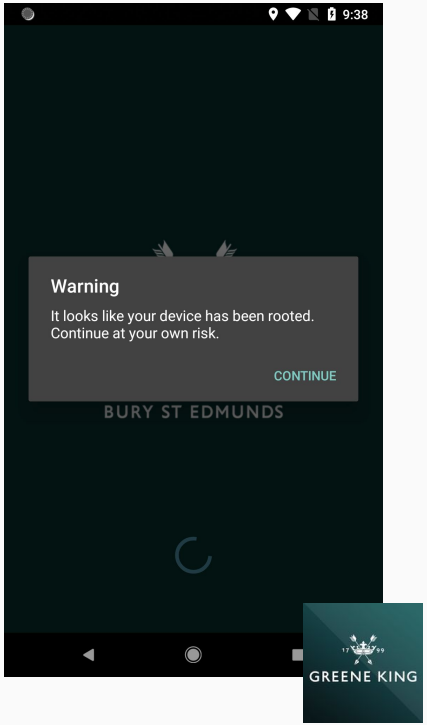
Ensure you have APKTool and Android Studio installed.

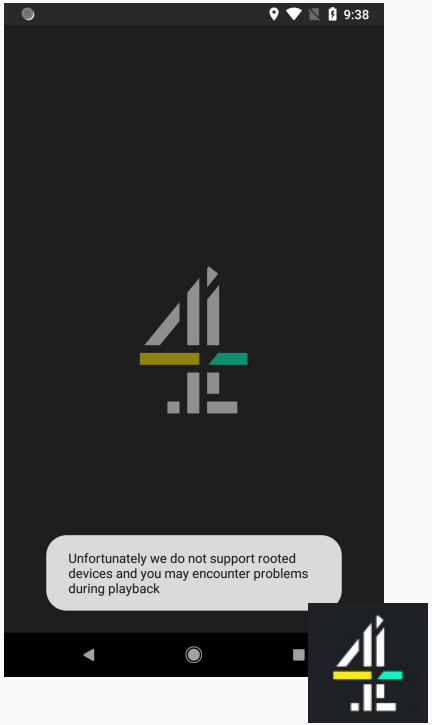
Compile your APK in Android Studio and run it on an emulator.

Then **reverse your app to SMALI.**









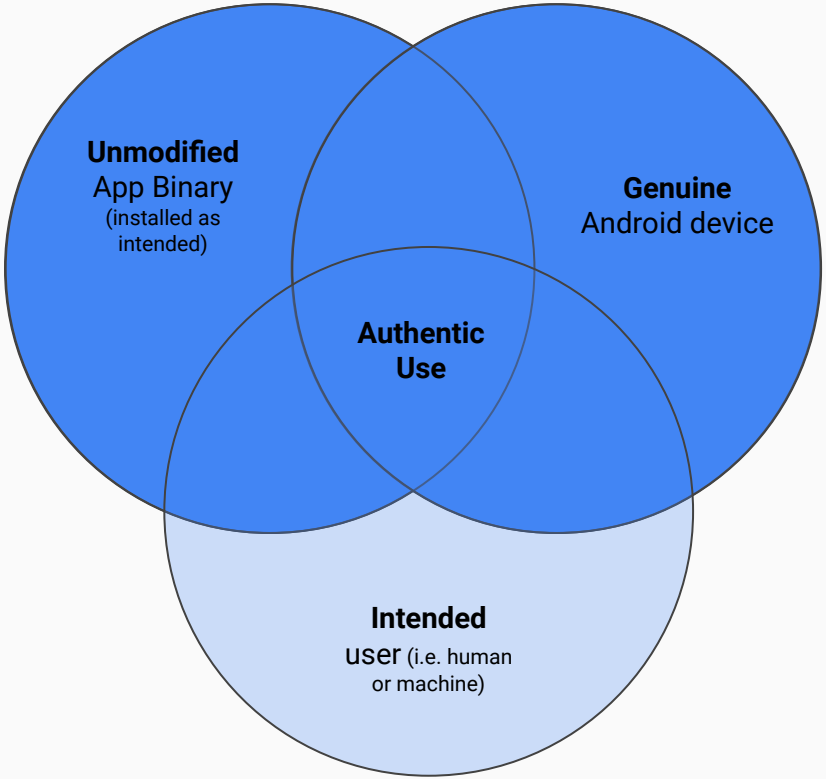
"**Rooting** is the process of ascertaining privileged control over various subsystems on an Android device." - *Some person on Wikipedia*

Normal Device

Moto G6 API level 28

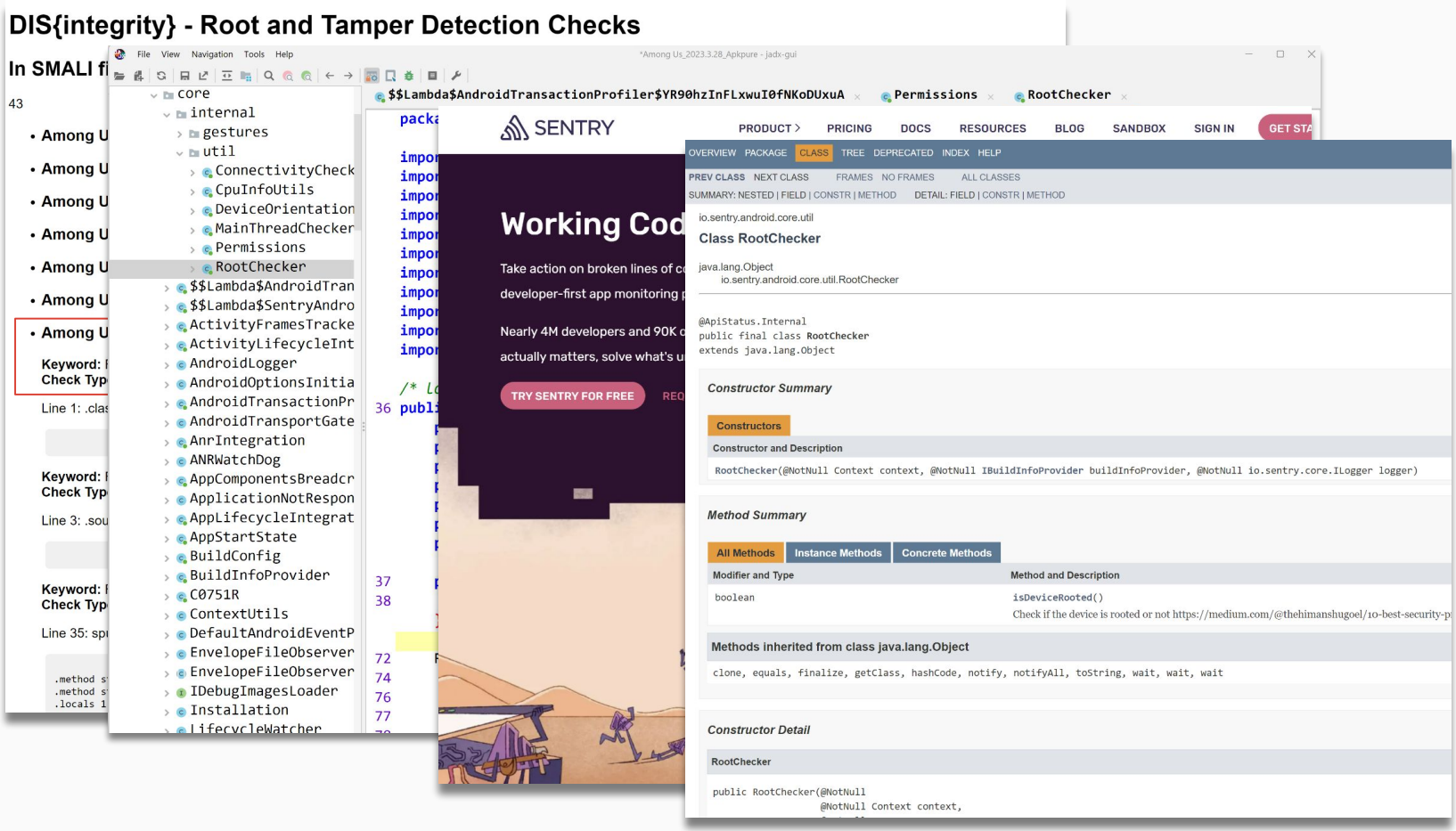
Rooted Device

Pixel 2 API level 27



What is ‘attestation’ looking to solve?

DIS{integrity} - Root and Tamper Detection Checks



Among Us - Android App Root Detection

Among Us - Android App Root Detection

io.sentry.android.core.internal.util.RootChecker()

checkTestKeys()

```
String buildTags = this.buildInfoProvider.getBuildTags();

return buildTags != null && buildTags.contains("test-keys");
```

checkRootFiles()

"/system/app/**Superuser.apk**","/sbin/**su**","/system/bin/su","/system/xbin/su","/data/local/xbin/su","/data/local/bin/su","/system/sd/xbin/su","/system/bin/failsafe/su","/data/local/su","/su/bin/su","/su/bin","/system/xbin/**daemonsu**"...

checkSUExist()

```
Process exec = this.runtime.exec(new String[] {
    "/system/xbin/which",
    "su"
  });
```

checkRootPackages()

"com.devadvance.rootcloak","com.devadvance.rootcloakplus","com.koushikdutta.superuser","com.thirdparty.superuser","eu.chainfire.supersu","com.noshufou.android.su" …

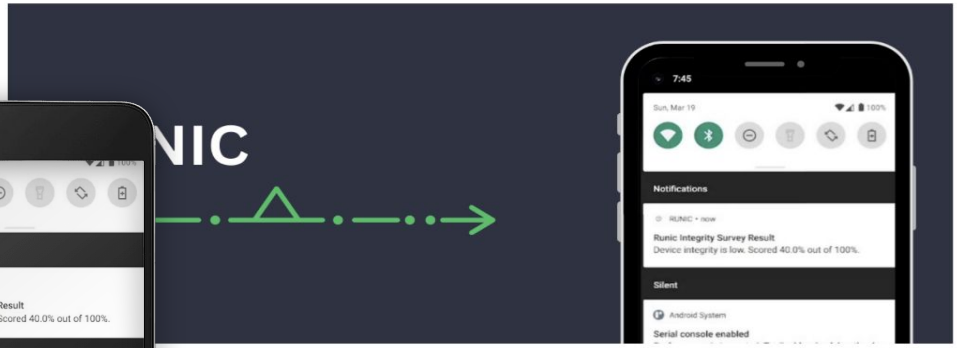
Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

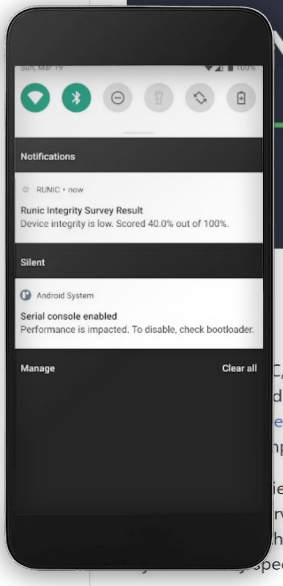
user1342 / RUNIC Public

Sponsor Pin Unwatch 3 Fork 0 Starred 2

Code Issues Pull requests Actions Security Insights Settings

README.md





RUNIC: Android tamper detection demo

contributors 1 last commit may

C, an Android tamper detection demo! RUNIC is designed to serve as a parallel and introductory
ding more complex tamper detection and integrity systems such as Google Play SafetyNet and
ect. By exploring the inner workings of RUNIC, you can gain insights into the fundamental concepts
nployed in these advanced solutions.

ient/server model where the application collects device variables and sends them to a server f
rver then provides a percentage score indicating the likelihood of the device being tampere th.
he successor to the previous tool, Tamper, and is currently in a minimal viable product (MV ate.
pecific improvements, issues, or feature requests, please use the issues tab.

Getting Started

About

RUNIC tamper detection demo - designed to serve as a parallel for understanding more complex tamper detection and integrity systems such as Google Play SafetyNet and Huawei Safety Detect.

android integrity safetynet root-detection tamper-detection safety-detect

Readme
GPL-3.0 license
2 stars
3 watching
0 forks

Sponsor this project

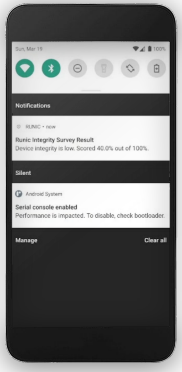
ko-fi.com/jamesstevenson

Languages

Java 100.0%

Github.com/user1342/RUNIC

RUNIC: Android tamper detection demo

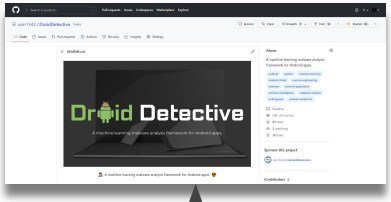


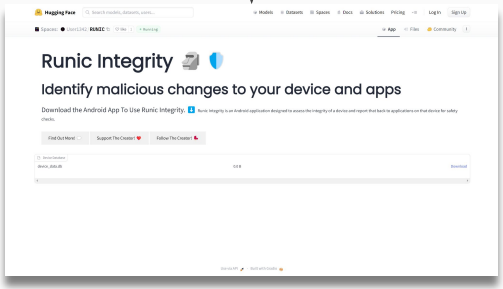

```
adb_enabled': 'true',
 'boot_state': '',
 'brand': 'Android',
 'debug_enabled': 'true',
 'emulator': 'true',
 'fingerprint_status': '2',
 'id': '935efe59519bcd25',
 'installer': 'false',
 'lock_screen_timeout': '0',
 'lock_screen_type': '0',
 'market_apps_enabled': 'true',
 'model': 'Android SDK built for x86',
 'notification_visibility': '0',
 'oem': '-1',
 'patch_level': '2020-11-05',
 'primary_certificate': '-793121132',
 'storage_encryption_status': '5',
 'unlocked': '-1',
 'verify': ''
```

X% Integrity Confidence

```
{'com.android.providers.contacts': ['android.permission.BIND_DIRECTORY_SEARCH',
                                    'android.permission.GET_ACCOUNTS',
                                    'android.permission.GET_ACCOUNTS_PRIVILEGED',
                                    'android.permission.INTERACT_ACROSS_USERS',
                                    'android.permission.MANAGE_USERS',
                                    'android.permission.PROCESS_PHONE_ACCOUNT_REGISTRATION',
                                    'android.permission.READ_CONTACTS',
                                    'android.permission.READ_PHONE_STATE',
                                    'android.permission.READ_PRIVILEGED_PHONE_STATE',
                                    'android.permission.READ_SYNC_SETTINGS',
                                    'android.permission.SEND_CALL_LOG_CHANGE',
```

github.com/user1342/DroidDetective





huggingface.co/spaces/User1342/RUNIC

RUNIC: Android tamper detection demo

Application Side

- Android Secure ID
- Application installer ID
- Boot state
- Verify mode
- Security patch level
- OEM unlock status
- Product brand
- Product model
- OEM unlock supported
- Debuggable status
- Application signature
- Emulator status
- Fingerprint status
- Storage encryption status
- Non-market apps enabled
- ADB enabled
- Lock Screen timeout
- Lock screen type
- Notification visibility
- Application permissions of installed apps

Server Side

- Debugging status
- Emulator detection
- Installation source (i.e. Google Play store)
- Boot state and verification status
- Previous log of the device and matching application certificates
- Comparison of variables between previous sightings of the application/ device
- Permission malware ML model

Create a Tamper Detection Android App

Breakout Two

Develop and build an Android application that implements a form of tamper detection - it should restrict application functionality when triggered.

Reverse Engineer and Patch an Application

Breakout Three

Swap applications with a peer and patch that application to mitigate their check.

Where To Go Next

🌐 www.JamesStevenson.me

🤖 user1342/Awesome-Android-Reverse-Engineering