

Identifying Rogue Android Devices

The World Of Android Attestation



Talking Points

Exploring how application and device authors **detect risk prone states**, and how **red teamers can get around these mechanisms**.

- 📱 Understanding tamper detection and examples in the wild:
 - 🛡️ Fundamentals of root detection (RUNIC)
 - 🎮 Gaming Anti-Cheat and root detection
 - 🔎 Google SafetyNet and Huawei SafetyDetect



Previous Roles:

👤 **SOC Analyst** - Alert Logic

📱 **Android Internals Software Engineer** - BT

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

💻 **Vulnerability Research** - Interrupt Labs

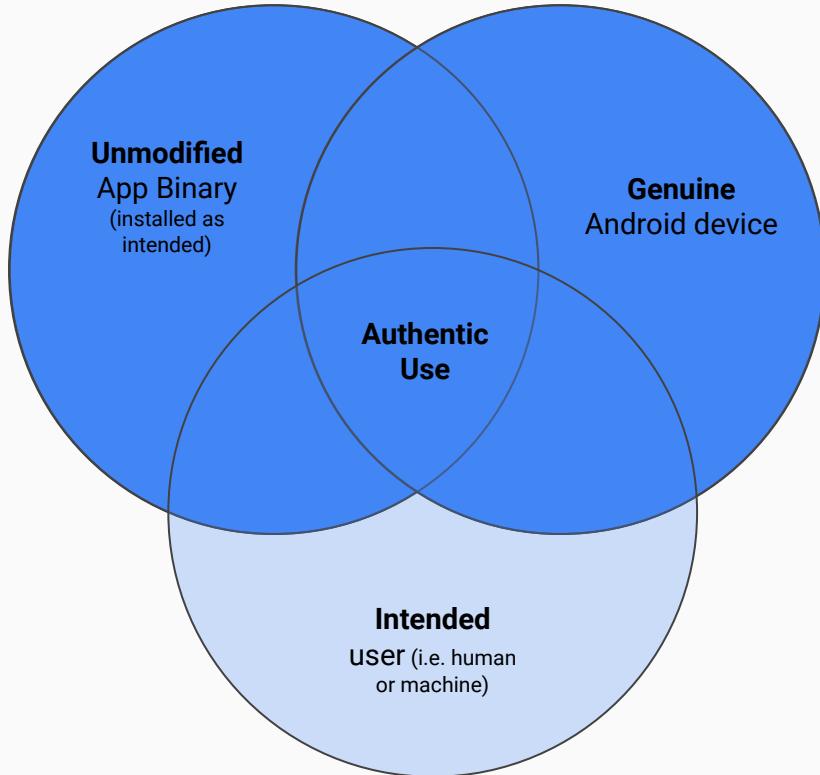
Side Projects:

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

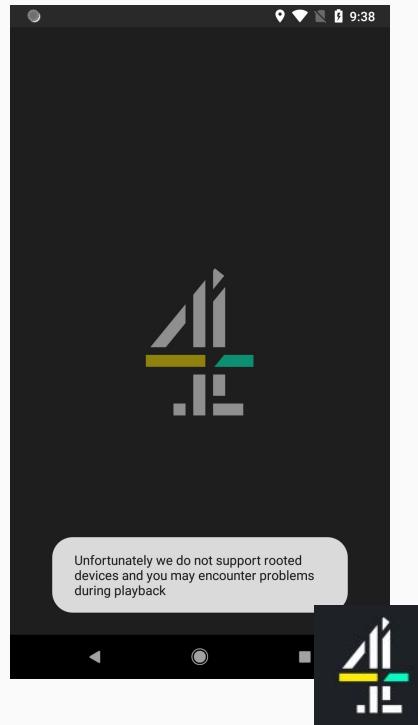
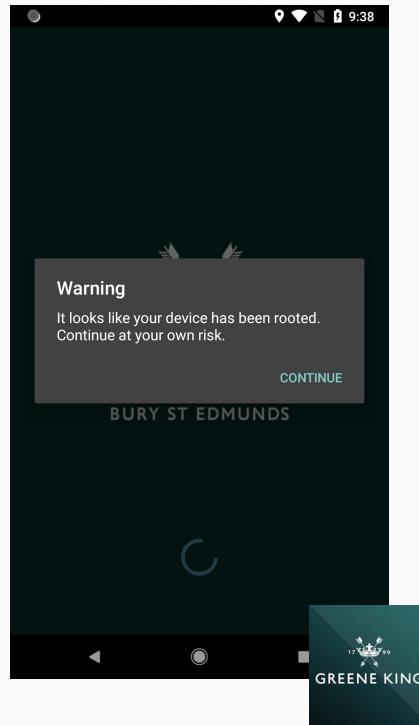
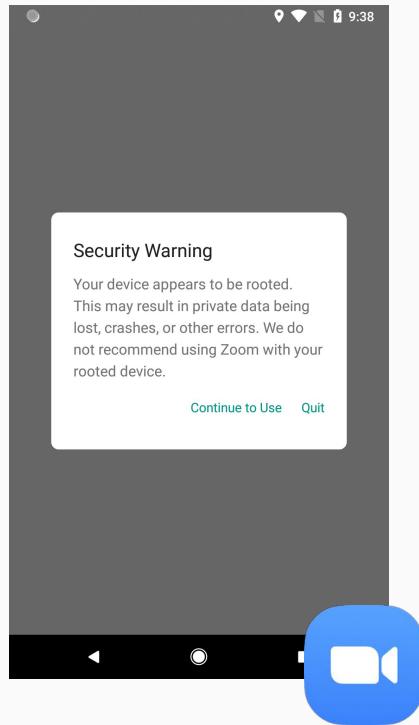
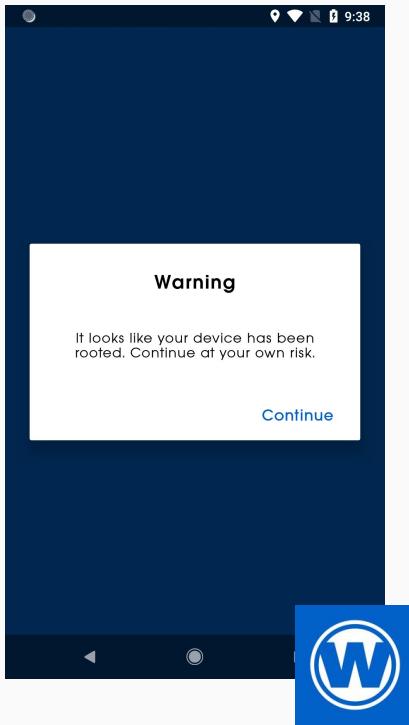
📚 **Self-Published Author** - Mobile Offensive Security Pocket Guide

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

✍️ **Occasional Conference Speaker**



What is 'attestation' looking to solve?



["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

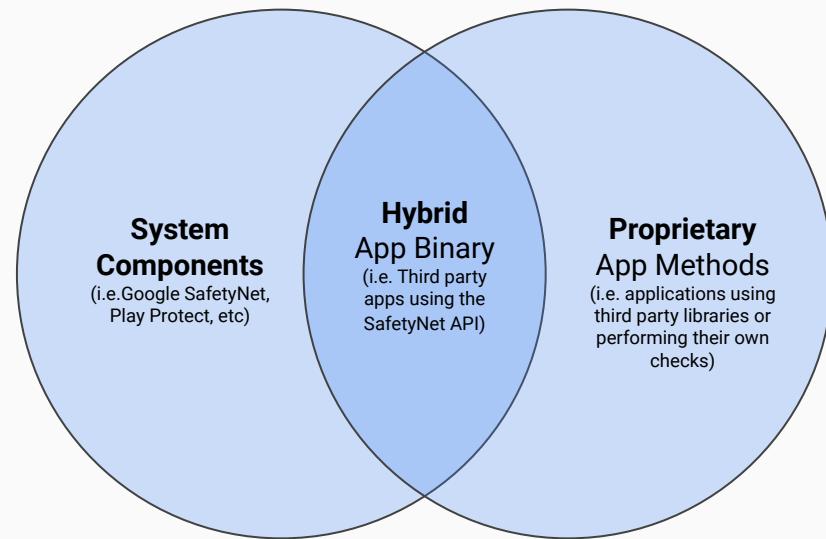


Rooted Device

Pixel 2 API level 27

Types of Integrity Checking

System and Proprietary Attestation Checks





Search or jump to...

Pull requests Issues Codespaces Marketplace Explore

Bell icon + More

user1342 / RUNCIC Public

Sponsor Pin Unwatch 3 Fork 0 Starred 2

Code Issues Pull requests Actions Security Insights Settings

README.md

RUNCIC

Notifications

RUNCIC • now
Runic Integrity Survey Result
Device integrity is low. Scored 40.0% out of 100.

Silent

Android System
Serial console enabled
Performance is impacted. To disable, check bootloader.

Manage Clear all

RUNCIC: Android tamper detection demo

contributors 1 last commit may

RUNCIC is an Android tamper detection demo! RUNCIC is designed to serve as a parallel and introductory tool for understanding more complex tamper detection and integrity systems such as Google Play SafetyNet and Huawei Safety Detect. By exploring the inner workings of RUNCIC, you can gain insights into the fundamental concepts employed in these advanced solutions.

RUNCIC follows a client/server model where the application collects device variables and sends them to a server for analysis. The server then provides a percentage score indicating the likelihood of the device being tampered with. RUNCIC is the successor to the previous tool, Tamper, and is currently in a minimal viable product (MVP) state. If you have specific improvements, issues, or feature requests, please use the issues tab.

Getting Started

RUNCIC: Android tamper detection demo

About

RUNCIC 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

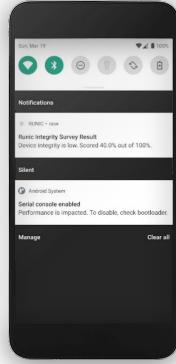
0 forks

Sponsor this project

ko-fi.com/jamesstevenson

Languages

Github.com/user1342/RUNCIC
Java 100.0%



```
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

The top part shows the Droid Detective application interface with a central dashboard and a sidebar. The bottom part shows the Runic Integrity website, which includes a download section for the Android app and a GitHub page for the project.

huggingface.co/spaces/User1342/RUNIC

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

DIS{integrity} - Root and Tamper Detection Checks

In SMALLI fi

```
43
• Among Us
Line 1: .class
Keyword: I
Check Type:
Line 3: .so
Keyword: I
Check Type:
Line 35: sp
.Keyword: I
Check Type:
Line 72: F
.Keyword: I
Check Type:
    .method s
    .method s
    .locals 1
```

*Among_Us_2023.3.28_Apkpure - jadx-gui

File View Navigation Tools Help

core
internal
 gestures
 util
 ConnectivityCheck
 CpuInfoUtils
 DeviceOrientation
 MainThreadChecker
 Permissions
 RootChecker
 \$\$Lambda\$AndroidTransactionProfiler\$YR90hzInFLxwuI0fNkoDuUxA
 Permissions
 RootChecker

product> pricing docs resources blog sandbox sign in get start

Working Code

Take action on broken lines of code in your app. Sentry's developer-first app monitoring platform makes it easy to identify what's actually matters, solve what's up.

Nearly 4M developers and 90K companies use Sentry to monitor their apps. If you're not using Sentry, you're missing out.

TRY SENTRY FOR FREE

OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

io.sentry.android.core.util

Class RootChecker

java.lang.Object
io.sentry.android.core.util.RootChecker

@ApiStatus.Internal
public final class RootChecker
extends java.lang.Object

Constructor Summary

Constructors

Constructor and Description

RootChecker(@NotNull Context context, @NotNull IBuildInfoProvider buildInfoProvider, @NotNull io.sentry.core.ILogger logger)

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type Method and Description

boolean isDeviceRooted()
Check if the device is rooted or not https://medium.com/@thehimanshuoel/10-best-security-p

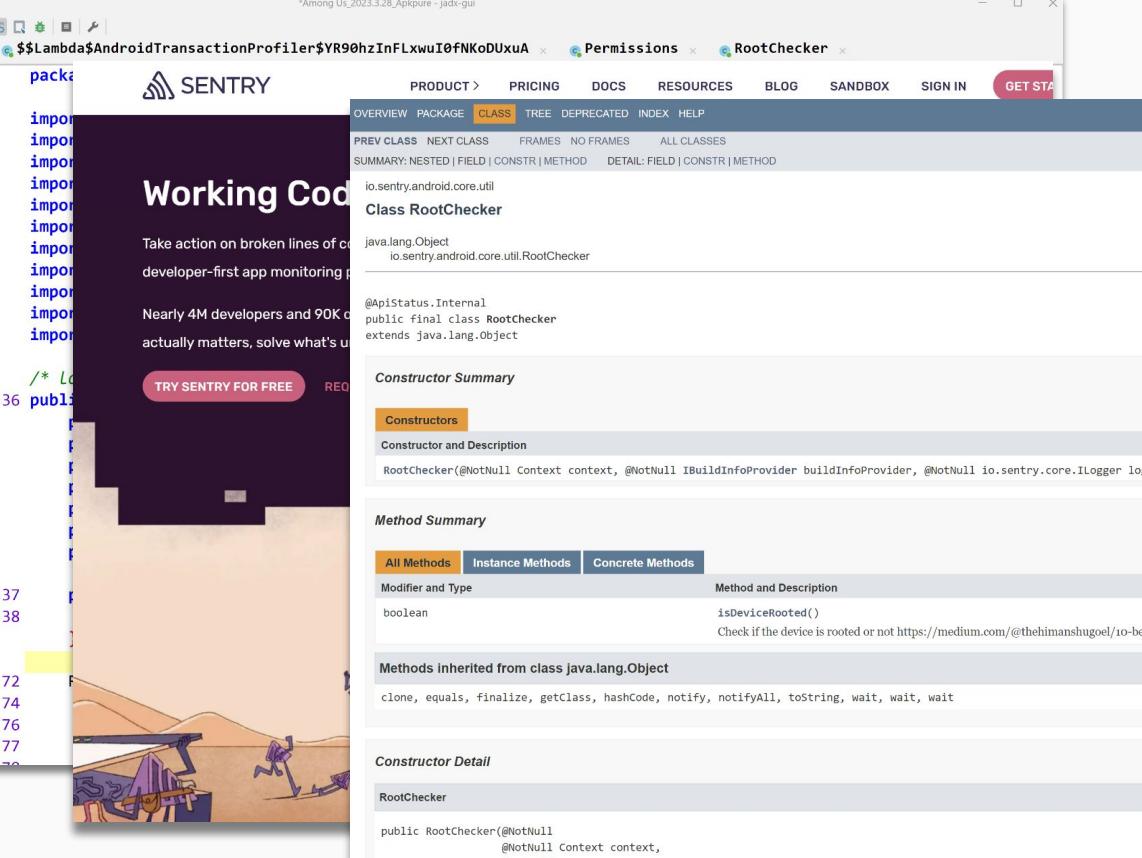
Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RootChecker

public RootChecker(@NotNull
 @NotNull Context context,
 @NotNull IBuildInfoProvider buildInfoProvider,
 @NotNull io.sentry.core.ILogger logger)



Among Us - Android App Root Detection

*io.sentry.android.core.internal.util.Root
Checker()*

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"
});
```

checkRootPacka
ges()

```
"com.devadvance.rootcloak","com.devadvance.rootcloakplus","com.koushikdutta.superuser","com.thirdparty.superuser","eu.chainfire.superusu","com.noshufou.android.su" ...
```

developers Platform Android Studio Google Play Jetpack Kotlin Docs Games Search English Android Studio Sign in

DOCUMENTATION

Overview Guides UI Guide Reference Samples Design & Quality

Mitigate security risks in your app
Security with data
Security with HTTPS and SSL
Network security configuration
Updating your security provider to protect against SSL exploits
Protecting against security threats with SafetyNet
About SafetyNet
SafetyNet Attestation API
Discontinuing SafetyNet Attestation
Play Integrity API
SafetyNet Safe Browsing API
SafetyNet reCAPTCHA API
SafetyNet Verify Apps API
Cryptography
Android Keystore System
Verifying hardware-backed key pairs with key attestation

On this page
Before you begin
Configure your app

DEVELOPERS Products Solutions Events Programs Community Training More

Search Materials Console Sign in Sign up

Android Developers > Docs > Guides

Protect against SafetyNet

SafetyNet provides a set of services and APIs to detect tampering, bad URLs, potentially harmful code, and more.

Before you begin

To prepare your app, first make sure that:

- A `minSdkVersion` of 19 or higher
- A `compileSdkVersion` of 28 or higher

Then complete the steps in the following sections:

Configure your app

In your `build.gradle` file, include the following dependency:

```
dependencyResolutionManagement {
```

Safety Detect

Introducing SysIntegrity, AppsCheck, URLCheck, and UserDetect functions. Guard against security threats to your app in today's vulnerable world.

Supported on [Android](#)

[View documents](#) [Try our demo](#)



Watch now

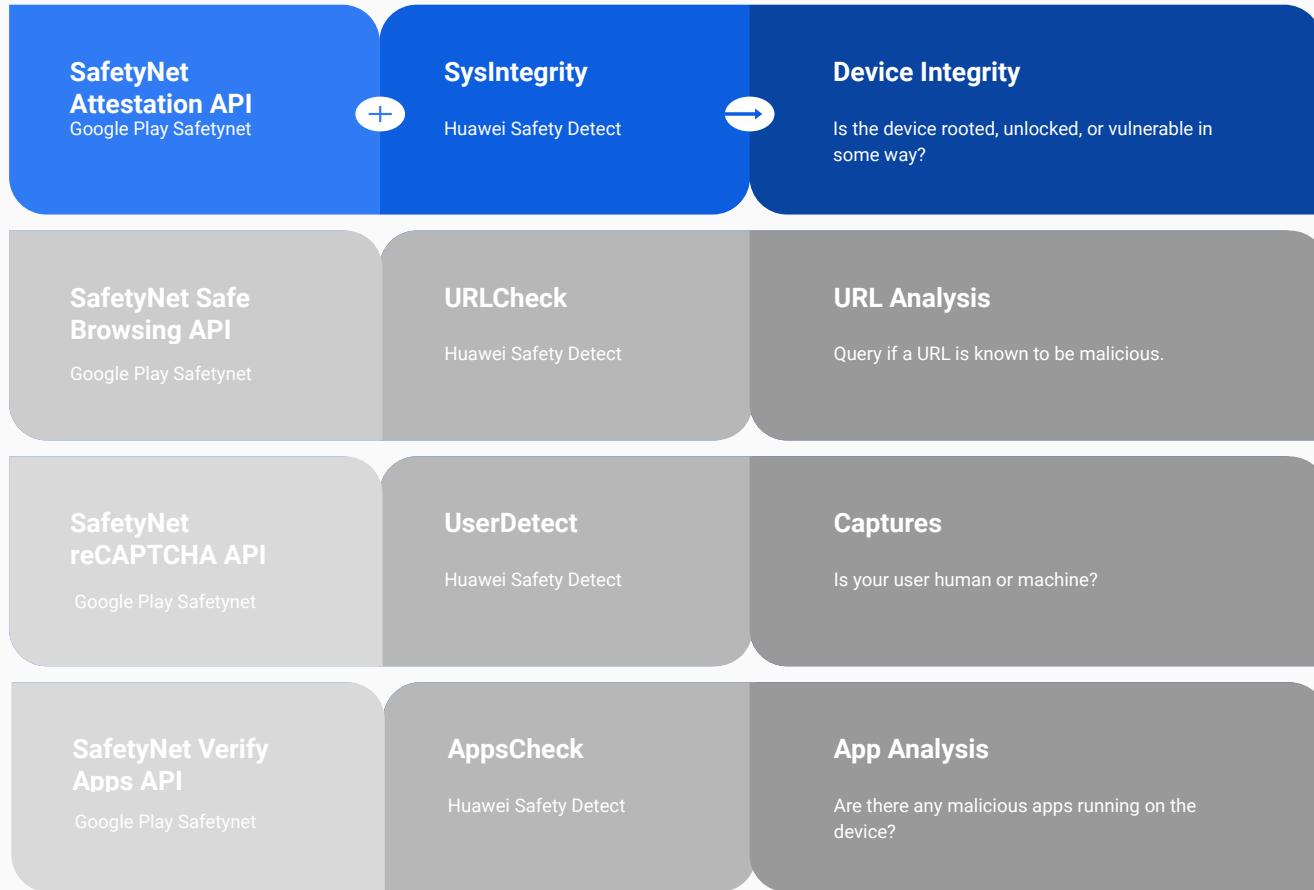
Functions Videos Advantages Success Stories Resources

Google Play Services SafetyNet
(replaced by Play Integrity API in Jan 2025)

Functions

Huawei AppGallery Safety Detect

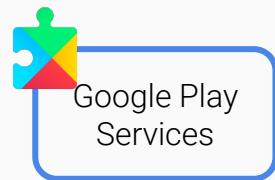
Main contenders in tamper detection



SNet Attestation

The SafetyNet Attestation API provides services for determining whether a device running an app satisfies Android security tests.

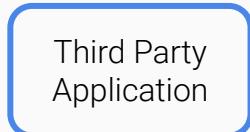
On Device



Downloads
and runs



Device variables are
aggregated and sent



Off Device



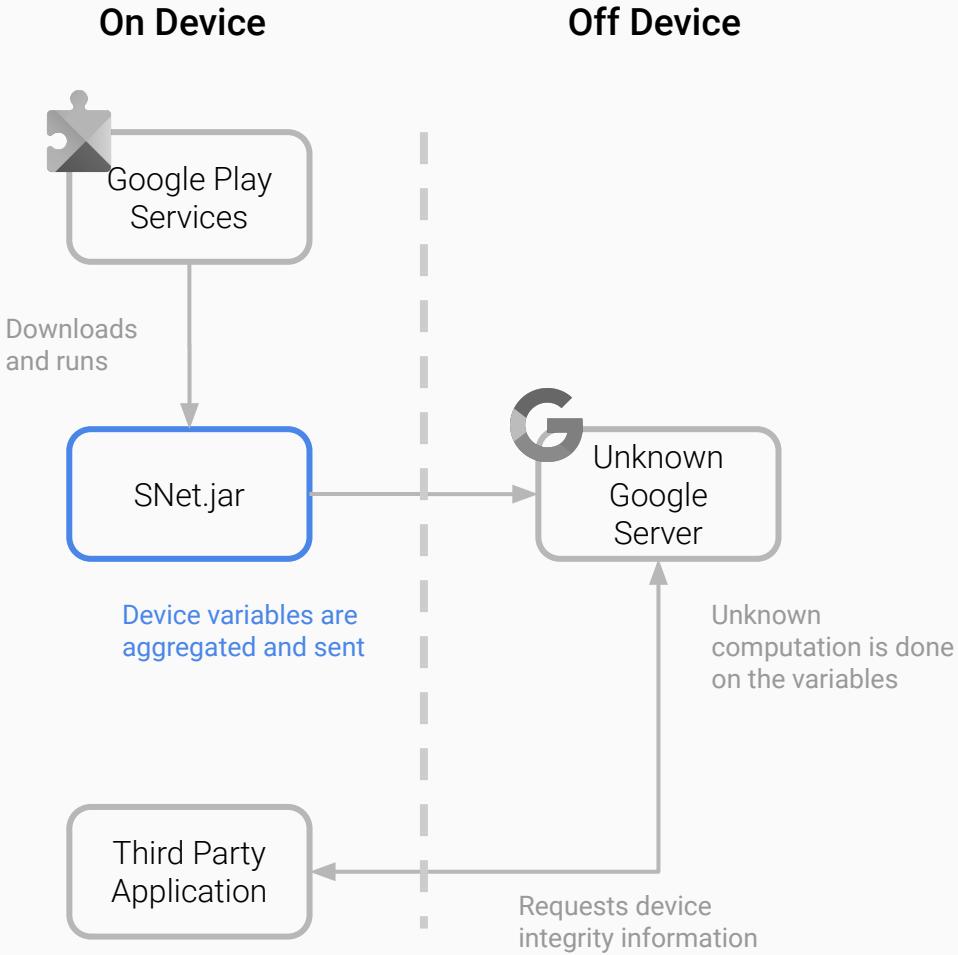
Unknown
computation is done
on the variables

ctsProfileMatch: A stricter verdict
of device integrity.

basicIntegrity: A more lenient
verdict of device integrity.

SNet Attestation

The SafetyNet Attestation API provides services for determining whether a device running an app satisfies Android security tests.



Device State Checker

Verify Boot State:

Green : device is LOCKED and user-settable root of trust is not used

Yellow: if device is LOCKED and user-settable root of trust is used

Orange: if device is UNLOCKED

Verifying Mode:

Enforcing - VERITY MODE DEFAULT

Logging - VERITY MODE LOGGING

Security Patch Level:

2020-01-05

2020-06-05

etc...

```
verifiedBootState = "orange"  
verityMode = "enforcing"  
securityPatchLevel = "2018-07-05"  
oemUnlockSupported = 1  
productBrand = "google"  
productModel = "Pixel 2"
```

OEM Unlock Supported:

1 - Bootloader Unlock
Supported

0 - Bootloader
Unlocked Not
Supported

Device State
Checker

Settings
Finder

ADB Enabled:

True - device can
communicate over the
Android Debug bridge

False - device cannot
communicate over the
Android Debug bridge

Fingerprint Status:

0 - FINGERPRINT
NOT SUPPORTED

1 - FINGERPRINT
ENROLLED

2 - FINGERPRINT
UNENROLLED

Lock Screen Timeout:

This preference allows the device to be locked a given time after the screen turns on

f adbEnabled = true
f fingerprintStatus = 2
f lockScreenTimeout = 0
f lockScreenType = 1
f nonMarketAppsEnabled = true
f notificationVisibility = 1
f smartLockEnabled = false
f smartLockStatusObtained = false
f storageEncryptionStatus = 5

Lock Screen Type:

0 - LOCK SCREEN
TYPE NONE

1 - LOCK SCREEN
TYPE UNKNOWN

2 - LOCK SCREEN
TYPE PIN

3 - LOCK SCREEN
TYPE PATTERN

4/5 - LOCK SCREEN
TYPE FACE PIN/
PATTERN

6 - LOCK SCREEN
TYPE PASSWORD

Device State
Checker

Settings
Finder

Non Market Apps

True - User can download applications from non-Play-Store sources

False - User cannot download applications from non-Play-Store sources

Notification visibility

0 - NOTIFICATION TYPE NONE

1 - NOTIFICATION TYPE PRIVATE
- *no information*

2 - NOTIFICATION TYPE PUBLIC
- *all information*

3 - NOTIFICATION TYPE SECRET
- *some information*

Android Smart Lock

An Android device will stay unlocked if connected to a ‘trusted device’ or when at a predefined location.

Storage Encryption Status

0 - ENCRYPTION STATUS UNSUPPORTED

1 - ENCRYPTION STATUS INACTIVE

2 - ENCRYPTION STATUS ACTIVATING

3 - ENCRYPTION STATUS ACTIVE

4 - ENCRYPTION STATUS ACTIVE DEFAULT KEY

5 - ENCRYPTION STATUS ACTIVE PER USER

```
f adbEnabled = true
f fingerprintStatus = 2
f lockScreenTimeout = 0
f lockScreenType = 1
f nonMarketAppsEnabled = true
f notificationVisibility = 1
f smartLockEnabled = false
f smartLockStatusObtained = false
f storageEncryptionStatus = 5
```

Device State
Checker

Settings
Finder

SD Card
Analyzer

Saves a .jpg onto the external storage

waits

Checks if the .jpg is still present

jpegMissing

Checks if the .jpg has been modified

jpegTampered

jpegTampered
Bytes

jpegTampered
Bytes

jpegWrongLen
gth

jpegModificatio
nTime

```
▶ f jpegFileName = "TAMPER_CHECK_JPEG_NAME"  
f jpegMissing = true  
f jpegModificationTime = 0  
f jpegNewlyTampered = false  
f jpegTampered = false  
f jpegTamperedBytes = 0  
f jpegWrongLength = 0
```

Device State
Checker

Settings
Finder

SD Card
Analyzer

Rooting File
Finder

Scans filesystem for list of known file paths

"/system/bin"

"/dev/block/loop"

"/system/xbin"

"/bin"

"/xbin"

"/proc/self/mou
ntinfo"

etc...

Stores filename, sha256, and if the file exists

```
▶ f filename = "/sbin/su"  
▶ f present = {Boolean@5486} true  
▶ f sha256 = "597ac3b8e31176ff2519f8d89995848c06caaa145f1011b95bbf9567f80b6f3e"
```

Device State
Checker

Settings
Finder

SD Card
Analyzer

Rooting File
Finder

Preferred
Package
Finder

```
> f processName = "com.google.android.packageinstaller"
> f publicSourceDir = "/system/priv-app/GooglePackageInstaller/GooglePackageInstaller.apk"
> f requiresSmallestWidthDp = 0
▼ f resourceDirs = [String[2]@5418]
  ▶ 0 = "/vendor/overlay/framework-res_auto_generated_rro.apk"
  ▶ 1 = "/vendor/overlay/PixelThemeOverlay.apk"
> f scanPublicSourceDir = "/system/priv-app/GooglePackageInstaller"
> f scanSourceDir = "/system/priv-app/GooglePackageInstaller"
> f selInfo = "default:privapp:targetSdkVersion=27"
> f processName = "com.android.chrome"
> f publicSourceDir = "/data/app/com.android.chrome-eFxYK7fPlyCINGO9Cd1Gjw==/base.apk"
> f requiresSmallestWidthDp = 0
> f resourceDirs = [String[2]@5455]
  ▶ f scanPublicSourceDir = "/data/app/com.android.chrome-eFxYK7fPlyCINGO9Cd1Gjw=="
  ▶ f scanSourceDir = "/data/app/com.android.chrome-eFxYK7fPlyCINGO9Cd1Gjw=="
> f selInfo = "google:targetSdkVersion=30"
```

Preferred Packages

Configured applications responsible for a particular role

Preferred Package
Installer

Preferred Web
Browser

Package Name

Source Directory

Version Name

Device State
Checker

Settings
Finder

SD Card
Analyzer

Rooting File
Finder

Preferred
Package
Finder

Captive
Portal
Detector

Connects to a hardcoded Google server

clients3.google.
com

Returns connection results

ipAddressUsed

responseBody

responseCode

```
f bodyEmpty = false
f ipAddressUsed = "216.58.211.174"
f responseCode = 204
f userAgent = "TMP_USER_AGENT"
f userAgentIndex = 0
```

Device State
Checker

SD Card
Analyzer

Preferred
Package
Finder

Settings
Finder

Rooting File
Finder

Captive
Portal
Detector

Locale

Proxy Analyzer

Report System
Apps

Report Suspicious
Google Page

Device Country

JVM Properties

SPKI Whitelist

Report More App
Info

Report SSL v3
Tests

Cached Logs

System
Properties

Is Sidewinder
Device

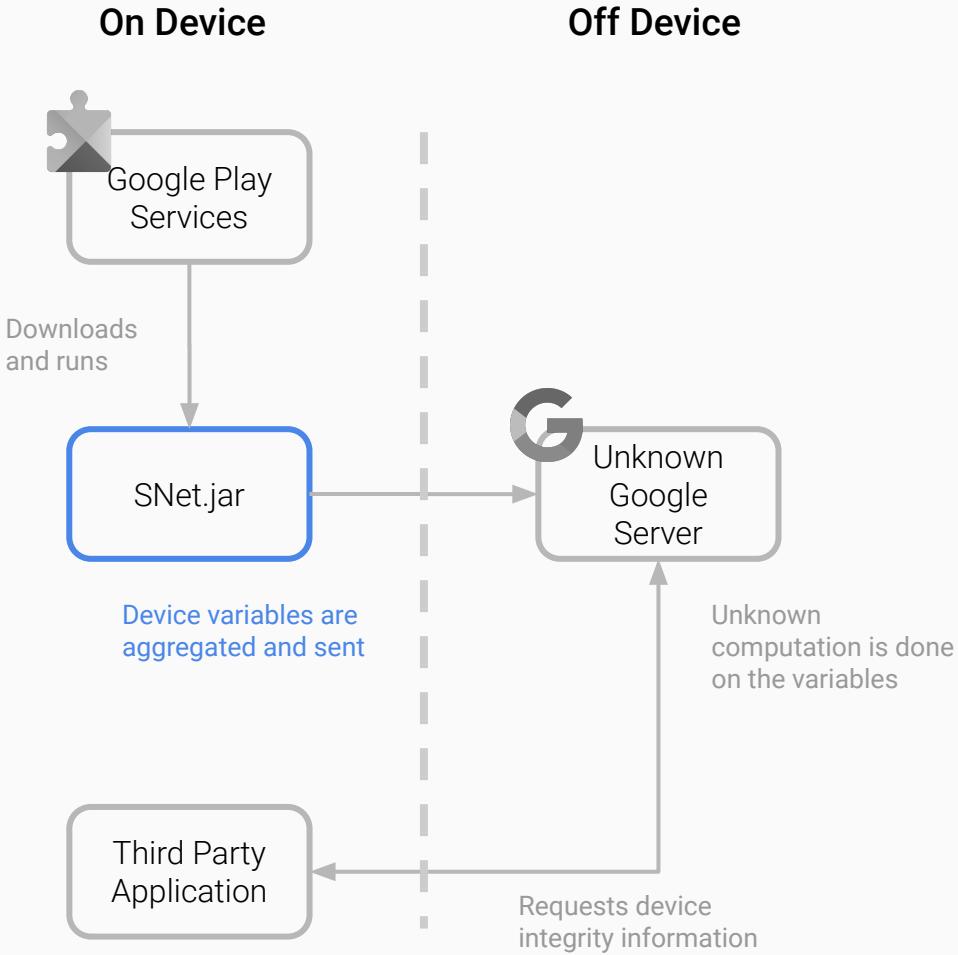
Report Google
Page

SNet Is Targeted
By GServices

~ 55 across all (*known*) versions

SNet Attestation

The SafetyNet Attestation API provides services for determining whether a device running an app satisfies Android security tests.



Wrap-up and Questions

'Blue Team' Wrap-Up:

- 🛡 Utilise Safetynet, and other system based attestation mechanisms for better protection.
- 📱 Also introduce local root detection checks for wider protection.

'Red Team' Wrap-Up:

- 🔍 Use tools like DIS{INTEGRITY} to find root and integrity checks inside of APKs
- 🔗 Hook local checks and calls to APIs like Safetynet, with tools such as Objection and Frida.



@_JamesStevenson



[Github.com/user1342](https://github.com/user1342)

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

user1342 / RUNIC Public

Issues Pull requests Actions Security Insights Settings

Github.com/user1342/RUNIC

README.md

RUNIC is an Android tamper detection demo. It features a notification bar with a green icon and a status bar showing battery level and signal strength. The main screen displays a large green arrow pointing right, with the word "RUNIC" above it. Below the arrow, there's a "Notifications" section showing a single entry: "Runic Integrity Survey Result" with the message "Device integrity is low. Scored 40.0% out of 100%". The "Silent" section shows "Android System Serial console enabled". The "Contributors" section shows 1 contributor, and the "Last commit" is by "may" on March 19.

RUNIC: Android tamper detection demo

contributors 1 last commit may

C, an Android tamper detection demo! RUNIC is designed to serve as a parallel more complex tamper detection and integrity systems such as Google Project. By exploring the inner workings of RUNIC, you can gain insights into the employed in these advanced solutions.

A client/server model where the application collects device variables and sends them to a server then provides a percentage score indicating the likelihood of the device being tampered with. This is the successor to the previous tool, Tamper, and is currently in a minimal viable state. For specific improvements, issues, or feature requests, please use the issues tab.

Getting Started

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

user1342 / DISintegrity Public

Issues Pull requests Actions Projects Wiki Security Insights Settings

Github.com/user1342/DISintegrity

README.md

DIS{INTEGRITY} is a tool for analysing Android APKs and extracting root, integrity, and tamper detection information. It uses APKTool to break down APK files and extracts data from Android code, and other resources to identify these security checks. The tool generates an easy-to-under

contributors 2 Stars 29 Watchers 2 last commit may

RUNIC and Dis{Integrity}