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




@_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

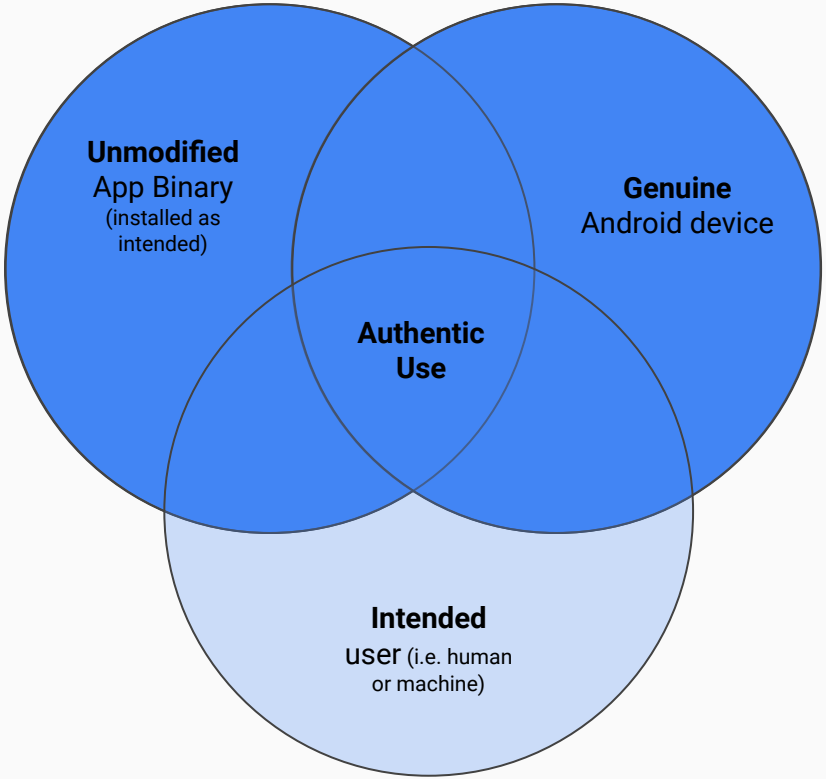
Side Projects:

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

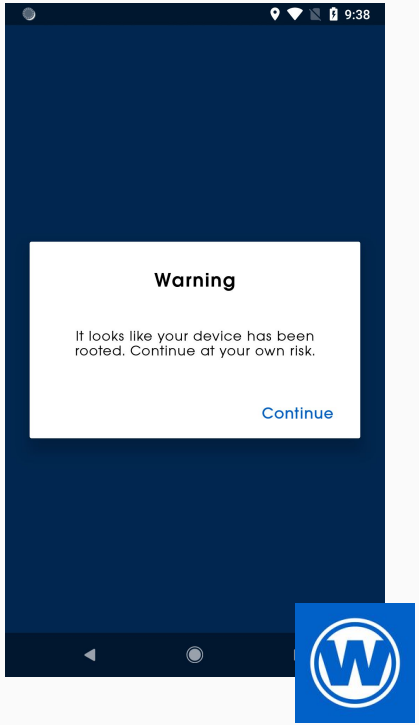
📚 **Self-Published Author** - Mobile Offensive Security Pocket Guide

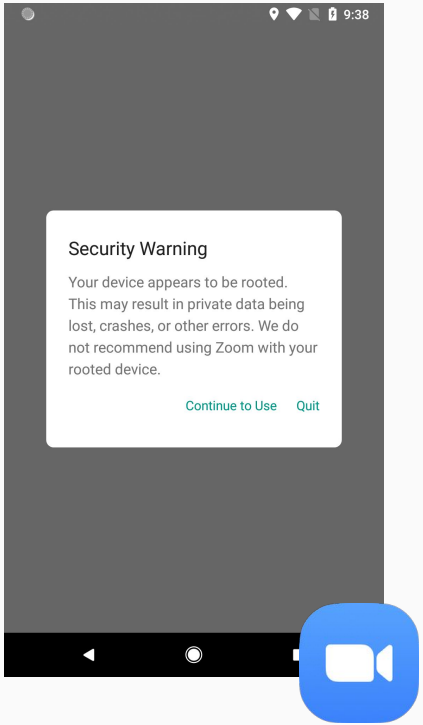
🎓 **Part-time PhD Student** - Bristol University

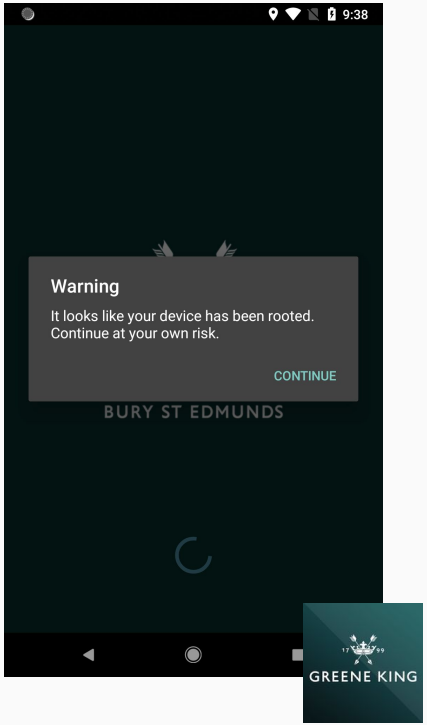
🎤 **Occasional Conference** Speaker

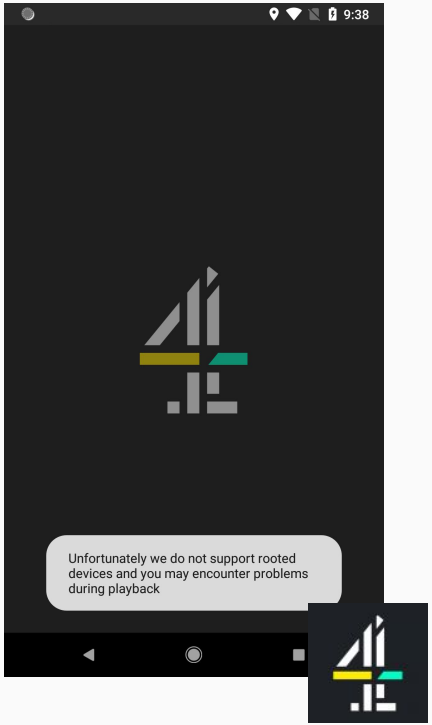


What is ‘attestation’ looking to solve?









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

Normal Device

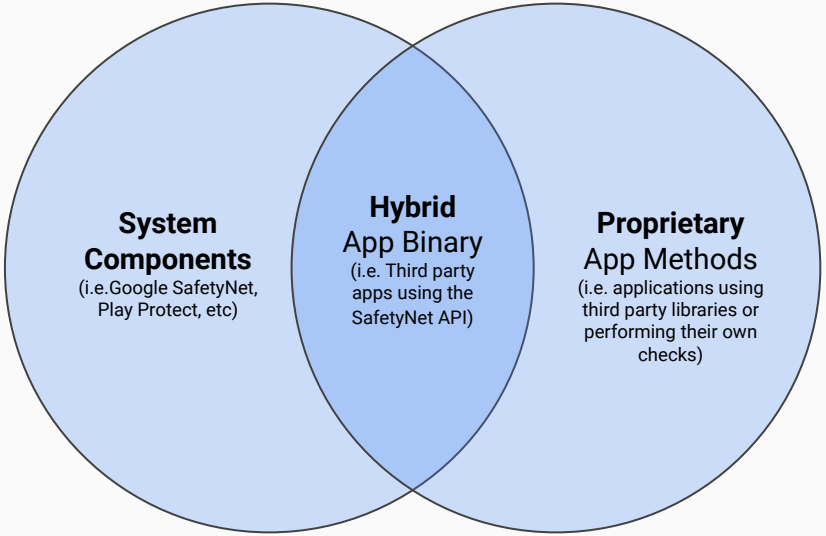
Moto G6 API level 28

Rooted Device

Pixel 2 API level 27

Types of Integrity Checking

System and Proprietary Attestation Checks



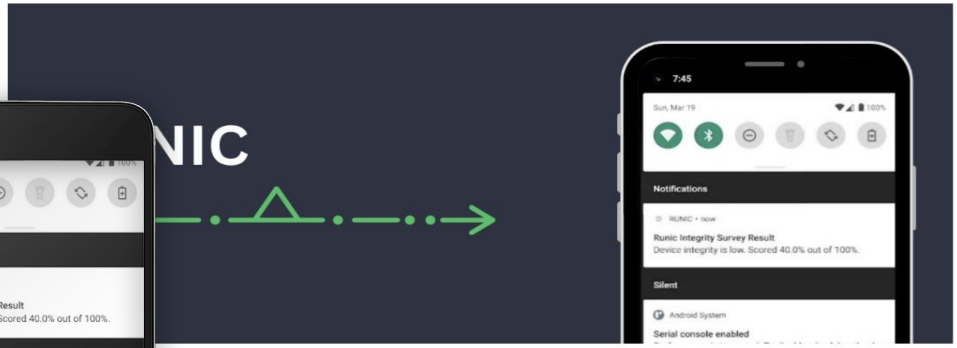
Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

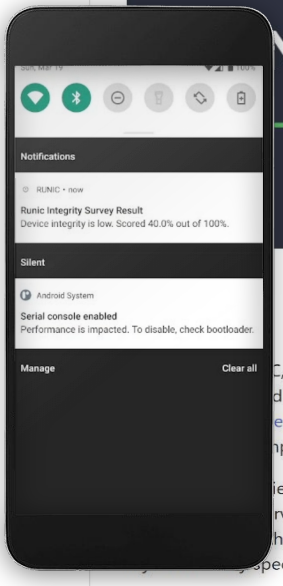
user1342 / RUNIC Public

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

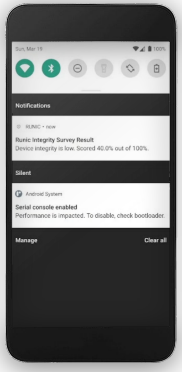
ko-fi.com/jamesstevenson

Languages

Java 100.0%

Github.com/user1342/RUNIC

RUNIC: Android tamper detection demo

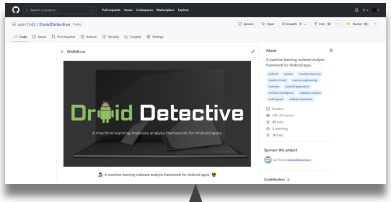


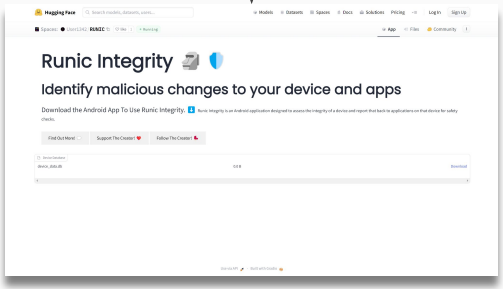

```
adb_enabled': 'true',
 'boot_state': '',
 'brand': 'Android',
 'debug_enabled': 'true',
 'emulator': 'true',
 'fingerprint_status': '2',
 'id': '935efe59519bcd25',
 'installer': 'false',
 'lock_screen_timeout': '0',
 'lock_screen_type': '0',
 'market_apps_enabled': 'true',
 'model': 'Android SDK 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

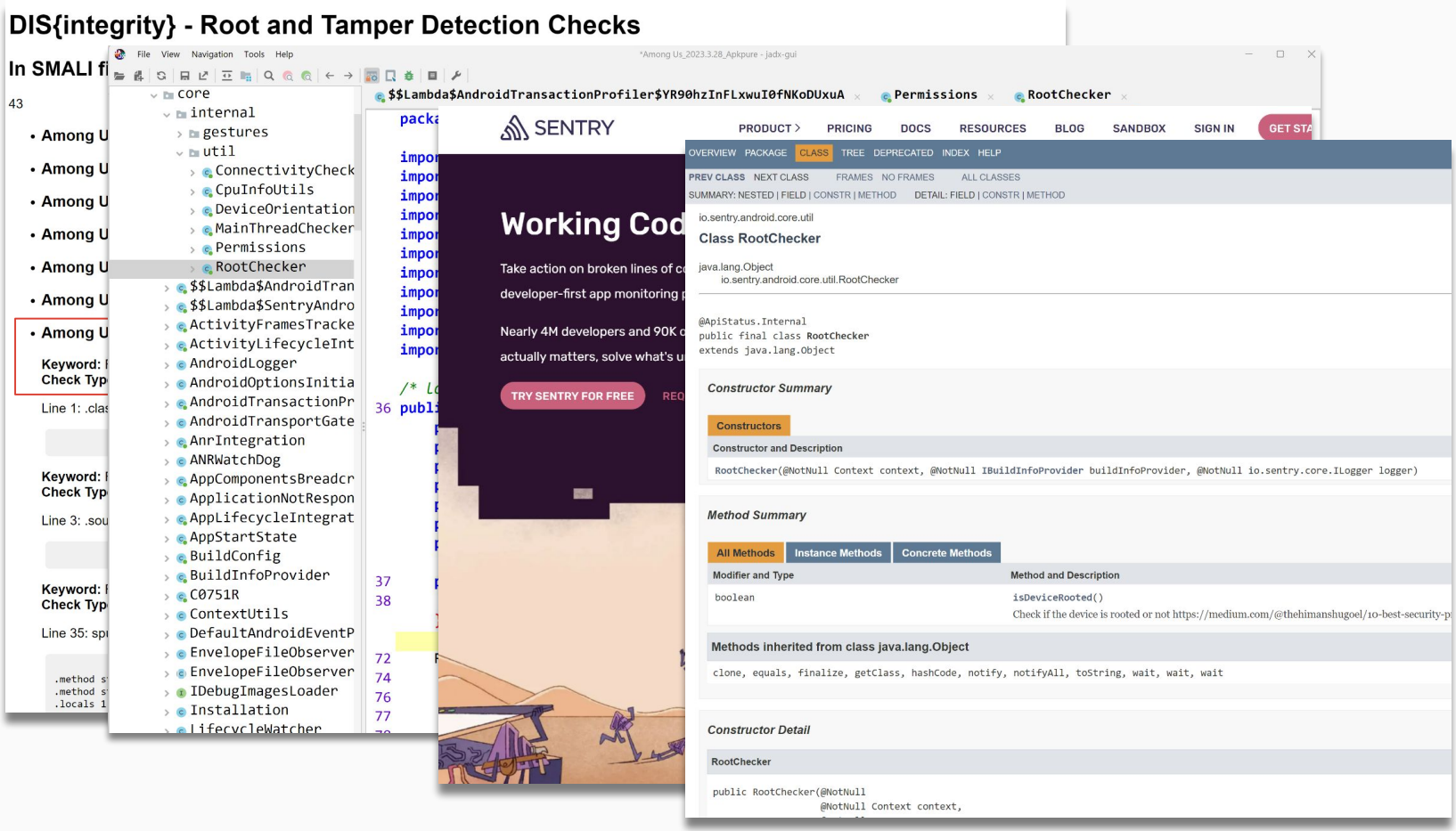
Application Side

- 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

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()

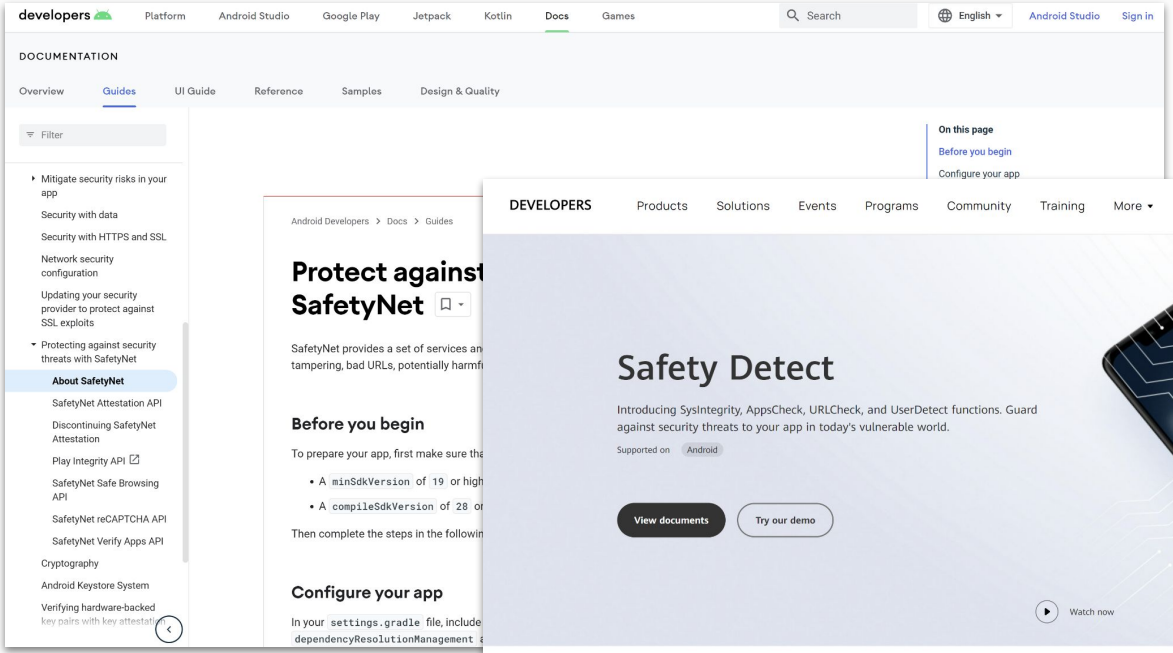
"/system/app/**Superuser.apk**","/sbin/**su**","/system/bin/su","/system/xbin/su","/data/local/xbin/su","/data/local/bin/su","/system/sd/xbin/su","/system/bin/failsafe/su","/data/local/su","/su/bin/su","/su/bin","/system/xbin/**daemonsu**"...

checkSUExist()

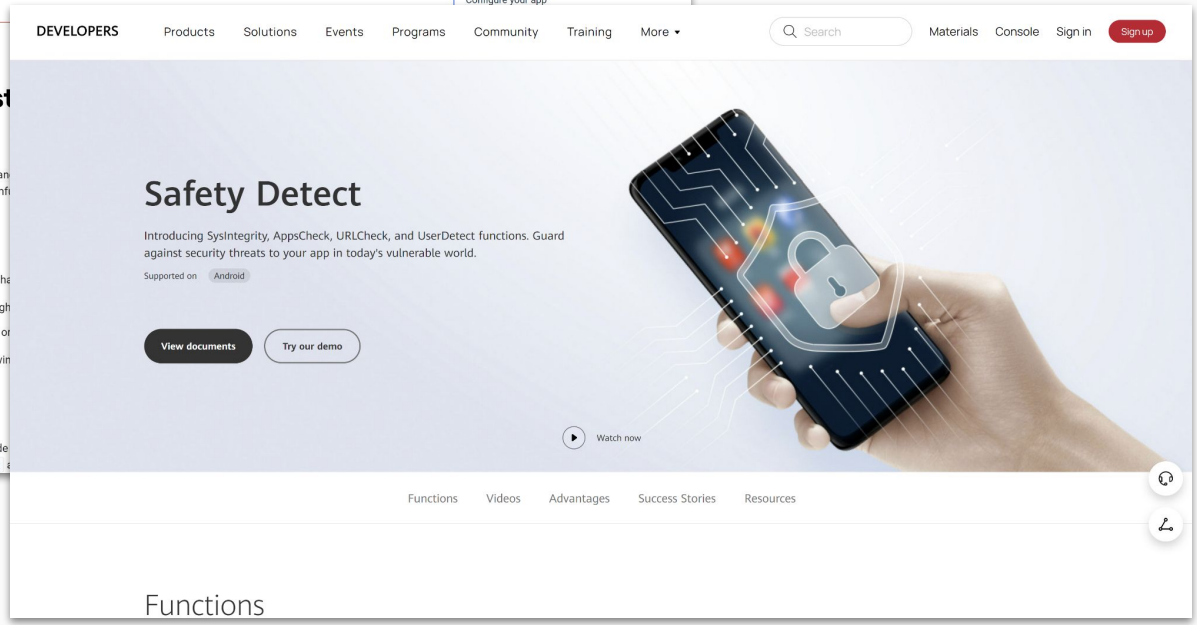
```
Process exec = this.runtime.exec(new String[] {
    "/system/xbin/which",
    "su"
  });
```

checkRootPackages()

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

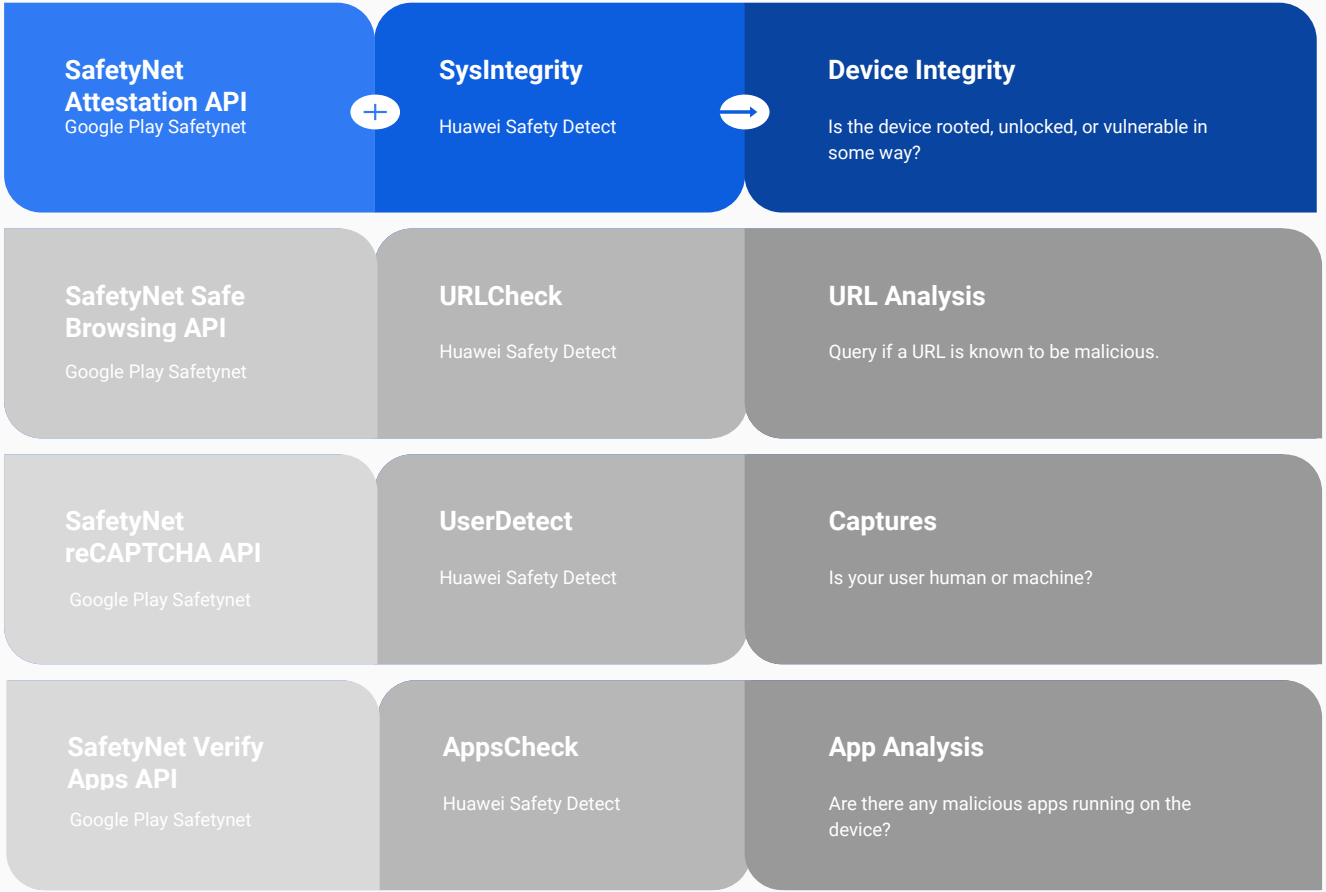


Google Play Services SafetyNet
(replaced by Play Integrity API in Jan 2025)



Huawei AppGallery Safety Detect

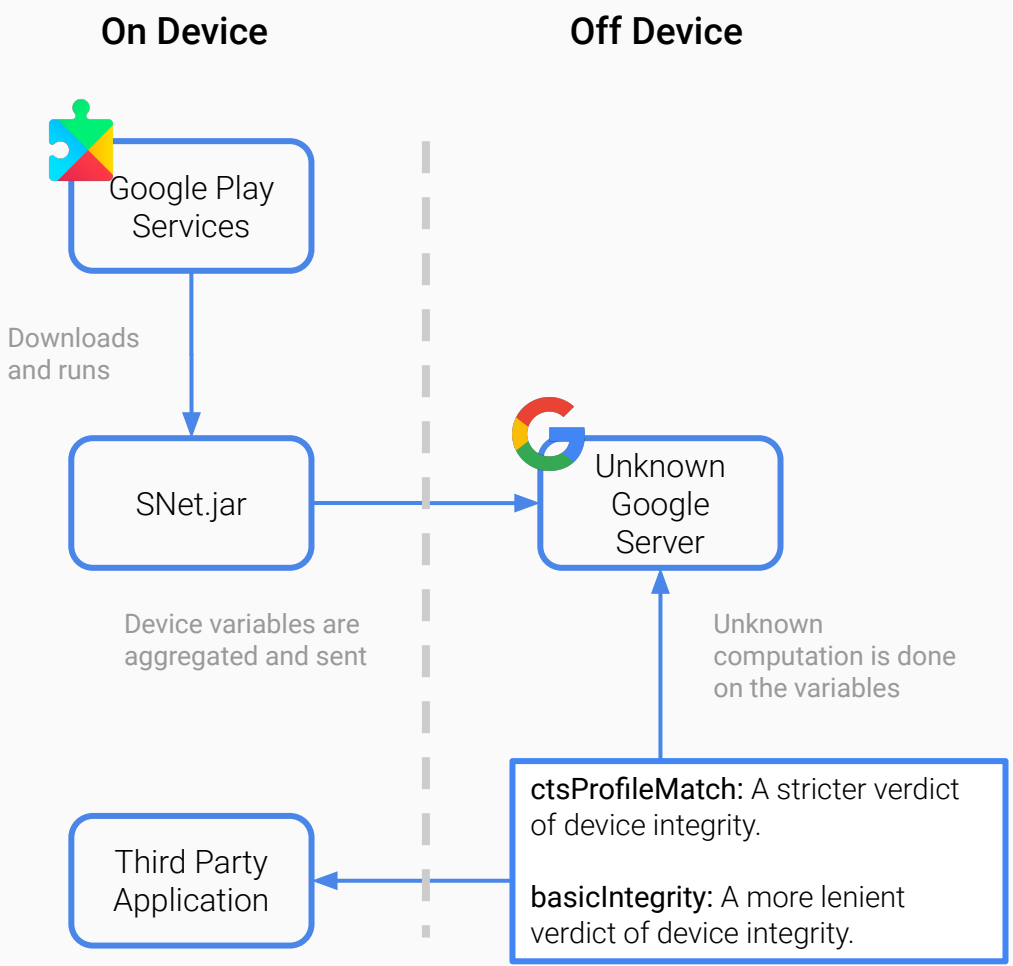
Main contenders in tamper detection



Play and AppGallery Integrity Checks

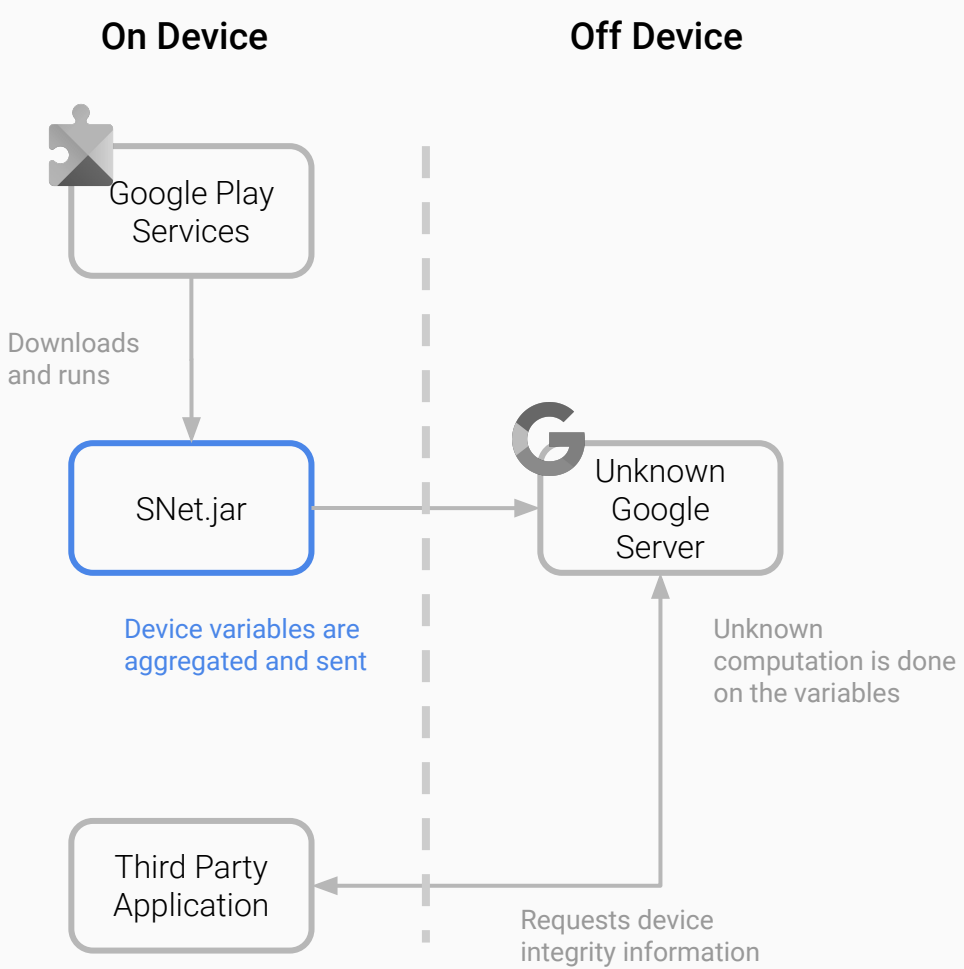
SNet Attestation

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



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

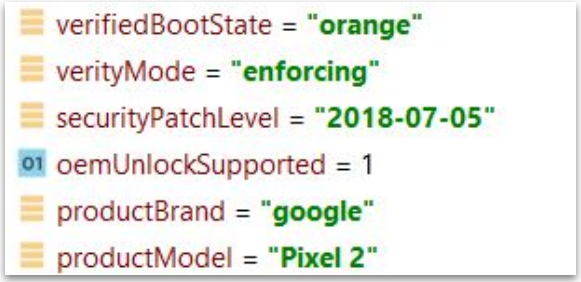
2020-06-05

etc...

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:

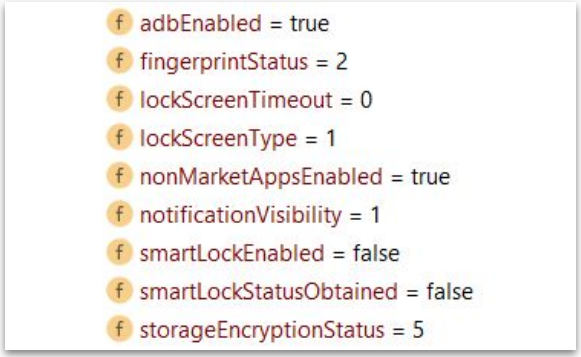
0 - FINGERPRINT NOT SUPPORTED

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



Lock Screen Type:

0 - LOCK SCREEN TYPE NONE

1 - LOCK SCREEN TYPE UNKNOWN

2 - LOCK SCREEN TYPE PIN

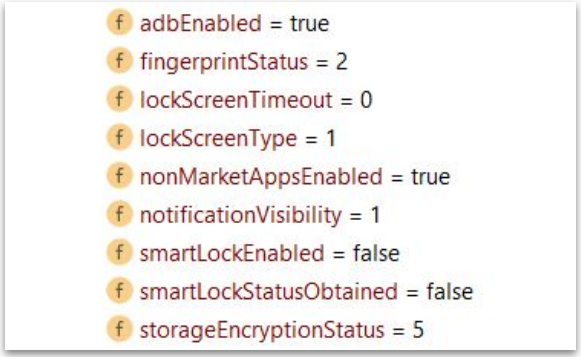
3 - LOCK SCREEN TYPE PATTERN

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

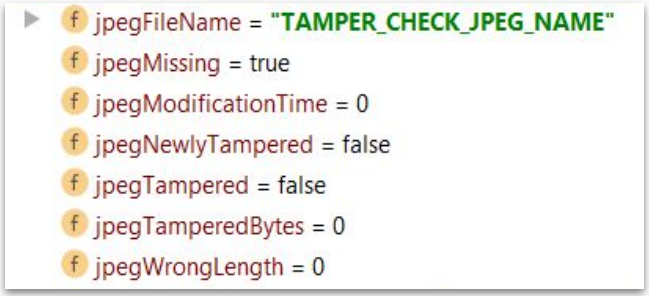
4 - ENCRYPTION STATUS ACTIVE DEFAULT KEY

5 - ENCRYPTION STATUS ACTIVE PER USER

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

jpegWrongLength

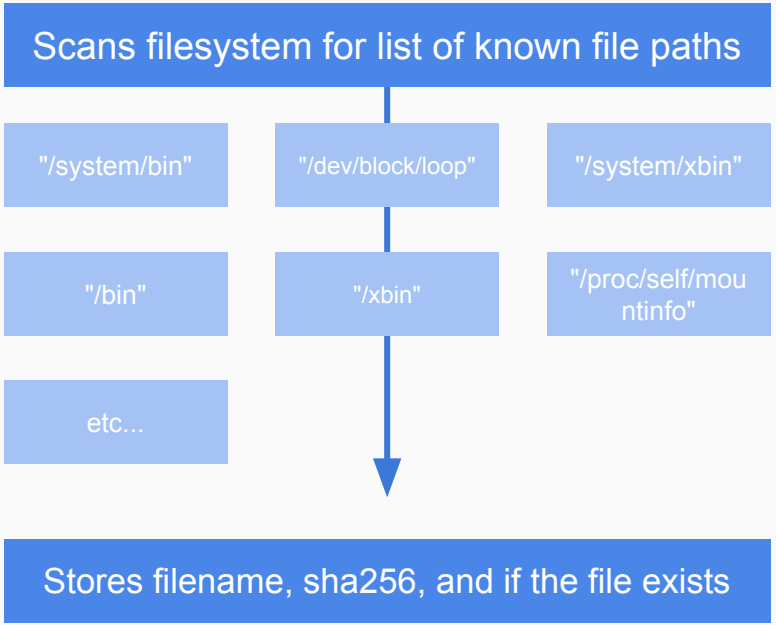
jpegModificationTime

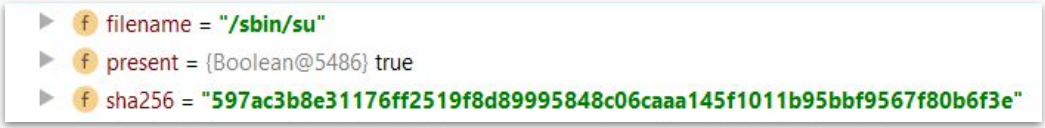
Device State Checker

Settings Finder

SD Card Analyzer

Rooting File Finder





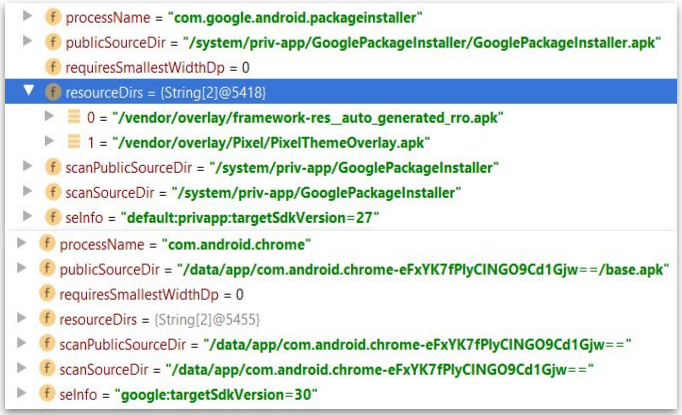
Device State Checker

Settings Finder

SD Card Analyzer

Rooting File Finder

Preferred Package Finder



Preferred Packages

Configured applications responsible for a particular role

Preferred Package Installer

Preferred Web Browser

Package Name

Source Directory

Version Name

Device State Checker

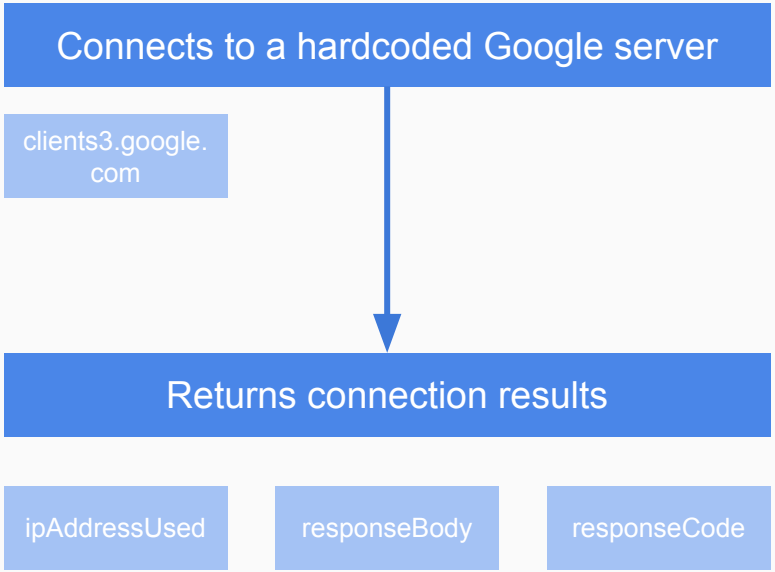
Settings Finder

SD Card Analyzer

Rooting File Finder

Preferred Package Finder

Captive Portal Detector




```
bodyEmpty = false
ipAddressUsed = "216.58.211.174"
responseCode = 204
userAgent = "TMP_USER_AGENT"
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

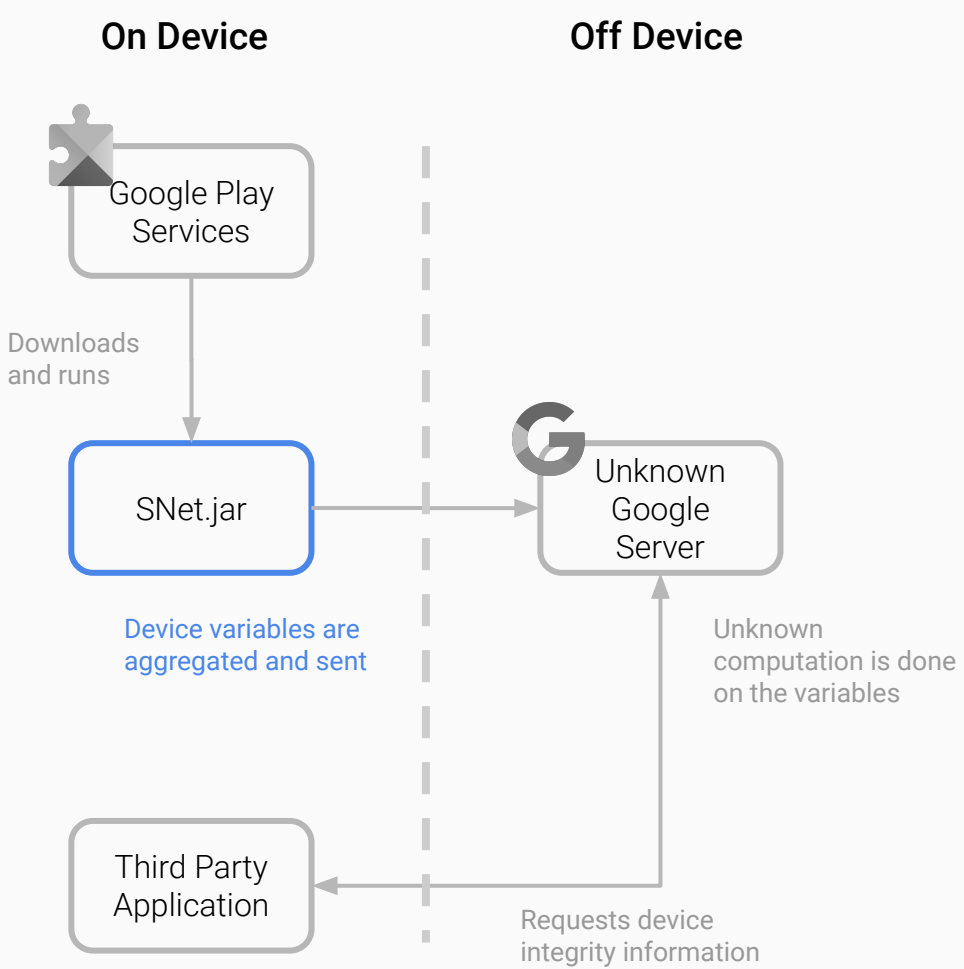
Report Google Page

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 D*IS{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

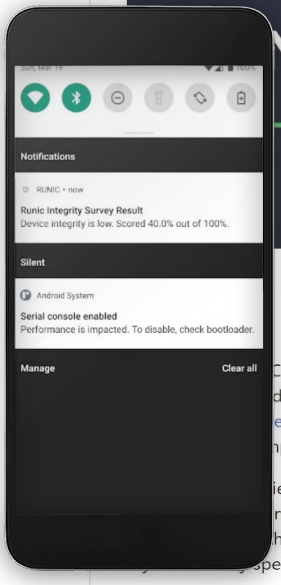
Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

user1342 / RUNIC Public

Issues Pull requests Actions Security Insights Settings

README.md

Github.com/user1342/**RUNIC**



RUNIC: Android tamper detection demo

contributors 1 last commit may

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

ient/server model where the application collects device variables and sends
rver then provides a percentage score indicating the likelihood of the device
he successor to the previous tool, Tamper, and is currently in a minimal viabl
pecific 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

README.md

Github.com/user1342/**DISintegrity**

A tool for analysing Android APKs and extracting root, integrity, and tamper detec

contributors 2 Stars 29 Watchers 2 last commit may

DIS{integrity} is a tool for analyzing Android APKs, focusing on identifying root, integrity, and ta
checks inside of the APK. It uses APKTool to break down APK files and extracts data from Andro
code, and other resources to identify these security checks. The tool generates an easy-to-unde

RUNIC and Dis{Integrity}