A Detailed Comparison of Apple, Android & Huawei Mobile App Security

approov

WHITEPAPER

A Detailed Comparison of Apple, Android and Huawei Mobile App Security

Contents

This whitepaper will initially present the overall security challenges presented by mobile apps and explain why app and device attestation is a critical part of a comprehensive approach to protecting apps and APIs they use. We will then describe in detail the app and device attestation security offerings from Apple, Google and Huawei:

- *Apple DeviceCheck and App Attest*
- *Google Play Integrity API*
- *Huawei Safety Detect*

Each of these solutions only work with specific platform/appstore pairs (iOS and Apple App Store, Android and Google Play, or HarmonyOS and Huawei Gallery). This means that there is significant overhead for an app developer to deploy these divergent solutions across multiple platforms.

More importantly, regulation around the world such as the EU's DMA (Digital Markets Act), the UK's DMCC (Digital Markets, Competition and Consumers Act 2024), and Japan's SSCPA (Smartphone Act) is forcing these tech giants to permit alternative app stores and sideloaded apps. The proprietary solutions from Apple, Google and Huawei may not work for sideloaded apps or alternative app stores.

This paper concludes by presenting a case for a new approach to mobile app security: one that secures all apps and APIs, for all devices and wearables, separate from app stores and independent of how the apps are delivered to consumers - and at the same time, an approach that provides end-to-end security but also makes life for developers and devops teams easy as they embrace cross-platform development tools.

The Threat Surfaces Presented by Mobile Apps

There are two fundamental security challenges with mobile apps: The first is that they can be reverse engineered, even if an obfuscation tool has been used. The second is that they run in a client environment which is open to hackers and is neither owned or controlled by the app owner. Unless steps are taken, apps can be analyzed, understood, cloned or copied, and the environments they run in can be hacked, rooted, instrumented and manipulated to interfere with the operation of an app.

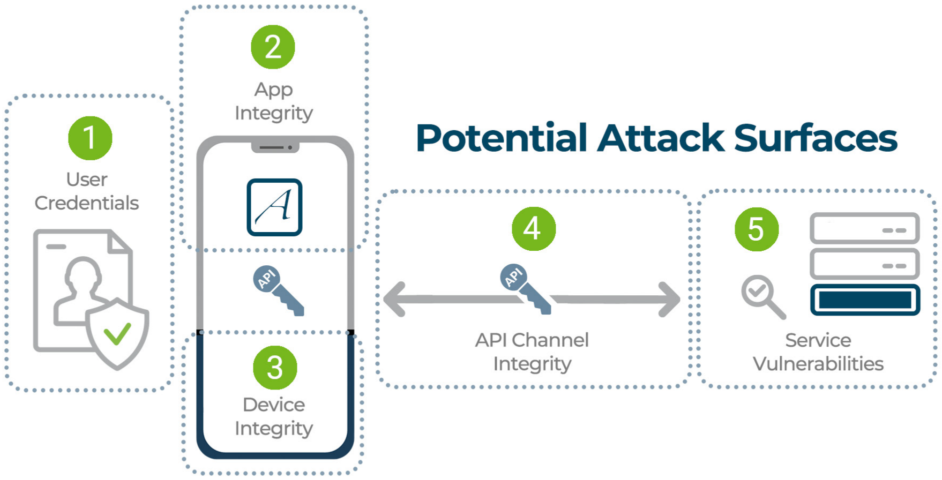

Diagram 1: Potential Attack Surfaces (Source: approov.io)

Refer to diagram 1 to see the principal attack surfaces employed by hackers:

1. **User Credentials:** Credentials can be stolen and reused.
2. **App Integrity:** Apps can be modified, cloned or imitated by scripts.
3. **Device Integrity:** Software running on devices can interfere with the operation of the app.
4. **API Channel Integrity:** Even with TLS, Man-in-the-Middle attacks can be executed using software running on the device.
5. **Service or API Vulnerabilities:** API vulnerabilities can be exploited in automated attacks.

Using these attack surfaces, hackers can:

- Directly intercept or tamper with data transferred between the app and its servers.
- Intercept or manipulate any transactions.
- Interfere with or stop the operation of the service.

Manipulated apps can be :

- Repackaged and redistributed with malware.
- Turned into automated tools (i.e. bots) to be used to attack APIs and backend servers.

Secrets can also be lifted from apps or from cloud repositories and then used in scripts to create bots.

Bad actors use all these techniques to carry out brute-force attacks, exploiting API vulnerabilities to steal data, or mount DDoS attacks on backend servers.

Of course there are security techniques which are currently used, for example:

- **Code Obfuscation:** Good development discipline and taking steps to protect your mobile app code from tampering or reverse engineering should always be employed. However, secrets required to attack your APIs can be acquired by hackers from other channels, so protecting the app code must be part of a broader security program that also focuses on protecting the mobile API and the critical assets on the server from bad actors who are imitating your app.
- **Testing:** Static analysis, MAST (Mobile Application Security Testing) and DAST (Dynamic Application Security Testing) all have a role to play before an app is deployed but testing relies on known vulnerability databases and predefined attack patterns and may not detect currently unknown or zero-day exploits. In addition, business logic issues, where the application behaves correctly from a technical standpoint but allows unintended actions due to logical errors, can be difficult to identify with scanning and test tools.
- **Server-side App Security:** Traditional server-side solutions such as a WAF or Bot Mitigation solutions rely on known patterns and involve constant maintenance and updates to keep up with the latest attack vectors. In addition, most are heavily reliant on browser fingerprinting which is not relevant with mobile apps and they don't effectively detect scripts impersonating mobile apps. In general, it is good practice to eliminate as much traffic as possible before it hits the backend.
- **Transport Level Security:** Deploying TLS certainly provides an encrypted channel between the mobile app and the server. Certificate pinning can prevent MitM attacks. However devops teams often push back on this best practice, citing concerns about performance and availability, meaning traffic is exposed. You can't depend on patients and healthcare professionals being on secure networks, and if your TLS is not managed properly third parties can steal secrets and manipulate your APIs.

However it is important to supplement these protections with a runtime zero trust approach which evaluates and validates every request and evaluates threats in real time.

Why App Attestation and Device Integrity Checks are Important

Because of this, taking steps to prevent apps and devices from being tampered with at runtime must be at the heart of any security strategy. However, this is only a fraction of the whole story, as we will explain. App and device attestation are an essential piece of the puzzle but not sufficient in themselves.

Different services are available to provide app attestation. Apple, Google and Huawei all provide some app attestation and client integrity checking. The rest of this paper looks at these in depth, and compares them with the solution from Approov, which provides a unique and patented (https://approov.io/info/patents) approach to end-to-end mobile app security, including dynamic app and device integrity checking.

A Brief History of Apple, Google and Huawei Mobile Security

The History of Apple DeviceCheck and Apple App Attest

DeviceCheck is an iOS framework, first introduced in iOS 11. The primary use-case for DeviceCheck when it was launched was to provide developers a way to tell when customers who have received premium content in an app, to stop abuse.

In August 2020, with the release of iOS 14, Apple added a new API to the framework called App Attest. As a part of DeviceCheck, it was intended to help to prevent the inappropriate use of developer servers through compromised apps.

More information about the history of DeviceCheck and App Attest can be found in this blog.

The History of Google Play Integrity API

SafetyNet attestation API was launched in 2017 as part of Google Play services, in order to provide an API for developers to remotely evaluate whether they were talking to a genuine Android device. Because it was difficult to use, it didnt take off, and in 2021 Google announced Play Integrity API, consolidating multiple integrity offerings (including the SafetyNet Attestation device verdict) under a single API.

At the same time they announced the deprecation of SafetyNet Attestation API, which will be turned off completely in January 2025, forcing Android developers who used SafetyNet to change to the new attestation service.

More information on the history of Google Play Integrity API can be found in this blog.

The History and Adoption of HarmonyOS and Huawei Safety Detect

HarmonyOS was launched in 2019 – initially as an OS for IoT devices, but later adapted to run on smartphones and other hardware. Huawei intends that the OS will offer a unified experience across wearables, tablet computers, smartphones, and smart TVs. Release 4 of HarmonyOS came in 2023.

Huawei's Safety Detect was officially launched in 2019 as part of Huawei Mobile Services (HMS) Core, designed to provide a range of security features to app developers. Huawei's device checking SysIntegrity closely resembles Android SafetyNet, both in terms of the API and the attestation process.

More information on the history and adoption of HarmonyOS and Huawei Safety Detect can be found in this blog.

The Scope of Apple, Google and Huawei Security - What's Covered and What's Not

Apple, Google and Huawei all depend on their proprietary app stores and the validation process before apps are published.

It is obvious that Apple, Google and Huawei attestation only works with their respective operating systems and doesn't help for other client OS or with cross-platform development frameworks. We will talk more about that in the next section.

We will also see in the next section that even if we look only at app attestation alone, there are some shortcomings in the approaches offered by these players.

To be fair, these tools are not intended to provide an "end-to-end" solution. Apple states that DeviceCheck and App Attest contribute to an "overall risk assessment" of your environment and they see it working as part of a bigger effort to secure your mobile apps.

To put things in some context, we can use, for example, the OWASP MASVS (Mobile Application Security Verification Standard) framework which is a comprehensive framework used to assess end-to-end mobile app security.

The Apple, Google and Huawei solutions ONLY partially address the guidelines in the category MASVS-RESILIENCE, which is only one of seven categories in the guidelines. MASVS-RESILIENCE aims to ensure that the app is running on a trusted platform, prevent tampering at runtime and ensure the integrity of the app's intended functionality.

In particular there are two other key things in MASVS that Apple, Google and Huawei do not address which you will need to take care of:

- **Network and Channel Security:** The APIs and the communications channel between app and APIs must also be protected.
- **Management and Security of API Keys and Secrets:** The secrets used to authenticate and authorize access to backend services from mobile apps must be protected from being stolen and abused. Developers need to manage cryptographic keys securely and handle key rotation appropriately for ongoing security.

Now we understand the overall scope, let's look specifically at the app attestation and device integrity checks provided by Apple and compare them with Approov.

Detailed Description of Apple, Google and Huawei App and Device Attestation

For each one, let's look at what it is supposed to do and how it works.

Overview of Apple DeviceCheck and App Attest

Apple DeviceCheck was introduced as a way to stop users claiming special offers more than once. When the App Attest feature was added, Apple noted in developer documentation that apps can be modified and distributed outside of the App Store, leading to versions of those apps with unauthorized features like "game cheats, ad removal, or access to premium content." App Attest was put in place to provide some guarantee that apps are genuine and untampered.

Implementing DeviceCheck and App Attest together allows you to:

Manage the Device State

DeviceCheck Binary flags: You can set and query two binary flags per device using Apple's servers. These flags are like switches you can flip based on your app's needs. It's up to you how you use them. For example, you could:

- Set a flag when a user claims a free offer, preventing them from claiming it again on the same device.
- Flag a device suspected of fraudulent activity for further investigation.

Apple maintains the state of these flags for you, even if the user uninstalls and reinstalls your app. This ensures consistency and prevents abuse.

Verify App Integrity

Using App Attest, your app generates a special cryptographic key on the device and has Apple verify its validity. This helps your server:

- Distinguish your genuine app from pirated or tampered versions.
- Grant access to sensitive resources like premium content or secure transactions with greater confidence. By guaranteeing the app's authenticity, App Attest helps prevent unauthorized access, fraud, and malware outbreaks.

Apple DeviceCheck

DeviceCheck provides developers a reliable way to identify a device and can store two bits of information about the device per app. The flags are maintained even if the app is deleted and reinstalled, a user logs in with different credentials or a system reset is performed. Typical flags can be used to indicate 'Has this device ever been compromised?' or 'Has this app's free promotion period been used for this device?'.

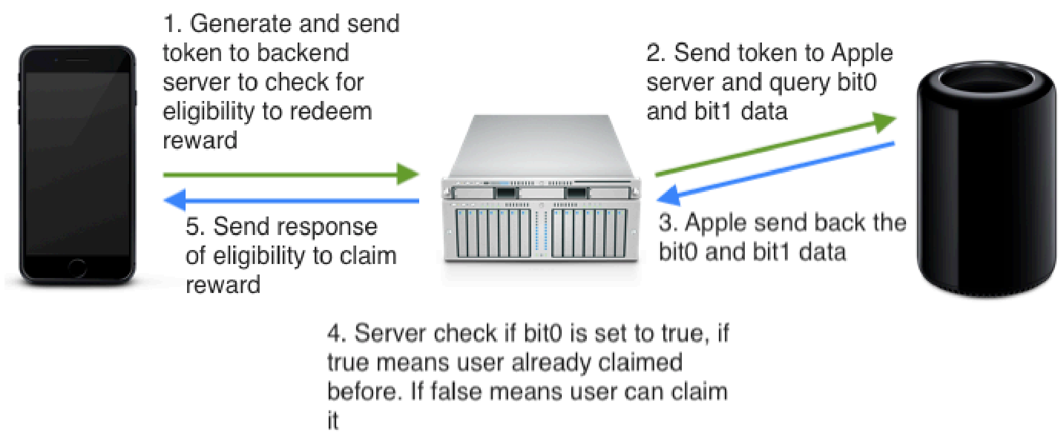
1. Generate and send token to backend server to check for eligibility to redeem reward
2. Send token to Apple server and query bit0 and bit1 data
3. Apple send back the bit0 and bit1 data
4. Server check if bit0 is set to true, if true means user already claimed before. If false means user can claim it
5. Send response of eligibility to claim reward

Diagram 2: (Source: Uniquely identify iOS device using DeviceCheck)

The DeviceCheck API generates an ephemeral token which uniquely identifies the device to Apple. Your API service combines this token with an authentication key from Apple and requests or updates the flags identified by the per-device bits and maintains a timestamp indicating when they were last modified.

The DeviceCheck implementation assures users of their privacy even while uniquely identifying the device. Developers receive only an ephemeral token, rather than a device serial number or any other identifying information. Similarly Apple stores the flags and timestamps but has no idea of how the flags are interpreted.

From an app authentication perspective, the flags could be used to identify compromised devices or apps, but it is still up to the developer to determine the app's authenticity. Since app and device conditions change frequently, the ability to store the last authenticity check is of only marginal value.

Apple App Attest

There are three high-level components involved in App Attest;

- Your iOS application
- The backend server of your application
- The App Attest service

Critically, the App Attest Service is an internet-exposed service, which may fail or timeout from time to time.

In summary, this is how it works:

1. Your server delivers a string of random bytes which represents a challenge to your app.
2. Your app creates a cryptographic key-pair using the Device Check API. The key resides in the device's "Secure Enclave" and the operation responds with a reference to that public/private key pair with an identifier string (the key ID string is a SHA256 hash of the public key).
3. A hash of the challenge data along with the key identifier is sent to the Apple App Attest service over the internet. The service responds with the attestation data as a string of bytes.
4. The attestation data is provided to your server, and the server validates the attestation data.
5. Subsequent requests to your server are then accompanied by assertion data which are generated on the device with the key identifier and the request body.
6. The assertion data is then validated and the request is processed.

Step 6 is actually very complicated and merits a 9-step explanation of its own in the Apple documentation. Ultimately, however, it lets you verify that the token you receive comes from your app on an Apple device.

Overview of Google Play Integrity API

You can call Play Integrity API to check that you're really interfacing with your genuine app binary, installed by Google Play, running on a genuine Android device. If something is off (for example it's a tampered or sideloaded app, or it's an unofficial emulator, or it's a rooted or compromised device), you can decide what defensive actions to take.

The Integrity API unifies Google Play anti-abuse features with a collection of integrity signals to help Android app and game developers detect potentially risky and fraudulent traffic. This traffic could come from modified versions of your app or game, untrustworthy devices, or other untrustworthy environments. By detecting this traffic, you can respond with appropriate action to reduce attacks and abuse such as fraud, cheating, and unauthorized access.

You can use the Play Integrity API to protect your apps and games from risky interactions. By identifying these interactions, your app can respond appropriately to reduce the risk of attacks and abuse.

The Integrity API unifies Google Play integrity signals to help app and game developers detect potentially risky and fraudulent traffic.

When a user performs an app or game-defined action, your server instructs the client-side code to invoke the Integrity API. The Google Play server returns an encrypted response with an integrity verdict about whether or not you can trust this device and its binary. Your app then forwards that response to your server for verification. Your server can decide what your app or game should do.

The API provides what is called an "integrity verdict" in a response that includes the following information:

- **Genuine app binary:** Determine whether you're interacting with your unmodified binary that Google Play recognizes.
- **Genuine Play install:** Determine whether the current user account is licensed, which means that the user installed or paid for your app or game on Google Play.
- **Genuine Android device:** Determine whether your app is running on a genuine Android device powered by Google Play services (or a genuine instance of Google Play Games for PC).

There are two types of requests supported by Google: "Classic" requests initiate a full assessment and require interpretation work on behalf of the user - these are recommended for the most sensitive requests, and can be slow. The new "standard" requests are faster but delegate some of the decision making to Google Play.

Overview of Huawei HarmonyOS Safety Detect

HMS Core Safety Detect Kit is intended to provide safety features which allow developers to focus on app development. It is a multi-feature security detection service which uses the Trusted Execution Environment (TEE) on Huawei phones.

Separately, if your app needs to be released to AppGallery, it must pass a series of validation tests which inspect for malware and other threats. Huawei uses signing information such as the digital certificate (in .cer format) and profile (in .p7b format) to ensure app integrity. certificate and release profile to sign the app before releasing it.

Currently, Safety Detect offers four features: SysIntegrity (system integrity check), UserDetect (fake user detection), AppsCheck (app security check), and URLCheck (malicious URL check).

SysIntegrity

SysIntegrity can check whether the user's device is rooted, unlocked, or has escalated privileges, and thus help developers evaluate whether to restrict their apps' behavior to avoid information leakage or financial loss of the user during user actions such as making an online payment or sending an email. The TEE allows apps with SysIntegrity integrated to be run in an isolated environment.

UserDetect

UserDetect can identify fake users based on screen touch and sensor behavior, as well as prevent batch registration, credential stuffing attacks, bonus hunting, and content crawlers through the use of CAPTCHA.

AppsCheck

AppsCheck is intended as anti virus protection. It can obtain a list of malicious apps on the user's device. The developer can then evaluate the risks and either warn the user about such risks or prompt the user to exit the app.

URLCheck

Intended to provide users with a secure Internet browsing experience, URLCheck allows apps to quickly check whether a URL that a user wants to visit is a malicious one.

There is a fifth API **Wifi Detect** which is intended to detect possible attacks from malicious Wi-Fi to your app but this is currently only available in mainland China.

You integrate the Core HMS SDK into your app and set up API calls in your app code for each of the services - **SysIntegrity, AppsCheck, URLCheck** and **UserDetect**. Your application interprets the returned result and decides what action to take.

Specific Shortcomings of Apple, Google and Huawei App and Device Attestation

Proprietary Solutions Tied to iOS, Android and HarmonyOS

These vendor solutions are intrinsically tied to the review processes of specific app stores, namely the Apple App Store, Google Play, and Huawei Gallery. The information used at runtime to check an app is created during the review process. Also, as we will see, all depend on key elements of the underlying OS (iOS, Android and HarmonyOS) and do not protect your apps if the OS has been tampered with.

Implementing diverging security solutions for Android, iOS and HarmonyOS does not align well with efforts to save development and maintenance costs by using cross-platform development tools such as Flutter and React Native.

Apple Device Check and App Attest	The Apple solution only works with Apple iOS devices and does not work with Android or other client/app ecosystems.
Google Play Integrity API	Google Safety Detect requires HMS Core Services
Huawei HarmonyOS Safety Detect	The Huawei solution only works with Huawei devices and does not work with iOS or other client/app ecosystems. Safety Detect features only work if the HarmonyOS SDK is installed in your app and requires the HarmonyOS TEE.
Approov	Approov covers all the devices that could be accessing your APIs, including iOS, WatchOS, Android and HarmonyOS. Approov also integrates easily with a number of cross-platform solutions including Flutter and React Native.

Limited Scope App Attestation

In the case of Apple, Google and Huawei, a key is generated during the app store review process which is used to validate the app. Unfortunately there appear to be ways to game the app attestation verdict at runtime via hooking and swizzling.

Apple Device Check and App Attest	There are a number of methods available to bypass App Attest for example this research.

Google Play Integrity API	App attestation is not an independent feature and is combined with checking the Google Play and Android device are genuine.
Huawei HarmonyOS Safety Detect	To be available on the Huawei AppGallery your app has to pass a 4 step validation process outlined here, which does issue a signature key for your app. Safety Detect SysIntegrity can return the SHA256 of the app package but it is not clear from the documentation how to use this to verify the app accessing your APIs is genuine since the SysIntegrity API is called by the app itself.
Approov	Approov provides a patented and unique remote attestation approach, where the running app must prove itself to be genuine through a sequence of integrity measurements. These results are then sent to the Approov cloud service using a patented challenge-response protocol, immune from replay attacks. The Approov service decides if an app is genuine or not.

Even Within the Ecosystem Not All Apps and Devices are Covered

Apple Device Check and App Attest	Not all devices can use the App Attest service, so it's important to have your app run a compatibility check before accessing the service. If the user's app doesn't pass the compatibility check, you are advised to gracefully bypass the service. In addition most app extensions don't support App Attest. The method to generate a key fails when you call it from an app extension, regardless of the value of `isSupported`. The only app extensions that support App Attest are watchOS extensions in watchOS 9 or later.
Google Play Integrity API	Like the SafetyNet APIs, the Play Integrity API is offered as part of Google Mobile Services and thus is not available on free Android environments. Therefore, apps that use the API may refuse to execute on AOSP builds and non-GMS apps are not protected.
Huawei HarmonyOS Safety Detect	Requires the HarmonyOS HMS Core Services and the TEE.
Approov	Approov works even with Jailbroken, rooted or manipulated iOS, Android and HarmonyOS client environments and gives detailed, highly granular visibility of what is going on in the client and equally granular policy enforcement to define what is acceptable or not. It also works on Apple WatchOS.

Limited Client Integrity Checks

While the limited app attestation offered by Apple, Google and Huawei can, to some degree, verify an app's authenticity, this doesn't guarantee that it's running on a genuine, unmodified device. Without taking steps to monitor the client environment, sophisticated attackers can install software to modify app behavior or potentially clone a device's identifiers and bypass app attestation or manipulate app attestation results.

Unfortunately, a high proportion of genuine users have rooted their phones in order to add features and capabilities so the dependence of these solutions on a root check to determine the 'goodness' of the device is problematic. In some vertical markets, such as financial services and healthcare, it is understandable and acceptable to block API access for rooted devices. For more general retail sectors, customer stickiness is a key metric.

Apple Device Check and App Attest

App Attest focuses specifically on verifying the integrity of the app itself. It doesn't address other security risks like malware or runtime attacks. App Attest doesn't detect jailbroken devices or runtime app manipulation via hooking or swizzling.

Google Play Integrity API

An environment is defined as problematic by Google, not by the app developer - Only integrity levels such as Virtual, Basic, Strong are reported back by the Integrity API in addition to some optional additional information about the state of the Google Play Environment. Some researchers have found that Google does not find all problematic scenarios in the client environment and the lack of granularity and visibility over what is checked could be an issue. Google Play API does detect if the phone is rooted, but other results are less reliable with a rooted device.

Huawei HarmonyOS Safety Detect

A client environment is defined as problematic by Huawei, not by the app developer; only the following integrity check results are reported:

unlocked: indicates that the device is unlocked

root: indicates that the device is rooted

emulator: an emulator is in use

attack: indicates that the device is attacked.

As with Google PlayIntegrity, details of the inner workings are not disclosed and information on how to interpret the results is limited. This lack of granularity and visibility over what is checked could be an issue.

Approov

Approov attests the actual executing code, while the others are restricted to session-based verification. This means Approov provides more robust protection against runtime tampering or compromises like hooking/function swizzling. It also protects against dynamic instrumentation tools and jailbroken devices. Approov provides a rich set of device attestation checks which are regularly updated as new threats emerge. Rooted and jailbroken phones are detected. Frameworks and hooking environments such as Cycript, Cydia, Xposed, Frida, Magisk, Zygisk are all detected. What is acceptable can be controlled with a high level of granularity via over the air policy updates.

Apple, Google and Huawei Provide Limited Analytics

Apple Device Check and App Attest

The only usage data you can get from App Attest according to Apple is "an approximate count of unique attestations for your app on a particular device. A count that's higher than expected might be an indication of a compromised device that's serving multiple compromised instances of your app. You can use this information to assess your risk". App Attest doesn't provide information about user behavior or device usage patterns, which could be valuable for detecting fraudulent activity or risk assessment.

Google Play Integrity API

The Play Console provides a report on the volume of requests your app makes daily to the Integrity API. The API includes a feature to detect large-scale abuse by checking how many attestations were generated by your app on a device in the last hour. This can help identify devices requesting large numbers of integrity tokens. While these features provide valuable insights, the reporting is focused on API usage, service status, and aggregate data rather than real-time monitoring of individual app or device activities.

Huawei HarmonyOS Safety Detect	There is no information available about runtime reporting and analytics capabilities of HarmonyOS Safety Detect. It doesn't appear to provide extensive runtime reporting or real-time analytics of device and app activities.
Approov	Approov analytics on the other hand, shows what is happening in your service, with real-time and historical data. Information is presented via a dashboard presenting a top level view and then allowing deeper investigation via the various graphs and options which are available. This is important not only to measure and report on the effectiveness of the solution but also to help secure and maintain regulatory compliance.

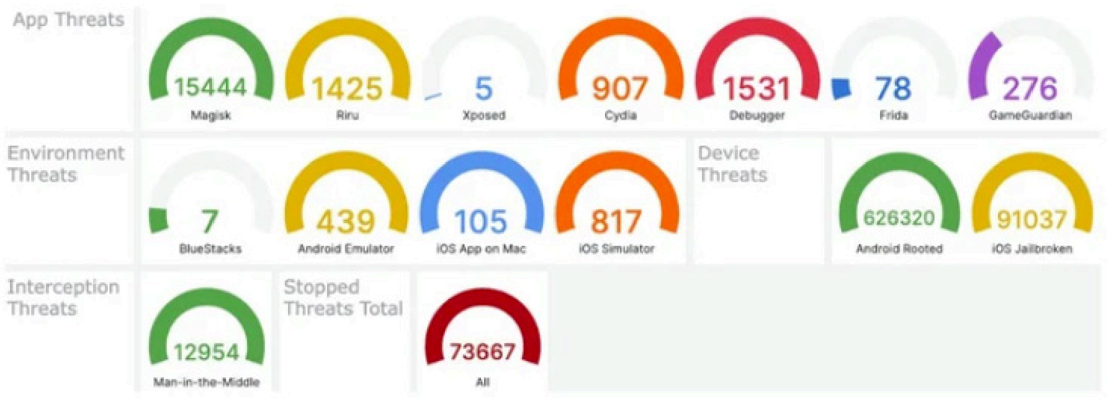

Diagram 3: Approov Real-time Threat Reporting (Source: approov.io)

Implementation Challenges

There have been a number of reports by developers of finding these solutions difficult to implement.

Apple Device Check and App Attest	Developers have raised concerns about how difficult App Attest is to implement properly and the solution also relies heavily on trusting developers to code more defensively. Each network call needs to be implemented using a complex list of sequential framework functions. The backend validation process is extremely complicated and integrating App Attest requires careful server-side implementation to handle key validation, challenge responses, and risk assessment. Apple documentation lists a 9 step process to determine the integrity of the attestation object at the backend and picking apart the binary data structure which is used is not straightforward.
Google Play Integrity API	The Implementation of Play Integrity requires app developers to defend API calls at a function level, meaning nothing is protected out of the box. Developers need to review/audit all of the API call points and make modifications to many of them. Each API request that should be secured, needs to be secured explicitly, which is achieved by using specific play integrity framework methods.
Huawei HarmonyOS Safety Detect	The Implementation of Safety Detect also requires app developers to defend API calls at a function level. Developers need to review/audit all of the API call points and make modifications to many of them. Each API request that should be secured, needs to be secured explicitly.

Approov	Approov makes life easy for developers. Once the SDK is integrated using one of our many quickstarts, app attestation works without any code changes in the app. Approov supports and integrates seamlessly into all modern app development frameworks including Flutter, React-Native, Ionic, as well a native iOS and Android. Check out our comprehensive list of quickstarts. The Approov SDK can be tailored to any framework. At the backend, integration is easy also - a simple check of a standard JWT token is required and again there are quickstarts for all major frameworks and environments.

Apple, Google and Huawei Do Not Stop API Secrets From Being Stolen and Abused

All mobile apps access primary and third party APIs and the keys used to validate requests to these APIs must be protected. Unfortunately, common mobile apps still expose secrets in their code and keys can be acquired by hackers from cloud repositories or even MitM attacks.

Apple Device Check and App Attest	Apple App Attest can help you distinguish scripts (using stolen secrets) from genuine apps but it does nothing to get secrets out of your app code. It also does not provide any help in keeping your apps running when secrets are compromised, e.g. allowing you to dynamically rotate stolen API keys.
Google Play Integrity API	The Integrity API can help you distinguish scripts (using stolen secrets) from genuine apps but it does nothing to get secrets out of your app code. It also does not provide any help in keeping your apps running when secrets are compromised, e.g. allowing you to dynamically rotate stolen API keys. Google does have a secret manager but it doesn't work with Play API to test the app integrity checks offered by Google Play Integrity API before delivering secrets to an app.
Huawei HarmonyOS Safety Detect	Huawei HarmonyOS currently does nothing to get secrets out of your app code. It also does not provide any help in keeping your apps running when secrets are compromised, e.g. allowing you to dynamically rotate stolen API keys. Huawei does separately offer a cloud secrets management service but it doesn't work with HarmonyOS to validate integrity checks before delivering secrets to an app.
Approov	Approov provides a secrets management solution that manages API keys and certificates securely in the cloud, delivering them "just-in-time" only when app and device integrity checks are passed. It also allows them to be easily rotated via over-the-air updates if they are compromised elsewhere. It also works for third party APIs.

Apple, Google and Huawei Do Not Prevent Man-in-the-Middle (MitM) Attacks

Mobile phones are particularly prone to Man-in-the-Middle attacks on the channel between the app and the API, even if the traffic is encrypted. This is because hackers can install tools like MitMproxy on the device.

Apple Device Check and App Attest	App Attest, when it is working, does ensure the attestation data is not modified in transit, if the phone is not jailbroken. However as we have indicated App Attest doesn't work with jailbroken phones or detect runtime app manipulation which are the paths attackers take to intercept traffic.

Google Play Integrity API	Google Play Integrity API does not prevent these attacks.
Huawei HarmonyOS Safety Detect	HarmonyOS Safety Detect does not prevent this type of attack.
Approov	Approov Dynamic Certificate Pinning protects the channel from mobile Man-in-the-Middle attacks and makes it easy to manage and rotate certificates over the air.

Apple, Google and Huawei Performance and Rate Limits

Implementing comprehensive security checks can introduce performance overhead, potentially affecting app performance and user experience, leaving services prone to DDoS attacks. Developers need to balance security with usability to maintain optimal app functionality. Google and Apple both provide information on the limitations on the number and frequency of integrity checks which can be performed but Huawei currently does not discuss this topic.

Apple Device Check and App Attest	Apple may throttle App Attest requests to prevent server overload, which could impact app functionality during high usage periods. Apple enforces a rate limit to the number of unique devices calling App Attest at any given moment, and much to the frustration of many app vendors the actual quota is not publicly available. In addition to this, once the quota is met, Apple offers you no SLA to increase it, which makes it very difficult to scale securely.
Google Play Integrity API	Developers have reported performance issues with Google Play API. Google has rolled out new standard requests which have lower latency (a few hundred milliseconds on average) vs. the "classic" requests which took on average several seconds. Unfortunately the classic requests are still recommended by Google for the "most sensitive" requests your app makes. Google currently offers a quota of 10,000 API integrity checks per day, if this quota is exceeded the service will no longer work for your app. Google offers a quota increase on a case by case basis but in these circumstances a form needs to be submitted in which a review will take up to 2 - 3 working days. In the meantime a DDoS (Distributed Denial of Service) attack could take down the service running on your app.
Huawei HarmonyOS Safety Detect	Huawei currently does not discuss this topic.
Approov	Approov has no quotas or thresholds on traffic and can easily scale to support millions of active mobile apps, always providing a consistently high performance. Allowing you to seamlessly and securely scale your apps is at the core of Approov, we'll provide you with a competitive SLA to suit your business, and you'll never get charged for attacks, including DDOS mitigation. See our pricing guidelines for more information.

Regional Restrictions

This is a particular issue with the current implementation of Huawei HarmonyOS Safety Detect.

Huawei HarmonyOS Safety Detect	Some features of Huawei HarmonyOS Safety Detect are restricted or less effective in certain regions due to regulatory and compatibility issues. Some features are implemented differently in different regions, for example in mainland China. This can impact the global applicability of the security measures provided as well as making life complicated for developers.
Approov	Approov is deployed worldwide and works consistently, including in mainland China, while also conforming with local regulations.

The Path Forward For Developers - How to Ensure Complete Protection for Mobile Apps and APIs

Proprietary attestation solutions from Apple, Google and Huawei provide limited security in very restrictive use-cases. The limitations in ease of integration, regional applicability, dependency on proprietary infrastructure, performance impacts, and occasional detection inaccuracies must be carefully considered by developers and security teams seeking to deploy robust but cost effective security for mobile apps and APIs.

On the other hand, Approov Mobile App Protection ensures that all mobile API traffic does indeed come from a genuine and untampered mobile app, running in a safe environment. Doing this blocks all scripts, bots and modified or repackaged mobile apps from abusing an API. Approov supports any apps running on Android, iOS, WatchOS and HarmonyOS, providing comprehensive and powerful security with easy and consistent management across all supported platforms. In addition Approov provides a unified way of implementing protection - only one single simple check is required at the backend and this is always consistently the same, irrespective of whether the originating device is running iOS, Android or HarmonyOS.

Approov continuously updates its detection capabilities to adapt to new threat vectors, thereby ensuring ongoing protection against new threats.

Compared with the three proprietary solutions we have evaluated, Approov provides a more robust dynamic attestation method which is combined with robust cryptographic methods and dynamic API key protection. This is done in a way that is independent of how apps are developed or deployed.

If however, there is a requirement to continue to use one of the three solutions, for example as an intermediate step in a migration plan, Approov can seamlessly integrate with the proprietary checking mechanisms of each solution. In this way, Approov can use all the information available and always provide a unified result.

Our recommendation is that enterprise teams who view the mobility ecosystem as being essential to their mission should consider Approov as a way to achieve greatly improved mobile security for devices, apps, and the supporting infrastructure.

Contact us for a free technical consultation - our security experts will show you how to protect your revenue and business data by deploying Approov Mobile Security

approov