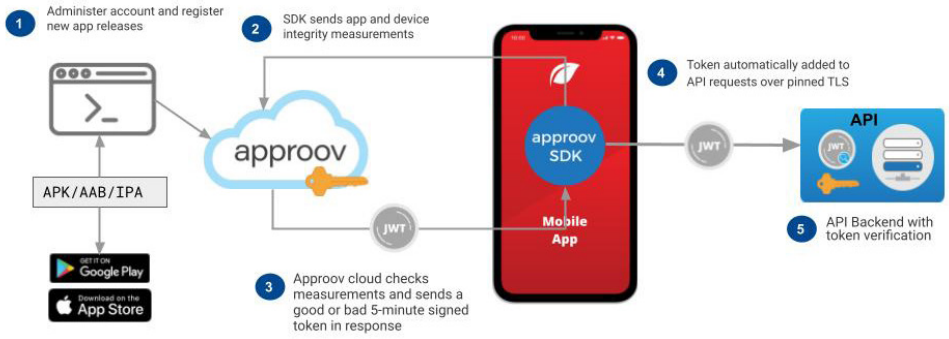
approov
MOBILE APP PROTECTION

# Approov Mobile App Protection & Apple DeviceCheck: Comparisons and Benefits of Integration

*This overview describes the Approov Mobile App Protection and Apple DeviceCheck technologies and how they work together to provide even better protection for your applications and the APIs they use.*

## Approov Mobile App Protection

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 consists of an SDK which is integrated into the customers' mobile app, and a cloud service which manages a dynamic attestation process on the app. Only positively attested app instances are provided with a valid, short lived, and cryptographically signed Approov token. This token can then be used to gain access to backend API services, ensuring only legitimate API requests are able to do so successfully. Since an industry standard JSON Web Token (JWT) is used, the check can be easily integrated into existing backend systems, either in a cloud service or at the network perimeter in an API gateway, CDN, WAF or similar.

The attestation process proves the integrity of the app, ensuring tampered, cloned or re-packaged versions fail. Further protection is provided against man-in-the-middle attacks, against apps that are running on rooted/jailbroken phones or an emulator, under the control of a debugger or in the presence of instrumentation or code hooking frameworks. The attestation process used by Approov is invisible to the end user and does not significantly impact app performance.

Approov customers are able to make fine grain choices about which attributes are allowable for their running apps, adaptable to particular end user markets and conditions. Metrics are gathered live from devices and made available to Approov customers to help them understand their user base and create good security access policies.

Approov protects against a very wide range of threats by its attestation approach, establishing the integrity of the requests being made. This blocks automated bots and scripts which might attempt to gain access to backend resources by spoofing API traffic as if it was coming from a real app. Such traffic will usually employ valid user credentials so user authentication

mechanisms offer no protection. Indeed such approaches are typically used by attackers for illicit automated account takeover or data scraping activities. Approov also protects against unauthorized access via modified or fake repackaged apps that may be subverting business models, redirecting ad revenue and putting end users at risk.

## Apple DeviceCheck

The Apple DeviceCheck solution comprises two technologies, a *Device Identification* technology introduced in iOS 11 and the more recent *App Attest* technology introduced in iOS 14.

*Device Identification* keeps a record of a physical mobile device without the need to track any personal information about the device or its user. Developers are able to set a two-bit status code for any device, have it recorded by Apple and then retrieve it as necessary. In this way, the history of the device can be retrieved in an anonymous way and business logic can implement agreed on policies based on the status. This useful functionality was designed for specific purposes which Apple articulated during the 2017 WWDC event, where the DeviceCheck API was announced in the context of user privacy. A key use case is to ensure that an individual user is not able to benefit from a free offer multiple times by reinstalling the app on the same device. A side benefit of the technology is that it involves the iOS device generating a token that can be checked with an Apple API. Independently of the two bit status, this shows that the token was generated on a real Apple device.

It is important to note that the responsibility for identification of the status of the device, (i.e. has it already redeemed a reward, has it shown abusive behavior, has it been guilty of fraud, etc.) is with the developer. Once the developer knows and has marked the status of the device, Apple will store and retrieve it in a way that does not require the implementation of any developer functionality to track device data. That is certainly helpful, but developers should not underestimate the additional effort required to identify abusive devices — in particular those which engage in fraud.

It should also be noted that *Device Identification* does not provide any additional support for proving that the device tokens are not being intercepted by a Man-in-the-Middle. Any such interception would then allow an attacker to spoof requests as if they were being made from a DeviceCheck protected device.

The more recent *App Attest* technology allows the integrity of an app running on a device to be assessed. It allows the establishment of a hardware-based cryptographic key, that is attested with Apple to verify it is from a valid instance of the app. This key can then be used to sign subsequent requests for sensitive or premium content and therefore to assert their legitimacy as being sent from a genuine app.

The DeviceCheck technology cannot provide reliable information if a particular iOS device has been jailbroken. Many of our Approov customers do allow execution on jailbroken devices, and rely on Approov to definitively tell them about the integrity of their running app, and whether it appears to be under specific attack, hooking or modification. They have no interest in the integrity of the device as a whole.

Since *App Attest* is only available since iOS 14, there are a significant number of devices that are running older iOS versions. Even the *Device Identification* technology, available from iOS 11, is not universally supported on all devices since some do not have the necessary secure enclave. Furthermore, 32-bit iOS devices cannot be upgraded beyond iOS 10 and there may be a need to support them. Thus a client side check is required for DeviceCheck availability with a bypass if it is not. Unfortunately, such a bypass creates a fundamental security weakness since this can be exploited by attackers on any device by faking the check and thus undermining the entire security model.

In common with Approov, the result of *App Attest* is a cryptographically signed attestation. The server side is responsible for providing a one time nonce value to initiate this process and to verify the result using custom server side logic. This is described in the Apple documentation, which notes that a number of different special cases have to be dealt with in the implementation. Thus it is not possible to implement the verification at the network perimeter in an API gateway, WAF, CDN or similar due to this complexity. This also makes integration into managed backend systems more challenging.

There is a limit to the number of DeviceCheck calls allowed from the device itself, and to Apple. They warn that "*Apple servers might throttle attestation traffic from a particular app to avoid becoming overwhelmed if too many instances of your app make this call simultaneously*". Careful onboarding and rate limiting of requests needs to be implemented.

## DeviceCheck and Approov Together

Approov provides an option for integration with DeviceCheck, both for the Device Identification mechanism and also for App Attest. This leverages all of the benefits of these technologies, whilst retaining the ease of integration of Approov.

To add *Device Identification* protection, all that is required is to use the Approov CLI to provide the DeviceCheck key

so that it is able to perform bit status lookups and check the validity of the device tokens. With this in place all iOS devices perform the integrity check in addition to all of the Approov protections. If the device token is rejected by Apple then no valid Approov token is issued. Furthermore it is possible to ban a particular Approov device ID and this causes the Apple state to be updated on the next request from the device, permanently banning the device even if the app is uninstalled and then reinstalled to provide a different device ID. An optional automatic banning mechanism is also provided whereby the number of installations of the app on a device can be counted using the two bits, allowing the physical device to be banned after four new installations in a month on the same device, which is likely to be indicative of some fraudulent behaviour.

To add *App Attest* protection, all that is required is the Team ID for the app and for the Apple required entitlement to be added to the app. Approov then automatically performs an attestation operation when the app is first launched. Since this is an expensive operation this state is retained in a server encrypted token bound to the particular device. This avoids the need to perform a full attestation on every Approov token fetch. Instead a cheaper assertion can be performed on every Approov token fetch (or at a configurable interval). Approov manages the creation of challenge data to prevent replay attacks, and all of the complex server side verification steps required. The result is an Approov token, a simple JWT with frontend integrations for adding these automatically to requests, whilst dynamically pinning connections, and numerous backend integrations for easy verification token verification.

The *App Attest* verification checks that the calling app is the expected one, matching the Team ID provided and Bundle ID that is added when performing a normal Approov app registration. This provides an additional level of security over the app attestation performed by Approov itself.

If a DeviceCheck key is provided via the Approov CLI, then a fraud lookup can also be performed on each *App Attest*. This obtains a fraud metric associated with the physical device, which measures how many keys have been generated for it associated with the Team ID. A maximum risk metric may be configured to prevent any device with an elevated fraud risk from obtaining a valid Approov token.

If desired you can force Approov to always require DeviceCheck or to allow it to work gracefully on iOS devices that do not support it. This is particularly important for *App Attest* which is only available from iOS 14.

## Summary

It is more important than ever for businesses to decide how to create a comprehensive security policy which will maximize business revenue while minimizing customer friction. The best approach is to implement layers of security, utilizing a range of approaches in order to thwart all the key attack vectors in the mobile domain.

Apple DeviceCheck is a useful addition in the armory of weapons which are available to to thwart API attacks from iOS apps or app impersonators. DeviceCheck should not be relied upon to solely judge the validity of a platform, as acknowledged in Apple's own documentation:

*"No single policy can eliminate all fraud. For example, App Attest can't definitively pinpoint a device with a compromised operating system. Instead, the DeviceCheck services provide information that you can integrate into an overall risk assessment for a given device."*

When using DeviceCheck alone, businesses may still be open to attacks performed on jailbroken devices. Thus they can benefit from the other signals that Approov provides about the compromised nature of the environment that their app is running within.

Indeed, a better approach is to create a more comprehensive security protocol that considers a number of detailed factors before deciding whether or not to allow a transaction. Additionally, DeviceCheck presents a number of practical challenges during implementation, such as the latency of the attestation process which makes it impractical to protect all requests without considerable additional implementation complexity.

Approov, together with DeviceCheck, provides a better solution. By delivering the server side operations as a service, providing finer grain detail and checking on the health of the actual app that is running, the full iOS ecosystem can be supported and secured with minimum integration effort. Furthermore, since Approov also supports Android a unified approach can be taken across both of these major platforms.