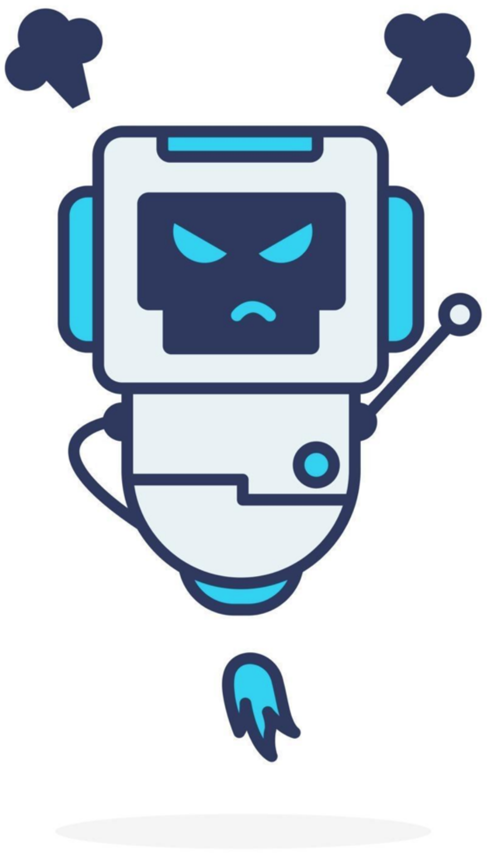

Addressing the Security Trust Gap in a Mobile + API World

Whitepaper

by

Haltdos

Current security and anti-fraud measures do not adequately address the needs of the mobile app world. Sensitive data that is being shared through APIs are still subject to exploits such as app impersonation, reverse engineering of API protocols, spoofing transactions and using bots and emulators to access backend API servers. Haltdos Secure Mobile App Protection creates a trusted environment that protects your APIs and your business by providing additional authentication — authenticating app instances, not users. By ensuring that the mobile app connecting through an API is a genuine, untampered instance, fraudulent transactions, malicious scripts, and bot attacks are blocked at the source.

Traditional Measures Don't Provide Enough Protection

Mobile apps have become the digital touchpoint of choice, overtaking web browser usage. This mobile app first, engagement model and the increased sophistication of the hackers opens up new vectors for security risk. Many security approaches originally developed for the desktop web channel are not sufficient for the mobile API economy. With web browsers, the client side platform is inherently insecure since code must be provided in the clear and run in an untrusted environment. Therefore, security sensitive logic has always been run in the web backend. However, mobile app users expect a responsive, frictionless experience so there has been a migration of business logic and security sensitive code to the apps themselves. This has resulted in a corresponding rise in the complexity and sensitivity of the data flow between the mobile app and the APIs of supporting backend servers.

These trends have caused a concerning trust gap to emerge in the mobile app world. End-to-end encryption provides little security when one of the ends may have been compromised. A significant and growing vulnerability is the ability of criminals and fraudsters to reverse engineer API protocols and then spoof transactions as if they had been generated by a genuine mobile app. This can be achieved by tampering with the app code itself or by engineering new applications that impersonate the real app. Static credentials, encryption keys, API keys or other secrets embedded inside the app can be discovered through reverse engineering. Bad actors can then leverage them, often alongside other stolen credentials, to gain access to sensitive digital assets or disrupt normal operations by scaling malicious access using botnets and mobile emulators hosted in the cloud.

Communication traffic from a remote client to a server should not be trusted to emanate from a genuine mobile app, even if it presents its credentials and initial behaviour as such. It needs to be verified as genuine. Protection focused only on the app provides little protection against this type of exploit. If app hardening or anti-tamper approaches are breached, the server doesn't know about it and is powerless to block the bad traffic. A dynamic authentication protocol is needed to check the veracity of the client software itself and then communicate this status live to the server in an un-spoof able way — closing the very real trust gap in today’s mobile first world.

Current Approaches are not Sufficient

App security efforts have been focused on signature-based behavioural approaches and anti-tampering solutions applied to the app code. The use of Transport Level Security (TLS) to encrypt communications between the mobile app and the server is also a common security measure which does help prevent trivial tampering and eavesdropping of the data. Certificate pinning in the app allows a mobile app to trust that it is communicating with a genuine server without a Man-in-the-Middle (MitM) eavesdropping attack. However, use of this one-way TLS does not provide a server with authentication that the client software it is communicating with is genuine. Attacker scripts are able to launch TLS sessions with the server even through encrypted channels.

Signature-based Approaches

Signature-based approaches rely on capturing large amounts of data and then applying rules based on known patterns. This attempts to differentiate between real data traffic from apps as opposed to bots or scripts attempting to spoof the traffic. The failing in this approach is its inability to prevent new styles of attack for which signatures have not been captured. Constant manual analysis and maintenance is required to keep up to date with the latest attack vectors. Moreover, certain attacks performed at a low velocity are intrinsically very difficult to distinguish from genuine traffic. In other words, this is a negative security model that enables traffic by default and attempts to detect irregular or unusual usage. A better solution is a positive security model that provides definitive authentication of the traffic up-front and at source.

The situation becomes complex if the same APIs are being used by Web client as well as mobile client. With high degree of false positives, this approach leaves many genuine interactions blocked and bot attacks unmitigated.

Anti-Tampering Efforts

Anti-tamper techniques may be employed to make it more difficult to tamper with or reverse engineer the operation of a mobile app. The use of anti-tamper with code guards does harden the app, but cannot prevent an attacker impersonating the traffic of a real app, especially if they have been able to mount MitM analysis of the traffic being communicated. Current anti-tamper technologies can also be difficult to integrate into existing app code, can be quite invasive in the development process and can cause measurable performance degradation. Often the use of these technologies is somewhat misdirected. The digital assets of value are on the server, not in the app, so the focus should surely be on ensuring that only a genuine untampered app can access those assets by allowing the app to prove its authenticity. Trying to prevent the tampering is insufficient because if it succeeds, or if a system is developed that can spoof the app communication, it is undetectable by the server.

Transport Level Security (TLS)

Mutual TLS attempts to establish two-way trust between the client mobile app and the server. This is achieved by installing a custom certificate (with a private key) on the mobile device itself. Only clients with access to the client side certificate are able to successfully initiate the mutual TLS session. Unfortunately, this approach has the same drawback as other attempts to conceal secret credentials inside the app. The app is subject to analysis and reverse engineering by attackers who are able to extract the certificate from the mobile app and then embed it into a malicious application that can successfully initiate a mutual TLS session.

API Keys

Existing approaches embed security credentials, such as an API key, into the app itself and these are relatively simple to reverse engineer by an attacker. In many cases, no attempt is made to conceal the key and widely available open source analysis tools can easily extract it. Once the key is available, it can be reused in a malicious application or script to gain access to the API. In some cases the API key or access credentials can be trivially extracted by using a MITM proxy software suite to observe the communication between the mobile app and the server. More sophisticated implementations never communicate the key directly, but use an HMAC implementation (message authentication code involving a cryptographic hash function) on the client side to show that the key is known. However since a static key still exists, it is still subject to reverse engineering. Server side mitigations may rate-limit the number of requests per second that use the same API key, but this does not fundamentally stop data exfiltration or other attacks – it simply slows down the rate at which it can happen. Hackers may probe the API endpoints to find vulnerabilities in your security logic, or to find potential code injection vulnerabilities, and this may be even more serious in terms of gaining access to your network and customer data.

Mobile apps typically access digital assets on your servers using APIs. The backend API servers may have access to sensitive corporate information and it is critical that appropriate security measures are put in place to keep them secure.

How Haltdos Mobile App Protection Works

The secret to create a secure API channel is for genuine apps to be able to identify themselves to backend servers, i.e. a positive trust model to authenticate genuine apps. In other words, the server needs to be able to trust that it is communicating with a true client mobile app, rather than with something else that is trying to impersonate typical app communication.

The Haltdos service uses a challenge-response cryptographic protocol allowing a server to establish the veracity of a connecting client app. This approach is not dependent upon any secret embedded inside the app and thus fundamentally different to other solutions. The integrity of the app is dynamically measured to establish what it is, not what it has in the form of a static secret. Haltdos authenticates app instances, not users.

App authentication provides an additional layer of protection that can be used alongside your existing authentication and authorization methods. By integrating the Haltdos SDK into your mobile app, the API requests it generates can include a special token in the header. A simple change in your backend API endpoint will check the presence of valid tokens for each API request and any requests not containing a valid token are simply rejected. With this protection in place a hacker can't launch scripts or individual probing requests against your API. All their requests will be immediately rejected.

Haltdos WAF solution that handles requests from apps and determines the authenticity of the app. It does this using a challenge-response protocol, built upon established and trusted underlying cryptography, which ensures a live interaction and prevents any attempts to replay a previous response.

The simple SDK library is embedded into the app itself. This is easily added to the app development project and support is provided for a wide range of different app development frameworks. The SDK provides a method to obtain an Haltdos token that can be added to the headers of API requests that are to be protected. Behind the scenes, the SDK automatically connects to Haltdos WAF solution when necessary and performs the integrity check. If the integrity check passes then the app is issued with a time limited token by the cloud service. The integrity process only needs to be repeated if the token is about to expire.

By adding Haltdos to your security architecture, you can be sure that mobile app instances in the wild will be authenticated, on top of your regular user authentication and connection encryption, further reducing risks of application layer attacks. The Haltdos positive authentication model allows genuine app requests to go through while allowing you to focus on actual threats. Untrusted soft- ware agents, such as attacker scripts or modified apps, are unable to generate valid tokens and are immediately rejected. Since Haltdos does not rely on hiding a secret, such as a static API key, traditional reverse engineering techniques used by attackers are ineffective

SDK and Tokens

Haltdos is based on the concept of software attestation. It allows your apps to uniquely identify themselves as the genuine, untampered software images you originally published. In exchange for this proof the app is granted a token which can then be presented to your API with each request. Your server side implementation can then differentiate between requests from known apps, which will contain a valid token, and requests from unknown sources. Haltdos does not interfere with any of the other traffic between the client mobile app and the server. The tokens are very quick to validate, ensuring minimal impact on the latency of API requests made by the app.

The developer interface to the Haltdos SDK is a method to obtain a token. These tokens have a short lifespan of a few minutes in order to mitigate against any potential of theft. If there is currently no valid token then the SDK initiates the process of communicating with Haltdos WAF to perform the integrity check to obtain a new token. Subsequent calls to the method made while there is still a valid token returns the cached version, resulting in very little overhead.

When the SDK requires a new token it begins the attestation process by requesting a random value (nonce) from the Haltdos WAF which it uses to seed the signature hash of the integrity check. Running heavily obfuscated and defended code, the integrity of the Haltdos SDK itself is measured along with the rest of the app content. The combined cryptographic hash is sensitive to any change in the app or the code used to measure it. Since the hash is seeded with the nonce value it will always be unique and cannot be replayed by an attacker. This resulting dynamic app signature is sent to the Haltdos WAF as part of a token request along with other information about the app and the device it is running on.

Since the Haltdos WAF knows the set of registered good apps it is able to perform the same calculation as the Haltdos SDK. If the dynamic app signature reported by the SDK is consistent with a registered app and the other security checks pass, then the issued token will be signed with a secret associated with the customer account. If not, the token is still issued but is not signed correctly. Because the app itself does not know the secret it does not know whether it passed or failed the integrity check making it difficult for anyone to reverse engineer the protocol.

Adding Haltdos to your iOS or Android app is a straight-forward process and can be easily integrated into most development flows.

Once registered, the app can be published as normal. Since Haltdos checks for changes in the app’s signature to detect app repackaging, it is important to repeat the registration process every time you release a new version of the app.

Solution Benefits

Haltdos Mobile App Protection has been designed for minimal impact on quality of service, user experience, and service load.

Low Overhead

The integrity checking process built into the Haltdos SDK is designed to be efficient and highly tamper resistant.

When an authenticity check is needed, it takes less than 100ms of compute time on current mobile devices, therefore not impacting battery life. Each authenticity check requires two endpoint transactions to Haltdos WAF servers. The messages and their responses have been optimized for length so that the impact on both battery life and data usage is minimized.

Ease of Integration

Haltdos can easily integrate into new or existing apps. The Haltdos SDK provides a simple interface for obtaining a token that can be added to the headers of API requests without the developer needing to know details of how the app integrity measurement is done. The entire integration process shouldn't take more than a few hours.

Anti-Automation

APIs are bedrock for Mobile Apps. These APIs are also consumed by other legitimate clients such as WebApp, Thin Clients, etc. As all the business logic resides in the API backend system, the APIs are constantly under various types of automated attack. Techniques like captcha and behavioural analysis help prevent automated attacks for WebApp. However, the same is not possible for Mobile Apps. Employing mobile app specific rules and behaviours can result in false positives and affect the functioning and utility of mobile apps and often frustration for legitimate users.

Haltdos provides a trusted method for apps to positively identify themselves to your API allowing better traffic filtration to backend servers. Valid apps which are registered with Haltdos WAF are dynamically issued a short-lived token which is then sent with each request to your API. Traffic with valid tokens is from known apps and can be prioritized.

App Legitimacy

In some cases, an app used to access an online service may be tampered with in some way. This might be in the form of a one-off modification by an individual hacker or a tampered app being repackaged and distributed to many users. In the latter case there is the possibility that a large number of your app users are accessing your service using an unofficial client mobile app over which you have no control. Any secrets or credentials embedded inside the app will have been compromised and are being used to access the online service. Moreover, these will be the same secrets or credentials used by the official version of the app. So if you revoke these credentials to block access to the tampered app, then you will also block your legitimate users. Even if you manage this transition smoothly, the updated secrets are likely to be stolen in exactly the same way again.

Some of these threats are targeted at the owners of mobile devices. Many apps depend on ad revenue as an income stream and fake apps have a big revenue impact. More indirect effects can be on a company’s reputation if user credentials are stolen or the perception is that the developers write unreliable, ad-infested or resource hungry apps. Fake apps might also gain access to web services accessed by the genuine app, such as analytics, usage information or scoring for online games. This data then becomes polluted and less usable for real app users.

Haltdos Mobile App Protection is a way to prevent successful repackaging of apps which use web services to provide some of their capabilities. In a process analogous to user authentication, the Haltdos Mobile SDK integrates with the app and provides a mechanism to verify the authenticity of the code being used to access an API. By positively identifying traffic from genuine apps, attempts to use the API from repackaged apps or other unofficial clients can be blocked. Fake or tampered apps are simply unable to access any of the features provided by the app servers.

Conclusion

The world is changing and traditional security measures are no longer sufficient to protect mobile API connections because they still are vulnerable to exploits such as Man-in-the-Middle attacks, reverse engineering, API key extraction, etc. They only focus on stopping an attack already in process.

Haltdos Mobile App Protection starts where the request originates - at the app.

Working with other security protocols, Haltdos helps build a true end-to-end API environment that can be trusted. By identifying, verifying and certifying that only your mobile apps, running in untampered environments and communicating over secured channels, can access your APIs and cloud services, your valuable and sensitive assets stay safe and secure.