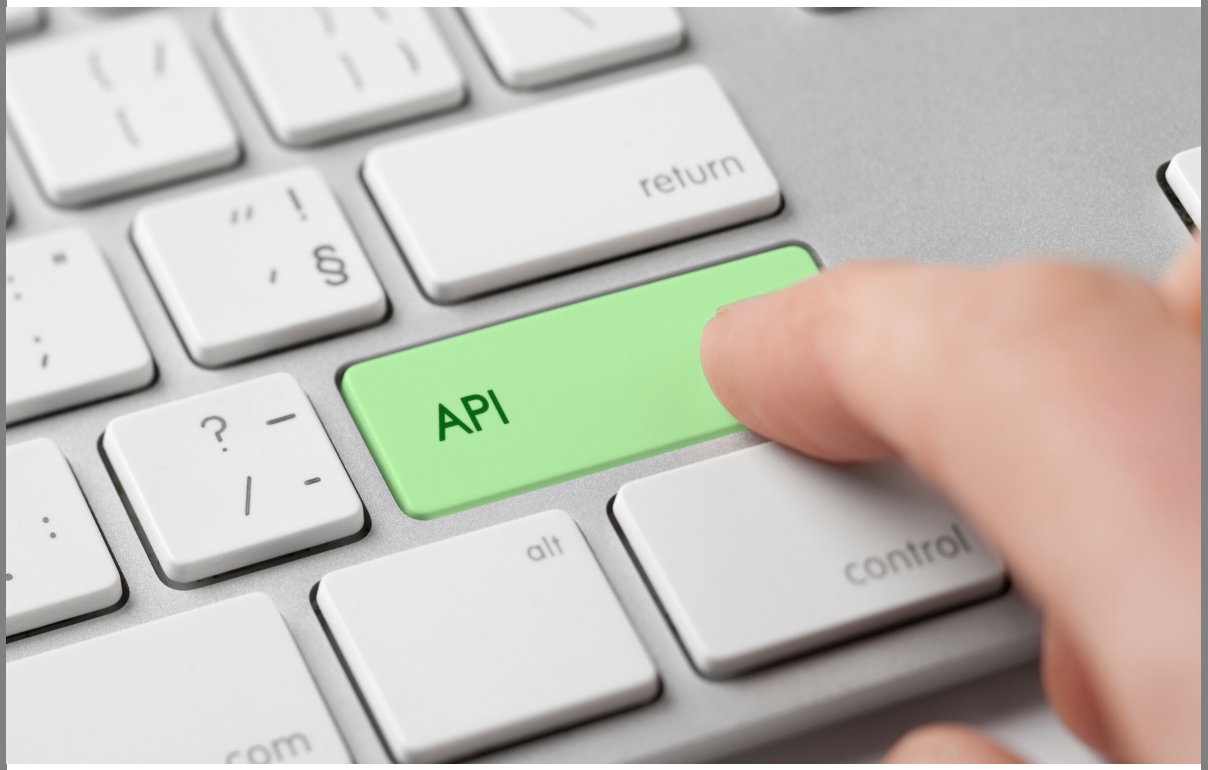
Solving API Security with Automated Methodologies



datatheorem

Prevent AppSec Data Breaches

What are APIs and how are they used?

Application Programming Interfaces (APIs) are endpoints that enable communication between different technologies, such as a home security camera and the Cloud. This technology is not new as computers have been able to talk to each other for several decades. Until about 10 years ago, API use was expensive and difficult to manage, requiring custom-coded or proprietary connectors.

Next-generation APIs are built with free and open standards that make it possible for an application to interoperate easily and inexpensively with virtually any other piece of software or data source. People use APIs in their everyday lives. When someone looks up the weather through an app on their phone, the app is calling on weather APIs to get the requested data in real-time. APIs are the backbone of mobile applications, web applications, and all internet-connected software (including many IoT devices). Companies now need to protect their RESTful APIs in a way that allows thousands of devices to access them.

Some APIs are built and maintained by companies directly, some are from partner organizations or third party SDKs (Software Development Kits). SDKs include open source tools for developers that make it easier to create an app. On average, one mobile app includes twelve to eighteen third party SDKs. While the iOS and Android security models have some safety measures on applications, there is no separation coverage for embedded third-party software. So the risk is that some developers are unknowingly allowing these third party SDKs to have full access to your app's data or they may carry malicious code. So while software development has sped up with the use of APIs, the act of securing them remains problematic.

Why is Securing APIs Important?

APIs are the new standard way to communicate between two entities, so it must be secured. Gone are the days where file transfer happens over FTP, as it happens on port 443 (TLS) via an API (either Box, Dropbox, or your inhouse storage cloud). Gone all the days where web apps have a client (thick browser) and application server, as it happens via a thin browser client and a server side API. Plus, that server side API is also serving other clients such as mobile apps, IoT apps, and web applications, using a one-size fits all model. So if all communication is happening over TLS (port 443 through firewalls) via APIs, the API must be locked down in terms of authentication, authorization, and encryption.

In our weather app example, it's up to each weather app provider to make sure they share the public weather modeling and prediction data while securing private data of the users at the same time. For example, a hacker could create an API call that extracts data from users of the weather app because the API is not configured to stop that from happening, nor does the app owner have an alert system to tell anyone that it has happened.

Gartner analysts have sounded the alarm that API security is a looming threat. Exposed APIs today account for 40% of attack surface area, but by 2021, Gartner predicts that 90% of web-enabled applications will have more surface area for attacks. Gartner also predicts that by 2022, API abuses will become the most frequent, resulting in data breaches for enterprise web applications.

Challenges With Securing APIs

The cloud is one factor in securing APIs correctly. While any APIs can be insecure if not managed properly, at least on-premises there is a more trackable process to develop and deploy an API. In the cloud, developers can quickly throw APIs up on cloud platforms using serverless, containers or microservices. Furthermore, developers may just store assets in the cloud and once those assets become attached to APIs, they create vulnerabilities and the data will need to be compliant. This creates several security problems.

One is the complexity of the environment, which may make it hard to determine how secure any given API is at a moment in time. Also, anyone can spin resources up and down so quickly and easily that it becomes hard to keep track of the APIs being hosted in the cloud. These are known as “shadow APIs.” How do you track these around the clock among several cross-functional teams that share data with numerous web and mobile apps?

A second factor is the speed of software development and the ability to secure APIs as they are created and deployed. The risk of an API getting deployed without any access controls or monitoring becomes inevitable as companies adopt a continuous integration/continuous deployment (CI/CD) cycle. This cycle enables software development and updates to occur at a rapid clip. This pace of development is simply too fast for traditional application security techniques to work.

A third complexity is that the underlying data store for many APIs is an unauthenticated cloud storage bucket. These buckets are often open by default, which can result in the data being deleted or modified, inappropriate content could be uploaded to the bucket, or the bucket could be deleted at any time by someone outside of the organization. Given these potential consequences, it’s important to know more about this threat and how to stop data from being publicly exposed. So having a security program that constantly scans for these kinds of vulnerabilities is also critical.

Finally, APIs are the connective tissue that runs through all of your mobile, web, and cloud applications and these all have broad permissions. This means attacks through the API can basically give bad actors visibility into everything within the application infrastructure. API calls are also prone to the usual web request pitfalls such as injections, credential brute force, parameter tampering, and session snooping.

How to Fail at API Security

According to Verizon, 56% of companies are taking months or longer to find a breach. Everyone has had the experience of receiving an email notification of a data breach, only to find out that the breach occurred months, if not years prior. The delay may be due to the ineffective ways that companies test their security posture.

In the past, an acceptable security strategy was to conduct pen testing and manual audits. This is now considered a temporary strategy given that APIs change constantly. There are also potential blind spots considering many companies don't know that some of their assets are in the cloud, even if they think they are only on premises. In that case, APIs may still be making calls to confidential information while accessing those assets.

Companies have also tried to secure these APIs with gateways. Companies assume that an API gateway can handle many forms of code all communicating through APIs, with hundreds of thousands of APIs in one enterprise. Also API gateways cannot be deployed to all platforms, so developers eventually find ways to avoid forcing everything through an arbitrary chokepoint. It's not scalable for them, nor cost effective for the company.

The most common API authentication strategy is OAUTH (Open Authorization), which is a token authorization system. After a user enters credentials, it then relays that user to a page on the destination server where they can enter their credentials, and then returns to the API client an access token for that user. This is effective for tracking APIs, but there are some gaps that can be hacked.

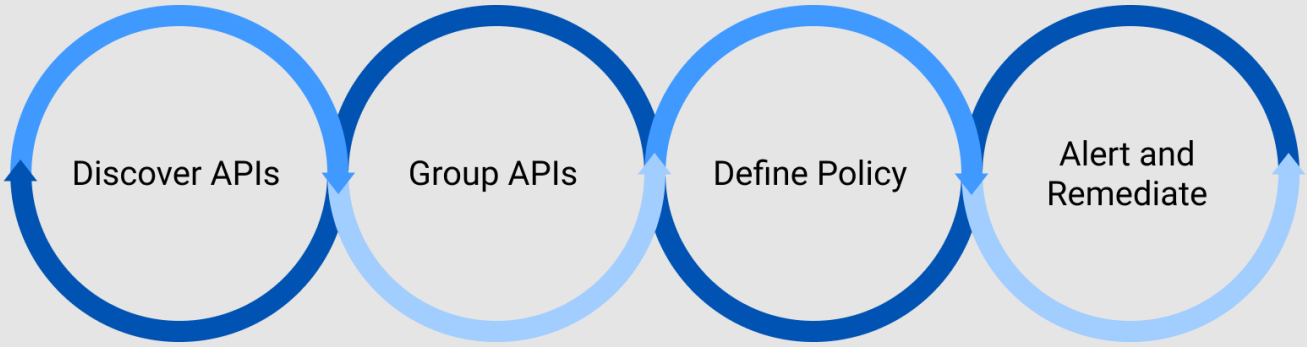
There are a lot of approaches and options, depending on the use case, when it comes to implementing authentication in an API: OAuth, JWT, API keys, etc. Because so many options are available, and due to the complexity of each approach, it is easy for developers to make mistakes when setting up authentication. However, any mistake can have drastic consequences, such as allowing anyone on the Internet to exercise the API and access unauthorized data.

Additionally, nothing is "set in stone" when it comes to deploying modern APIs in the cloud: code and configuration changes can happen several times a day, and any bad update could result in exposure.

Automation makes effective API security possible

A new approach today involves continuous follow up with dynamic, run-time analysis that can uncover real security problems in your entire application architecture. Done right, automated analysis identifies critical issues with a clear path to remediation for mobile apps, cloud apps on containers or serverless, web apps, and their associated APIs. Once a problem is uncovered, the developer can address it as a software “bug,” in the form of a JIRA ticket that includes secure code samples and recommendations to make the remediation straightforward. This can all be achieved by connecting the multiple cloud and on-premise environments to a single aggregator that acts as the source of truth, enabling automated discovery and monitoring of assets so that your visibility is always up to date.

Here is how Data Theorem evaluates your current API security posture and helps you ensure a security posture that can withstand attacks:



1.Inventory APIs & Discover Shadow Assets:

The old adage is still true, you can secure what you can’t see. To begin the program, you will need to start with a full mapping of how many APIs you have, who has access to them, where they live in the cloud/on premises, and whether they are working through web apps that may leave you vulnerable . Then you’ll want to group them according to how you want to set policies that will be enforced. For example, an internal API will be secured much differently than an external API.

2. **Inspect APIs for Data Leaks**

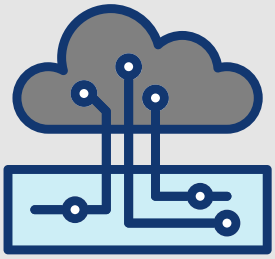
Next you'll want to analyze all internet-facing APIs that require authentication. Should any API move from "authentication required" to "anonymous & publicly accessible", that is a leaky API!. Believe it or not, the root cause for many AppSec data breaches is due to developer changes that left APIs open (leaky) on the Internet, not due to an elaborate SQL Injection attack by a malicious hacker.

3. Hack and Extract

While leaky APIs might be the root cause of most of your AppSec data breaches, hackers are still trying to break in. Thus, you must mimic attacker behavior by testing your authentication and authorization controls on each API, preferably on a daily basis (just like hackers do)! Should a hacker bypass any of your authentication or authorization control, parse any data for Non-public Private Information (passwords, credit card numbers, social security numbers, geolocations and other sensitive information categories such as PII, PHI, and/or PFI).

4. Injection

In addition to hacking your APIs, you must test them for Injection attacks. Ensure all apps, including partner APIs, Serverless, and Single Page WebApps are tested for Internet accessible injection points (e.g RESTful API with GET, POST or PUT methods). Once detected, perform injection attacks on each one on a daily basis, just like attackers do! .Alert & Remediate After security issues are found, it is important your AppSec program is part of the solution. Gone are the days where you can hand over a bag of issues to the engineering team and pray they prioritize the tickets ahead of the feature request. Thus, ensure you have some ability, via your Cloud provider and AppSec tool, to auto-remediate selected issues that should never be vulnerable in "real-time". Examples include, but not limited to, KMS Keys , PII exposure, open storage buckets, SQS Injection, among many others. This should close the loop on the most important aspect of keeping your data safe.



Finally, based on the reporting and current security posture you will want to establish an API security policy, along with your broader application security strategy. This will require your team to create clear guidelines and best practices so that APIs are scanned in pre-production to avoid any breaches or leaky data. Normally, multiple people are hired to fulfill all of these needs for an API security program. However, our automated, continuous solution provides monitoring and remediation across all of your modern application attack layers.

If you have questions about Automated API security, please drop us a line datatheorem.com/contact.

Data Theorem is a leading provider of modern application security. Its core mission is to analyze and secure any modern application anytime, anywhere. The Data Theorem Analyzer Engine continuously scans APIs and mobile applications in search of security flaws and data privacy gaps. Data Theorem products help organizations build safer applications that maximize data security and brand protection. The company has detected more than 400 million application eavesdropping incidents and currently secures more than 4,000 modern applications for its Enterprise customers around the world.