



Anatomy of an API Attack

Understanding BOLA and inventory management exploits

Introduction

Most security teams now understand that proactive threat hunting is an essential element of an effective enterprise security program, especially when it comes to application programming interfaces (APIs). APIs often provide direct access to data, functionality, and workflows. And while baseline perimeter security measures are widely used to protect applications, API abuse and other types of attacks are on the rise. In fact, some of the highest-profile security incidents to hit the headlines in recent years have been API-related. To better understand these attack profiles, such as Broken Object Level Authorization (BOLA) and improper inventory management exploits, this paper will:

- Review the basics of APIs
- Explore why API security is a topic of growing importance
- Use some high-profile API security incidents to highlight key API security areas
- Illustrate the capabilities needed to perform API threat hunting effectively

API and endpoint basics

To start, let's review some basic terminology. APIs are used for many purposes, ranging from business-to-consumer (B2C) functionality, business-to-business (B2B) collaboration and integration, and internal development and integration functions. Web APIs, which communicate over the same HTTP protocol used by web browsers, are the most common implementation model. The specific functionality these APIs provide may also sometimes be referred to as services or API products.

When thinking about API security, it's also important to understand the concept of an endpoint. While this term is sometimes used to refer to end-user computing devices, they have a different meaning in the context of APIs. You can think of an API endpoint as a single accessible resource that is part of the API, along with the operation that can be performed on it.

Here's a simple example. An API endpoint that returns order information for a specific company might be represented as: `GET /orders/{orderID}`. In this case, `GET` is a specific HTTP method, while `orders` and `orderID` represent the particular resource being requested through the API.

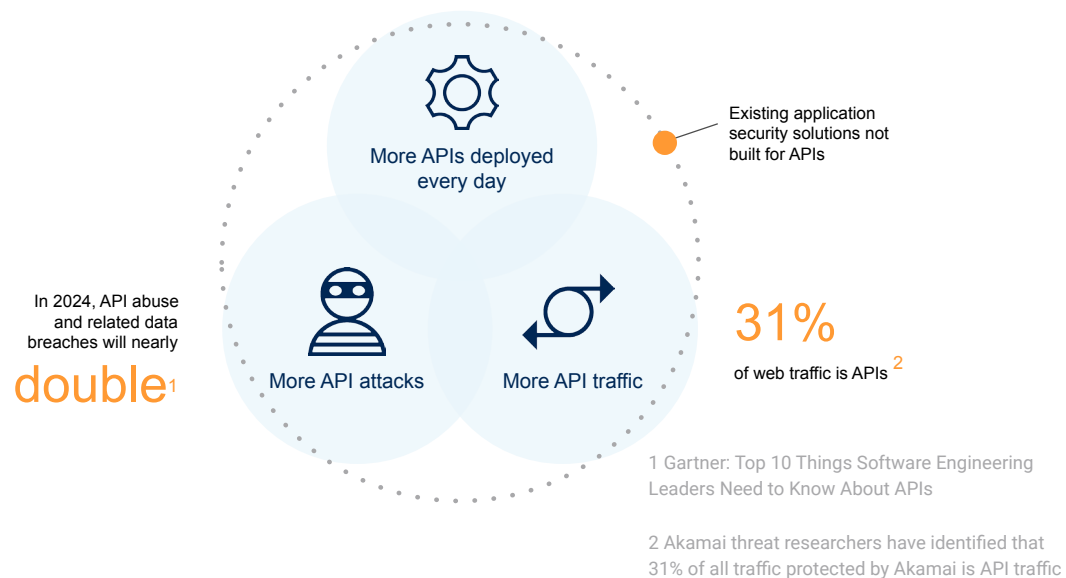
Why are APIs the next big security challenge?

In the past, an attacker may have set their sights on breaching an enterprise data center to access and exfiltrate an organization's data from a specific server. Or they may have attempted to inspect enterprise network traffic to capture sensitive data. In these scenarios, proactive threat hunting might center on activities like penetration testing to cut off threat actors' possible points of entry.

In an API-enabled world, this dynamic is different. Many APIs are inherently accessible to anyone in the outside world, with credentials and keys sometimes acting as the only line of defense. And threat actors are increasingly adept at compromising these elements. In addition, some of the most damaging types of API abuse can originate from parties who have been granted access to APIs but choose to use them in unsanctioned ways.

API attacks in the real world

At Akamai, 31% of all traffic we protect is API traffic. This increasing API traffic leads to downstream effects, such as increasing attacks and abuse. [Gartner projects that in 2024](#), API abuse and data breaches will double. Meanwhile, many security teams are stuck in catch-up mode. APIs just keep multiplying, while existing application security tools offer very limited API protection.



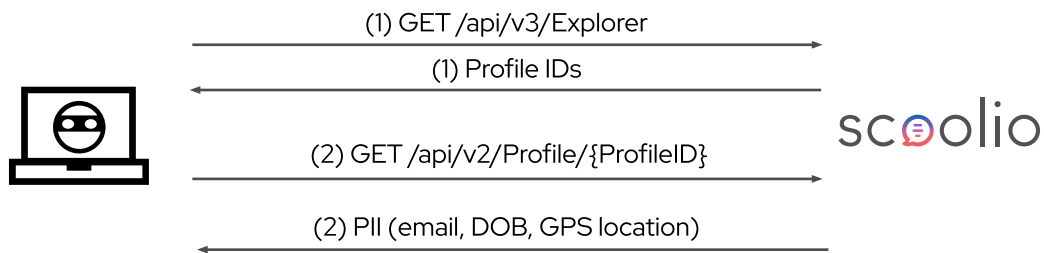
To bring this issue to life, let's review a case study that illustrates the real-world impact that API attacks can have on businesses – and their customers.

Case study

Account takeover | Scoolio

A high-profile example is a 2021 incident that affected the German education app Scoolio. The app collects extensive information from the student users. For example, it conducts personality tests, provides social networking and chat features, and manages activities like study planning and tutoring. These features amass a trove of PII. Security researcher Lilith Wittmann discovered a BOLA vulnerability in the education app's APIs that made it possible to utilize two API calls to access PII and other data for any other user of the education app.

Here's how it worked:



Step 1

Send a GET /api/v3/Explorer API call.

This call returned UUIDs, which were referred to as ProfileID in this implementation.

Step 2

Send a GET /api/v2/Profile/{ProfileID} API call.

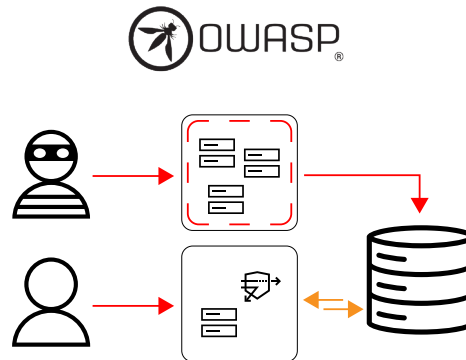
This request returned extensive PII for the relevant user, including email, date of birth, GPS location, and more.

The value of UUID usage

While both the scenarios focus on the use of UUIDs, the use of UUIDs is in fact a very good practice. Using randomly generated numbers instead of a predictable sequence of user identifiers makes it more difficult for a threat actor to access information for users en masse. The problem arises when the UUID information is exposed too permissively and combined with BOLA vulnerabilities.

Improper inventory management

Another facet of this API vulnerability exploit is that it took advantage of **improper inventory management**, which is number 9 on the OWASP API Top 10 list. If you look closely at the attack sequence, you will notice the first step is applied to version 3 of the API, while the second step was taken against version 2. Improvements were made in version 3 that provided tighter governed access to PII. However, these improvements were undermined by the fact that the more vulnerable version 2 remained accessible to all. Ultimately, both version 2 and version 3 were affected by the BOLA vulnerability. But the unnecessary presence of version 2 made the impact of the vulnerability more severe.



What steps do organizations take today to protect their APIs?

Many organizations approach API security by focusing on these three pillars:

1. Centralized authorization – First, implementing a centralized authorization engine for all API access ports will reduce API vulnerability risk by heading off mistakes in development that result in flawed authorization mechanisms.
2. API testing – A second important practice is API testing. Testing for all vulnerabilities, especially broken authorization, using both static code analysis and dynamic testing, will surface issues early in the development process.
3. Runtime protection – The third foundational pillar is a set of runtime protections for the production environment. Even the most proactive teams won't catch every vulnerability in advance of deployment. So it's essential to inspect user access to production data and prevent exploitation of known categories of vulnerabilities to the extent possible.

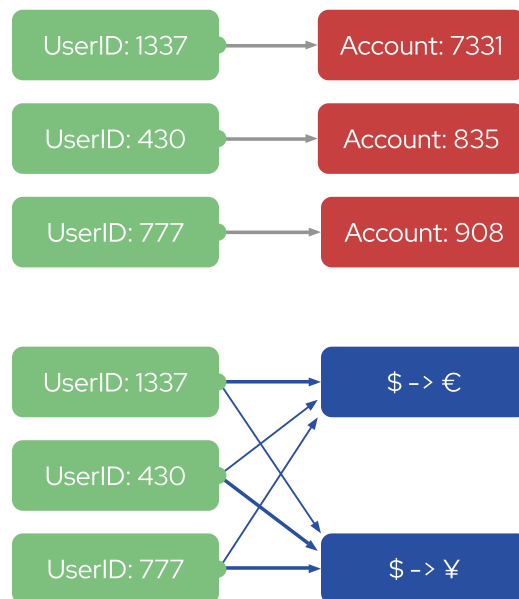
These three practices provide an excellent foundation for your API security strategy. But it's also important to remember that they aren't perfect or comprehensive. For example, even organizations with centralized authorization do not have a guarantee that developers will always follow best practices. And finally, existing application protection tools are often good at detecting known attack patterns but less capable of detecting more nuanced threats like BOLA.

How can you build on this foundation with more advanced BOLA detection techniques?

One of the keys to detecting and mitigating BOLA and other nuanced API vulnerabilities is to model the relationships between the entities involved in API activity. This includes actors, such as users, trying to access resources, in addition to the resources themselves. If you can map these connections between the actor entities and business process entities interacting with an API, it unlocks the ability to differentiate between legitimate and illegitimate activity when analyzing otherwise identical API events.

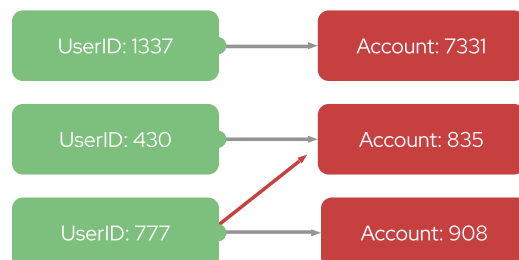
Relationship mapping illustrated

To better understand relationship mapping, consider this basic example. A banking application supports two actions. One action is to read your account data, including information like account balance, recent transactions, etc. The second action is to view currency exchange rates. The relationship between users and resources in these examples is very different. Access to account information should be limited to a single user. In contrast, the exchange rate function should be generally available to all users.



While this is a very basic example, building a more sophisticated model of relationship mappings between entities makes it much more practical to prevent or detect BOLA.

Here we see a user attempting to access an account they do not own. The specific API call may be identical, but the added context provided by entity mapping makes it clear that it should not be allowed.





Advanced BOLA attack detection in practice

Next, let's apply this concept to more complex examples, such as the case study vulnerabilities. Below are snippets of the entities involved the scenario:

scoolio

GET/api/v3/Profile/{ProfileID}

Headers:

- Authorization: <MyAccessToken>

The actor entity is highlighted in green, and the requested resource (the profile ID) is highlighted in red. Once these relationships are understood, steps can be taken to enforce general logic such as limiting access by an actor to a single resource when appropriate. This is far from trivial, since relationships can be more complex than this and include one-to-many dimensions. But techniques like machine learning and behavioral analytics make it possible. For example, a successful detection of a BOLA vulnerability for one of our customers would look like this:

The screenshot displays a security dashboard with a 'Suspicious Data Access' alert. The alert is titled 'Suspicious Data Access' and is marked as 'Open'. The description includes the following points:

- Endpoint "/users/v1/{username}/password" in service "/users"
- A User should not access more than one username
- The User "MyDemoUser" accessed more than one username: "MyDemoUser", "admin"




The alert is categorized as 'Account Takeover' with a 'Medium' severity. Below the alert, a table shows two rows of activity:

TL	ENTITY TYPE	ENTITY ID	ENDPOINT	S.	S.	LABELS	CONTENT
21 Sep 2022 18:24:24	User	MyDemoUser	PUT vampi...	204	10.3...		→application/json(27) ←application/json(0)
21 Sep 2022 18:24:17	User	MyDemoUser	PUT vampi...	204	10.3...		→application/json(27) ←application/json(0)

For this example, a BOLA vulnerability was simulated in a lab environment. Through entity mapping and behavioral analytics, our platform detected the BOLA and generated an information-rich alert. A security analyst or threat hunter who views the alert will see that MyDemoUser accessed their own user profile to change their password, a sanctioned action. But soon after that on the timeline, we can see that they performed another API call to change the administrator password. Since this is clearly an unsanctioned act based on the relationship between the actor and the resource, the alert was generated.

Where to start with your API security initiative

API security is a continuous work in progress for most organizations. This can make it difficult to know where to start. While the three foundational pillars above provide a useful starting point, the effectiveness of your approach will be greatly enhanced if you follow these three recommendations with your implementation:

-  1. Ensure an always up-to-date API inventory
-  2. Monitor non-production and production API environments
-  3. Enforce relationships between entities

You can't protect APIs that you don't know about. So effective API protection starts with an up-to-date API inventory and security posture assessment. Similarly, as you develop your API security monitoring capabilities, it's important to extend them to both product and non-production API implementations. And most importantly, your API monitoring and enforcement must extend beyond actions alone and consider the relationships between the entities involved in your API activity. This will allow you to find vulnerabilities and protection gaps and enforce compliance with intended API usage models. Understanding behavior within your APIs will allow you to see any abuse.

Interested in understanding more about API attacks and how you can protect against them? Check out our [OWASP API Top 10](#) breakdown.



Akamai protects your customer experience, workforce, systems, and data by helping to embed security into everything you create — anywhere you build it and everywhere you deliver it. Our platform's visibility into global threats helps us adapt and evolve your security posture — to enable Zero Trust, stop ransomware, secure apps and APIs, or fight off DDoS attacks — giving you the confidence to continually innovate, expand, and transform what's possible. Learn more about Akamai's cloud computing, security, and content delivery solutions at akamai.com and akamai.com/blog, or follow Akamai Technologies on [X](#), formerly known as Twitter, and [LinkedIn](#).