THREATX

Guide to Getting Started With API Security

How to Start, Evolve, and Mature Your Program

INTRODUCTION

As critical building blocks of modern applications, it's clear that APIs are here to stay.

As critical building blocks of modern applications, it's clear that APIs are here to stay. As one of the most appealing attacker targets and fastest-growing facets of your attack surface, it's also clear that APIs need protection.

An API security program should include strategies, protocols, and tools to protect application programming interfaces (APIs) — both those that are externally facing and those that power internal services — from unauthorized access, misuse, or malicious attacks, ensuring the confidentiality, integrity, and availability of data they handle and the services they enable.

A comprehensive API security program gives you the full visibility you need to know your APIs inside and out, implements guardrails to ensure developers are building robust APIs, and blocks attacks in real-time. It helps minimize the risk of an API-based attack while enabling developers to continue to benefit from the speed and flexibility afforded by APIs.

Keep in mind that API security requires a coordinated program that evolves and matures. Effective API security features a comprehensive, ongoing program that encompasses people, process, and technology — rather than simply a one-off project. New threats are emerging all the time — the threat landscape is dynamic, and your defense must be as well.

This guide will explain how to establish, evolve, and mature your API security program.

Step One

Discover and Document Your Internal and Externally Facing APIs

Discover

Collect data for every API

Document

Document and catalog every API in your environment

Address Zombie and Rogue APIs

Avoid outdated or undocumented APIs

1

Step Two

Protect Your APIs

Red Teaming

Conduct offensive security such as penetration testing and security testing

Blue Teaming

Establish steps to prevent, mitigate, and respond to security incidents

Purple Teaming

Optimize collaboration between red and blue teams

2

Step Three

Measure and Report the Effectiveness of Your API Security Program

Initial Metrics

What to measure immediately

Metrics Over Time

What to measure as the program matures

Long-Term Metrics

How to tie the program to business objectives

3

Side Note

Getting Leadership Buy-In

Most traditional security tools aren't sufficient for protecting APIs, so you may need to request additional resources. Here's what leadership needs to know:

Why APIs Are Important

APIs enable companies to deliver new services by connecting data and apps with one another. In addition, they enable developers to build applications faster. Instead of building capabilities from scratch, they can reuse APIs to carry out common functionality. (Consider providing an example from your own applications.)

Why APIs Are at Risk

APIs are designed to expose application logic and sensitive data, making them a natural attack target. And, just like applications, APIs are susceptible to vulnerabilities. Since many companies lack visibility into their APIs, it's often easier for attackers to take advantage of these vulnerabilities while going virtually undetected.

Gaps in Current Approaches

Scanning code for vulnerabilities will always be a best practice. However, code scanners are primarily used to identify known vulnerabilities pre-production. They can't detect what appears to be legitimate user behavior executed by attackers in production code. In addition, while API gateways play an important role in controlling and monitoring authentication and authorization, their focus is on API operations — not security.

THE IMPACT OF AN API ATTACK

$75 Billion

Overall cost of API breaches in 2022

$4.35 Billion

Average cost per breach

Examples of API breaches include:

- A single API vulnerability led to the breach of 5.4 million Twitter users' data.
- Venmo leaked more than 200 million transactions via a publicly accessible API.
- A vulnerable API allowed attackers to obtain private account data of Peloton users.
- A threat actor stole the personal information of 37 million T-Mobile customer accounts via an exposed API.

Step One

Discover and Document Your Internal and Externally Facing APIs

1

The first step to establishing your API security program is to obtain visibility. The proliferation of APIs across a distributed infrastructure inevitably leads to API sprawl, and most organizations don't know how many APIs they have or how they're utilized. Developers often stand up new APIs or reuse existing ones, but much of this is out of sight of security. As a result, the attack surface created by APIs is significantly larger than most organizations realize.

Discover

When conducting API discovery, consider, for each API:

- → Is it public or private?
- → Is it external or internal?
- → Who owns it?
- → What kind of calls can it make?
- → What is the intended use case?
- → How is it written? (JSON, XML, GraphQL)
- → What kind of data is it transmitting?

Document

Once you have an inventory of every API in your environment, it's time to document and catalog them. This involves understanding the code behind each API. Ideally, each API has an API schema that provides definitions of acceptable use for that API, including what can be exposed, what methods can be used, what are the parameters, keys that enforce utilization, etc.

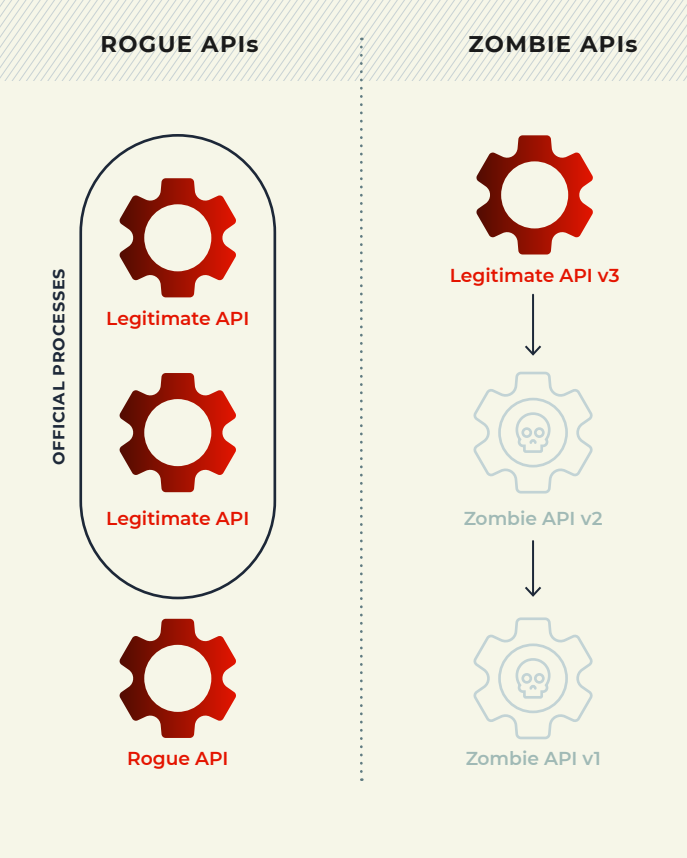
However, when you start your program, you're likely to find that few APIs actually have defined schemas, and you will need to develop schemas as your API security program matures. While not all API protocols have easily definable specifications, the vast majority of APIs currently deployed are RESTful APIs and can be described by an OpenAPI Specification and defined in a JSON schema.

Discovery and documentation can be performed manually or automated with a tool. The best mode of discovery is real-time analysis of the traffic hitting your endpoints. With this data, you can prioritize security efforts on APIs getting traffic. Analytics can identify which endpoints are no longer used and which clients are still actively using old endpoints. Once you know which API endpoints are serving responses, take a hard look at what's deployed and exposed, and compare your actual attack surface with your theoretical API inventory.

After you complete the initial discovery and documentation, shift these documentation functions left, being careful not to slow down development. For example, for RESTful JSON endpoints, you can easily generate an Open API 3.0 schema file from your CI/CD pipeline that will define, in excruciating detail, what is and isn't expected. A tool with OpenAPI schema support functionality can compare what your build system thinks is in your environment with what's actually there, allowing you to quickly pinpoint undefined or unspecified functionality.

Address Zombie and Rogue APIs

It is likely that the discovery process will uncover some APIs that were previously unknown. Like shadow IT, rogue or shadow APIs are those that exist outside of an organization's official security and operational maintenance processes. Countless scenarios can lead to a rogue API.



For example, a developer may quickly stand up an API to resolve a problem or to develop a proof of concept for a larger project. Similarly, zombie APIs are APIs that were previously valid and approved but were eventually abandoned or replaced by newer versions. If the old APIs aren't properly removed from production, they can leave organizations with outdated, vulnerable points attackers can exploit.

To address zombie and rogue APIs, you need visibility, governance strategies, documentation, and communication between development and security teams. Specifically for zombie APIs, make sure you have proper versioning and defined processes and procedures for decommissioning old or unused APIs. For rogue APIs, ensure you have a strategy and process to enforce how and why an API gets deployed with the proper security controls. Think ownership, documentation, access, authentication, etc.

Step Two

Protect Your APIs

2

With step one complete, you know what APIs you have and are familiar with their functionality. Now you can protect them. Protection falls under three categories that most security professionals are familiar with: red teaming, blue teaming, and purple teaming. ***(Note: the following is not an exhaustive list of protection mechanisms, but key concepts that will give your program a solid start.)***

Protection Categories

Red Teaming

Offensive security

- Authentication and Authorization
- Encryption
- Rate Limiting and Throttling

Blue Teaming

Defensive security

- API Management Via API Gateway
- Logging and Monitoring
- API and Application Protection
- Oauth and Tokens

Purple Teaming

Collaborative security

- People
- Process
- Technology

Red Teaming

Red teaming is offensive security.

This includes proactive efforts such as penetration testing and security testing in an attempt to breach your organization's security and see how far one can get in an organization. Uncovering vulnerabilities is a primary focus for typical red-teamers. For the purposes of an API security program, red teaming efforts should focus on the following:

Authentication and Authorization

Many of the most common API vulnerabilities are related to authentication and authorization flaws. Developers may, for example, loosely configure an API's authorization checks based on assumptions about the clients interacting with it. This leaves application functions or API calls available to any attacker who steals the API keys or — worse — discovers that an API doesn't require authentication keys. Even when developers configure APIs with strong authentication requirements, they may violate the principle of least privilege by granting API clients access to more data and application functions than necessary.

So, first and foremost, from a red teaming perspective, ensure that authentication and authorization mechanisms are working as intended and that only authorized users have access to API resources. Test for weaknesses in these mechanisms, such as weak passwords, insufficient validation of access tokens, and ensure that the people or machines authorized to access or pull the data from an API have a legitimate need to do so. The concept of "Zero Trust" comes into play here — don't trust anyone or anything until you do proper authentication and authorization. Validate and verify.

Encryption

Another way to prevent unauthorized access to your data is via encryption, which creates secure communication between the user and source. SSL/TLS encryption is mainstream and should be used for both public and private APIs, especially those that are holding or transmitting sensitive data. Always use the strongest encryption methods to protect your APIs, such as TLS 1.3.

In addition to enabling encryption, it's important to test your API's encryption mechanisms to ensure that they're working as intended, and that sensitive data transmitted via the API is actually protected. This can include tests for SSL/TLS encryption and key strength.

Rate Limiting and Throttling

Next, look at rate limiting and throttling. Rate limiting restricts the number of requests an API can process, regulating legitimate users while protecting against malicious attacks. Rate limiting is especially important for public-facing APIs to ensure that they are not leveraged in a distributed denial-of-service attack. Simple rate limits are available in many web servers and proxies.

Test APIs against these mechanisms to ensure that they're able to prevent abuse and protect against, for example, DDoS attacks.

Stress-test your APIs and ask:

- Can my APIs handle high volumes of traffic?
- What is considered a "normal" amount of traffic for each API?
- Is an API being called upon excessively?

Over time, establish baselines for what constitutes normal traffic, including periods of peak traffic. This will enable you to more easily recognize when traffic has reached an uncharacteristically high volume and requires investigation and perhaps a response.

Blue Teaming

Blue teaming is the defensive side of security.

This includes efforts to prevent, mitigate, and respond to security incidents. For the purposes of an API security program, blue teaming efforts should focus on the following:

API Management Via API Gateway

An API gateway is a mediator between your APIs and external clients. API gateways manage the operational aspects that keep APIs running properly. API gateways cover a variety of operational tasks such as controlling access, translating RESTful requests to SOAP, or elastically scaling resources if there is a spike of traffic hitting an API.

Logging and Monitoring

To detect an attack, you need accurate and efficient logging of API activity. Capture all the events that occur and put the logs in a centralized tool or technology that is actively monitored and assessed, such as a SIEM or data lake. Monitor these logs and track API activity over time, identifying potential security threats and responding to incidents as they arise. Set up alerts to flag anomalous activity so you can get immediate notification and decide if a response is required.

API and Application Protection

Real-time scanning of all inbound and east-west API traffic is critical to effectively identify and block attacks, and can be accomplished with the support of technology. API threat protection solutions, for example, support all API infrastructures as well as many traditional web assets with the sole purpose of monitoring and identifying risky behavior and attacks. These solutions focus on understanding the API attack surface, detecting and blocking the ever-growing spectrum of modern API threats, and keeping up with the changing techniques attackers use to evade detection. An API threat protection solution can also provide an integrated view of risk, which is essential for repelling modern attacks and demands considerable expertise and focus to keep pace with evolving threats.

RUNTIME API PROTECTION

With the growing use of cloud infrastructure, APIs are now increasingly facing runtime threats, and organizations should consider solutions that can defend against these attacks. Many attackers now prioritize finding backdoors — often by exploiting vulnerabilities in running APIs and applications — both external and internally facing — to circumvent edge and perimeter defenses. Runtime threat protection describes the ability to monitor the environment where an application or API is executed and take action to stop malicious behavior.

Oauth and Tokens

When it comes to authentication, a username and password are not typically passed in day-to-day API calls. Instead, Oauth and JWT tokens are the standard to authenticate to an API. An API key or bearer authentication token is passed in the HTTP header or in the JSON body of a RESTful API. Tokens should expire regularly. Most enterprises use an internal database or DLAP authentication store, though Oauth may be an option for highly public APIs.

Purple Teaming

Once the red teaming and blue teaming pieces are in place, mature your API security program with purple teaming.

This concept emphasizes collaboration between your red and blue teams. Purple teaming is your people, process, and technology working together to improve security.

People

Regardless of the technology you implement, people will remain the weakest link. Security awareness training is a must-have for all employees, and developers should have their own dedicated training. Make sure security "reaches across the aisle" and has transformative conversations with development to ensure developers have bought-in to the program and you're actually protecting your applications throughout the entire software development lifecycle, to production.

Process

There are a number of processes that can help improve your API security and mature your program:

Stress testing your APIs.

Do a tabletop exercise of a DDoS attack or botnet attack against one of your applications. Determine what would happen if you had downtime and how that risk can be reduced or prevented.

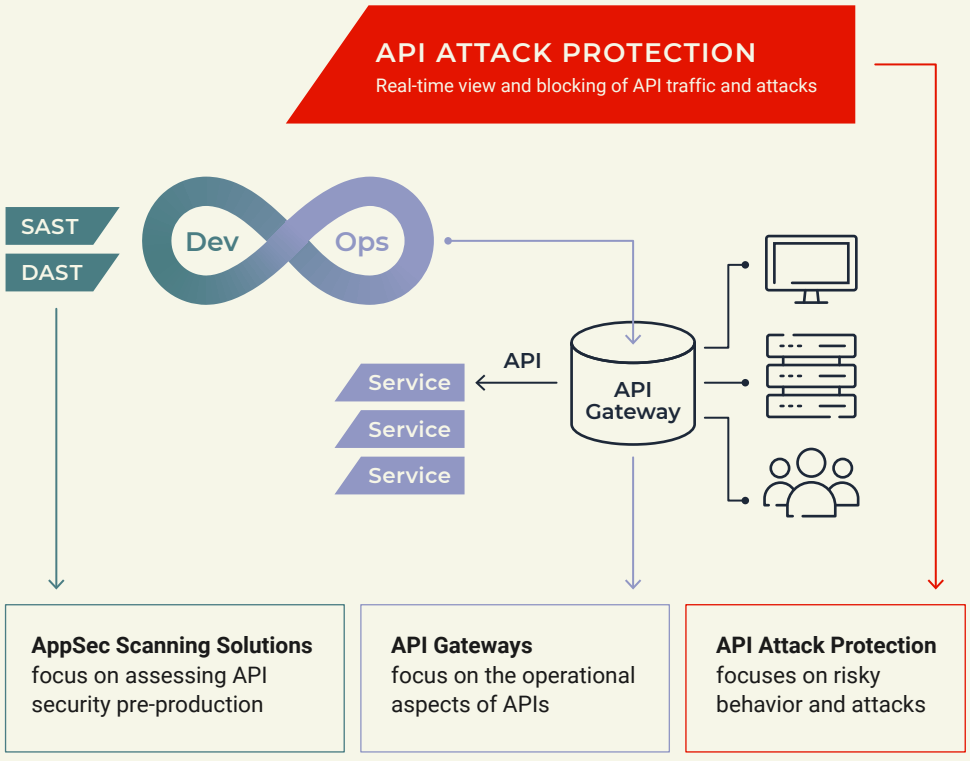
Policies and procedures are important, but they must be flexible.

They must be able to change over time as new threat actors advance their techniques against your organization. Also consider compliance and regulations and how your APIs may be transmitting certain data types, particularly personally identifiable information (PII) that may be covered by various industry and governmental mandates.

Technology

Defense in depth is a security best practice that involves leveraging multiple security technologies to provide a layered defense. For your API security program, defense-in-depth includes the previously mentioned red teaming and blue teaming controls and measures, including API gateways and an API protection platform to protect against real-time threats. While not exhaustive, consider deploying technologies throughout your SDLC (software development lifecycle) such as SAST, DAST, and SCA. In addition, at least annually, perform penetration testing and vulnerability assessments against your applications and APIs.

API DEFENSE IN DEPTH



Step Three

Measure and Report the Effectiveness of Your API Security Program

3

Once an API security program is in place, you can begin collecting metrics. These will be useful for keeping stakeholders (such as the board) current on your progress. Tracking a set of metrics over time also allows you to measure the effectiveness of your API security program.

Initial Metrics

Begin with the following metrics:

- Number of APIs
- Internal vs. external
- What are they connecting to
- Percentage of APIs targeted
- Percentage of blocked attacks
- Application uptime/availability by quarter
- SLAs met
- Top three attack vectors

Metrics Over Time

As your program matures, measure the following:

- Total API requests vs. API requests blocked over time
- Number of APIs targeted over time
- Your security posture vs. your peers
- The cost of an API breach (consider ransom, non-compliance fines, lost business, and the time and cost to respond)
- How are we trending — are we getting better with time, are we protecting our applications, or are we degrading?

Long-Term Metrics

The long-term goal, as you collect these metrics, is to tie the API security program back to specific business objectives; for example, our APIs staying operational is a direct impact to our business revenue by ensuring our mobile app is up 24/7. If the mobile app goes down for 1 day, we lose $X. This way, you're taking your API security program from tactical metrics to operational metrics, which should resonate all the way up to the board.

CONCLUSION

API security requires a dedicated program that organizations can evolve and mature to meet business needs and address an ever-changing threat landscape.

By starting with the strategies, protocols, and tools outlined here, you can establish a solid foundation for an API security program.

NEED HELP?

ThreatX has helped many companies build out their API security programs, and we'd be happy to help you build yours.

Learn more about API protection. →

Reach out today for guidance. →

THREATX

www.threatx.com
info@threatx.com

ABOUT THREATX

ThreatX is managed API and application protection that lets you secure them with confidence, not complexity. It blocks botnets and advanced attacks in real time, letting enterprises keep attackers at bay without lifting a finger. Trusted by companies in every industry across the globe, ThreatX profiles attackers and blocks advanced risks to protect APIs and applications 24/7. Learn more at www.threatx.com.