Definitive Guide™

to

API Attack Protection

Reduce API security risk and stop API-based attacks in their tracks

Crystal Bedell

SRC CYBER SOLUTIONS LLP

Compliments of:

THREATX

FOREWORD BY:
Bret Settle

ABOUT THREATX

ThreatX's API protection platform makes the world safer by protecting APIs from all threats, including DDoS attempts, bot attacks, API abuse, exploitations of known vulnerabilities, and zero-day attacks. Its multi-layered detection capabilities accurately identify malicious actors and dynamically initiate appropriate action. ThreatX effectively and efficiently protects APIs for companies in every industry across the globe. For more information, visit: https://www.threatx.com/

ABOUT SRC CYBER SOLUTIONS LLP

At SRC Cyber Solutions LLP, we provide Next Generation, Automated and User-Friendly solutions in partnership with AUTOMOX for Patch and Endpoint Management, IRONSCALES for Comprehensive Email Security and Anti-Phishing Protection, THREATX for WAAP (WAF++) with an Attack-Centric approach for Web Application and API protection and Project Ares for Online Gamified Simulation-based Cyber Security Training.

Definitive Guide™

to

API Attack Protection

Reduce API security risk and stop API-based attacks in their tracks

Crystal Bedell

Foreword by Bret Settle

CYBEREDGE PRESS

Definitive Guide™ to API Attack Protection

Published by:
CyberEdge Group, LLC
1997 Annapolis Exchange Parkway
Suite 300
Annapolis, MD 21401
(800) 327-8711
www.cyber-edge.com

ISBN: 978-1-948939-33-1 (Paperback)
ISBN: 978-1-948939-34-8 (eBook)

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgements

CyberEdge Group thanks the following individuals for their respective contributions:

Editor: Susan Shuttleworth
Graphic Design: Debbi Stocco
ThreatX Contributors: Suzanne Ciccone, Bret Settle, and Dave Howell

Table of Contents

Foreword

API security is not a new problem. APIs and their vulnerabilities have been around a long time. What is new is the volume of APIs that are now being created at lightning speeds. Thanks to this staggering increase in adoption, security is struggling to keep up, and APIs are increasingly featured in breach headlines and becoming top of mind for security professionals.

However, the rapid increase in numbers is not the only factor leading to risk and breaches. Some of what makes APIs so attractive from a business perspective also makes them vulnerable. Their ease of integration, multi-use components, and advanced automation can lead to unauthorized access to capabilities, information disclosure, functional abuse, denial of service, security misconfiguration abuse, and more. But these attacks are not new, either. They're similar to attacks against web applications that we've seen for years. What is different and more challenging about API security? According to our recent conversations with customers, it's the following:

Lack of visibility and awareness — This is a huge factor. Organizations know they have a lot of APIs, but they don't know how to discover or inventory them all. Most have an assortment of APIs sitting behind various gateways, some with no schema attached, or some that have been essentially orphaned after long periods of development. This is a hard problem for security teams to get their hands around.

Accountability for acceptable API usage — Organizations struggle to understand who is using their APIs and what constitutes acceptable usage — things like the amount of information APIs should provide, how they provide it, and the necessary levels of authentication and authorization are often a mystery.

Sophisticated, multi-step attacks — Because of a lack of visibility, a lot of attacks now focus on APIs, and they are not easy to detect. It's challenging to identify abnormal behavior related to APIs without the right technology with correlation capabilities to identify modern attacker techniques.

Complex, bot-based attacks — Complicating everything is the fact that API attacks are increasingly bot enabled. Attackers can now distribute loads across large numbers of bots — often tens to hundreds of thousands — to build complex, bot-based attacks aimed at manipulating API functionality.

These are big challenges and addressing them isn't simple. This book is a good place to start. We're thrilled to play a role in its development and think that organizations looking to understand the scope of the API security problem and key requirements for solving it will find the *Definitive Guide to API Attack Protection* a valuable and practical handbook.

Bret Settle
Co-founder and Chief Strategy Officer
ThreatX

Introduction

The world runs on APIs. Every modern software application uses *or is* an API. The explosive growth of APIs, fueled by accelerated digital transformation initiatives, has surpassed security's ability to protect these valuable applications. Unfortunately, attackers know this only too well.

APIs are not easy to protect. By their very nature, APIs expose application logic and sensitive data. Attacks can look like legitimate user behavior. Also, attackers can combine a variety of evasion techniques and attack methods to bypass traditional security controls.

Protecting APIs is critical for safeguarding valuable business assets and preserving the organization's ability to rapidly innovate. And sophisticated attacks require sophisticated solutions, which you'll read about in this Definitive Guide.

Chapters at a Glance

Chapter 1, "Understanding APIs," discusses what APIs are, why the API economy is exploding, and the risk associated with this growing threat landscape.

Chapter 2, "Exploring the State of API Security," reviews the security measures organizations have in place today to protect their APIs.

Chapter 3, "Deconstructing API Attacks," outlines how attackers leverage various evasion techniques and attack methods to go undetected.

Chapter 4, "Protecting APIs Is Hard," describes the challenges that must be overcome to effectively protect APIs against attacks in real time.

Chapter 5, "Solving API Security Challenges," explores how attacker-centric behavioral analytics enables real-time API protection.

Chapter 6, "Evaluating API Protection Solutions," examines the top 10 criteria for an effective API protection solution.

Helpful Icons

Tips provide practical advice that you can apply in your own organization.

When you see this icon, take note as the related content contains key information that you won't want to forget.

Proceed with caution because if you don't it may prove costly to you and your organization.

Content associated with this icon is more technical in nature and is intended for IT practitioners.

Want to learn more? Follow the corresponding URL to discover additional content available on the web.

Chapter 1

Understanding APIs

In this chapter

- Learn what an API is
- Explore the API economy
- Tour the risk landscape

APIs are everywhere in the tech world. You hear about them. You read about them. Your organization creates, uses, and shares them. And now you need to somehow protect them. But first, you need to understand them. This chapter takes a close look at APIs to help you understand what they are and why they're taking over the digital world.

The Foundation of Digital Transformation

The acceleration of digital transformation initiatives is fueling API growth, and you can argue that API growth is fueling digital transformation initiatives. The two now go hand in hand as APIs have become the de facto standard for modern computing.

What is an API?

An *application programming interface (API)* is an application that enables two or more software components to communicate with each other using a specified set of definitions and protocols, as shown in Figure 1-1. APIs generally use HTTP or HTTPS to transport application requests and responses, often with payloads in JavaScript Object Notation (JSON) or XML format. As is common in technology, newer API formats like GraphQL, gRPC, Kafka, and others are being introduced and

adopted, which increases complexity and compliance requirements. Some APIs are used in server-to-server connections. However, APIs are more commonly used as the back end to mobile apps or web user interfaces. Developers often use them to achieve feature parity between web and mobile versions of the same application.

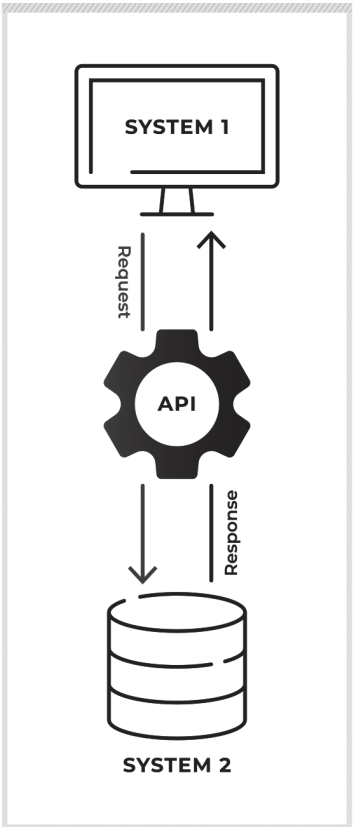

Figure 1-1: An API enables software components to communicate.

APIs are a big deal because they make it easy to integrate and connect disparate systems. Prior to APIs, the incompatibilities of different software platforms prevented separate software components from communicating effectively. APIs overcome these issues and enable any two separate software components to communicate — regardless of their platforms, data structures, or underlying technologies.

As a result, APIs enable employees, consumers, and businesses to instantaneously connect to the information and services they need — anytime, anywhere. This ability to seamlessly connect and share data among multiple applications opens the door to a whole new world that can be summed up in two words: *API economy*.

The API economy

APIs give organizations the ability to expose their digital services and assets in a controlled way, enabling them to connect systems to achieve new efficiencies and create new lines of revenue. Consequently, APIs play a critical role in the digital solutions that impact an organization's bottom line. Companies accelerated their API adoption during the COVID-19 pandemic as they looked for new ways to deliver data to customers and create new revenue streams, further spurring the growth of the API economy.

The API economy refers to business models and practices designed to enable the secure exchange of data. The API economy enables businesses to profit off their APIs by monetizing their data and services. But they can also increase the bottom line in indirect ways, for example, by improving operational efficiencies.

Companies are creating entire business models around their APIs, turning their organization into a platform. This isn't another empty marketing term.

Take Uber as an example. The platform leverages an API for Google Maps (which is a discrete go-to application in its own right) to send location data and directions to Uber drivers.

Whether your company has created an entire business model around its APIs or not, chances are good it is an active participant in the API economy, as explained in Figure 1-2.

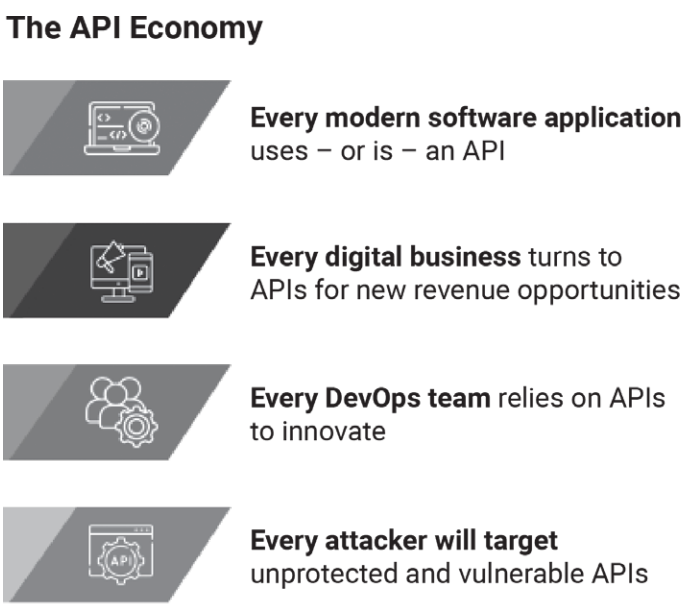

Figure 1-2: The API economy has many facets.

It's all about time to market

Developers are under constant pressure to get code out the door, and APIs help make that happen. APIs provide developers with a fast, highly modular way to add capabilities to their applications. The reuse of APIs helps facilitate rapid development by eliminating the need to build out common functionality from scratch. For example, Uber could have spent years building out its own mapping service – assuming it had the resources to do so. By integrating with Google Maps, Uber can leverage what Google's already achieved and focus on its own value-add.

As I mentioned earlier, APIs are a big deal because they facilitate systems integration. Developers use APIs to connect services and transfer data, allowing users to access the same data across multiple solutions. APIs can also automate repeatable tasks for better efficiency and accuracy. Importantly, APIs provide an adaptive way to work with mobile devices and cloud applications. At the end of the day, though, it's all about time to market. APIs enable DevOps teams to quickly deliver new services and capabilities.

Introduction to API Protocols

Developers have a variety of API protocols and architectures at their disposal to establish communications between disparate systems. Here are just a few of the more popular protocols in use today.

RESTful APIs – The majority of modern APIs are built on the representational state transfer (REST) architecture. REST utilizes a client/server design to separate the front end and back end of an API. RESTful APIs can communicate directly or operate through intermediate systems such as API gateways (which I'll explain further in Chapter 2) and load balancers.

OpenAPI – OpenAPI is a technical specification that can be used to describe APIs. OpenAPI defines the structure and syntax for RESTful APIs in a way that is readable by both humans and machines. When properly defined, an OpenAPI's capabilities can be easily understood without having to access the source code or additional documentation.

GraphQL – GraphQL was designed by Facebook as an alternative to RESTful APIs. It is an open-source data query and manipulation language and server-side runtime for APIs. GraphQL allows developers to construct requests that pull data from multiple sources in a single API call.

gRPC – Google Remote Procedure Call is an open-source remote procedure call system developed by – you guessed it – Google. The framework can run in any environment and generates cross-platform client and server bindings for many languages. gRPC is commonly used for connecting microservices or for connecting mobile device clients to backend services.

SOAP API – Originally developed by Microsoft, the Simple Object Access Protocol (SOAP) is a messaging standard clearly defined by the World Wide Web Consortium. SOAP depends on XML by design and is highly structured. It is often used for internal or partner APIs.

The Risks APIs Pose

As a security professional, you know very well that every technology has downsides as well as benefits. Along with many opportunities, APIs bring their share of risk – and it's not insignificant.

A growing landscape

The size and scale of the API threat landscape cannot be overstated. Nearly every modern software application uses or *is* an API. Consider Gartner's recent statement that "By 2021, 90% of web-enabled applications will have more surface area for

attack in the form of exposed APIs rather than the UI, up from 40% in 2019." (Source: Gartner®, "Gartner's API Strategy Security Model," Saniye Alaybeyi, Mark O'Neill, Refreshed 27 April 2021, Published 22 October 2019.)[1]

No industry is immune to API-based attacks. Any company that uses APIs is at risk. And these aren't proof of concept attacks carried out in a research lab. Large enterprises and government agencies – the likes of Venmo, Peloton, Facebook, GitLab, and the U.S. Postal Service – have experienced security breaches that were the result of unprotected APIs.

An unknown landscape

The growth of APIs is not only rapidly expanding the attack surface, but it's also making it more unknown and ambiguous. It's challenging to secure something you don't know about, and the API landscape is littered with unknown *rogue* and *zombie* APIs.

Rogue or shadow APIs are APIs that exist outside of an organization's official security and operational maintenance processes. Let's say a developer needs to quickly stand up an API to resolve an immediate problem or develop a proof of concept. This API may be "cowboyed" into production outside of official security and operational maintenance processes — no vulnerability scan, no consultation with security architects, no Layer 7 protection. In short, no checks and balances. Now you have an API that is not in your API schema. The rogue API is snuck into production and represents a new and unintended attack surface.

Zombie APIs are the APIs that time forgot. These are APIs that were previously valid and approved but were eventually abandoned or replaced by newer versions.

Let's say a development team is doing canary deployments. They create an API that is valid and approved, and a client repeatedly calls it. The development team creates API version 2 and starts sending 1% of that traffic to the newer API. The team watches version 2 to see if there are any problems and, if not, starts sending more traffic to version 2 and less to

1. *GARTNER is a registered trademark and service mark of Gartner, Inc. and/or its affiliates in the U.S. and internationally and is used herein with permission. All rights reserved.*

version 1. Eventually, version 2 gets all the traffic. However, version 1 may still be out there. Its useful life is over, but it has not been deprecated, resulting in a zombie API, as illustrated in figure 1-3.

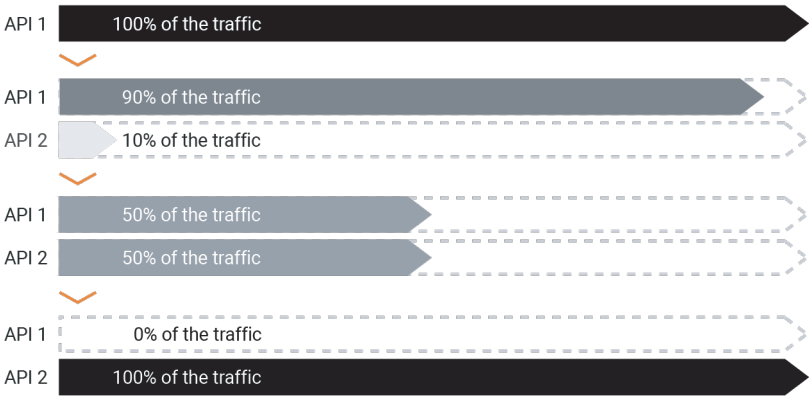

Figure 1-3: Zombie APIs are replaced but not deprecated.

Why attackers target APIs

We know that attackers go where the greatest opportunity lies. Given the huge size of the API landscape, we can logically conclude that APIs have caught attackers' attention. But there's more to it than the sheer quantity of opportunities.

Traditional forms-based web applications were discrete applications. Attackers had to use specific tools and techniques to abuse application functionality. That's not the case with APIs.

By their very nature, public-based APIs expose application logic and (potentially) sensitive data such as personally identifiable information (PII). An attacker can simply imitate an actual user to abuse and exploit legitimate application functionality – no fancy tools or techniques required. (That doesn't stop them from using an arsenal of tools and techniques, however. I'll explain more in Chapter 3.) Here are some examples:

- ☑ If a web application invokes an API via JavaScript, an attacker can simply use a browser's "view source" capability and monitor the network requests made by the script.
- ☑ An attacker can submit invalid data to an API until something breaks.
- ☑ An attacker can catalog system objects, often by their JSON representation, and gain access to objects that they shouldn't be allowed to.

It's easy to see why APIs are a prime attack target: there's a lot of them, and the data is right there for the taking.

Compliance issues

Just a quick word on compliance before we move on to what your developers are and aren't doing to protect APIs. Regulators and standards bodies haven't quite caught on to the severity of API risk. But that doesn't mean they won't. If the scale and scope of API risk don't drive organizations to action, then at some point the threat of regulatory fines will likely do so.

Chapter 2

Exploring the State of API Security

In this chapter

- Understand the role of an API gateway
- Learn how static and dynamic application security testing can and can't improve API security
- Read how common development practices exacerbate API risk

APIs are abundant, and they present a clear and present danger (hat tip to Tom Clancy). So, what is your organization doing about it? Let's peek inside the development process to assess the controls that your organization likely has in place.

Existing API Security Measures

The rapid growth of APIs has outpaced the industry's ability to protect these assets, and that's evident by the technologies organizations are using to secure their APIs.

API management

Developers frequently use an *API gateway* to create and deploy APIs. API gateways focus on the operational aspects that keep APIs running properly. An API gateway sits between a client and a collection of backend services. When a client makes a request, the API gateway breaks it into multiple requests, routes the requests to the right places, and produces a response for the client. API gateways cover a variety of operational tasks such as controlling access, translating a RESTful request to SOAP, or elastically scaling resources if a spike in traffic hits an API.

API operations management includes:

- ☑ Managing known APIs
- ☑ Authenticating API calls
- ☑ Authorizing API use cases
- ☑ Monetizing APIs

API gateways do provide some form of API security. However, the add-on security functionality is very limited and lacks the depth and breadth needed for adequate protection. The security is there because API gateway providers recognize the need, but you don't buy an API gateway for its security functionality.

API gateway security features include:

- ☑ Authentication and encryption
- ☑ Rate limiting
- ☑ Transaction logging
- ☑ Runtime governance policy

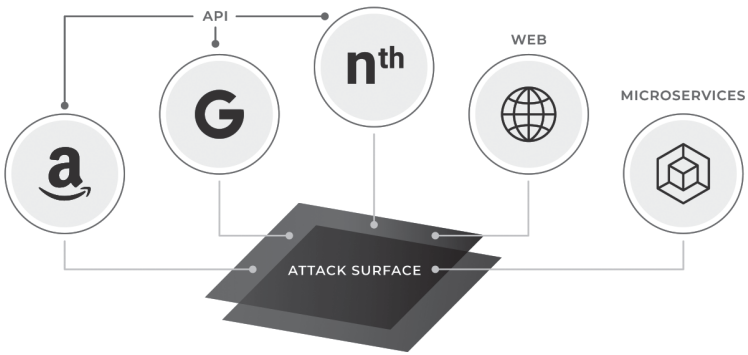

Figure 2-1: As point API gateway solutions accumulate, the attack surface grows.

Organizations commonly use multiple API gateways. Major cloud providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud each offer their own solutions, so as organizations expand into hybrid and multi-cloud deployments, the point solutions accumulate and the attack surface grows, as illustrated in Figure 2-1.

Each API gateway operates in isolation. An AWS gateway has different capabilities from a Microsoft Azure gateway and neither one has visibility into the other. Using multiple gateways increases operational overhead and the risk that policies are inconsistently applied.

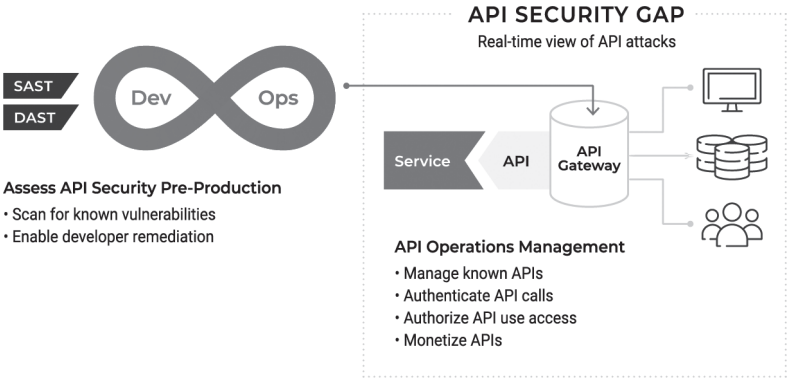

Figure 2-2: Traditional security controls and API gateways leave an API security gap.

Application security scanning

To assess API security pre-production, mature development teams use scanning solutions to run static application security testing (SAST) and dynamic application security testing (DAST) against compiled object code. SAST looks at the internal structure of the application without executing the program. It scans the source code, byte code, or application binaries for vulnerabilities. DAST runs while the application is operating and tries to manipulate it to uncover security vulnerabilities that expose the application to an attack. Both SAST and DAST are important components of an application security program, as they enable developers to remediate critical vulnerabilities before APIs are deployed into production.

There is, however, one very big caveat: SAST and DAST scanners only detect *known* vulnerabilities, which comprise a small portion of API security risk. Remember, as I explained in Chapter 1, APIs by their very nature expose application logic and sensitive information. A known vulnerability isn't necessary to exploit an API. Furthermore, a SAST or DAST scanner can't detect what appears to be legitimate user behavior executed by attackers in production code.

Legacy web application firewalls

A *web application firewall (WAF)* has become a standard security technology for many organizations nowadays. Legacy WAFs are static, rule-based solutions. They look at individual ingress transactions and make crude, simple decisions — good traffic or bad traffic. A legacy WAF can detect attacks like SQL injection, cross-site scripting (XSS), and cross-site request forgery (XSRF). However, discrete attacks make up only a small portion of API attacks. I'll explain this further in Chapter 3.

Key API Security Considerations for Developers

Defensive coding practices and secure design are critical components of an API security program. Make sure your developers are covering the basics.

API encryption – SSL/TLS encryption should be used for both public and internal (private) APIs to protect against man-in-the-middle attacks, replay attacks, and snooping. For external APIs, the web server can handle encryption, or developers can employ a reverse proxy. For internal APIs, developers can use libraries or a service mesh. A service mesh adds automatic encryption on top of service discovery and routing.

API authentication – Authentication helps protect against XSS and XSRF attacks targeting endpoints. Username and password are not typically passed in API calls. However, an API key or bearer authentication token is passed in the HTTP header or in the JSON body of a RESTful API. Make sure tokens expire regularly to protect against replay attacks. Most enterprises use an internal database or LDAP authentication store for authentication. For highly public APIs, OAuth is also an option.

Where appropriate, developers should also consider client-side authentication. Client certificates

and certificate pinning in the application can prevent man-in-the-middle attacks and ensure that only your application can access the API.

API authorization – Not every user or microservice using your API needs the same access rights. A non-admin user may only need read-only access, while an admin may also need the ability to create, update, or delete records. Authorization restricts the entity's access to only what's required. The best way to address authorization is typically through application logic, but it can also be handled by an API gateway.

Common API Security Issues

Adding to the lack of effective security controls is the fact that APIs are built by developers and developers are human. Humans don't always follow established processes and sometimes they simply make mistakes.

Lack of schema

An *API schema* describes the operations of a RESTful API and provides instructions for developers on how to interact with the API. A schema is metadata, that is, data about data. A schema is supposed to make integration between platforms easier for developers. Developers validate schemas to ensure they define what operations can be performed and are within specification. An API schema is both machine and human readable.

Not all API protocols have easily definable specifications. However, the majority of currently deployed APIs are RESTful APIs, which can be described by an OpenAPI Specification (OAS) and defined in a JSON schema. The schema can be used to map an API's endpoints, input/output parameters, and authentication methods. GraphQL, gRPC, Kafka, and others have different protocols, but the strategy is the same in terms of specifying how the API should be used.

Remember that I said API schemas show developers how to interact with an API? Well, that's true — as long as the organization consistently manages and maintains its schemas. Multiple API gateways, different technology stacks, and plain old lack of security hygiene can all impact schema compliance.

Vulnerabilities in APIs

As applications coded by human developers, APIs are subject to vulnerabilities. In fact, APIs have their very own top 10 list published by the Open Web Application Security Project (OWASP) Foundation. The OWASP API Top 10 (outlined in Figure 2-3) serves as an addendum to the foundation's list of common web application vulnerabilities, the OWASP Top 10. But while the OWASP API Top 10 is a step in the right direction, API vulnerabilities and the attacks used to exploit them are far more complex than can be reflected in a top 10 list.

	OWASP API Security Top 10
1	Broken Object Level Authorization
2	Broken User Authentication
3	Excessive Data Exposure
4	Lack of Resources and Rate Limiting
5	Broken Function Level Authorization
6	Mass Assignment
7	Security Misconfiguration
8	Injection
9	Improper Assets Management
10	Insufficient Logging and Monitoring

Figure 2-3: The OWASP API Top 10 is an addendum to the OWASP Top 10.

In the previous chapter, I explained that APIs are not discrete like a traditional web application. Well, neither are their vulnerabilities. Developers tend to create APIs in isolation, without visibility into how one API impacts another or how they can be used to escalate an attack. And that's exactly what attackers do. An individual API may present low risk, but when used in combination with other APIs, it becomes part

of a larger vulnerability. Attackers chain the vulnerabilities together to create an exploit.

Authentication and authorization flaws

Many of the most common API vulnerabilities are related to authentication and authorization flaws. For example, developers may 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 – who discovers an API that doesn't require 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. They're not doing this to make your job harder. Rather, it's to make their job easier.

The practice of violating the principle of least privilege stems from the well-meaning intention to present information as completely as possible and to future-proof an API, making it applicable for applications that weren't originally envisioned at the time of development. It's much easier and faster to reuse an API than it is to develop a new one. However, providing access to more information than absolutely necessary for the given use case presents a fair amount of risk. The data returned via the API can represent an information leak or otherwise function as an attack vector.

A Secure API Development Checklist

Most security professionals are familiar with the adage, "Complexity is the enemy of security." That concept holds true for APIs. Every feature of an API is a potential attack vector. The simpler an API, the fewer the attack vectors. Following the best practices in this checklist will ensure that your APIs are as simple as possible.

- Use existing standards and conventions to lessen the likelihood of an attacker exploiting a loophole in a custom data format.
- Use a stateless authorization model to decrease the amount of session state transitions that can be potentially hijacked and misdirected for nefarious means by an attacker.
- Deprecate old endpoints to reduce potential attack vectors that adversaries may seek to exploit.
- Push functionality to clients to reduce an API's presentation and display responsibilities and the chance of vulnerabilities.
- Be conservative about returned data to reduce the footholds a determined attacker can use to understand, undermine, and exploit your API.
- Reduce variant resource representations to reduce the risk of an attacker taking advantage of inconsistent parameter handling and/or data handling logic.
- Separate endpoints into read versus write permissions to make it easier to analyze the meaning behind traffic patterns and enforce permission structures.
- Align resource boundaries to permission boundaries to keep attackers from executing a privilege escalation attack.
- Implement generic audit logs to obtain insight into web application activity and address the insufficient logging and monitoring vulnerability identified in the OWASP API Security Top 10.

Chapter 3

Deconstructing API Attacks

In this chapter

- Explore the tools and tactics attackers use to attack APIs
- Understand how attackers evade security controls
- Learn why a traditional web application firewall can't stop an API attack

Over the last couple of decades, attackers have built up a sizeable arsenal of tools, techniques, and knowledge related to common technology stacks and security controls. They put all these resources into play when attacking APIs. As a result, threat actors are more innovative, and their attacks are more advanced, than ever before.

API Attacks Are Sophisticated

API attacks aren't one-and-done, but are complex, multi-step processes, often involving large-scale botnets and denial of service techniques. Attackers take their time, methodically sizing up the target and strategically avoiding detection by using multiple tools and techniques that aren't easily recognized by traditional defenses. You can think of an API attack as an advanced persistent threat on steroids.

Step by step

API attacks are commonly spread out over several coordinated steps, which individually appear innocuous. And because attackers often attempt to abuse the business logic of an application, their input isn't obviously malicious. These multi-

step attacks can be very hard to detect as attackers build upon earlier reconnaissance and information leaks. The attacks can appear to be normal traffic when requests are evaluated individually.

Evasion techniques

When not mimicking normal API behavior, modern attackers disguise their activity. They know traffic is being inspected, so they use evasion techniques to bypass traditional security controls.

One form of evasion is the use of an anonymizer to mask the attacker's location. However, anonymizers are relatively easy to detect, so attackers take it a step further by impersonating a legitimate user with false user agent information. Imagine, as an example, that a Russian attacker spoofs a location to appear to come from the United States. Most security solutions will detect the malicious activity and block or flag the IP address. Attackers know this, so they use multiple IP addresses and vary different elements of the request identifiers. Not only does this method help the attackers evade detection, but it also helps them determine defense thresholds.

Attackers use many other evasion techniques as well. Further, when they combine evasion techniques with various attack types, they make it nearly impossible for a signature-based security solution to identify every combination.

Misdirection

Like a magician, attackers are experts at the art of misdirection. They draw attention over *here* so that the target doesn't notice what's going on over *there*. Ransomware and distributed denial of service (DDoS) attacks are favorite forms of misdirection. While the security team is occupied with shutting down the attack and restoring service, attackers interleave the "real" attack against other assets.

Unfortunately, this is no magic show. Security teams can't afford to ignore what's going on over here any more than they can afford to ignore what's going on over there — that is, assuming they discover the other attack at all. These blended,

mixed-mode attacks are difficult to get a handle on, which is why they so often succeed. Security teams are left playing a never-ending game of whack-a-mole while trying to proactively shore up their security.

Militarized attack patterns

Let's imagine you're responsible for security at a regional financial services institution. A national competitor succumbs to a damaging API attack. Your board of directors is relieved – if attackers are focused on the big guy, then they couldn't possibly care about your small organization – or do they?

Companies or organizations within the same industry are, in fact, at risk of getting caught by the same attacks. The reason is simple: organizations within the same vertical often use the same software and are therefore susceptible to the same vulnerabilities. Attackers may profile one organization and use the knowledge to attack others that deploy a similar tech stack.

Low and slow

Attackers play the long game. They understand where the tripwires are and how much pressure they can sustain before an alarm sounds. Attackers often spend months or more poking around the edges of an organization to see what the thresholds are. As a second phase, they'll meter their attack to come in under that threshold and go after high-profile assets.

Malicious bots

Attackers make their work more effective, more efficient, and more dangerous by using botnets to automate their activities. With botnets, attackers can use multiple IP addresses (sometimes tens of thousands), making the attack that much harder to identify. And bots can mimic human behavior, so they look very much like normal usage of the API. Attackers use bots for everything from general scanning and reconnaissance to attacks like scraping proprietary pricing information from web storefronts.

Attackers know that it's difficult to mitigate bot traffic. Because some bots are not malicious, coarse-grained mitigation efforts can disrupt or degrade the experience for legitimate users. Furthermore, advanced bots that use headless browsers or impersonate legitimate users can easily bypass agent-based detection and fool legacy WAFs and web applications into thinking that the attacking bots are legitimate human users.

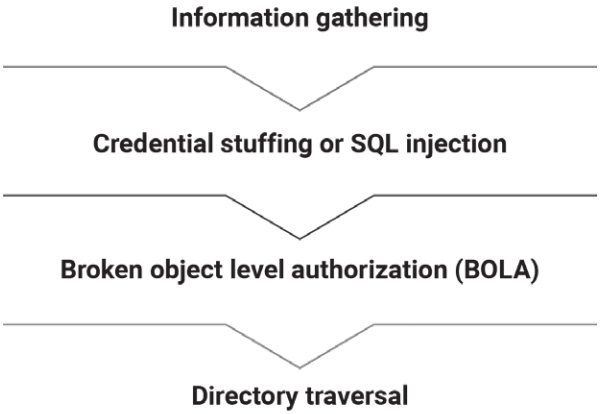

Figure 3-1: Malicious actors combine multiple attacks techniques when exploiting APIs.

Types of API Attacks

If complexity is the enemy of security, it's an attacker's best friend. Attackers take advantage of the complexity of APIs and IT infrastructure to layer different attack techniques to distract security teams and bypass traditional controls. Following are a sampling of these techniques.

Authentication and authorization attacks — These can be at the object, user, or function level. *Broken object level authorization (BOLA)* is the most common attack and is also fairly simple to execute. These attacks attempt to access insecure objects (generally targeting PII or privilege escalation) by manipulating identities or other parameters.

Login attacks — Brute force and credential stuffing are the most common login attacks, but APIs are also susceptible to compromised tokens and API keys. Once an attacker gains access via a login attack, it becomes much harder to detect and stop a wide range of other attacks.

API DoS and DDoS — Most API attacks are not volumetric but rather target specific API vulnerabilities in an effort to consume all memory or CPU services and disrupt service availability. Bots are especially effective at impersonating legitimate users and staying below normal detection thresholds.

The Problem with Legacy WAFs

Now that you understand the complexity of an API attack, let's take another look at web application firewalls (WAFs). As I mentioned in the previous chapter, legacy WAFs provide minimal protection against API attacks.

Because API attacks are typically spread out over several coordinated steps, they can only be recognized when these steps are looked at together. However, the rules in most traditional WAFs are written based on individual events. The WAF looks at each inbound request and tries to match it to a known signature. Most OWASP-type signatures, for example, look for an exact expression match within a request — and it either matches or it doesn't. A multi-step, automated attack makes it difficult for a WAF to detect and block the right requests without causing false positives or writing hundreds, sometimes thousands of rules. A traditional WAF can't protect against flows buried deeper in the application logic, and it won't block malicious activity if it looks like a normal request.

Putting It All Together: The Progression of an Attack

Let's walk through a hypothetical bank breach to see how a successful attack might play out.

One of the attacker's first steps is to gather information and explore the bank's website and its APIs — looking for forms, checking for weaknesses in forms by adding various data and information, and generally trying to discover any holes or vulnerabilities.

The attacker then uses a bot to perform a credential stuffing attack on an API — slowly cycling through a database of stolen usernames and passwords at a pace that won't trigger a legacy WAF rule — until the attacker gets a hit and gains access to an account.

Alternatively, the attacker could try to obtain access by using active enumeration in advance of a SQL injection attack.

Once inside, the attacker can attempt a BOLA by manually testing fields until finding one that allows a transfer of funds.

The attacker empties the account. But that's not the end.

While logged in, the attacker might also use insights from their previous information gathering to carry out a directory traversal attack on a vulnerable check lookup field. Now they can log directly into the application server and have complete control over the system.

Chapter 4

Protecting APIs Is Hard

In this chapter

- Explore the challenges associated with API protection
- Learn about the difficulties of blocking API attacks in real time
- Understand the value of enforcing schema compliance

At this point, you've probably concluded that protecting your APIs isn't going to be easy – and you're right about that. But it's not impossible. You need to get over a few challenges – and this chapter can help.

Finding All of Your APIs

As a security professional, you know this mantra well: you can't protect what you can't see. Well, guess what? The attack surface created by APIs is significantly larger than you probably realize.

The proliferation of APIs across a distributed infrastructure inevitably leads to *API sprawl*. Organizations often don't know how many APIs they have or how they're utilized. Some of those APIs may be known – that is, they were created and are managed via established processes – but organizations often have a host of unknown APIs as well.

Some APIs are never known in a formal sense because they exist outside of security and maintenance processes. Maybe a developer was resolving an immediate problem or developing a proof of concept for a larger project and quickly deployed a new API endpoint to get the work done.

It can be tempting to worry about API security and maintenance later. Thus, you end up with rogue or shadow APIs.

Because they were never established within a standard specification, no one knows how these APIs are supposed to be used or that they even exist.

Similarly, building the next piece of code or revenue-generating application can take priority over sunsetting or deprecating old APIs. Zombie APIs were previously valid and approved but were eventually abandoned or replaced by newer versions. Sometimes these old APIs are intentionally left in production to support legacy versions of an application. But if zombie APIs aren't properly removed, they can leave organizations with outdated, vulnerable points that attackers can exploit.

Organizations also need to be able to find APIs across a variety of environments and tools. Developers stand up APIs behind multiple gateways (and sometimes no gateways) due to different deployments across multiple cloud and on-premises environments.

Bottom line: you have more APIs than you think – and you need to protect them all.

Visualizing the Risk to Your API Attack Surface

Once you have an inventory of all your APIs, you need the ability to understand how each endpoint is being utilized. Everything from failed authentication calls to unexpected methods and parameters can be used to focus detection efforts.

Adding visibility to attack data allows you to better understand which APIs are being targeted and what techniques are being used. Cumulatively, this additional insight allows you to understand the relative risk for your APIs and where you should be focusing your efforts, be it adopting better detection tools and/or addressing potential underlying code issues.

Detecting and Blocking Real-time Attacks

Organizations must be able to detect and block attacks in real time, and this isn't easy to do. It requires real-time scanning of all inbound API traffic. Multi-step attacks involving bots that mimic human behavior can be very difficult to detect. Traffic must be analyzed well enough to separate legitimate from illegitimate traffic without impacting application performance. Because attackers use everything in their arsenal while exploiting legitimate application functionality, it requires sophistication to detect tactics such as:

- ☑ The distribution of attack steps across multiple IP addresses
- ☑ Advanced credential stuffing
- ☑ API DoS and DDoS attacks
- ☑ Sensitive data disclosure mining
- ☑ Exploiting authorization/authentication vulnerabilities
- ☑ Exploiting security misconfigurations

In addition, when you are actively scanning real-time traffic, you can see all APIs that are getting traffic, even those you didn't know existed.

It may seem obvious, but it's worth stating that the ability to analyze and block an attack in real time requires a tool that sits online. Tools that perform offline analysis or simply push data over to another system like a traditional web application firewall (WAF) can't stop an attack in real time.

Simply blocking the OWASP API Top 10 vulnerabilities won't deliver sufficient protection, either. APIs provide a multitude of new paths to the back end of an application, and in many cases, these paths could even be east-west connections between modules and microservices.

Additionally, attacks can be obscured with API-specific protocols like JSON. Your organization needs visibility into

the right traffic and the native ability to decode the relevant protocols. Otherwise, even the threats in the OWASP API Top 10 will pass you by.

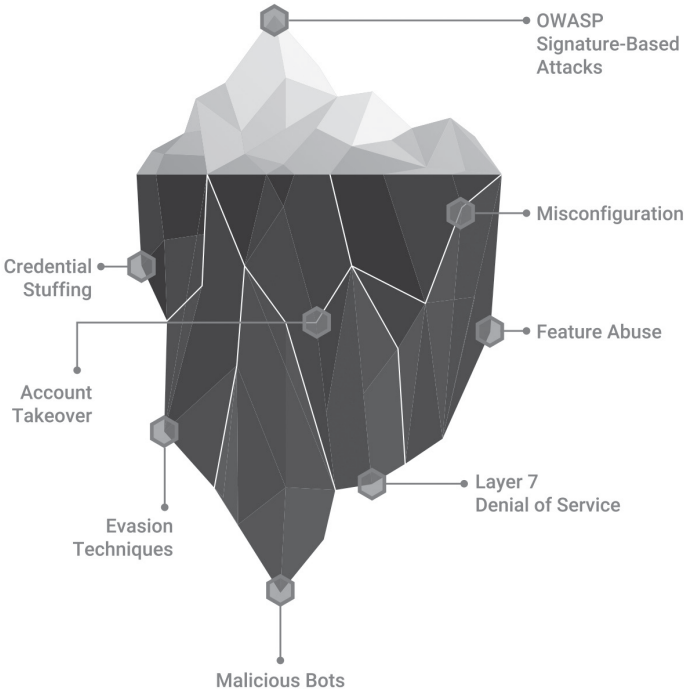

Figure 4-1: Signature-based attacks are just the tip of the iceberg.

Exposing Insights to Enable Attack Forensics

Vendors that rely on rules-based blocking (and that's most of them) aren't doing their customers any favors. Not only is it difficult to determine the general efficacy of these solutions, but they don't provide enough information during a major attack to enable organizations to respond quickly. Without adequate information, teams can't build policies to protect themselves.

While these products may reduce security risk for basic attacks, they don't do anything to improve an organization's

security posture. So, you're essentially running on a treadmill, dodging attacks here and there but never making any progress in reducing your overall risk.

Security teams need insights into whether malicious traffic should be a concern and if it dictates additional security measures or hardening. Security teams need to understand the level of sophistication and characteristics of attackers. They need insights to answer questions like:

- ☑ Are attackers executing high-volume (spray and pray) or low and slow attacks leveraging highly targeted techniques?
- ☑ How persistent and perseverant are the attackers?
- ☑ Are certain APIs more vulnerable due to external interest, poor security hygiene in development, or highly vulnerable technologies?
- ☑ Are peer organizations seeing similar attack patterns and techniques, too?

All this data can be used to improve the organization's security posture by helping teams make better-informed decisions about where to focus their time and effort to reduce their API risk.

Assessing and Enforcing API Schema Compliance

An API schema can help protect APIs by defining how they can be utilized. There's just one caveat: developers must comply with the schema for it to be effective. Oftentimes, developers don't follow the schema, or a schema doesn't exist at all. As a result, real-world API usage is very different from expectations.

Ideally, organizations should have a way to assess and enforce API schema compliance to help with their API protection efforts. But before this can happen, organizations need discovery capabilities to inventory their APIs and then assessment capabilities to compare those APIs against either a schema or a customized format.

IT Team Gets Nights and Weekends Back with ThreatX at the Helm

The IT team for an Oregon-based credit union knows firsthand just how hard it is to protect APIs. The team, which is responsible for delivering best-in-class services to 143,000 members, was diligently working nights and weekends to protect its APIs and sensitive data. When the COVID-19 pandemic triggered an uptick in cyberattacks against financial services institutions everywhere, the team knew its manual approach wasn't sustainable.

The IT team's legacy security solution was only getting them so far. When they identified unusual user behavior that could potentially be malicious, they'd pull IP addresses out of the suspected database queries, put them into Excel spreadsheets, remove any duplicates, and then correlate the IP addresses with other information to validate their suspicions. If the behavior was indeed malicious, they'd write a rule to block the behavior going forward. For example, multiple logins with different usernames from the same IP address would be something to block. Ultimately, this approach required the team to create security rules for each identified IP address in the security solution. Despite all this work, the organization was still at risk, because even this detailed and time-consuming process couldn't account for ISPs that mask their separate customer IPs behind a single IP.

In its quest to deliver services without downtime or risk, the credit union evaluated several WAF vendors and bot management services. In addition, the organization pursued next-generation API protection vendors. One of those was ThreatX.

ThreatX provides an API and web application protection platform to defend against complex, multi-mode attacks, including massive botnets and DDoS. Rather than simply identifying attack signatures such as suspect IP addresses, the ThreatX platform monitors and tracks user behavior, looking for suspicious patterns over time. As a result, ThreatX is much more accurate than legacy WAFs – and much less likely to block legitimate traffic.

The credit union was up and running on the ThreatX platform within an hour. Using ThreatX's behavior-based analysis, the security team was actively identifying and blocking suspicious users. The team determined that the other solutions couldn't match the level of preventative measures and efficacy delivered by ThreatX.

"ThreatX has been a game changer for my team and me, and has provided an additional layer of security for our members," says the director of IT. "We don't have to be watching it all the time. It doesn't occur to me to check it over the weekend. We were all taking turns working evenings and weekends, and the need for that stopped."

Chapter 5

Solving API Security Challenges

In this chapter

- Learn the key capability required to overcome API protection challenges
- Read about the cost of taking API protection offline
- Understand how risk scoring enables a dynamic response to suspicious behavior

Clearly, a rules-based solution doesn't suffice when it comes to API protection. Organizations need a solution that is as sophisticated as the attacks they're trying to protect against. Fortunately, just like attackers, organizations can have an arsenal of their own.

The Foundational API Protection Capability

Using advanced technologies like machine learning (ML) and artificial intelligence (AI), organizations can beat attackers at their own game.

Attacker-centric behavior analysis

API protection begins with *attacker-centric behavioral analytics,* the identification and correlation of attacker behavior across multiple attack vectors to identify threats more precisely. By tracking and analyzing the behavior of attackers in real time and over time, organizations can obtain a much more comprehensive and precise view of risk – both immediately and through low and slow attacks over time.

Using attacker-centric analysis, an API protection solution can pinpoint malicious behavior, identify the type of threat that's causing the problem, and determine who/what needs to be blocked in real time. The solution can track the progression of an attacker or a campaign across the kill chain and identify techniques, tools, and procedures (TTPs). This level of behavioral insight provides the most comprehensive defense against API attacks.

The Cost of Going Offline

Some API protection tools collect API utilization data and then take it offline for analysis. These offline API protection products can develop sophisticated correlation rules to detect complex attacks — but at a considerable cost.

First and foremost, taking traffic analysis offline eliminates the possibility of real-time detection and response. It puts organizations in a reactionary position. The need to collect logs, ship them offline, perform the analysis, and write a rule extends the time to detect an attack.

Once a rule is developed, it must be applied to traffic to stop the attack. With an offline solution, the rule must be fed into a blocking system, increasing your time to respond. Even if the systems are integrated, a traditional WAF or API gateway may not understand the rule and require it to be converted to another format – further delaying your response. When all of this is said and done, the attack vector may be too complex to block with legacy technology.

Since real-time blocking by an offline solution is out of the question, it should come as no surprise that interactive responses are also off the table. For all their complexity, offline solutions can't take advantage of real-time techniques like IP interrogation and fingerprinting, active deception, or user-level tarpitting/rate limiting.

API Protection in Action

As the foundation of API protection, attacker-centric behavior analysis addresses the challenges and requirements outlined in Chapter 4 to provide a holistic solution. Let's take another look at those capabilities within the context of attacker-centric behavior analysis.

API discovery

To be effective, attacker-centric behavioral analytics requires all traffic to be analyzed in real time. This is the only way to ensure that the system has all the potential pieces of an attack puzzle and can therefore make accurate decisions. That real-time analysis of traffic hitting endpoints is also the best mode of API discovery. The solution can find all your rogue and zombie APIs across a distributed infrastructure, behind any API gateway.

But that's not all! Analytics can also identify which endpoints are no longer used, and which clients are still actively using old endpoints. As part of the discovery, the system can gather key metrics such as request counts, methods, parameters, and error codes, plus how APIs are utilized and how they're being invoked.

Visibility of the attack surface

Real-time traffic analysis generates a significant amount of data. This data can be used to visualize the entirety of the API attack surface and observe the utilization of APIs both in real time and over time.

The data can also be presented via a dashboard that provides a centralized view of how and where APIs may be deployed. In addition, the dashboard can highlight the most important data, including:

- ☑ endpoints with potential vulnerabilities
- ☑ abnormal traffic and utilization
- ☑ malicious attackers, the techniques they utilized, and what they targeted
- ☑ prioritized issues with visualizations that convey risk levels

Detecting and blocking attacks

An attacker-centric approach to identifying and correlating activity makes it possible to identify API threats more precisely and to understand the context behind them. Security

teams gain the insight to monitor and block suspicious behavior and mount a more holistic defense. Analyzing behaviors from multiple vantage points enables organizations to block a suspicious entity without relying on a single event or attack signature – but there's more to it than that.

Risk scoring

Attacker-centric behavioral analytics gives organizations more flexibility in how they respond to an attack. It allows for adaptive enforcement and action in which attackers are systematically confronted to understand their intent and gain further insight into the risk they pose.

Continuously monitoring attacker behavior and correlating that data over time can produce a unified risk score that increases or decreases based on the attacker's behavior. You can see and respond to attack patterns over time and adjust to the attacker's behavior with the following tactics:

- ☑ ***IP interrogation and fingerprinting*** — The ability to transparently present challenges and data-gathering techniques to determine human versus non-human users and create an identification profile used to track user activity across multiple IP addresses.
- ☑ ***Active deception*** — The ability to transparently present atypical objects and responses to tempt attackers into revealing themselves.
- ☑ ***Tarpitting/rate limiting*** — The ability to artificially add delayed responses to a specific user suspected of malicious intent (DDoS, data leakage exploitation, etc.)

You can also block suspicious entities and IP addresses when behaviors surpass an acceptable risk threshold.

Risk scoring also enables organizations to utilize flexible blocking modes that are designed to block malicious requests and deter suspicious entities from attacking a site, while allowing benign traffic and real users through. Risk scoring can also reduce false positives. For example, because an IP address currently exhibiting suspicious or malicious behavior

may later be reassigned to a legitimate user, each entity can be allowed a certain number of opportunities to exhibit questionable behavior before they're permanently blocked.

Valuable insights for forensics

With the abundance of data generated via attacker-centric behavior analysis, organizations can glean valuable insights for forensics. An API protection platform can identify key attributes of an attack, such as attack patterns over time (e.g., low and slow), the use of advanced evasion techniques, and details on the attack target. These insights allow security teams to understand the goals and nature of a threat, in turn enabling them to adopt a more comprehensive security strategy that reduces future risk.

Enforcing API schema compliance

As I mentioned in Chapter 2, organizations often struggle to consistently manage and maintain API schemas. The best solutions include API discovery based on real traffic and the ability to compare traffic metrics against a security schema based on integration of legitimate schema files, or to build a customized version on the fly based on actual traffic.

Not only is a specification a best practice for API design and implementation, it's also a great way to evaluate API utilization and ensure it conforms to the specification. One of the best indicators of malicious activity is attempted misuse of an API. Attackers are constantly probing API endpoints to identify any number of vulnerabilities that can be exploited. Comparing actual utilization against what's expected is one of the best ways to identify malicious users and their techniques. Given that many sophisticated attacks are distributed across hundreds, if not thousands, of IP addresses, detection often requires complex behavior analysis, and mitigation calls for tight coupling of this analysis with blocking capabilities.

Global Restaurant Chain Beats Attackers at Their Own Game

When the COVID-19 pandemic forced folks to go into isolation, restaurants everywhere shifted their business model to takeout only and started conducting the majority of their sales online. For one global restaurant chain, this shift also marked the start of relentless cyberattacks.

In addition to launching volumetric attacks against the restaurant chain, attackers targeted long-running queries to consume server resources — both against the chain's website and against the APIs that powered the rewards program on their mobile app. Initially, the restaurant was getting hit by 30,000 to 40,000 IP addresses at a time, which was enough to impact availability. The pattern was a combined attack where attackers used a slow version of brute force/credential stuffing to gain access to a customer account. From there, the attackers used multiple techniques, including privilege escalation and enumeration to modify rewards credits.

Once the attackers got in, they had their run of the system. The security team would eventually discover the discrepancies, deactivate the offending accounts, and force password changes. Unfortunately, this would simply restart the cycle with new DoS attacks followed by account takeover attempts.

It wasn't until the restaurant partnered with ThreatX for its attacker-centric behavioral analytics that it could keep up with the attackers. Only then could the chain use risk-based blocking to block IP addresses fast enough and render the attacks ineffective. At one point, the attackers were using well over 100,000 IP addresses at any given moment, but ThreatX was blocking them so fast that it was starting to become expensive for attackers to maintain the assault. Eventually the attackers gave up, but not without lying low for a few weeks and then attempting to come back in under the radar.

After partnering with ThreatX the restaurant chain was able to prioritize its security efforts by focusing on actual attacks, not just attackers testing out various techniques or creating distractions. The restaurant chain can now track attacker behavior over time and recognize a real attack that warrants attention versus attacker reconnaissance.

Chapter 6

Evaluating API Protection Solutions

In this chapter

- Review the criteria for a modern API protection solution
- Understand what capabilities organizations need beyond threat detection and blocking
- Learn how to combine API protection with web application protection

APIs are under siege, and API gateways and legacy technologies are not designed to protect against complex, multi-step, high-volume attacks. Organizations need an API protection solution that matches the sophistication of these attacks and leverages a holistic approach to detection and response.

10 Criteria for an API Protection Solution

Effective API protection brings together complementary technologies that deliver a highly accurate, reliable, and complete approach to protecting applications against the full breadth of security threats. Here's what you need to think about when evaluating API protection solutions.

Behavioral analytics

As I explained in Chapter 5, attacker-centric behavioral analytics is the key to minimizing false positives while effectively stopping API-based attacks. Attacker-centric behavior analysis enables organizations to uncover the entirety of an

attack – not just what attackers want you to see – so that you can protect and defend on all fronts. A solution that leverages attacker-centric behavior analysis can also identify and track malicious identities even as traits such as IP addresses and user agents change.

API discovery and profiling

An API protection solution should include profiling and discovery capabilities. Look for a solution that automatically discovers API endpoints without requiring you to lift a finger. The solution should also profile the tech stack and endpoint responses to indicate operational health. This will ensure that APIs are continually protected, including those that developers roll out on the fly, while enabling you to identify shadow APIs that have been deployed and zombie APIs that need to be deprecated.

In addition, look for a solution that helps facilitate schema compliance. This capability enables organizations to compare API traffic to specifications to determine whether compliance gaps exist so that they can work to mitigate them. It also makes it easier to identify whether an API call is legitimate or malicious.

Integrated, real-time detection and blocking

API protection must deliver active, integrated, real-time attack blocking – before damage is done. Immediately upon detecting a threat, the solution should block it. (Note: API observability *does not* equal real-time protection.) And yet you'll likely come across solutions that essentially serve as anomaly detection tools, feeding data to another tool, such as a WAF, for blocking. This arrangement not only precludes real-time capabilities, but it also presumes that static rules and a signature engine are sufficient for blocking an API attack. But this thinking is flawed. Attackers don't use simple attacks that fit the model for static rules. Sure, you can block a suspicious IP address, but attackers know this and will morph the attack accordingly.

Fully managed policies, attack defense, and threat analysis

A holistic API protection solution does much more than identify and block attacks. It provides the policies, attack defense, and threat analysis required to proactively improve your security posture and reduce your overall risk. For example, a modern protection platform provides virtual patching, a highly effective defense-in-depth approach to newly identified vulnerabilities and exploits that can be completed in hours or even minutes.

Complete visibility

Visibility is table stakes. Look for an API protection solution that provides real-time visibility into all the threat activity your APIs face. An attack dashboard should visualize both malicious and benign traffic over time and allow security teams to drill down into the data and perform deep inspection of attacking entities and responsive actions taken by the solution.

Flexible deployment options

Given the complexity and ever-changing nature of today's IT environments, look for an API protection provider that offers flexible deployment options. A container-based deployment allows you to deploy images in the cloud or on premises. Whether you're managing the deployment yourself or the provider is hosting it, make sure the provider can deliver visibility and protection across a heterogeneous environment.

An API protection solution is normally deployed on the network edge to look at outbound traffic because that's where most of the risk lies. However, consider a solution that can also be deployed to analyze and block east-west traffic to protect private APIs. Even here, you need the ability to block traffic as opposed to collecting traffic and sending the data to another solution.

DDoS and bot detection

Distributed denial of service (DDoS) and bots come standard with most API attacks. They should therefore be addressed by an API protection solution. Look for a solution that uses

multiple detection techniques and a combination of automated challenges, IP interrogation, and tarpitting to ensure long-running queries, HTTP floods, and other Layer 7 attacks are mitigated quickly and appropriately without impacting valid users.

Integrated application protection

Attackers don't operate in silos – attacking discrete portions of an application but not others. Likewise, API protection should extend to the broader application portfolio, eliminating the need for yet another point solution. Look for a solution that is built to protect your web applications and your APIs from all threats.

Agentless deployment

It's time to say goodbye to agents. They're simply too cumbersome for today's highly dynamic, distributed IT environments. Look for an API protection solution with an agentless deployment. This will support AppSec and DevOps teams without locking either into architectural decisions or sacrificing their autonomy or flexibility. An agentless architecture also ensures that there is no need to disrupt either your applications or your operations. Deployment is fast and provides complete coverage with no downtime.

Glossary

account takeover: a form of cyberattack in which the attacker gains unauthorized access to a user account and leverages it to commit fraud or steal data.

active deception: the ability to transparently present atypical objects and responses to tempt attackers into revealing themselves.

API gateway: a software tool used to manage API operations, such as controlling access, translating requests to different protocols, or elastically scaling resources during a traffic spike.

application programming interface (API): an application that enables two or more software components to communicate with each other using a specified set of definitions and protocols.

attacker-centric behavioral analytics: the identification and correlation of attacker behavior across multiple attack vectors to identify threats more precisely.

broken object level authorization (BOLA): A common and simple authentication and authorization attack in which the attacker attempts to access insecure objects by manipulating identities or other parameters.

credential stuffing: a type of cyberattack in which stolen account credentials are systematically tried against other user accounts to gain access.

GraphQL: an open-source data query and manipulation language and server-side runtime for APIs developed by Facebook as an alternative to RESTful APIs.

Google Remote Procedure Call (gRPC): an open-source remote procedure call (RPC) system developed by Google that runs in any environment and generates cross-platform client and server bindings for many languages.

IP interrogation and fingerprinting: the ability to transparently present challenges and data-gathering techniques to determine human versus non-human users and create an

identification profile used to track user activity across multiple IP addresses.

OpenAPI: a technical specification used to describe RESTful APIs in a way that is both machine and human readable.

OWASP API Top 10: a prioritized list of the most critical API security issues published by the Open Web Application Security Project (OWASP).

principle of least privilege: the best practice of granting an entity access to the minimum resources required to complete a task.

RESTful API: an API built on the representational state transfer (REST) architecture, which uses a client/server design to separate the front end and back end of an API.

rogue APIs: APIs that exist outside of an organization's official security and operational maintenance processes and are therefore unknown (also called shadow APIs).

schema: a description of how a RESTful API operates, along with instructions on how to interact with the API.

simple object access protocol (SOAP) API: a highly structured, clearly defined messaging standard that uses XML.

tarpitting/rate limiting: the ability to artificially add delayed responses to a specific user suspected of malicious intent (DDoS, data leakage exploitation, etc.).

web application firewall (WAF): a type of firewall designed to filter, monitor, and block HTTP traffic as a means of protecting web applications.

zombie APIs: previously valid and approved APIs that have been abandoned or replaced by newer versions and are no longer actively maintained.

SRC CYBER SOLUTIONS LLP

THREATX

Protecting APIs is Really (Really) Hard

We should know. We wrote the book on API Protection.

We've Got You Covered

ThreatX's API protection platform makes the world safer by protecting APIs from all threats, including DDoS attempts, BOT attacks, API abuse, exploitations of known vulnerabilities, and zero-day attacks. Its multi-layered detection capabilities accurately identify malicious actors and dynamically initiate appropriate action. ThreatX effectively and efficiently protects APIs for companies in every industry across the globe.

- **DETECT & BLOCK**
  real-time attacks on APIs
- **EXPOSE**
  insights to enable attack forensics
- **DISCOVER & DEFEND**
  APIs against future threats
- **VISUALIZE**
  risk to your API attack surface
- **ASSESS & ENFORCE**
  API schema compliance

THREATX

learn more at threatx.com/request-a-demo

Discover how to detect and block API-based attacks in real time while reducing false positives and enabling developers to work at the speed of business.

APIs have become the de facto standard for modern computing – and a rapidly growing attack surface that is largely unprotected. Real-time API protection is critical for preserving an organization's ability to innovate, but it requires more than an API gateway or legacy web application firewall. This guide will show you how to protect APIs with a sophisticated defense.

- Understanding APIs — review what APIs are and why the API economy is taking off
- Exploring API Security — understand why API gateways and legacy web application firewalls fail to protect APIs
- Deconstructing API Attacks — explore the intricacies of an API attack
- Protecting APIs Is Hard — examine the challenges of detecting and stopping API attacks in real time
- Solving API Security Challenges — understand how attacker-centric behavior analysis enables API protection
- Evaluating API Protection Solutions — learn the 10 criteria for an effective API protection solution

About the Author

A former editor of SearchSecurity.com, Crystal Bedell is a senior marketing consultant specializing in cybersecurity. She's been helping technology providers create engaging content since 2000.

CYBEREDGE PRESS

Not for resale

ISBN 978-1-948939-33-1