State of API Exposure 2024

Fortune 1000 at Risk: How we discovered 30,000 exposed APIs & 100,000 API issues in the world's largest organizations

escape

We analyzed the domains of the world's largest organizations included in the Fortune 1000 and CAC 40. Here's what we found

30,784

Exposed APIs

3,945

Development APIs exposed

1,816

secrets accessible in API services

2,038

Highly critical vulnerabilities

Key Takeaways

"Scaling API security is a fundamental challenge. As organizations deploy more APIs to meet digital demands, their security processes are falling behind. Our research shows that a majority of APIs are left unmanaged, which not only exposes data, but also magnifies risk at every level of operation."

Tristan Kalos, CEO of Escape

The State of API Exposure 2024 report provides an in-depth analysis of the significant risks associated with exposed APIs, highlighting vulnerabilities across large organizations, including Fortune 1000 companies and CAC 40 entities. Key findings reveal the pervasive nature of API security issues and the need for improved security measures:

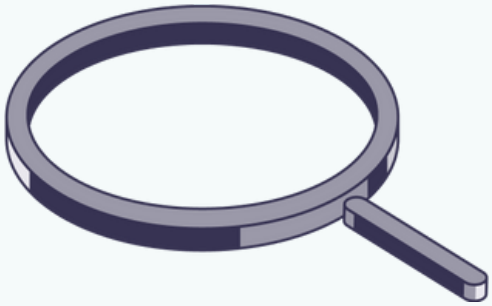
1,834 Highly Critical Vulnerabilities in Fortune 1000 Companies: Among the identified vulnerabilities, 1834 were classified as highly critical, directly affecting Fortune 1000 companies. These critical vulnerabilities, often associated with broken authentication and configuration errors, place essential systems and sensitive data at significant risk of exposure and exploitation. Critical issues span various sectors, impacting tech platforms, financial companies, insurance, healthcare, tech, and others.

Development APIs: Nearly 4,000 development APIs were publicly accessible, often lacking adequate security controls. Exposing these APIs can inadvertently reveal sensitive information and offer attackers potential entry points.

Exposed Secrets: The report found 1,816 highly sensitive secrets exposed within API environments, including access tokens, API keys, and authentication credentials. These exposed secrets significantly heighten the risk of unauthorized access and potential misuse of critical systems.

Call to action: We recommend to start auditing all APIs—particularly shadow and legacy APIs—to identify and secure vulnerabilities, and restricting access to development APIs with production-level security standards. Implementing continuous monitoring and scanning tools is crucial to detect risks early, while ensuring that all secrets are encrypted and securely managed to prevent exposure.

Contents



ARE WE IGNORING HIDDEN DANGERS?

10%

of IT organizations fully document their APIs*

$31B

estimated losses due to API breaches***

57%

of organizations suffered an API-related data breach in the past two years**

THE CONSEQUENCES OF UNSECURED APIS EXPOSED IN THE WILD

API sprawl has emerged as a significant challenge in recent years. As organizations increasingly rely on APIs to fuel digital transformation, connect services, and deliver data-driven experiences, the sheer number of APIs in production has skyrocketed.

As we move through 2024, the exponential growth of APIs presents new challenges. According to recent Gartner's market guide, APIs - especially shadow and dormant ones - are causing data breaches among organizations that, on average, **exceed the magnitude of other breaches**.

Since 2022, at least 190M sensitive data records have been breached. In our previous research, we estimate that enterprise companies lost $31B due to breaches.

Many APIs are pushed to production too soon, often bypassing critical security testing stages or, in some cases, being tested only after deployment rather than within development environments.

This rush to release frequently leads to gaps in security coverage, allowing untested or under-tested endpoints to expose sensitive information or become vulnerable to attack.

Another significant factor is the proliferation of "shadow APIs"—APIs that exist outside the knowledge or management of security teams. These undocumented, unmanaged, or abandoned APIs often lack essential security protocols, making them especially prone to exploitation. Shadow APIs can emerge from development shortcuts, legacy applications, or third-party integrations that slip through regular security oversight, creating hidden risks within the system's architecture.

Given these trends, API security deserves greater attention. It is now widely recognized as a critical challenge requiring stringent security management and thorough testing before APIs are released into production environments. By analyzing Fortune 1000 and CAC 40 exposed API services we're here to prove it.

escape

*according to a 2023 report from Enterprise Management Associates (EMA)

**according to the 2025 Global State of API Security report

***according to Escape's API Threat Landscape Report

2024'S CASES OF VULNERABLE APIS EXPOSED IN THE WILD

In May 2024, Dell experienced a significant data breach when a threat actor exploited an unsecured API endpoint on a partner portal. This vulnerability allowed unauthorized access to approximately 49 million customer records, including names, physical addresses, and order information. The breach highlighted the risks associated with insufficient API security measures and the potential for large-scale data exposure.

In July 2024, Twilio's Authy service suffered a significant breach due to an exposed API endpoint. This vulnerability allowed unauthorized access to authentication data, putting millions of users at risk. The attackers managed to exploit this unsecured endpoint to access one-time passcodes, which are a critical layer of security for multi-factor authentication. This breach highlighted how even security-focused companies are vulnerable when API endpoints aren't adequately protected.

A vulnerability in Trello's API configuration led to a massive data leak in January 2024, where over 15 million user records were exposed on a dark web forum. Trello, a popular project management platform, had an API endpoint that inadvertently allowed open access to sensitive user data, including project details, personal information, and task management records. Hackers were able to scrape this data due to misconfigurations in Trello's API permissions, underscoring how quickly API flaws can result in serious data leaks.

In July 2024, German security expert Lilith Wittmann discovered an unprotected API from Deutsche Telekom, that could be used, to retrieve details about landline connections via their internet access. While the exact number of affected users was not disclosed, it exposed sensitive user data such as names, email addresses, and service usage details. The open API lacked adequate access controls, allowing unauthorized parties to access this information. This incident highlights the critical need for robust API governance to prevent data exposure and protect customer privacy.

IN 2024, DEUTSCHE TELEKOM REPORTED APPROXIMATELY 259M MOBILE CUSTOMERS WORLDWIDE

escape

Methodology

Data Gathering Strategy

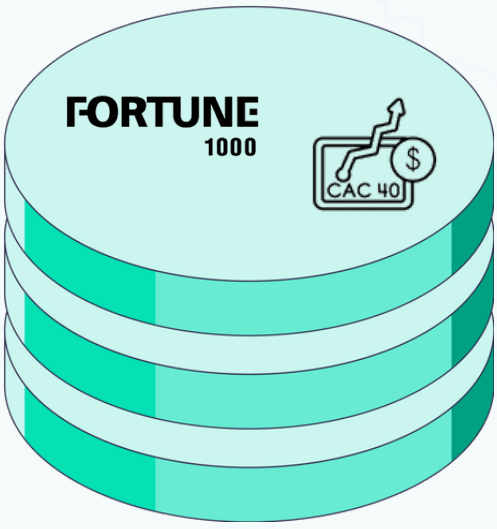
In our comprehensive analysis, we aimed to examine API security measures across a broad spectrum of domains. We selected domains from two primary sources:

- Fortune 1000: A list of the 1000 largest public companies in the United States, ranked by revenue. We excluded several very large tech companies like Amazon, Google, and Meta to avoid skewing the results.
- CAC 40: A capitalization-weighted index of the 40 most significant stocks among the top 100 companies by market cap on Euronext Paris.

While this approach provided a substantial data set, we recognize potential biases. Larger domains with extensive resources may employ stronger security measures, potentially leading to fewer vulnerable APIs. However, our study focused on the largest American companies without specifically accounting for this bias.

Alternatively, with over 365 million domain names reported across the internet, our sample size becomes relatively small, potentially leading to greater volatility in the number of findings.

The data collection was a one-time process. During the collection, we encountered several limitations. To respect legal and ethical boundaries, we deliberately excluded certain types of domains. This included governmental, educational, and health-related domains, as regular users are not typically authorized to explore these. This decision ensured that our study aligned with ethical norms for web crawling and data collection practices, prioritizing responsible research standards.



Methodology

In-Depth API Discovery Process

After finalizing our domain list, we gradually added these domains into Escape's platform to initiate a comprehensive scan of exposed APIs. We examined not only the primary domains but also dived into the numerous subdomains associated with each. This approach enabled us to achieve a thorough and granular discovery process.

~ 1K

unique top-level domains analyzed

Escape uses a sophisticated combination of techniques to identify and inventory APIs by scanning exposed source code:

Subdomain enumeration
Escape begins by performing subdomain enumeration. This process involves scanning for all subdomains associated with the main domain you previously entered into the platform. Subdomains often host APIs or services that may not be immediately apparent. By identifying these subdomains, Escape can uncover additional endpoints that might otherwise be missed. This initial step lays the foundation for a comprehensive discovery process.

AI-powered fingerprinting
Once subdomains are identified, Escape employs AI-powered fingerprinting to recognize and classify the APIs. Fingerprinting involves analyzing various characteristics of the APIs, such as their structure, endpoints, and response patterns. The AI algorithms used by Escape can detect and categorize different API types (REST, GraphQL, gRPC) with high accuracy. This machine learning-based approach ensures that APIs are identified and classified correctly, even if they have unique or non-standard configurations.

OSINT techniques
Escape also leverages Open Source Intelligence (OSINT) techniques. OSINT involves gathering and analyzing publicly available information to enhance the discovery process. By examining code repositories, documentation, and other public resources, Escape can identify additional API endpoints and services. This technique helps in discovering APIs that are not directly exposed but can still be found through public information.

Through this multi-layered process, we discovered **158,079 subdomains,** allowing for extensive coverage and a highly detailed analysis. This broad scope provided a deep view into the exposure and security practices around APIs across various industries, offering critical insights into the current landscape of API security and vulnerability.

Methodology

Automated API Specification Generation

4.5K

API specifications found in the wild

29.7K

API specifications programmatically generated

One of the most challenging aspects of our study was ensuring we had API specifications to effectively scan newly discovered exposed API services for vulnerabilities.

Having an OpenAPI Specification (OAS) is particularly beneficial as it provides a standardized, machine-readable format for documenting RESTful APIs, promoting clarity and consistency across services.

Through our initial scan, we located 4,547 exposed API specifications, so we had to generate most of the specifications ourselves. This process involved parsing the Abstract Syntax Tree (AST) from the code to create dynamically detailed and accurate API specifications.

Luckily, Large Language Models (LLMs) have recently become extremely good at analyzing and generating code. Moreover, they show great performance across a wide variety of code languages, frameworks, and coding styles, which is exactly what we want for framework and language-agnostic OAS generation software. Finally, LLMs can also leverage information in code comments, which the traditional static analysis approach cannot do.

In our current approach, **integrated into Escape's platform**, we focused on two key areas:

- **Semantic Analysis:** We identify essential code fragments using custom rules (e.g., specific Semgrep patterns), optimizing the data sent to the LLM and enhancing prompt quality.
- **Specification Generation**: The LLM processes each identified fragment to generate precise OAS methods, with contextualization ensuring accuracy by resolving dependencies and references within the code.

This method enables Escape to not only generate API documentation but also to continuously monitor and detect any changes or versions in the API documentation over time.

Methodology

Final step - API Security Scanning

30,784

API services scanned
for vulnerabilities

After the comprehensive specification generation process, the final step is API security scanning. Using Escape's Dynamic Application Security Testing (DAST) solution, we conducted in-depth analysis of each identified API endpoint to detect potential vulnerabilities and risks.

Escape's DAST approach is specifically designed for API security. Unlike traditional web application DAST tools, which are often limited in scope and primarily test web applications at the surface level, Escape's DAST is purpose-built to handle the unique requirements and complexities of APIs. This makes it exceptionally effective at identifying security flaws in API configurations, authentication, authorization, and more.

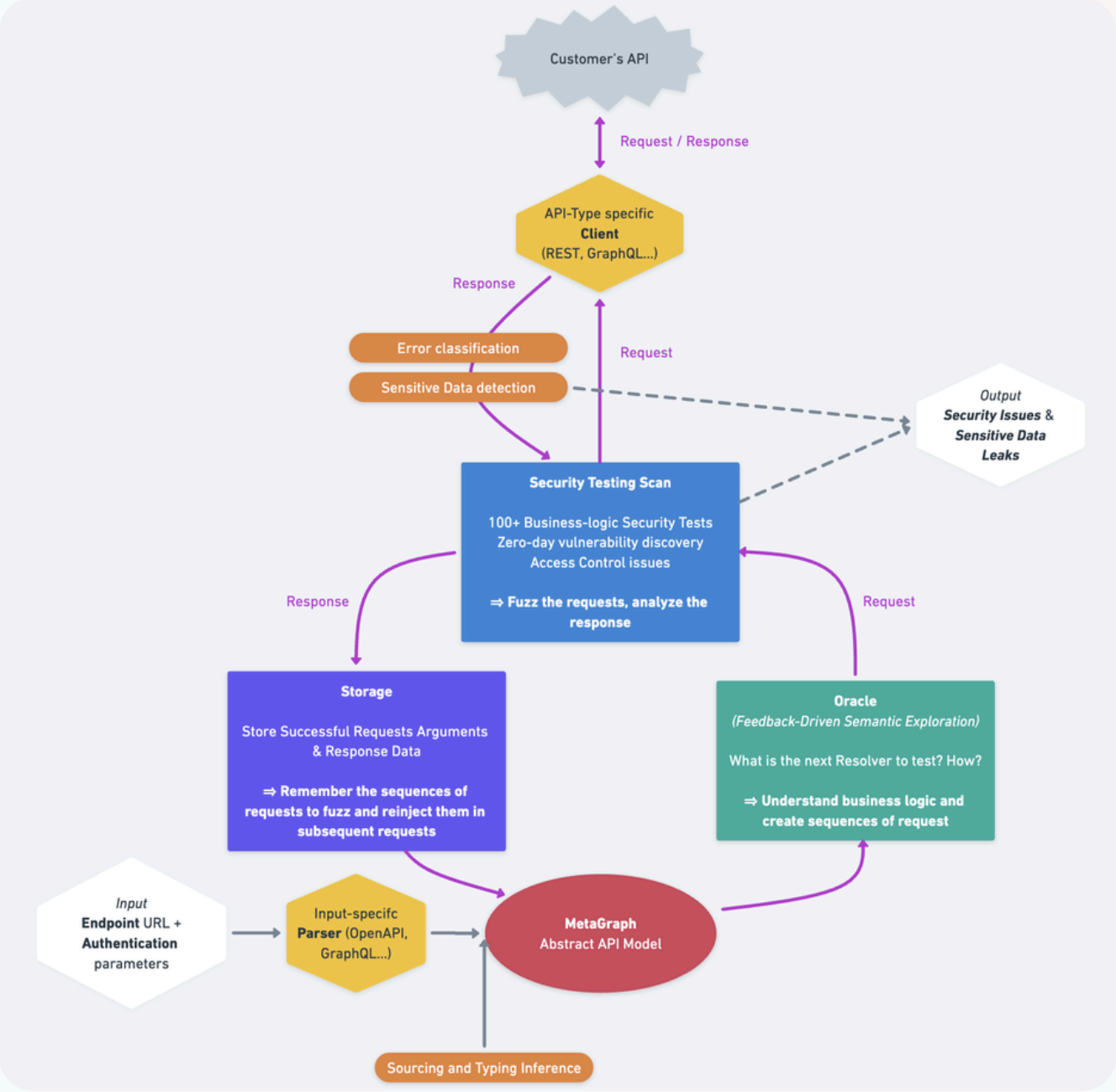
At the core of Escape's DAST is a proprietary algorithm that combines static and dynamic analysis to deliver precise, high-confidence results. The algorithm operates in multiple stages:

1. **Contextual Analysis:** The algorithm first conducts a contextual analysis of each endpoint, identifying the API's structure, parameters, and dependencies. This allows it to adapt its scanning approach to each specific API, ensuring relevance and reducing false positives.
2. **Fuzzing and Payload Injection:** In the next stage, the algorithm uses fuzzing techniques to test each endpoint by sending random, unexpected, or malformed inputs. This helps identify weaknesses like input validation flaws, injection vulnerabilities, and misconfigurations. The algorithm adapts these payloads based on real-time feedback, simulating complex attacks more effectively than conventional static methods.
3. **Behavioral Monitoring:** As it interacts with each endpoint, the algorithm monitors response patterns and behaviors to detect anomalies. By observing responses over multiple interactions, it can identify issues such as data exposure, error leakage, and misconfigured permissions, providing insights into the API's security posture.
4. **Risk Prioritization:** Escape's DAST concludes with an analysis that ranks detected vulnerabilities based on risk and classifies them based on the metric named Escape severity.

You can find an in-depth technical explanation of the algorithm here.

Methodology

Final step - API Security Scanning



The complete event-based architecture of the Escape Security Scanner, with the reinforcement loop in the center

Findings - API Exposure

30,784
API services exposed

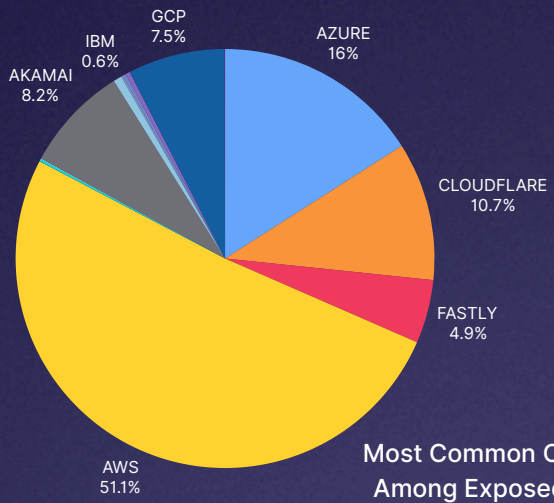
28.5K
among Fortune 1000

3,001
APIs exposed, including
more than 166 exposed staging APIs
by one Fortune 1000 organization

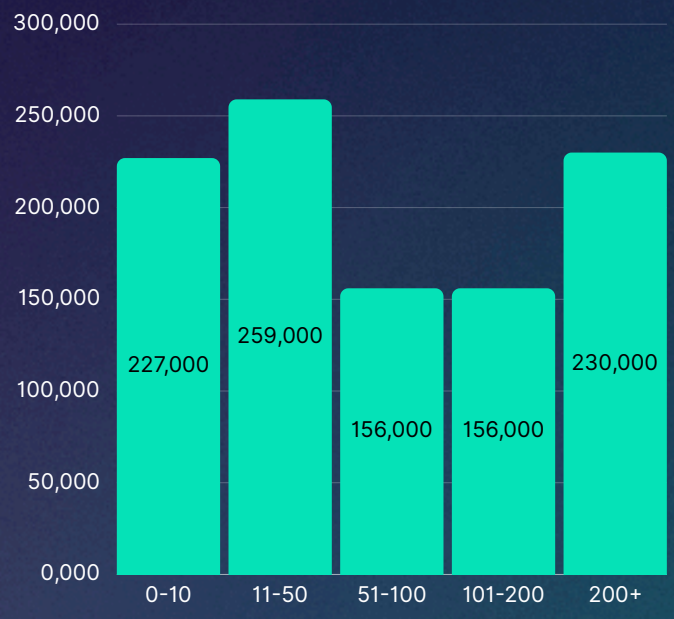
3,945
development APIs exposed,
including 6 organizations **with more than 100 development APIs exposed per domain**
- 5 from Fortune 1000 and 1 from CAC40

91%
REST APIs

1,189
maximum exposed APIs per one domain



Most Common Cloud Providers
Among Exposed API Services



Number of Domains Exposing a Given
Range of API Services

escape

Findings - API Security

107,368
Total vulnerabilities found

2,038
Highly critical

98.8K
vulnerabilities found among Fortune 1000 organizations (large tech orgs. like Amazon, Google or Meta excluded)

1.8K
highly critical

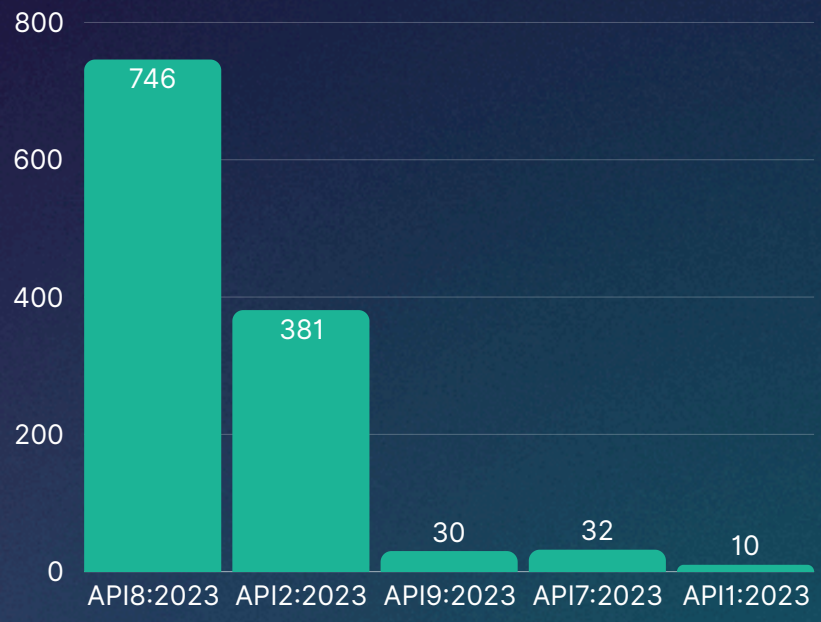
8.1K
API vulnerabilities found among CAC40 organizations, including **240 highly critical**

Distribution of highly critical vulnerabilities

713
high risk CVEs found

11
API services vulnerable to SQL injection

10
API services vulnerable to BOLA



Alignment of high-risk Fortune 1000 vulnerabilities with OWASP Top 10 2023 classification

escape

Findings

42

average vulnerabilities per domain

468

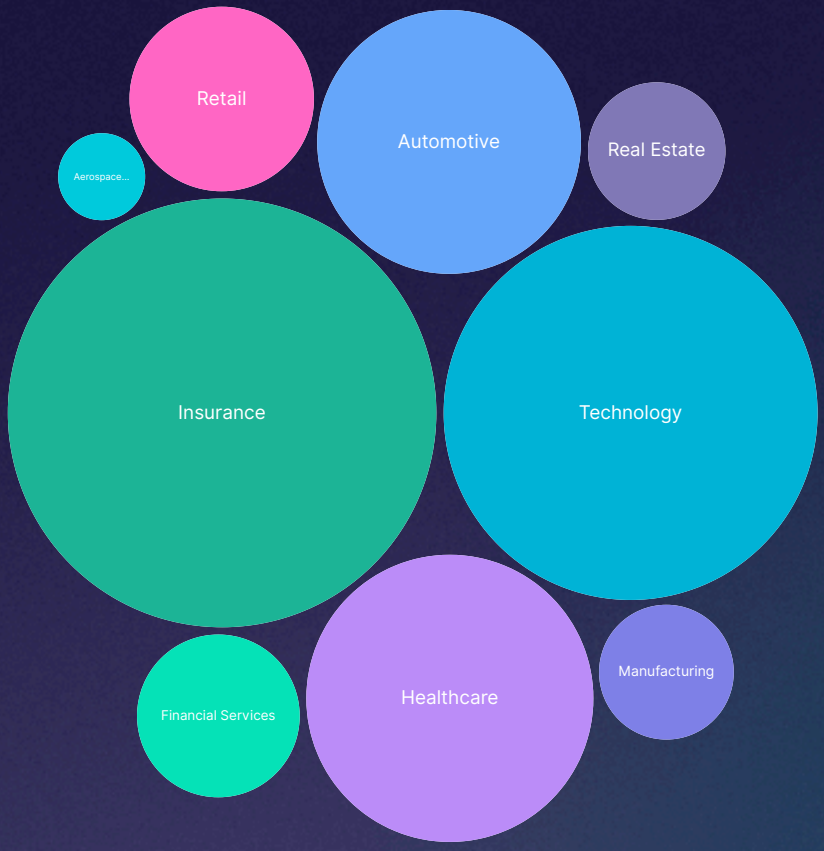
domains are above the average

1,869

the biggest number of **medium-risk** vulnerabilities per exposed domain
(an insurance company based in the US)

205

the biggest number of **high-risk** vulnerabilities per exposed domain
(same insurance company...)



Industries with the highest number of high-risk vulnerabilities

Findings - API Secrets

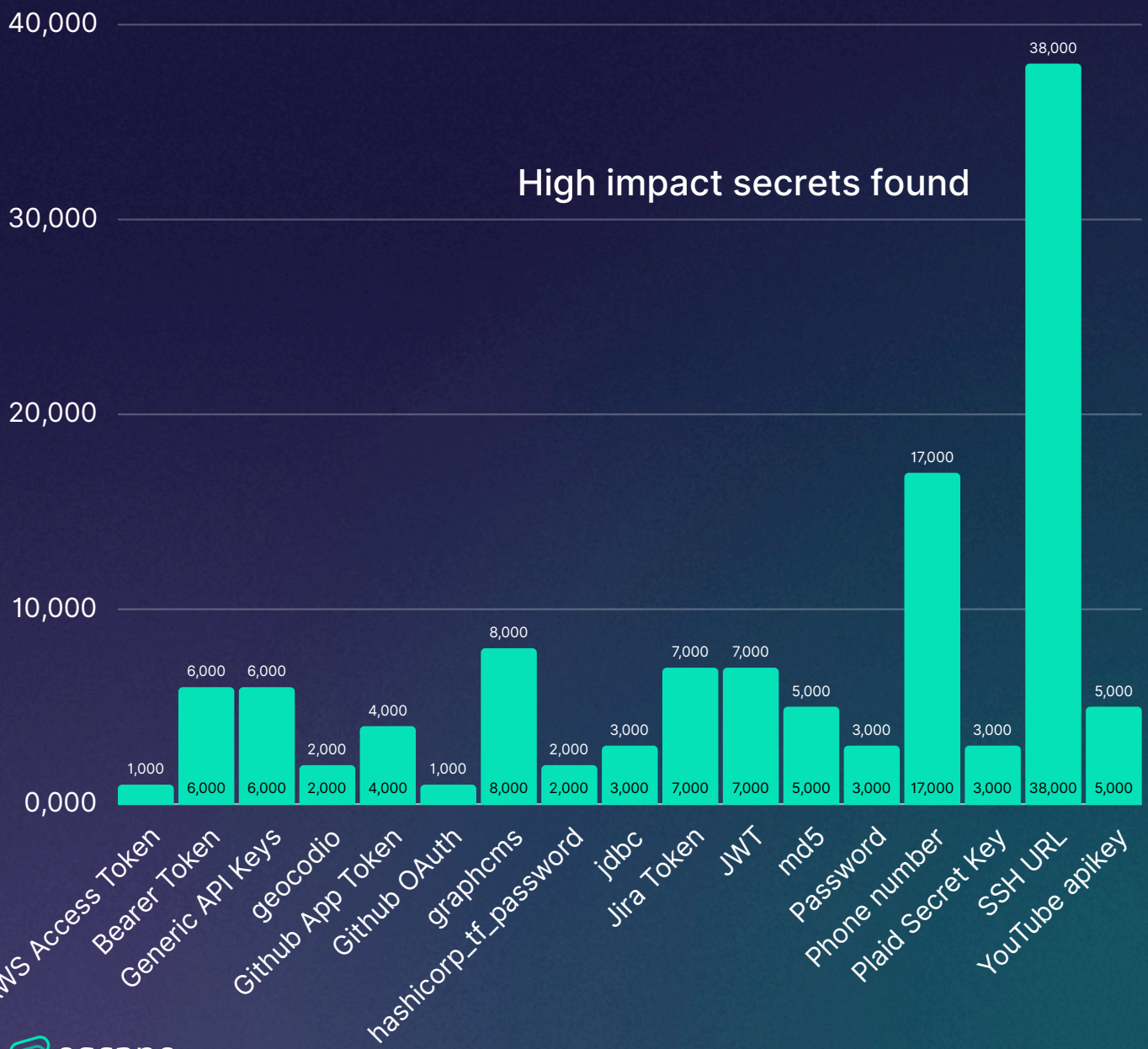
This time, we did not scan for secrets exposed in front-end applications, as we have already covered that topic in our previous report. Instead, we focused on secrets accessible through vulnerable APIs identified by our DAST scans.

1,816

Total secrets found

29

Highly critical



escape

Critical Findings: Exposed and Vulnerable Development APIs

3,945
exposed development APIs

198
highly critical vulnerabilities

In our research, we uncovered a significant volume of exposed development APIs across multiple domains. A total of 3,945 development APIs were identified as publicly accessible, posing considerable security risks due to their lack of adequate protection —3,650 of these exposed APIs were found among Fortune 1000 companies. Exposing development APIs, which often bypass rigorous security controls in favor of testing and iteration, can unintentionally reveal sensitive internal systems, configurations, and potential vulnerabilities, leaving organizations open to exploit.

Among the exposed APIs, we identified at least 198 critical vulnerabilities, including various CVEs and OWASP Top 10 risks, such as API2:2023: Broken Authentication and API8:2023: Security Misconfiguration. These vulnerabilities include risks that could allow unauthorized access to sensitive data, privilege escalation, and potential unauthorized control over critical applications. The nature and distribution of these vulnerabilities indicate a systemic gap in security practices across development environments, as many of these APIs lack proper access control, authentication, and monitoring.

More than that, our analysis highlighted that six organizations were particularly impacted, each with **over 100 exposed development APIs within their domains**. Of these:

- Five organizations are listed within the Fortune 1000 companies, representing some of the largest and most influential corporations globally.
- One organization is part of the CAC40 index, underscoring that even prominent European enterprises are not immune to these security oversights.

The exposure of development APIs at this scale poses a high risk to both data integrity and operational security. Development environments often house experimental or untested code, which may inadvertently expose sensitive configuration details, endpoints, and paths into production systems. Given the volume and scale of the exposure, there is an elevated risk of data breaches, intellectual property theft, and reputational damage, especially for those organizations within the Fortune 1000 and CAC40.

Critical Findings: Fortune 1000 at high risk

1.8K

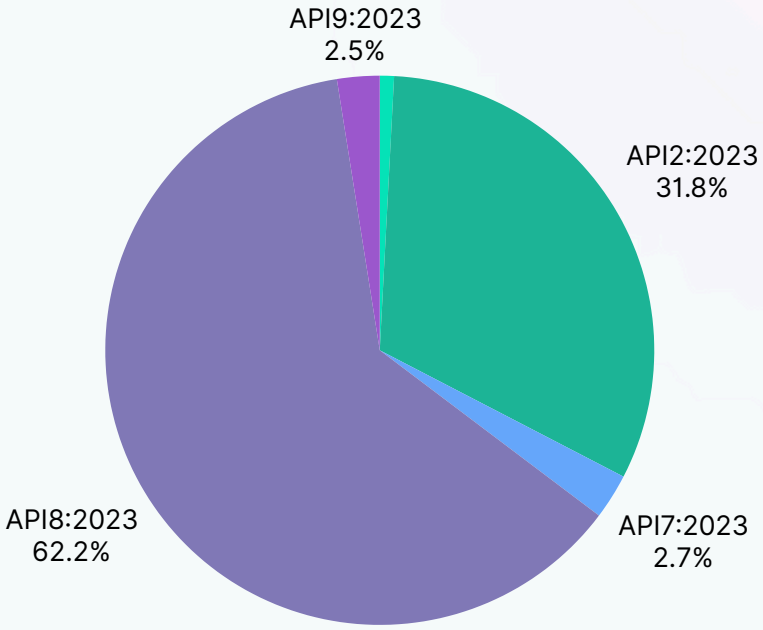
highly critical vulnerabilities found (very large tech orgs. like Amazon, Google or Meta etc. were excluded)

316

companies impacted, some in highly regulated industries like financial services

We uncovered a substantial number of critical vulnerabilities within the exposed APIs of 316 Fortune 1000 companies. In total, we identified 1,830 highly critical vulnerabilities affecting a broad range of companies, including those in highly regulated sectors like financial services, healthcare, and telecommunications. This level of exposure poses significant risks, potentially compromising sensitive data and allowing unauthorized access to critical systems.

We dug into the specifics and found some troubling patterns. According to the OWASP API Security Top 10 for 2023, vulnerabilities like Broken Authentication (API2:2023) and Security Misconfiguration (API8:2023) were rampant, with 381 and 746 instances respectively.



Alignment of high-risk Fortune 1000 vulnerabilities with OWASP Top 10 2023 classification

Many of these APIs lacked even the most basic access controls, allowing for unauthorized data access or manipulation. A staggering 19 NoSQL injection vulnerabilities and 11 SQL injection flaws were scattered across APIs, with 10 instances of Broken Object Level Authorization (BOLA)—all creating a perfect storm for potential exploitation.

It is clear that API security practices hadn't kept up with the growing reliance on APIs across industries, revealing a systemic gap in both development and deployment practices. To tackle this, companies need to close the gap with immediate action: strengthen access controls, enforce secure configurations, and **regularly test APIs** to catch issues before they become threats.

American multinational tech company: Exposed Spring Boot Actuator

Spring Boot Actuator is a sub-project of Spring Boot that provides production-ready features to help you monitor and manage your application.

By design, Actuator exposes diagnostic and monitoring data, such as environment variables, configuration properties, and detailed mappings of application routes. However, if not properly secured, these endpoints can inadvertently expose sensitive information about an application's internal state, providing attackers with critical insights that can be leveraged to exploit vulnerabilities.

During a security assessment of one of the exposed API endpoints owned by a major American technology company, we gained unrestricted access to several Spring Boot Actuator endpoints, specifically:

- https://[redactedcompanydomain.com]/admin/actuator/env
- https://[redactedcompanydomain.com]/admin/actuator/mappings
- https://[redactedcompanydomain.com]/admin/actuator/httptrace.

Each of these endpoints revealed sensitive details about the application's environment, structure, and operations, collectively presenting a substantial security risk.

The first significant finding was with the /env endpoint, which revealed critical environment variables in plain text. Upon loading the endpoint's response, it became clear that sensitive information—such as database credentials, API keys, and service tokens—was accessible without any obfuscation. Exposure of these variables presents a substantial risk, as they can provide unauthorized actors with the means to gain elevated access, move laterally through systems, and potentially launch further targeted attacks on backend services.

Further investigation led to the /mappings endpoint, which disclosed a comprehensive map of the API's routing structure. This included not only public endpoints but also private routes intended for restricted internal use. By detailing the API's route mappings and configuration, the endpoint unintentionally provided a roadmap of the system's structure, revealing potential entry points and avenues for exploitation. In the hands of an attacker, this visibility into the application's internal architecture could simplify the task of identifying weak spots and exploiting vulnerabilities in a targeted manner.

The third exposed endpoint, /httptrace, offered a detailed log of recent HTTP requests, complete with request headers, response statuses, and metadata. This level of trace data, including potential session cookies and internal IP addresses, can significantly aid an attacker in reconstructing user behavior, mimicking legitimate requests, or even hijacking active sessions. Such logs can facilitate a range of attacks, from session replay to broader reconnaissance efforts, making the exposure of this endpoint particularly sensitive.
as database credentials and API keys, to prevent accidental exposure.

American multinational tech company: Exposed Spring Boot Actuator

These Actuator endpoints—/env, /mappings, and /httptrace—collectively provided a depth of insight into the API's environment and operations that, in the absence of access controls, posed a considerable security risk. The availability of these diagnostics endpoints without restriction would allow any external party to gain a detailed understanding of the application's configuration, user activity, and internal routes, effectively dismantling critical layers of operational security.

Recommended Remediation Steps

1. Restrict Access to Actuator Endpoints:
   - Implement access control to ensure that sensitive Actuator endpoints, such as /env, /mappings, and /httptrace, are accessible only to authorized users on the internal network or authenticated users with administrative privileges.
2. Disable Sensitive Actuator Endpoints in Production:
   - Configure the Spring Boot Actuator settings to disable endpoints like /env, /mappings, and /httptrace in the production environment, or at least restrict them to authorized, internal users only.
3. Obfuscate Sensitive Data in /env:
   - In cases where /env must be used, consider redacting or obfuscating sensitive environment variables, such as database credentials and API keys, to prevent accidental exposure.
4. Monitor and Log Access Attempts to Actuator Endpoints:
   - Implement logging and monitoring of all access attempts to these endpoints, enabling detection of any unauthorized access and facilitating incident response.

Overall, this assessment underscores how easily operational details can become visible without proper restrictions, turning what might otherwise be routine endpoints into critical vulnerabilities. Addressing these exposures is essential to uphold the security and integrity of the company's infrastructure.

Standout Scenarios: High-Impact CVEs in the Wild

713

Total High Impact CVEs Found

98

Distinct High Impact CVEs Types

102

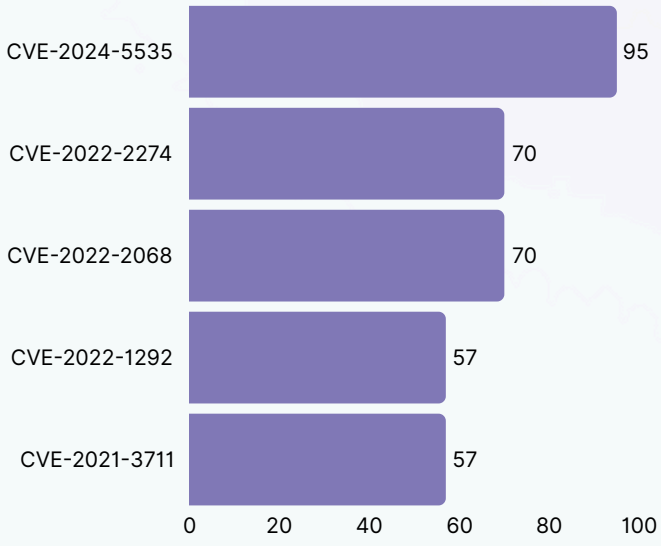
High Impact CVEs in 2024, with CVE-2024-5535 as a clear leader

This section highlights critical cases where APIs, compromised by known CVEs (Common Vulnerabilities and Exposures), are openly accessible on the internet. These vulnerabilities, if unaddressed, can serve as entry points for malicious actors to exploit vulnerable API services.

Key findings:

- **Prevalence of OpenSSL Vulnerabilities:** A lot of found CVEs are associated with OpenSSL, a widely used library for secure communications. APIs leveraging OpenSSL must ensure they are using updated versions to mitigate these vulnerabilities.
- **Risk of Remote Code Execution:** Several vulnerabilities, notably CVE-2022-2274 and CVE-2021-3711, can lead to remote code execution if exploited, posing severe risks to API integrity and security.
- **Importance of Input Validation:** Vulnerabilities like CVE-2022-2068 and CVE-2022-1292 highlight the critical need for proper input validation to prevent command injection attacks.
- **Legacy Vulnerabilities:** A significant number of older CVEs, frequently appearing from 2020 through 2023, continue to impact APIs. This trend suggests that many organizations struggle to patch legacy vulnerabilities. These unpatched older CVEs are often well-documented, making them attractive targets for attackers who can easily exploit known weaknesses.
- **Organizational Impact of Unaddressed CVEs:** The presence of these CVEs in production may affect compliance with security standards and lead to potential regulatory issues. Furthermore, should these CVEs be exploited, the resultant damage could impact not only security but also customer trust and business continuity.

Top 5 CVE occurencies



CVE-2021-3711 (57 instances)
As an older CVE, if it pertains to cryptographic weaknesses or data exposure, it would directly impact APIs by potentially making data-in-transit vulnerable

Standout Scenarios: Mapping CVE to CWE

Diiscovered CVEs map to several key CWEs associated with common API weaknesses. Here are the main categories of weaknesses found in the list and their implications for API security:

Improper Input Validation (CWE-20):

- Relevance: Improper input validation is one of the most common vulnerabilities in API systems, where unvalidated or poorly sanitized inputs can lead to injection attacks, unauthorized access, or unintended behavior.
- Implications: APIs with CWE-20 vulnerabilities are susceptible to attacks that bypass intended restrictions, exploit business logic flaws, or cause unexpected outcomes.

Improper Neutralization of Special Elements used in an OS Command (Command Injection, CWE-78):

- Relevance: Command injection can lead to unauthorized command execution on the server, enabling attackers to gain deeper access to systems.
- Implications: CWE-78 vulnerabilities in APIs are especially dangerous as they can provide attackers with access to server operations, potentially allowing unauthorized data access, privilege escalation, or remote code execution.

Integer Overflow or Wraparound (CWE-190):

- Relevance: Integer overflows occur when calculations exceed the allocated storage for integers, leading to unexpected or unsafe values. In APIs, such vulnerabilities can be used to bypass restrictions or overflow buffers.
- Implications: Vulnerabilities in this category make APIs susceptible to buffer overflows or logic manipulation, often leading to crashes or arbitrary code execution.

Cryptographic Issues (CWE-310):

- Relevance: API communications rely heavily on encryption for data protection, especially when handling sensitive information. Poorly implemented cryptographic measures expose APIs to data interception or decryption attacks.
- Implications: Weak cryptographic implementations in APIs can lead to data leakage, man-in-the-middle (MitM) attacks, or unauthorized data access.

NULL Pointer Dereference (CWE-476):

- Relevance: A null pointer dereference occurs when a program attempts to access or modify data at a null (nonexistent) memory location. In APIs, these vulnerabilities often lead to application crashes or denial of service.
- Implications: API systems with CWE-476 vulnerabilities may be exploited to cause application disruptions, affecting service availability and reliability.

Addressing these CWEs through consistent validation, secure cryptographic practices, and boundary checks is essential to reduce attack surfaces in API environments.

Essential remediation steps

This extensive exposure of vulnerable APIs underscores a critical security issue.

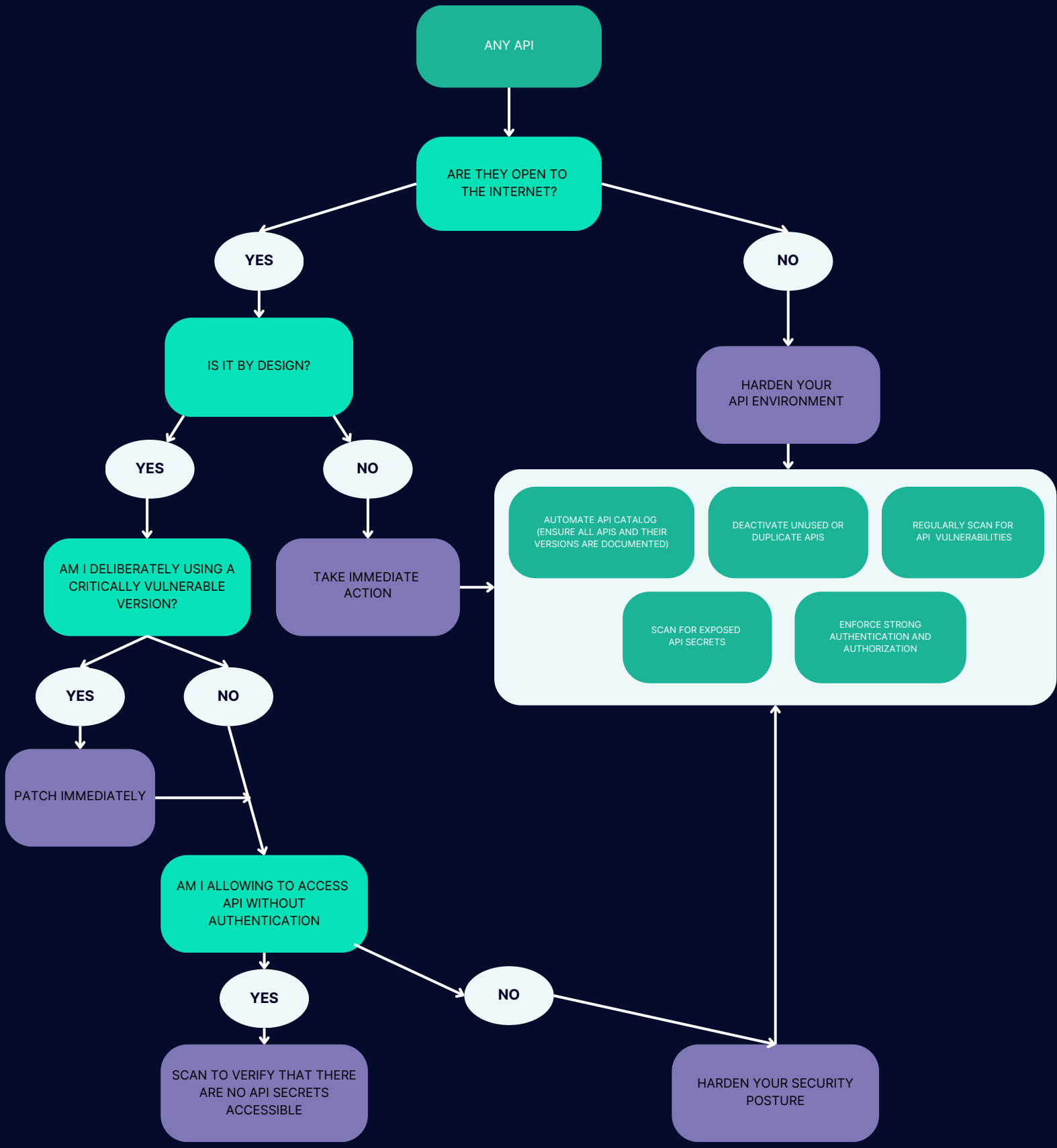
If you have publicly accessible APIs or are uncertain about whether your developers adhere to best practices to prevent unsecured staging or development APIs from being exposed, we recommend taking the following actions **immediately:**

- **Conduct a Comprehensive API Audit:** Review all public-facing APIs to identify any endpoints that are exposed unintentionally. This includes testing for staging, development, and unused APIs that may have been left accessible. Categorize APIs based on sensitivity, usage, and the type of data they handle. This helps prioritize which APIs require immediate attention and remediation.
- **Deactivate Unused or Duplicate APIs:** Regularly review and disable APIs that are no longer in use or required. This practice helps reduce the attack surface of your application and minimizes potential security risks associated with outdated or redundant APIs.
- **Recheck for Exposed API Secrets:** Conduct a thorough check to ensure that API keys, tokens, and other secrets are not exposed in public repositories, documentation, or within API endpoints. Implement automated secret scanning tools to prevent such exposures continuously.
- **Implement an API Governance Strategy:** Establish a robust API governance strategy to manage the growing number of APIs across your organization. This strategy should include standardized processes for API development, documentation, versioning, and security, ensuring consistency and reducing the risk of unsecured APIs.
- **Enforce Authentication and Authorization:** Ensure all APIs are protected with strong authentication and authorization mechanisms. Use industry-standard methods like OAuth 2.0, JWT tokens, and role-based access controls to restrict access and prevent unauthorized use.
- **Implement Automated API Discovery:** Deploy automated API discovery tools to continuously scan and catalog all APIs, including shadow, unused, and potentially exposed APIs. This proactive approach ensures full visibility of your API environment.
- **Implement an Automated Security Solution that Can Start Scanning in Minutes:** Utilize a security solution that enables immediate scanning of your API ecosystem for vulnerabilities and misconfigurations. A solution that provides quick setup and fast scanning capabilities can help identify and mitigate risks efficiently without disrupting workflows.
- **Educate Your Internal Teams:** Ensure that all team members understand the importance of token security and adhere to best practices. Consider enhancing the security experience through gamification or implementing a Security Champion Program, following The Security Champion Program Success Guide.

These measures are crucial for protecting your APIs and maintaining secure and compliant applications. For more detailed information, you can use our API Security Checklist or refer to the resources provided on the Escape blog.

Immediate steps to take

We also prepared a handy diagram to help you understand your environment and mitigate the risks with confidence:



escape

Conclusion

Securing all your APIs is hard. It’s even harder when you don’t know what you have to secure, want to ship (insecure) APIs too fast in the wild and have to monitor them at large scale. Your organization is now not only prone to data breach risks but also to severe financial implications.

Our study reveals that over 100,000 vulnerabilities are present across Fortune 1000 and CAC 40 companies, with 1,800 classified as highly critical, directly impacting some of the largest organizations globally.

One of the most striking findings was the extensive exposure of vulnerable development APIs and the high level of overall vulnerability criticality. This highlights the real and present dangers for each organization and the necessity of thoroughly testing applications both before and after deployment.

Moreover, we identified that organizations continue to expose sensitive secrets, including access tokens and API keys, within their API environments, amplifying security risks and potential misuse.

Organizations must respond fast, adopting best practices for risk mitigation and integrating continuous, automated testing of their applications.

Are you looking to automate discovery and security of your APIs at scale, improve your organization's security posture and need to ensure compliance with standards like HIPAA, GDPR, and PCI DSS?

Our team is here for you. We'll help identify your online exposures and provide strategic advice on implementing effective security practices. Feel free to reach out!

Do you need help in assessing whether your APIs are exposed and at risk? We're here for you. With Escape you can:

- Automate the **discovery of all APIs at scale**
- Automate **API documentation generation**
- Ensure comprehensive **security coverage** with 130+ API security tests, including OWASP Top 10, business logic, and access control
- **Automated security** scanning by plugging Escape's modern DAST into your CI/CD systems
- Gain instant access to the affected repository and **developer-friendly remediation** code snippets

Learn more

escape.tech
ping@escape.tech
+1 (707) 615 6448