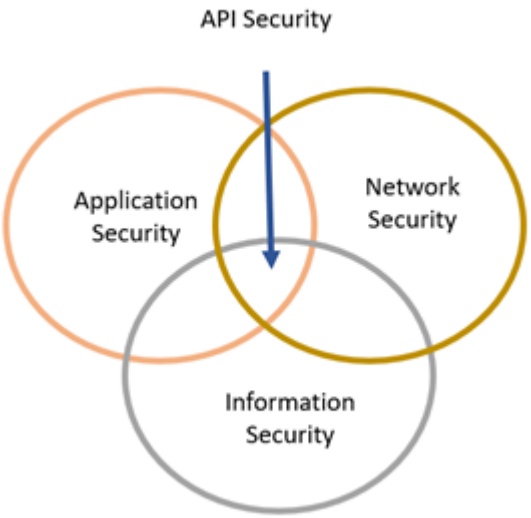
API Security 101: Understanding the Risks and Implementing Best Practices

What is API Security?

API security is the process of effectively securing APIs owned by the organization and external APIs used by implementing API-specific security strategies.API security secures API vulnerabilities and misconfigurations and prevents their exploitation by attackers.

API security lies at the intersection of three broad security areas:



API security also deals with security issues, including content validation, access control, rate limiting, monitoring & analytics, throttling, data security, and identity-based security.

With sensitive data getting transferred via API, a secure API can guarantee the confidentiality of its message by making it available to the applications, users, and servers with proper permissions.

Similarly, securing APIs guarantees content integrity by ensuring the message has not been altered after transmission.

Why is API Security Important?

API has become essential to modern software development, allowing different software applications to communicate and share data. However, this increased connectivity also poses significant security challenges.

APIs can be vulnerable to attacks from malicious actors seeking to exploit them for their own purposes. This has led to a growing focus on API security challenges in recent years, driven by several key factors:

Digital Transformation

More and more businesses are moving their operations online and embracing digital technologies. They increasingly rely on APIs to integrate different systems and services. However, this also means sensitive data is transmitted through APIs, creating potential security risks.

Cloud Computing

Cloud-based applications and services rely heavily on APIs to exchange data and interact with each other. Any security vulnerabilities in these APIs can have far-reaching consequences.

Easy to Bypass Security Measures

API vulnerabilities and security weaknesses are unique, and so are the security risks. Organizations often rely on security solutions built for web apps to detect and secure from API threats. Such solutions cannot detect unique vulnerabilities and gaps in APIs. So, attackers can effortlessly exploit APIs by bypassing security measures.

How are APIs Abused?

By design, the Application Programming Interface is not insecure. However, the immense volume of API deployed has created challenges for the security team. Further, insufficient skills in API development and failure to incorporate the web and cloud API security rules may lead to vulnerable APIs.API vulnerabilities can be observed across various areas like data exposures, denial of service, authorization flaws, security misconfigurations, and endpoints (virtual environment, devices, servers, and more).

Likewise, attackers can use various other techniques to abuse APIs.

OWASP has listed the potential risks associated with APIs in its **OWASP Top 10 API Risks list,** which includes:

API1:2019 Broken Object Level Authorization: An application may allow users to perform actions or access resources they should not be able to. For example, if a user is only authorized to view their own personal information, but the application allows them to view another user's information, this would be a Broken Object Level Authorization vulnerability.

API2:2019 Broken User Authentication: This refers to an API vulnerability in web applications where user authentication mechanisms are poorly implemented or configured, allowing attackers to gain unauthorized access to user accounts or sensitive information.

API3:2019 Excessive Data Exposure: This vulnerability can occur when an API returns more data than necessary or returns sensitive information in an unencrypted form. This can lead to identity theft, financial fraud, and other serious security incidents.

API4:2019 Lack of Resources & Rate Limiting: Without adequate resources, an API can become overwhelmed by incoming requests, resulting in service downtime, slow response times, or even a complete denial of service. In addition, if an API does not enforce rate-limiting policies, it can be vulnerable to attacks that flood the API with high requests, leading to similar outcomes.

API5:2019 Broken Function Level Authorization: Function level authorization is the process of controlling access to specific functions or resources based on the privileges or permissions of the user or client. If an API implementation does not properly enforce function-level authorization, attackers can gain access to sensitive data, perform unauthorized actions, or even take over the entire system.

API6:2019 Mass Assignment: Mass assignment is used in many API implementations to allow clients to submit data to an API to create or modify objects. However, suppose an API implementation does not properly validate the data received from clients. In that case, attackers can alter or create unintended attributes or parameters, leading to security vulnerabilities such as privilege escalation or data tampering.

API7:2019 Security Misconfiguration: This vulnerability occurs when the API is not configured securely. Security misconfiguration can lead to data exposure, unauthorized access to sensitive information, system compromise, and other types of attacks that exploit the misconfiguration.

API8:2019 Injection: An injection attack on APIs allows an attacker to inject malicious code or commands into an API input field. The attacker can then use this vulnerability to manipulate the API's behavior, access sensitive data, or execute unauthorized actions on the target system.

The most common type of injection attack on APIs is **SQL injection**, where an attacker exploits vulnerabilities in the SQL database query language to gain access to sensitive data or modify data within a database. Other injection attacks on APIs include XML, LDAP, and command.

API9:2019 Improper Assets Management: Improper Assets Management can occur for various reasons, including insecure storage, weak encryption, or a lack of proper

access controls. This vulnerability can lead to data breaches, unauthorized access to sensitive information, and other security incidents if left unaddressed.

API10:2019 Insufficient Logging & Monitoring: Not detecting and responding to security incidents on time can be challenging without proper logging and monitoring, significantly damaging the API, its users, and their data. This vulnerability can occur for various reasons, including a lack of logging and monitoring capabilities, inadequate or inconsistent logging practices, and failure to analyze and respond to log data on time.

API Security Breaches Examples

API security breaches can be particularly damaging as they expose sensitive data and functionality of an organization's software systems to unauthorized users.

Here are some notable examples of **API security breaches:**

Equifax API Breach (2017): Equifax, a credit reporting agency, experienced a massive data breach when hackers accessed sensitive data on over 143 million customers. The breach occurred due to a vulnerability in the company's API, allowing attackers to access the data without proper authentication.

Strava API Breach (2018): Strava, a fitness app widely used by military personnel to track their exercise routines inadvertently revealed sensitive information about military bases across the globe. This occurred as a result of a vulnerable API that transmitted users' location data online.

Facebook Data Breach (2019): More than 530 million Facebook users' personal information, such as phone numbers, account names, and Facebook IDs, was compromised in April 2019. The information was contained in two datasets from third-party Facebook applications that were publicly exposed. The attacker leveraged a vulnerability in the third-party developer's API to carry out the attack. This allowed them to acquire access tokens and escalate their privileges to compromise the affected accounts.

LinkedIn API Breach (2021): In June 2021, a significant security breach occurred when a public API without authentication was exposed. It resulted in the compromise of data associated with 92 percent of its users. This allowed a malicious actor to scrape the platform for information on approximately 700 million users, including their email addresses and phone numbers.

T-Mobile API Breach (2022): T-Mobile has reported a recent data breach where a threat actor accessed the personal information of 37 million current postpaid and prepaid customer accounts via one of its APIs.

API Security vs. Application Security: What's the difference?

Click here for a complete replay of the webinar "**Comprehensive Risk-based API Protection."**

As this short clip illustrates, API security is more than web security. API security shares many of the same principles as web security. However, securing APIs presents unique challenges that require specialized security measures.

APIs are often accessed over the web and use HTTP as the underlying protocol. As a result, many of the same security principles that apply to web security also apply to API security.

For example, API security involves protecting against attacks such as **SQL injection**, **cross-site scripting (XSS)**, and cross-site request forgery (CSRF), which are all common in web security. API security also involves using secure communication protocols such as HTTPS to protect data in transit, which is also an important part of web security.

However, some unique API security issues extend beyond the realm of web security.

One of the biggest API security challenges is that APIs are often designed to be accessible by third-party applications or services. This means that APIs are exposed to a wider range of potential attackers than traditional web applications. Attackers

can use APIs to exploit vulnerabilities in the application, steal sensitive data, or launch attacks against other applications or services.

Another challenge of API protection is that APIs are often designed to be highly flexible and customizable, making them more vulnerable to attacks. For example, APIs may allow users to specify the data type they want to receive or the format in which the data is returned. This flexibility can make it easier for attackers to exploit API code or configuration vulnerabilities.

API protection also presents challenges when it comes to authentication and access control. APIs often use tokens or other authentication forms to control API access. However, these tokens can be stolen or compromised, allowing attackers to access the API and its data.

Finally, API security can be challenging because of the sheer number of APIs used in modern software systems. Applications may use dozens or hundreds of APIs to communicate with other applications or services. This makes it difficult to monitor and protect APIs effectively.

Is an API Gateway Enough to Ensure Robust API Security?

An API gateway is essential in securing APIs, but it is not enough to ensure robust API security on its own.An API gateway can provide some security features, such as

- Rate limiting and throttling to ensure APIs aren’t overused and/or abused
- The authorization and routing of API calls to the right backend services and frontend endpoints
- Access management to prevent abuses and ensure only authorized entities access different functionalities
- Authentication through credential and token validation to verify identities
- Analytics, monitoring, logging, and alerts

However, it is still vulnerable to various attacks. An API gateway covers only the endpoints and does not cover every layer of advanced API security.

Check out why you need API security solution if you have API Gateway:

Some instances where API gateway security is found wanting are:

- It doesn't provide visibility and control over the entire API architecture
- It doesn't unearth misconfigured, rogue, or shadow APIs, leaving them open for attackers to find and exploit
- It cannot effectively distinguish between human and bot activity. Given how sophisticated today's bots are, they cannot thwart complex and highly damaging API-specific bot attacks
- It is not equipped to stop advanced, multi-vector DDoS attacks
- To ensure robust API security, it is essential to implement a defense-in-depth strategy, including multiple security control layers

Top 10 API Security Best Practices

As attackers continue to expose and exploit the weaknesses in APIs, the need for API security is urgent and critical. Use this API security checklist to start solidifying your API security posture.

1. API Discovery and Inventorying

Continuous discovery and inventorying of APIs are essential for effective security measures. You can secure APIs only if you know they exist.

- Automate the discovery of all API endpoints, parameters, and data types using intelligent API vulnerability scanners. Automated tools infuse speed, agility, and accuracy in discovery.
- Analyze API traffic metadata using an intelligent engine to discover APIs that weren't on the security practitioners' radar earlier.

- Discover APIs in the lower environment (beyond the production environment), API dependencies, and third-party APIs.
- Tag, label, and segment APIs and microservices.
- Real-time analysis of traffic hitting API endpoints. This allows organizations to identify workloads and tighten perimeters around risky endpoints. It helps de-provision and remove old functionalities and zombie APIs. Also, it minimizes blind spots created by rogue APIs.

2. Implement A Zero Trust Philosophy

When it comes to “What is API Security?” many people would highlight API authentication, but API security is more about API threat prevention.

Zero Trust is a security policy centered on the principle that companies should not trust anyone by default and must verify everything trying to access their systems.

Zero-Trust ideology should be applied to even authorized API endpoints, authenticated clients, and unauthenticated and unauthorized entities.

Critical factors to consider while implementing a zero-trust policy on your API include:

- API Protocol Support
- API Deep Request Inspection
- Cloud-native Deployment Method
- API Discovery – Up-to-date API Inventory
- Data leakage prevention.

3. Identify API Vulnerabilities and Associated Risks

So, another important requirement in the API security checklist is proactive API threat detection and protection. It is dangerous to ignore API vulnerabilities and risks.

- Use behavioral, pattern, and heuristic analysis to detect threats proactively instead of relying on rudimentary signature-based detection.
- Implement multi-layered security to tackle and protect APIs against different kinds of threats, including DDoS attacks, bot attacks, OWASP Top 10 API security risks, etc.
- Leverage advanced technology such as self-learning AI, analytics, and automation for advanced API protection.
- Real-time visibility into security posture is important.
- Encrypt all data to strengthen API protection.
- Implement virtual patching to keep vulnerabilities secure until developers fix them.
- Assess the API endpoints before any code changes to ensure data handling requirements and security are not compromised.

Continuous security testing is another key API security checklist requirement. With thorough API security testing, discover which parts of your API are vulnerable to known threats.

Automated scanning has limits and cannot identify security misconfigurations or business logic flaws. Regular manual security testing by certified experts through pen tests and audits is necessary.

4. Enforce Strong Authentication and Authorization

Though authentication and authorization play different roles, these two API best practices work as powerful tools for API protection when implemented together.

Authentication is necessary for securely verifying the user of the API. Authorization is concerned with what data they have access to.

API authentication allows to restriction or removal of users who abuse the API. API authorization usually starts after the identity is confirmed through authentication and verifies whether users or applications can access the API.

API authentication and authorization serve the following purposes:

- Authenticate calls to the API to legitimate users only
- Track the requesters
- Track API usage
- Enable different levels of permissions for different users
- Block the requester who exceeds the rate limit

API Best practices for authentication and authorization

- Include human and machine identities while implementing access controls and authentication. Do not forget third-party applications and services.
- Use modern authorization protocols for robust security.
- Public APIs should not be exposed to unvalidated requests, even if the users are authorized.
- Implement strong and complex passwords in combination with multifactor authentication.
- Non-admin users should only be granted read-only privileges to data.
- Sessions must have a limited duration.
- Tokens must expire at regular intervals to prevent replay attacks.

Here's how to implement strong **authentication mechanisms for APIs.**

5. Expose Only Limited Data

When we think of web API security best practices, we often think of blocking out malicious activity. It is also vital to limit the accidental exposure of sensitive information.

As APIs are a developer's tool, they often include passwords, keys, and other secret information that reveal too many details about the API endpoints. Ensure APIs only expose as much data as needed to fulfill their operation. Further, enforce data

access controls and the principle of least privilege at the API level, track data, and conceal if the response exposes confidential data.

6. Implement Rate Limits

DDoS (Distributed Denial of Service) is the most common practice of attacking an API by overwhelming it with an unlimited API request. This attack affects the availability and performance of APIs.

Rate limiting, also known as API limiting, enforces a limit on how often an API is called (to ensure that an API remains available to legitimate requests). Beyond **DDoS attack mitigation**, it limits abusive actions like aggressive polling, credential stuffing, and rapidly updating configurations. API rate limiting not only deals with fair usage of shared resources but also can be used to:

- Implement different access levels on API– based services
- Meter the API usage
- Guarantee API performance
- Ensure system availability

7. API Design and Development

Due to developers' pressure to ensure speed to market, they often do not have time for security testing. Further, they tend to deploy APIs with known vulnerabilities and dark spots for future functionalities, leave old versions deployed for backward compatibility, etc.

So, API security should not be limited to production. An advanced API security tactic is implementing security controls and techniques from the design and development stages.

- Build secure-by-design APIs
- Use secure development frameworks, code, templates, libraries, and so on

- Restrict source code accessibility to only those who need it
- Review design and code for flaws, especially those related to business logic
- Include security configuration checks in your API security checklist to root out misconfigurations
- Look for hidden form fields and document API response

8. API Logging and Monitoring

Logging and monitoring APIs help construct a baseline for what is considered ‘normal’ so that outlier incidents can be quickly detected.

- Identify and clearly define all elements, infrastructure, and apps that need to be logged
- Track and log non-security parameters such as API performance, speed, and uptime
- Review anomalies at regular intervals and tune your APIs accordingly

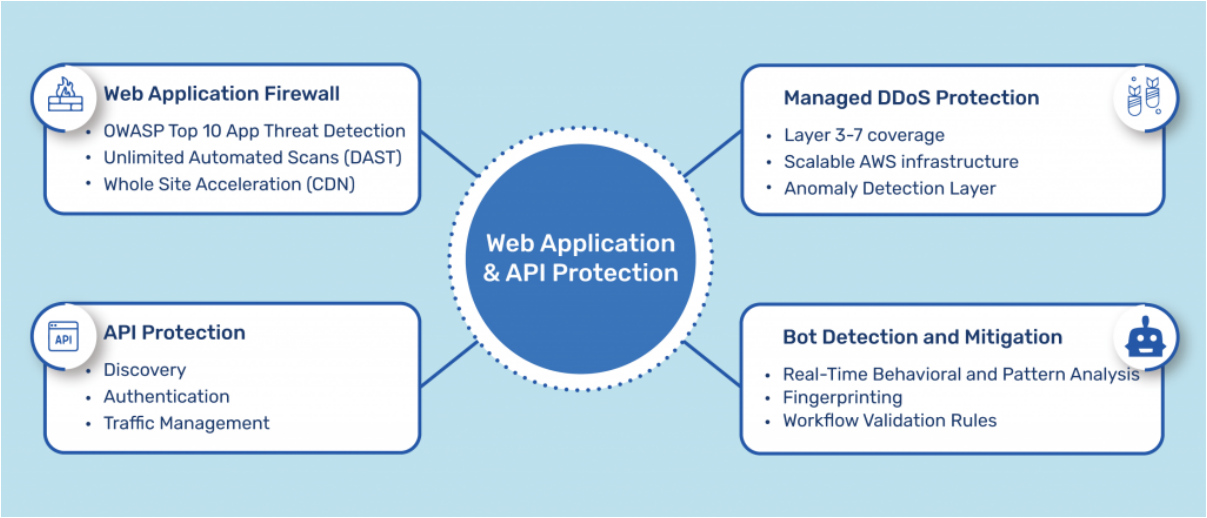
9. Incidence Response

Detecting and stopping breaches is just a part of the security response. Data breaches, at times, are unavoidable. In such cases, having a robust incident response plan is essential as it enables organizations to bounce back quickly and minimize the impact of breaches. It should clearly define policies and measures related to immediate response, investigation, forensics, escalation, compliance, etc.

10. Implement Web Application and API Protection (WAAP)

We recommend a Web Application and API Protection (WAAP) solution for business use cases where API calls are made from the web and mobile apps. These apps commonly have access to ample amounts of sensitive information, and APIs in these channels is tough to defend.

Common security tools like traditional firewalls and API gateways are insufficient to prevent API attacks. WAAP solution is centered around four consolidated capabilities: DDoS protection, Web Application Firewall, Bot Management, and API protection.



It employs a fully managed and risk-based application security approach by monitoring traffic to detect abnormal activities and malicious traffic across all four vectors. The data collected across all the applications assess risks and update mitigation strategies to enhance cyber defense in real-time.

WAAP also aids in reducing operational complexity by reducing the number of parameters that need to be managed, streamlining security rulesets, and automatically suggesting rules with their AI capabilities.

While WAF protects against OWASP top 10 attacks and API gateway defends against standard attacks, AI-enabled behavioral analysis of WAAP ensures the defense against automated and more sophisticated attacks.

As APIs become a strategic necessity to offer your business the speed and agility needed to succeed, your goal should be defending them from evolving attacks.

With **Indusface WAAP**, keep API threats and malicious bots at bay. Our AppTrana platform ensures that your API is highly available and guarantees the confidentiality and integrity of the data it processes.

AUTHOR - Himangshu Sarkar

Github - https://github.com/Himangshu30