API9:2023 Improper Inventory Management

Threat agents/Attack vectors	Security Weakness	Impacts
API Specific : Exploitability **Easy**	Prevalence **Widespread** : Detectability **Average**	Technical **Moderate** : Business Specific
Threat agents usually get unauthorized access through old API versions or endpoints left running unpatched and using weaker security requirements. In some cases exploits are available. Alternatively, they may get access to sensitive data through a 3rd party with whom there's no reason to share data with.	Outdated documentation makes it more difficult to find and/or fix vulnerabilities. Lack of assets inventory and retirement strategies leads to running unpatched systems, resulting in leakage of sensitive data. It's common to find unnecessarily exposed API hosts because of modern concepts like microservices, which make applications easy to deploy and independent (e.g. cloud computing, K8S). Simple Google Dorking, DNS enumeration, or using specialized search engines for various types of servers (webcams, routers, servers, etc.) connected to the internet will be enough to discover targets.	Attackers can gain access to sensitive data, or even take over the server. Sometimes different API versions/deployments are connected to the same database with real data. Threat agents may exploit deprecated endpoints available in old API versions to get access to administrative functions or exploit known vulnerabilities.

Is the API Vulnerable?

The sprawled and connected nature of APIs and modern applications brings new challenges. It is important for organizations not only to have a good understanding and visibility of their own APIs and API endpoints, but also how the APIs are storing or sharing data with external third parties.

Running multiple versions of an API requires additional management resources from the API provider and expands the attack surface.

An API has a "documentation blindspot" if:

- The purpose of an API host is unclear, and there are no explicit answers to the following questions
    - Which environment is the API running in (e.g. production, staging, test, development)?
    - Who should have network access to the API (e.g. public, internal, partners)?
    - Which API version is running?
- There is no documentation or the existing documentation is not updated.
- There is no retirement plan for each API version.
- The host's inventory is missing or outdated.

The visibility and inventory of sensitive data flows play an important role as part of an incident response plan, in case a breach happens on the third party side.

An API has a "data flow blindspot" if:

- There is a "sensitive data flow" where the API shares sensitive data with a third party and
    - There is not a business justification or approval of the flow
    - There is no inventory or visibility of the flow
    - There is not deep visibility of which type of sensitive data is shared

Example Attack Scenarios

Scenario #1

A social network implemented a rate-limiting mechanism that blocks attackers from using brute force to guess reset password tokens. This mechanism wasn't implemented as part of the API code itself but in a separate component between the client and the official API (`api.socialnetwork.owasp.org`). A researcher found a beta API host (`beta.api.socialnetwork.owasp.org`) that runs the same API, including the reset password mechanism, but the rate-limiting mechanism was not in place. The researcher was able to reset the password of any user by using simple brute force to guess the 6 digit token.

Scenario #2

A social network allows developers of independent apps to integrate with it. As part of this process a consent is requested from the end user, so the social network can share the user's personal information with the independent app.

The data flow between the social network and the independent apps is not restrictive or monitored enough, allowing independent apps to access not only the user information but also the private information of all of their friends.

A consulting firm builds a malicious app and manages to get the consent of 270,000 users. Because of the flaw, the consulting firm manages to get access to the private information of 50,000,000 users. Later, the consulting firm sells the information for malicious purposes.

How To Prevent

- Inventory all and document important aspects of each one of them, focusing on the API environment (e.g. production, staging, test, development), who should have network access to the host (e.g. public, internal, partners) and the API version.

  API hosts

- Inventory and document important aspects such as their role in the system, what data is exchanged (data flow), and their sensitivity.

  integrated services

- Document all aspects of your API such as authentication, errors, redirects, rate limiting, cross-origin resource sharing (CORS) policy, and endpoints, including their parameters, requests, and responses.
- Generate documentation automatically by adopting open standards. Include the documentation build in your CI/CD pipeline.
- Make API documentation available only to those authorized to use the API.
- Use external protection measures such as API security specific solutions for all exposed versions of your APIs, not just for the current production version.
- Avoid using production data with non-production API deployments. If this is unavoidable, these endpoints should get the same security treatment as the

production ones.

- When newer versions of APIs include security improvements, perform a risk analysis to inform the mitigation actions required for the older versions. For example, whether it is possible to backport the improvements without breaking API compatibility or if you need to take the older version out quickly and force all clients to move to the latest version.