# Best Practices in API Lifecycle Management

**Mark Boyd**
**mark@platformable.com**

# Key lifecycle issues

- Security
- Traceability

# Security

https://raw.githubusercontent.com/OWASP/API-Security/master/2019/en/dist/owasp-api-security-top-10.pdf

**T10 OWASP API Security Top 10 - 2019**

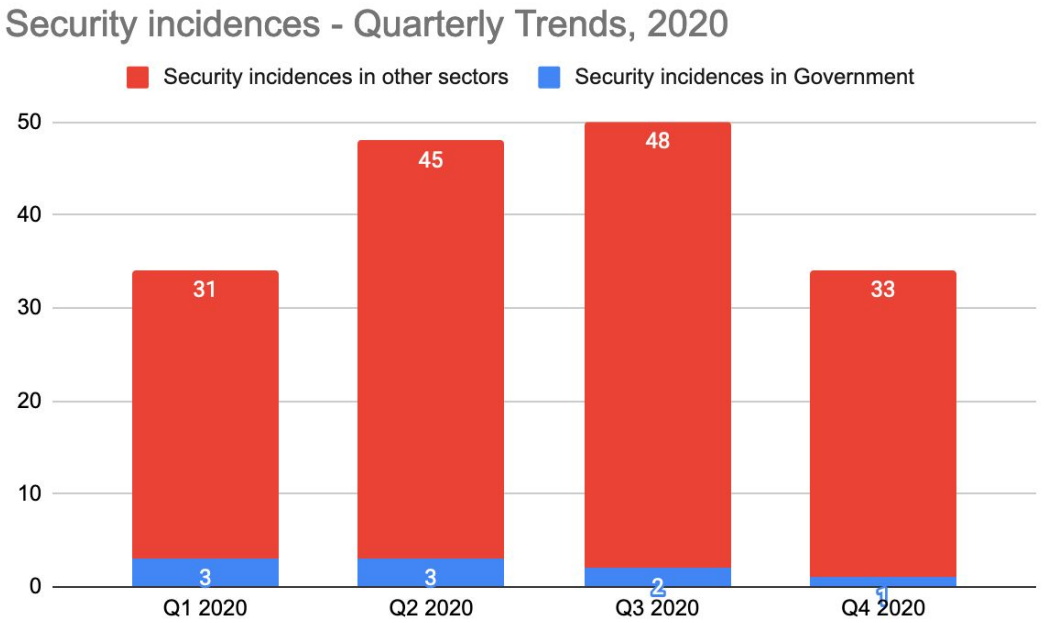
| | |
|---|---|
| API1:2019 - Broken Object Level Authorization | APIs tend to expose endpoints that handle object identifiers, creating a wide attack surface Level Access Control issue. Object level authorization checks should be considered in every function that accesses a data source using an input from the user. |
| API2:2019 - Broken User Authentication | Authentication mechanisms are often implemented incorrectly, allowing attackers to compromise authentication tokens or to exploit implementation flaws to assume other user's identities temporarily or permanently. Compromising system's ability to identify the client/user, compromises API security overall. |
| API3:2019 - Excessive Data Exposure | Looking forward to generic implementations, developers tend to expose all object properties without considering their individual sensitivity, relying on clients to perform the data filtering before displaying it to the user. |
| API4:2019 - Lack of Resources & Rate Limiting | Quite often, APIs do not impose any restrictions on the size or number of resources that can be requested by the client/user. Not only can this impact the API server performance, leading to Denial of Service (DoS), but also leaves the door open to authentication flaws such as brute force. |
| API5:2019 - Broken Function Level Authorization | Complex access control policies with different hierarchies, groups, and roles, and an unclear separation between administrative and regular functions, tend to lead to authorization flaws. By exploiting these issues, attackers gain access to other users' resources and/or administrative functions. |
| API6:2019 - Mass Assignment | Binding client provided data (e.g., JSON) to data models, without proper properties filtering based on a whitelist, usually lead to Mass Assignment. Either guessing objects properties, exploring other API endpoints, reading the documentation, or providing additional object properties in request payloads, allows attackers to modify object properties they are not supposed to. |
| API7:2019 - Security Misconfiguration | Security misconfiguration is commonly a result of unsecure default configurations, incomplete or ad-hoc configurations, open cloud storage, misconfigured HTTP headers, unnecessary HTTP methods, permissive Cross-Origin resource sharing (CORS), and verbose error messages containing sensitive information. |
| API8:2019 - Injection | Injection flaws, such as SQL, NoSQL, Command Injection, etc., occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization. |
| API9:2019 - Improper Assets Management | APIs tend to expose more endpoints than traditional web applications, making proper and updated documentation highly important. Proper hosts and deployed API versions inventory also play an important role to mitigate issues such as deprecated API versions and exposed debug endpoints. |
| API10:2019 - Insufficient Logging & Monitoring | Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems to tamper with, extract, or destroy data. Most breach studies demonstrate the time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring. |

# Security

**Example from government:**

An unemployed computer programmer found an API vulnerability in the Arkansas Pandemic Unemployed Assistance (PUA) website when applying for assistance.

The portal had an unprotected API connecting to the backend. This exposed highly confidential personal information — including social security numbers and bank account numbers — of about 30,000 applicants who had used the portal.



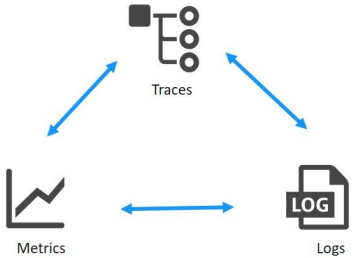
# Security

**Example from industry:**

https://42crunch.com/white-paper-api-security-enterprise/

# Traceability


Three data silo

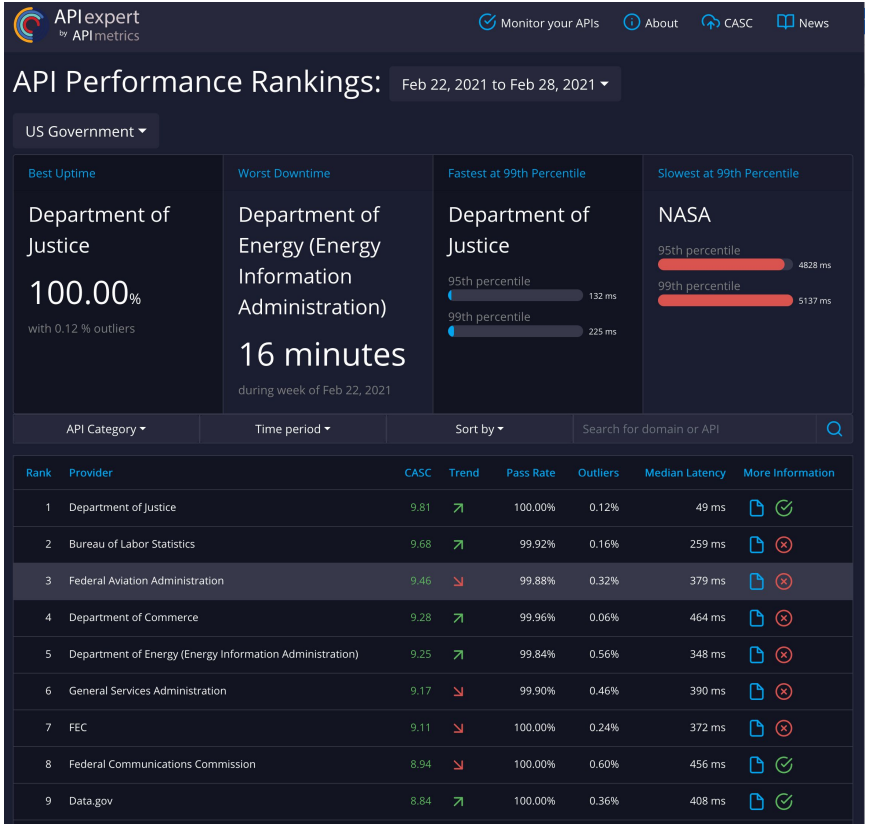



Three pillars of observability, in context

# Traceability

**Example from government:**

New Zealand Government uses CASC scores from API Metrics



# Traceability

**Example from industry:**

To meet GDPR requirements, some private companies are only collecting partial logs to monitor performance without risk of collecting personal data

# Discussion

- What are the main security threats they are facing?
- What solutions are you using?
- What traceability measures are you putting in place?
- What tools are using or do you know about?

# API Framework for Digital Government

**https://bit.ly/API-framework**


https://bit.ly/API-framework

https://bit.ly/API-framework-tool

mark@platformable.com

# Security

- How regularly do you conduct security audits of your APIs?
- Do you review your APIs against the OWASP Top 10?
- Do you use OpenAPI Specification files to document the security requirements and aspects of your APIs?

# Traceability

- What traces do you carry out to determine if API calls follow the correct authorisation to action path?
- What metrics are you collecting? Who collects them? Who analyses and acts on them?
- What logs are you collecting? How are you ensuring logs do not collect private data?