OWASP API Security Top 10 and Beyond part1

Prepared by: Falilat Owolabi
Date: Dec 5th, 2025

This week, I understand what led to OWASP, a non profit making foundation established on Dec 1st 2021 to develop the Owasp API security Top 10. I delve deeper into the first 3 OWASP API Security top 10 which include, Broken Object Level Authorization, Broken Authentication and Broken Object Property Level Authorization. Below is a detailed summary of what I have understood better about each vulnerability.

OWASP (Open Worldwide Application security protocol) is created to help improve application security. The OWASP API Security Top 10 release was first published in 2019 and this release was led by 3 key factors surrounding the use API:

1) **The Rapid Rise of APIs:** Api was becoming one of the most used software on the internet and it is powering most of the most valuable data, as business do not need to create everything from scratch to get their goals implemented, Api is faster to adopt and implement in different aspect of the business easily rather than building feature for it which might be difficult and time consuming to build. APIs are a major business enabler, which explains the global rapid adoption.
2) **A Major Gap in Security:** The security of API was not considered in all Security tools that were being developed leaving a gap for protocols adopting API not knowing how to manage and secure their API leaving them vulnerable to data breaches.
3) **A New Leading Attack Vector:** The attack of APIs has nothing to do with the traditional attack vectors that involve reconnaissance, weaponization, delivery, exploitation, installation, command and control, actions on objective, for Api security breaches it only involves, identification of vulnerabilities straight to exploitation. For example , an attacker can use an insecure API and have direct access to sensitive data.

The 2023 OWASP API Security Top 10 list is put together by the project team based on, "internal research based on publicly available data (e.g. bug bounty platforms, news)"

Mapped to external resources.

The owasp api security top 10 are mapped to external resources which include Common Weakness Enumeration (CWE), other OWASP projects, and National Institute of Standards and Technology (NIST) guidance.

Understanding the external sources and how they are associated with the given OWASP API Security Top 10 risk will help provide you with additional insight.

OWASP API1. Broken Object Level Authorization (BOLA)

BOLA is one of the most common and easy to exploit vulnerabilities on the OWASP list and it is very severe because any attacker who is able to exploit a BOLA vulnerability can get access to sensitive information about other users and use the information to their advantage. Identifying a BOLA vulnerability in an application involves the modification of the ID in the request header for a GET request or the property object for a POST request. IDs can be anything from sequential integers, UUIDs, or generic strings.

Threat agents/Attack vectors	**Security Weakness**	**Impacts**
API Specific : Exploitability Easy	**Prevalence Widespread : Detectability Easy**	**Technical Moderate : Business Specific**
Attackers can exploit API endpoints that are vulnerable to broken object-level authorization by manipulating the ID of an object that is sent within the request. Object IDs can be anything from sequential integers, UUIDs, or generic strings. Regardless of the data type, they are easy to identify in the request target (path or query string parameters), request headers, or even as part of the request payload.	This issue is extremely common in API-based applications because the server component usually does not fully track the client's state, and instead, relies more on parameters like object IDs, that are sent from the client to decide which objects to access. The server response is usually enough to understand whether the request was successful.	Unauthorized access to other users' objects can result in data disclosure to unauthorized parties, data loss, or data manipulation. Under certain circumstances, unauthorized access to objects can also lead to full account takeover.

OWASP Preventive Measures include:
Implement a proper authorization mechanism that include access control and user policies and hierarchy, make sure at every point a user is requesting access to any data/information, check if they have authorization level to do so, do not use easy to guess or incremental ID for user identification and lastly make sure to test for all possible scenarios on authorization level.

OWASP API2: Broken Authentication

Authenticating a user is the process of confirming users are who they claim they are. A Broken Authentication is any weakness in the process of authenticating a user, Authentication-related vulnerabilities typically occur when an API provider either doesn't implement a strong authentication mechanism or implements an authentication process incorrectly

Threat agents/Attack vectors	**Security Weakness**	**Impacts**
API Specific : Exploitability Easy	**Prevalence common: Detectability Easy**	**Technical Severe: Business Specific**
The authentication mechanism is an easy target for attackers since it's exposed to everyone. Although more advanced technical skills may be required to exploit some authentication issues, exploitation tools are generally available.	Software and security engineers' misconceptions regarding authentication boundaries and inherent implementation complexity make authentication issues prevalent. Methodologies of detecting broken authentication are available and easy to create.	Attackers can gain complete control of other users' accounts in the system, read their personal data, and perform sensitive actions on their behalf. Systems are unlikely to be able to distinguish attackers' actions from legitimate user ones.

Authentication process should be done with best practices without compromising because if the authentication process can be exploited it can lead to attacker making modification to sensitive information or even deleting data from the system depending on what the attacker decide to do, Authentication is not only done on user, it include devices and system as it is paramount for the server to know who accessed it, with what was it accessed it with, and from where is it being accessed.

A weak authentication process include

- Weak password policy
- Credential stuffing
- Predictable token
- Misconfigured json web token

OWASP Preventive Measures

As a developer implementing the Authentication process should be aware of all the possible flow of authentication, read and fully understand what they mean and how they are used, always use the standard and not try to create a new wheel which has not been tested, use the Owasp Authentication cheatsheet. Api keys should not be used for authentication, they should only be used for API client Authentication.

Broken Object Property Level Authorization (BOPLA)

This is a combination of two vulnerabilities from the Owasp top 10 Api security which are

- ➔ Mass Assignment
- ➔ Excessive Data Exposure

Mass Assignment is a vulnerability that allows for users to alter/input sensitive data into the property object. Allowing a normal user to alter a property that is meant for super user can have a great impact on the flow of a business because if a normal user can determine if they can be an admin in the system and perform all features an admin can perform, they will take advantage of it to damage the business flow in a way a legit admin would not have done.

Excessive Data Exposure This is a vulnerability in which the Api discloses information more than what is required from a request like sending an entire object as response to a request, this happens because the APi refused to filter out sensitive information and unneeded data before sending a response, malicious user can use this information to understand the system on how it can be attacked.

Threat agents/Attack vectors	**Security Weakness**	**Impacts**
API Specific : Exploitability Easy	**Prevalence common: Detectability Easy**	**Technical Severe: Business Specific**

APIs tend to expose endpoints that return all object's properties. This is particularly valid for REST APIs. For other protocols such as GraphQL, it may require crafted requests to specify which properties should be returned. Identifying these additional properties that can be manipulated requires more effort, but there are a few automated tools available to assist in this task.	Inspecting API responses is enough to identify sensitive information in returned objects' representations. Fuzzing is usually used to identify additional (hidden) properties. Whether they can be changed is a matter of crafting an API request and analyzing the response. Side-effect analysis may be required if the target property is not returned in the API response.	Unauthorized access to private/sensitive object properties may result in data disclosure, data loss, or data corruption. Under certain circumstances, unauthorized access to object properties can lead to privilege escalation or partial/full account takeover.

OWASP Preventive Measures include:

Before exposing an endpoint to users, always check if the user has the authorization to access that endpoint. Avoid using generic methods such as to_json() and to_string(). Instead, cherry-pick specific object properties you specifically want to return.Allow changes only to the object's properties that should be updated by the client.Keep returned data structures to the bare minimum, according to the business/functional requirements for the endpoint.

End Of Report!