API Security Overview

Sun* Cyber Security Research Team

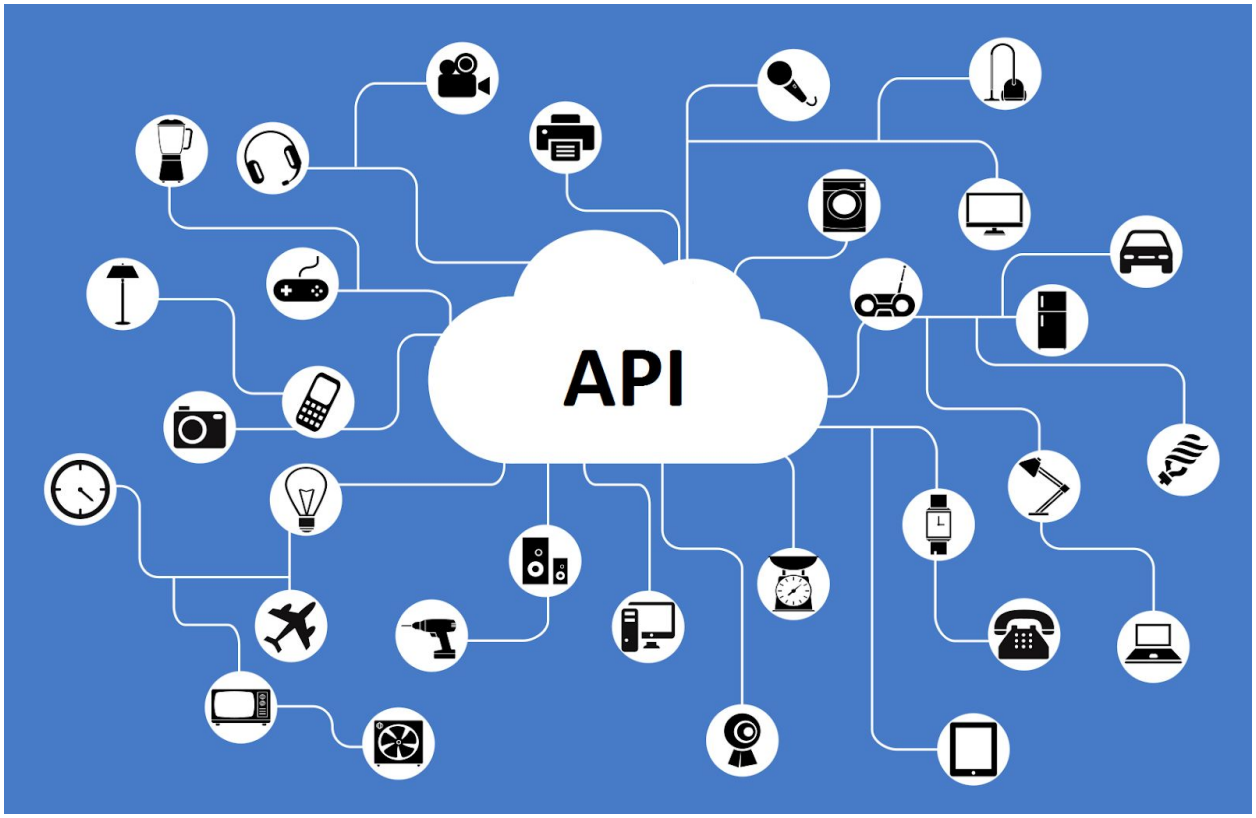

Table of Content

1.Why Do You Need API Security Testing?

API security breaches

- In September of 2018, hackers used a vulnerability in Facebook's Developer API to expose millions of users

- In 2019, A computer science student has scraped seven million Venmo transactions to prove that users' public activity can still be easily obtained

- In 2019, Starbucks Devs Leave API Key in GitHub Public Repo
- API allows data exchange between applications. If a hacker breaches API security, he/she can access sensitive data stored on your website.

- Data leaks of customers. This data is then sold in the black market.
- Defacement to your website & business. It can severely affect your & your brand's reputation in the market.

2. API Security Top 10

1. Broken Object Level Authorization
2. Broken User Authentication
3. Excessive Data Exposure
4. Lack of Resources & Rate Limiting
5. Broken Function Level Authorization
6. Mass Assignment
7. Security Misconfiguration
8. Injection
9. Improper Assets Management
10. Insufficient Logging & Monitoring

Threat Agents	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
API Specific	Easy: 3	Widespread 3	Easy 3	Severe 3	Business Specific
API Specific	Average: 2	Comon 2	Average 2	Moderate 2	Business Specific
API Specific	Difficult: 1	Difficult 1	Difficult 1	Minor 1	Business Specific

2.1 Broken Object Level Authorization

- OLA is an access control mechanism that is implemented at the code level to validate that one user can only access objects that they should have access to.

- API receives an ID of an object, and performs any type of action on the object, should implement object level authorization checks. The checks should validate that the logged-in user does have access to perform the requested action on the requested object.

- Leading to unauthorized information disclosure, modification, or destruction of all data

Threat agents/Attack vectors (3)	Security Weakness (2)	Impacts (3)
Attackers can exploit API endpoints by manipulating the ID of an object that is sent within the request	Access control detection is not typically amenable to automated static or dynamic testing	- Data disclosure to unauthorized parties, data loss, or data manipulation - Full account takeover

2.2 Broken User Authentication

- Permits credential stuffing whereby the attacker has valid usernames and passwords.

- Permits attackers to perform a brute force attack on the same user account, without presenting captcha/account lockout mechanism.
- Permits weak passwords.

- Sends sensitive authentication details (auth tokens and passwords) in the URL.
- Doesn't validate the authenticity of tokens.

- Accepts unsigned/weakly signed JWT tokens ("alg":"none")/doesn't validate their expiration date.
- Uses plain text, non-encrypted, or weakly hashed passwords.

- Uses weak encryption keys.

Threat agents/Attack vectors (3)	Security Weakness (2)	Impacts (3)
Engineer misconceptions about what are the boundaries of authentication and how to implement it correctly	- Lack of protection mechanisms - Misimplementation of the mechanism	Attacker gains control to other users' accounts in the system, read their personal data, and perform sensitive actions on their behalf

2.3 Excessive data exposure

- The API returns sensitive data to the client by design. This data is usually filtered on the client side before being presented to the user. An attacker can easily sniff the traffic and see the sensitive data.

Threat agents/Attack vectors (3)	Security Weakness (2)	Impacts (2)
Usually performed by sniffing the traffic to analyze the API responses, looking for sensitive data exposure that should not be returned to the user	- APIs are used as data sources, developers implements them in a generic way without thinking about the sensitivity of the exposed data - Automatic tools usually can't detect this type of vulnerability	Excessive Data Exposure commonly leads to exposure of sensitive data

2.4 Lack of Resources & Rate Limiting

- No authentication is required. Multiple concurrent requests can be performed from a single local computer or by using cloud computing resources.

Threat agents/Attack vectors (2)	Security Weakness (3)	Impacts (2)
Engineer misconceptions about what are the boundaries of authentication and how to implement it correctly	- APIs that do not implement rate limiting or APIs where limits are not properly set	Exploitation may lead to DoS, making the API unresponsive or even unavailable.

2.5 Broken Function Level Authorization

Perform deep analysis of the authorization mechanism:

- Can a regular user access administrative endpoints?

- Can a user perform sensitive actions (e.g., creation, modification, or erasure) that they should not have access to by simply changing the HTTP method (e.g., from GET to DELETE)?

- Can a user from group X access a function that should be exposed only to users from group Y, by simply guessing the endpoint URL and parameters (e.g., /api/v1/users/export_all)?

Threat agents/Attack vectors (3)	Security Weakness (2)	Impacts (2)
- API exposed to anonymous users or regular, non-privileged users. - Easier to discover in APIs are more structured, and the way to access certain functions is more predictable	- Implementing proper checks can be a confusing task, since modern applications can contain many types of roles or groups (e.g., sub-users, users with more than one role	Attackers access unauthorized functionality. Administrative functions are key targets for this type of attack.

2.6 Mass Assignment

- Objects in modern applications might contain many properties. Some of these properties should be updated directly by the client (e.g., user.first_name or user.address) and some of them should not (e.g., user.is_vip flag)

- API endpoint is vulnerable if it automatically converts client parameters into internal object properties, without considering the sensitivity and the exposure level of these properties

    - Permission-related properties: user.is_admin should only be set by admins.
    - Process-dependent properties: user.cash should only be set internally after payment verification.
      Internal properties: article.created_time should only be set internally by the application

Threat agents/Attack vectors (2)	Security Weakness (2)	Impacts (2)
- Requires an understanding of the business logic, objects' relations, and the API structure. - Expose the underlying implementation of the application along with the properties' names.	- Modern frameworks use functions that automatically bind input from the client into code variables and internal objects - Attackers use it to update or overwrite sensitive object's properties that the developers never intended to expose	Exploitation may lead to privilege escalation, data tampering, bypass of security mechanisms, and more.

2.7 Security Misconfiguration

- Appropriate security hardening is missing across any part of the application stack, or if it has improperly configured permissions on cloud services.
- The latest security patches are missing, or the systems are out of date.
- Unnecessary features are enabled (e.g., HTTP verbs).
- Transport Layer Security (TLS) is missing.
- Security directives are not sent to clients (e.g., Security Headers).
- A Cross-Origin Resource Sharing (CORS) policy is missing or improperly set.
- Error messages include stack traces, or other sensitive information is exposed

Threat agents/Attack vectors (3)	Security Weakness (3)	Impacts (2)
- Attackers will often attempt to find unpatched flaws, common endpoints, or unprotected files and directories to gain unauthorized access or knowledge of the system.	- Automated tools are available to detect and exploit misconfigurations such as unnecessary services or legacy options.	Security misconfigurations can not only expose sensitive user data, but also system details that may lead to full server compromise.

2.8 Injection

- Client-supplied data is not validated, filtered, or sanitized by the API.
- Client-supplied data is directly used or concatenated to

- SQL/NoSQL/LDAP queries, OS commands, XML parsers, and Object Relational Mapping (ORM)/Object Document Mapper (ODM)

- Data coming from external systems (e.g., integrated systems) is not validated, filtered, or sanitized by the API

Threat agents/Attack vectors (3)	Security Weakness (3)	Impacts (3)
- Attack API with malicious data through whatever injection vectors are available (e.g., direct input, parameters, integrated services, etc.), expecting it to be sent to an interpreter.	- Injection flaws are very common and are often found in SQL, LDAP, or NoSQL queries, OS commands, XML parsers, and ORM. - Review source code or scanner	Injection can lead to information disclosure and data loss. It may also lead to DoS,or complete host takeover.

2.9 Improper Assets Management

- The purpose of an API host is unclear
- There is no documentation, or the existing documentation is not updated.

- There is no retirement plan for each API version.
- Hosts inventory is missing or outdated.

- Integrated services inventory, either first- or third-party, is missing or outdated.
- Old or previous API versions are running unpatched

Threat agents/Attack vectors (3)	Security Weakness (2)	Impacts (2)
- Old API versions are usually unpatched and are an easy way to compromise systems.	- Outdated documentation makes it more difficult to find and/or fix vulnerabilities. - Lack of assets inventory and retire strategies leads to running unpatched systems	Gain access to sensitive data, or even takeover the server through old, unpatched API versions connected to the same

2.10 Insufficient Logging & Monitoring

- It does not produce any logs, the logging level is not set correctly, or log messages do not include enough detail.

- Log integrity is not guaranteed (e.g., Log Injection).
- Logs are not continuously monitored.

- API infrastructure is not continuously monitored.

Threat agents/Attack vectors (2)	Security Weakness (1)	Impacts (2)
- Attackers take advantage of lack of logging and monitoring to abuse systems without being noticed.	- It is almost impossible to track suspicious activities and respond to them in a timely fashion	Without visibility over on-going malicious activities, attackers have plenty of time to fully compromise systems