Practical DevSecOps

OWASP API Security Checklist for 2023

Also, Understand Best Practices to prevent it!

practical-devsecops.com | #certifiedapisecurityprofessional

1. Injection

The injection of malicious code or data into an API can result in unauthorized access to its sensitive data or functionality.

Best Practices to prevent it:

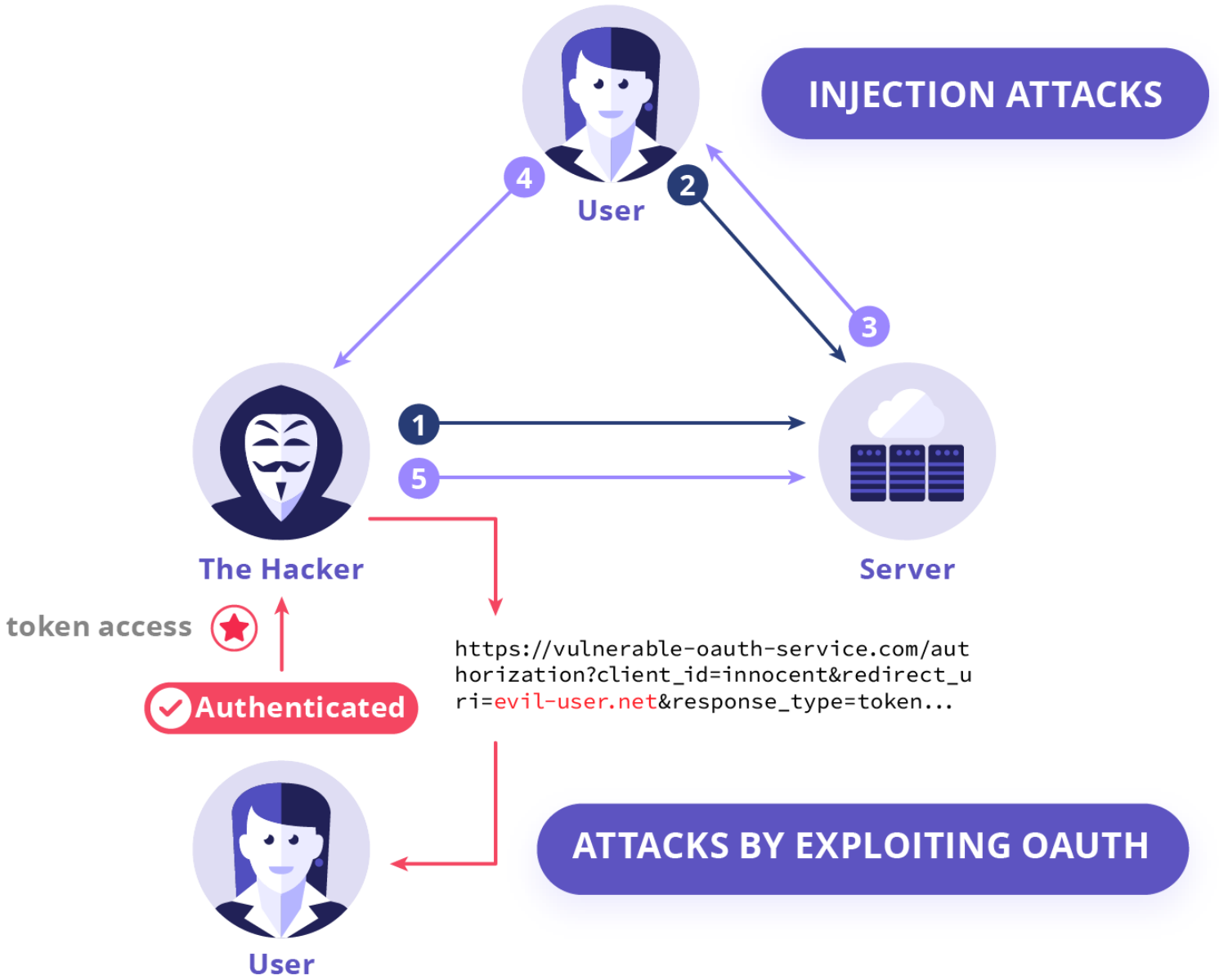
- Parameterize queries
- White-list input validation
- API rate limiting
- Web Application Firewalls (WAF)
- Limit database privileges
- Regular security audits
- Secure authentication and authorization
- Content-type validation
- API security testing

2. Broken User Authentication

Broken authentication and session management can enable attackers to impersonate valid users and compromise data privacy and infrastructure.

Best Practices to prevent it:

- Implement two-factor authentication
- Secure session management
- Enforce strict password policies
- Implement access control and authorization
- Regularly reviewing session logs
- Token-Based Authentication
- Account lockouts and brute-force protection:
- User account monitoring:
- Security incident response



1. An attacker introduces malicious code into a server
2. A victim accesses the compromised server and gives input
3. Server returns page code with injected script
4. Victims browser executes script and sends session cookies to attacker
5. Attacker hijack the user session

3. Excessive Data Exposure

Excessive data exposure is an API vulnerability that allows unauthorized access to sensitive information, including PII, financial data, and business data. Attackers exploit this vulnerability through misconfigured security settings, weak authentication measures, or exposed error messages.

Best Practices to prevent it:

- Implement strong access controls
- Use encryption to protect data
- Implement input validation techniques
- Use proper error-handling techniques
- Implement secure data handling procedures
- Schedule regular security audits

4. Lack of Resources & Rate limiting

A lack of rate-limiting strategies or resources can place an API at risk of Denial of Service (DoS) or Distributed DoS (DDoS) attacks, which can compromise its functionality and data availability.

Best practices to prevent it:

- Implement rate-limiting techniques, such as rate-limiting headers, caching, and use of AWS WAF.
- Use token-based HTTP authentication to prevent credential-stuffing attacks.
- Enable CAPTCHA for suspicious activities.

5. Broken Function-Level Authorization

Broken function-level authorization involves unauthorized access to sensitive functions or data due to misconfigured or weak access controls. This allows actors to perform escalated actions potentially, leading to data breaches or application hijacking.

Best practices to prevent it:

- Implement strict access controls to ensure appropriate role-based access to sensitive data.
- Use an automated access control mechanism.
- Implement regularly scheduled device updates.
- Stay current with information and vulnerability feeds and exploit databases.

6. Mass Assignment Vulnerabilities

Mass assignment vulnerabilities allow an anonymous actor to assign values without validation or filtering. As a result, unauthorized access and modification of data can take place this way.

Best Practices to prevent it:

- Use specialized frameworks that prevent mass-assignment attacks.
- Use a whitelist object binding within the application's data service layer.
- Use an immutable or read-only objects policy to prevent unnecessary objects modification.

7. Security Misconfiguration

Security Misconfiguration Improperly configured systems and software pose risks to APIs. Common security misconfigurations include insufficiently secured cryptography protocols, incorrect file permission configuration, and poor endpoint protection.

Best Practices to prevent it:

- Follow the OWASP secure coding principles and guidelines.
- Enforce strict access controls
- Use a secure configuration management process that reduces the attack surface of the API.

8. Improper Asset Management

Improper Asset Management Poor asset management or failure to identify vulnerable applications make it easier for attackers to exploit APIs and gain access to sensitive data and infrastructure.

Best Practices to prevent it:

- Use well-documented inventory lists of all physical and virtual devices.
- Use a comprehensive asset discovery system to identify vulnerability exposures and risks.
- Regularly test application environments to identify any potential vulnerabilities and gaps in security practices.

9. Insufficient Logging and Monitoring

Insufficient logging and monitoring make it difficult to detect and prevent data breaches as it prevents businesses from keeping an eye on assets.

Best Practices to prevent it:

- Use an automated logging and monitoring system to keep track of user and client activity.
- Use Security Information and Event Management (SIEM) tools.
- Identify, analyze, investigate, and respond to suspicious activity on the API in real-time via alerts and notifications.

10. Broken Object-Level Authorization

Broken object-level authorization is a critical API security vulnerability that allows attackers to gain unauthorized access to sensitive data by bypassing authorization checks. Attackers do this by manipulating object ID values, parameters, and authorization frameworks.

Best Practices to prevent it:

- Use Role-Based Access Control (RBAC) technology
- Properly implement authorization checks at the object level.
- Validate all inputs when accessing object data
- Implement proper session management

Become an API Security Expert with Us!

Certified API Security Professional

Link in the description

practical-devsecops.com | #certifiedapisecurityprofessional

Practical DevSecOps

Making Product Security Accessible to Everyone