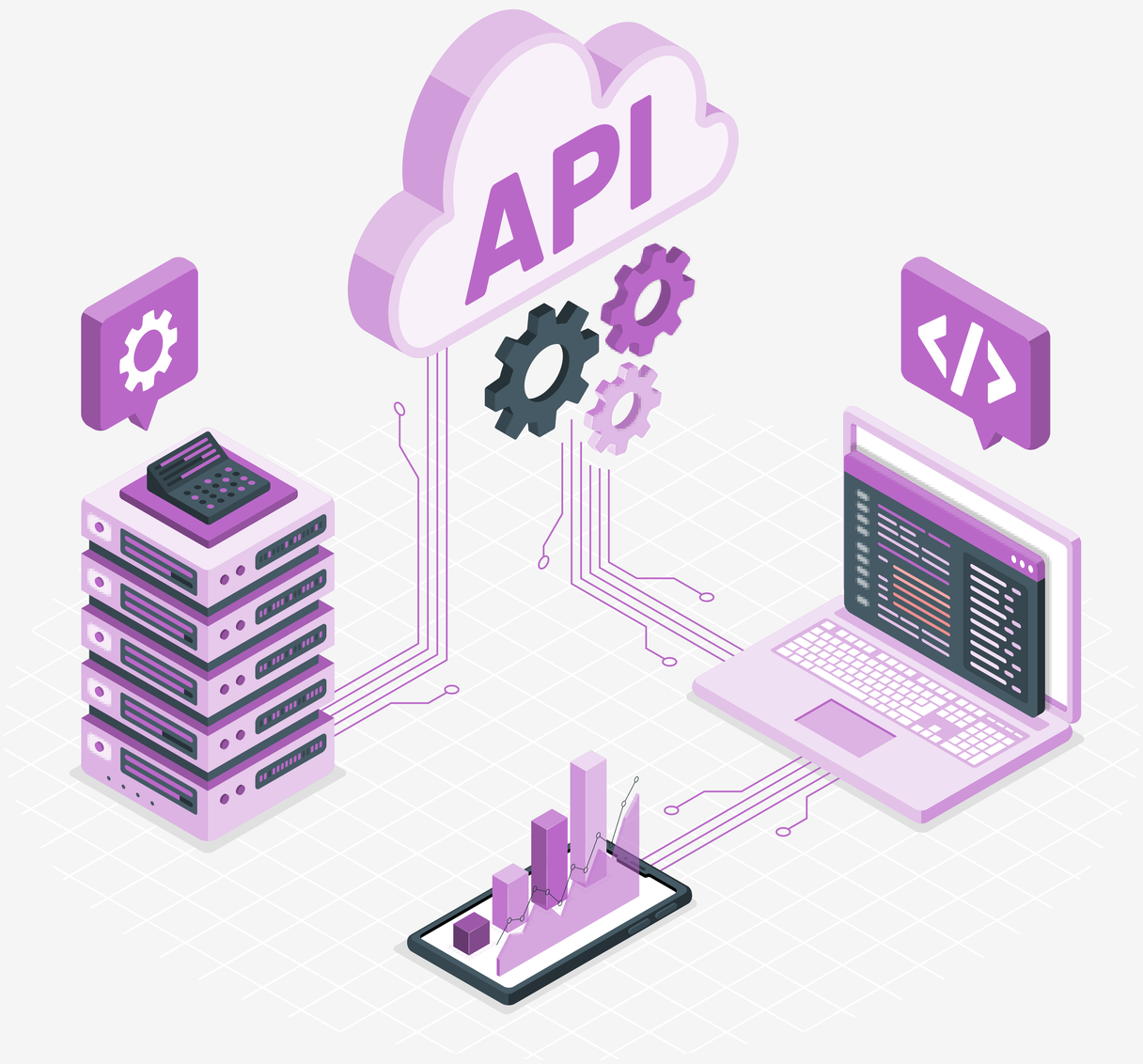
API Security Checklist



Nabi Karampoor

@thisisnabi

Mohamed Abukar

@moabukar

1- Authentication

- Implement strong authentication mechanisms such as JWT (JSON Web Tokens), OAuth, or OpenID Connect.
- Use secure password storage techniques, such as salted hashing or encrypt, to protect user passwords.
- Consider implementing multi-factor authentication for sensitive operations or privileged access.
- Protect against brute force attacks by enforcing account lockouts or rate limiting.

2- Authorization

- Implement role-based access control (RBAC) or claims-based authorization to restrict access to API resources based on user roles or permissions.
- Validate user roles or claims on the server-side before allowing access to sensitive operations or data.
- Implement attribute-based authorization to enforce fine-grained access control at the controller or action level.

3- Input Validation

- Validate and sanitize all input received from clients to prevent common attacks like cross-site scripting (XSS), SQL injection, or command injection.
- Use parameter binding techniques that automatically validate input, such as model binding or request validation.
- Avoid dynamic SQL queries or raw SQL concatenation and instead use parameterized queries or an ORM (Object-Relational Mapping) tool.

4- Secure Transmission

- Enforce the use of HTTPS (TLS/SSL) to encrypt the communication between clients and the API server.
- Disable or remove support for weak SSL/TLS protocols and ciphers to mitigate known vulnerabilities (e.g., SSLv3, TLS 1.0/1.1, weak cipher suites).
- Implement HSTS (HTTP Strict Transport Security) to enforce HTTPS usage across all requests.

5- Input and Output Data

- Validate and sanitize all input and output data, including query parameters, headers, and request/response bodies.
- Implement input/output validation and encoding techniques, such as XML/JSON parsing libraries that protect against XML/JSON external entity (XXE) attacks or deserialization vulnerabilities.
- Apply output encoding or HTML encoding to prevent cross-site scripting (XSS) attacks.

6- Error Handling

- Avoid displaying detailed error messages or stack traces to clients in production environments. Instead, log them securely on the server and provide generic error messages to clients.
- Implement structured error responses with appropriate HTTP status codes to indicate the result of API requests.
- Be cautious about leaking sensitive information in error responses, such as database connection strings or internal server paths.

7- Logging and Monitoring

- Implement comprehensive logging to capture security-related events, such as authentication failures, authorization errors, or suspicious activities.
- Monitor and analyze log data to detect anomalies or potential security breaches.
- Utilize intrusion detection and prevention systems (IDS/IPS) or security information and event management (SIEM) tools for real-time threat detection and response.

8- Secure Configuration

- Keep sensitive configuration settings, such as API keys, database credentials, or encryption keys, outside the application code and store them securely, such as in environment variables or a secure configuration store.
- Regularly review and update the security configuration of the API server, including server hardening, firewall rules, and security patches.

9- API Rate Limiting

- Implement rate limiting mechanisms to prevent abuse, DDoS attacks, or excessive resource consumption by a single client.
- Apply rate limits based on client IP addresses, API keys, or user accounts to control the number of requests per unit of time.

10- Third-Party Libraries and Dependencies

- Regularly update and patch all third-party libraries and dependencies used in your API to mitigate known security vulnerabilities.
- Monitor security advisories and subscribe to notifications to stay informed about any security patches or updates.

11- Security Testing

- Perform regular security testing, including vulnerability scanning, penetration testing, and security code reviews, to identify and address potential security flaws.
- Conduct thorough security testing during the development lifecycle and before deploying the API to a production environment.
- Use automated security testing tools, such as static analysis tools or dynamic application security testing (DAST) tools, to identify common vulnerabilities and security weaknesses.
- Collaborate with security professionals or third-party security firms to perform comprehensive security assessments and ensure a thorough evaluation of your API's security posture.
- Regularly review and update security test cases and scenarios to account for emerging threats and attack vectors.

While all the items in the checklist are important, these five items form the core pillars of securing a Web API.

- Authentication and Authorization
- Secure Transmission
- Input Validation
- Secure Configuration
- Security Testing