Top 7 REST API Security Strategies to Secure Your Endpoints :

Table of Content

What is REST API?

REST, short for Representational State Transfer, is an architectural style defining constraints and principles for designing networked applications. REST APIs, in turn, are the practical embodiment of these principles regarding communication between clients and servers.

At the heart of REST lies simplicity. RESTful APIs utilize standard HTTP methods such as GET, POST, PUT, and DELETE to perform predefined sets of operations on resources. These resources can be data, services, or any information that can be represented digitally.

REST APIs are known for their flexibility in data format, accommodating JSON, XML, plaintext, and more. They enable developers to create dependable, user-friendly systems seamlessly interacting with various devices and platforms.

However, this flexibility and openness expose REST APIs to many security threats. As cyber threats evolve, it becomes increasingly crucial to implement robust **API security strategies(A)** to protect these vital endpoints.

7 Key Strategies to Strengthen REST API Security

1. Limit HTTP Verb Operations

REST APIs allow apps to execute different HTTP verb operations. Attackers can intercept and exploit your apps by using some HTTP verb operations. For instance – GET, PUT, DELETE, and POST, among others. It is a REST API security best practice to limit such insecure HTTP methods.

Preventing the usage may not be possible in some cases. In such cases, you must develop and apply strict policies. Verb operations that are not on the allowed list should be rejected. You should assess these insecure verb operations with strict authorization policies. This way, only authorized users can delete, add, or modify resource collections.

2. Use TLS (HTTPS)

REST APIs use HTTP requests to access and use data. Attackers may compromise keys, authentication parameters, and data in transit during client-server communications. You can prevent this by enabling secure HTTP or HTTPS connections.

Deploy the latest version of TLS/ SSL certificates to protect your data in transit. Also, purchase **TLS(B)** from a trustworthy Certificate Authority (CA).

3. Authenticate Every User

You must validate every user to ensure you know who is calling your APIs. Attackers easily access your API resources and data when you don’t verify users.

You can use any of the following to authenticate users:

Username and password

API keys
JSON tokens
Access tokens

Make sure to implement strong password policies. Store passwords in a secure manner. Implement multifactor authentication to ensure robust REST API security.

Consider using OAuth2 if you offer single sign-on (SSO). **SSO(C)** enables users to verify themselves using third parties like Google, Microsoft, and Azure. Users don't have to sign up from scratch. And you don't have to maintain login/ logout or safe-keep passwords.

OAuth2 is an authorization protocol that allows users to secure access to resources. It works using access tokens. Even though it doesn't directly handle authentication, it helps the process.

4. Restrict Access to Resources

This is one of the best ways to secure rest API endpoints. When users have **just enough access** to API resources, you can prevent various API attacks.

Use pagination when APIs return large volumes of data. With pagination, you limit the data clients can access in a single request. Otherwise, attackers may cause crashes and downtimes, requesting all resources in a single request.

Create Content Security Policies (CSP) to prevent unauthorized access. Define what content types are allowed in your REST API. The server should reject those requests outright if there are mismatches between the allowed and requested content types. Doing so can avert different lethal attacks, including XSS attacks.

5. Use Rate Limiting

Another important strategy is to use rate limiting. Without **rate limiting(D)**, users can send voluminous requests within short time spans. These request floods make the API unresponsive. The API becomes unavailable to genuine users.

With rate limits, users cannot exceed the defined number of requests within a specific time frame. If they do, their access to the API will be temporarily blocked. Rate limiting helps prevent certain types of DDoS and DoS attacks, brute force attacks, etc.

6. Validate Input Parameters

REST APIs pass information from client to server using

- Path
- Request body
- Query
- Header parameters

You must validate inputs from each parameter before reaching the application logic/ server. This is critical for REST API protection. Use strong input validation checks, rejecting all requests failing these tests.

Consider these top three validation practices:

Implement Positive Security Models: This approach focuses on defining strict boundaries for API input using Swagger files (OpenAPI Specifications). It whitelists allowed input values or patterns, ensuring that only valid data is processed, effectively blocking non-conforming data.

In AppTrana WAAP, DevSecOps teams possess the ability to conduct vulnerability scans on APIs and request positive security policies for each individual API endpoint.

Validate Data Attributes: Ensure input data matches expected criteria by validating data type, format, range, and length. This safeguards against common security issues like injection attacks and maintains data integrity.

Use Validation Libraries: Leverage validation libraries and frameworks available in modern programming languages (e.g., Django validators for Python, Express-validator for Node.js). These simplify input validation, covering various scenarios and supporting custom validation rules.

7. Frequent API Security Testing

Consistently perform security evaluations for your APIs, involving activities such as penetration testing, vulnerability scanning, and code reviews, to detect and rectify potential security concerns.

By utilizing the **Indusface Infinite API scanner(E)**, you can access unlimited scans, encompassing both manual penetration and revalidation tests, all under a single license.