## API Security Best Practices

**API Middleware Controller Best Practices**

Middleware is a special types of controller executed after request but before in response. It is a type of filtering mechanism to ensure API securities and more. Middleware acts as a bridge between a request and a response.

**Middleware Uses:**

* Use to implement API key, user-agent restriction, CSRF, XSRF security, token based API authentication.
* Use to implement API request rate limit.
* Logging of incoming HTTP requests.
* Redirecting the users based on requests.
* Middleware can inspect a request and decorate it, or reject it, based on what it finds.
* Middleware is most often considered separate from your application logic.
* Middleware gives you enough freedom to create your own security mechanism.

**Output Validation**

**Output Header:**

* Provide proper http response status code.
* Provide proper content type, file type if any.
* Provide cache status if any.
* Authentication token should provide via response header.
* Only string data is allowed for response header.
* Provide content length if any.
* Provide response date and time.
* Follow request-response model described before.

**Output Body:**

* Avoid providing response status, code, message via response body
* Use JSON best practices for JSON response body.
* For single result, can use String, Boolean directly.
* Provide proper JSON encode-decode before writing JSON Body.
* Follow discussion on JSON described before.

**Request Rate limit- Throttling**

API limiting, which also known as rate is limiting, is an essential component of Internet security, as DoS attacks can tank a server with unlimited API requests. Rate limiting also helps make your API scalable. If your API blows up in popularity, there can be unexpected spikes in traffic, causing severe lag time.

**CSRF/XSRF Protection**

Cross-site request forgery attacks (CSRF or XSRF for short) are used to send malicious requests from an authenticated user to a web application.

* Use request-response header to pass CSRF token
* CSRF token should be unique for every session
* For self API CSRF token works well.

**User Agent Protection**

User agent is a request header property, describe client identity like operating system, browser details, device details etc. Moreover every web crawler like Google crawler, Facebook crawler has specific user-agent name.

* Using user agent we can prevent REST API from search engine indexing, social media sharing.
* Can stop subspecies request from who is hiding his identity.
* We can add user agent along with REST API usage history.
* We can add device/OS usage restriction.

**Bearer Authentication/ Auth 2.0**

Bearer authentication (also called token authentication) is an HTTP authentication scheme that involves security tokens called bearer tokens, passes through request-response header. In General JSON Web Tokens JWT used for this purposes.

**JWT (JSON WEB TOKEN):**

* Compact and self-contained way for securely transmitting information between parties as a JSON object.
* Information can be verified and trusted because it is digitally signed.