**Web Security Best Practices**

Security practices may varies

1. May varies from application to application.
2. May varies from developer to developer.
3. May varies from environment to environment.
4. May varies from use case to use case.

But we have to know

1. The best practices
2. Know about the security layers.
3. Security Placement.

**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.

***Request Rate limit-Throttling***

We need to make sure our APIs are running as efficiently as possible. Otherwise, everyone using your database will suffer from slow performance. Performance is not the only reason to limit API requests, either.API limiting, which also known as rate is limiting, is a essential component Of internet security, as DOS attack 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 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.

***API Key***

* This is the most straightforward method and the easiest way for auth.
* With this method, the sender places a username: password/ID/keys into the request header.
* The credentials are encoded and decode to ensure safe transmission.
* This method does not require cookie, session IDs, login pages, and other such specialty solutions.

***Bearer Authentication/Auth 2.0***

Bearer authentication is an HTTP authentication scheme that involves security tokens

called bearer tokens, passes through request-response header. In general JSON web token 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.

**Uses of JWT:**

**Authorization:** Allowing the user to access routes, services and resources.

**Information Exchange:** Way of securely transmitting information between parties.

***JWT structure***

* Header
* Payload
* Signature

**JWT payload:**

* Registered claims: iss(issuer), exp(expiration time), sub(subject), aud(audience)
* Public claims: These can be defined at will by those using JWTs.
* Private claims: These are the custom claims created to share information between parties.

**JWT Signature:**

* To create the signature part take the encode header
* Take the encoded payload, a secret
* The algorithm specified in the header