Date ..../..../.....

OWASP TOP-10

① Injection

→ Occurs when untrusted data is sent to an interpreter to trick the interpreter into executing unintended commands or accessing data without proper authorization.

→ Some common injections are SQL, NoSQL, LDAP, OS command etc.

→ To Prevent injection:
1. Use Safe API
2. Use whitelist server side validation
3. Use LIMIT to prevent mass disclosure

② Broken Authentication

→ Occurs when application function related to authentication and session management aren't correctly implemented allowing attacker to compromise sensitive data

→ To Prevent Broken Authentication:
1. implement multi-factor authentication
2. Implement weak password check
3. Limit or increasingly delay failed login attempts.

Date ..../..../.....

(3) Sensitive Data Exposure

→ Occurs when web applications and APIs fails to protect sensitive data like credit card details, health care info, PII etc.

→ To Prevent Sensitive DATA Exposure :—

1. Classify the data and apply controls.
2. Don't store sensitive data unnecessarily.
3. Use strong encryptions and ensure they are up-to-date.
4. Disable caching that contain sensitive data.

(4) XML External Entities (XXE)

→ Occurs in XML Parser that evaluate external entity references causing disclosure of internal files.

→ If attacker controls the external entity, they can access and manipulate sensitive data or execute arbitary code on server-side

→ To Prevent XXE :-

1. Try using less complex data formats.
2. Patch and/or update all XML processors and libraries
3. Implement whitelist serverside input validation.
4. Use SAST tools to detect XXE in code.

Date ..../..../.....

(5) Broken Access Control

→ Occurs when a user is allowed to do what they weren't allowed to do.
→ Attackers can exploit these flaws to access unauthorized functionalities and/or data.
→ To Prevent Broken AC :-
(1) Implement AC Mechanisms
(2) Enforce unique app. business limit requirements
(3) Log AC failures.
(4) Rate limit API and controller access

(6) Security Misconfiguration

→ Result of insecure default configurations, incomplete or ad hoc config., open cloud storage, misconfigured HTTP headers etc.
→ To Prevent Security Misconfiguration :-
(1) A Repeatable Hardening process.
(2) Remove or don't install unused features and frameworks.
(3) Review and update configs. as part of patch management process.
(4) Automated process to verify the effectiveness of the configs and settings.

Date ..../..../.....

⑦ Cross-Site Scripting (XSS)

→ Occurs when attacker execute scripts in the victim's Browser which can hijack user sessions, deface web pages or redirect user to malicious sites.

→ There are 3 forms of XSS :-

① Reflected XSS :- injected script is reflected back to the user from the server

② Stored XSS :- Injected script is stored on server which can affect other user visiting the site.

③ DOM XSS :- injected code is executed when the DOM is modified.

→ To prevent XSS :-

① Use frameworks that escape XSS by design.
② Escape untrusted HTTP request data
③ Apply context-sensitive encoding.
④ Enable a CSP (Content security Policy).

⑧ Insecure Deserialization

→ Occurs when an application deserialize untrusted data which allow attacker to execute code, bypass security & tamper data.

→ To Prevent Insecure deserialization :

① Implement Integrity check like Digital Signature.
② Enforce strict type constraints during deserialization
③ Isolate and run code that deserialize in low privilege environment.
④ Logging deserialization exceptions & failures

Viral

Date ..../..../.....

(9) Using Components with known Vuln.

→ Occurs when a vuln component (libs, fws or other s/w) is exploited. Apps and API using comp. with known vuln. may undermine app defenses

→ To Prevent this:-

1. Remove unused and unnece. comps., files etc.
2. Inventory versions of components and monitor sources for vulnerability.
3. Obtain comp. from official sources.
4. monitor comps. that are unmaintained or don't create security patches.

(10) Insufficient logging and Monitoring

→ Attackers rely on the lack of monitoring and timely response to achieve their goal without being detected.

→ To Prevent this:-

1. Ensure all login, AC failure and server-side input validation failures are logged.
2. Ensure that logs are generated in a format that can be easily consumed.
3. Establish effective monitoring and alerting
4. Establish or adopt an IR and recovery plan.

Date ..../..../.....

* Cross Site Request Forgery (CSRF).

→ Occurs when an attacker tricks the victim into clicking on malicious link or visiting a malicious website. This results in sending a malicious request from victim's browser to the targeted web app using victim's session cookies or authentication credentials.

→ To Prevent CSRF :-

1. Check the referer headers.
2. Use same-site cookies.
3. Use HTTPS.
4. Implement Request validation.

* File Inclusion (FI).

→ Occurs when attacker is able to include and execute remote file on web servers.

→ It is of 2 types :-

1. RFI :- Remote FI. Attacker provide URL that point to a remote file.
2. LFI :- Local FI. Attacker specify path to a sensitive file on the server

→ To Prevent FI :-

1. Validate and Sanitize the input by white-listing.
2. Use absolute paths.
3. Limit file permissions.
4. Use a CSP

Date ..../..../.....

* Click Jacking

→ User is tricked into clicking on a button or link on a web page that is invisible or disguised by another element.

→ To ~~Prev~~ Prevent ClickJacking:

1. Implement X-frame option
2. Use frame-busting code.
3. Use CSP and JS.

* File Upload

→ Occurs when anyone is allowed to upload file to the server as attacker can upload malicious file to infect server.

→ To prevent file upload:-

1. File type validation should be done.
2. Scan file for Malicious content.
3. Store file outside web roots.

* Insecure Captcha

→ Insecure captchas can easily be bypassed by bots, defeating the purpose of captcha.

→ To prevent this:-

1. Use strong Captcha.
2. Regularly update captcha algorithm.
3. Perform regular security audit.

Date ..../..../.....

* Server-side Request forgery (SSRF).

→ Occurs when a web app fetch a remote resource without validating the user-supplied URL.

→ To Prevent SSRF :-

1. Disable HTTP redirection.
2. Sanitize and validate input.
3. Do not send raw responses to client.