Assignment 1
Name: Niraianbu Porchezhian

OWSP top 5 vulnerabilities

The Latest List of OWASP Top 10 Vulnerabilities and Web Application Security Risks

The newest OWASP Top 10 list came out on September 24, 2021 at the OWASP 20th Anniversary. If you're familiar with the 2020 list, you'll notice a large shuffle in the 2021 OWASP Top 10, as SQL injection has been replaced at the top spot by Broken Access Control.

1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8. Software and Data Integrity Failures
9. Security Logging and Monitoring Failures
10. Server-Side Request Forgery

Description and details of top 5 owsp vulnerebilities:-

1. ***Broken Access Controls***

Website security access controls should limit visitor access to only those pages or sections needed by that type of user. For example, administrators of an ecommerce site need to be able to add new links or add promotions. These functions should not be accessible for other types of visitors.
Developers must be encouraged to internalize “security first” discipline to avoid pitfalls, such as content management systems (CMS) that generate all-access permission by default (up to and including admin-level access). Broken access control can give website visitors access to admin panels, servers, databases, and other business-critical applications. In fact, this OWASP Top 10 threat could even be used to redirect browsers to other targeted URLs.

Broken Access Controls measures:
Broken access control vulnerability can be addressed in a number of ways:
• Adopt a least privileged approach so that each role is granted the lowest level of access required to perform its tasks.
• Delete accounts that are no longer needed or active.
• Audit activity on servers and websites so that you are aware of who is doing what (and when).
• If there are multiple access points, disable the ones that are not required at that moment.
• Keep servers lean by shutting down unnecessary services.

2. Cryptographic Failures

Data in transit and at rest — such as passwords, credit card numbers, health records, personal information, and business secrets — require extra protection due to the potential for cryptographic failures (sensitive data exposures). This is especially true if the data falls under any of the privacy laws such as GDPR, CCPA, and others. Is any data is sent in plain text? Are there any outdated or insecure cryptographic algorithms or protocols in use by default or in older code? Is it possible that default crypto keys are being utilized, that weak crypto keys are being generated and re-used, or that proper key management and rotation are being overlooked? Is it possible to check crypto keys into source code repositories? Is encryption not enforced, and is the received data encrypted?

Control measures for cryptographic failures:

- On forms that collect data, turn off autocomplete.

- Reduce/minimize the size of the data surface area.
- Encrypt data while it is in transit and at rest.
- Use the most up-to-date encryption techniques.
- Disable caching on data-collecting forms.
- Use Strong adaptive and salted hashing functions when saving passwords.

3.Injection

Injection vulnerabilities can occur when a query or command is used to insert untrusted data into the interpreter via SQL, OS, NoSQL, or LDAP injection. The hostile data injected through this attack vector tricks the interpreter to make the application do something it was not designed for, such as generating unintended commands or accessing data without proper authentication.

Any application that accepts parameters as input can be susceptible to injection attacks. The level of the threat is highly correlated with the thoroughness of the application's input validation measures.
Injection Remediation

Injection attacks can be prevented by any combination of the following approaches:

Segregate commands from data to avoid exposure to attacks that replace data with unintended command execution.

Code SQL queries with parameters rather than structuring the command from user input content only. These are called parameterized queries or prepared statements.

Eliminate the interpreter altogether through the use of a safe API.

Implement positive server-side validation as well as an intrusion detection system that spots suspicious client-side behaviors.

4. Insecure Design

Insecure design is a wide term that encompasses a variety of flaws and is defined as "missing or poor control design." Threat modeling, secure design patterns, and reference architectures are among the new categories for 2021, with a demand for increasing the usage of threat modeling, safe design patterns, and reference architectures. As a community, we must move beyond "shift left" coding to pre-code tasks that are important to the Secure by Design principles.

<u>Control measure Insecure Design :</u>

To help analyze and build security and privacy-related measures, establish and use a safe development lifecycle with AppSec professionals.

Create and use a library of secure design patterns or components that are ready to use.

Use threat modeling for crucial authentication, access control, business logic, and key flows.

User stories should include security language and controls.

Integrate plausibility checks into your application at each level (from frontend to backend).

To ensure that all important flows are resistant to the threat model, write unit and integration tests. Make a list of use-cases and misuse-cases for each tier of your app.

Depending on the exposure and protection requirements, divide tier tiers on the system and network layers.

Limit user and service resource consumption.

<u>5. Security Misconfiguration</u>

Gartner estimates that up to 95% of cloud breaches are the result of human errors. Security setting misconfigurations are one of the prime drivers of that statistic, with OWASP noting that, of the top ten, this vulnerability is the most common. There are many types of misconfiguration that expose the company to cybersecurity risk, including:

Accepting default settings that are insecure

Overly accessible cloud storage resources

Incomplete configurations

Misconfigured HTTP headers

Verbose error messages that contain sensitive information

Security Misconfiguration Remediation

Security misconfigurations can strike almost anywhere in the environment, including network-attached devices, databases, web and application servers, and containers. The following practices can help maintain a well-configured environment:

Use templates to deploy development, test, and production environments that are preconfigured to meet the organization’s security policies.

Leverage segmented application architectures that minimize the risk from an insecurely configured element; maintain a library of properly configured container images.

Deploy minimal platforms and remove unused features and services.

Continuously monitor cloud resources, applications, and servers for security misconfigurations and remediate detected issues in real time, using automated workflows wherever possible.