TryHackMe Labs - OWASP Top 10

1. **Broken Access Control:** Allows attackers to bypass authorization, allowing them to view sensitive data or perform tasks they aren't supposed to.
   - **Vulnerability:** Insecure Direct Object Reference (IDOR) allows attackers to bypass authorization.
   - **Exploitation**: Manipulate URLs or parameters to access sensitive data or perform unauthorized tasks.
   - **Mitigation**: Implement robust access controls, validate user permissions, and use indirect object references.

2. **Cryptographic Failures:** refers to any vulnerability arising from the misuse (or lack of use) of cryptographic algorithms for protecting sensitive information
   - **Vulnerability**: Misuse or lack of cryptographic algorithms compromises sensitive information.
   - **Exploitation**: Intercept or manipulate encrypted data to access sensitive information.
   - **Mitigation**: Use secure encryption algorithms, implement secure key management, and ensure proper SSL/TLS configuration.

3. **Injection:** These flaws occur because the application interprets user-controlled input as commands or parameters. Injection attacks depend on what technologies are used and how these technologies interpret the input.
   - **Vulnerability:** Injection flaws allow attackers to execute malicious commands or queries.
   - **Exploitation:** SQL Injection, Command Injection, etc., to access, modify, or delete sensitive data.
   - **Mitigation:**
     i. **Using an Allow list**: When input is sent to the server, this input is compared to a list of safe inputs or characters. If the input is marked as safe, then it is processed. Otherwise, it is rejected, and the application throws an error.
     ii. **Stripping input**: If the input contains dangerous characters, these are removed before processing. Dangerous characters or input is classified as any input that can change how the underlying data is processed.

4. **Insecure Design:** Insecure design refers to vulnerabilities which are inherent to the application's architecture. They are not vulnerabilities regarding bad implementations or configurations, but the idea behind the whole application (or a part of it) is flawed from the start. Most of the time, these vulnerabilities occur when an improper threat modeling is made during the planning phases of the application and propagate all the way up to the final app.
   - **Vulnerability**: Flawed application architecture or design.

- **Exploitation:** Varies depending on the specific design flaw.
- **Mitigation:** Perform threat modeling at the early stages of the development lifecycle and implement secure design principles.

5. **Security Misconfiguration:** Security Misconfigurations are distinct from the other Top 10 vulnerabilities because they occur when security could have been appropriately configured but was not. Even if you download the latest up-to-date software, poor configurations could make your installation vulnerable.
    - **Vulnerability:** Poor configuration of security settings.
    - **Exploitation:** Access sensitive data, exploit default credentials, or inject malicious code.
    - **Mitigation:** Implement secure configuration guidelines, regularly review and update configurations.

6. **Vulnerable and Outdated Components**
    - **Vulnerability:** Using outdated or vulnerable third-party components.
    - **Exploitation:** Exploit known vulnerabilities in outdated components.
    - **Mitigation:** Regularly update and patch components, use secure dependency management.

7. **Identification and Authentication**
    - **Vulnerability:** Flaws in authentication mechanisms.
    - **Exploitation:**
        i. **Brute force attacks:** If a web application uses usernames and passwords, an attacker can try to launch brute force attacks that allow them to guess the username and passwords using multiple authentication attempts.
        ii. **Use of weak credentials:** Web applications should set strong password policies. If applications allow users to set passwords such as "password1" or common passwords, an attacker can easily guess them and access user accounts.
        iii. **Weak Session Cookies:** Session cookies are how the server keeps track of users. If session cookies contain predictable values, attackers can set their own session cookies and access users' accounts.
    - **Mitigation:**
        i. To avoid password-guessing attacks, ensure the application enforces a strong password policy.
        ii. To avoid brute force attacks, ensure that the application enforces an automatic lockout after a certain number of attempts. This would prevent an attacker from launching more brute-force attacks.
        iii. Implement Multi-Factor Authentication. If a user has multiple authentication methods, for example, using a username and password and receiving a code on

their mobile device, it would be difficult for an attacker to get both the password and the code to access the account.

iv. Secure session management

8. **Software and Data Integrity Failures:** When talking about integrity, we refer to the capacity we have to ascertain that a piece of data remains unmodified. Integrity is essential in cyber security as we care about maintaining important data free from unwanted or malicious modifications.
    - **Vulnerability:** Lack of integrity checks on software or data.
    - **Exploitation:** Modify software or data without detection.
    - **Mitigation:** Implement digital signatures, hash checks, and secure update mechanisms.

9. **Security Logging and Monitoring Failures:** When web applications are set up, every action performed by the user should be logged. Logging is important because, in the event of an incident, the attackers' activities can be traced. Once their actions are traced, their risk and impact can be determined. Without logging, there would be no way to tell what actions were performed by an attacker if they gain access to particular web applications.
    - **Vulnerability:** Insufficient logging and monitoring.
    - **Exploitation:** Undetected malicious activity, regulatory non-compliance.
    - **Mitigation:** Implement comprehensive logging, monitor for suspicious activity.

10. **Server-Side Request Forgery (SSRF):** This type of vulnerability occurs when an attacker can coerce a web application into sending requests on their behalf to arbitrary destinations while having control of the contents of the request itself. SSRF vulnerabilities often arise from implementations where our web application needs to use third-party services.
    - **Vulnerability:** Coerced requests to arbitrary destinations.
    - **Exploitation:** Enumerate internal networks, abuse trust relationships.
    - **Mitigation:** Validate user input, implement network segmentation, monitor for suspicious requests.