**School of Computer Science**

**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

**DEHRADUN, UTTARAKHAND**



**IT SECURITY**

**Assignment File**

**(2023-2024)**

**for**

**4rd Semester**

**Submitted To: Submitted By:**

Dr Gopal Rawat Akshat Negi

Assistant Professor-SS B.Tech CSF [4th Semester]

School of Computer Science 500106533

Batch 2

**Technical Report: Insecure Design**

1. **Introduction**

Insecure designing be a critical vulnerability highlighted in the OWASP Top 10 (2021) as a major risk to software systems. It refers to an absence of secure considerations throughout the development lifecycle leading to flaws deeply ingrained within the application's core functioning. This report delves into the technical aspects of insecure design, exploring its description, forms of attack, countermeasures, and a real-life example for better understanding.

1. **Description of Insecure Design**

Insecure design arises when security ain ‘t prioritized during the software develop process. This manifests in various ways, including:

Lack of Threat Modelling: Failing to identify potential threats and vulnerabilities early on. Flawed Authorization and Authentication: Granting excessive privileges or not adequately verifying user identities. Insecure Data Storage: Storing sensitive data in an unencrypted format or without proper access controlling. Unvalidated User Input: Allowing users to inject malicious code into the application. Broken Session Management: Using predictable session IDs or neglecting session timeouts. Insecure Direct Object References (IDOR): Exposing sensitive data based on predictable identifiers in URLs or requests. These designing flaws create vulnerabilities that attackers can exploit to gain unauthorized access, steal data, disrupt operations, or compromise system integrity.

1. **Attack Forms**

Insecure designing can be exploited through various attack forms, including:

Injection Attacks (SQL Injection, XSS): Attackers inject malicious code into user input fields to manipulate the application's logic or steal data. Broken Authentication: Attackers exploit weaknesses in authentication mechanisms to gain unauthorized access to accounts or systems. Cross-Site Scripting (XSS): Attackers inject malicious scripts into web pages, which are then executed by users' browsers, potentially stealing session cookies or redirecting users to phishing sites. Man-in-the-Middle Attacks: Attackers intercept communication between users and the application to steal data or inject malicious code. Privilege Escalation: Attackers exploit vulnerabilities to gain higher privileges within the system than intended.

1. **Countermeasures**

Mitigating the risks associated with Insecure design requires a security-focused development approach. Here are some effective countermeasures:

Shift Left Security: Integrate security considerations into the early stages of development, such as threat modelling and secure designing patterns. Secure Coding Practices: Train developers on secure coding practices to prevent common vulnerabilities. Static and Dynamic Application Security Testing (SAST & DAST): Analyse code for vulnerabilities before and after deployment. Regular Penetration Testing: Conduct penetration testing to identify and address exploitable vulnerabilities. Input Validation and Sanitization: Validate and sanitize all user input to prevent injection attacks. Least Privilege Principle: Grant users only the minimum privileges necessary to perform their tasks. Data Encryption: Encrypt sensitive data at rest and in transit. Session Management Best Practices: Implement secure session management with strong randomness and timeouts.

1. **Real-Life Example: Insecure E-commerce Platform**

Imagine an e-commerce platform that allows users to store their credit card information for faster checkout. The developers, focused on user convenience, neglect to encrypt credit card data at rest. Here's how Insecure designing creates a vulnerability:

Insecure Data Storage: Unencrypted credit card data becomes vulnerable if attackers gain access to the database. This could lead to a major data breach and financial losses for customers.

1. **Conclusion**

Insecure designing be a persistent threat that can have serious consequences. By adopting a security-conscious development approach and implementing the countermeasures outlined above, developers can build more robust and secure software systems. Remember, security is not an afterthought; it needs to be woven into the fabric of the development process from the very beginning.