FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

BIS20303

KESELAMATAN WEB

Lab 1 (12/4/2025)

PREPARED FOR:

Encik Khairul Amin Bin Mohamad Sukri

PREPARED BY:

NAME	MATRIX
MUHAMMAD ZAHIN HARITH BIN ZULKIFLI	CI230141
MOHAMAD RAFI BIN ALMEN	CI230082
WAN HASYRAF BIN ABD RAHMAN	CI230079
MUHAMMAD IRFAN BIN ISHAK	AI220138

Link to Blog:

1.

a) What is OWASP, and why is it important?

The Open World Wide Application Security Project is also known as OWASP. It is an organization that provides useful information and tools free of charge in trying to help software become safer. Because OWASP provides exact steps on how to prevent common security issues, people accept it. It is essential because it teaches developers and companies alike how to protect their web applications from hackers and other dangers.

b) What is OWASP Top 10, and how does it help organizations?

The most important security risks that websites and apps are vulnerable to are listed in the OWASP Top 10. It is something of a roadmap that allows companies to determine where to focus the most attention first. Businesses can minimize their likelihood of being hacked, improve data protection, and ensure that they are using solid security practices by adhering to the guidance on this list.

c) How often is the OWASP Top 10 updated, and why does it change?

The Top 10 list is refreshed by OWASP every few years, normally every three or four years. Because new types of attacks are continually arising and old ones might decrease in frequency, they refresh it. The refreshes serve to keep developers aware of the latest threats and are derived from real data from security experts.

2.

a) What does injection mean?

Injection is one of the most significant and widespread security vulnerabilities in web applications, according to the OWASP Top 10 list. This vulnerability occurs when an application sends untrusted or improperly validated input to an interpreter, like an operating system shell, SQL database, or LDAP server. If the input is interpreted as a command or query, attackers may be able to execute unanticipated activities or collect unauthorized data. For instance, an attacker can change a query and potentially circumvent authentication or retrieve sensitive information by introducing malicious SQL code into input fields. This often occurs when developers dynamically construct queries using user input without using secure coding practices.

b) How attackers exploit the weakness using injection

SQL injection attacks begin when hackers identify vulnerable areas in an online application, such as URL parameters, login forms, or search bars, where they may surreptitiously insert malicious code. By carefully creating SQL instructions that trick the system into doing unexpected activities, they frequently bypass security filters by modifying queries or taking advantage of vulnerabilities. These malicious instructions have the ability to alter records, erase whole tables, or steal confidential information once they are introduced. In more serious situations, attackers may use database vulnerabilities to take over the server and execute system instructions or get access to files that are forbidden. If the application fails to adequately sanitize user input, these attacks might be disastrous.

c) Prevention: How can developers protect web applications from this risk?

By employing parameterized queries (prepared statements), developers may prevent SQL injection by making user input be handled as data rather than executable code. When dynamic queries cannot be avoided, escaping special characters and using stored procedures with strong input validation are also helpful. Whitelisting proper formats (such as using only alphanumeric characters for usernames) and rejecting anything questionable should be the goals of input validation. By automating secure queries, ORM frameworks (such as Hibernate or Entity Framework) provide an additional degree of security. In the case of a breach, least-privilege database accounts restrict harm, while frequent security audits, which include penetration testing, identify vulnerabilities early. Lastly, harmful payloads can be filtered out by web application firewalls (WAFs) before they enter the database.

3. Research a real-world web security breach that reflects your selected risk.

In 2012, Yahoo’s content platform known as Yahoo Voices became the target of a serious cyberattack. This incident serves as a prime example of an SQL injection vulnerability being exploited in the real world. The attackers gained access to Yahoo’s backend systems by injecting malicious SQL commands into user input fields that weren't properly validated.

As a result, they were able to extract sensitive user information, including around 453,000 email addresses and passwords. One major issue was that Yahoo had stored the passwords in plain text, meaning the exposed data could be immediately used by attackers. This failure to protect user credentials highlights the danger of combining poor input validation with weak data protection practices.

The breach also revealed a broader issue: a significant portion of affected users had reused the same passwords across multiple platforms. This opened the door to further exploitation, including attempts to access other services using the same credentials a technique known as credential stuffing.

This case is directly linked to the OWASP Top 10 risk category A03:2021 Injection, where untrusted data is processed as part of a command or query. Yahoo's lack of secure coding practices, such as failing to use parameterized queries or input sanitization, made this attack possible.

4. Answer the following in relation to the chosen risk:

a) What happened in the breach?

In mid-2012, Yahoo's publishing platform, Yahoo! Voices, was compromised by hackers who exploited a weakness in the system's database handling. This attack resulted in the exposure of around 453,000 user credentials, which included email addresses and unencrypted passwords. The data was accessible due to a lack of proper security practices, including storing passwords in plain text. Investigations later revealed a high rate of password reuse among users, especially across different platforms.

b) How does the breach relate to the selected OWASP risk?

This incident is a clear example of the OWASP Top 10 threat category: Injection Attacks specifically, SQL Injection. The attackers manipulated improperly handled input fields to send malicious SQL commands, allowing unauthorized access to user data. Furthermore, it also aligns with the Cryptographic Failures category, since the leaked passwords were not protected using encryption or hashing. This left users highly vulnerable to credential theft and reuse on other platforms.

c) What could have prevented the attack?

The breach could avoided through a combination of secure coding and data protection strategies:

- Use of Parameterized Queries: These ensure that user inputs are handled safely by the database and prevent malicious code execution.
- Encrypted Password Storage: Implementing strong hashing algorithms like bcrypt or Argon2 would have made the leaked data useless to attackers.
- Access Control and Principle of Least Privilege: Limiting database access could have reduced the scope of the breach.
- Regular Security Testing: Conducting penetration tests and code audits can uncover vulnerabilities early on.
- Web Application Firewalls (WAFs): These could have identified and blocked suspicious input behavior before it reached the backend systems.

Reference

1. Injection Theory | OWASP Foundation

2. A03 Injection - OWASP Top 10:2021

3. What Is an Injection Attack? | CrowdStrike

4. SQL Injection Attack: How It Works, Examples and Prevention

5. What is SQL Injection (SQLi) and How to Prevent Attacks

6. What happened in the Yahoo data breach? | Twingate