Initial Post

Collaborative Discussion 1: UML flowchart (updated)

Task:

Open source tools are available to create UML diagrams, some are listed below. This list is not exhaustive. The benefit of using such tools is that they ensure that the recognised UML components are used to represent the parts of the model correctly.

- Visual Paradigm
- Sequence Diagram
- Umbrello

Choose an open-source UML tool from the list above. Select one of the coding weaknesses which have been identified by OWASP and create a flowchart of the steps which may have led to the weakness occurring. Which UML models might you use to present the design of your proposed software, and why are they the most appropriate choice(s)?

Initial Post:

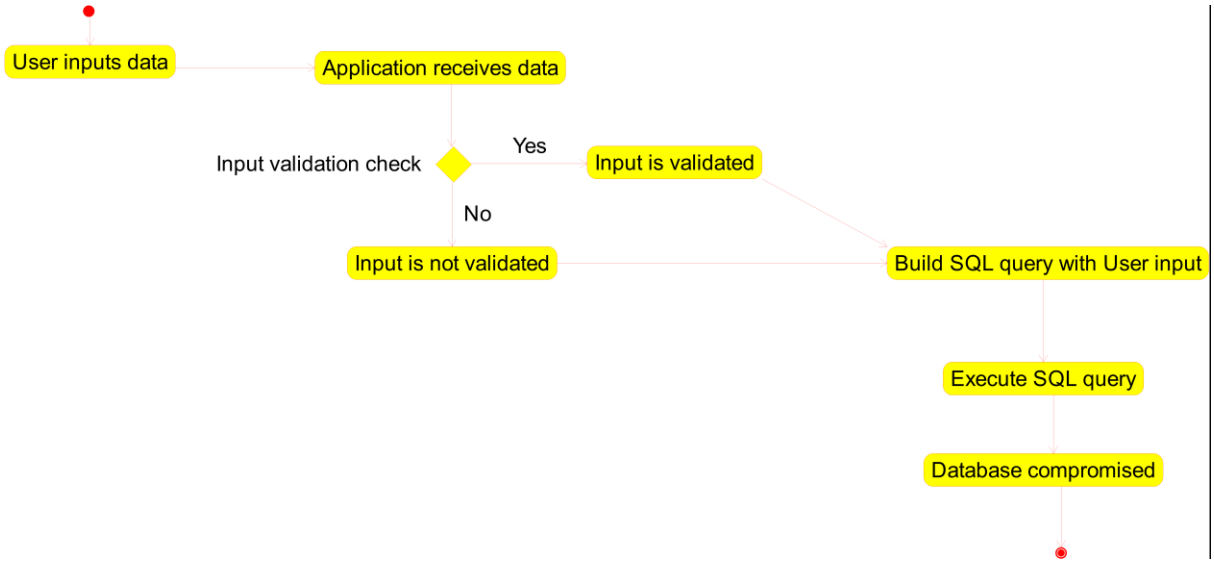
In modern web application development, protecting applications from vulnerabilities is essential, especially when handling sensitive information. SQL Injection (SQLi), a prominent threat listed in the OWASP Top 10, is one of the most prevalent and hazardous risks. SQLi attacks occur when applications inadequately validate or sanitize user input, enabling malicious actors to insert harmful SQL code into database queries. This can lead to unauthorized data access, database content alteration, or even complete compromise of the database (OWASP, 2023; Halfond, Viegas, & Orso, 2006).

SQLi vulnerabilities are particularly common in applications that directly incorporate user input into SQL queries without proper safeguards. This weakness allows attackers to alter the query structure by inserting malicious commands. For example, an attacker might input carefully crafted SQL code that, if not properly sanitized, could erase records, circumvent authentication, or extract sensitive information. Such breaches can result in severe data leaks, identity theft, or substantial financial and reputational damage to organizations (Boyd & Keromytis, 2004; Halfond et al., 2006).

Umbrello, an open-source UML modeling tool, may be used to create a flowchart showing the steps where input validation and security measures are essential to demonstrate the SQLi vulnerability route. By modeling this process in Umbrello, developers may gain a thorough understanding of the application's data flow. This enables them to pinpoint and fortify any weak places susceptible to SQLi attacks (Umbrello, 2023).

Flowchart: Steps Leading to SQL Injection Vulnerability

The following outlines a simplified flowchart illustrating the typical process leading to an SQLi attack. Each step emphasizes critical decision points where security measures should be enforced.



1. **Start:** The process begins when a user engages with the application, usually by entering data through forms or search fields. In the absence of protective measures, any input could potentially contain malicious SQL code designed to exploit the system (OWASP, 2023).
2. **User Inputs Data:** The user provides information that the application will utilize to construct an SQL query. This input may be legitimate or deliberately crafted to include SQL code. If the application fails to screen for SQL keywords or special characters, harmful SQL commands can be inserted at this stage (Boyd & Keromytis, 2004).
3. **Application Receives Data:** The application gathers the user-supplied input, preparing it for query formation. This step is vital as it represents the initial processing phase where potentially harmful data should ideally be identified and flagged before progressing further (Halfond et al., 2006).
4. **Input Validation Check:** At this point, the application should ideally conduct input validation to identify potentially harmful content. Proper validation examines SQL keywords, special characters, or unexpected input formats. Without effective validation or sanitization, malicious data may pass through

undetected, exposing the system to vulnerabilities (OWASP, 2023; Boyd & Keromytis, 2004).

5. **Build SQL Query with User Input:** The application constructs the SQL query by incorporating user input. If this input is not validated, the query may contain SQLi code that alters the intended SQL logic, enabling attackers to perform unauthorized actions. This is one of the most critical stages of proper input validation is lacking (Halfond et al., 2006).
6. **Execute SQL Query:** The SQL query is transmitted to the database for execution. If it contains malicious SQLi code, the database interprets it as valid SQL, allowing attackers to access, modify, or delete data they should not have permission to manipulate. This can result in data breaches and compromise data integrity (OWASP, 2023; Boyd & Keromytis, 2004).
7. **Database Compromised:** The final step illustrates the consequence of a successful SQLi attack. The database is now compromised, granting attackers unauthorized access to sensitive information or control over database operations. This can lead to data theft, loss of data, or complete system takeover (Halfond et al., 2006).

Conclusion

SQLi vulnerabilities pose a significant security risk that can result in severe breaches if left unaddressed. The flowchart, created using Umbrello, demonstrates how SQLi attacks can exploit vulnerabilities from the initial user input to the ultimate compromise of the database, highlighting areas where input validation is crucial. By identifying these key stages, software developers can implement critical security measures, including input validation, prepared statements, and parameterized queries, to reduce the risk of SQLi attacks (OWASP, 2023).

Tackling SQLi early in the software development process and utilizing UML modeling tools like Umbrello promotes a proactive approach to security. These models enable developers to visualize the vulnerability pathway and strategically implement safeguards against SQL Injection. By incorporating secure modeling practices, companies can substantially decrease SQLi risks, fostering the development of safer and more robust applications (Umbrello, 2023; OWASP, 2023).

References

Boyd, S.W. & Keromytis, A.D. (2004) 'SQLRAND: Preventing SQL injection attacks', *Lecture Notes in Computer Science,* pp. 292–302. doi:10.1007/978-3-540-24852-1_21.

Halfond, W. G., Viegas, J., & Orso, A. (2006) A Classification of SQL-InjectionAttacks and Countermeasures.

OWASP (2023) *SQL Injection | OWASP Foundation.* Available from: https://owasp.org/www-community/attacks/SQL_Injection [Accessed: 30 October 2024].

UML (2023). Umbrello Project - Welcome to Umbrello - The UML Modeller. Available from: https://uml.sourceforge.io/ [Accessed: 25 October 2024].