Unit 1: Introduction to Secure Software Development

Initial Post

Collaborative Discussion 1: UML flowchart

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:

I chose Umbrello for this task. Umbrello is an open-source Unified Modeling Language (UML) tool. It is commonly used for designing software models, including various UML diagrams such as class, sequence, and use case diagrams, making it well-suited for modeling and documenting system architectures. It is specifically designed for Linux systems but is cross-platform and integrates easily into various development environments (Umbrello, 2023). Notable for its simplicity and accessibility, Umbrello is often favoured by developers and educators for projects requiring UML visualization (UML Modeller, N.D.). This post explores Umbrello's strengths and weaknesses and

how it can be effectively used to model an injection vulnerability through a sequence diagram.

Strengths:

1. Open-Source and Cost-Effective: Umbrello's open-source nature makes it a cost-effective choice suitable for individuals, educational institutions, and small organizations. Its open accessibility allows developers to contribute to and enhance the functionality of the tool continuously (UML Modeller, ND).; Schardt & Chonoles, 2013).
2. Platform Compatibility: Primarily developed for Linux, Umbrello is also available on Windows and MacOS, offering flexibility to a wide range of users, especially those utilizing open-source operating systems (Roff, 2003).
3. Support for Multiple UML Diagrams: Umbrello supports diverse UML diagrams, including use case, class, sequence, and state diagrams, making it versatile for modeling various aspects of software development projects (Fowler, 2018).
4. Ease of Use: The simple drag-and-drop interface makes Umbrello beginner-friendly, an advantage, particularly in educational settings where users are introduced to UML modeling (Pilone & Pitman, 2005).

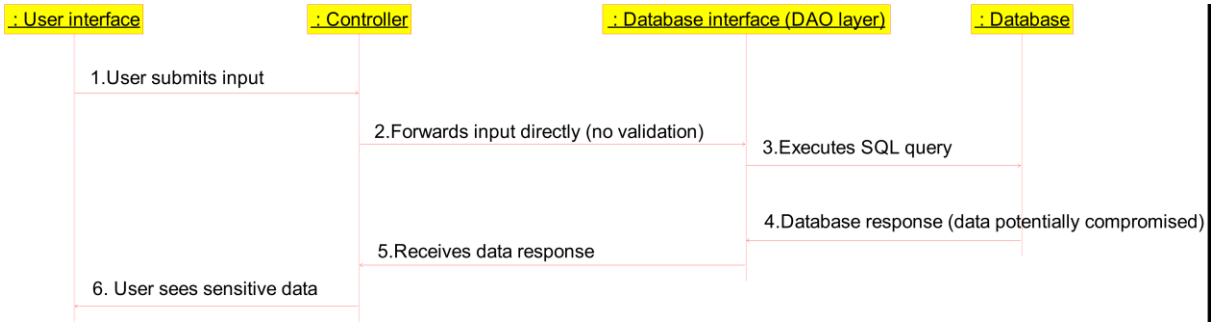
Weaknesses:

1. Limited Advanced Features: Umbrello lacks certain advanced features such as robust code generation and reverse engineering, limiting its effectiveness for complex large-scale projects (UML Modeller, N.D.).
2. Performance and Stability Issues: Reports indicate that Umbrello experiences slowdowns and occasional crashes when handling larger models, affecting the productivity of high-complexity projects (Pilone & Pitman, 2005).

3. Lack of Integration with Development Environments: Umbrello offers limited integration with popular development environments, such as Visual Studio or Eclipse, which can create additional workflow steps in model-intensive projects (Fowler, 2003).
4. Outdated Interface: Umbrello's user interface, while functional, may appear dated to users accustomed to modern UI design, impacting its appeal in professional settings (Roff, 2003).

Sequence Diagram for Injection Vulnerability in Umbrello

To address this task, let us explore one of the well-documented coding weaknesses of the Open Web Application Security Project (OWASP) Top 10, specifically injection vulnerabilities. These vulnerabilities typically arise when an application accepts untrusted data and inadvertently includes them as part of a command or query without adequate validation. This oversight allows attackers to inject malicious SQL and OS commands or scripts, leading to unauthorized data access, database corruption, and severe data breaches (OWASP, 2023). A UML sequence diagram effectively illustrates the interactions between the system components where these vulnerabilities manifest, showing the points where validation errors occur. This makes the sequence diagram an essential tool for visualizing and addressing security gaps that contribute to injection vulnerabilities.



In Umbrello, the sequence diagram for an injection vulnerability features the following components.

1. User Interface: The entry point where a user (or attacker) provides data, such as a login form.
2. Controller: The application component is responsible for processing input, which, in this case, fails to validate user input.
3. Database Access Layer (DAO): The intermediary layer executes SQL queries based on the input provided.
4. Database: Target system executing an SQL query.

Sequence of Events

1. User Input: The user enters potentially malicious data (e.g., DROP TABLE Users; --), bypassing security.
2. Controller Processing: The controller forwards the input to the DAO without validation.
3. DAO Query Execution: The DAO layer executes the query, allowing malicious commands to be executed within the database.
4. Database Compromise: The database executes the query, which results in data leakage or modification.
5. Response: The database returns the results, including potentially sensitive data that the controller then displays to the user.

Using Umbrello to model this sequence of interactions helps developers identify where input validation should be introduced to prevent such vulnerabilities. This diagram, with Umbrello's accessible interface, provides developers and security teams with a clear view of each interaction step, making it easier to identify and mitigate injection risks (Pilone & Pitman, 2005).

Conclusion

While Umbrello's limitations in advanced features and integration may restrict its use in complex projects, its simplicity, cross-platform support, and versatility make it an effective tool for basic-to-intermediate-level modeling. Using it to model security threats, such as injection vulnerabilities, provides a structured approach for identifying security gaps, particularly in the development phase.

References

Fowler, M. (2018) *UML distilled: A brief guide to the standard object modeling language*. Pearson Education (US) : Addison-Wesley Professional.

OWASP (2023) *OWASP Top 10:2021*. Available from: https://owasp.org/Top10/ [Accessed: 24 October 2024].

Pilone, D. & Pitman, N. (2005) *UML 2.0 in a Nutshell: Includes index*. Sebastopol, CA: O'Reilly Media.

Roff, J.T. (2003) *UML: A beginner's guide*. New York: Osborne/McGraw-Hill.

Schardt, J.A. & Chonoles, M.J. (2013) *UML 2 for dummies*. Hoboken, N.J: John Wiley & Sons.

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