**Discussion Topic**

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**](https://www.visual-paradigm.com/)
* [**Sequence Diagram**](https://sequencediagram.org/%20%20)
* [**Umbrello**](https://umbrello.kde.org/)

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)?

**SQL Injection (Initial Post)**

SQL injection attacks are as old as time itself. Well, not really, but they have been around for a really long time. According to malwarebytes.com "The SQL injection exploit was first documented in 1998 by cybersecurity researcher and hacker Jeff Forristal." (What is SQL injection - Examples & prevention | Malwarebytes, 2022). That was over 20 years ago. SQL attacks are one of the most common attacks out there, and they are one of the first security exploits you learn about in computer science. In fact, according to OWASP (What is SQL injection - Examples & prevention | Malwarebytes, 2022), it is number 3 of the ten most critical security risks in a web application. Therefore it is really important to know and understand how an attacker would perform such an attack and how an application developer can prevent that from happening.

**The exploit**

The exploit is typically performed when a web application takes the user input, like the username and password, and concatenates it with an SQL query to allow the user to log in.

Such query could look like this

UserInputUsername = document.getElementById("username").value;

UserInputPassword = document.getElementById("password").value;

SELECT \* FROM Users WHERE username = UserInputUsername AND password = UserInputPassword

This is typically fine if the user actually provides their username and password. However, a malicious attacker could provide an SQL statement that will unknowingly be executed by the web server when it concatenates it with the original SQL query.

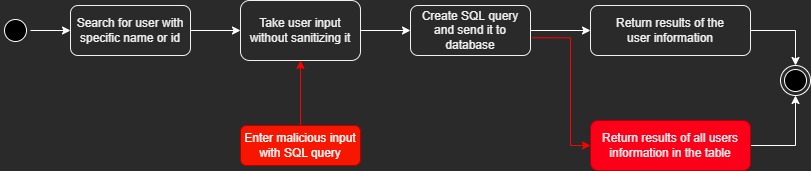
For example, they could provide the following input

UserInputUsername = " or ""="

UserInputPassword = " or ""="

SELECT \* FROM Users WHERE username = UserInputUsername AND password = UserInputPassword

Which would return all rows from the "Users" table, since **OR ""=""** is a true statment.



**The solution**

To prevent this exploit, the developer must run the user input through a sanitization function that will either remove suspicious words like "OR" or "AND" or it would escape a character like single quotes or double quotes.

Most web frameworks typically have some sort of sanitization function like Clean in Django or Prepare in PHP. No one sanitization function does everything for you. You must choose the appropriate tool for the job.

References:

Malwarebytes. 2022. What is SQL injection - Examples & prevention | Malwarebytes. [online] Available at: <https://www.malwarebytes.com/sql-injection#:~:text=The%20SQL%20injection%20exploit%20was,long%20running%20hacker%20zine%20Phrack.> [Accessed 8 October 2022].

Owasp.org. 2022. OWASP Top Ten | OWASP Foundation. [online] Available at: <https://owasp.org/www-project-top-ten/> [Accessed 8 October 2022].