University of Essex | Online

UML Flowchart: Initial Post

Course: MSc Computer Science

Module: Secure Software Development (Computer Science)

Assignment: ePortfolio

Date: Saturday 30th October 2021

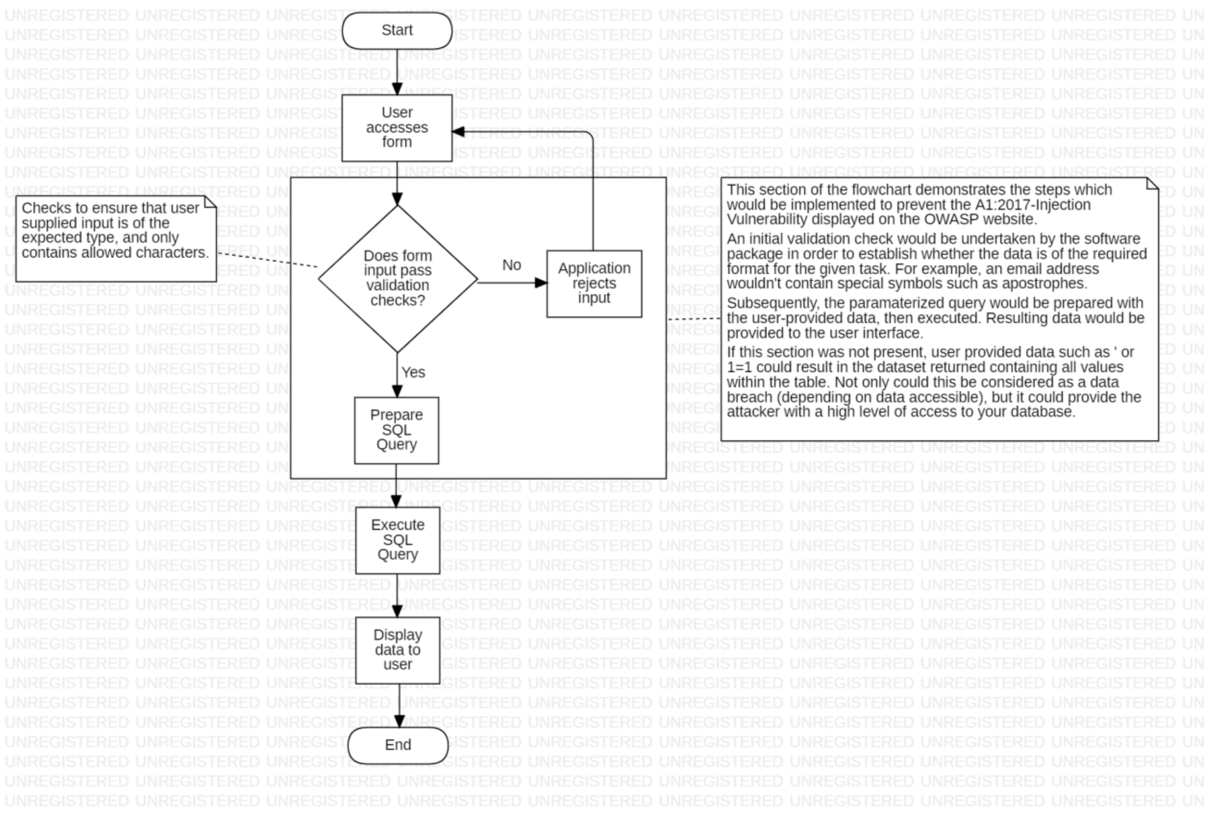
Student ID: 126853

Post:

SQL Injection (A1:2017-Injection) (OWASP, 2017) is a form of attack which can lead to an attacker gaining control over an applications database, potentially leading to the exposure of sensitive consumer/user information (Halfond et al., 2006). The OWASP Foundation (n.d.) considers SQL Injection to be a high severity attack, often only limited by the attacker's skill and imagination. Malicious users often use tools called 'Scanners' which will enumerate through a list of websites (or online search results from sites such as Google) to identify potentially vulnerable sites en-masse. Not only does this reduce the time needed to perform an attack, but it provides fast access to high volumes of data stores.

In most scenarios, SQL Injection attacks can be prevented by using a few techniques during the development process. User Input Sanitisation is the first step that should be completed. The system would ensure that the provided data matches the expected format (For example, phone numbers shouldn't contain any characters from the alphabet). Secondly, the application should use prepared statements and parameterisation to ensure that the data provided cannot modify the SQL query directly (StackOverflow, 2011).

The diagram below shows the stages of a form being completed within a web-based environment. The annotated stages within the rectangle are the steps that would be implemented to prevent/minimise the risk of SQL injection occurring; these steps would likely be missing/incomplete within a vulnerable environment.



Checks to ensure that user supplied input is of the expected type, and only contains allowed characters.

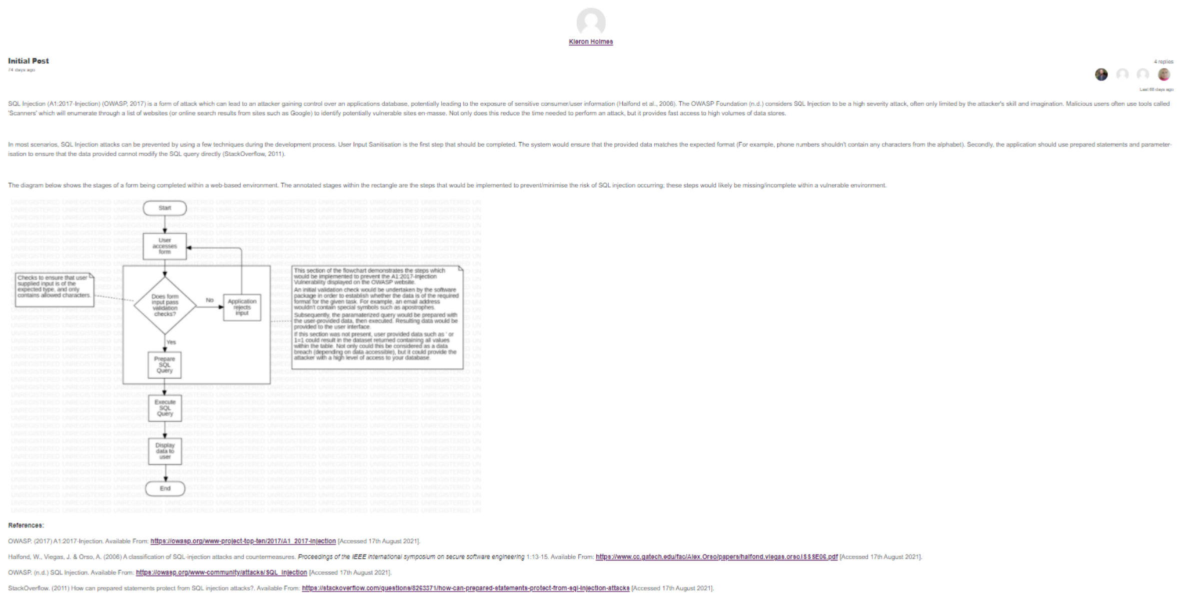
This section of the flowchart demonstrates the steps which would be implemented to prevent the A1:2017-Injection Vulnerability displayed on the OWASP website.

An initial validation check would be undertaken by the software package in order to establish whether the data is of the required format for the given task. For example, an email address wouldn't contain special symbols such as apostrophes.

Subsequently, the parameterized query would be prepared with the user-provided data, then executed. Resulting data would be provided to the user interface.

If this section was not present, user provided data such as ' or 1=1 could result in the dataset returned containing all values within the table. Not only could this be considered as a data breach (depending on data accessible), but it could provide the attacker with a high level of access to your database.

Screenshot:



Kieron Holmes

Initial Post

74 days ago

4 replies

Last 68 days ago

SQL Injection (A1:2017-Injection) (OWASP, 2017) is a form of attack which can lead to an attacker gaining control over an applications database, potentially leading to the exposure of sensitive consumer/user information (Halfond et al., 2006). The OWASP Foundation (n.d.) considers SQL Injection to be a high severity attack, often only limited by the attacker's skill and imagination. Malicious users often use tools called 'Scanners' which will enumerate through a list of websites (or online search results from sites such as Google) to identify potentially vulnerable sites en-masse. Not only does this reduce the time needed to perform an attack, but it provides fast access to high volumes of data stores.

In most scenarios, SQL Injection attacks can be prevented by using a few techniques during the development process. User Input Sanitisation is the first step that should be completed. The system would ensure that the provided data matches the expected format (For example, phone numbers shouldn't contain any characters from the alphabet). Secondly, the application should use prepared statements and parameter-isation to ensure that the data provided cannot modify the SQL query directly (StackOverflow, 2011).

The diagram below shows the stages of a form being completed within a web-based environment. The annotated stages within the rectangle are the steps that would be implemented to prevent/minimise the risk of SQL injection occurring; these steps would likely be missing/incomplete within a vulnerable environment.

References:

OWASP. (2017) A1:2017-Injection. Available From: https://owasp.org/www-project-top-ten/2017/A1_2017-Injection [Accessed 17th August 2021].

Halfond, W., Viegas, J. & Orso, A. (2006) A classification of SQL-injection attacks and countermeasures. *Proceedings of the IEEE international symposium on secure software engineering* 1:13-15. Available From: https://www.cc.gatech.edu/fac/Alex.Orso/papers/halfond.viegas.orso.ISSSE06.pdf [Accessed 17th August 2021].

OWASP. (n.d.) SQL Injection. Available From: https://owasp.org/www-community/attacks/SQL_Injection [Accessed 17th August 2021].

StackOverflow. (2011) How can prepared statements protect from SQL injection attacks?. Available From: https://stackoverflow.com/questions/8263371/how-can-prepared-statements-protect-from-sql-injection-attacks [Accessed 17th August 2021].

References:

Halfond, W., Viegas, J. & Orso, A. (2006) A classification of SQL-injection attacks and countermeasures. Proceedings of the IEEE international symposium on secure software engineering 1:13-15. Available From: https://www.cc.gatech.edu/fac/Alex.Orso/papers/halfond.viegas.orso.ISSSE06.pdf [Accessed 17th August 2021].

OWASP. (2017) A1:2017-Injection. Available From: https://owasp.org/www-project-top-ten/2017/A1_2017-Injection [Accessed 17th August 2021].

OWASP. (n.d.) SQL Injection. Available From: https://owasp.org/www-community/attacks/SQL_Injection [Accessed 17th August 2021].

StackOverflow. (2011) How can prepared statements protect from SQL injection attacks?. Available From: https://stackoverflow.com/questions/8263371/how-can-prepared-statements-protect-from-sql-injection-attacks [Accessed 17th August 2021].