A learning path for Injection Vulnerabilities

Injection ranked as 3rd in the 2021 OWASP top 10 list.
Over 500,000 applications were tested for some form of injection with the average incidence rate of 3.37% with a maximum of 19% and 274k occurrences. [1]

Beginner Level

This attack type is considered a major problem in web security. In fact, injection attacks, such as SQL injection and Cross Site Scripting, are very dangerous and also widespread, especially in legacy applications.
“Injection is an attacker’s attempt to send data to an application in a way that will change the meaning of commands being sent to an interpreter”. [2]
Injection attacks are a very well understood vulnerability class and therefore there are many free available and reliable tools that allow, even inexperienced attackers, to abuse these vulnerabilities automatically, increasing the possibility of this kind of attacks. [3]

In injection attacks the malicious user supplies untrusted input to a program. This is interpreted and processed as a usual data by the program. This may lead to an altered execution of the program, changing the behaviour of the application. This is an important risk to consider because almost any source of data can be an injection vector.

Therefore, the main vehicle for injection attacks is untrusted data. Untrusted data usually comes from HTTP requests in the form of parameters, form fields, headers or cookies. Also, data coming from a database or web services is frequently untrusted from a security perspective. In conclusion, we can define untrusted data as input that can be manipulated to contain a web attack payload. [2]

The primary cause of this kind of vulnerabilities is usually insufficient user input validation.
An application is vulnerable to this category of attacks, for example, when data inserted by users is not correctly filtered or validated by the application, when dynamic queries are used directly in the interpreter.
Therefore, untrusted data should always be treated as it contains an attack. This means you should not send it anywhere without making sure that any possible attack is detected and neutralized.

A successful injection attack can easily result in significant data breaches, or even loss of control over the entire application.

Examples of CWEs falling in this family are Cross Site Scripting (XSS) and SQL injection.

Cross Site Scripting attacks are a type of injection, in which malicious script are injected into otherwise trusted websites.
XSS is a very common security vulnerability and hard to eliminate, even for organizations with good security programs. In fact, a report by HackerOne says that XSS flaws accounted for 18% of all reported cybersecurity issues in 2020. [4]
Interestingly, XSS vulnerabilities are hitting the biggest businesses around the world harder, including top sites like Facebook and Steam. In 2020, one researcher named Vinoth Kumar found an XSS vulnerability affecting the ‘Login with Facebook’ SDK commonly used by other sites to authenticate users. [5]

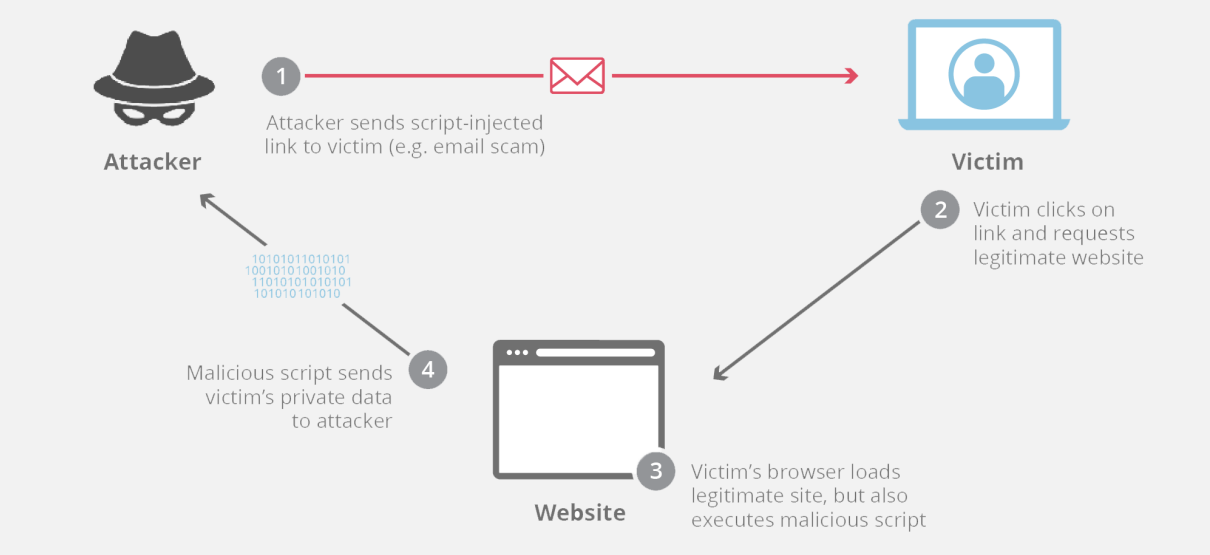

Fig. 3.1 XSS Schema [6]

XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different user.
The end user's browser has no way to know that the script should not be trusted and will execute the script. As he thinks the script comes from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site.

Cross Site Scripting attacks occur when:

- Data enters a web application through an untrusted source, most frequently a web request.
- The data is included in dynamic content that is sent to a web browser without being validated for malicious content. [7]

SQL injection is a code injection technique used to attack data-driven applications.
A SQL injection attack consists of insertion or "injection" of a malicious SQL query via the input data from the client to the application. [8]
SQL injection exploit a security vulnerability in an application software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed.

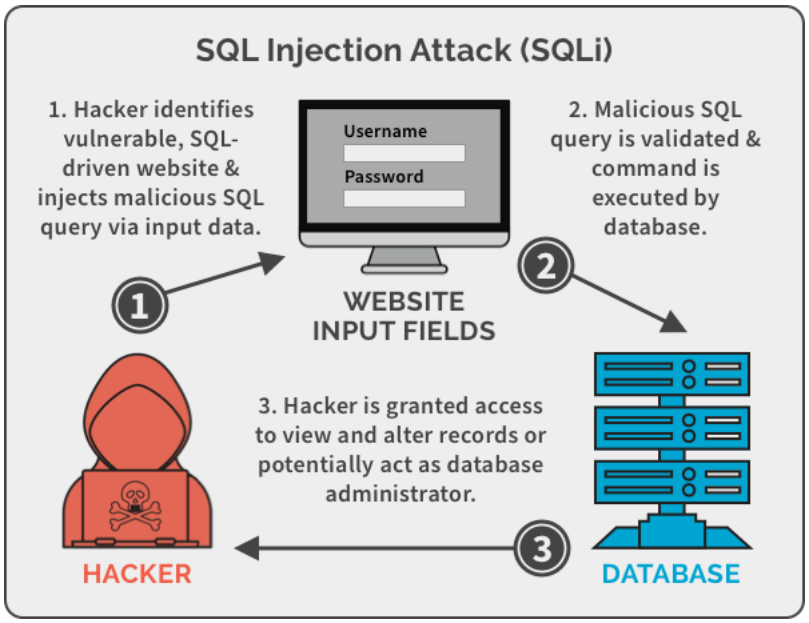

Fig. 3.2 SQLi Schema [9]

A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content files present on the DBMS file system and in some cases issue commands to the operating system.
In general, SQL Injection can be considered a high impact severity.
An SQL injection attack occurs when unintended data enters a program from an untrusted source and that data is used to dynamically construct a SQL query. [8]
The attack is essentially performed by placing a meta character into data input to then place SQL commands to control the application output, which could not be possible otherwise. This flaw depends on the fact that SQL makes no real distinction between the control and data planes.

The flaw is easily detected, and easily exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind.
The majority of SQL injection vulnerabilities can be found quickly and reliably using some kind of web vulnerability scanner or manually, by using a systematic set of tests against every entry point in the application.
These tests typically involve:

- Submitting the single quote character ' and looking for errors or other anomalies.
- Submitting some SQL-specific syntax that evaluates to the base (original) value of the entry point, and to a different value, and looking for systematic differences in the resulting application responses.
- Submitting Boolean conditions such as OR 1=1 and OR 1=2 and looking for differences in the application responses.
- Submitting payloads designed to trigger time delays when executed within a SQL query, and looking for differences in the time taken to respond.
- Submitting OAST (Out-of-band Application Security Testing) payloads designed to trigger an out-of-band network interaction when executed within a SQL query, and monitoring for any resulting interactions. [10]

Bibliography

[1] «A03:2021 - Injection,» [Online]. Available: https://owasp.org/Top10/A03_2021-Injection. [Consulted 2023].

[2] «Injection Theory,» [Online]. Available: https://owasp.org/www-community/Injection_Theory. [Consulted 2023].

[3] «What are injection attacks,» [Online]. Available: https://www.acunetix.com/blog/articles/injection-attacks/. [Consulted 2023].

[4] «Top ten vulnerabilities,» [Online]. Available: https://www.hackerone.com/top-ten-vulnerabilities. [Consulted 2023].

[5] «xxs vulnerability in login with Facebook,» [Online]. Available: https://portswigger.net/daily-swig/xss-vulnerability-in-login-with-facebook-button-earns-20-000-bug-bounty. [Consulted 2023].

[6] [Online]. Available: https://www.cloudflare.com/img/learning/security/threats/cross-site-scripting/xss-attack.png.

[7] «Cross Site Scripting,» [Online]. Available: https://owasp.org/www-community/attacks/xss/. [Consulted 2023].

[8] «SQL Injection,» [Online]. Available: https://owasp.org/www-community/attacks/SQL_Injection. [Consulted 2023].

[9] [Online]. Available: https://spanning.com/wp-content/uploads/2019/07/SQL-injection-attack-example.png.

[10] «What is SQL injection,» [Online]. Available: https://portswigger.net/web-security/sql-injection. [Consulted 2023].