Prevent injection attacks

Previously, you learned that **Structured Query Language** (SQL) is a programming language used to create, interact with, and request information from a database. SQL is one of the most common programming languages used to interact with databases because it is widely supported by a range of database products.

As you might recall, malicious **SQL injection** is a type of attack that executes unexpected queries on a database. Threat actors perform SQL injections to modify, delete, or steal information from databases. A SQL injection is a common attack vector that is used to gain unauthorized access to web applications. Due to the language's popularity with developers, SQL injections are regularly listed in the OWASP® Top 10 because developers tend to focus on making their applications work correctly rather than protecting their products from injection.

In this reading, you'll learn about SQL queries and how they are used to request information from a database. You will also learn about the three classes of SQL injection attacks used to manipulate vulnerable queries. You will also learn ways to identify when websites are vulnerable and ways to address those gaps.

SQL queries

Every bit of information that's accessed online is stored in a database. A **database** is an organized collection of information or data in one place. A database can include data such as an organization's employee directory or customer payment methods. In SQL, database information is organized in tables. SQL is commonly used for retrieving, inserting, updating, or deleting information in tables using queries.

A *SQL query* is a request for data from a database. For example, a SQL query can request data from an organization's employee directory such as employee IDs, names, and job titles. A human resources application can accept an input that queries a SQL table to filter the data and locate a specific person. SQL injections can occur anywhere within a vulnerable application that can accept a SQL query.

Queries are usually initiated in places where users can input information into an application or a website via an input field. Input fields include features that accept text input such as login forms, search bars, or comment submission boxes. A SQL injection occurs when an attacker exploits input fields that aren't programmed to filter out unwanted text. SQL injections can be used to manipulate databases, steal sensitive data, or even take control of vulnerable applications.

SQL injection categories

There are three main categories of SQL injection:

- In-band
- Out-of-band
- Inferential

In the following sections, you'll learn that each type describes how a SQL injection is initiated and how it returns the results of the attack.

In-band SQL injection

In-band, or classic, SQL injection is the most common type. An in-band injection is one that uses the *same communication channel* to launch the attack and gather the results.

For example, this might occur in the search box of a retailer's website that lets customers find products to buy. If the search box is vulnerable to injection, an attacker could enter a malicious query that would be executed in the database, causing it to return sensitive information like user passwords. The data that's returned is displayed back in the search box where the attack was initiated.

Out-of-band SQL injection

An out-of-band injection is one that uses a *different communication channel* to launch the attack and gather the results.

For example, an attacker could use a malicious query to create a connection between a vulnerable website and a database they control. This separate channel would allow them to bypass any security controls that are in place on the website's server, allowing them to steal sensitive data

Note: Out-of-band injection attacks are very uncommon because they'll only work when certain features are enabled on the target server.

Inferential SQL injection

Inferential SQL injection occurs when an attacker is unable to directly see the results of their attack. Instead, they can interpret the results by analyzing the *behavior* of the system.

For example, an attacker might perform a SQL injection attack on the login form of a website that causes the system to respond with an error message. Although sensitive data is not returned, the attacker can figure out the database's structure based on the error. They can then use this information to craft attacks that will give them access to sensitive data or to take control of the system.

Injection Prevention

SQL queries are often programmed with the assumption that users will only input relevant information. For example, a login form that expects users to input their email address assumes the input will be formatted a certain way, such as *jdoe@domain.com*. Unfortunately, this isn't always the case.

A key to preventing SQL injection attacks is to *escape user inputs*—preventing someone from inserting any code that a program isn't expecting.

There are several ways to escape user inputs:

- **Prepared statements**: a coding technique that executes SQL statements before passing them on to a database
- **Input sanitization**: programming that removes user input which could be interpreted as code.
- **Input validation**: programming that ensures user input meets a system's expectations.

Using a combination of these techniques can help prevent SQL injection attacks. In the security field, you might need to work closely with application developers to address vulnerabilities that can lead to SQL injections. OWASP's SQL injection detection techniques is a useful resource if you're interested in investigating SQL injection vulnerabilities on your own.

Key takeaways

Many web applications retrieve data from databases using SQL, and injection attacks are quite common due to the popularity of the language. As is the case with other kinds of injection attacks, SQL injections are a result of unexpected user input. It's important to collaborate with app developers to help prevent these kinds of attacks by sharing your understanding of SQL injection techniques and the defenses that should be put in place.