People's Democratic Republic of Algeria

Ministery of Higher Education and Scientific Research

Ferhat Abbas University of Setif 1



Faculty of Sciences

Computer Science Department

**THEME**

Detecting SQL injection using Deep Learning

Realizedby: Supervised by:

ZITOUNI AHMED FAOUZI DR. BENZINE MEHDI

SEDJAL MOHAMED AYMEN DHIAEDDINE

**2024/2025**

**Table of contenets**

General introduction

Chapter 1 SQL injection

1.1 Introduction

1.2 Web application

1.2.1 Definition…………………………………………………………………………………………………………

1.3 SQL injection………………………………………………………………………………………………………….

1.3.1 Definition6

1.3.2 How SQL Injection Works……………………………………………………….7

1.3.3 Impact of SQL Injection Attacks…………………………………………………

1.4 Techniques of SQL Injection…………………………………………………………….

1.4.1 Tautology-Based SQL Injection…………………………………………………….

1.4.2 Error-Based SQL Injection………………………………………………………….

#### 1.4.3 Blind SQL Injection…………………………………………………………………

**Chapter 1**

SQL Injections

**1.1 Introduction**

Withincreasingly digital living, web applications are at the core of daytoday lifefrommanaging finances and online purchasing to collaborating and communicating. This ease of the virtual world comeswithinherentsecurity challenges. Cyber attackerspersistentlyevolvetheirmethods to exploit weaknesses, therebyendangeringunauthorized data access, downtime of services, and irreparable damage to reputation.

        Instead of goingwith the traditionalenumeration, itis pertinent to talk about these five new categories of risksthatcontemporary web applications have to face:

**Inadequate Input Validation:**

Not validating and sanitizing user-supplied data properly can facilitate a range of attacks, such as injection and cross-site scripting (XSS). Suchneglectpermitsattackers to injectmalicious code into the system, therebyviolating data integrity and application functionality.

**InsufficientAuthentication and Session Management:**

Insufficientmethods of validating user identities or handling active sessions can permit unauthorizedusers to obtainaccess. Adequatecontrolssuch as multi-factor authentication and secure session managementare essential for protecting user accounts and sensitive information.

**Lack of data protection:**

Irrespective of whether the data is at rest or in motion, poorencryption or weak key management practices can lead to exposure of sensitive data. Robustcryptographic standards have to beimplemented, and sensitive data has to bemaintainedconfidentiallyfromits life cycle for successful protection of data.

**DefectiveIntegration&DependencyManagement:**

Contemporary web applications tend to includeexternal services and libraries. Whentheseunderlyingelements are insecure or outdated, they can injectvulnerabilitiesintothe system overall. Diligent monitoring ofdependencies, alongwith routine security audits, isrequired to avertthesethreats.

**Poor error handling and logging:**

Poorlycontrollederror messages and a lack of logging can inadvertentlydisclose system information to prospective attackers or hamper the timelydetection of a security incident. Havingwelldefinederrorhandling routines and utilizingsolid monitoring systems are essential for the timelydetection and containment of maliciousactivity.

One of the mostrisky implications of thesevulnerabilitiesisexposure to SQL injection attacks. Where an application does not sanitize input sufficiently, malicioususers can injectmalicious SQL statements, and these have the potential to result in unauthorized data alteration or even system hijacking. The implication of this type of exploitation can becatastrophicresulting in data loss, system crash, and long-termreputational damage for an organization.

        Through the use of proactive design principles and audits on a regular basis, developers can greatlyminimize the chances of exploitation in today'sintricate digital world.

**1.2 Web applications**

**1.2.1 Definition**

Web applications are software programs thatoperatewithin a web browser, allowing the user to get a set of interactive functionalitiessuch as login systems for user authentication, and real-time chat or messaging features for immediate communication. By doingso, the user is capable of enjoyingadvanced and interactivefunctionalitieswithout the installation of other software. Conversely, the entire code thatisrequiredis run in the browser of the web, whereas application logic and data are handled on the server side via the implementation of a Database Management System (DBMS), web applications that use DBMS are known as Database-driven Web applications.

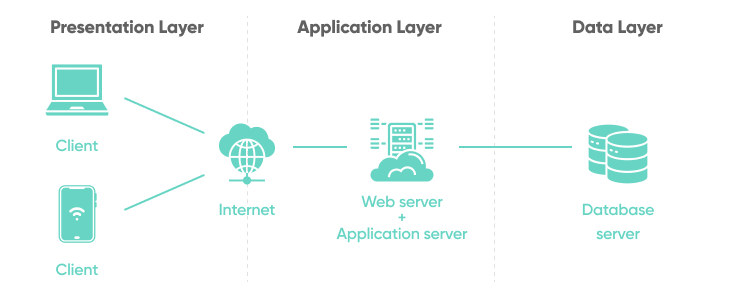
Database-driven Web application areusuallycomposed of a back end database and web pagesthatinclude server-side scripts thatretriveor update data from the database, dependingon user actionslike submitting a form, logging in, or searching for products.

Typically, the application relies upon a three-tier structure:

**PresentationLayer:** User interface tier, typicallyconsisting of a web browser or rendering engine using HTML, CSS, and JavaScript.

**LogicLayer:**It handlesretrinving andprocessing the datawith the help of server-sidescripts.

**Data Layer:** There the data resides, managed by databasesincluding Microsoft SQL Server, MySQL, Oracle, and many more.



**1.3 SQL injection**

**1.3.1 Definition**

A SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modifydatabase data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injectedinto data-plane input in order to affect the execution of predefined SQL commands[[2].](#_[2]_OWASP_“SQL)

**1.3.2 How SQL Injection Works**

SQL injection is a type of cyberattack which exploits vulnerabilities within the input validation process of an internet application. It occurs when a malicious SQL code is inserted or placed through user inputs, which are later executed by a database. The reason being the application fails to validate, sanitize, or parameterize user input appropriately prior to their inclusion within SQL queries.

**Mechanism of SQL Injection**

**1 User Input Handling :**

Web applications are prone to accept user inputs (e.g., from login forms, search fields, or URL parameters) and use them to construct SQL queries dynamically. For example:



**2 Malicious Input Injection :**If the application does not sanitize or validate the input, an attacker can inject a malicious SQL code. For example, entering ' OR '1'='1 as the username and an empty password transforms the query into:



The condition **'1'='1'** is always true, causing the query to return all rows in the users table. This allows the attacker to bypass authentication.

**3 Database Execution** :  
The database executes the malicious query, leading to unauthorized access, data leakage, or other malicious activity.

**1.3.3 Impact of SQL Injection Attacks**

SQL injection (SQLi) is an attack that takes advantage of vulnerabilities in the application's software by injecting malicious SQL statements into form fields. The consequences of SQL injection are severe and include:

**1. Data Theft**: Sensitive information from the database, such as personal data, credentials, and financial information, can be stolen by attackers.

**2. Modification of Data**: SQL injection can enable attackers to modify or delete data, resulting in integrity issues and loss of critical information.

**3. Bypassing Authentication**: SQL query attacks can enable unauthorized access to users' accounts or administrative rights.

**4. Denial of Service (DoS):** Attackers can generate queries that overwhelm the database, which will subsequently fail or become unavailable.

**5. Remote Code Execution**: In some cases, SQL injection is employed to run unauthorized code on the server, taking full control of the system by attackers.

#### **6. Reputation Damage**: Organizations that fall victim to SQL injection attacks risk losing business reputation, customers' trust, and possible legal repercussions, most significantly if it involves sensitive information.

#### **7. Loss of Funds**: The expense of data breaches, including recovery, compliance penalties, and lost business, can be significant.

#### Briefly, SQL injection is a security threat that demonstrates the value of good security controls, such as input validation, parameterized queries, and security auditing.

#### **1.4 Techniques of SQL Injection**

**1.4.1 Tautology-Based SQL Injection**

This method exploits the use of conditional SQL statements that always return true, effectively bypassing authentication and other access controls.

A tautology is a logical statement that is always true, regardless of its components. Attackers inject such statements into SQL queries, which forces the database to process and accept unauthorized requests.



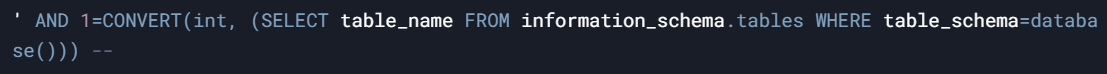
* Here, OR '1'='1' ensures the condition is always true, granting access.
* **Real-World Scenario:** An attacker targets a login form on a website, bypassing user authentication and accessing an administrator account.
* **Impact:**
  + Unauthorized access to accounts.
  + Potential data breaches and manipulation.
  + Loss of user trust and legal consequences.
* **Countermeasures:**
  + Employ parameterized queries.
  + Sanitize user inputs rigorously

.

**1.4.2 Error-Based SQL Injection**

This technique relies on triggering and extracting database error messages to obtain information about the database structure.

By deliberately generating SQL errors, attackers can gather information such as table structures, column types, and even data.



* **Real-World Scenario:** A hacker attacks a poorly configured e-commerce site, revealing table names containing customer payment details.
* **Impact:**
  + Exposure of database schema.
  + Facilitation of more precise attacks.
  + Leakage of sensitive data.
* **Countermeasures:**
  + Disable detailed error reporting in production.
  + Implement comprehensive error handling.

#### **1.4.3 Blind SQL Injection**

This technique is used when the application does not display database errors, forcing attackers to infer database behavior indirectly.

##### **1.4.3.1 Content-Based Blind SQL Injection**

* **Detailed Explanation:** Attackers use conditional statements to observe subtle changes in the application's content or behavior.



* **Impact:**
  + Gradual extraction of data through logical inference.
  + Highly challenging to detect.
* **Countermeasures:**
  + Use of web application firewalls (WAFs).
  + Regular code reviews.
* ssss

**References**

[1]

# [2] OWASP “SQL Injection (SQLI)”: [https://owasp.org/search/?searchString=sql+injection+](%20https:/owasp.org/search/?searchString=sql+injection+)