

Contents

Part I Design Network Intrusion Detection System using ML methods

1	SQL Injection attacks on Web Application Database	5
	MANASA.M (212IS015)	
1.1	Introduction	5
1.1.1	Problem Description	6
1.1.2	Motivation	6
1.1.3	Scope	7
1.1.4	Objectives	7
1.1.5	Organization of the Report	7
	Related work	8
	The Proposed Approach	10
	Experimental Results and Analysis	12
4.1	Code Injection	12
4.2	SQL Manipulation	13
4.3	Like Queries	14
4.4	Insert subselect Queries:	15
4.5	Tautology:	16
	Conclusion	17
	References	18
	References	18

List of Contributors

MANASA M

National Institute of Technology Karnataka

Surathkal, Mangaluru, India, e-mail: manasa,212is015@nitk.edu.in

Dr MAHENDRA PRATAP SINGH

National Institute of Technology Karnataka

Surathkal, Mangaluru, India , e-mail: mp_singh@nitk.edu.in

Acronyms

- SQL** Structured Query Language is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDBMS)
- ASP.NET** ASP stands for Active Server Pages. ASP is a development framework for building web pages. ASP supports many different development models: Classic ASP. ASP.NET Web Forms.
- SQLIA** Structured Query Language Injection Attack, is a type of cyber attack in which a hacker uses a piece of SQL (Structured Query Language) code to manipulate a database and gain access to potentially valuable information.
- URL** URL stands for Uniform Resource Locator. A URL is nothing more than the address of a given unique resource on the Web. In theory, each valid URL points to a unique resource. Such resources can be an HTML page, a CSS document, an image, etc.

Part I
Design Network Intrusion Detection
System using ML methods

Chapter 1

SQL Injection attacks on Web Application Database

MANASA.M (212IS015)

Abstract The ability to use the internet at any time is extremely beneficial to the technical area. As the internet becomes more widely used, so are web-based attacks. The expanding number of users for services like social networks, news, online retail, and financial services makes them an enticing source of sensitive data for hackers. The majority of these flaws are caused by a lack of input validation, which occurs when web applications employ malicious input as part of a sensitive process without adequately validating or sanitising the information. The most serious vulnerability is SQLIA, which attacks the database of a web application. The malicious SQL command is bound to the original web application query and made to run on the server by malicious users. In this paper we discuss about the various methods of SQL Injection Attacks and implements them in a php based web application to clearly understand how vulnerability of the system database is compromised for attacker benefit. To prevent SQL injection attacks, the most important security mechanism for online application databases is also discussed in paper.

Key words: SQL Injection, SQLIA, Web applications Sensitive operation, Sanitization

1.1 Introduction

The popularity of database-driven online applications has accelerated due to rapid advancements in web technologies. With the rapid advancement of digital technology, the majority of individuals are conducting business online. The main goal of a SQL injection attack is to get around authentication and get access to secret and sensitive

Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangaluru, India

data, to alter database data, execute malicious code remotely, and gain control of the server. A number of recent surveys, the Open Web Application Project [7] indicate that SQLIA is among top three worst vulnerabilities discovered in today's web-based applications after their deployment. The SQL injection can be carried out by two ways:

1. Through users input
2. Through URL

Many people are unaware of the importance of data protection and privacy. By supplying specially designed input sources, attackers deceive the target operations (SQL sink or SQL query structure) into executing undesired commands (inputs either provided by the user or another program). An attacker can access, edit, or even delete database information using SQL injection vulnerabilities. This information is frequently confidential or sensitive. Malicious users add the malicious SQL command to the original web application query and force it to run on the server. SQL injection attacks may be conducted with a basic understanding of SQL commands and some clever conjecture work to vital table names. Before proceeding, it is necessary to review some basic definitions in order to have a better understanding of SQL injection attacks on web applications and their underlying databases [8]:

1. Vulnerabilities are inconsistencies, gaps, defects, or faults/imperfection in an existing system.
2. An attack is an unauthorised access, or a way of exploiting vulnerabilities.
3. Threat: a set of events that use the system in an illegal way, jeopardising the principles of information security, such as the system's confidentiality, integrity, and availability.
4. Risk: Impact of the threat

The goal of this work is to present a taxonomy of SQL Injection Attacks against a web application's database. The malicious SQL query is evaluated on database MYSQL to detect how hackers can easily get database confidential content through illegal access [9].

1.1.1 Problem Description

Deployment of web application to clearly understand the different types of SQL Injection Attack and discuss various counter measures to prevent SQL Injection attack on web application

1.1.2 Motivation

This project clearly helps us to understand how attacker can exploit the vulnerability of database in the system to gain illegal access of the corporate database..

1.1.3 Scope

This work is based on database security web application. It helps testers to or take certain measures before releasing a web application in the internet.

1.1.4 Objectives

The objectives include:

1. To discuss various SQL injection types
2. To design a 2 tier Web Application to understand the various SQLIA attacks
3. To discuss various counter measures to prevent Injection attack on SQL systems.

1.1.5 Organization of the Report

The remainder of this paper is organized as follows: Section II focuses on Related papers in past years Section III discusses SQL injection attacks preview and as well briefly introduce about the implementation of web application to show how SQLIA attack are done. Section IV describes observations and analysis on the result obtained on entered queries. Lastly, Section VII presents conclusions.

Chapter 2

Related work

This section lists some of the related research works that have been completed over the years.

William G.J. Halfond and Alessandro Orso [1] proposed a model-based approach, it detects unauthorised queries before they are run on the database. In its static component, the approach employs programme analysis to automatically construct a model of the application's authorized queries. The technique's dynamic component employs runtime monitoring to evaluate dynamically created queries and compare them to the statically built model. They created AMNESIA, a tool that implements their technique, and they used it to test it on seven different online web application. According to the results of the investigation, their technique was able to thwart all attempted attacks while producing no false positives.

Mazoon Al Rubaiei [2] proposed a foundation on the many varieties of SQLIA, as well as detection and prevention measures, as well as an assessment of these tactics against those attacks. There are various approaches for preventing and detecting SQLIA, but Java Static Tainting is the most powerful, and Store procedures are present as related work. It then looks into a variety of detection and prevention methods. A comparison of detection and prevention strategies is discussed in the evaluation section in terms of their capacities to detect, prevent, or partially stop the attack.

Verifying the line comments inside the query, testing concatenation, checking for the obfuscation string, checking for the encrypted string, ASCII encoding, and hexadecimal encoding were among the detection techniques employed in the SQL Injection algorithm by Bandi Aruna and Bhimavarapu Usharani [3]. They also determined the execution time using the SQL Injection detection algorithm, which is about $O(\log N)$, where N is the number of characters in each line of the dataset. The maximum time required by the procedure is proportional to the logarithm of the input size. The proposed algorithm runs pretty faster.

The proposed approach by Sarjiyus, O. and El-Yakub, M. B. [5] was shown to be effective in terms of preventing all sorts of SQL injection attacks. The vulnerability of the code was tested using Acunetix, and the code was implemented on a small website with a simple database. Unlike most approaches, the proposed solution is

both simple and effective to execute. The results show promise in terms of detecting SQL injection attacks, with a high accuracy rate.

Mohammad Abu Kausar, Mohammad Nasar and Aiman Moyaid [4] introduced a different type of process to protect against the present SQLIA approach and instruments used in ASP.NET apps to identify and halt these assaults. They tried to address the SQL Injection Attack, ASP, Net web applications frequent attack. They also looked at strategies for preventing and detecting these attacks that we could learn about and use to avoid them.

Young-SuJangJin-YoungChoi [6] present a novel method for automatically transforming web applications so that they are safe from SQL injection attacks. Their method dynamically evaluates the query result size intended by the developer for any input and detects threats by comparing it to the actual query result. Their technique is to prevent web application designed using java. Their work is effective against SQL injection vulnerabilities, according to an experimental study.

Chapter 3

The Proposed Approach

Proposed approach For the better understanding of SQLIA proposed system is design of a web application . Web applications are a set of web pages and programs which reside on a web server [17]. The inputs provided by the user are sent to the server in the form of parameter string. These inputs are used to engender SQL query to retrieve information from the database. An authorized user can access it over the cyber world or over a public network and store the data in the database. A web application utilizes a web browser as an interface to extract the data from database server to accommodate the queries placed by the users [11]. Every web application is predicated on 3 –tier architecture consisting of three layers [6] .Each layer can run potentially on a different machine and each layer should be independent of other layers. The three layers are Presentation Layer: Presentation layer contains presentation logic. It is the top most level of application and handles the interactions with users. Its main function is to receive input from the user and provide the result in a convenient way that user can facilely understand.

Presentation Layer: The presentation layer is where the presentation logic is stored. It is the highest level of the application and is in charge of user interactions. Its primary job is to take user input and present the outcome in an easy-to-understand format.

Business Layer:Between the presentation layer and the database layer is the business layer. It's a logic layer that consists of a set of rules for transferring data between two layers. It includes application process commands for retrieving data from the database and sending it to the presentation layer for display. Any server scripting language, such as JSP, PHP, and ASP, can be used to create this layer.

Database Layer:This is a data persistence physical storage layer. It controls all database and file system access. The database is used to store and retrieve data. It's then transmitted back to the presentation layer to be processed before being returned to the user. The major purpose of this layer is to grant access to authorised users while restricting access to malevolent users.

The presentation layer gets the request from the web browser, processes it, and then transmits the dynamic portion to the business layer, which handles server scripting

languages. The database server receives all database access requests. The output is subsequently sent to a web browser in the form of web pages.

The focus on implementing the functionality rather than security from internal and external environment causes the susceptibilities in web applications. These are described as most solemn threat for web application as it may allow attacker to gain access to the web application and its underlying database. The potential of attacker perforates the system to extract sensitive information. Exploitation of loopholes in the design breaches the fundamental principles of information security i.e. confidentiality, Integrity, authenticity and availability of information. Information stolen is loss of confidentiality. Loss of integrity takes place when data is modified in an unexpected manner. The inputs provided by the user form the dynamic SQL query to access the backend database. If these inputs are not opportunely sanitized, they can cause the web application to generate unintended outputs. The basic underlying fact is that SQL injection attacks are very easy to execute without any professional training. This web application takes original query and manipulated query as input and demonstrate how that produces the same result and how attacker can inject a query in web application. The paper also displays the query output to demonstrate different types of SQL injection.

Chapter 4

Experimental Results and Analysis

The goal of a SQL Injection Attack (SQLIA) is to break into a database system and run malicious code that can divulge sensitive information. To acquire illegal access, the SQL queries and expressions are injected as an input string. SQL injection is a danger that can lead to a high level of compromise, giving the attacker access to any database query. It's a type of web-based application-level assault that connects to a backend database and gets over the firewall. The attacker takes advantage of vulnerable code and shoddy input validation to perform unauthorised SQL requests. SQL Injection attacks against web application databases can be divided in four sections as follows:

1. Code Injection
2. SQL Manipulation
3. Function Call Injection
4. Buffer Overflows

In this section the above briefed attacks are discussed in detailed with their code injecting mechanism, types and subtypes with appropriate SQL statements and queries.

4.1 Code Injection

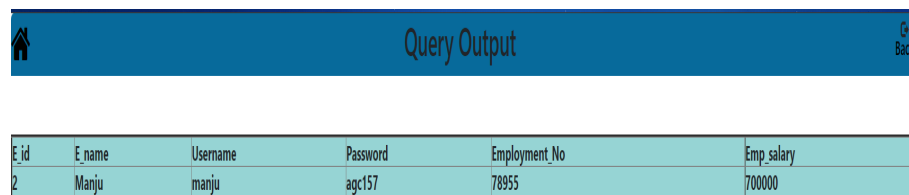
In code injection, attacker attempt to add additional SQL statement or commands to the existing SQL statements.

Original Query:

```
SELECT * FROM employee WHERE  
E_name= 'Manju' and E_id='2';
```

Output

A salary statement of desired employee is extracted from the database.



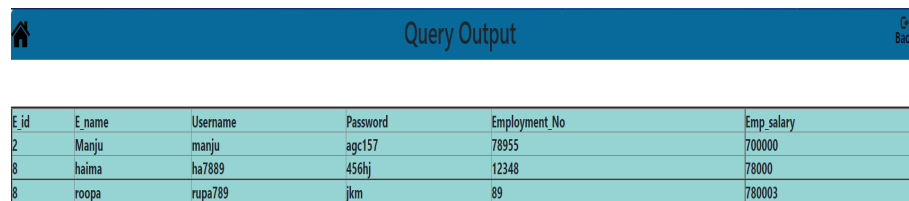
The image shows a web application interface titled "Query Output". It features a home icon on the left and a "Back" button on the right. Below the header is a table with the following data:

E_id	E_name	Username	Password	Employment_No	Emp_salary
2	Manju	manju	agc157	78955	700000

Fig. 4.1: Output:Details of desired employee is extracted from the database.

Manipulated Query:

```
SELECT * FROM employee WHERE E_name= 'Manju' and E_id='2' or
'2'>'1';
```



The image shows a web application interface titled "Query Output". It features a home icon on the left and a "Back" button on the right. Below the header is a table with the following data:

E_id	E_name	Username	Password	Employment_No	Emp_salary
2	Manju	manju	agc157	78955	700000
8	haima	ha7889	456hj	12348	78000
8	roopa	rupa789	jkm	89	780003

Fig. 4.2: Output:Information about all records in employee table is listed .

Output

Information about all records in employee table is listed .

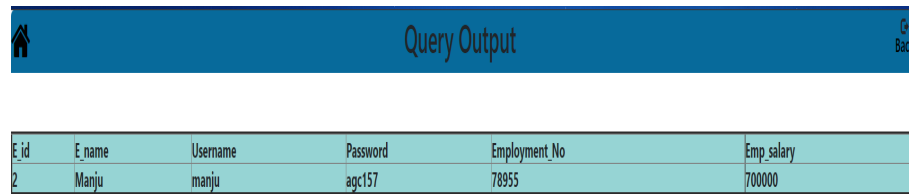
4.2 SQL Manipulation

A very familiar kind of SQL Injection attack is SQL manipulation. The most significant examples of this type is by adding elements to the WHERE clause with set operators like UNION, INTERSECT etc. Or comparators like OR, greater than, smaller than and many more. The simplest example is login authentication that a web application may check.

Original Query:

```
SELECT * FROM employee WHERE
username='roopa' and Password='agc157';
```

Output:

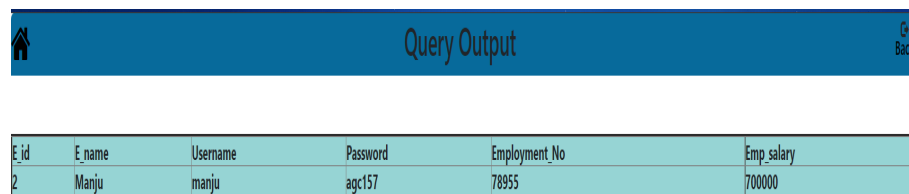


E_id	E_name	Username	Password	Employment_No	Emp_salary
2	Manju	manju	agc157	78955	700000

Fig. 4.3: Output:Details of desired employee is extracted from the database.

Manipulated Query:

```
SELECT * FROM employee WHERE
username= 'Manju' OR 'x'='x' and Password='-----';
```



E_id	E_name	Username	Password	Employment_No	Emp_salary
2	Manju	manju	agc157	78955	700000

Fig. 4.4: Manipulated Query output after sql Manipulation

Based on operator precedence, the clause WHERE is true for every entry and (——) regarded as comment consequently ignored by server granting access to attacker.

4.3 Like Queries

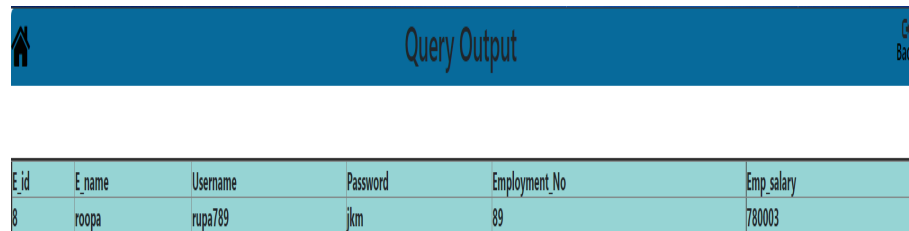
: Like query is used to compare a value to similar values using wildcard operators. In reality two wildcards are used

- i) % represents zero, one or multiple characters
- ii) underscore (_) represents a single number or character. The attacker manipulates the SQL statement using these wildcards. A Denial of Service attack can be launched with a few changes in a LIKE query by overloading the database

Original Query:

```
SELECT * FROM Employee WHERE
E_Name = 'roopa';
```

Output:

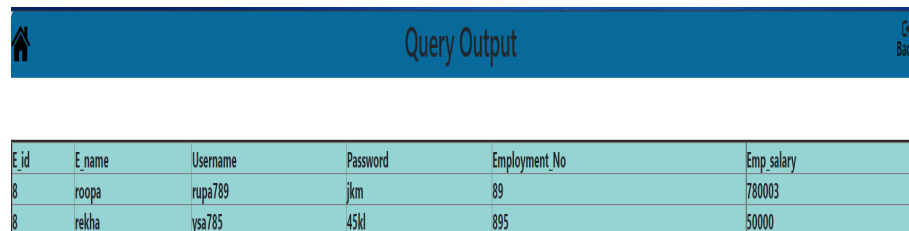


E_id	E_name	Username	Password	Employment_No	Emp_salary
8	roopa	rupa789	ikm	89	780003

Fig. 4.5: Original Query:Output

Manipulated Query:

```
SELECT * FROM Employee WHERE
E_Name like 'r%';
```



E_id	E_name	Username	Password	Employment_No	Emp_salary
8	roopa	rupa789	ikm	89	780003
8	rekha	ysa785	45kl	895	50000

Fig. 4.6: Manipulated Query output after sql Manipulation

It provides the list of all employees starting with name A or having name an alphabet r

4.4 Insert subselect Queries:

The insertion of a sub query into a query can also help the attacker to access all the records of the database

```
SELECT * FROM employee WHERE Employment_No =
(SELECT id FROM users WHERE user_name='Mamatha');
```

4.5 Tautology:

In this type of SQLIA, an attacker exploits an injectable field that is used in a query. The query always returns result upon evaluation of a WHERE conditional parameter. The aim of this attack is to inject malicious codes into one or more conditional statements which are always evaluated to be true. All the rows in the database table targeted by the injected query return the conditional WHERE into a tautology.

```
SELECT *FROM employee WHERE  
E_name='Manju' OR 'A'='A' AND Password='-----';
```


Chapter 5

Conclusion

The full examination of several types of SQL injection attacks, as well as related vulnerabilities and injection methodologies, is described in this work. A typical approach is SQL Injection. On web-based data-centric apps, attackers use a variety of techniques. These attacks manipulate SQL queries in order to change the results. application's behaviour This paper also includes the following: a taxonomy of mechanisms for avoiding, preventing, and mitigating harm These assaults can be detected. A Threat model will be used in future projects. As a database security mechanism, it will be proposed. web-based applications

References

1. Halfond, William G. J. and Orso, Alessandro: *AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks*. Association for Computing Machinery, vol 29, no 2, pp. 174–183, 2005.
2. M. Al Rubaiei, T. Al Yarubi, M. Al Saadi and B. Kumar, "SQLIA Detection and Prevention Techniques," 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), 2020, pp. 115-121, doi: 10.1109/SMART50582.2020.9336795.
3. Bandi Aruna and Bhimavarapu Usharani , "SQLID Framework In Order ToPerceive SQL Injection Attack on Web Application", *IOP Conf. Series: Materials Science and Engineering* 981 no. 022013, 2020.
4. Mohammad Abu Kausar, Mohammad Nasar, Aiman Moyaid, "SQL Injection Detection and Prevention Techniques in ASP.NET Web Application", *International Journal of Recent Technology and Engineering (IJRTE)* ,vol. 8, no. 3, 2019
5. Sarjiyus, O. and El-Yakub, M. B. , "Neutralizing SQL Injection Attack on Web Application Using Server Side Code Modification ", *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 5, pp. 2456–3307, 2019
6. Young-SuJangJin-YoungChoi, " Detecting SQL injection attacks using query result size", *textitScience Direct*, vol.44 ,pp, 104–118, 2014
7. OWASP (2010). Open Web Application-TopTen-Projects.
8. <https://www.bcs.org/pdf/infotec-pdf-principles.pdf>
9. Chandershekhar Sharma, Dr. S. C. Jain Dr. Arvind K Sharma, "Explorative Study of SQL Injection Attacks and Mechanisms to Secure Web Application Database- A Review", vol. 7.no. 3, 2016