International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

http://ijpsat.ijsht-journals.org
Vol. 27 No. 1 June 2021, pp.481-488

Security Measures Against SQL Injection Attacks

Intisar Milad Mohamed ALSSULL[1] and Aisha Abadalla Mahmoud LUSTA[2]

[1]ORCID no.: 0000-0003-2968-3856

[2]ORCID no.: 0000-0003-4521-0848

Abstract – Most technical web applications have been using the SQL standard in recent years. These Web applications transmit a structured query language to the database through SQL. Values are returned from the database using queries or parameters in web applications as per user requests. The results returned are displayed in certain formats to the user or administrator, depending on the programme architecture. However, malicious code entered by the attacker in web applications can be used to exploit SQL queries dynamically created. SQL is injected with the user's malicious code during the process. In other words, it is possible to retrieve the information by inserting malicious SQL commands in areas where data such as window address bars or access controls can be entered. This detail is not freely accessible and proprietary. By incorporating new and related dimensions to the SQL injection situation, the attacker will access other information in the database. Through running any commands on the database server, an attacker may harm the programme or server. An example and review of a SQL injection attack on a web application built on MSSQL-ASP.NET is discussed in this article. In addition, protection precautions and recommendations for solutions against browser-based web apps security vulnerabilities are listed.

Keywords – SQL, Program, Web, Application, Language, Database.

I. INTRODUCTION

SQL (Structured Query Language) is a structured query language used to perform operations such as selecting, deleting and updating data from databases. SQL, which is both ANSI and ISO standards, forms the basis of widely used modern database management systems such as Oracle, PostgreSQL, MSSQL Server, MySQL, and DB2 [1].

Today, almost all of the web applications that offer professional services use databases. These online web applications communicate with the database through SQL, which is a structured query language [2]. SQL Injection can be defined as the manipulation of SQL queries created with user input from web applications [3]. In data-based web applications that users interact with, the information in the database tables is filtered according to certain conditions by using queries or parameters and transferred to the application interface. These transferred result values are presented to the user or manager in certain formats according to the design of the application. The SQL injection method is done exactly when these processes are carried out. The attacker performs the SQL injection attack by inserting malicious code into the web browser address bar or access controls found in the application. Information that is not publicly available but obtained in this way may be important and confidential. The attacker can access other information in the database by adding different dimensions to the SQL injection scenario with this important information about the system and database. Then he realizes his goal by using the information he has obtained.

Corresponding Author: Intisar Milad Mohamed ALSSULL

In this study, SQL injection attack and analysis, which is one of the web attack techniques, on an ASP.NET-MSSQL based web application is presented. In addition, possible solutions are given against this attack.

II. WEB SECURITY

In today's technology age, there is a direct proportion between the use of the internet and the globalization of information. As the use of the Internet increases, information becomes global and information in the most remote corners is easily accessible via the Internet. According to the 2012 Information Society Measurement report of the International Telecommunication Union, the number of internet users in the world is increasing every year. Compared to 2011, the number of internet users reached 2.3 billion in 2012 with an increase of 11% [4]. With the widespread use of the Internet, ensuring the privacy and security of the information on the web has become more important personally and institutionally. Users use corporate information resources directly or indirectly while receiving and providing services through information systems. These resources can contain personal information. This situation has made the security of institutional information sources even more important [5].

Personal information kept in databases as a result of internet users' interactions with information sources can be used by internet hackers for malicious purposes through illegal access. This kind of illegal access causes serious damage not only to users but also to the economies of the country. According to Symantec Norton Cybercrime Report of 2012, total net loss of Cost of cyber crime in Turkey to find 556 million dollars. According to the report, this cost corresponds to as high as 110 billion dollars worldwide [6]. Looking at the numbers pronounced in the report, it is understood how important and necessary security is in the cyber field.

Web applications hosted on web servers; It consists of scripts that interact with databases or other content. In these applications, the server and the clients interact [7]. Since user requests are run on web servers, the servers are vulnerable to attack by attackers and are an open target.

With the development of web technologies, people, institutions and organizations started to use web applications more. The use of web applications has increased day by day, as it is easy to access online web services from anywhere, reduces time loss and makes transactions easier. Parallel to all these, there has been an increase in web attacks.

There have been many studies in the literature on SQL injection attacks. In Halfond et al., SQL Injection method; They defined it as a method that uses data provided by the client in SQL queries via web applications and a technique of adding meta characters and commands to web-based input fields in order to execute SQL queries running in the background. In addition, they classified SQL injection attacks and talked about the precautions that can be taken against them [8]. SQL on the OWASP (The Open Web Application Security Project) website

Injection method; It is defined as the injection of the input data from the client to the application into SQL queries. He talked about the dangers of this attack and what the attackers could do [9]. Gregory T.

In Buehrer et al., SQL Injection method; security attack type and defined it as the user input that a malicious user sends to the database as part of the SQL statement in the background. In his studies, he explained how this attack could be prevented by using the Parse Tree method [10]. Khaleel Ahmad et al. SQL injection method; He defined it as a method that uses the database of the web application, and stated that it was created by injecting SQL statements as an input string and gaining unauthorized access to the database [11]. In the study, SQL attacks are classified and exemplified.

III. SQL INJECTION ATTACK APPLICATION

In this section, an example of an SQL injection attack on a web application using an SQL database is presented. The attack was demonstrated on a simple ASP.NET news application using an MSSQL database. Tables and relationships of the application in the database are shown in Figure 1.

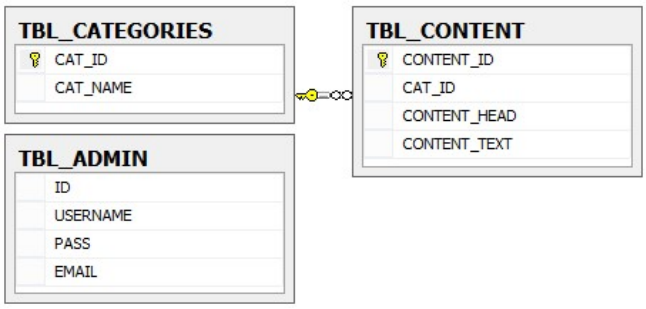

Figure 1. Database Tables and Relations

The attack begins by looking for the presence of a SQL injection vulnerability in the web application. Figure 2 shows the list of news categories of the sample web application Security Measures Against SQL Injection Attacks.

Figure 2. A Sample Snapshot from the Web Application

1.1. Detecting Vulnerabilities for SQL Injection Attack

There are several ways to find the SQL Injection vulnerability. One of these ways is accomplished by typing various SQL commands into the address bar of the browser used. This method mentioned in this application is used. In order to find the SQL Injection vulnerability with this method, we first need to find the pages that send data with QueryString and find out whether there is a security vulnerability on these pages. When clicking on the "Policy News" category from the categories shown in Figure 2, a QueryString value is sent to the Default2.aspx web form with the "QueryString" variable named "id" from the Default.aspx web form, as shown in Figure 3.

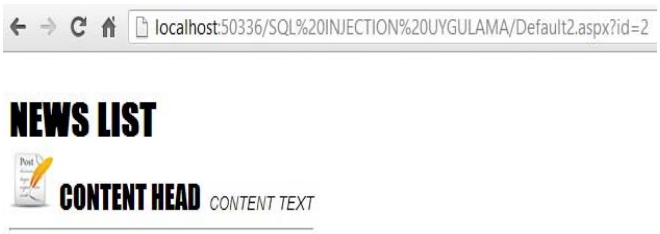

Figure 3. QueryString Submission

This value, sent with the variable name "id", shows the record number of the row in the related table in the database. After finding this page that sends data with QueryString, we need to check whether there is a SQL Injection vulnerability on this page. In order to do this, characters such as single quotes are written at the end of the URL address in the address bar, as in Figure 4. This process shows us whether an error has occurred in the database.

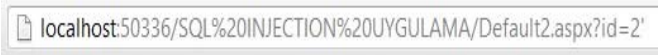

Figure 4. Using Single Quotation Marks

System.Data.SqlClient.SqlException (0x80131904): Unclosed quotation mark after the character string ".

Figure 5. Database error Example

As can be seen in the above figure, database error was tried to be taken by interfering with the URL address when sending data with QueryString. The occurrence of the error as in Figure 5 gives us a clue that this page can be used for the SQL Injection attack.

1.2. Gathering Information About the Database

We need to collect information about the database for the purpose of attack. We can learn the names of the table and table fields in the database by using the page where the error is shown in Figure 5. The attack should be directed according to the table and table field names learned. For this, we first need to determine what to do for the website to be attacked. If this website is to be taken over completely, it can be started by finding the sql table where the username and password of the administrator are kept.

When the "HAVING 1 = 1" SQL command is typed in the browser address bar for the attack, the error shown in Figure 6 was received. The difference of this error message from the error in Figure 5 is that it shows the name of the table containing the news in the database and the name of the primary key column in this table.

The HAVING statement is an expression used with the GROUP BY statement. The function of the HAVING statement is similar to that of the WHERE statement. However, since the clustering function GROUP BY and WHERE statements cannot be used together, the HAVING statement was needed. If there is no table space defined in the HAVING statement, the table area, which is the primary key, is defined by default after the HAVING statement. This adds the name of the table used and the name of the primary key area to the error as in Figure 6 [12].

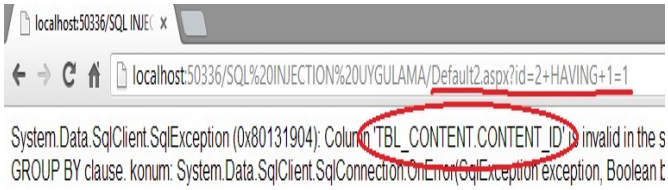

Figure 6. HAVING Command Application Result Screen Output

It can be understood from Figure 6 that there is a table named TBL_CONTENT in the database where the attack was applied and this table has a field called CONTENT_ID. Some modifications should be made to the SQL query to find the names of other fields in this table. Since it is known that there is a field named CONTENT_ID in the table, it is necessary to learn the other table names by using the GROUP BY statement before the HAVING statement. As seen in Figure 7, when GROUP BY CONTENT_ID expression is written in the address bar before the HAVING statement, the name of another field in the table was seen in the error message. In order to learn other domain names, the other domain name obtained after the GROUP BY statement should be written in the address bar as in Figure 8, separated by commas. This way all other table field names can be found.

Figure 7. Error Screen Output

Figure 8. Error Screen Output

In this way, the names of a table in the database and all the fields in this table are learned. Now, the name of the table where the manager information is kept, which is the main target, must be found. For this, a few changes have to be made to the SQL statement in the address bar. Using the UNION statement, SQL queries can be combined and an error can be generated on the screen. The UNION statement is used in SQL language to combine the results of two or more queries. The point to note when using this command is that the number of columns selected by select queries on both sides of the statement is the same. It was mentioned above how the names of the fields in the TBL_CONTENT table are found. It can also be seen from the database diagram in Figure 1 that this table has four fields. To correspond to these four fields, a select query should be written after the UNION statement. The query expression listing the table names in the database for this select query is shown in Figure 9.

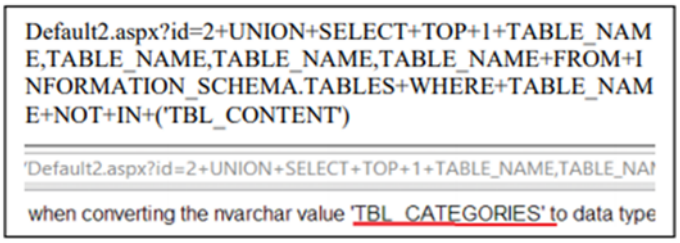

Figure 9. UNION Command Application and Output

On the left side of the UNION statement, the SQL statement showing the content with the id variable value 2 is available in the background in the web form. The statement added to the right of the UNION statement as seen in Figure 9 and the results of select queries are combined. The expression "TABLE_NAME" was repeated four times as there are four fields in the TBL_CONTENT table and

With the statement of NOT IN ("TBL_CONTENT"), the name of the table, whose name was previously known, is not reappeared. As seen in Figure 10, the name of another table in the database was found. Since the purpose is to find the manager table, the name of the other table has been added to the SQL statement in the address bar as in Figure 10. In this way, the name of each found table must be appended to the end of the SQL statement until the manager table is found. As a result of this process, TBL_ADMIN table is included in the error message as seen in Figure 10. A new administrator can be added to the TBL_ADMIN table and the system can be taken over with the added login information or the tables in the database can be deleted. In summary, as long as the attacker knows the SQL language, he can inflict any damage on the system through the vulnerability he finds.

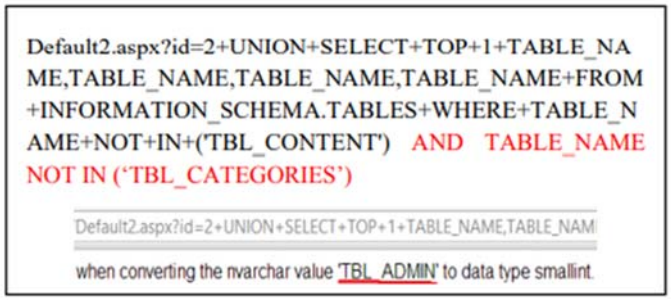

Figure 10. Learning the Name of the Manager Table

1.3. Scenario Plan and Implementation of the Attack

After finding the manager table of the application, the field names of the TBL_ADMIN table can be learned with the HAVING statement, as described at the beginning of the study in line with the target of the attack. This will be required to add a new manager to the TBL_ADMIN table. In order to add a new manager, the SQL statement in the address bar must be changed as in Figure 11.

Figure 11. Adding a New Manager with the INSERT INTO Statement

The image of the TBL_ADMIN table in the database before this command is shown in Figure 12. After the command is applied, the table view is shown in Figure 13. Now that a new manager is added with the INSERT INTO command, the entire administration of the website is taken over.

Security Measures Against SQL Injection Attacks

	ID	USERNAME	PASS	EMAIL
1	1	ADMIN	PASS123	EXAMPLE@EXAMPLE.COM

Figure 12. Admin Table Before INSERT INTO Command

	ID	USERNAME	PASS	EMAIL
1	1	ADMIN	PASS123	EXAMPLE@EXAMPLE.COM
2	2	Hacker	12345	hacker@example.com

Figure 13. Admin Table After INSERT INTO Command

1.4. Application Results

As a result, the actions to be taken after a SQL Injection vulnerability is detected on a website is determined according to the result of the errors received, and the management of the website can be taken by acting in this direction. The size of the attack may differ from the information obtained through these processes.

IV. Conclusion And Recommendations

While making a web application;

- Determining the table and table field names in the database in a way that is difficult to guess [13],
- Using parametric queries in web forms [14],
- Verifying these data while entering data into the web forms access controls, controlling the input length [14],
- For numerical values used in web forms and representing a record line in the database (eg QueryString value), checking whether these values are numerical values in transitions between forms [15],
- In SQL Server-based web applications, after the user has entered data, the SQL query characters sent to the database are searched and dangerous characters, replace etc. Converting into harmless characters with commands that will not cause an error in SQL Server [15],
- Ensuring that the errors that will occur in SQL Server are not displayed in web forms [15],
- Managing operations such as writing, reading, and deleting databases used in web applications by only one administrator [16],
- Instead of writing queries to web forms in the application, writing these queries as a Stored Procedure in the database section [14, 16],
- Filtering the words that can be used in SQL Injection method (select, insert, update, etc.) with a function,

- Operating the database with a restricted user account [16],
- Removing unused stored procedures and administrator accounts [17],
- Removal of public access for system objects,
- Installing a firewall as software or hardware on the server with the web application or using the existing firewall in the server operating system [18]

It is one of the security measures that can be taken against the SQL Injection method. These items can be reproduced according to the size of the attacks, but in general, if these recommendations are observed, the application will be largely protected from SQL Injection attacks.

REFERENCES

[1]. Tipton, H. F., & Krause, M. (Eds.). (2006). *Information Security Management Handbook, Volume 3* (Vol. 3). CRC press.

[2]. Vacca, J. R. (Ed.). (2013). *Managing information security*. Elsevier.

[3]. Burkhead, R. L. (2014). *A phenomenological study of information security incidents experienced by information security professionals providing corporate information security incident management* (Doctoral dissertation, Capella University).

[4]. Lyall, F. (2011). *International communications: The international telecommunication union and the universal postal union*. Ashgate Publishing, Ltd..

[5]. Von Solms, R. (1999). Information security management: why standards are important. *Information Management & Computer Security*.

[6]. Peltier, T. R. (2016). *Information Security Policies, Procedures, and Standards: guidelines for effective information security management*. CRC Press.

[7]. Rieck, K., & Laskov, P. (2006, July). Detecting unknown network attacks using language models. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 74-90). Springer, Berlin, Heidelberg.

[8]. Halfond, W. G., & Orso, A. (2005, November). AMNESIA: analysis and monitoring for neutralizing SQL-injection attacks. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering* (pp. 174-183).

[9]. Borade, M. R., & Deshpande, N. A. (2013). Extensive Review of SQLIA's Detection and Prevention Techniques. *International Journal of Emerging Technology and Advanced Engineering*, *3*(10), 614-626.

[10]. Boyd, S. W., & Keromytis, A. D. (2004, June). SQLrand: Preventing SQL injection attacks. In *International Conference on Applied Cryptography and Network Security* (pp. 292-302). Springer, Berlin, Heidelberg.

[11]. Halfond, W. G., Viegas, J., & Orso, A. (2006, March). A classification of SQL-injection attacks and countermeasures. In *Proceedings of the IEEE international symposium on secure software engineering* (Vol. 1, pp. 13-15). IEEE.

[12]. Clarke-Salt, J. (2009). *SQL injection attacks and defense*. Elsevier.

[13]. Bruchez, R. (2012). *Microsoft SQL Server 2012 security cookbook*. Packt Publishing Ltd.

[14]. Muthuprasanna, M., Wei, K., & Kothari, S. (2006, September). Eliminating SQL injection attacks-A transparent defense mechanism. In *2006 Eighth IEEE International Symposium on Web Site Evolution (WSE'06)* (pp. 22-32). IEEE.

[15]. Wei, K., Muthuprasanna, M., & Kothari, S. (2006, April). Preventing SQL injection attacks in stored procedures. In *Australian Software Engineering Conference (ASWEC'06)* (pp. 8-pp). IEEE.

[16]. Sadeghian, A., Zamani, M., & Manaf, A. A. (2013, September). A taxonomy of SQL injection detection and prevention techniques. In *2013 international conference on informatics and creative multimedia* (pp. 53-56). IEEE.

[17]. Lee, I., Jeong, S., Yeo, S., & Moon, J. (2012). A novel method for SQL injection attack detection based on removing SQL query attribute values. *Mathematical and Computer Modelling*, *55*(1-2), 58-68.

[18]. Newman, R. C. (2006, September). Cybercrime, identity theft, and fraud: practicing safe internet-network security threats and vulnerabilities. In *Proceedings of the 3rd annual conference on Information security curriculum development* (pp. 68-78).