International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 07 | July - 2022 Impact Factor: 7.185 ISSN: 2582-3930

SQL INJECTION VULNERABILITIES

Meghana M[1], Dr. Vibha M B[2]

[1]Student, [2]Associate Professor,

Department of MCA, Dayananda Sagar College of Engineering, Bengaluru, Karnataka, India

Abstract

In present world the internet has become the most salient aspect of everyone’s day to day life. It is important to consider the security of the web applications and software services in order to protect devices from external attacks. So, any web application becomes prone to cyber threats where these threat attempts to steal a person’s information from various onsite applications. Every web application or software or hardware will have certain vulnerabilities through which a hacker can get into the target systems database to destroy services. This paper focuses on detecting the SQL injection attack and vulnerabilities of the software system which can be exploited by attacker. This will help us to understand the security level of one’s system which is exploited due to poor coding practices.

Keywords – SQL injection, vulnerabilities, detection, prevention

1. Introduction

SQL injection was considered as the top most attack based on OWASP[2] in a security vulnerability report and responsible for countless data breaches across the world. The personal data that we insert into website gets stored in the database so that we can access it from anywhere using the network. Thus, making the attacker to Grab those data from the vulnerable software, Hardware and web application.

SQL injection is done through SQL query under data manipulation. A strong SQL injection exploit has ability to unpack sensitive data also modify database data such as insert, update and delete. Apart from modifying database data, SQL injection can execute administration operations to retrieve data from database, manipulate database and , track down the content of a given file present in the database file system. In this attack a malicious code is injected into the strings that are passed into a SQL server for parsing and execution. Any parsing techniques that build SQL statements must be checked against injection vulnerabilities since SQL server will execute all queries that are synthetically right and returns true response back to the application. SQL injection vulnerability has the capacity to bypass the web application to the backend database. The professional hacker who is highly skilled can manipulate parameterized data also.

2. Literature survey

Firstly, SQL injection was openly started to appear in 1998. Jeff Forristal, alias RainForest puppy, has been recognised as an industry expert, a cybersecurity researcher and hacker who is responsible for documentation and discovery of SQL injection. His discoveries were published in a hacker zine called Phrack. This Digital magazine includes both hacking and computer security topics. The article about SQL injection first appeared in the 54th article of Phrack, where Forristal found something unusual while penning about how to break into Windows NT server. He reprimanded for use of raw SQL on SQL server in web applications and passing user’s dynamic input as parameters[3]. This was discovered as a side script to their experimental conduct of SQL vulnerabilities against Microsoft SQL server 6.5[2].

Likewise, the Phrack article paved a way for SQL injection analysis and research. Secondly, an advisory called Allaire group Once posted regarding the ability to batch commands and generalized attack methods against Sybase SQL, ASP and Microsoft SQL applications[2]. Allaire's Advisory assumed that only a few groups of systems were vulnerable due to variables being enclosed within the quotes[1]. Thirdly, Microsoft raised an argument with the issues figured by RainForest puppy and stated them as features but not vulnerabilities[1]. As a result, Rainforest puppy eagerly started his research on SQL injection techniques. As contrary to Microsoft's claim RainForest puppy was successfully able to demonstrate Barriers to SQL injection and exploited implied security features, where they broke into the structure of the database by inserting random data into the database. By analysing the error reports generated by the SQL commands the structure of the database was recited closely[2].

3. SQL injection vulnerability exploit

When you access a website, you are trying to access different components of the website from Several different Technologies. The web components include a web server that has images, documents, MP4 and MP3 files. In addition to this there is a database that's connected to the web application, that includes important information pertaining to that website. This information could be the name and address of the users and their banking details such as debit card details[7]. Primarily, an injection attacks takes place when an attacker tries to take advantage of a poorly written code that benefits the attacker to gain access to the database containing those typical Information that are intended to be protected. When a web application is not tested against exploitable SQL commands ,it allows SQL commands to run against the database [7]. Let's take an example of a web application that accepts username and password , then provides users access to the website. At the back-end website has a database which stores username and password and other sensitive information. When a user enters username and password, an SQL Query is being created. The SQL query is executed to search against that database to verify its user, if the value matches with the database, then the user is authenticated to the website. But if the application doesn't do proper user input checks, that hacker could manipulate the username and password so that the outcome would still execute SQL command. If hacker uses appropriate SQL syntax then it could modify the database to return true response back to the web application and the manipulates user access to that site.

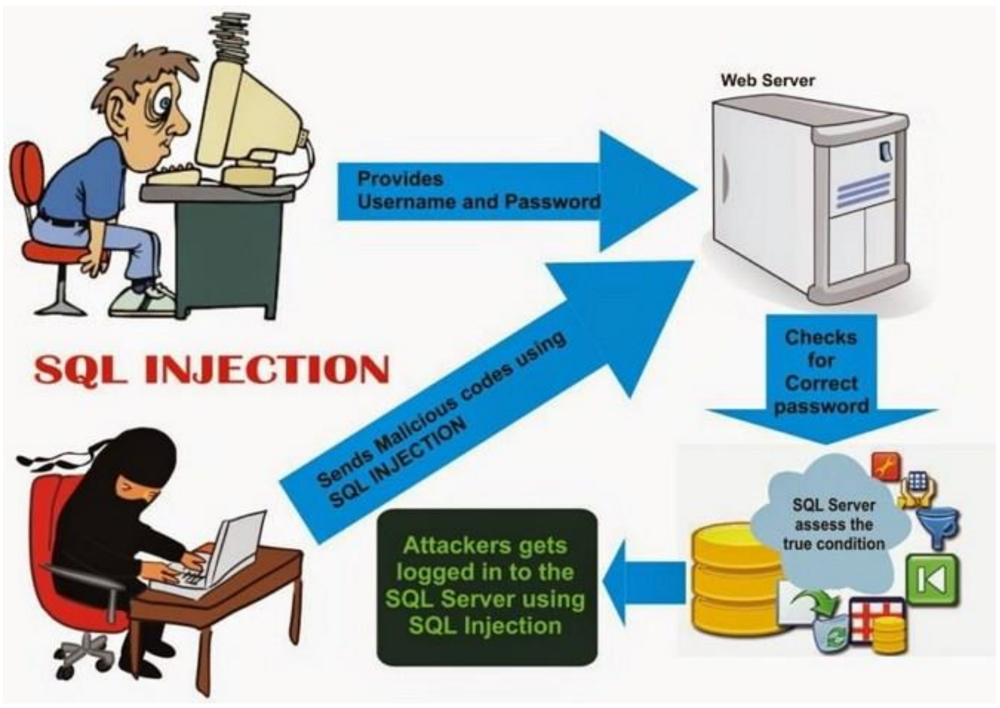
Web Server
Provides Username and Password
Checks for Correct password
SQL INJECTION
Sends Malicious codes using SQL INJECTION
SQL Server assess the true condition
Attackers gets logged in to the SQL Server using SQL Injection

Figure 1 : SQL injection attack works

In this case the username could be manipulated as ' ' OR 1=1 it manipulates SQL commands which looks like this.

Select * from users where name= ' ' OR 1=1

Even though the username seems to be weird still it makes valid sql command to the database. As long as the username is either of course or as long as the statement 1 = 1 is true, the database returns true statement even though there is no username that has a quote. But the logical statement 1 = 1 is true. When the user prompts to enter the phone number of the user. If he enters 8317439125, The Query arranged looks similar to the below script.

Select * from users where userPhoneNo = ' 8317439125';

Now Assume that the attacker types the following script.

8317439125'; drop table users--

In this case the below query is assigned by the script.

Select * from users where userPhoneNo= '8317439125'; drop table users - -'

Hence, true response is returned back to the application[5].

Below shows SQL query on attaching it to the two hardcoded strings together with attacker provided string.

Var userPhoneNo;
userPhoneNo = Request.form("userPhoneNo");
var sql= "select * from users where userPhoneNo= ' " + userPhoneNo + " ' ";

Use of semicolon will terminate one query and start another query. The double hyphen used as comment and to restore the query. When the query is syntactically correct SQL server processes the statement. SQL server will select all records in the users table where userPhoneNo is 8317439125. Then SQL server will drop the user's table. From this we understood that when the syntax of inserted sql code is correct, modifications cannot be detected programmatically.

4. SQL injection vulnerabilities

Potentially, this loophole allows attackers to surge to more damaging attacks within a network that sits behind the firewall.
SQL injection are assorted into three major types:

4.1. In-band SQLi

In-band SQL is the most common type of SQLi where an attacker uses same communication channel to launch the attack and to collect results through the same channel. In-band SQL is classed into 2 types.

4.1.1. Error based SQLi

Here, the attacker performs operations that trigger the database to produce error messages which are given to the attacker. The hints could be Type of database it utilizes, version of the server and location of the handlers.

Example 4.1.1.1 : Comment line(--) causes the database to ignore some Valid part of the query[6].
Select * from users where user_ID=abc' or 1=1--
Example 4.1.1.2 : By using a valid statement which returns true[6].
Select * from users where username=abc' or 'a'='a'-- and password='pass'

4.1.2. Union based SQLi

Here, the union SQL operator is used to get the combined results of two or more selected statements into a Single statement that is returned as a part of http response.

Example 4.1.2.1 : creating malicious inputs to execute malicious queries[6].
Select * from users where user_ID=1 Union select 1,Database(), user(),4#
Example 4.1.2.2 : Inserting logically incorrect queries to see error messages and get useful hints from the database[6].
select * from order where order_ID=1'
This will generate a Syntax error and will reveal the backend database type.

4.2. Inferential SQLi

In inferential SQLi no malicious input is injected to the web application to see the attack results like in-band SQLi. Instead, the attacker will be able to reconstruct the web applications database by observing web application responses on sending payloads. They are classified into two types.

4.2.1. Boolean SQLi

The attackers forces the SQL Query that makes database to send different Results whether the Query returns true or false. As a result, the content of the http response might change or remain the same. Payloads sent by the attacker returns either true or false No data is returned back as a response. Since it is a slow attack, it favours the attacker to enumerate the database.

Example 4.2.1.1 : A vulnerable data access layer of the application might have the below url:

https://GOTGOTOmicffr.com/goals.php?id=2
select * title, description, body from goals where ID = 2 and 1=2
If the application is vulnerable to the attack IT returns true you are the attacker will send a query which returns true condition. And the content of the page is different, it infers that the query is working.

4.2.2. Time delay SQLi

In Time SQLi, the attacker injects SQL Query function that Forces the database to wait before generating results. The delay in server response helps attack to decide whether the result is true or false.

4.3. Out-of-band SQLi

Out of Band SQLi technique is when the attacker is unable to use same communication channel for both launching attack and gathering results. This technique works only when the features on the database being used by the web application are enabled. Out of Band SQL technique would make the database server vulnerable and make DNS or http request to deliver data to the attacker.

5. Detecting exploit

The following simple steps can be followed to detect the exploit in any web application.

1. Take any web application which can be tested against SQL injection attack.
2. Try to bypass the login by giving username = Any SQL injection test and user pass= 123[1].

3. Try to login to the main page by giving the right password.
4. You can also test by attacking manually by inserting string keys, to detect anomalies. For example, abc', abc'), 1 OR, 1)
5. After the query executes, the page returns an error. If the page Returns 1064 error pointing to the SQL syntax, that corresponds to the MySQL server version and suggests the right syntax.
6. From the above security flaw, it is more difficult to say what exactly the loopholes are. But it is obvious that malicious input has interrupted the server-side validation. That facilitates attackers to pass some more malicious inputs to bypass the tests.

Apart from detecting potential vulnerabilities manually, there are detecting tools where vulnerabilities can be detected. some well-known detecting tools are SQLMAP, BURP SUITE etc.

6. Prevention

1. Validate user inputs.
2. Mitigating inadequate data sanitization, where string concatenation is abandoned. Including mysql_real_escape_string() can avoid authenticated queries.
3. Validating user inputs and sanitizing data alone cannot fix all problems. Use of Prepared statements and parameterized inputs has to be mandatory.
4. Enforcing improved coding practices.
5. reducing the database privileges to the users.
6. Regular auditing of the database and conducting penetration tests to check the security of the application.
7. Avoid shared database and user accounts
8. Use software based or web-application based firewalls.

7. Conclusion

A web application becomes vulnerable to sql injection when the inputs are not properly validated. When the attacker gets access to the site it endangers the authentication bypass, disturbs the Data integrity and threatens the information disclosure. Since everything is made digital such as cloud storage, it could be easily accessible by a professional attacker. Before developing any application, one needs to understand the structure of the database administrators, also needs to follow certain methodological approach to detect SQL injection vulnerabilities. A secured website accepts validated input, provides minimum database access privileges to users and involves strong firewall that monitors networks and error message customisation. The security experts are continuously working towards the security of the systems from being attacked. Hence, the developers are instructed to incorporate those security measures in their software.

8. References

1. Abdul-Aziz A. Sarhan , Shehab A. Farhan and Fahad M. Al-Harby Understanding and Discovering SQL Injection Vulnerabilities.
2. Rubidha Devi.D , R.Venkatesan, and Raghuraman.K a study on sql injection techniques.
3. http://phrack.org/issues/54/8.html
4. https://www.esecurityplanet.com/networks/how-was-sql-injection-discovered/
5. https://docs.microsoft.com/en-us/sql/relational-databases/security/sql-injection?view=sql-server-ver16
6. https://www.greycampus.com/opencampus/ethical-hacking/types-of-sql-injection
7. https://www.simplilearn.com/tutorials/cyber-security-tutorial/what-is-sql-injection
8. https://www.acunetix.com/websitesecurity/sql-injection2/