International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320-088X

IMPACT FACTOR: 7.056

IJCSMC, Vol. 10, Issue. 5, May 2021, pg.60 – 66

SQL Injection and Areas of Security Concern

Shubham Singh¹; Pranju Mishra²; Samruddhi Kshirsagar³; Shubham Bharadia⁴;
Assistant Prof. Narendra Joshi⁵

¹Department of Computer Science, Sandip University, India

²Department of Computer Science, Sandip University, India

³Department of Computer Science, Sandip University, India

⁴Department of Computer Science, Sandip University, India

⁵Department of Computer Science, Sandip University, India

¹ shusig2603@gmail.com; ² pranjumishra@gmail.com; ³ samruddhikshirsagar100@gmail.com;
⁴ shubhambharadia11@gmail.com; ⁵ narendra.joshi@sandipuniversity.edu.in

DOI: 10.47760/ijcsmc.2021.v10i05.006

Abstract— *Cyber-crimes are growing rapidly and to prevent these crimes one should share all the knowledge he/she has to make people aware of these attacks. In the field of Application Security there is a very well-known vulnerability “SQL INJECTION”. In this paper, we have focused on what are the type of SQL Injection attacks and where it can be found in any application.*

Keywords— *SQL Injection (SQLi), Burpsuite, Injecting, vulnerability, parameters*

I. INTRODUCTION

SQL injection is a vulnerability mostly found in applications that uses Structured Query Language (SQL) to manage their databases. It is a vulnerability using which any user can inject SQL queries inside the database and access data that he/she was not supposed to access, directly from the application which is having this vulnerability. SQL queries can be inserted inside the database from the application by modifying the current running query in the current page of the application. This modification of the query can be done either directly from the application or by using a proxy tool such as Burpsuite. To learn more about burpsuite please visit the official website of portswigger.

It is a very critical vulnerability because a successful SQL injection can lead to unauthorized data access, compromising the database, database operating system, escalating privileges, uploading shell, compromising the server, compromising other application hosted on the same server ...etc.

It is powerful and dangerous when performed manually by making your own payloads and using it. However, it can be performed easily using different opensource tools as well and one of the best tools is ‘sqlmap’. As a hacker one should know that automated exploitations are not as efficient as a manual exploitation.

There are different types of SQL injection techniques which can be used to determine the database used by the application and can be used to extract data from the database. This extracted data could

be visible on the application page or somewhere out of the application scope like email or some server possessed by you.

SQL injection in the current time period is very difficult to find because organizations are very well aware about it and taking all the precautions to prevent this attack, but we do know that nothing is completely secure. So, in this paper we will see some of the very interesting places where one can test for SQL injection vulnerability, but before that we will see different types of SQLi attacks.

NOTE: -Please note that before start exploiting SQL injection vulnerability one should have a basic grasp on SQL. In this paper we will focus on MySQL. But you can learn about other database languages as well such as MS-SQL, Oracle, NoSQL. Etc.

II. TYPES OF SQL INJECTION ATTACK

There are three types of SQL injection: -

- 1) - Error Based
- 2) - Union Based
- 3) - Blind SQL injection

- Error Based SQL Injection: -

It is a very common type of injection attack.

In this type of attack, an attacker uses error, which is thrown by the database, to extract data from the database after supplying a malicious input on the application.

What is a malicious input? - an input that breaks the syntax of the SQL is known as malicious input. e.g. - “ ‘ “, “--”, “//”, “#”...etc.

To determine whether an application is prone to error-based injection or not, you have to supply these malicious inputs to the application. (Don't worry we will talk about “where to supply” further in this paper) If the database throws an error which is visible on the application's page, that means the application is vulnerable to error-based injection.

As you can see above, we have encountered an error in the application after supplying malicious input. Now we can use this error to extract information from the database. We will create payloads such that whatever we want to extract gets extracted in the error message that we encountered before.

So, to put your desired query and also generate an error message, we will use the most used trick. i.e., type conversion trick. Using this trick, we can put our desired query in the database and simultaneously produce error so that the error message will contain or extracted information.

Function used: - CAST (it converts one data type into another datatype)

Payload: - ‘ or 1 in SELECT CAST(<fieldname> as numeric) from ‘table_name’ where ‘condition’

e.g.: - ‘ or 1 in SELECT CAST(@@version as numeric)

(here we are extracting version information but by generating an error using the CAST operator)

- Union Based SQL Injection: -

In this type of attack, we use a very powerful operator of SQL i.e., “UNION”.

This is used to extend and query and add two or more ‘select’ statements in a single query.

NOTE: - Each ‘SELECT’ statement before and after ‘UNION’ has same number of columns.

Suppose, an online shopping website is using following SQL query to fetch products: -

```
SELECT price,quantity from products where category='mobile'
```

Now, we can extend this query by using UNION operator: -

```
SELECT price,quantity from product where category='mobile' UNION SELECT  
username,password from users' --
```

The output of the above query will contain product information as well as user’s information.

Now, before using a “UNION” operator, we first need to know that how many columns are in the current page’s query. So, there are different tricks of determining the number of columns of a particular web page using UNION operator.

-Determining number of columns: -

To determine the number of columns, you simply have to use the same “UNION” operator, but in a different and clever way. We will use “NULL” trick as follow -

```
‘ UNION SELECT NULL --
```

(if you get an error on the webpage, that means the web page has more than one column and if the webpage is loaded correctly, then the webpage is having one column)

```
‘ UNION SELECT NULL, NULL --
```

(if you get an error on the webpage, that means the page has more than two column and if the webpage is loaded correctly, then the webpage is having two column)

```
‘ UNION SELECT NULL, NULL, NULL --
```

```
‘UNION SELECT NULL, NULL, NULL, NULL -- (and so on)
```

NOTE: - number of “NULL” will determine the number of columns.

NOTE: - whenever you are trying to exploit a SQL Injection vulnerability, you should always take care of the query syntax as well as the syntax in which the application is processing the input. Even if your query syntax is correct but the syntax of the application is not added by you properly still you won’t be able to exploit the vulnerability and you will not get the desired result.

- Blind SQL injection

This type of SQL Injection does not display any output or any error message and that's why its name is blind SQLi. It is difficult to perform and detect because it depends on the observation power of the attacker. If you get to know that an application is taking SQL input but there is no error message or extra information is getting displayed on the webpage. Then at that point of time you may use this attack.

In this attack, an attacker has to use “true” - “false” SQL statement and check the response of the server on the webpage. Based on the observation an attacker puts the information that he/she wants to extract from the database in the query in a true-false manner.

E.g.: -

True condition's output: -

False condition's output: -

Now based on these above outputs, we can send payloads to server constructed in a true-false manner.

E.g., Payload: -

`1' and substring(@@version,1,1)=1;- -`

(if the first character of the database's version is '1' then the true condition's output will be visible and if '1' is not the first character then false condition's output will be visible)

Now we have seen what are the types of SQL Injection attacks. Further we will move to the most interesting part of the paper that is “Where to find these SQL Injection attacks?”. Even after taking all the precautions to protect application from SQL Injection, there may be some places left where SQLi can be tested which developers and the team would never have thought of. We will see which are these interesting places.

III.IMPORTANT AND INTERESTING PLACES WHERE SQL INJECTION COULD BE TESTED IN ANY APPLICATION/AREAS OF SECURITY CONCERN.

We have seen above that what are the different types of SQL Injection attacks and how to exploit them. Now in this section we will focus on where we can find these injection places. Before moving to the injection places, first one should know that what is the basic of detecting an SQL injection vulnerability.

-Detecting a SQL Injection Vulnerability: -

To detect an SQL Injection vulnerability, one should observe the response of the web application when supplied different and unexpected input. If a web application throws an error or behaves unexpectedly or displays some extra information or displays less information after supplying those unexpected input. Then there is a very high probability that the application is vulnerable to SQLi. In such situations, one should investigate the case further by supplying more such payload which are intended to give some output.

What are these unexpected inputs?

These inputs are any input that could break or add some extra information to the syntax of any the database language such as: -

- i) - ‘
- ii) - “
- iii) - ”
- iv) - //
- v) - - -
- vi) - --
- vii) - #
- viii) -)))
- ix) -))
- x) -)
- xi) - ‘ or 1=1 - -
- xii) - ‘ or “
- xiii) - ‘ or ‘ ’ = ‘
- xiv) - ‘ or ‘x’=’x
- xv) - AND true
- xvi) - AND false
- xvii) - %00
- xviii) - 1-false
- xix) - 1-true

There are so many such unexpected input available on the internet, you can search such inputs on google for more information.

Now the question comes, where to inject these inputs?

To answer this question, we would like say just one word i.e., “parameters”.

Yes, these inputs should be injected in parameters. A web application consists of number of parameters for processing there functionality properly. These parameters could be visible either on the URL itself if “GET” request is being used and could be visible on a traffic intercepting tool if “POST” request is being used by the application for a particular request.

Places and parameters to be tested for SQLi: -

i) Place: - **URL**

Parameter: - any parameter that is intended to fetch some result or to validate an object.

E.g.: - ID=1, name=amar, user=41025, product=1 ...etc.

ii) - Place: - **POST request** (can be seen by any traffic intercepting tool)

Parameter: - any parameter that is intended to update, check, enter some information from the database.

E.g.: - username=test@fmail.com and password=test123, name=test, user=134526...etc.

iii) - Now if you have observed properly, then you would have seen that some websites are actually storing its user's metadata such as user-agent, timing of login, location of user. Etc in their database. There is a very high probability that these metadata are not getting sanitized properly or not even getting sanitized a bit. So, in such case it is worth testing these parameters. These parameters could be seen in the post request using some traffic intercepting tools.

Place: - POST request

Parameter: - any parameter that is processing users meta data.

E.g.: - time=10:00AM, user-agent=mozilla, request_used=get, current=india ...etc.

NOTE: - There could be n number of parameters that a web application could be using based on their functionality. You have to check each and every request of that you are making to use that web application's functionality. You to have understand how that web application is working with those parameters and based on the that try to find out which are all the places where SQLi could be tested. A hacker should always have following mindset: -
"Its target is not secure".

CONCLUSION

A hacker is always successful if he/she has observed, Recon, its target carefully and properly. So, before testing and exploiting the target first know you target.

Happy Hacking!

REFERENCES

- [1]. Dafydd Stuttard, Marcus Pinto, "The web application hackers handbook: Discovering and Exploiting Security Flaws"
- [2]. <https://pentest-tools.com>
- [3]. <https://en.wikipedia.org/wiki/Wiki>
- [4]. <https://owasp.org/>
- [5]. <https://portswigger.net/>
- [6]. Patrick Engebretson, "The Basic of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy"
- [7]. Peter Yaworski, "Real World Bug Hunting: A Field Guide to Web Hacking"