International Journal For Technological Research In Engineering
Volume 7, Issue 3, November-2019 *ISSN (Online): 2347 - 4718*

A REVIEW PAPER ON SQL INJECTION ATTACK IMPLEMENTED THROUGH PHP AND PREVENTION TECHNIQUES

Payal Arora[1], Deepa Dhabhai[2]
(PG Student[1], Assistant Prof.[2] Dept. Computer Science)
Shekhawati Institute of Engineering and Technology, Rajasthan, India

Abstract: SQL Injection Attack (SQLIA) is a generic and critical security issue towards to the web application and database security. In general, not well validated and verified web applications are highly prone and vulnerable by the attackers. Due to the creative and dynamic SQLIA methods and techniques, users can save their valuable, integral and confidential data in the web site to save their market stability towards their self as well as social enrichment.

Many tools and techniques are addressed to the references regarding the SQL Injection issues, but we are present and used pattern matching techniques in SQL statements to implement the SQLIA in web application. At the outset pattern matching algorithm is used and get better solution towards on implementation of SQLIA.

Keywords: SQLIA, SQL queries, web application, php.

I. INTRODUCTION

The key to understanding SQL Injection is in its name: SQL + Injection. The word "injection" here doesn't have any medical connotations, but rather is the usage of the verb "inject." Together, these two words convey the idea of putting SQL into a web application.

Putting SQL into a web application . . . hmmm . . . Isn't that what we're doing anyway? Yes, but we don't want an attacker to drive our database

Web App or Web applications are the tools to share and display their personnel or self-valuable information to access in worldwide network from anywhere. Regarding the usage of the web applications have many benefits, out of which some of the issues are much more risky. Now a day's information as well as network security is a big concern, where users can face many types of attack, out of which one simple and common category is SQLIA.As we know that Web applications are one such tool to access and transfer various self -information across the world with internet facility. Meanwhile some creative and dynamic hackers are also waiting and accomplish to hack this precious information from internet and breach the privacy of users. So that attackers have use many techniques, out of which SQL Injection Attack (SQLIA) is the major and common attacks performed by the attacker. As per the internet and network security concern SQLIA is one of the serious threats in the web application and SQLIA is considered to be in top ten vulnerability for web application usage. Now a day's SQLIA will be the easiest way to hack or attack the Web applications with various servers in using in World Wide Web protocol, i.e. if the web applications is coded very Poor in programming language or if the system files are not uploaded in the system, then it dam sure that, such web applications are very easy to hack by the attackers. It has been found that to detect and prevent SQLIA tour system, we must use some efficient algorithm to safeguard our personal data, in this paper we analyze and use pattern matching algorithm to detect and prevent SQLIA. Naive String pattern matching algorithm is used and tested efficiently in SQLIA.

1.1 Importance and Motivation of SQL Injection Attack (SQLIA)

SQLIA is one kind of attack, towards the vulnerability; where attacker is injecting some malicious code within this'll statement inserted into the web application. In the database package, if the application encounter the SQL Statement is SQL statement will be executed immediately and perform SQLIA within the system, so that the attacker can modify, extract and execute very sensitive and valuable information in the database. Basically hacker will take the advantages of poorly coded and very weak validated input based web applications has major targeted system, but once hacker will get successful execution of SQLIA to that application, the sanctity, integrity and confidentiality of the data to be bared, which results loss of social sanctity in the society as well as organization values.

1.2 Preventing SQL injection in PHP

Now, given that database connections, queries, and user inputs are part of life, how do we prevent SQL injection? Thankfully, it's pretty simple, and there are two ways to do it:

1) Disinfect user input
2) Use prepared statements.

Disinfect user input

If you're using an older PHP version (5.5 or lower, and this happens a lot on shared hosting), it's wise to run all your user input through a function called mysql_real_escape_string(). Basically, what it does it remove all special characters in a string so that they lose their meaning when used by the database.

For instance, if you have a string like I'm a string, the single quote character (') can be used by an attacker to manipulate the database query being created and cause a SQL injection. Running it through mysql_real_escape_string()produces I\'m a string, which adds a backslash to the single quote, escaping it. As a result, the whole string now gets passed as a harmless string to the database, instead of being able to participate in query manipulation.

There is one drawback with this approach: it's a really, really old technique that goes along with the older forms of

database access in PHP. As of PHP 7, this function doesn't even exist anymore, which brings us to our next solution.

Use prepared statements

Prepared statements are a way to make database queries more safely and reliably. The idea is that instead of sending the raw query to the database, we first tell the database the structure of the query we'll be sending. This is what we mean by "preparing" a statement. Once a statement is prepared, we pass the information as parameterized inputs so that the database can "fill the gaps" by plugging in the inputs to the query structure we sent before. This takes away any special power the inputs might have, causing them to be treated as mere variables

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);
// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();
$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();
$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();
echo "New records created successfully";
$stmt->close();
$conn->close();
?>
```

SQL injection and modern PHP frameworks

The SQL injection is so common, so easy, so frustrating and so dangerous that all modern PHP web frameworks come built-in with countermeasures. In Word Press, for example, we have the $wpdb->prepare() function, whereas if you're using an MVC framework, it does all the dirty work for you and you don't even have to think about preventing SQL injection. It's a little annoying that in Word Press you have to prepare statements explicitly, but hey, it's Word Press we're talking about.

Anyway, my point is, the modern breed of web developers don't have to think about SQL injection, and as a result, they aren't even aware of the possibility. As such, even if they leave one backdoor open in their application (maybe it's a $_GET query parameter and old habits of firing off a dirty query kick in), the results can be catastrophic. So it's always better to take the time to dive deeper into the foundations.

II. CONCLUSION

SQL Injection is a very nasty attack on a web application but is easily avoided. As we saw in this Paper, being careful when processing user input (by the way, SQL Injection is not the only threat that handling user input brings) and querying the database is all there is to it. That said, we aren't always working in the security of a web framework, so it's better to be aware of this type of attack and not fall for it.

REFERENCES

[1] Ava Fedorov. (2014) SC Magazine UK. [Online]. http://www.scmagazineuk.com/sql-injection-attacks-stilla-major-threat/article/347374/

[2] V. Nithya, R. Regan, and J. Vijayaraghavan, "A survey of SQL injection attacks, their Detection and Prevention techniques," International Journal of Engineering and Computer Science(IJECS), vol. 2, no. 4, April 2013.

[3] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso, "A Classification of SQL Injection Attacks and Countermeasures," IEEE, 2006.

[4] Sruthy Manmadhan and T. Manesh, "A Method of detecting SQL Injection attack to secure web applications," International journal of Distributed and Parallel Systems(IJDPS), vol. 3, no. 6, November 2012.

[5] Aanal Bhanderi and Nency Rawal, "A review and Detection mechanism for SQL injection attacks," International Journal of Innovative Research in Science, Engineering & Technology, vol. 4, no. 12, pp. 12446-12452, December 2015.

[6] Pupendra Kumar and R.K. Pateriya, "A Survey on SQL injection attacks, detection and prevention techniques," IEEE ICCCNT, 2012.

[7] Atefeh Tajpour, Suhaimi Ibrahim, and Mohammad Sharifi, "Web Applications Security by SQL Injection Detection Tools," International Journal of Computer Science Issues(IJCSI), vol. 9, no. 2, pp. 332-339, March 2012.

[8] Mahima Srivastava, "Algorithm to Prevent backend database against SQL injection attacks," IEEE, pp. 754- 757, 2014.

[9] Pankajdeep Kaur and Kanwalpreet Kaur, "SQL Injection: Study and Augmentation," International Conference on Signal Processing, Computing and Control (2015 ISPCC), 2015. [Sonakshi* et al., 5(7): July, 2016] ISSN: 2277-9655 IC™ Value: 3.00 Impact Factor: 4.116 http: // www.ijesrt.com © International Journal of Engineering Sciences & Research Technology [189]

[10] Sonakshi, Rakesh Kumar, and Girdhar Gopal, "Pprevention of SQL Injection Attacks using RC4 and Blowfish Encryption Techniques," International

Journal of Enineering Research & Technology (IJERT), vol. 5, no. 06, pp. 25-29, June 2016.

[11] Shubham Mukherjee, Sudeshna Bora, Pritam Sen, and Chittaranjan Pradhan, "SQL Injection: A Sample Review," 6th ICCCNT, Denton, U.S.A., December 2015.

[12] Chandrashekhar Sharma and Dr. S.C. Jain, "Analysis and Classification of SQL injection vulnerabilities and Attacks on Web Applications," IEEE International Conference on Advances in Engineering and Technology Research(ICAETR 2014), 2014.

[13] Debrata Kar and Suvasini Panigrahi, "Prevention of SQL Injection attack using query transformation and hashing," 3rd IEEE International Advance Computing Conference (IACC), pp. 1317-1323, 2013.

