Date of publication July 31, 2022, date of current version Aug. 12, 2022.

SQL Injection Detection and Preventive Approach for Web Applications

GJM Ariyathilake1 [1#], MHR Sandeepanie [2] ,and PL Rupasinghe [3]

[1],

Centre for Defence Research and Development, Ministry of Defence, Sri Lanka, [2]General Sir John Kotelawala Defence University, Sri Lanka,

[3]Sri Lanka Institute of Information Technology, Sri Lanka

[1#]awert1232003@gmail.com

ABSTRACT Presently, the most highly used method of global communication is web applications and used for long-distance communication, online marketing, research and development, distance learning, e-banking and social media networks. Since web applications are available for the global community with access for anyone, web applications confront numerous security issues, specifically due to web-based c yber-attacks. The SQL injection attack is the most prevailing web-based cyber-attacks globally, belonging to high-rank classifications. Because of the increased number of global online services with a high rate, SQL injection attacks also are amplified r apidly. Most SQL injection attacks are successful due to a lack of proper validation. However, a successful SQL injection attack highly interferes with databases' integrity, availability, and confidentiality. Therefore, there is a vital global requirement to overcome SQL injection attacks. Accordingly, there are three key objectives. The first objective is to detect the SQL injection attacks affecting web servers. The second objective is to explore the preventive solution for SQL injection attacks affecting the web servers. The third objective is to share the knowledge on SQL injection attacks with other researchers. Towards overcoming predominant issues, a periodically and continuously running PHP-based programme, which can identify patterns of SQL injection attacks recorded in PHP Apache log files and blocking the identified suspicious IP addresses, was designed as the adopted me thodology. Statistics of total suspicious IP addresses and black listed IP addresses with their hitting counts and time were obtained while preventing access of black listed IP addresses to the Apache webserver. The proposed solution facilitates continuous monitoring of suspicious activities while blocking vulnerable hosts using its IP addresses automatically with securing web servers from the SQL injection attack.

INDEX TERMS: Cyber-attacks, Global Communication, SQL injection attacks, Web applications.

I INTRODUCTION

The most highly used method of global communication is web applications. Web applications are used globally for long-distance communication, online marketing, health services, research and development, distance learning, e-banking and social media networks. Ever since the web applications are accessible for the global community with having access for anyone at any time, web applications have been confronted with numerous challenges comprising the security issues, precisely owing to web-based cyber-attacks. Among various cyber-attacks, the Structured Query Language (SQL) injection attack is the most prevailing web-based cyber-attacks globally, belonging to high-rank classifications. In view of that, the line of codes describe the basic SQL injection attack as follows: The statement = "select * from customers where name = '" + customerName + "';"

Above mentioned SQL code is created to pull up all the user records specified "customer name" from its table of "customers". Conversely, if the "customerName" variable is crafted and explicitly designed by one of the vulnerable users, the SQL statements may perform more than the author intended. For instance, setting the "customerName" variable using as follows:

' OR '1'='1

Alternatively, consuming comments even to block the rest of statements of the query (In here, mentioned three types of different SQL comments). All the lines have a specified space at the end of each of three statements as follows:

i. 'OR '1'='1' –
ii. ' OR '1'='1'
iii. ' OR '1'='1' /*

The above codes render one of the above mentioned SQL statements by parent language as follows:

i. select * from customers where name = '' or '1'='1';
ii. select * from customers where name = '' or '1'='1' – ';

When these codes are to be consumed in an authentication role procedure, then the above example could be utilised to force to get a selection of every field of data (*) from a customer SQL table, excluding one specified customer name, as the author intended, due to the evaluation of code '1'='1' usually is always true. The above value of

"customerName" in the statement mentioned below would cause the deletion of the "customers" table (SQL) as well as get a selection of all the data from the "customerinfo" table (in essence that revealing the information regarding every user), using user API that allows more SQL statements:
a';DROP TABLE customers; SELECT * FROM customerinfo WHERE 't' = 't

Such input renders the executing final SQL statements as follows:
select * from customers where name = 'a';drop table customers;
select * from customerinfo where 't' = 't';
To prevent SQL injection cyber-attacks, web application developers may use specific tools to check the availability and prevention of SQL injection attacks. At present, such tools are WAF (Web Application Firewall) , "Positive Tainting", "SQLrand", "CSSE", "CANDID" etc.

Nevertheless, web application security is extremely vital in preventing SQL injection attacks. The developers are subjected to numerous cyber-attacks because of improper security coding practices, particularly malicious source code injection. Further, several improper and insecure coding practices are frequently used with low encryption, which is subjected to a lack of protection. Typically, SQL injection cyber-attacks execute through inserting malicious code into a SQL query. Such malicious codes, which the cyber attackers insert, are pretended as legitimate SQL query statements. Hence, the web servers' sequential execution of such malicious codes affects the internal system and database management systems, leading to SQL injection cyber-attacks to execute improper SQL commands. Most SQL injection attacks are effective due to a deficiency of proper validation. A successful SQL injection attack vastly interferes with the databases' integrity, availability, and confidentiality. In addition, based on the research findings and general statistics and the available data on the internet, such SQL injection cyber-attacks have a severe impact on global organisations. Accordingly, a practical solution is a vital global requirement to overcome SQL injection attacks. With this view, there are three key objectives in this research. The first objective is to detect the SQL injection attacks that affect the web servers. Afterwards, the second objective is to explore the preventive solution for SQL injection attacks affecting the web servers. Finally, the third objective is to share the knowledge on SQL injection attacks with other researchers.

II LITERATURE REVIEW

At present, most people use web applications, which are accessed through World Wide Web, precisely for long distance communications, online marketing, distance learning, e-banking and social media networks. Most of the web applications are available for anyone globally without any restrictions. Because of such reasons, it is exposed to many challenges comprising more security issues cum cyber-attacks via the internet. Consequently, Lijiu (2010) revealed about the web application vulnerabilities, such as malicious file execution, cross site scripting, SQL injection and cross site request forgery, which have the connection with secure coding of web applications. Further, Mark (2006) also studied web application security vulnerabilities, including different analysis tools. Moreover, Mark (2006) identified different analysis tools such as source code analysers, Black box scanners, DB scanners, Binary analysis tools, Runtime analysis tools, Configuration analysis tools and Proxy analysis tools. Accordingly, the "MUSIC" tool checks the mutants in the SQL source code queries. Further, the tool termed "SUSHI" is used to resolve existing constraints in the strings. Moreover, another tool termed "Ardilla" is used to create SQL injection attacks and test web scenarios. In addition, the tool termed "String Analyser" is used to analyse the web strings.

In the prevailing literature, the usage of web applications with validation using cryptographic modules and increasing cyber threats related to security of web applications have been explored (Dima, 1999). In view of that, web applications are able to use the modules for password cryptography, password generating and so on (Dima, 1999). Further, Dima (1999) explored the usages connected to web application components and how they are developed overcoming the increasing cyber threats. Further, the usages related to firewalls as a way of network site protection against external intrusions and attacks were also explored in the prevailing literature. In addition, Dima (1999) explored the different components in a firewall policy such as filtering packets, proper authentication, and application gateways. Web based cyber-attacks occur as SQL injection attacks and they prevail globally and cause severe impacts with web applications. SQL injection attacks are conducted with including a segment of malicious code into SQL query via none or without proper validated environment and that will receive by web servers. It was found that there are faults regarding web applications; the most hazardous types of vulnerabilities are Cross site scripting and SQL injection attacks (Jose, 2008). It was identified the different types of issues related to web application cyber-attacks such as injection of commands, traversal of path, LDAP injection, SQL injection and Spoofing of content (Sven, 2008). Further, more critical vulnerabilities are occurred due to cross site scripting and SQL injection attacks (Jose, 2008). Moreover, Lijiu (2010) revealed that web application vulnerabilities such as malicious file execution, cross site scripting, SQL injection and cross site request forgery connect with secure coding of web applications. It was explained regarding the vulnerabilities of SQL injection attacks & cross site scripting, which caused harm

to several web applications (Andrea, 2012). There are several SQL injection detection, and prevention tools are available. Some are IDPs, Green SQL, dotDefender, Code scan labs Etc. A mole is an open-source tool for detecting SQL injection attacks. It generates reports regarding SQL injection attacks. It evaluates provided URL of clients (Pavitra Shankdhar, 2021).

Green SQL is an open-source application for detecting and preventing SQL injection attacks. It supports "My SQL" databases and evaluates SQL commands with a risk scoring matrix. It generates reports regarding the SQL injection attacks and blocks the vulnerable hosts(Ivano Alessandro Elia, 2010).

SQLsus is an open source tool for detection of SQL injection attacks. This tool can use for MySQL data bases. It is written with "PERL" computer language. This tool is fast and effiecient with detection of SQL injection attacks(Ivano Alessandro Elia, 2010). SQLMap is an open source automatic SQL injection attacks and database take over application which is used for penetration testing. This tool automates detection and exploitation of SQLi flaws.(Drew Robb, 2022) Several researchers have introduced different SQL detection and preventive solutions based on the prevailing literature. Accordingly, Rai and Nagpal (2019) studied SQL injection attacks and proposed methods and tools for detection and preventive solutions while discussion their effectiveness. Further, Singh et al. (2014) also proposed a model to block the SQL injections while analysing the existing detection prevention techniques against SQL injection attacks. Moreover, Jemal et al. (2020) also proposed solutions to mitigate SQL injection, specifically through ontology and machine learning. A differential process to safeguard against SQL injection attacks, used in ASP.NET apps, has been introduced (Kausar et al., 2019). In addition, Hu (2017) introduced a defence resistance and remedy model of SQL injection attack, established from non-intrusive SQL injection attack and defence.

III METHODOLOGY AND EXPERIMENTAL DESIGN

In achieving the study's objectives, the methodology adopted by the researchers was creating an environmental variable for the "php.exe" file as the first step. As the second step, a "bat" file for run "sql_injection_block.php" file was created. As the third step, a "task scheduler" adding "bat" file to run the "sql_injection_block.php" file continuously with appropriate time intervals was created. APACHE log files to the proposed application with the given command prompt command were linked as the final step. The adopted SQL injection attack identification and IP address blocking process are descriptively displayed (Figure 1).

Accordingly, when the user input malicious code for SQL injection attack, it will compare with SQL injection attack patterns and if the user input compares with specified patterns, then the user input attempt will take as a suspicious attempt. If the number of attempts exceeded more than the specified number of attempts, then that host IP address will be blocked automatically. All the suspicious attempts will be stored in the "suspicious_ips" file. Blocked IPs are too added to another file called "blocked_ips".

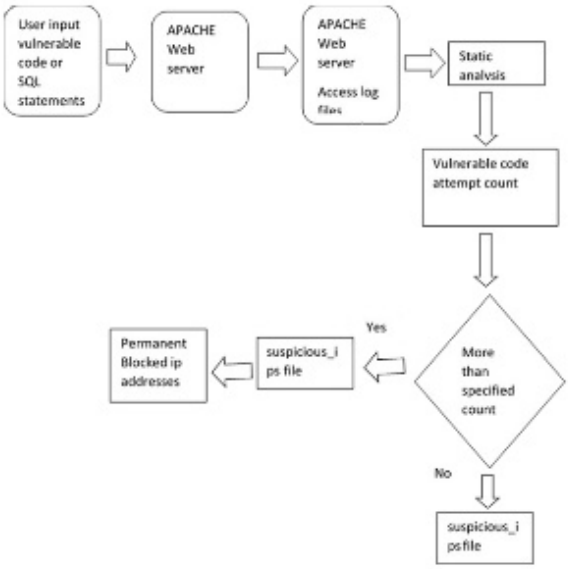

Figure 1. SQL Injection attack identification and IP address blocking process
Source: Developed by the researchers based on the research study

The proposed solution has removal facility of blocked IP addresses from the blacklisted list. User input time will also be stored in the "suspicious_ips" file, and it will be able to analyse later.

A *Access Log Analysis Methodology*

First, have to set the path to APACHE access log files in the "apache_acess.bat" file. Then it has to connect to the task scheduler, and it is required to set the interval of time that want to run reiterately. Source code files have been located and it is required to give path of the "sql_injection_block.php" with suitable parameters in the bat file. All the installation and operatable processes will be mentioned later in a detailed manner. After installation, Apache access log files will be analysed after a specified period in the task scheduler, and all the suspicious user attempts in the apache log files will be stored in the "suspicious_ips". If there are a considerable number of suspicious attempts made by a user, then the IP is automatically

blocked after exceeding previously defined value and added to the "blocked_ips" list. If it is required to remove some identified blocked IP from the blocked IP list, it will remove such IP from the blocked IP list. Such operations are mentioned in a detailed manner later. POST or GET user inputs will be analysed, and therefore any POST or GET malicious user inputs will be blocked with this solution.

B Specified SQL Injection Comparing Patterns

apacheaccesspaterns[] = "/|select[*]from|select * from|select*from|'or'1'=1|/i"
apacheaccesspaterns[] = "/or1=1|update set|insert into|delete from|/i"
apacheaccesspaterns[] = "/order by|1'1|select count([*])|1 and 1=1|/i"
apacheaccesspaterns[] ="/&#49|&#32|&#79|&#82|&#61|&#39|1 UNION ALL SELECT 1,2,3,4,5,6,name FROM sysObjects WHERE xtype = 'U' –|/i"

C Installation Process for Manual Process

This solution was designed for Windows Operating System, but later, the research will continue for Linux Operating System. This solution was designed with "XAMPP" installer. At first, it is required to install "XAMPP" software. Then it is required to set environmental variable path to PHP folder as follows; First, go to the control panel.

- First, go to the control panel.
- Then, go to "system".
- Next, go to "change setting".
- Then, go to "Advanced" tab.
- Then, go to environmental variables.
- Then, select the "Path" environmental variable (Figure 2) and go to "Edit", and click.
- Then, click new and type or copy and paste the path to the "PHP" folder (Figure), select the area and click the "ok" button.

Afterwards, it is required to locate the "sql_injection_block" folder as your preference. Then, it is required to open a command prompt and change the command prompt location to the "sql_injection_block" directory.

D Manual Operating Process

At first, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command, "php sql_injection_block.php", "php sql_injection_block.php -h" or "php sql_injection_block.php –help".

Obtaining user operating options and details option 1

```
C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php
```

Obtaining user operating options and details option 2

```
C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -h
```

Obtaining user operating options and details option 3

```
C:\xampp\htdocs\sql_injection_block> php sql_injection_block.php –help
```

1 Obtaining Statistics:

Firstly, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command,
"php sql_injection_block.php –statistics" or
"php sql_injection_block.php -s".

Obtaining statistics option 1

```
C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -s
```

Obtaining statistics option 2

```
C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php –statistics
```

When entering the above mentioned command for the first time, it will be appeared as "No data!" due to the absence of the "suspicious_ips" file. Before obtaining the statistics it is required to parse the apache log files as below figure entering command "PHP sql_injection_block.php –parse-apache-log –path=C:\xampp\apache\logs\access.log".

Initially, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command,
"php sql_injection_block.php –parse-apache-log –path=C:\xampp\apache\logs\access.log".

Parsing APACHE log files option 1

```
C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php –parse-apache-log –
path=C:\xampp\apache\logs\access.log
```

Parsing APACHE log files option 2

```
C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -a -C:\xampp\ apache\
logs\access.log
```

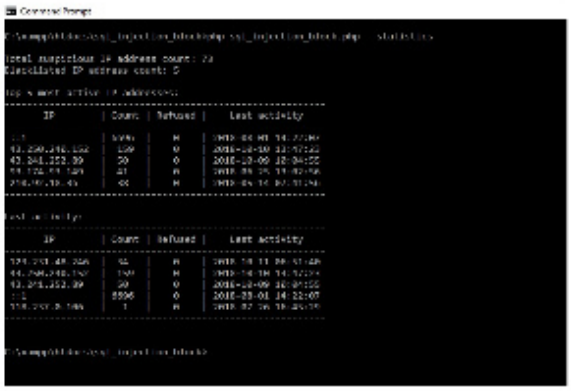

Figure 2. Obtaining statistics
Source: Developed by the researchers based on the research study

Firstly, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command, "PHP sql_injection_block.php –statistics" or"PHP sql_injection_block.php -s".

Obtaining statistics option 1

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -s

Obtaining statistics option 2

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php –statistics

After parsing APACHE access log files, it is possible to obtain the statistics (Figure 5).

2 Obtaining List of Black Listed IP Addresses:

Initially, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command, "PHP sql_injection_block.php –list" or"PHP sql_injection_block.php -l".

Obtaining black listed IP addresses option 1

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -l

Obtaining black listed IP addresses option 2

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php –list

When entering the above mentioned command for the first time, it is appeared as "No data!" due to absence of the "suspicious_ips" file. Before obtaining statistics, it is required to parse the apache log files as in below (Figure 3) entering command "PHP sql_injection_block.php –parse-apache-log –path=C:\xampp\apache\logs\access.log".

Firstly, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command, "php sql_injection_block.php –parse-apache-log –path=C:\xampp\apache\logs\access.log". –path=C:\xampp\apache\logs\access .log".

Figure 3. Obtaining blacklisted IP addresses
Source: Developed by the researchers based on the research study

Firstly, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command, "PHP sql_injection_block.php –list" or"PHP sql_injection_block.php -l".

Obtaining black listed IP addresses option 1

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -l

Obtaining black listed IP addresses option 2

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php–list

After parsing APACHE access log files, it is possible to get blacklisted IP addresses.

3 Obtaining List of Black Listed IP Addresses with Suspicious Activity Count:

Firstly, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command,"PHP sql_injection_block.php–list –count" or"PHP sql_injection_block.php -l -c".

Obtaining black listed IP addresses with suspicious count option 1

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -l -c

Obtaining black listed IP addresses with suspicious count option 2

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php–list –count

When entering the above mentioned command for the first time, it is appeared as "No data!" due to absence of

the "suspicious_ips" file. Before obtaining statistics it is required to parse the apache log files as below figure entering command "PHP sql_injection_block.php –parse-apache-log –path=C:\xampp\apache\logs\access.log".

4 *Obtaining Black Listed IPs with Suspicious Activity Time:*

Initially, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command "PHP sql_injection_block.php –list –time" or"php sql_injection_block.php -l -t".

Obtaining black listed IPs with suspicious activity time option 1

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -l -t

Obtaining black listed IPs with suspicious activity time option 2

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php –list –time

When enter the first time above mentioned command, then will get "No data!" due to absence of the "suspicious_ips" file. Before obtaining statistics, have to parse the apache log files as below figure entering command "PHPsql_injection_block.php–parse-apache-log–path=C:\xampp\apache\logs\access.log".

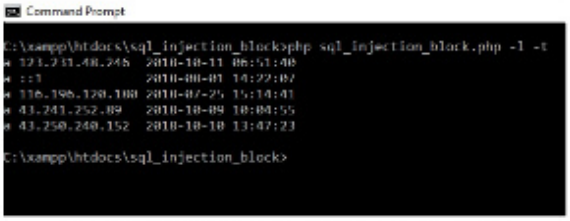

Figure 4. Blacklisted IPs with last activity time
Source: Developed by the researchers based on the research study

5 *Obtaining Black Listed IPs with Suspicious Activity Count and Time:*

Initially, it is required to take the command prompt location to "sql_injection_block" directory location and enter the command, "PHP sql_injection_block.php –list –count –time" or"php sql_injection_block.php -l -c -t".

Obtaining black listed IPs with suspicious activity count and time option 1

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -l -c -t

Obtaining black listed IPs with suspicious activity count and time option 2

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php –list –count –time

When entering the above mentioned command for the first time, it is appeared as "No data!" due to absence of the "suspicious_ips" file. Before obtaining statistics, have to parse the apache log files as below figure entering command "PHP sql_injection_block.php –parse-apache-log –path=C:\xampp\apache\logs\access.log".

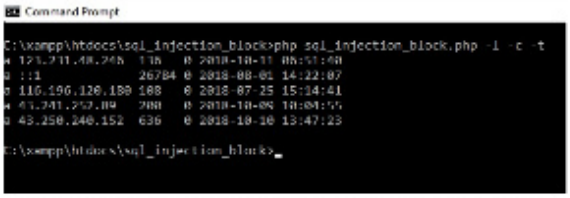

Figure 5. Parsing APACHE access log files obtaining blacklisted IPs with suspicious activity count and time
Source: Developed by the researchers based on the research study

6 *Removing Black Listed IP Addresses and Adding to White List:*

Removing black listed IP option 1

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php -r123.231.48.246

Removing black listed IP option 2

C:\xampp\htdocs\sql_injection_block>
php sql_injection_block.php –romove=123.231.48.246

Figure 6. Removing blacklisted IPs
Source: Developed by the researchers based on the research study

E *Installation Process for Automated Process*

This solution was designed for Windows Operating System and later research will be continued for Linux Operating System. This solution was designed with "XAMPP" installer and. At first, it is required to install "XAMPP" software.

1 Setting the Environmental Variable Path to PHP Folder:

Setting the environmental variable path to PHP folder as follows;

1. First, go to the control panel.
2. Then, go to "system".
3. Next, go to "change setting".
4. Then, go to "Advanced" tab.
5. Then, go to environmental variables.

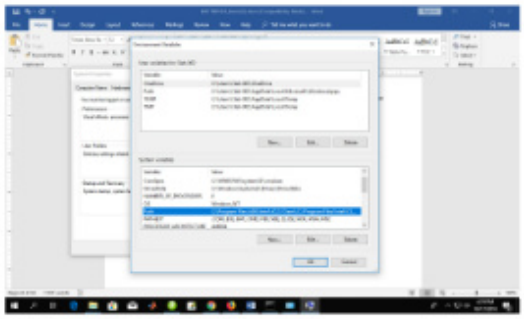

Figure 7. Environmental variables
Source: Developed by the researchers based on the research study

6. Then, select the "Path" environmental variable as in the above "Figure 10" and go to "Edit" and click.
7. Then, click new and type or copy and paste the path to the "PHP" folder as in the below figure, selected area and click the "ok" button.
8. Create the "sql_injection_block.bat" file as in below (Figure 8).

Figure 8. sql_injection_block.bat file
Source: Developed by the researchers based on the research study

In here, "cd <sql_injection_block directory path> "PHP <path to the apache access log file>" are inserted.

9. Then locate the "sql_injection_block.bat" file in the sql_injection_block directory.

2 Adding the Bat File to the "Task Scheduler":

1. Go to the start menu and type "control panel" and click it.
2. Then, go to "Administrative tools".
3. Then, go to "Task schedular".
4. Create new task"sql_injection_block". It is required to set triggering settings for at least thirty minutes and repeat activity after every thirty minutes and it is required to make sure not to set run multiple processes. Then, it is required to set settings as Queue.
5. Then run the task "sql_injection_block".

F IP Addresses Blocking Process

After the vulnerable IP addresses are detected, the identified IP addresses will be added to the "suspicious_ips" file. Then, that suspicious IP address will be added to the ".htaccess" file for access denied. When it is required to remove a blocked IP address, then the IP address will be removed from the ".htaccess" file.

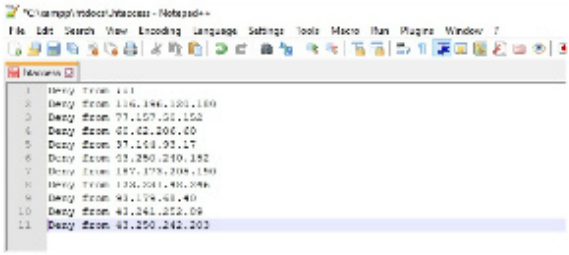

Figure 9. htaccess file inside
Source: Developed by the researchers based on the research study

G Performance Analysis and Evaluation of the Current System

When the user requests and inputs malicious codes or any input that caused to SQL injection attack or any valid user purposes, it will be compared with SQL injection primitive attack patterns and then user requests and inputs will be compared with specified patterns in the proposed system. As well as, if such user requests are matched with specified malicious patterns in the proposed system, such user input attempts will be taken as suspicious attempts, and the IP address of such attempts coming will be taken as the suspicious IP address. If several such attempts are exceeded more than a specified number of malicious attempts, then that host IP address will be blocked automatically. All the suspicious attempts will be stored in the "suspicious_ips" file. Blocked IPs added to another file called "blocked_ips". If it is required to remove the blocked IP address from the list, this solution has a facility to do that. It was

explained earlier. User input times will also be stored in the "suspicious_ips" file, which will be able to analyse later.

As the first step, it is required to set the path to APACHE access log files in the "apache_acess.bat" file. Then it is required to connect to the task scheduler, and it is required to set the interval of time that want to run reiterately. Source code files have to be located and it is required to give path of the "sql_injection_block.php" with suitable parameters in the bat file. All the installation and operatable processes will be mentioned later in a detailed manner. After installing the Apache access log files, it will be analysed after the specified period in the task scheduler, and all the suspicious user attempts in the apache log files will be stored in the "suspicious_ips". If user suspicious attempts are more than the specified count of the source code, then that user will be blocked automatically and added to the "blocked_ips" list. If it is required to remove some identified blocked IP from the blocked IP list, it will remove such IP from the blocked IP list. Such operations were mentioned in a detailed manner earlier with commands. POST or GET user inputs will be analysed, and therefore any POST or GET malicious user inputs will be blocked with this solution. After processing of the "suspicious_ips" file, if suspicious pattern, matching count is exceeded the specified count in the proposed system, then such IP addresses will be added to the ".htaccess" file as "deny access ¡IP address¿". Then that IP address will be blocked for external users for internet access.

Figure 10. Blacklisted IP addresses
Source: Developed by the researchers based on the research study

The detailed results are descriptively elaborated under the section of Results.

IV RESULTS

Under this section, the statistics of the user requests are explained. The result issuing command, namely, "—statistics", the most active top five addresses termed 127.0.0.1, 43.250.240.152, 93.174.93.149, 103.242.0.73 and 103.45.9.123 were obtained. The recorded occurrence of the IP address of 127.0.0.1 was 254448. The recorded occurrence of the IP address of 43.250.240.152 was 6042. The recorded occurrence of the IP address of 93.174.93.149 was 1558. The recorded occurrence of the IP address of 103.242.0.73 was 1444. The recorded occurrence of the IP address of 103.45.9.123 was 1444.

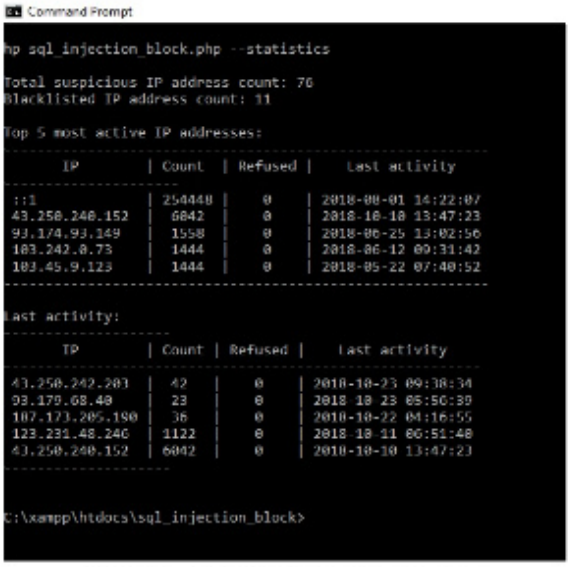

Figure 11. Analysed user request statistics
Source: Developed by the researchers based on the research study

The analysed and processed statistics of user requests, which the users requested, are descriptively displayed (Figure 11). The counted malicious attempts and the top five IP addresses are descriptively displayed in Figure 12. Further, the last activity time figures also are displayed. The last activity recorded date and time for IP address 127.0.0.1 was 2018-08-01 at 14:22:07. The last activity recorded date and time for IP address 43.250.240 was 2018-10-10 at 13:47:23. The last activity recorded date and time for IP address 93.174.93.149 was 2018-06-25 at 13:02:56. The last activity recorded date and time for IP address 103.242.0.73 was 2018-06-12 at 09:31:42. The last activity recorded date and time for IP address 103.45.9.123 was 2018-05-22 at 07:40:52. According to the second table of Figure 16, the last five IP addresses with the last activity details are displayed.

A Listing of Black Listed IP Addresses

According to figure 16, they were obtained using the "–list" command in the console. IP address blacklist happened due to the host trying for vulnerable patterns as HTTP requests several times. After exceeding the predefined maximum count, IP addresses were blacklisted as vulnerable IP addresses. The results of the listing of blacklisted IP addresses is descriptively displayed.

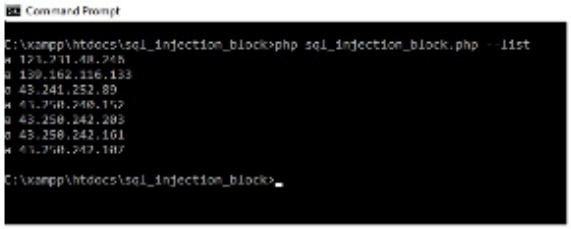

Figure 12. Listing of blacklisted IP addresses
Source: Developed by the researchers based on the research study

The above mentioned "Listing of blacklisted IP addresses". Listing of blacklisted IP addresses that user requests are coming from. The IP addresses mentioned in "Figure 11" shows requested vulnerable requests more than the specified vulnerable attempt count in the proposed solution. After entry of the statement termed, "Deny from <IP address>" to the ".htaccess" file, accessing the webserver was blocked for that specific IP address. "403 forbidden" Error occurred after that host tried to access again. The blacklisted IP addresses are 123.231.48.246, 139.162.116.133, 43.241.252.89, 43.250.240.152, 43.250.242.203 43.250.242.161, and 43.250.242.107 were received after analysing of apache access.log file. If it is required to remove some IP addresses from the blacklisted list, then it will not appear in the blacklisted IP address list and that IP address will be able to access the webserver continuously without any hindrance. Then, IP details of the "suspicious_IPs" file will be updated stored in the "suspicious_IPs" file. "–remove = < IP address >" command used to remove IP address from the blacklisted IP address list. After analysing apache access.log files, these blacklisted IP address details will be stored in the "suspicious_IPs" file and then later also could be able to analyse and will be able to get the backup copies. When using the "–list" command other details such as; blacklisted time, suspicious occurrences count, last activity time like such details regarding that IP address will not be displayed, and only the IP address will be displayed. If such details are required, then it is required to enter other commands and that commands will be explained in a detailed manner later.

B Listing of Black Listed IP Addresses with Suspicious Attempt Count

According to figure 18, the results of listing blacklisted IP addresses with a count of vulnerable activities tried as HTTP requests are descriptively shown. When using the "–list –count" command other details such as; blacklisted time, last activity time like such details regarding that IP address will not be displayed and only IP address with a count of occurrences of vulnerable activities as HTTP requests will be displayed. If such details are required, it is required to enter other commands, which will be explained later. After issuing the "–list –count" command, blacklisted IP addresses with vulnerable activity count is shown in figure 17, "5.3 listing of blacklisted IP addresses with suspicious attempt count".

Figure 13. Listing of blacklisted IP addresses with suspicious attempt counts
Source: Developed by the researchers based on the research study

Above mentioned "Listing of blacklisted IP addresses with suspicious attempt count" provides the listing of blacklisted IP addresses that user requests with suspicious attempts count in front of them. In here 123.231.48.246, 139.162.116.133, 43.241.252.89, 43.250.240.152, 43.250.242.203, 43.250.242.161, 43.250.242.107 were the blacklisted IP addresses. The blacklisted IP address 123.231.48.246 was recorded with 225 vulnerable activity counts. The blacklisted IP address 139.162.116.133 was recorded with 11 vulnerable activity counts. The blacklisted IP address 43.241.252.89 was recorded with 254 vulnerable activity counts. The blacklisted IP address 43.250.240.152 was recorded with 260 vulnerable activity counts. The blacklisted IP address 43.250.242.203 was recorded with 140 vulnerable activity counts. The blacklisted IP address 43.250.242.161 was recorded with 76 vulnerable activity counts. The blacklisted IP address 43.250.242.107 was recorded with 1136 vulnerable activity counts. After analysing apache access.log files, these blacklisted IP address details were stored in the "suspicious_IPs" file.

There is a PHP function called "parseFile" in the Apacheaccesslogparser.php file and within that function, new IP details were added to the "suspicious_IPs" file. When issuing the command "–list –count", these details were taken from the "suspicious_IPs" file. When using the "–list –count" command other details such as; blacklisted time, last activity time like such details regarding that IP address were not displayed and only blacklisted IP addresses with vulnerable activity count were displayed. If such details are required, then it is necessary to enter other commands and that commands will be explained in a detailed manner well ahead.

C Listing of Black Listed IP Addresses with Last Suspicious Attempt Time

The results of the blacklisted IP addresses with the last activity time is displayed in figure 19. The results were obtained using the "–list –time" command in console. After

analysing apache access.log files these blacklisted IP addresses and other details will be stored in the "suspicious_IPs" file, and when issuing the command "–list –time", then these details will be taken from the "suspicious_IPs" file.

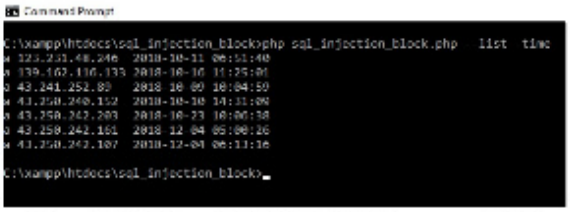

Figure 14. Listing of blacklisted IP addresses with last suspicious attempt times
Source: Developed by the researchers based on the research study

The above mentioned "Listing of black listed IP addresses with last suspicious attempt time" (Figure 14) shows the listing of black listed IP addresses that user requests coming from with their suspicious last attempted time in front of them.

In here, 123.231.48.246, 139.162.116.133, 43.241.252.89, 43.250.240.152, 43.250.242.203, 43.250.242.161, 43.250.242.107 were the blacklisted IP addresses. The blacklisted IP address 123.231.48.246 was recorded with the last vulnerable activity date and time as 2018-10-11 at 06:51:40. The blacklisted IP address 139.162.116.133 was recorded with the last vulnerable activity date and time as 2018-10-16 at 11:25:01. The blacklisted IP address 43.241.252.89 was recorded with the last vulnerable activity date and time as 2018-10-09 at 10:04:59. The blacklisted IP address 43.250.240.152 was recorded with the last vulnerable activity date and time as 2018-10-10 at 14:31:09. The blacklisted IP address 43.250.242.203 was recorded with the last vulnerable activity date and time as 2018-10-23 at 10:06:38. The blacklisted IP address 43.250.242.161 was recorded with the last vulnerable activity date and time as 2018-12-04 at 05:00:26. The blacklisted IP address 43.250.242.107 was recorded with the last vulnerable activity date and time as 2018-12-04 at 06:13:16. These details were added to the "suspicious_IPs" file from the "$ipInfo" array. The new IP details were added to the "$ipInfo" array within the "Apacheaccesslogparser.php" file. A PHP function called "parseFile" was included there and within that function, new IP details were added to the "suspicious_IPs" file.

D Listing of Black Listed IP Addresses with Suspicious Attempt Count and Last Suspicious Attempt Time

The result of listing blacklisted IP addresses with the last activity time and count of suspicious activities are shown below (Figure 15). That results were obtained using the "–list–count –time" command in console.

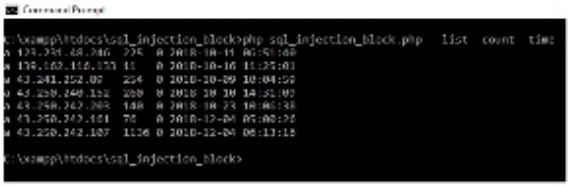

Figure 15. Listing of blacklisted IP addresses with suspicious attempt count and last suspicious attempt time
Source: Developed by the researchers based on the research study

Above mentioned "Listing of blacklisted IP addresses with suspicious attempt count and last suspicious attempt time" with the listing of blacklisted IP addresses that user requests coming from with their suspicious last attempted time and suspicious attempt count in front of them. In here 123.231.48.246, 139.162.116.133, 43.241.252.89, 43.250.240.152, 43.250.242.203, 43.250.242.161, 43.250.242.107 were the blacklisted IP addresses. The blacklisted IP address 123.231.48.246 was recorded with the last vulnerable activity date and time as 2018-10-11 at 06:51:40 and the count of vulnerable activities as 225. The blacklisted IP address 139.162.116.133 was recorded with the last vulnerable activity date and time as 2018-10-16 at 11:25:01 and the count of vulnerable activities as 11.

The blacklisted IP address 43.241.252.89 was recorded with the last vulnerable activity date and time as 2018-10-09 at 10:04:59 and the count of vulnerable activities as 254. The blacklisted IP address 43.250.240.152 was recorded with the last vulnerable activity date and time as 2018-10-10 at 14:31:09 and the count of vulnerable activities as 260. The blacklisted IP address 43.250.242.203 was recorded with the last vulnerable activity date and time as 2018-10-23 at 10:06:38 and the count of vulnerable activities as 140. The blacklisted IP address 43.250.242.161 was recorded with the last vulnerable activity date and time as 2018-12-04 at 05:00:26 and the count of vulnerable activities as 76.

The blacklisted IP address 43.250.242.107 was recorded with the last vulnerable activity date and time as 2018-12-04 at 06:13:16 and the count of vulnerable activities as 1136. After analysing apache access.log files these blacklisted IP addresses and other details were stored in the "suspicious_IPs" file, and when issuing the command "–list –count –time", then these details were taken from the "suspicious_IPs" file.

E Apache Access Log File Analysis

The results of parsing Apache access.log file analysis is displayed below (Figure 16). That results were obtained using

the "–parse-apache-log –path = <path to the Apache access.log file>" command in console. In here, "suspicious IP addresses before processing: 76" means, before parsing Apache access.log file for the processing which was previously stored suspicious IP addresses count in the "suspicious_IPs" file is 76. When single suspicious activity was encountered from an IP address, then that IP address was taken as a suspicious IP address. Further, it became a blacklisted IP address when exceeding the predefined suspicious activity count.

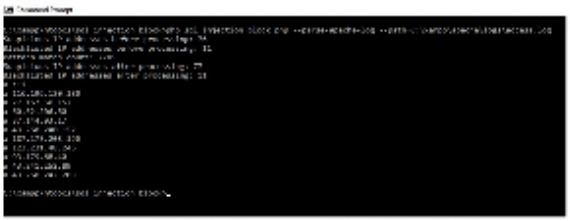

Figure 16. Apache access log file analysis
Source: Developed by the researchers based on the research study

Above mentioned "Apache access log file analysis" in "Figure 16" shows the listing of blacklisted IP addresses that user requests coming from with suspicious IP addresses count before processing, Blacklisted IP addresses count before processing, Total vulnerable pattern match count, suspicious IP addresses count after processing, Blacklisted IP addresses count after processing.

Here, "Blacklisted IP addresses before processing was 11" means, before parsing Apache access.log file for processing. Previously stored blacklisted IP addresses count in the "suspicious_IPs" file is 11. Here total vulnerable pattern match count was 7785. Here "suspicious IP addresses after processing: 77" means, after parsing Apache access.log file for processing total stored suspicious IP addresses count in the "suspicious_IPs" file is 77 and new one suspicious IP address added to the "suspicious_IPs" file after parsing the Apache access.log file for processing.

Here "Blacklisted IP addresses after processing was 11" means that after parsing Apache access.log file for processing, the total stored blacklisted IP addresses count in the "suspicious_IPs" file was 11. It means no new blacklisted IP address was added to the "suspicious_IPs" file.

F Removing Blacklisted IP Address

The removal of blacklisted IP addresses is shown below (Figure 17). That results were obtained using the "–remove = <IP address>" command in console. After removing the blacklisted IP address, it was stored in the "suspicious_IPs" file.

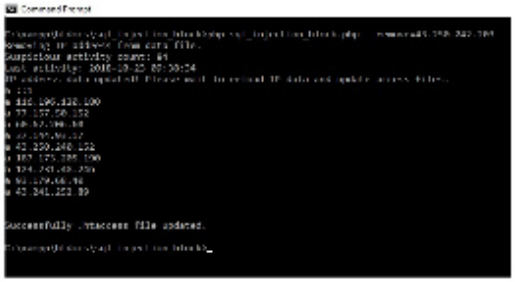

Figure 17. Removing blacklisted IP address
Source: Developed by the researchers based on the research study

Figure 17 shows removing blacklisted IP addresses and after removing that IP address, all suspicious activity count of that IP address, last activity time of that IP address. All blacklisted IP addresses are listed hereafter, removing the specified IP address. When removing some IP addresses from the blacklisted IP address list, it did not appear in the blacklisted IP address list and that IP address was able to access the webserver continuously without any hindrance. Then IP details of the "suspicious_IPs" file were updated and stored in the "suspicious_IPs" file then; later can be analysed and will be able to get backup copies. The command "–remove = < IP address > " was used to remove an IP address from the blacklisted IP address list. Removing blacklisted IP addresses will do from handling ".htaccess" file. In here, ".htacess" was used to block vulnerable hosts, adding "Deny from <ipaddress>" code inside it, and this code will be added to every vulnerable blacklisted IP address to block the server access. Then it will be given a "403 Forbidden" error to the vulnerable host preventing access to the server. After removing the blacklisted IP address from the blacklisted list, then "Deny from <ipaddress>" entry will be removed from the ".htaccess" file for the relevant removed IP address.

G Test an Evaluation of Final Host IP Address Blocking

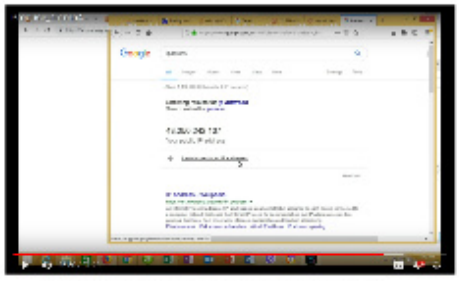

Figure 18. Host public IP address
Source: Developed by the researchers based on the research study

Above Figure 18 shows the tested vulnerable host public IP address (43.250.242.107).

Figure 19. Blacklisted IP addresses
Source: Developed by the researchers based on the research study

Above figure 15 shows the black listed vulnerable host public IP addresses. Above figure 19 shows that the public IP address (43.250.242.107) did not belong to the black listed IP addresses after removing the public IP address (43.250.242.107) from black listed IP addresses list in figure 20.

Vulnerable host (public IP address (43.250.242.107)) was trying to access a web server (public IP address (43.250.242.107)) with vulnerable user inputs "'or'1'=1" continuously and after the exceeding of maximum count of vulnerable accesses IP address, 43.250.242.107 added to the blacklisted IP address list.

Figure 20. Trying to access web Server with vulnerable codes
Source: Developed by the researchers based on the research study

Above figure 20 shows the IP address, 43.250.242.107 added to the black listed IP address list.

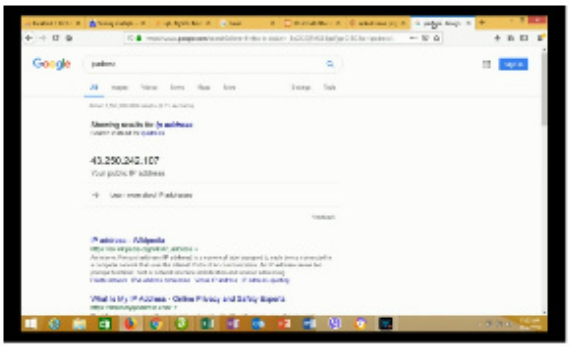

Figure 21. Trying to access web Server after vulnerable host blacklisted
Source: Developed by the researchers based on the research study

Above figure 21 shows web results when trying to access web server after vulnerable host (IP address 43.250.242.107) got blacklisted with a legitimate URL.

V DISCUSSION AND CONCLUSION

The proposed solution for SQL injection prevention facilitates the continuous monitoring of suspicious activities. Conferring to this proposed solution, there is no requirement for the user to be concerned about monitoring IP address blocking activities in web applications. Further, the proposed solution automatically blocks the vulnerable hosts using its IP address. Moreover, the proposed solution facilitates a listing of blocked IP addresses if the user needs to remove some IP addresses from the blacklisted IP address list. As well as, the user could be able to customise the blocked IP address list according to his will. Further, this proposed solution facilitates the user to view the last activity time of the suspicious IP addresses with the suspicious activity count; then, the user will compare each of the suspicious IP addresses. In view of that, all the suspicious activities will be stored in a file, including suspicious activity time and activity count; then, the user will be able to process later or analyse such details further, and such data backups are also able to take. However, the proposed solution is designed mainly for "Windows" operating systems and have to install "XAMPP" or "WAMP" software, which is freely available on the internet. The proposed solution is composed of a set of vulnerable user HTTP request patterns & it is recommended to add more vulnerable user HTTP request patterns. Then the user faithfulness to the proposed system will be increased. Further, it is recommended to use XAMPP version 7 or above. Finally, the proposed solution is recommended for "Windows 7" or above.

REFERENCES

[1] S.W. Booyd, and A.D. Keromytiss, "SQL Rand: Preventing SQL injection attacks," Colombia University, Available: https://www1.cs.columbia.edu/~angelos/Papers/sqlrand.pdf, [Accessed May 5, 2018].

[2] G. Buehrer, B. Weide, and P. Sivilotti, "Using parse tree validation to prevent SQL injection attacks," Research Gate, Available: Error! Hyperlink reference not valid. 221215947_Using_parse_tree_validation_to_prevent_SQL_injection_attacks [Accessed June 5, 2018].

[3] S. Christensen, A. Moller, and M. S. Precise, Analysis of String Expressions. Berlin, Germany: Springer, 2003, pp. 1-50.

[4] S. Faker, M. Muslim, and H. Dachlan, "A Systematic Literature Review on SQL Injection Attacks Techniques and Common Exploited Vulnerabilities," International Journal of Computer Engineering and Information Technology, vol. 9, 2017, Available: http://www.ijceit.org/ published/ volume9/ issue12/ 2Vol9No12.pdf [Accessed Jan. 26, 2018].

[5] W. Halfond, and A. Orso, Malware Detection. Boston, USA: Springer, 2007, pp. 86.

[6] E. Janot, and P. Zavarsky, "Preventing SQL Injections in Online Applications: Study, Recommendations and Java Solution Prototype Based on the SQL DOM," Research Gate, 2008, Available: file:///C:/Users/CRD/Downloads/2008_OWASP_AppSec_Preventing_SQL_injections_in_online_applications.pdf [Accessed Mar. 15, 2018].

[7] I. Jemal, O. Cheikhrouhou, H. Hamam, and H. Mahfoudhi, "SQL Injection Attack Detection and Prevention Techniques Using Machine Learning," International Journal of Applied Engineering Research, vol. 15, 2020, pp. 569-580.

[8] M.A. Kausar, M. Nasar, and A. Moyaid, "SQL Injection Detection and Prevention Techniques in ASP.NET Web Application," International Journal of Recent Technology and Engineering, vol. 3, 2019, pp. 7759-7766.

[9] H. Kaur and S. Dhingra, "A Review: Prevent SQL Injection Attacks Using IPS," International Journal of Advanced Research in Computer and Communication Engineering, vol. 3, 2014, pp. 8124-8126, Available: https://ijarcce.com/wp-content/uploads/2014/10/IJARCCE1I-a-amit-harpreet1-A-Review-Prevent-SQL-Injection-Attacks-Using-IPS.pdf [Accessed August 07, 2018].

[10] H. Mehta, "Threat Intelligence.," Symantec enterprise blogs security, 2018 [online] Available: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/microsoft-patch-tuesday-november-2018 [Accessed Nov. 15, 2018].

[11] F. Mavituna, (2008). Deep Blind SQL Injection. Portcullis Security, 2008 [online] p.A11. Available: Error! Hyperlink reference not valid. / [Accessed Aug. 20, 2018].

[12] A. Makiou, Y. Begriche, and A. Serhrouchni, "Hybrid Approach to Detect SQLi Attacks and Evasion Techniques. HAL archives, 2015, Available: https://hal.archives-ouvertes.fr/hal-01138604/document [Accessed Oct. 10, 2018].

[13] R. Muhammad, S. Habib and R. Bashir, "Detection and Prevention of SQL Injection Attack by Dynamic Analyser and Testing Model," Research Gate, 2017, Available: https://www.researchgate.net/publication/319453593_Detection_and_Prevention_of_SQL_Injection_Attack_by_Dynamic_Analyzer_and_Testing_Model [Accessed Nov. 02, 2018].

[14] S. Rai, and B. Nagpal, "Detection and Prevention of SQL Injection Attacks: Developments of the Decade," 3rd International Conference on Reliability, Infocom Technologies and Optimisation (ICRITO) (Trends and Future Directions), AIIT, Amity University Uttar Pradesh, Noida, India, 2014.

[15] J. Singh, "Analysis of SQL Injection Detection Techniques," 2017, Available: Error! Hyperlink reference not valid. 1605.02796.pdf [Accessed Jun. 08, 2018].

[16] S. Singh, U. Tripathi, and M. Mishra, "Detection and Prevention of SQL Injection AttackUsing Hashing Technique. International Journal of Modern Communication Technologies and Research (IJMCTR), [online], vol. 2, 2014, Available: https://www.academia.edu/9378445/Detection_and_Prevention_of_SQL_Injection_Attack_Using_Hashing_Technique [Accessed Aug. 22, 2018].

[17] F. Valeur, D. Mutz, and G. Vigna, "A Learning-Based Approach to the Detection of SQL Attacks," Research Gate, 2005, Available: Error! Hyperlink reference not valid. 225239186_A_Learning-Based_Approach_to_the_Detection_of_SQL_Attacks [Accessed Jul. 15, 2018].

[18] O. Voitovych, and L. Kupershtein, "SQL injection prevention system," Research Gate, 2016, Available: https://www.researchgate.net/publication/310454603_SQL_injection_prevention_system [Accessed Aug. 15, 2018].

[19] D. Robb, "Best SQL Injection (SQLi) Detection Tools 2022," Serverwatch, 2022, Available: https://www.serverwatch. com/reviews/sql-injection-detection-tools/ [Accessed Mar. 08, 2022].

[20] P. Shankdhar, "Best free and open source SQL injection tools [updated 2021]," INFOSEC, 2021, Available: https:// resources. Infosecinstitute .com / topic / best-free – and – open – source – sql - injection-tools/ [Accessed Mar. 05, 2022].

ACKNOWLEDGMENT

We immensely thank all the professionals who supported in developing this noteworthy proposed system for SQL injection detection and prevention in various web applications. Further, we greatly thank all the researchers in the fields of Cyber Security and Software Engineering who contributed greatly to enhancing the pool of literature, which helped us in order to succeed in our creation.

AUTHOR BIOGRAPHY/IES

GJM Ariyathilake BSc(hons) in IT, MSc in IT (Specialization in Cyber Security) is working as Research Officer at Centre for Defence Research and Development

Mrs MHR Sandeepanie MBA, BSc(Special)(Hons), National Dip. Training & HRD, National Dip. HRM, IPICT(Denmark) is working as Senior Assistant Registrar at General Sir John Kotelawala Defence University and presently reading for PhD in Management at University of Sri Jayewardenepura.

Dr PL Rupasinghe PhD (Curtin University of Technology, Australia), MBA (PIM, USJ), BSc(Hons) is working as Senior Lecturer at Sri Lanka Institute of Information Technology.