

SQL Injection Detection and Prevention Solution for Web Applications

GJM Ariyathilake^{1#}, MHR Sandeepanie² and PL Rupasinghe³

¹*Centre for Defence Research and Development, Sri Lanka*

²*General Sir John Kotelawala Defence University, Sri Lanka*

³*Sri Lanka Institute of Information Technology, Sri Lanka*

#awert1232003@gmail.com

Abstract— Presently, the most highly used method of global communication is web applications. It is used for long-distance communication, online marketing, health services, research and development, distance learning, e-banking and social media networks. Since web applications are available for global community with access for anyone at any time, web applications are confronted with numerous challenges that comprise of security issues, specifically owing to web-based cyber-attacks. The SQL injection attack is the most prevailing global web-based cyber-attack, and it belongs to high rank classifications. Because of the increased number of global online services with a high rate of cyber-attacks, SQL injection attacks also are amplified rapidly. Most of the SQL injection attacks are successful, due to lack of proper validation. However, a successful SQL injection attack highly interferes with integrity, availability and confidentiality of the data in the databases. Therefore, there is a vital global requirement to overcome SQL injection attacks. Towards overcoming predominant issues, a periodically and continuously running PHP based programme, which is able to identify patterns of SQL injection attacks recorded in PHP Apache log files, and is capable to block the identified suspicious IP addresses was designed as the adopted methodology. In this empirical research, statistics of total suspicious IP addresses and blacklisted IP addresses with their hitting counts and time were obtained, while preventing access of blacklisted IP addresses to the Apache web server. The proposed solution facilitates for continuous monitoring of suspicious activities, while blocking vulnerable hosts using its IP addresses automatically with securing web servers from the SQL injection attack.

Keywords: *SQL injection attacks, web applications, communication*

I. INTRODUCTION

Right now, the most highly used method of global communication are 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 confront with numerous challenges comprising the security issues, specifically owing to web based cyber-attacks. Among various cyber-attacks, the Structured Query Language (SQL) injection attack is the most prevailing web based cyber-attack globally, which belongs to high rank classifications. In view of that, the line of codes describe the basic SQL injection attack is as follows:

The statement = "select * from customers where name = '" + customerName + "';"

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

' OR '1'='1

or consuming comments even to block the rest statements of the query (In here, mentioned 3 types of different SQL comments). All the lines

have a specified space in the end of the 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 above example could be utilized to force to get selection of every field of data (*) from customers SQL table, excluding one specified customer name as the author intended, due to the evaluation of code '1'='1' is normally always true. The above value of "customerName" in the statement mentioned below, would cause to deletion of the "customers" table (SQL) as well as get 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 for check availability and prevention of SQL injection attacks. At present, such tools are WAF (Web Application Firewall), "Positive Tainting", "SQLrand", "CSSE", "CANDID" etc.

Nevertheless, the web application security is extremely vital in preventing SQL injection attacks. Because of the improper security coding practices, the developers are subjected to numerous cyber-attacks, particularly with malicious source code injecting cyber-attacks. Further, several improper and insecure coding practices are frequently used with low encryption, which are subjected to the lack of

protection. SQL injection cyber-attacks conduct with segment of malicious code into SQL query through none or without proper validated environment and that will be received by the web servers. Such malicious codes, which are inserted by the cyber attackers are pretend as the legitimate SQL query statements. Hence, sequential execution of such malicious codes by the web servers affect to the internal system and database management systems, which leads to SQL injection cyber-attacks in order to execution of improper SQL commands. Most of the SQL injection attacks are effective due to deficiency of proper validation. Though, a successful SQL injection

attack vastly interferes with integrity, availability and confidentiality of the data in the data bases. In addition, based on the research findings and prevailing statistics, as well as based on the available data in the internet, such SQL injection cyber-attacks have a serious impact with global organizations. Accordingly, there is a vital global requirement to overcome SQL injection attacks with an effective solution. With this view, there are three key objectives in this research. The first objective is to detect the SQL injection attacks affect to the web servers. Afterwards, the second objective is to explore the preventive solution for SQL injection attacks affect to the web servers. Finally, the third objective is to share the knowledge on SQL injection attacks with other researchers.

II. LITERATURE REVIEW

At present, majority of 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. Amongst the web applications, the most of them are available for anyone globally without any restrictions. Because of such reasons, it is exposed to confront with many challenges comprising more security issues cum cyber-attacks via 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 regarding security

vulnerabilities related to web applications including different types of analysis tools. Moreover, Mark (2006) identified different types of 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 tool termed "MUSIC" tool is used to check 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 to test the 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 as well as how they develop overcoming increasing cyber threats. Further, the usages related to firewalls as a way of network site protection against external intrusions and attacks also were explored in the prevailing literature. Moreover, it was identified about the explorations of components, which are basically included in a firewall policy including filtering of packets, proper authentication and application gateways (Dima, 1999).

Among the web based cyber-attacks, which are occurred as SQL injection attacks, prevail globally and cause serious impacts with web applications. SQL injection attacks conduct with including 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, the 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, which have the connection with secure coding of web applications. It was explained regarding vulnerabilities of SQL injection attacks & cross site scripting which caused harm to a number of web applications (Andrea, 2012).

Based on the prevailing literature, several researchers have explored and introduced different SQL detection and preventive solutions. Accordingly, Rai and Nagpal (2019) studied on 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 the solutions to mitigate SQL injection, specifically through ontology and machine learning. A differential process to safeguard against SQL injection attacks, which is 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, which is established from the perspective of non-intrusive SQL injection attack and defence.

III. METHODOLOGY AND EXPERIMENT

In achieving the objectives of the study, the methodology adopted by the researchers was creating an environmental variable for "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. As the final step, APACHE log files to the proposed application with the given command prompt command was linked. The adopted method of SQL injection attack identification ip address blocking process is descriptively displayed (Figure 1).

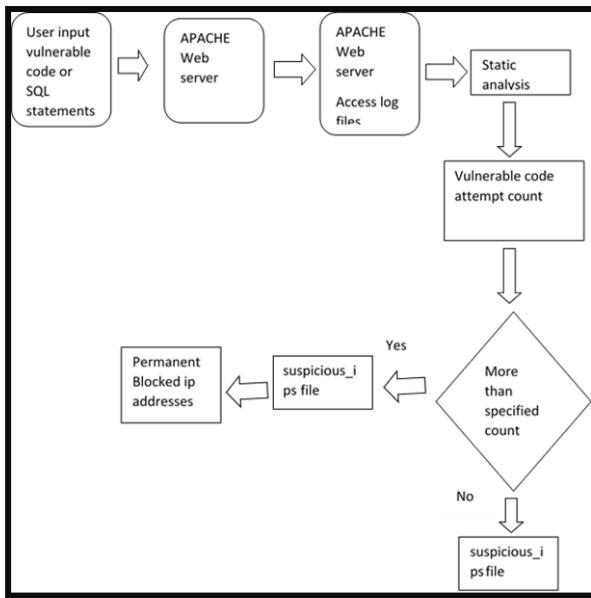


Figure 1. SQL Injection attack identification IP address blocking process

Source: Developed by the researchers based on the research study

Accordingly, when the user input malicious code or any input for SQL injection attack or any purpose, then it will compare with SQL injection attack patterns and if the user input compares with specified patterns, then that user input attempt will take as suspicious attempt. If such number of attempts exceeded more than 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 too added to another file called "blocked_ips". If it is required to remove blocked IP address from blocked IP addresses list, then

this solution has a facility to do that. User input time also will be stored in the "suspicious_ips" file and it will be able to analyze later too.

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 reiterate. Source code files have located and it is required to give path of the "sql_injection_block.php" with suitable parameters in the bat file. All the installation process and operatable process will be mentioned later in detailed manner. After installation Apache access log files will be analyzed after specified time 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 more than specified count of the source code, then that user will be blocked automatically and added to "blocked_ips" list. If it is required to remove some identified blocked ip from blocked ip list, then it will be able to remove such ip from blocked ip list. Such operations are mentioned in detailed manner later. POST or GET user inputs will be analyzed and therefore any POST or GET malicious user inputs will be blocked with this solution.

B. Specified SQL Injection Comparing Patterns

```

apacheaccesspatterns[] = "/|select[\\*]from|select
\\* from|select\\*from|'or'1'=1|i"
apacheaccesspatterns[] = "/or1=1|update
set|insert into|delete from|i/i"
apacheaccesspatterns[] = "/order by|1'1|select
count([\\*])|1 and 1=1|i"
apacheaccesspatterns[]
="/&#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 be continued for Linux Operating System too. 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 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 2), select area and click "ok" button.

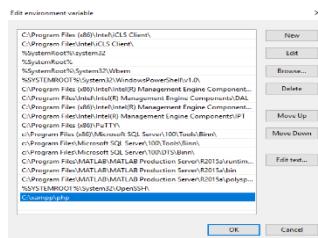


Figure 2. New Environmental Variable for “PHP” Folder

Source: Developed by the researchers based on the research study

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

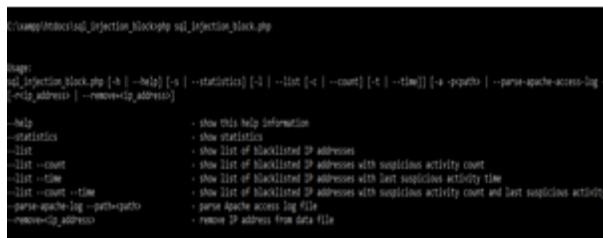


Figure 3. Obtaining user operating options
Source: Developed by the researchers based on the research study

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 at the first time, it will be appeared as “No data!” due to the absence of “suspicious_ips” file. Before obtaining the statistics it is required to parse the Apache log files as below Figure 4 entering command “php sql_injection_block.php --parse-apache-log --path=C:\xampp\apache\logs\access.log”

```
--
```

```
path=C:\xampp\apache\logs\access.log
```

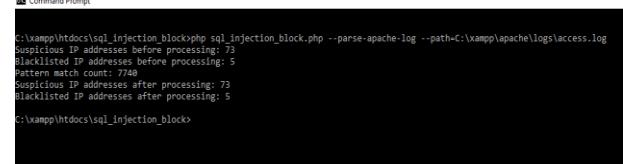


Figure 4. Parsing APACHE access log files

Source: Developed by the researchers based on the research study

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
```

```
C:\ Command Prompt
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --statistics
Total suspicious IP address count: 73
Blacklisted IP address count: 5
Top 5 most active IP addresses:
IP | Count | Refused | Last activity
--:1 | 6696 | 0 | 2018-08-01 14:22:07
43.250.240.152 | 159 | 0 | 2018-10-10 13:47:23
43.241.252.89 | 59 | 0 | 2018-10-09 10:04:55
93.174.93.149 | 41 | 0 | 2018-06-25 13:02:56
210.92.18.36 | 38 | 0 | 2018-05-14 07:41:56

Last activity:
IP | Count | Refused | Last activity
--:1 | 34 | 0 | 2018-10-11 06:51:40
123.231.48.246 | 159 | 0 | 2018-10-10 13:47:23
43.250.240.152 | 59 | 0 | 2018-10-09 10:04:55
43.241.252.89 | 6696 | 0 | 2018-08-01 14:22:07
118.237.0.106 | 38 | 0 | 2018-07-26 16:45:29

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

Figure 5. 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 in the first time, it is appeared as “No data!” due to absence of “suspicious_ips” file. Before obtaining statistics, it is required to parse the Apache log files as in below (Figure 6) 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”.

C:\ Command Prompt

```
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --list
a 123.231.48.246
a ::1
a 116.196.120.180
a 43.241.252.89
a 43.250.240.152

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

Figure 6. Obtaining black listed 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 black listed 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 mention command for the first time, it is appeared as “No data!” due to absence of “suspicious_ips” file. Before obtaining statistics, it is required to parse the Apache log

files as below Figure 7 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 you enter first time above mention command, then you will get as “No data!” due to absence of “suspicious_ips” file. Before obtaining statistics, you have to parse the Apache log files as below Figure 7 entering command “php sql_injection_block.php --parse-apache-log --path=C:\xampp\apache\logs\access.log”.

```
C:\ Command Prompt
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php -l -t
a 123.231.48.246 2018-10-11 06:51:40
a ::1 2018-08-01 14:22:07
a 116.196.120.180 2018-07-25 15:14:41
a 43.241.252.89 2018-10-09 10:04:55
a 43.250.240.152 2018-10-10 13:47:23
C:\xampp\htdocs\sql_injection_block>
```

Figure 7. Black listed 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
-t
```

When entering the above mention command for the first time, it is appeared as “No data!” due to absence of “suspicious_ips” file. Before obtaining statistics, you have to parse the Apache log files as below Figure 8 entering command “php sql_injection_block.php --parse-apache-log --path=C:\xampp\apache\logs\access.log”.

```
C:\ Command Prompt
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php -l -c -t
a 123.231.48.246 136 0 2018-10-11 06:51:40
a ::1 26784 0 2018-08-01 14:22:07
a 116.196.120.180 108 0 2018-07-25 15:14:41
a 43.241.252.89 200 0 2018-10-09 10:04:55
a 43.250.240.152 636 0 2018-10-10 13:47:23
C:\xampp\htdocs\sql_injection_block>
```

Figure 8. Parsing APACHE access log files obtaining black listed 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 --remove=123.231.48.246
```

```
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php -r123.231.48.246
Removing IP address from data file.
Suspicious activity count: 68
Refused SSH authorization count: 0
Last activity: 2018-10-11 06:51:40
IP address data updated! Please wait for daemon to reload IP data and update access files if needed.
C:\xampp\htdocs\sql_injection_block>
```

Figure 9. Removing black listed 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 too. 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;

- i. First, go to control panel.
- ii. Then, go to "system".
- iii. Next, go to "change setting".
- iv. Then, go to "Advanced" tab.
- v. Then, go to environmental variables.

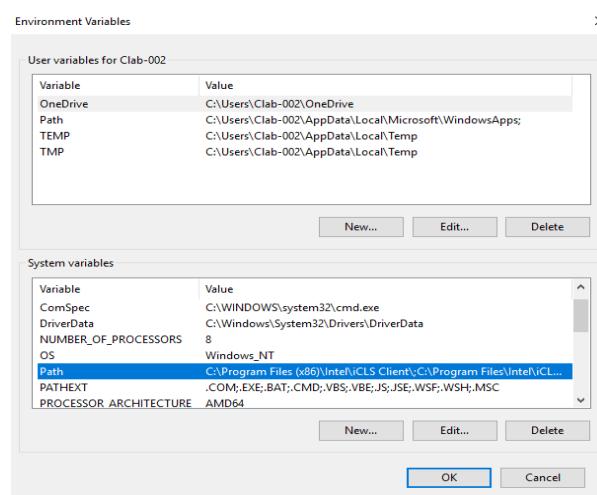


Figure 10. Environmental variables

Source: Developed by the researchers based on the research study

- vi. Then, select the "Path" environmental variable as in the above "Figure 10" and go to "Edit" and click.
- vii. Then, click new and type or copy and paste the path to the "PHP" folder as in the below Figure 11, selected area and click "ok" button.
- viii. Create "sql_injection_block.bat" file as in below (Figure 11).

```
sql_injection_block.bat - Notepad
File Edit Format View Help
ECHO OFF
cd C:\xampp\htdocs\sql_injection_block
php C:\xampp\htdocs\sql_injection_block\sql_injection_block --parse-apache-log --path=C:\xampp\apache\logs\access.log
```

Figure 11. 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.

- ix. Then locate the "sql_injection_block.bat" file in the sql_injection_block directory.

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

- i. Go to start menu and type "control panel" and click it.
- ii. Then, go to "Administrative tools".
- iii. Then, go to "Task scheduler".
- iv. Create new task "sql_injection_block".

It is required to set triggering settings at least thirty minutes and repeat activity after every thirty minutes and it is required to make sure not to set run multiple processes. The, it is required to set settings as Queue.

- v. Then run the task "sql_injection_block".

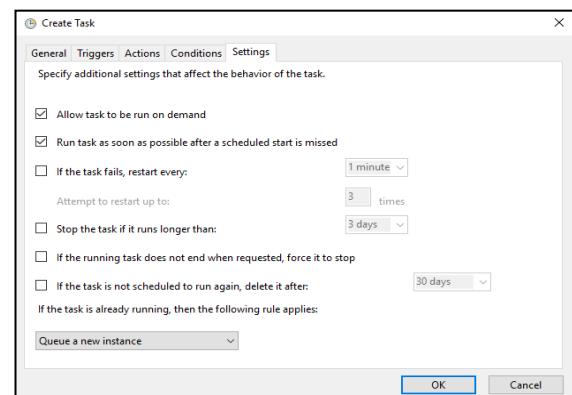


Figure 12. Queuing in Task Scheduler

Source: Developed by the researchers based on the research study

F. IP Addresses Blocking Process

After detection of the vulnerable IP addresses, 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 deny. When it is required to remove blocking IP address, then IP address will be removed from the “.htaccess” file.

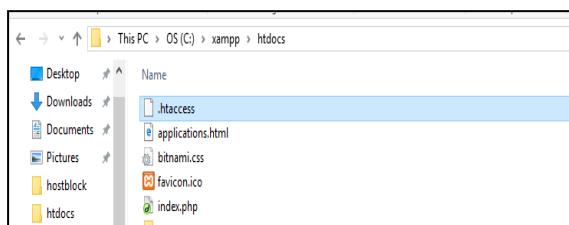


Figure 13. .htaccess file

Source: Developed by the researchers based on the research study

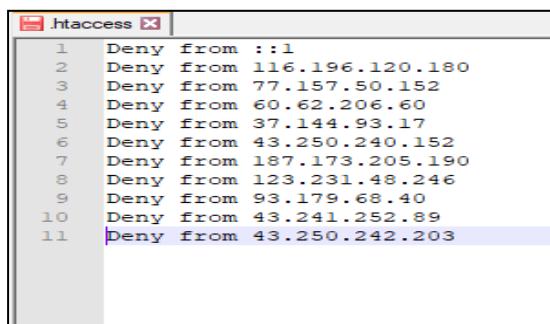


Figure 14. .htaccess file inside

Source: Developed by the researchers based on the research study

G. Performance Analysis and Evaluation of the Current System

When user requests and inputs malicious codes or any input that caused to SQL injection attack or any user valid purposes, then 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, then such user input attempts will be taken as suspicious attempt and the IP address such attempts coming will be

taken as the suspicious IP address. If such number of attempts are exceeded more than 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 too are added to another file called “blocked_ips”. If it is required to remove the blocked IP address from blocked IP addresses list, then this solution has a facility to do that. It was explained earlier. User input times also will be stored in the “suspicious_ips” file and it will be able to analyse later too.

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 iteratively. 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 process and operatable process will be mentioned later in detailed manner. After installation of the Apache access log files, it will be analysed after specified time 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 specified count of the source code, then that user will be blocked automatically and added to “blocked_ips” list. If it is required to remove some identified blocked IP from blocked IP list, then it will be able to remove such ip from blocked IP list. Such operations mentioned in 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 the internet access.



```

C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --list
b 139.162.116.133
b 43.241.252.89
b 43.250.240.152
b 43.250.242.203
b 43.250.242.161

```

Figure 15. Blacklisted IP addresses

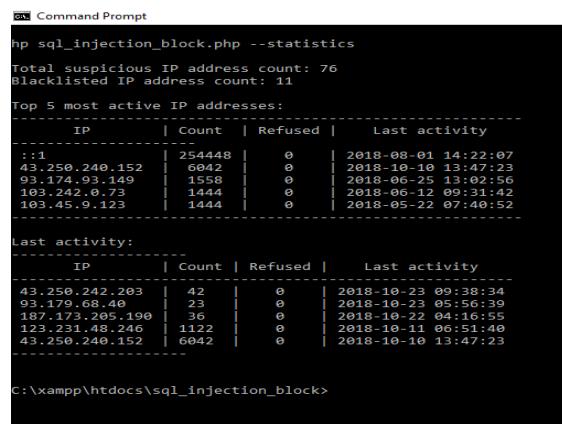
Source: Developed by the researchers based on the research study

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

IV. RESULTS

Under this section, the statistics of the user requests are explained. Through 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.



```

C:\ Command Prompt
hp sql_injection_block.php --statistics
Total suspicious IP address count: 76
Blacklisted IP address count: 11
Top 5 most active IP addresses:
-----+-----+-----+-----+
IP | Count | Refused | Last activity
-----+-----+-----+-----+
::1 | 254448 | 0 | 2018-08-01 14:22:07
43.250.240.152 | 6042 | 0 | 2018-10-18 13:47:23
93.174.93.149 | 1558 | 0 | 2018-06-25 13:02:56
103.242.0.73 | 1444 | 0 | 2018-06-12 09:31:42
103.45.9.123 | 1444 | 0 | 2018-05-22 07:40:52

Last activity:
-----+-----+-----+-----+
IP | Count | Refused | Last activity
-----+-----+-----+-----+
43.250.242.203 | 42 | 0 | 2018-10-23 09:38:34
93.179.68.40 | 23 | 0 | 2018-10-23 05:56:39
187.173.205.190 | 36 | 0 | 2018-10-22 04:16:55
123.231.48.246 | 1122 | 0 | 2018-10-11 06:51:40
43.250.240.152 | 6042 | 0 | 2018-10-10 13:47:23

C:\xampp\htdocs\sql_injection_block>

```

Figure 16. Analysed user request statistics

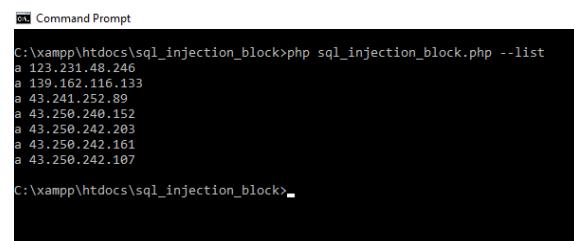
Source: Developed by the researchers based on the research study

The analysed and processed statistics of user requests, which were requested by the users, are descriptively displayed (Figure 16). The counted malicious attempts and the most top five IP addresses are descriptively displayed in Figure 16. 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, last five IP addresses with the last activity details are displayed.

A. Listing of Black Listed IP Addresses

The results according to Figure 16 were obtained by using “--list” command in the console. IP address blacklist was happening due to the host trying for vulnerable patterns as http requests in several times. After exceeding of the predefined maximum count, IP addresses were blacklisted as vulnerable IP addresses. The results of listing of black listed IP addresses is descriptively displayed (Figure 17).



```

C:\ Command Prompt
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --list
a 123.231.48.246
a 139.162.116.133
a 43.241.252.89
a 43.250.240.152
a 43.250.242.203
a 43.250.242.161
a 43.250.242.107
C:\xampp\htdocs\sql_injection_block>

```

Figure 17. Listing of blacklisted IP addresses

Source: Developed by the researchers based on the research study

Above mentioned “Listing of blacklisted IP addresses” in the “Figure 17” shows the Listing of blacklisted IP addresses that user requests coming from. From such IP addresses mentioned in the “Figure 15” shows requested vulnerable requests more than 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 was occurred after that host tried to access again. The blacklisted IP addresses such as; 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 be not

appeared in the blacklisted IP address list and that IP address will be able to access the web server continuously without any hindrance. Then, IP details of “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 “--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 it is required such details then it is required to enter other commands and that commands will be explained in 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 count of vulnerable activities tried as http requests are descriptively shown. When using “--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 count of occurrences of vulnerable activities as http requests will be displayed. If it is required such details then it is required to enter other commands and that commands will be explained in detail later. After issuing “--list --count” command, blacklisted IP addresses with vulnerable activity count is shown in Figure 17, “5.3 listing of black listed IP addresses with suspicious attempt count”.

```
Command Prompt
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --list --co
a 123.231.48.246 225 0
a 139.162.116.133 11 0
a 43.241.252.89 254 0
a 43.250.240.152 260 0
a 43.250.242.203 140 0
a 43.250.242.161 76 0
a 43.250.242.107 1136 0

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

Figure 18. Listing of black listed IP addresses with suspicious attempt count

Source: Developed by the researchers based on the research study

Above mentioned “Listing of black listed IP addresses with suspicious attempt count” in “Figure 17” shows the listing of blacklisted IP addresses that user requests coming from with their 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 count of 225 vulnerable activity counts. The blacklisted IP address 139.162.116.133 was recorded with count of 11 vulnerable activity counts. The blacklisted IP address 43.241.252.89 was recorded with count of 254 vulnerable activity counts. The blacklisted IP address 43.250.240.152 was recorded with count of 260 vulnerable activity counts. The blacklisted IP address 43.250.242.203 was recorded with count of 140 vulnerable activity counts. The blacklisted IP address 43.250.242.161 was recorded with count of 76 vulnerable activity counts. The blacklisted IP address 43.250.242.107 was recorded with count of 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 command “--list --count” then these details were taken from “suspicious_IPs” file. When using “--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 explain in detailed manner well ahead.

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

The results of listing of black listed IP addresses with last activity time is displayed in the Figure 19. The results were obtained using “--list --time” command in the console. After analysing Apache access.log files these blacklisted IP address and other details will be stored in the “suspicious_IPs” file and when issuing command “--list --time”, then these details will be taken from “suspicious_IPs” file.

```
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --list -time
a 123.231.48.246 2018-10-11 06:51:40
a 139.162.116.133 2018-10-16 11:25:01
a 43.241.252.89 2018-10-09 10:04:59
a 43.250.240.152 2018-10-10 14:31:09
a 43.250.242.203 2018-10-23 10:06:38
a 43.250.242.161 2018-12-04 05:00:26
a 43.250.242.107 2018-12-04 06:13:16

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

Figure 19. Listing of black listed IP addresses with last suspicious attempt time

Source: Developed by the researchers based on the research study

Above mentioned “Listing of black listed IP addresses with last suspicious attempt time” (Figure 19) 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 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 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 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 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 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 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 last vulnerable activity date and time as 2018-12-04 at 06:13:16. These details were added to the “suspicious_IPs” file from “\$ipInfo” array. The new IP details were added to the “\$ipInfo” array with in “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 black listed IP addresses with last activity time and count of suspicious activities are shown in below (Figure 20). That results were obtained using “--list--count --time” command in the console.

```
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --list -count -time
a 123.231.48.246 225 0 2018-10-11 06:51:40
a 139.162.116.133 11 0 2018-10-16 11:25:01
a 43.241.252.89 254 0 2018-10-09 10:04:59
a 43.250.240.152 260 0 2018-10-10 14:31:09
a 43.250.242.203 140 0 2018-10-23 10:06:38
a 43.250.242.161 76 0 2018-12-04 05:00:26
a 43.250.242.107 1136 0 2018-12-04 06:13:16

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

Figure 20. Listing of black listed 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 black listed IP addresses with suspicious attempt count and last suspicious attempt time” in the “Figure 19” shows 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 last vulnerable activity date and time as 2018-10-11 at 06:51:40 and count of vulnerable activities as 225. The blacklisted IP address 139.162.116.133 was recorded with last vulnerable activity date and time as 2018-10-16 at 11:25:01 and count of

vulnerable activities as 11. The blacklisted IP address 43.241.252.89 was recorded with last vulnerable activity date and time as 2018-10-09 at 10:04:59 and count of vulnerable activities as 254. The blacklisted IP address 43.250.240.152 was recorded with last vulnerable activity date and time as 2018-10-10 at 14:31:09 and count of vulnerable activities as 260. The blacklisted IP address 43.250.242.203 was recorded with last vulnerable activity date and time as 2018-10-23 at 10:06:38 and count of vulnerable activities as 140. The blacklisted IP address 43.250.242.161 was recorded with last vulnerable activity date and time as 2018-12-04 at 05:00:26 and count of vulnerable activities as 76. The blacklisted IP address 43.250.242.107 was recorded with last vulnerable activity date and time as 2018-12-04 at 06:13:16 and count of vulnerable activities as 1136. After analysing Apache access.log files these blacklisted IP address and other details were stored in the “suspicious_IPs” file and when issuing command “--list --count --time”, then these details were taken from “suspicious_IPs” file.

E. Apache Access Log File Analysis

The results of parsing Apache access.log file analysis is displayed below (Figure 21). That results were obtained using “--parse-apache-log --path = <path to the Apache access.log file>” command in the console. In here “suspicious IP addresses before processing: 76” means, before parsing Apache access.log file for processing which was previously stored suspicious IP addresses count in the “suspicious_IPs” file is 76. When single suspicious activity encountered from an IP address, then that IP address was taken as suspicious IP address. Further, it was become as blacklisted IP address when exceeding the predefined suspicious activity count.



```
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --parse-apache-log --path=C:\xampp\apache\logs\access.log
Suspicious IP addresses before processing: 76
Blacklisted IP addresses before processing: 11
Pattern match count: 7785
Suspicious IP addresses after processing: 77
Blacklisted IP addresses after processing: 11
a ::1
a 116.96.120.180
a 77.157.59.152
a 58.53.298.68
a 37.144.93.37
a 43.250.240.152
a 187.173.205.190
a 123.231.48.246
a 93.179.68.48
a 43.241.252.89
a 43.250.242.203

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

Figure 21. Apache access log file analysis

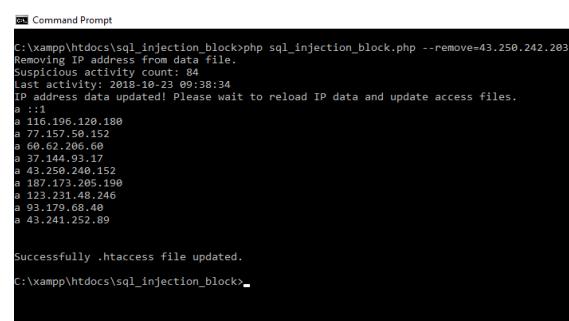
Source: Developed by the researchers based on the research study

Above mentioned “Apache access log file analysis” in the “Figure 21” 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.

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

F. Removing Blacklisted IP Address

The results of removing blacklisted IP addresses is shown below (Figure 22). That results were obtained using “--remove = <IP address>” command in the console. After removing blacklisted IP address, then it was stored in the “suspicious_IPs” file.



```
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --remove=43.250.242.203
Removing IP address from data file.
Suspicious activity count: 84
Last activity: 2018-10-23 09:38:34
IP address updated! Please wait to reload IP data and update access files.
a ::1
a 116.96.120.180
a 77.157.59.152
a 60.62.206.60
a 37.144.93.17
a 43.250.240.152
a 187.173.205.190
a 123.231.48.246
a 93.179.68.48
a 43.241.252.89

Successfully .htaccess file updated.
C:\xampp\htdocs\sql_injection_block>
```

Figure 22. Removing blacklisted IP address

Source: Developed by the researchers based on the research study

Figure 22 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 listed here after removing the specified IP address. When removing of some IP address from the blacklisted IP address list, then it was not appeared in the blacklisted IP address list and that IP address was able to access the web server continuously without any hindrance. Then IP details of “suspicious_IPs” file was updated and stored in the “suspicious_IPs” file, then later too can be analysed and will be able to get backup copies. The command “--remove = < IP address >” was used to remove IP address from the blacklisted IP address list. Removing blacklisted IP address will do from handling “.htaccess” file. In here “.htaccess” was used to block vulnerable hosts adding “Deny from <ipaddress>” code inside it and this code will be added to each and every vulnerable blacklisted IP address to block the server access. Then it will be given “403 Forbidden” error to vulnerable host preventing access to the server. After removing the black listed IP address from the black listed 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

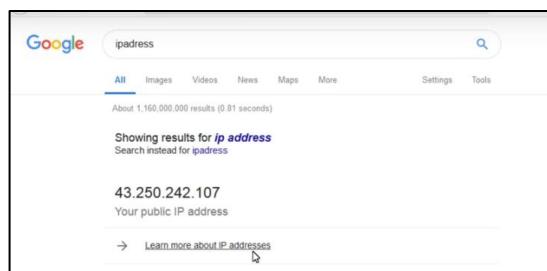
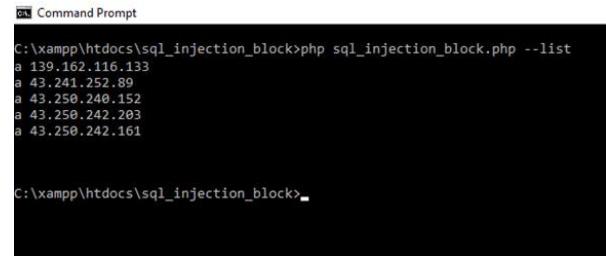


Figure 23. Host public IP address

Source: Developed by the researchers based on the research study

Above Figure 23 shows the tested vulnerable host public IP address (43.250.242.107).



```
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --list
a 139.162.116.133
a 43.241.252.89
a 43.250.240.152
a 43.250.242.203
a 43.250.242.161

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

Figure 24. Blacklisted IP addresses

Source: Developed by the researchers based on the research study

Above Figure 20 shows the black listed vulnerable host public IP addresses. Above Figure 24 shows public IP address (43.250.242.107) wasn't belong to the black listed IP addresses after removing public IP address (43.250.242.107) from black listed IP addresses list of Figure 20

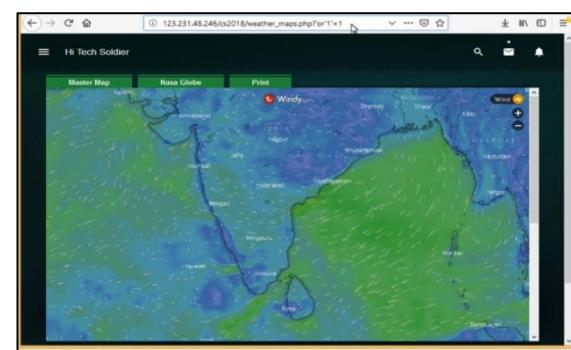
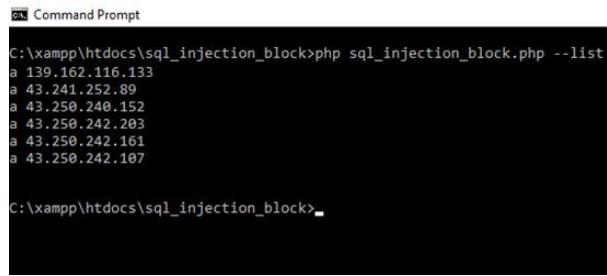


Figure 25. Trying to access web Server with vulnerable codes

Source: Developed by the researchers based on the research study

Above Figure 25 shows vulnerable host (public IP address (43.250.242.107)) was trying to access 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 black listed IP address list.



```
C:\xampp\htdocs\sql_injection_block>php sql_injection_block.php --list
a 139.162.116.133
a 43.241.252.89
a 43.250.240.152
a 43.250.242.283
a 43.250.242.161
a 43.250.242.107

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

Figure 26. Trying to access web Server with vulnerable codes

Source: Developed by the researchers based on the research study

Above Figure 26 shows the IP address, 43.250.242.107 added to the black listed IP address list.

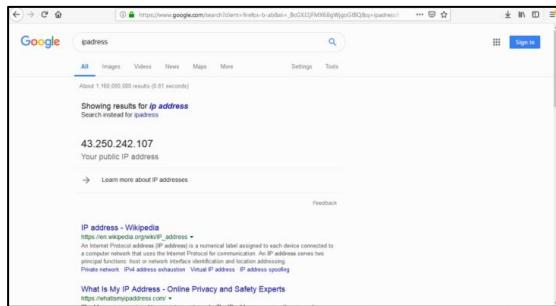


Figure 27. Trying to access web Server after vulnerable host black listed

Source: Developed by the researchers based on the research study

Above Figure 27 shows web results when trying to access web server after vulnerable host (IP address 43.250.242.107) got blacklisted with legitimate URL.

V. DISCUSSION AND CONCLUSION

The proposed solution for SQL injection prevention facilitates for the continuous monitoring of suspicious activities. Conferring to this proposed solution, there is no requirement for the user to concern about monitoring or else 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 for listing of blocked IP addresses if the user needs to remove some IP address from the black listed IP address list. As well as, the user

could be able to customize 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 be able to compare each of suspicious IP addresses. In view of that, all the suspicious activities will be stored in a file including suspicious activity time, suspicious activity count, then the user will be able to later process or analyze such details further and such data backups 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 in the Internet. 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

- Boyd, S., Keromytis, A. and Rand, S.O.L. (2018) Preventing SQL Injection Attacks, *Columbia University*, Available at: <https://www1.cs. columbia. edu/~angelos/ Papers/ sqrlrand.pdf> [Accessed 05 May. 2018].
- Buehrer, G., Weide, B. and Sivilotti, P. (2005) Using parse tree validation to prevent SQL injection attacks, *Research Gate*, Available at: https://www.researchgate.net/publication/221215947_Using_parse_tree_validation_to_prevent_SQL_injection_attacks [Accessed 05 Jun. 2018].
- Christensen, S., Moller, A. And Schwartzbach, M. (2003) *Precise Analysis of String Expressions*. Berlin, Germany: Springer, PP.1-50.
- Cisco. (2014) A Review: Prevent SQL Injection Attacks Using IPS. *International Journal of Advanced Research in Computer and Communication Engineering*, [online] Volume 3, Available at: <https://ijarcce.com/wp-content/uploads/2014/10/IJARCCE1I-a-amit-harpreet1-A-Review-Prevent-SQL-Injection-Attacks-Using-IPS.pdf> [Accessed 07 Aug. 2018].
- Faker, S., Muslim, M. and Dachlan, H. (2017) A Systematic Literature Review on SQL Injection Attacks Techniques and Common Exploited Vulnerabilities.

International Journal of Computer Engineering and Information Technology, [online] Volume 9, Available at: <http://www.ijceit.org/published/volume9/issue12/2Vol9No12.pdf> [Accessed 26 Jan. 2018].

Halfond, W. and Orso, A. (2007). *Malware Detection*. Boston, USA: Springer, P. 86.

Janot, E. and Zavarsky, P. (2008). Preventing SQL Injections in Online Applications: Study, Recommendations and Java Solution Prototype Based on the SQL DOM. *ResearchGate*, Available at: file:///C:/Users/CRD/Downloads/2008_OWASP_App_Sec_Preventing_SQL_injections_in_online_applications.pdf [Accessed 15 Mar. 2018].

Jemal, I., Cheikhrouhou, O., Hamam, H. and Mahfoudhi (2020). SQL Injection Attack Detection and Prevention Techniques Using Machine Learning. *International Journal of Applied Engineering Research*, 15 (6), pp. 569-580.

Kaur, H. and Dhingra, S. (2014). A Review: Prevent SQL Injection Attacks Using IPS. *International Journal of Advanced Research in Computer and Communication Engineering*, [online] Volume 3, Available at: <https://ijarcce.com/wp-content/uploads/2014/10/IJARCCE1I-a-amit-harpreet1-A-Review-Prevent-SQL-Injection-Attacks-Using-IPS.pdf> [Accessed 07 Aug. 2018].

Kausar, M.A., Nasar, M. and Moyaid, A. (2019). SQL Injection Detection and Prevention Techniques in ASP.NET Web Application. *International Journal of Recent Technology and Engineering (IJRTE)*, 8 (3), pp. 7759-7766.

Mehta, H. (2018). *Threat Intelligence*. [online] Symantec enterprise blogs security. Available at: <https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/microsoft-patch-tuesday-november-2018> [Accessed 15 Nov. 2018].

Mavituna, F. (2008). Deep Blind SQL Injection. *Portcullis Security*, [online] p.A11. Available at: <https://labs.portcullis.co.uk/whitepapers/> [Accessed 20 Aug. 2018].

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

Muhammad, R., Habib, S. and Bashir, R. (2017). Detection and Prevention of SQL Injection Attack by Dynamic Analyzer and Testing Model. *ResearchGate*, Available at: https://www.researchgate.net/publication/319453593_Detection_and_Prevention_of_SQL_Injection_Attack_by_Dynamic_Analyzer_and_Testing_Model [Accessed 02 Nov. 2018].

Pooja. and Monika. (2016). SQL Injection: Detection and Prevention Techniques. *International Journal of Scientific & Engineering Research*, [online] Volume 7, Available at: <https://www.ijser.org/researchpaper/SQL-Injection-Detection-and-Prevention-Techniques.pdf> [Accessed 02 Sep. 2018].

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

Singh, J. (2016). Analysis of SQL Injection Detection Techniques. *arXiv*, Available at: <https://arxiv.org/ftp/arxiv/papers/1605/1605.02796.pdf> [Accessed 08 Jun. 2018].

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

Valeur, F., Mutz, D. and Vigna, G. (2005). A Learning-Based Approach to the Detection of SQL Attacks. *ResearchGate*, Available at: https://www.researchgate.net/publication/225239186_A_Learning-Based_Approach_to_the_Detection_of_SQL_Attacks [Accessed 15 Jul. 2018].

Voitovych, O. and Kupershtain, L. (2016). SQL injection prevention system. *ResearchGate*, Available at: https://www.researchgate.net/publication/310454603_SQL_injection_prevention_system [Accessed 03 Aug. 2018].

ACKNOWLEDGMENT

We immensely thank to 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 to all the researchers in the fields of Cyber Security and Software Engineering who contributed greatly to enhance the pool of literature, which helped us in order to succeed our creation.

AUTHOR BIOGRAPHIES



Major GJM Ariyathilake is presently working as the Commissioned Officer of Sri Lanka Army (Software Engineer) at Centre for Defence Research and Development. Major Ariyathilake has completed MSc in IT (Specialization in Cyber Security) and BSc (Hons) from Sri Lanka Institute of Information Technology (SLIIT).



Mrs MHR Sandeepanie is presently working as the Senior Assistant Registrar at General Sir John Kotelawala Defence University. Mrs Sandeepanie is presently reading for PhD in Management at University of Sri Jayewardenepura. She has completed MBA (Kelaniya), BSc (Special) (Hons), National Dip. Training & HRD (IPM), National Dip. HRM (NIBM), IPICT (UNIC, Denmark). Mrs Sandeepanie is a Member of IMSL and an Associate Member of SLITAD.



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