Employing Neural Networks for the Detection of SQL Injection Attack

Naghmeh Moradpoor Sheykhkanloo

Employing Neural Networks for the Detection of SQL Injection Attack

Naghmeh Moradpoor Sheykhkanloo
School of Science, Engineering, and Technology (SET)
University of Abertay Dundee
Dundee, United Kingdom
n.moradpoor@abertay.ac.uk

ABSTRACT
Structured Query Language Injection (SQLI) attack is a code injection technique in which malicious SQL statements are inserted into the SQL database by simply using web browsers. SQLI attack can cause severe damages on a given SQL database such as losing data, disclosing confidential information or even changing the values of data. It has also been rated as the number-one attack on the Open Web Application Security Project (OWASP) top ten. In this paper, we propose an effective model to deal with this problem based on Neural Networks (NNs). The proposed model is built from three main elements of: a Uniform Resource Locator (URL) generator in order to generate thousands of malicious and benign URLs, a URL classifier in order to classify the generated URLs to either benign or malicious URLs, and an NN model in order to detect either a given URL is a malicious URL or a benign URL. The model is first trained and then evaluated by employing both benign and malicious URLs. The results of the experiments are presented in order to demonstrate the effectiveness of the proposed approach.

Categories and Subject Descriptors
G.4 [Mathematics of Computing]: MATHEMATICAL SOFTWARE (MATLAB) and D.3.2 [PROGRAMMING LANGUAGES]: Language Classifications (C, C++, Java)

General Terms
Algorithms, Design, Experimentation, Measurement, Performance, Reliability, Security, Verification

Keywords
Anomaly detection, SQL injection attack, machine learning, Artificial Intelligence, Neural Networks, NNs

1. INTRODUCTION
Thousands of organisations store important and confidential information related to them, their customers, and their business partners in databases all across the world. The stored data ranges from personal information such as first name, last name, date of birth, student identification number, staff identification number, home address, work address, mobile phone number, national insurance number, email address, and job title to more sensitive information such as username, password, pin code, and credit card information.

CIA which stands for: Confidentiality, which is a set of rules that limits or restricts access to certain type of information, Integrity, which assures the accuracy of information during its full life-cycle, and Availability, which certifies the availability of information at a required level of performance, are the main features of computer security. If a given database is attacked, the data in the database can be disclosed to unauthorised users or in worst-case scenario, it can be modified by the hackers or completely wiped out from the database. Therefore, it is important for any organisation to protect their databases in order to prevent any loss to themselves and to their customers. SQL is a programming language, which is designed for managing data, including data definition, data insertion, data removal and data modification, in a Relational Database Management System (RDBMS).

SQLI attack is a code injection technique and is one of the most popular application-layer attacks in which hackers try to disclose, modify or remove data from a given database by simply using SQL language and web browser. The SQLI attack takes advantages of inappropriate or poor coding of web applications that allows hackers to inject crafted SQL commands into say a login form in order to gain un-authorised access to data within a given database. For instance, imagine a simple login page where a legitimate user needs to enter his or her username and password in order to see his/her personal information retrieved from a given database lie behind the login page. After a given user enters his or her username and password, the login page communicates with the associated database to verify the username and password. On the verifications, the legitimate user is granted an appropriate access. If the inputs from user side are not properly sanitised, a hacker can generate crafted SQL commands and can inject them into the database in order to pass the login barrier and see what lies behind it.

Generally speaking, the SQLI attack is a technology vulnerability that comes from dynamic script language such as Hypertext Processor (PHP), Active Server Pages (ASP), Java Server pages (JSP) and Common Gateway Interface (CGI).

In this paper, we present a new approach for the detection of SQLI attack based on Artificial Intelligence (AI) techniques and particularly NNs. The proposed NN-based model for the detection of SQLI attack includes three main elements as follows: a URL generator, a URL classifier, and an NN model. The first element of the NN-based model for the detection of SQLI attack is a URL generator which is capable of generating both benign and malicious URLs. The benign URLs come from the popular website addresses in the UK while the malicious URLs are generated by taking into account the popular SQLI attack keywords as well as the popular SQLI attack URL patterns in conjunction with the popular website addresses in the UK. The second element of the proposed model is a URL classifier which is capable of classifying all the generated URLs created by the first element into malicious or benign URLs. The third element of the proposed NN-based model for the detection of SQLI attack is an NN model which comprises of 10 hidden layers (neurons) with 2 input and 2 output nodes. The NN model uses the generated URLs along with their classifications for training, validating, and then testing steps. The remainder of this paper is organised as follows. In Section II, the existing works related to the SQALI attack are briefly discussed. The proposed NN-based model for the detection of SQLI attack is discussed and detailed in Section III. Sections IV and V include the implementations and captured

results, respectively followed by conclusions of the work in Section VI, acknowledgments and references.

2. RELATED WORK

SQLI attack is one of the many web-based attacks used by the intruders in order to steal data from organisations. The attack is formed based on the vulnerabilities detected in improper coding of web applications that allows hackers to inject crafted SQL commands into databases lies behind the web applications. This results in threatening the Confidentiality, Integrity and Availability of the data, which are the three main components of the computer security in any organisation. To date, a wide range of research has been carried out for the successful detections and presentations of the web-based attacks.

In [3], authors presented the possible types of SQLI attack as well as the different tools that can help to detect and prevent this form of attack. The represented types of SQLI attack were also been assessed among current tools in terms of their ability to stop this category of attack.

In [4], authors investigated the SQLI detection and prevention techniques and their ability to stop this type of attacks.

In [5], the authors categorised the possible SQLI attack into Tautologies, Logically Incorrect Queries, Union Query, Stored Procedure, Piggy-Backed Queries Attacks, Alternate Encodings, Inference Based Attacks, Blind Injection Attacks, and Timing Injection Attacks. They have also presented a comprehensive analysis of different types of detection and presentation techniques for SQLI attack.

In [6], authors proposed a combinational approach for protection web applications against SQLI attacks in which the signature based method and auditing method are combined. Based on the captured results, their proposed approach doesn't require modifications of the runtime system and also imposes a low execution overhead.

In [7], the authors presented an approach for the detection of SQLI attack based on information theory. They have also implemented a prototype tool in Java for their proposed approach. Their proposed approach is capable of detecting known and also unknown SQLI vulnerabilities.

In this paper, we present a novel approach for the detection of SQLI attack based on AI and particularly NNs which comprises of three main elements of: a URL generator, a URL classifier and an NN model. The proposed NN-based model shows a good performance in terms of accuracy, true-positive rate, and false-positive rate. The proposed NN-based model is discussed in next section.

3. THE NN-BASED PROPOSED MODEL

The proposed NN-based model for the detection of the SQLI attack includes three main elements: a URL generator, a URL classifier, and an NN model as follows.

3.1. A URL Generator

The URL generator part of the proposed NN-based model for the detection of SQLI attack includes two components of benign URLs and malicious URLs. The benign URLs comprise the popular website addresses in the UK while the malicious URLs are simply generated by adding the popular SQLI attack keywords and the popular SQLI attack URL patterns to the benign URLs.

The popular SQLI attack keywords and the popular SQLI attack URL patterns, which are the two main components of the URL generator, are represented as follows.

3.1.1 SQLI Attack Keywords

The SQLI attack keywords are the popular keywords in SQL language which are generally used in order to perform operations on the tables inside a given SQL database. The popular SQLI attack keywords and their operations performed on a given SQL database are specified in Table 1. The SQLI attack keywords can also be considered as independent and non-independent keywords in which the former operates on a given database independently (e.g. drop, create) while the latter needs to be used in conjunction with the other SQLI attack keywords (e.g. where, set).

3.1.2 SQLI Attack URL Patterns

SQLI attacks URL patterns are the possible combinations between the SQLI attack keywords, seized from Table1, as well as the popular patterns used by hackers against a given database. Some of the common SQLI attack URL patterns are as follows:

- ending with "--"
- ending with "/* */"
- containing UNION, SELECT and FROM
- containing "information_schema"
- containing "load_file"

A group of SQLI attacks URL patterns, which we have considered in our implementations, are represented in Table 2.

3.2. A URL Classifier

The classifier part of the proposed NN-based model for the detection of SQLI attack is developed in order to group the URLs, which are generated by the URL generator from the previous section, into either malicious URLs or benign URLs. It is also responsible for converting the original generated URLs into strings of logic (1 as true and0 as false). Let a URL characteristic r_i generated by the URL generator is defined by a random variable R_i:

$$R_i = \begin{cases} 1, \text{if discovered by the keyword detectors} \\ 0, \text{if not discovered by the keyword detector} \end{cases}$$

Let *C* be a random variable representing the generated URL class, malicious or benign:

$$C \in \{\text{malicious, benign}\}$$

Every generated URL is assigned a vector defined by $\bar{r} = (r_1, r_2, , \dots, r_n)$ with r_i being the result of thei-th random variable R_i.

3.3. A Neural Network (NN) Model

The third component of the proposed NN-based model for the detection of SQLI attack includes an NN model comprises *n* layers (*n* hidden nodes or neurons) with 2 input and 2 output nodes. The NN model uses the malicious and benign URLs, which are generated by the URL generator and classified by the URL classifier from the previous sections, for training, validating, and testing. The components of the NN model are put together in two groups of training components and validating components as follows.

Table 1. SQLI attack keywords

Keyword	Operation performed on a given SQL database
Drop	This keyword is used in order to remove data elements such as tables/databases from a given SQL database.
Create	This keyword is used in order to create new data elements such as new tables/databases in a given SQL database.
Delete	This keyword is used in order to remove data (row) from tables in a given SQL database.
Insert	This keyword is used in order to add data (row) to tables in a given SQL database.
Select	This keyword is used in order to select data (row) from tables in a given SQL database.
Union	This keyword is used in order to combine the results of two or more SELECT statements having the same number of columns and the same data type.
Update	This keyword is used in order to update data (row) from tables in a given SQL database. The update keyword is usually used with the set keyword.
Set	This keyword is used in order to set data (row) with specific value in tables from a given SQL database. The set keyword is usually used with the update keyword.
Alter	This keyword is used in order to alter tables (add/delete/modify columns) in a given SQL database.
Where	This keyword is used in order to filter records retrieved from a given SQL database based on a specific criterion.
Like	This keyword is used in order to search for a specific pattern in a column. The like keyword is usually used with the where keyword.
From	This keyword is used in order to specify on which databases/tables the operation has to be performed in a given SQL database.
Table	This keyword refers to a Table as a data element in a given SQL database which includes table name, columns and data type. The Table keyword is usually used with: create, delete, drop, insert, from, and where keywords.
Database	This keyword refers to a Database as a data element in a given SQL database which includes database name and table within the database. The Database keyword is usually used with: create and drop keywords.
And	This keyword is used in order to filter data retrieved from a given SQL database based on more than one condition. The AND condition display values form a given database/table if the first condition, and the second condition are true.
Or	This keyword is used in order to filter records retrieved from a given SQL database based on more than one condition. The OR condition display values form a given database/table if either the first condition or the second condition is true.
=	This keyword is used in order to check if the values of two operands are equal or not. If they are equal then the condition becomes true.
--	This keyword is one of the SQL comment operators specifies that the comment continuous to the end of line. Therefore, everything after it is ignored by a database. Hackers inject executable commands into a given SQL database by putting them after '--'.
/* */	This keyword is one of the SQL comment operators specifies that the comment is an in-line or multiple-line comment. Therefore, everything after it is ignored by a database. Hackers inject executable commands into a given SQL database by putting them inside /* */.
Information_schema	This keyword is used in order to retrieve metadata (data about data) about the object from a given SQL database.
Load_file	This keyword is used in order to read either a text or a binary file.

Table 2. SQLI attack URL patterns

Independent keywords		**Non-independent keywords**					
		And =	**Or =**	**load_file**	**information _schema**	**--**	**/* */**
ADMIN	**DROP**	x	x	x	x	x	x
CREATE	**TABLE**	x	x	x	x	x	x
DELETE	**FROM**	x	x	x	x	x	x
DROP	**TABLE**	x	x	x	x	x	x
INSERT	**INTO**	x	x	x	x	x	x
SELECT	**UNION**	x	x	x	x	x	x
UPDATE	**SET**	x	x	x	x	x	x

3.3.1 Training Components

- 'Input' matrix: this matrix includes the data which the proposed NN model uses in training stage. It comprises all the benign and malicious URLs generated by the URL generator part of the proposed model.

- 'Target' matrix: this matrix includes all the decisions (malicious or benign) for each string of data stored in 'Input' matrix.
- Fitness network: this is the neural network with *n* neurons in hidden layer in which the data from 'Input' and 'Target' matrixes will be used for training, validating and testing, consequently.

3.3.2 Validating Components

- 'Sample' matrix: this matrix contains sample data from the 'Input' matrix discussed in previous section. The trained NN model uses 'Sample' data as input in validation stage.
- 'Output' matrix: this matrix contains output data for the data represented in 'Sample' matrix. The trained NN Network predicts the output value for 'Sample' matrix and stores it in 'Output' matrix.

The implementation of the proposed NN-based model for the detection of the SQLI attack, which includes the three components of: a URL generator, a URL classifier, and an NN model, is discussed in the next section.

4. IMPLEMENTATIONS

In this paper, an NN-based model for the detection of SQLI attack is proposed. The proposed model includes three main elements of: a URL generator, a URL classifier, and an NN model. All three elements are implemented as follows.

4.1 The URL Generator

As it was discussed in the previous section, the URL generator includes two main components of the benign URLs and the malicious URLs. The benign URLs includes a list of the most popular website addresses in the UK while the malicious URLs are generated by adding the popular SQLI attack keywords and the popular SQLI attack URL patterns to the benign URLs. In our implementations, we are used the top 500 popular website addresses in the UK [2] as benign URLs. We are also generated the malicious URLs by adding the SQLI attack keywords from Table 1 and SQLI attack URL patterns from Table 2 to the top 500 website addresses in the UK using PHP language. The total number of the malicious URLs in our scenario is 23,500 URLs which are generated from 500 benign URLs, by taking into consideration the independent and the non-independent SQLI attack keywords as well as their combinations.

4.2 The URL Classifier

As it was discussed in the previous section, the URL classifier component of the proposed NN-based model is first responsible for classifying the URLs, which are generated by the URL generator component, either to malicious or benign URLs (1 as true or malicious and 0 as false or benign) and second to convert the generated URLs into strings of logics (0s and 1s). For example, imagine a generated URL has the SQLI attack independent keywords of: 'ADMIN DROP', 'CREATE TABLE', 'DELETE TABLE', 'DROP TABLE', 'INSERT INTO', 'SELECT UNION', and 'UPDATE SET' as well as a non-independent keyword of: 'AND ='. After passing the generated URL to the URL classifier, the URL classifier first classifies it as a malicious URL; therefore, a logical value of 1 will be assigned to it. It then allocates a vector of $r^- = (r_1, r_2, , \dots, r_{13})$, which is 1111111100000 in our example, to the whole generated URL.

In our implementations, each component of the r^-vector assigned to a particular SQLI attack keyword/pattern, which is 13 in our scenario, Table 3. However, the implementation can be easily extended in order to cover any new combination.

Table 3. Assigned vectors to the SQLI attack keywords

Vectors	SQLI attack keywords
r_1	CREATE TABLE
r_2	DELETE FROM
r_3	DROP TABLE
r_4	INSERT INTO
r_5	SELECT UNION
r_6	UPDATE SET
r_7	And =
r_8	Or =
r_9	load_file
r_{10}	information_schema
r_{11}	--
r_{13}	/* */

4.3 The NN Model

As it was discussed in the previous section, the NN model of the proposed scheme comprises of *n* layers (*n* hidden nodes or *n* neurons) with 2 input and 2 output nodes and uses the malicious and benign URLs for training, validating, and testing. These URLs are generated by the URL generator and then classified by the URL classier of the proposed model. In our implementations, we have implemented an NN model of the 10 layers (10 hidden nodes or 10 neurons) in which 70%, 15% and 15% of the total benign and malicious URLs for training, validating and testing, respectively. MATLAB [1], which is popular software for the numerical calculations and formulas with the vast library of functions and algorithms, is used in order to develop, train, validate and test the proposed NN model. The training and validating components of the NN model are configured as follows.

4.3.1 Training Components

- 'Input' matrix: this matrix is a logical *n x13* matrix where the data represented in strings of logics; *1* as true and *0* as false.
- 'Target' matrix: this matrix is a logical *nx1* matrix where the data represented in logics; *1* as malicious and *0* as benign.
- Fitness network: this is the neural network with 10 neurons in hidden layer in which 70%, 15%, and 15% of the data from 'Input' and 'Target' matrixes will be used for training, validating and testing, respectively.

4.3.2 Validating Components

- 'Sample' matrix: this matrix is a logical *nx13* matrix contains sample data from the 'Input' matrix.
- 'Output' matrix: this matrix is a logical *nx1* matrix contains output data for the data represented in 'Sample' matrix. The trained NN Network predicts the output value for 'Sample' matrix and stores it in the 'Output' matrix.

The capture results are discussed in the next section.

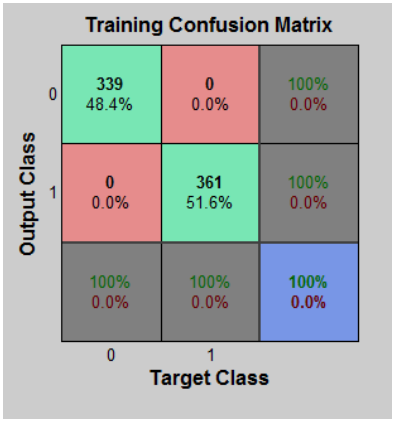

Figure 1. The confusion matrix for training

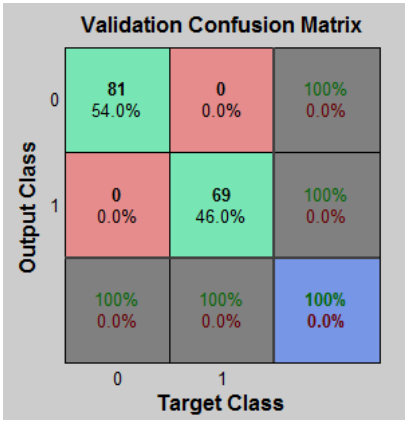

Figure 2. The confusion matrix for validating

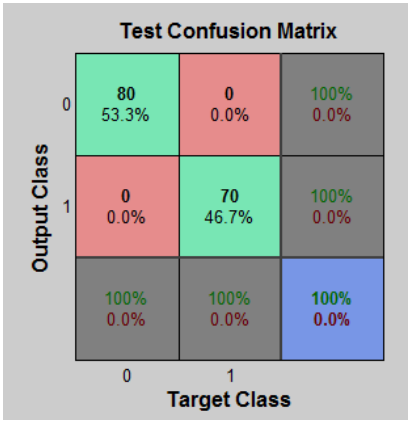

Figure 3. The confusion matrix for testing

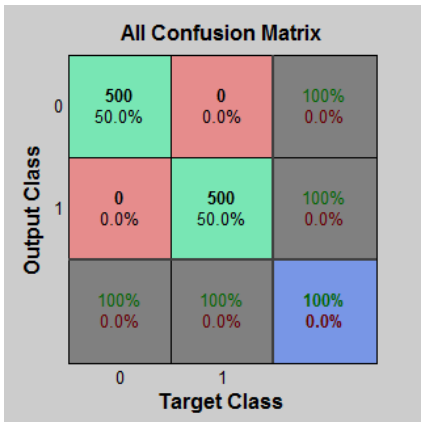

Figure 4. The confusion matrix for all (training, validating, and testing)

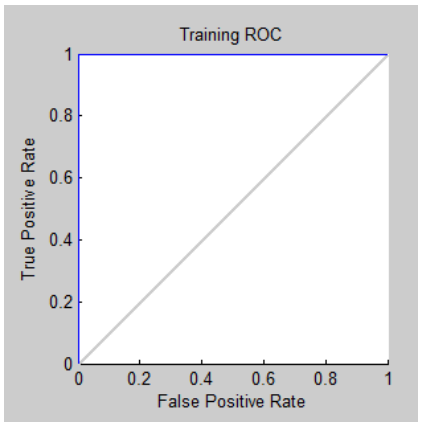

Figure 5. The ROC curve for training

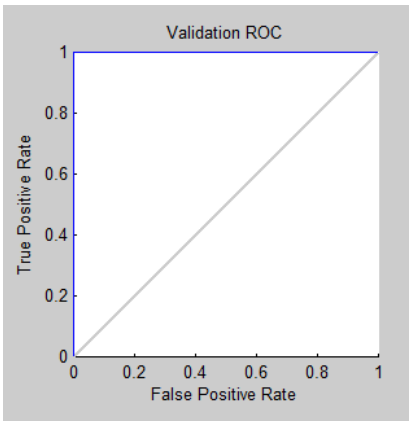

Figure 6. The ROC curve for validating

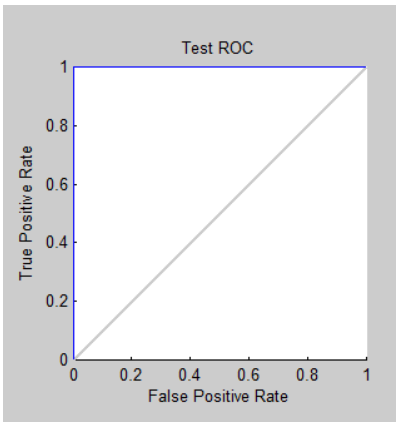

Figure 7. The ROC curve for testing

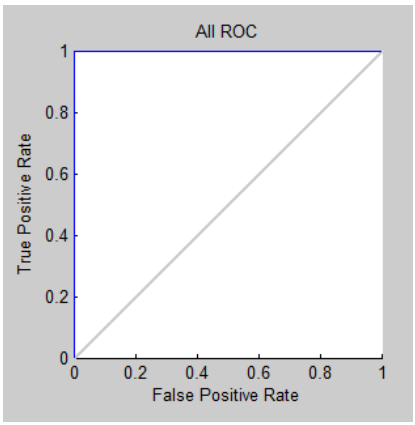

Figure 8. The ROC curve for all (training, validating, and testing)

5. RESULTS

As it was discussed in the previous sections, our proposed scheme includes three main elements of the: URL generator, URL classifier, and NN model. In order to capture the performance of the proposed NN-based model for the detection of the SQLI attack, we decided to consider 1,000 generated URLs, including 500 benign URLs and 500 malicious URLs. The 500 benign URLs are the top 500 popular URL addresses in the UK while the malicious URLs are generated by adding the popular SQLI attack keywords and the popular SQLI attack URL patterns into benign URLs using PHP. The 1,000 URLs are then classified into either the malicious URLs or the benign URLs by using the URL classifier element of the proposed model. At the end, we train, evaluate and then test the NN model, which has 10 hidden layers, with all 1,000 URLs along with their classifications using MATLAB. The captured results are as follows. The confusion matrices for training, validating, and testing are shown in Figure 1, Figure 2, and Figure 3, respectively. We have also captured the confusion matrix for all in Figure 4 in which all three types of data (training, validating, and testing) are combined. The numbers of the correct responses are shown in green squares (left squares in the first rows and middle squares in the second rows) while the numbers of the incorrect responses are shown in the red squares (middle squares in the first row and left squares in the second rows). The grey squares at the end of the first and the second rows as well as the last rows illustrate the percentages of the accuracies (upper numbers) and inaccuracies (bottom numbers) for output classes and target classes, respectively. The lower-right blue squares illustrate the overall accuracies (upper numbers) and overall inaccuracies (bottom numbers) by taking into account the accuracies and inaccuracies in output classes and target classes. As they are depicted in Figure 1 to Figure 3 and based on the configurations in our implemented scenario, the 1000 selected URLs (500 benign plus 500 malicious URLs) are scattered in three phases of training, validating, and testing with distribution rates of 70%, 15%, and 15%, respectively. By taking into account the distribution rates in three phases:

- The training phase receives 700 out of 1000 URLs. This includes 339 benign URLs and 361 malicious URLs while the former is 48.4%, and the latter is 51.6% of the total URLs used in this phase.
- The validating phase receives 150 out of 1000 URLs. This includes 81 benign URLs and 69 malicious URLs while the former is 54%, and the latter is 46% of the total URLs used in this phase.
- The testing phase receives 150 out of 1000 URLs. This includes 80 benign URLs and 70 malicious URLs while the former is 53.3%, and the latter is 46.7% of the total URLs used in this phase.

Addressing the captured percentages of the incorrect responses in the red squares, Figure 1 to Figure 4, as well as the overall percentages of the accuracies and inaccurate in blue squares, we can say that the outputs are accurate. Therefore, the proposed model is trained correct and hence, it performs correct. The Receiver Operating Characteristic (ROC) curves for training, validating, and testing is shown in Figure 5, Figure 6, and Figure 7, respectively. We have also captured the ROC curve for all in Figure 8 in which all three types of data (training, validating, and testing) are combined. The colored lines in each axis represent the ROC curves. The ROC curve is a plot of the true-positive rate (sensitivity) against the false-positive rate (specificity). In our implementations, the true-positive rate (sensitivity) is the percentages of the benign URLs which are correctly identified as benign URLs and the percentages of the malicious URLs which are correctly identified as malicious URLs. Additionally, the false-positive rate (specificity) is the percentages of the benign URLs which are incorrectly identified as malicious URLs and the percentages of the malicious URLs which are incorrectly identified as benign URLs. A perfect test would show points in the upper-left corner, with 100% sensitivity (i.e. predicting all benign URLs as benign and all malicious URLs as malicious) and 100% specificity (i.e. not predicting any benign URL as malicious and not predicting any malicious URL as benign). Addressing Figure 5 to Figure 8, the proposed network performs correct.

6. CONCLUSIONS

In this paper, a new approach based on the Neural Networks (NNs) for the detection of Structure Query Language Injection (SQLI) attack is proposed. The proposed model includes three main elements of a URL generator, a URL classifier and an NN model. The URL generator and the URL classifier are employed in order to provide required malicious and benign URLs for three phases of testing, validating, and training of the NN model. Addressing the captured results, the proposed NN-based model for the detection of SQLI attack shows a good performance in terms of accuracy, true positive rate as well as the false-positive rate.

7. ACKNOWLEDGMENTS

The author would wish to acknowledge the support of the University of Abertay Dundee for funding this work.

8. REFERENCES

[1] MATLAB R2012a, available at:: http://www.mathworks.co.uk [retrieved: May, 2014]

[2] Alexa, Bringing Information into Focus, available at: http://www.alexa.com/topsites/countries/GB [retrieved: May, 2014]

[3] Tajpour, A., Masrom, M., Heydari, M. Z., and Ibrahim, S., 2010, SQL injection detection and prevention tools assessment, *In Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology* (Chengdu, July 09-11, 2010), v.9, pp.518 – 522

[4] Tajpour, A., and JorJor Zade Shooshtari, M., 2010, Evaluation of SQL Injection Detection and Prevention Techniques, *In Proceedings of the second International Conference on Computational Intelligence, Communication Systems and Networks* (Liverpool, July 28-30, 2010), pp. 216 - 221

[5] Kumar, P., and Pateriya, R.K, 2012, A Survey on SQL injection attacks, detection and prevention techniques, *In Proceedings of the third International Conference on Computing Communication & Networking Technologies* (Coimbatore, July 26-28, 2012), pp.1 – 5

[6] Ezumalai, R., Pondicherry, and Aghila, G., 2009, Combinatorial Approach for Preventing SQL Injection Attacks, *In Proceedings of the IEEE International Conference in Advance Computing* (Patiala , March 6-7, 2009), pp.1212 – 1217

[7] Shahriar, H., and Zulkernine, M., 2012, Information-Theoretic Detection of SQL Injection Attacks, *In Proceedings of the 14th International Symposium on Date of High-Assurance Systems Engineering* (Omaha, NE, October 25-27, 2012), pp. 40 - 47