Every time [some software engineer] says, “Nobody will go to the trouble of doing that”, there’s some kid in Finland who will go to the trouble.

-Alex Mayfield

IMPLEMENTATION OF PREVENTION OF XPATH INJECTION ATTACK USING PYBRAIN MACHINE LEARNING LIBRARY

GAJENDRA DESHPANDE

Asst. Professor, Department of Computer Science and Engineering,

KLS Gogte Institute of Technology, Udyambag, Belagavi, Karnataka

Scipy India 2017

Lecture Hall Complex, IIT Bombay – 30th November 2017

Contents

Introduction

- Cyber Space is a national asset
- XML is a heart of many mainstream technologies, Web Services, Service Oriented Architecture(SOA), Cloud Computing etc.
- Web Services vulnerabilities can be present in Operating System, Network, Database, Web Server, Application Server, Application code, XML parsers and XML appliances
- New technologies – New Challenges → (Old challenges + New Challenges)

Problem Definition and Proposed Solution

Problem Definition

- To secure web resources from XPath injection attack using modular recurrent neural networks.

Proposed Solution

- The proposed solution uses modular recurrent neural network architecture to identify and classify atypical behavior in user input. Once the atypical user input is identified, the attacker is redirected to sham resources to protect the critical data.
  - Count based validation technique

Introduction to XPath Injection

- An attacker can craft special user-controllable input consisting of XPath expressions to inject the XML database and bypass authentication or glean information that he normally would not be able to.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
    <user>
        <username>gandalf</username>
        <password>!c3</password>
        <account>admin</account>
    </user>
</users>
```

```
string(//user[username/text()='gandalf' and password/text()='!c3']/account/text())

string(//user[username/text()='' or '1' = '1' and password/text()='' or '1' = '1']/account/text())
```

CAPEC on XPath Injection

Factor	Description
Attack Prerequisites	XPath Queries and unsanitized user controllable input
Typical Likelihood of Exploit	High
Attacker Skills	Low
Indicators	Too many exceptions generated by the application as a result of malformed XPath queries
Resource Required	None
Attack Motivation Consequences	Confidentiality- gain privileges and read application data
Injection Vector	User-controllable input used as part of dynamic XPath queries
Payload	XPath expressions intended to defeat checks run by XPath queries
Activation Zone	XML Database
CIA Impact	High, High, Medium
Architectural Paradigms	Client-Server, Service Oriented Architecture (SOA)
Frameworks, Platforms, Languages	All

Related Work

Authors	Title, Year, Publication	Methods Used
[1] Thiago Mattos Rosa et.al.	Mitigating XML Injection Attack through Strategy-based Detection System, 2011, IEEE Security and Privacy[2011 Impact Factor:0.898]	This paper applies ontology to build a strategy based knowledge (XID) to protect web services from XML injection attack and to mitigate from zero-day attack problem. In strategy based design new attack input will be automatically added to the ontology database. As the number of attacks in the ontology database increase, the technique will result in increased response time.
[2] Nuno Antunes et.al.	Effective Detection of SQL/XPath Injection Vulnerabilities in Web Services, 2009, IEEE International Conference [Research Track Acceptance Rate: 17%]	The approach is based on XPath and SQL commands learning and posterior detection of vulnerabilities by comparing the structure of the commands issued in the presence of attacks to the ones previously learned. In this approach results were not promising since the workload generation took few seconds of time, but learning phase took a few minutes of time per operation. The overall time taken by the detection process is approximately 15 minutes per operation.

Related Work

Authors	Title, Year, Publication	Methods Used
[3] Nuno Laranjeiro et.al.	A Learning-Based Approach to Secure Web Services from SQL/ XPath Injection Attacks, 2010, IEEE Pacific Rim International Symposium	The approach is to learn valid request patterns (learning phase) and then detect and abort potentially harmful requests (protection phase). The authors achieved 76% accuracy in detecting the SQL/XPath injection attacks.
[4] V. Shanmughaneethi et.al.	PXpathV: Preventing XPath Injection Vulnerabilities in Web Applications, 2011, IJWSC	In this paper XPath Expression Scanner is integrated with XPath Expression Analyzer to validate XPath Expressions. The response time was not promising compared to earlier approaches.

Related Work

Authors	Title, Year, Publication	Methods Used
[6] Mike Shields, Matthew Casey	A theoretical framework for multiple neural network systems, 2008	A theoretical framework for multiple neural network systems where a general instance of multiple networks is strictly examined. The authors claim that using an arbitrary number of redundant networks to perform complex tasks often results in improved performance
[7] Hanh H. Nguyen, Christine W. Chan	Multiple neural networks for a long term time series forecast, 2004, Springer, Neural Computing & Applications 13: 90–98	The concept of multiple artificial neural networks was used for long term time series prediction where prediction is done by multiple neural networks at different time lengths. The authors showed that the multiple neural network system performed better compared to single artificial neural network for long term forecast
[8] Anand R. et. al,	Efficient classification for multiclass problems using modular neural networks, 1995, IEEE Transactions on Neural Networks, Volume 6, Issue 1	The modular neural network was used to reduce k - class problems to a set of k two-class problems, where each problem was dealt with separately trained network to achieve better performance compared to non-modular networks .

Research Gap Identified

Neural network approach to identify and classify atypical behavior in input

The study showed different approaches to handle XPath injection attacks. It also showed methods applied and their disadvantages. We can conclude from the study that neural networks are not applied to detect Xpath injection attacks and existing results are not promising.

The study showed, how modularity in case of neural networks helps to achieve improved performance. Modular neural networks have not been applied to cyber security particularly to the detection of SQL/XPath injection attacks.

System Design

Some valid inputs:
Email-id
Mobile number
Alphanumeric word

Some malicious inputs:
'1 or 1=1
user' or 'a'='a
%00

Some invalid inputs:
Very large input string
String with special characters
String formed from different character set

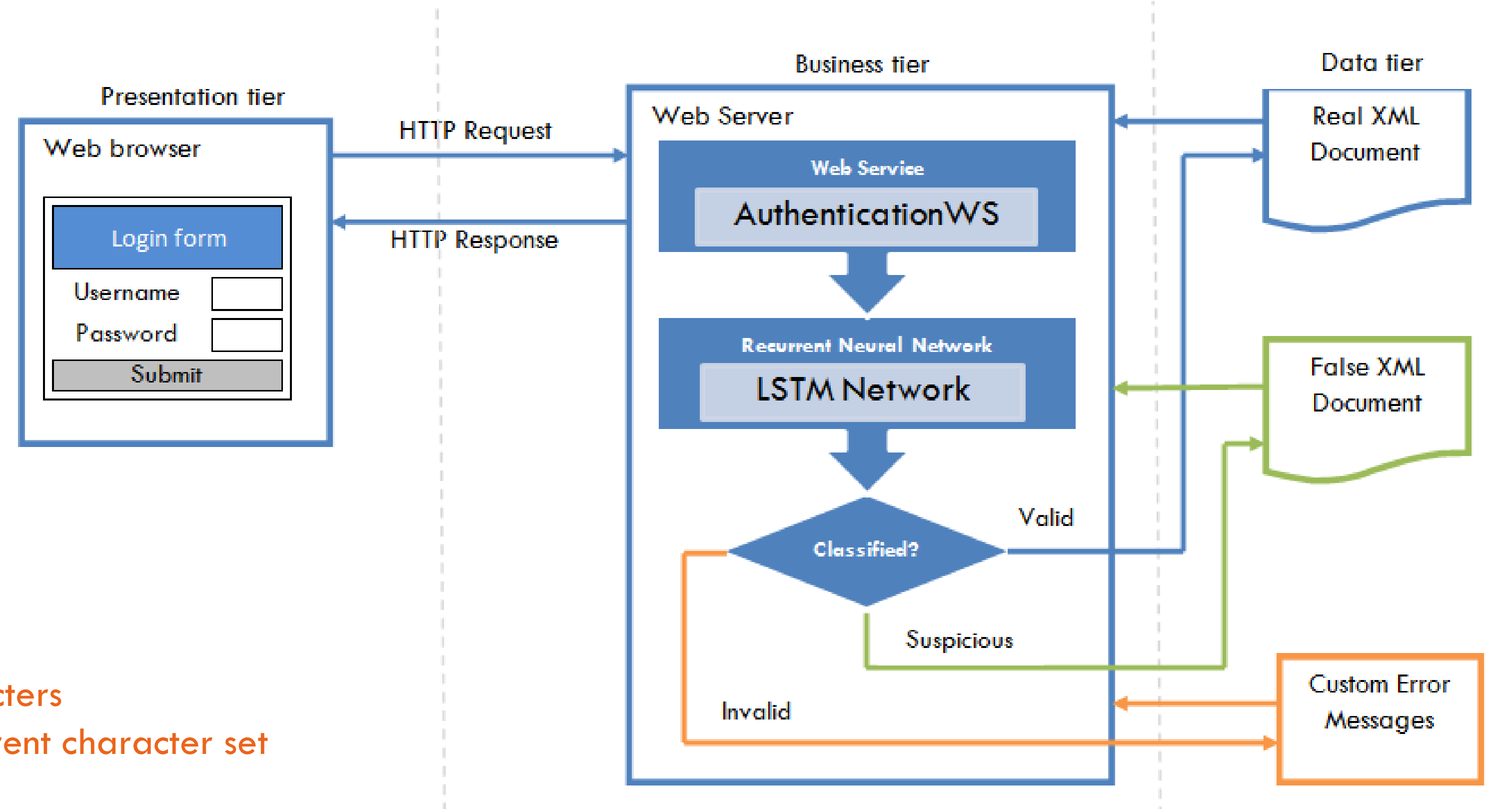


Fig. 1: Three tier architecture of the proposed system

Algorithm

Algorithm

1. Scan the user input.
2. Determine the length of user input.
3. Count the frequency of every character in the user input [a-z, A-Z, 0-9, ‘ “ . @ # % + = ? :].
4. If the frequency of character is below the threshold value set for that particular character in Table 4 then set the error code to 40.
5. Else if the frequency of characters [. @ # % + = ‘ “] is above the threshold value set for that particular character in Table 4 then set the error code to 4000.
6. Else set the error code to 400.
7. Build a recurrent neural network 1 consisting of 50 neurons with hidden layer as LSTM network and output layer as SoftMax.
8. Use Rprop- trainer to train the network using the training dataset created using error codes in Table 2.
9. Use the test dataset created in real time to validate against the training dataset.
10. Build a recurrent neural network 2 consisting of 50 neurons with hidden layer as LSTM network and output layer as SoftMax.
11. Use Rprop- trainer to train the network using the training dataset created using number of login attempts in Table 1.
12. Use the test dataset created in real time to validate against the training dataset.
13. If train error and test error of both the networks are 0.0% then
    1. Finally classify the input vector based on the outputs of both the neural networks in Table 3.
    2. If the user input is successfully classified as ‘valid’ and found in the real XML file then Return the message “login successful”.
    3. Else if the user input is classified as ‘malicious’ then Return the contents of the fake XML file.
    4. Else if the user input is classified as ‘invalid’ then Return the ‘error’ message.
14. Else repeat the steps 8 through 13.

Table 1. Training dataset for classification of login attempts (Neural network 1)

Number of login attempts	Class
1	Valid
2	Valid
3	Valid
4 or more	Malicious

Table 2. Training dataset for classification of error codes (Neural network 2)

Error code	Class
40	Valid
400	Invalid
4000	Malicious

Table 4. Characters with threshold value

Special Character	Threshold	Error Code
Single quotes (‘)	1	40
Double quote (“)	0	4000
Dot (.)	2	40
Alphabets ([a-zA-Z])	Any	40
Digits ([0-9])	Any	40
At the rate (@)	1	40
Equal to (=)	0	400
Square Brackets ([,])	0	400
Round Brackets ((,))	0	400
Curly Brackets ({,})	0	400
Slashes (\, /)	0	400
Asterisk (*)	0	400
Pipe (|)	0	400
Any other character	0	400

Algorithm

Table 3. Final classification of input vector

Output of Neural Network 1	Output of Neural Network 2	Final Classification
Valid	Valid	Valid
Valid	Malicious	Malicious
Malicious	Valid	Malicious
Invalid	Valid	Invalid
Valid	Invalid	Invalid
Invalid	Malicious	Malicious
Malicious	Invalid	Malicious
Malicious	Malicious	Malicious

System Environment

Table 5: Tools and technologies used for experimentation

Software Environment		
Technology	**Server Side**	**Client Side**
Neural Networks	PyBRAIN [14]	-
Web Services	BottlePy Micro Web Framework [15]	-
Web Server	WSGIRefServer of BottlePy and Apache	-
Web Browser	Firefox, Konquerer	Firefox, Konquerer
Scripting Language, Graphs	Python, numpy, matplotlib [16]	-
Operating Systems	Fedora Linux 14	Fedora Linux 14
Hardware Environment		
System	Intel i3 processor, 3GB RAM	Intel i3 processor, 3GB RAM

Note: Same environment is used for Development and Testing of the System. The system may also be deployed on machines with lower configurations and on different platforms.

PyBRAIN Machine Learning Library

- PyBrain is a modular Machine Learning Library for Python.
- PyBrain is short for **Py**thon-**B**ased **R**einforcement Learning, **A**rtificial **I**ntelligence and **N**eural Network Library
- To download and Install PyBrain

```
$ git clone git://github.com/pybrain/pybrain.git
$ python setup.py install
```

For more detailed installation instructions visit

http://wiki.github.com/pybrain/pybrain/installation

For Information on PyBrain visit http://www.pybrain.org

Bottle- Python Web Framework

- Bottle is a fast, simple and lightweight WSGI micro web-framework for Python.
- It is distributed as a single file module and has no dependencies other than the Python Standard Library.
- It includes built in Routing, Templates, Utilities and Server
- Bottle does not depend on any external libraries. You can just download bottle.py into your project directory and start coding:

```
$ wget https://bottlepy.org/bottle.py
```

- For more information on Bottle Framework visit http://www.bottle.org

Results (True Positives)

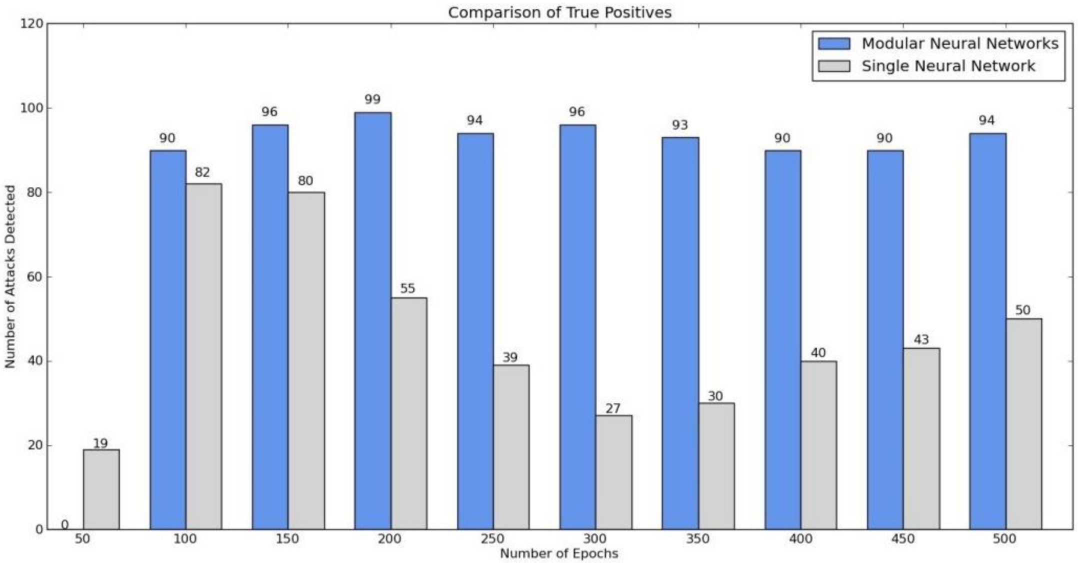


Fig. 2: Comparison of true positives

Table 6: Comparison of true positives

Number of epochs	Modular Neural Network	Single Neural Network
50	0	19
100	90	82
150	96	80
200	99	55
250	94	39
300	96	27
350	93	30
400	90	40
450	90	43
500	94	50

Results (False Positives)

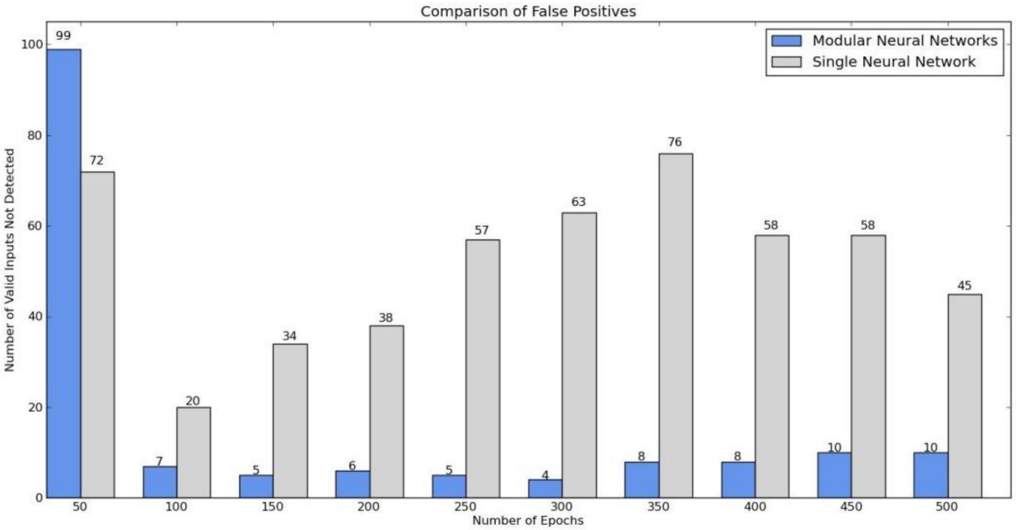


Fig. 3: Comparison of false positives

Table 7: Comparison of false positives

Number of epochs	Modular Neural Network	Single Neural Network
50	99	72
100	07	20
150	05	34
200	06	38
250	05	57
300	04	63
350	08	76
400	08	58
450	10	58
500	10	45

Results (True Negatives)

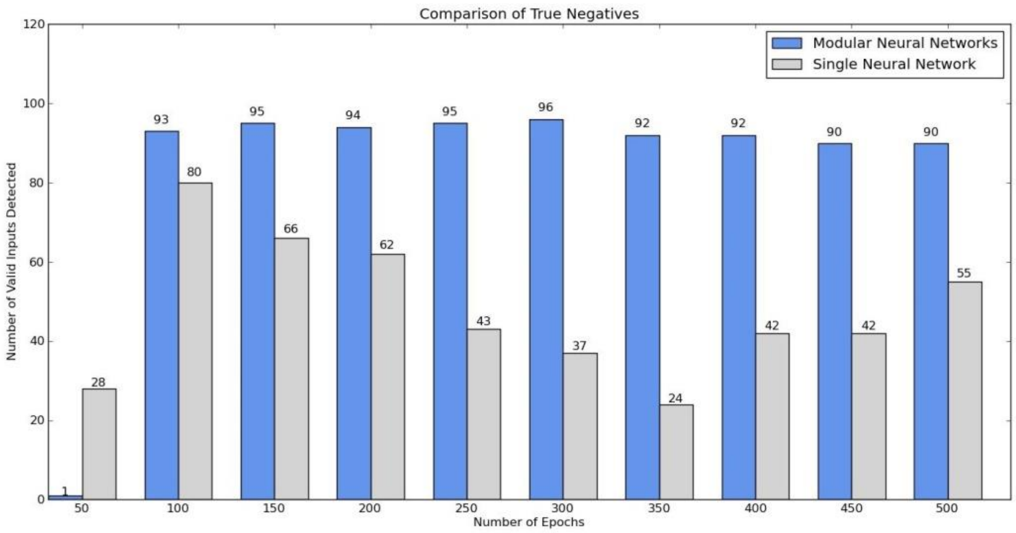


Fig. 4: Comparison of true negatives

Table 8: Comparison of true negatives

Number of epochs	Modular Neural Network	Single Neural Network
50	1	28
100	93	80
150	95	66
200	94	62
250	95	43
300	96	37
350	92	24
400	92	42
450	90	42
500	90	55

Results (False Negatives)

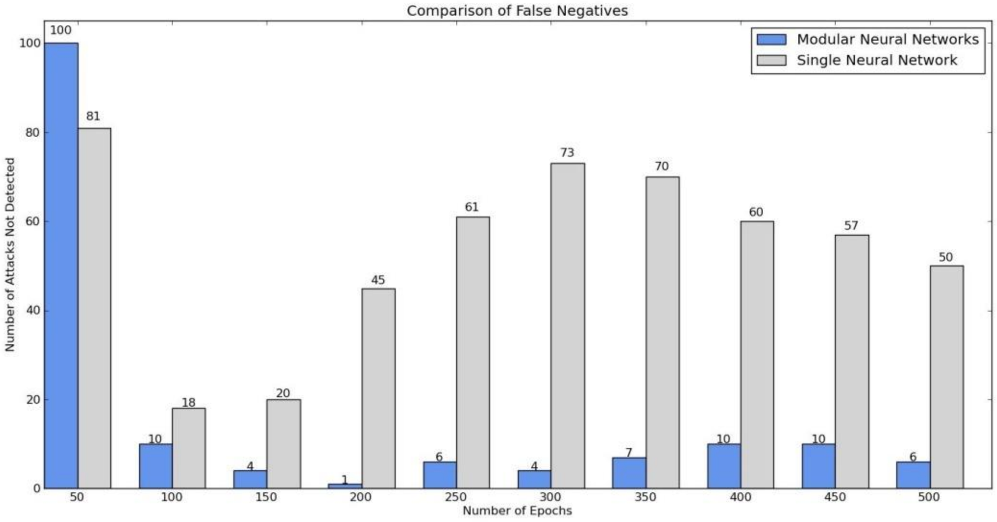


Fig. 5: Comparison of false negatives

Table 9: Comparison of false negatives

Number of epochs	Modular Neural Network	Single Neural Network
50	100	81
100	10	18
150	04	20
200	01	45
250	06	61
300	04	73
350	07	70
400	10	60
450	10	57
500	06	50

Results (Response Time)

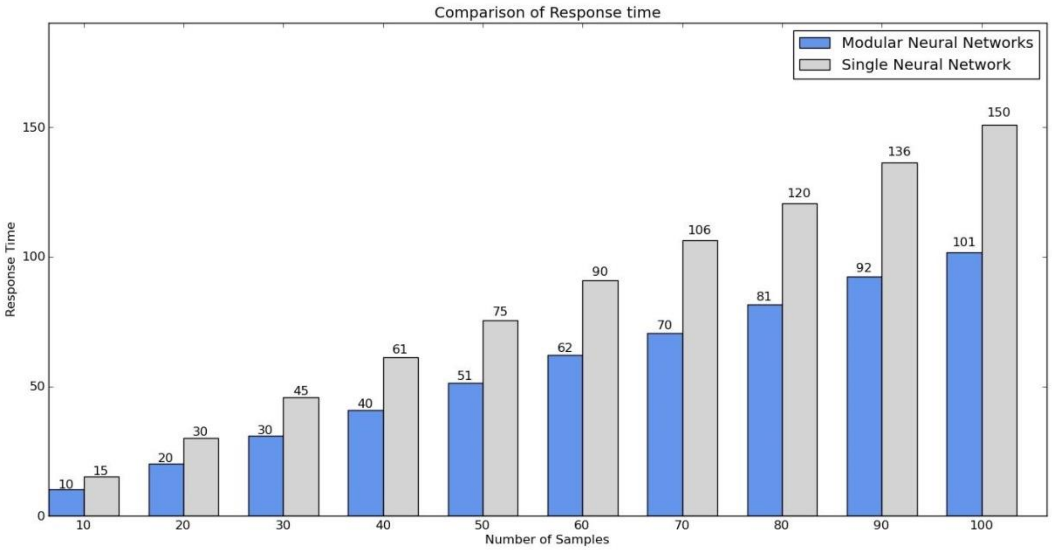


Fig. 6: Comparison of response time

Table 10: Comparison of response time

Number of samples	Modular Neural Network	Single Neural Network
10	10.23	15.31
20	20.27	30.20
30	30.98	45.74
40	40.74	61.32
50	51.31	75.61
60	62.05	90.78
70	70.54	106.34
80	81.47	120.45
90	92.27	136.17
100	101.75	150.87

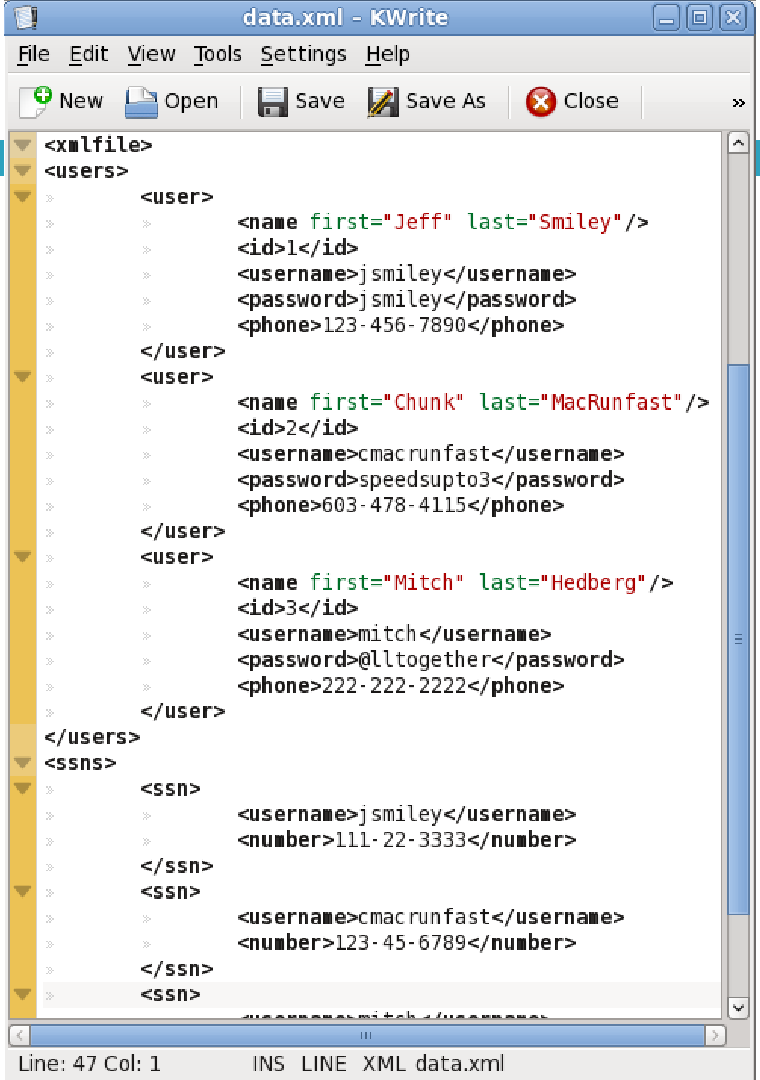
Summary of Results

Table 11: Average detection rate including and excluding an outlier

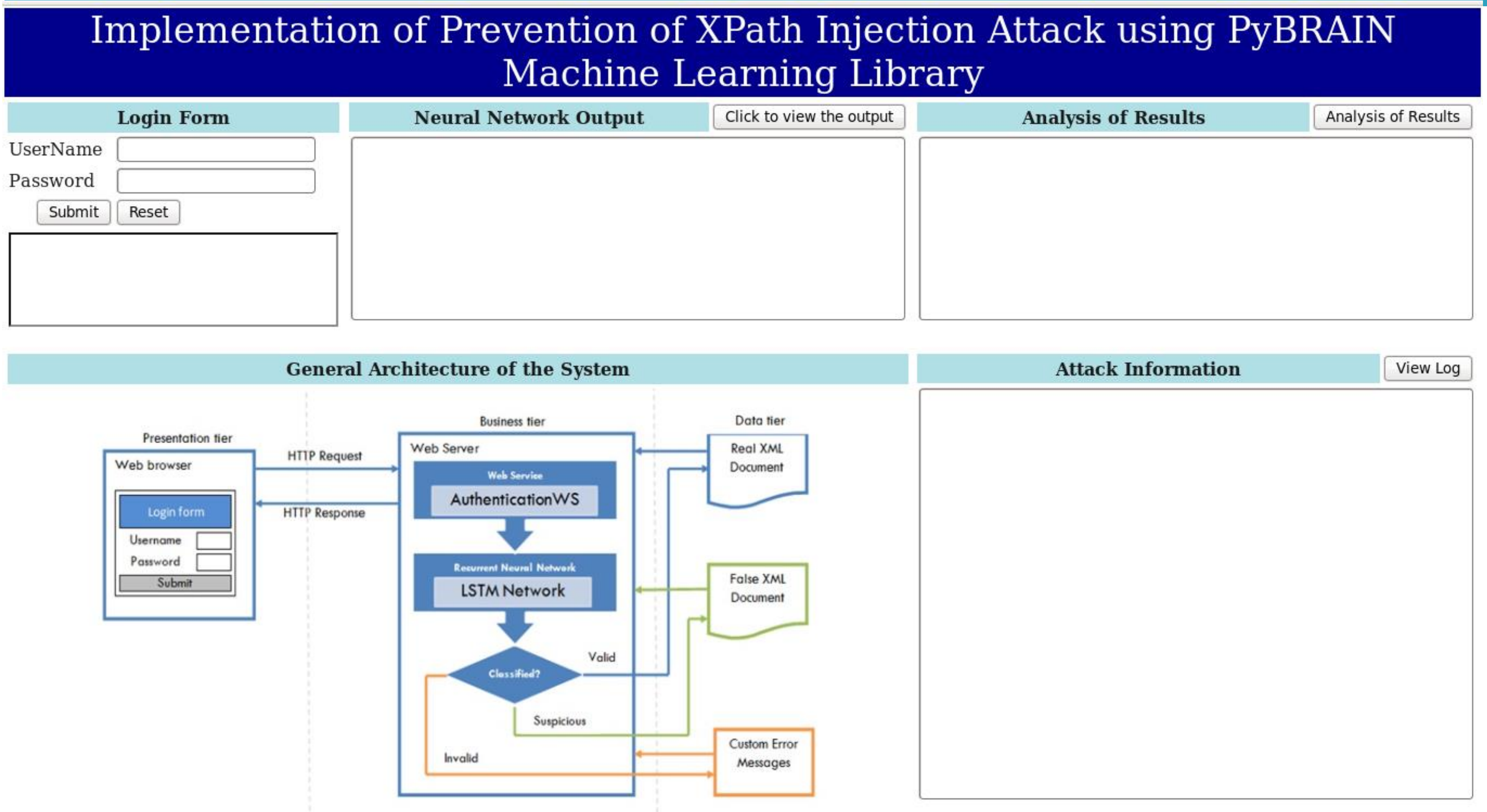
	Average detection rate including an outlier		Average detection rate excluding an outlier	
	MNN %	SNN %	MNN %	SNN %
True Positives	84.2	46.5	93.55	51.66
False Negatives	15.8	53.5	6.45	48.33
True Negatives	83.8	47.9	93.11	53.22
False Positives	16.2	52.1	6.88	46.77

Snapshots

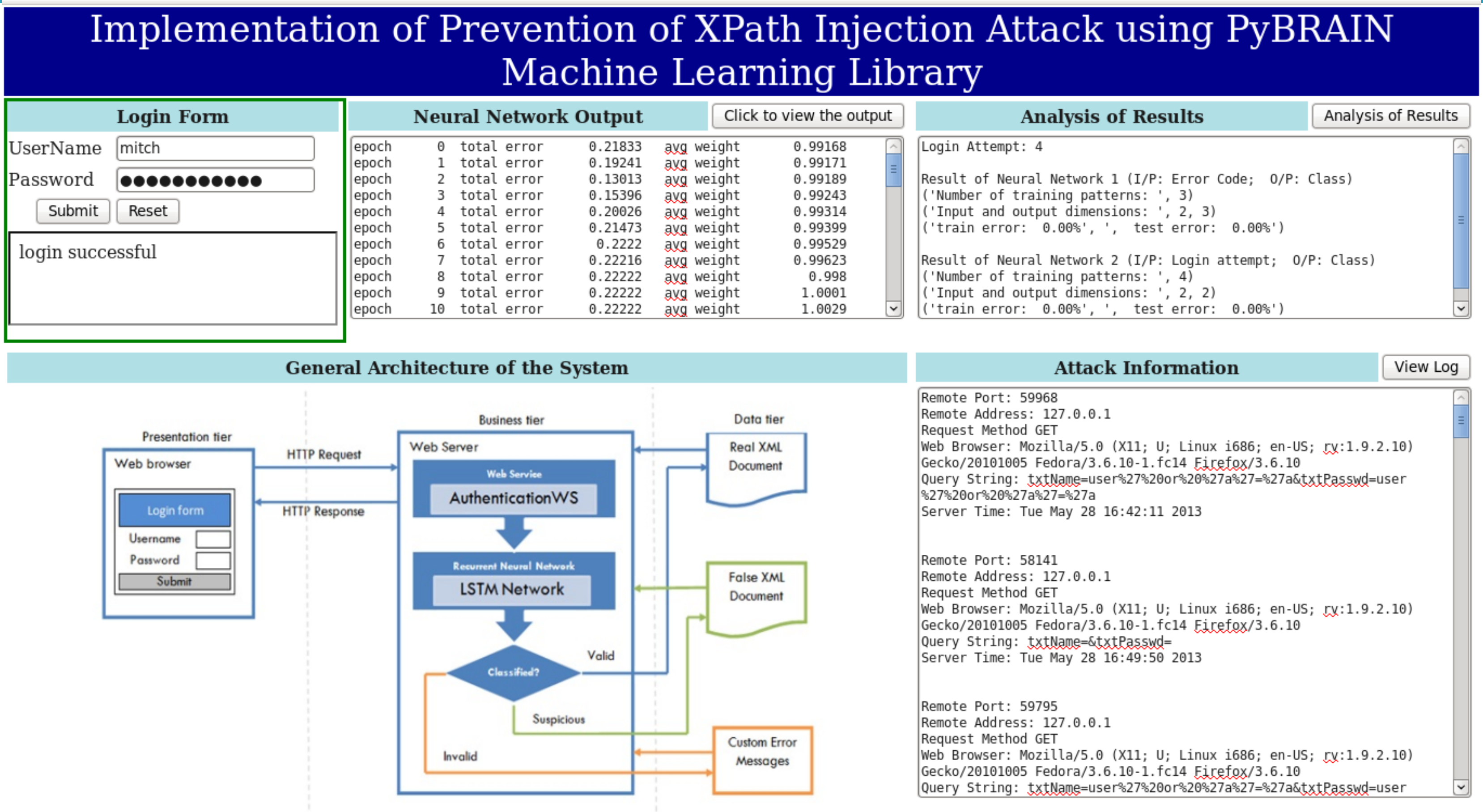

fakedata.xml - KWrite
File Edit View Tools Settings Help
New Open Save Save As Close Undo
<xmlfile>
<users>
<user>
<name first="Jeff" last="Smiley"/>
<id>1</id>
<username>fakejsmiley</username>
<password>fakejsmiley</password>
<phone>fake123-456-7890</phone>
</user>
<user>
<name first="Chunk" last="MacRunfast"/>
<id>2</id>
<username>fakecmacrunfast</username>
<password>fakespeedsupto3</password>
<phone>fake603-478-4115</phone>
</user>
<user>
<name first="Mitch" last="Hedberg"/>
<id>3</id>
<username>fakemitch</username>
<password>fake@lltogether</password>
<phone>fake222-222-2222</phone>
</user>
</users>
<ssns>
<ssn>
<username>fakejsmiley</username>
<number>fake111-22-3333</number>
</ssn>
<ssn>
<username>fakecmacrunfast</username>
<number>fake123-45-6789</number>
</ssn>
<ssn>
Line: 1 Col: 1 INS LINE XML fakedata.xml


data.xml - KWrite
File Edit View Tools Settings Help
New Open Save Save As Close
<xmlfile>
<users>
<user>
<name first="Jeff" last="Smiley"/>
<id>1</id>
<username>jsmiley</username>
<password>jsmiley</password>
<phone>123-456-7890</phone>
</user>
<user>
<name first="Chunk" last="MacRunfast"/>
<id>2</id>
<username>cmacrunfast</username>
<password>speedsupto3</password>
<phone>603-478-4115</phone>
</user>
<user>
<name first="Mitch" last="Hedberg"/>
<id>3</id>
<username>mitch</username>
<password>@lltogether</password>
<phone>222-222-2222</phone>
</user>
</users>
<ssns>
<ssn>
<username>jsmiley</username>
<number>111-22-3333</number>
</ssn>
<ssn>
<username>cmacrunfast</username>
<number>123-45-6789</number>
</ssn>
<ssn>
Line: 47 Col: 1 INS LINE XML data.xml

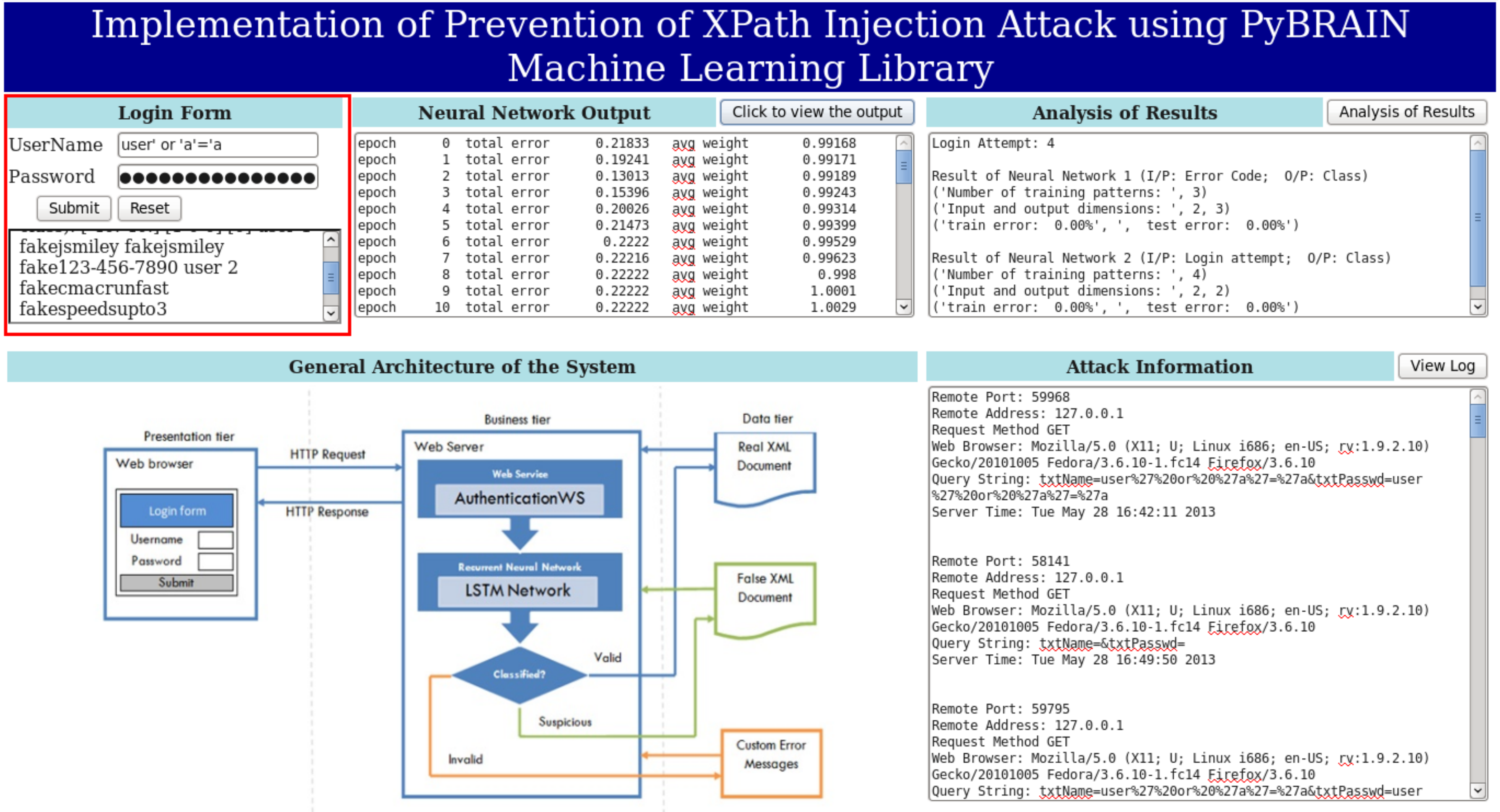
Snapshots (initial output)



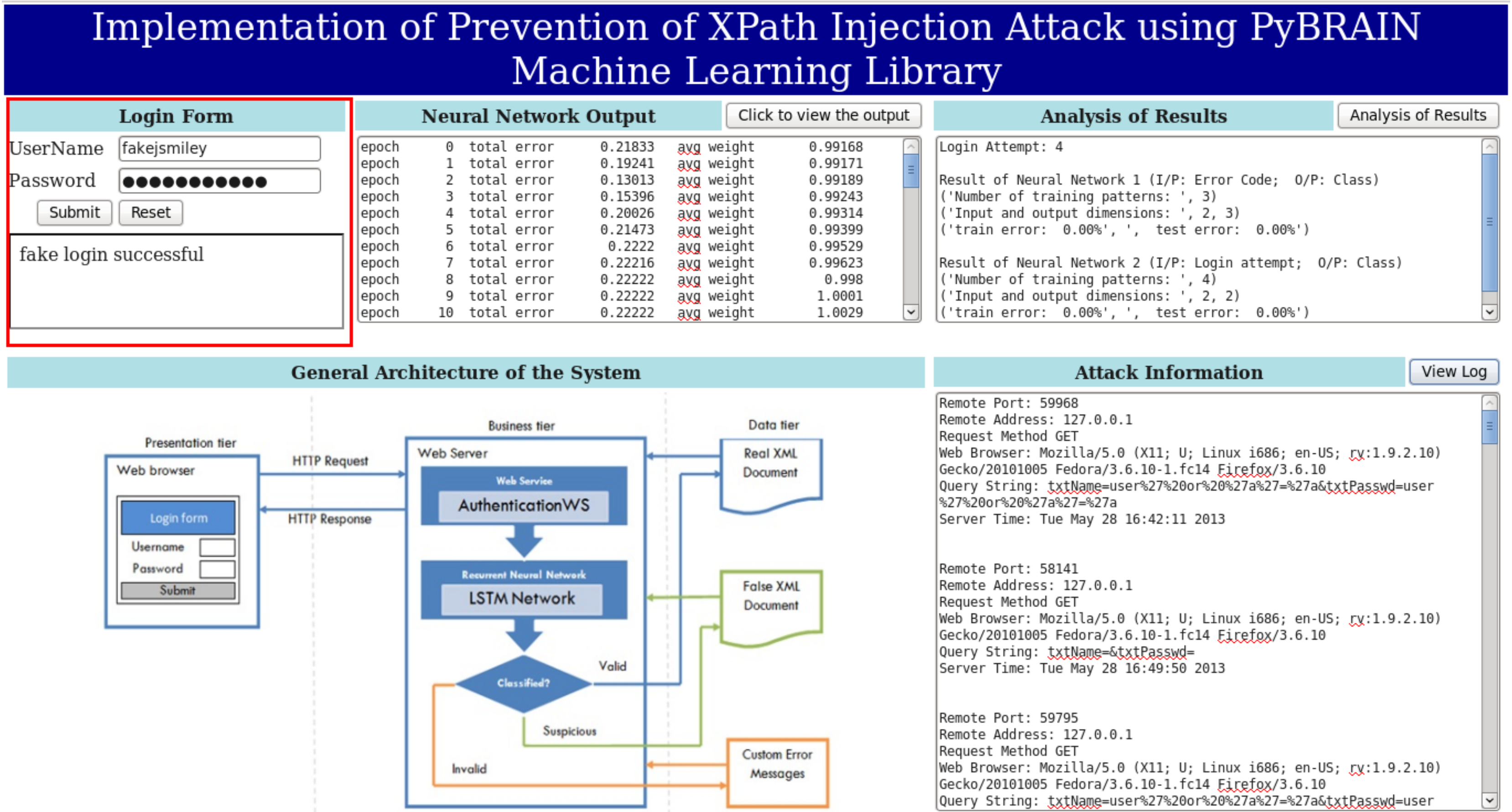
Snapshots (valid input scenario)



Snapshots (malicious input scenario)



Snapshots (fake login scenario)



Conclusion

- Our solution offers improved security over existing methods by misleading the attackers to false resources and custom error pages
- Our results also show that the system accepts legitimate input although the user input may contain some special characters and rejects only truly malicious inputs.
- Our solution combines modular neural networks and count based validation approach to filter the malicious input
- Our solution has resulted in increased average detection rate of true positives and true negatives and decreased average detection rate of false positives and false negatives
- The security systems have to be successful every time. But attacker has to be successful only once.

References

[1] **Thiago Mattos Rosa, Altair Olivo Santin, Andreia Malucelli, “Mitigating XML Injection Attack through Strategy based Detection System”, IEEE Security and Privacy, 2011**

[2] Nuno Antunes, Nuno Laranjeiro, Marco Vieira, Henrique Madeira, “Effective Detection of SQL/XPath Injection Vulnerabilities in Web Services”, IEEE International Conference on Services Computing, 2009

[3]Nuno Laranjeiro, Marco Vieira, Henrique Madeira, “A Learning Based Approach to Secure Web Services from SQL/XPath InjectionAttacks”, Pacific Rim International Symposium on Dependable Computing, 2010

[4] V. Shanmughaneethi, R. Ravichandran, S. Swamynathan, “PXpathV: Preventing XPath Injection Vulnerabilities in Web Applications”, International Journal on Web Service Computing, Vol.2, No.3, September 2011

[5] CAPEC-83: XPath Injection, http://capec.mitre.org/data/definitions/83.html [Accessed on: 02/12/2012]

[6] Mike W. Shields, Matthew C. Casey, “A theoretical framework for multiple neural network systems”, 2008

[7] Hanh H. NguyenÆ Christine W. Chan, “Multiple neural networks for a long term time series forecast”, Springer, Neural Comput & Applic (2004) 13: 90–98

[8] Anand, R., Mehrotra, K., Mohan C.K., Ranka S., "Efficient classification for multiclass problems using modular neural networks", IEEE Transactions on Neural Networks, Volume 6, Issue 1, 1995

References

[9] S. Hochreiter and J. Schmidhuber. "Long short-term memory. Neural Computation", 9 (8): 1735–1780, 1997.

[10] Derek D. Monner, James A. Reggia, "A generalized LSTM-like training algorithm for second-order recurrent neural networks"

[11] Anders Jacobsson, Christian Gustavsson, "Prediction of the Number of Residue Contacts in Proteins Using LSTM Neural Networks", Technical report, IDE0301, January 2003

[12] P.A. Mastorocostas, "Resilient back propagation learning algorithm for recurrent fuzzy neural networks", ELECTRONICS LETTERS, Vol. 40 No. 1, 2004

[13] Martin Riedmiller, Rprop – Description and Implementation Details, Technical report, 1994

[14] Tom Schaul, Justin Bayer, Daan Wierstra, Sun Yi, Martin Felder, Frank Sehnke, Thomas Rückstieß, Jürgen Schmidhuber. "PyBrain", Journal of Machine Learning Research, 2010

[15] Bottle: Python Web Framework, http://bottlepy.org/docs/dev/ [Accessed on: 05/04/2013]

[16] matplotlib, http://matplotlib.org/contents.html, [Accessed on: 06/07/2013]

Thank You

Widescreen Test Pattern (16:9)

Aspect Ratio Test

(Should appear circular)

4x3

16x9