pycononlineau

Deceptive security using Python

GAJENDRA DESHPANDE

KLS Gogte Institute of Technology, India

https://gcdeshpande.github.io

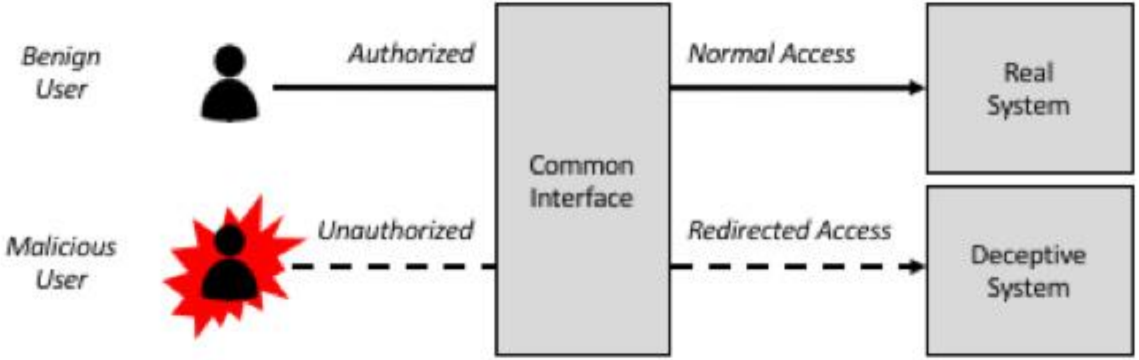
Contents

Introduction

Imagine you are passing through an unknown street at midnight and you find that some anti-social elements are following you. To save yourself from them you start running and look for a safe place to hide yourself. On the way, you will find a good person and requests him to help you. He hides you in his place to protect you. When these anti-social elements visit a good person’s place and enquire about you, the good person misguides them and redirects them to some other place in order to protect you. This is exactly how deception works. In this analogy, YOU are the resources to be protected, anti-social elements are the hackers who want to gain access to the resources, and a good person is a deception technique that protects the resources from hackers by making them fall in the trap.

Deception – Basic Idea

- Deception is a technique where hackers methods will be used as security mechanism i.e., phishing the phishers.
- Deception is military tactic used by both attackers and defenders.



Source: https://www.helpnetsecurity.com/2018/12/06/introduction-deception-technology/

Deception – Types

There are two types of Deception Technology described below.

- **Active Deception:** Active Deception will provide inaccurate information intentionally to the subjects (intruders or hackers) to fall for the trap.
- **Passive Deception:** Passive Deception will provide incomplete information, other half of information. Intruders will try to gain all the information and the fall for the trap.

Source: https://www.geeksforgeeks.org/deception-technology/

They can also be classified as

- Client side deception – used by hackers
- Server side deception – used by security providers

Better Deception = Active Deception + Passive Deception

Deception – Evolution - Advantages

- HoneyPots (1998) → HoneyNets(2000) → HoneyToken (2003) → HoneyPot 2.0 (2012) →Deception Technology (2016)

- Advantages
  - Increased accuracy
  - Minimal investment
  - Future ready (applicable to new technology)

WebTrap

- Designed to create deceptive webpages to deceive and redirect attackers away from real websites.
- The deceptive webpages are generated by cloning real websites, specifically their login pages.

The project is composed of two tools:

- Web Cloner - Responsible for cloning real websites and creating the deceptive web page
- Deceptive Web server - Responsible for serving the cloned webpages, and reporting to a syslog server upon requests

Installation:

```
pip install requests
apt install gir1.2-webkit2-3.0 python-gi python-gi-cairo python3-gi python3-gi-cairo gir1.2-gtk-3.0
```

https://github.com/IllusiveNetworks-Labs/WebTrap

WebTrap – Web Cloner

```
usage: WebCloner.py [-h] [-o OUTPUT_DIRECTORY] website_url

positional arguments:
  website_url           The URL path to the web page you desire to clone

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT_DIRECTORY, --output-directory OUTPUT_DIRECTORY
                        Setting the output directory for the cloned webpage
```

```
python ./WebCloner.py -o ~/WikiPediaLoginPage/ https://en.wikipedia.org/w/index.php?title=Special:UserLogin
```

WebTrap – Deceptive Web Server

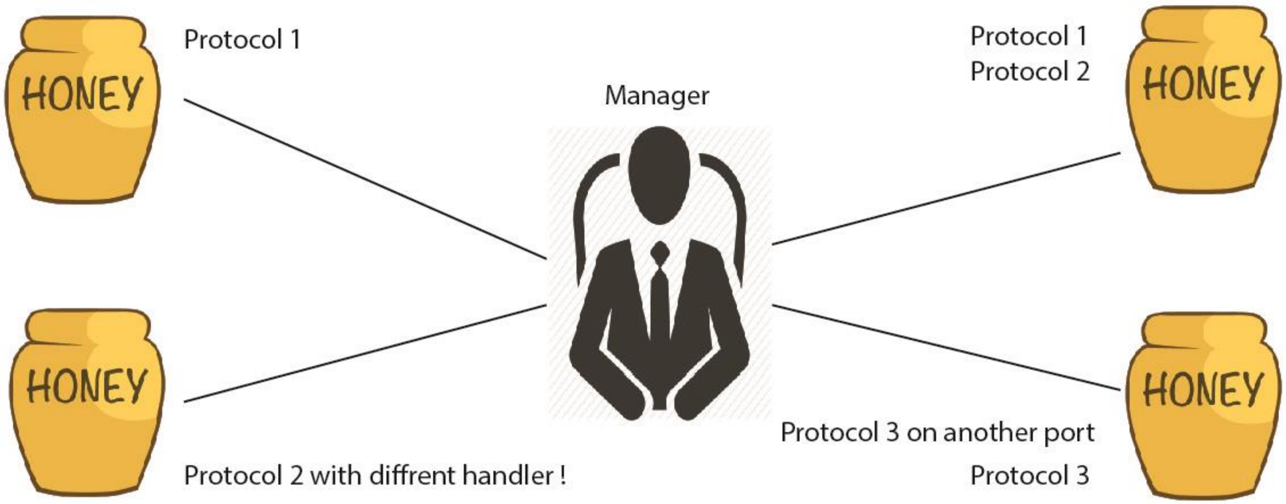
```
usage: TrapServer.py [-h] [--webroot-directory WEBROOT_DIRECTORY]
                     [--syslog-server SYSLOG_SERVER]
                     [--log-file-path LOG_FILE_PATH]

optional arguments:
  -h, --help            show this help message and exit
  --webroot-directory WEBROOT_DIRECTORY, -d WEBROOT_DIRECTORY
                        root directory for the HTTP server
  --syslog-server SYSLOG_SERVER, -s SYSLOG_SERVER
                        syslog server that the deceptive user will report the
                        request to it
  --log-file-path LOG_FILE_PATH, -l LOG_FILE_PATH
                        access log file path
```

```
sudo python ./TrapServer.py -d ~/WikiPediaLoginPage/ -s <SYSLOG_SERVER>
```

DemonHunter

- To create low interaction Honeypot servers and their agents, plus a manager to check logs
- DemonHunter allows you to create your honeynet all customized by yourself, from ports to protocol handlers.



https://github.com/skrtu/DemonHunter

Why we developed deception tool

- 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

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

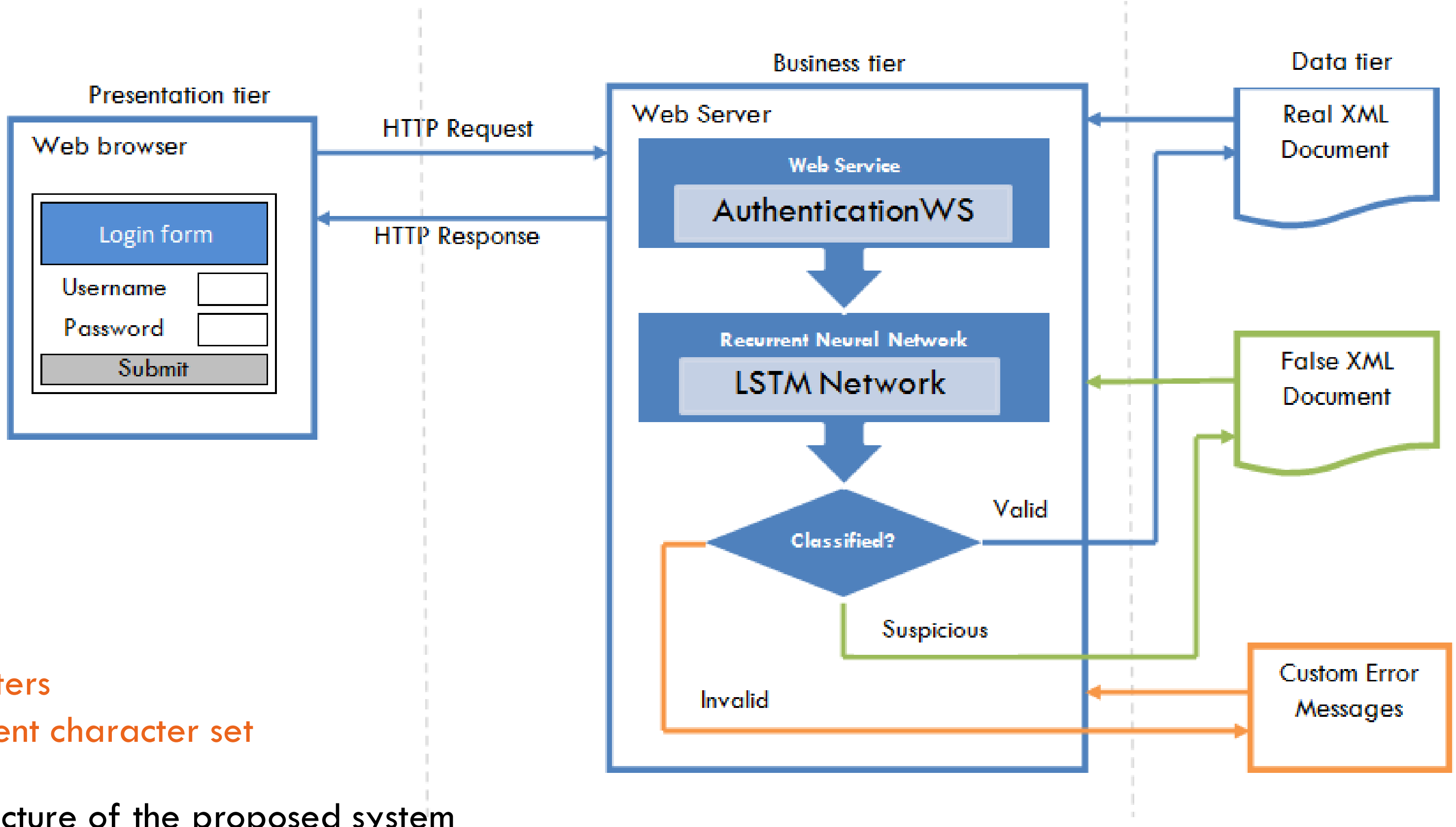


Fig. 1: Three tier architecture of the proposed system

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)

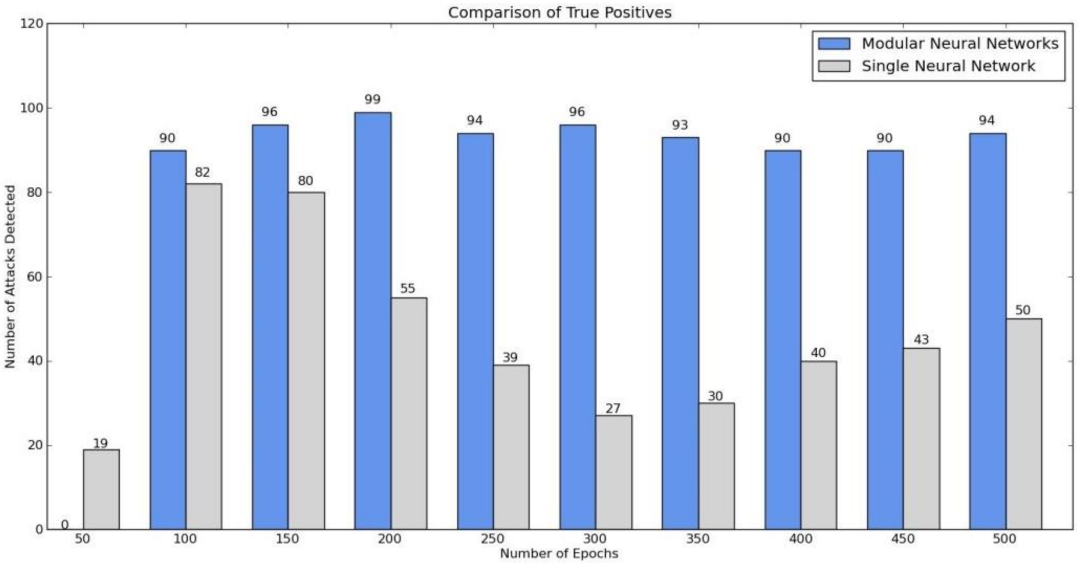


Fig. 2: Comparison of true positives

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)

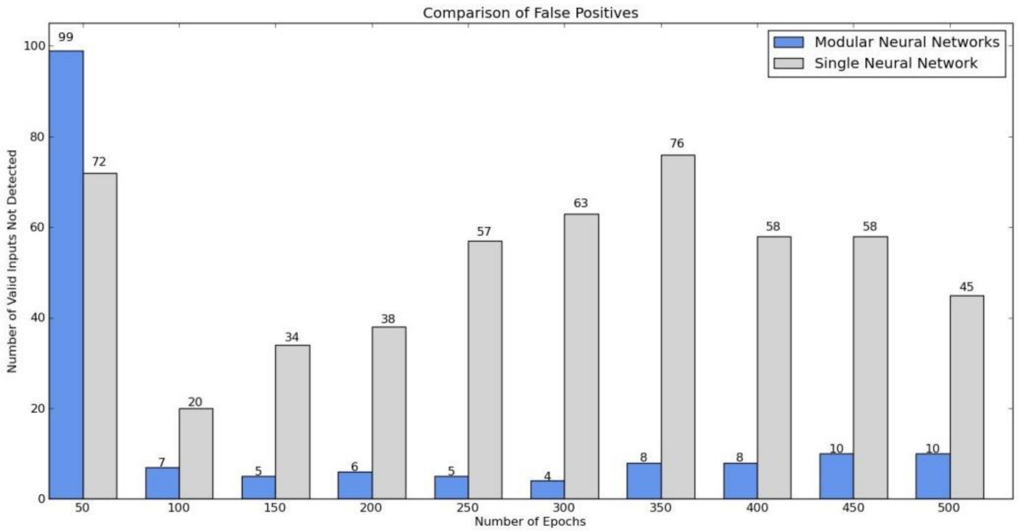


Fig. 3: Comparison of false positives

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)

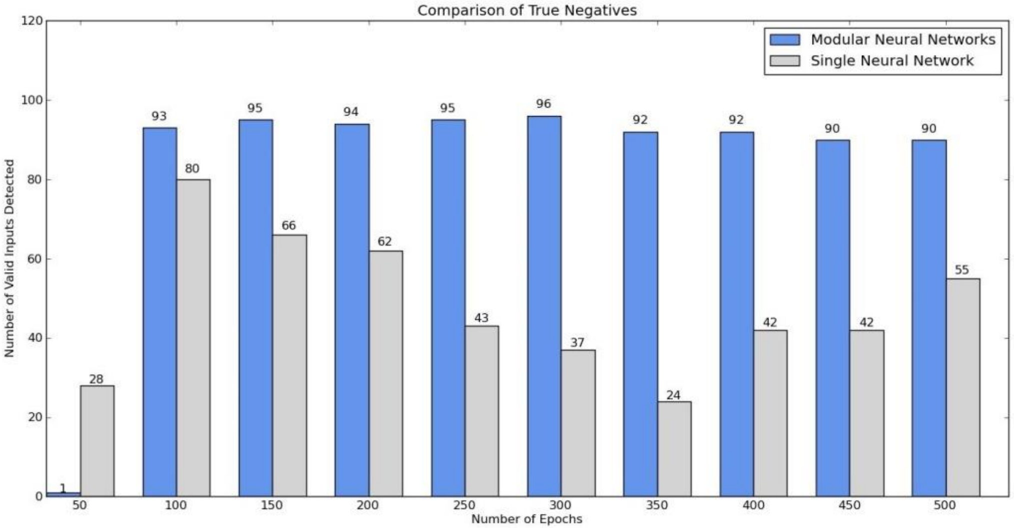


Fig. 4: Comparison of true negatives

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)

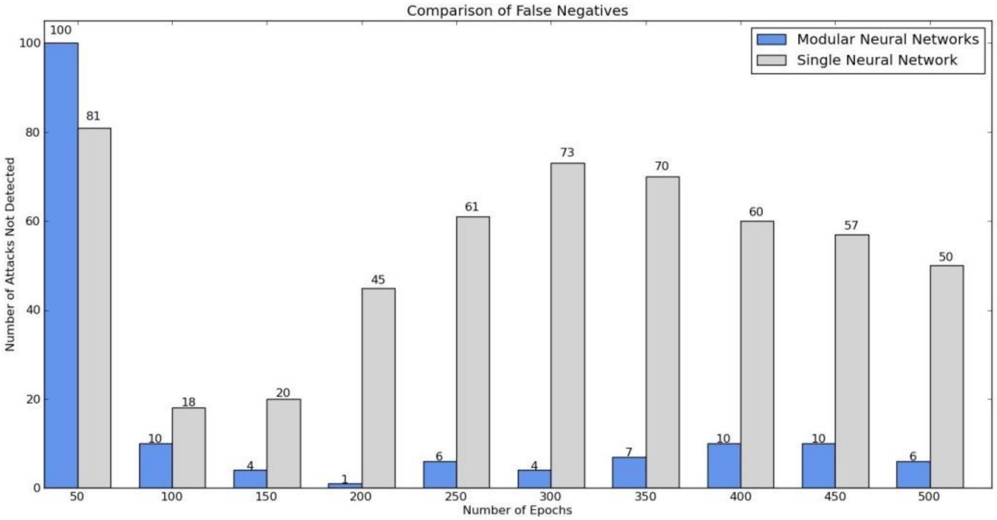


Fig. 5: Comparison of false negatives

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)

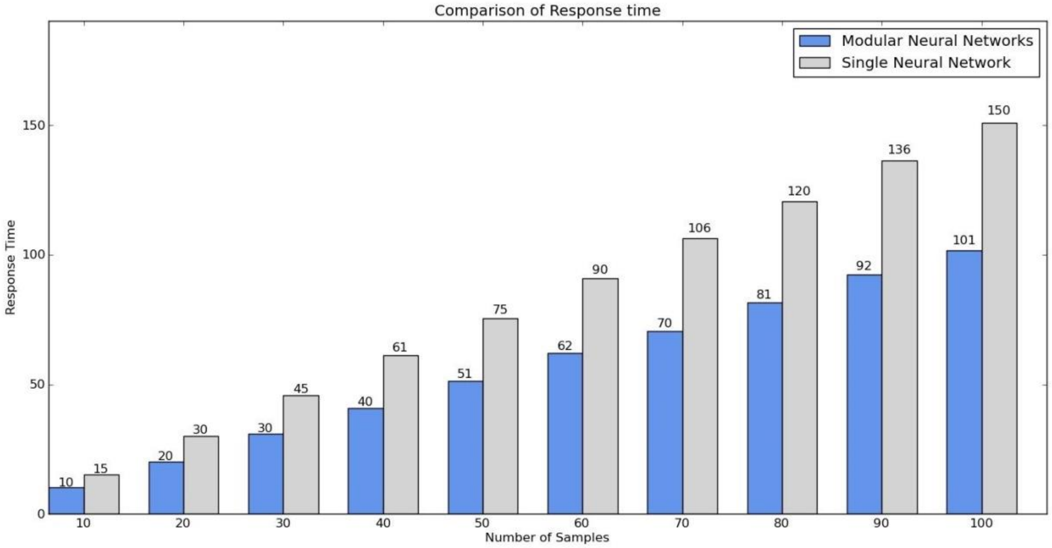


Fig. 6: Comparison of response time

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

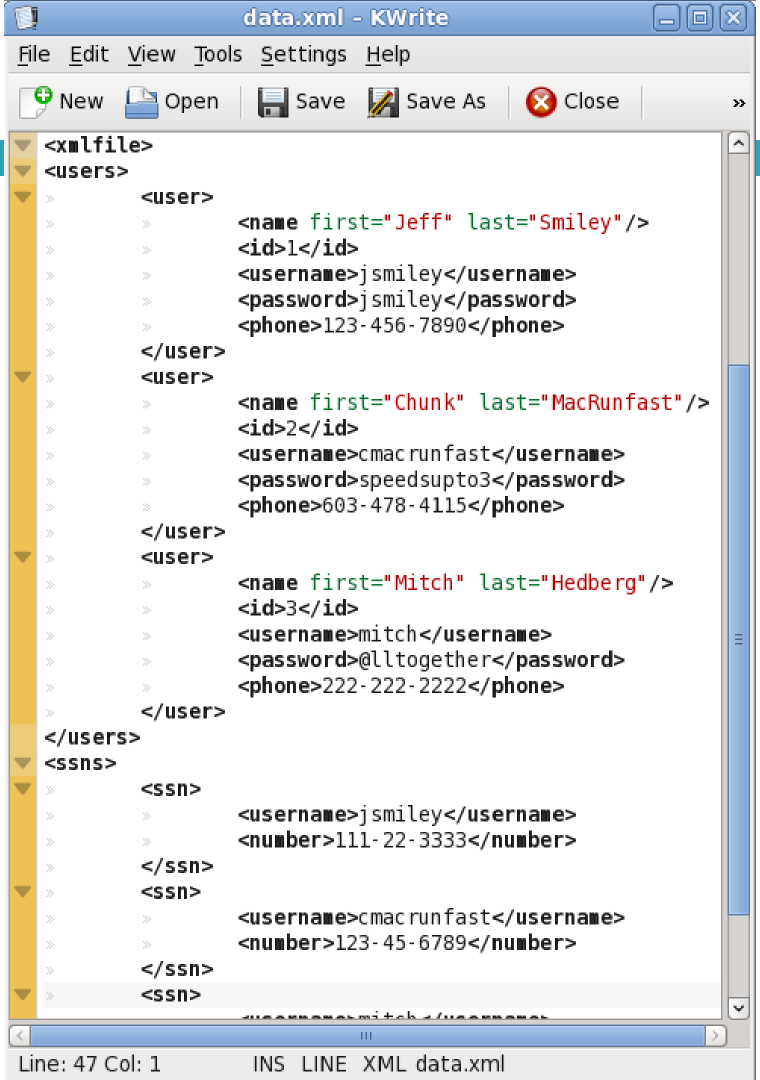
Summary of Results

Table 11: Average detection rate including and excluding an outlier

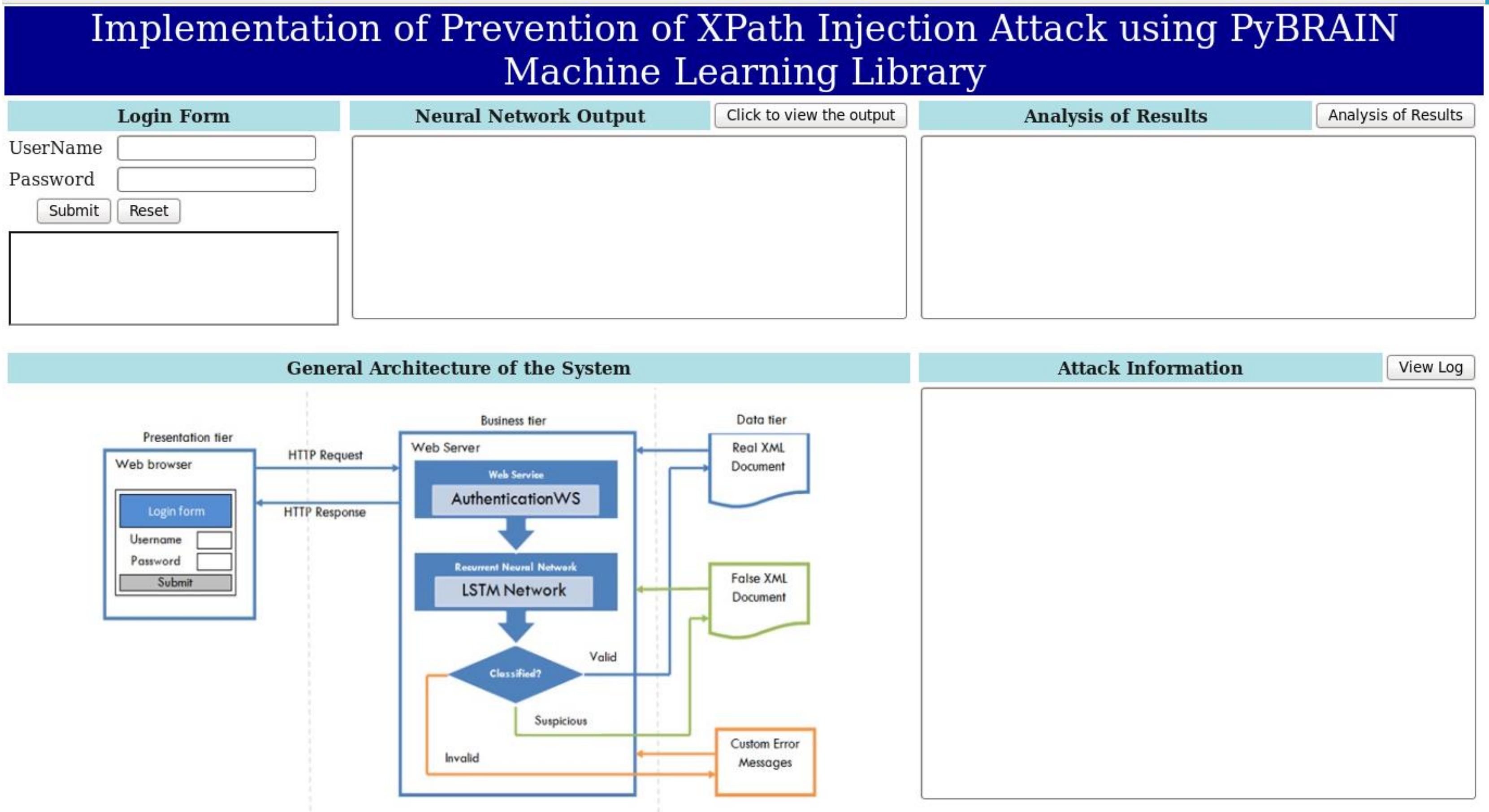
	Average detection rate including an outlier		Average detection rate excluding an outlier	
	MNN %	SNN %	MNN %	SNN %
True Positives	84.2	46.5	93.55	51.66
False Negatives	15.8	53.5	6.45	48.33
True Negatives	83.8	47.9	93.11	53.22
False Positives	16.2	52.1	6.88	46.77

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)

Implementation of Prevention of XPath Injection Attack using PyBRAIN Machine Learning Library

Login Form

UserName: mitch

Password: ●●●●●●●●●●●

Submit | Reset

login successful

Neural Network Output

Click to view the output

```
epoch     0  total error     0.21833   avg weight     0.99168
epoch     1  total error     0.19241   avg weight     0.99171
epoch     2  total error     0.13013   avg weight     0.99189
epoch     3  total error     0.15396   avg weight     0.99243
epoch     4  total error     0.20026   avg weight     0.99314
epoch     5  total error     0.21473   avg weight     0.99399
epoch     6  total error      0.2222   avg weight     0.99529
epoch     7  total error     0.22216   avg weight     0.99623
epoch     8  total error     0.22222   avg weight       0.998
epoch     9  total error     0.22222   avg weight      1.0001
epoch    10  total error     0.22222   avg weight      1.0029
```

Analysis of Results

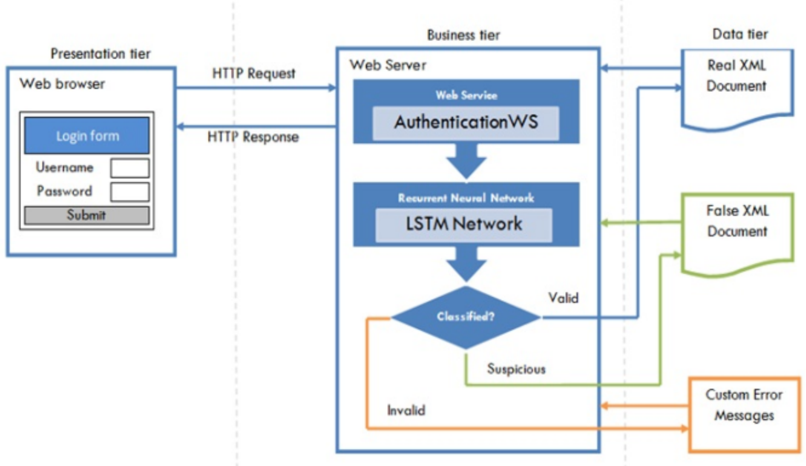
Analysis of Results

```
Login Attempt: 4

Result of Neural Network 1 (I/P: Error Code;  O/P: Class)
('Number of training patterns: ', 3)
('Input and output dimensions: ', 2, 3)
('train error:  0.00%', ',  test error:  0.00%')

Result of Neural Network 2 (I/P: Login attempt;  O/P: Class)
('Number of training patterns: ', 4)
('Input and output dimensions: ', 2, 2)
('train error:  0.00%', ',  test error:  0.00%')
```

General Architecture of the System



Attack Information

View Log

```
Remote Port: 59968
Remote Address: 127.0.0.1
Request Method GET
Web Browser: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.10)
Gecko/20101005 Fedora/3.6.10-1.fc14 Firefox/3.6.10
Query String: txtName=user%27%20or%20%27a%27=%27a&txtPasswd=user
%27%20or%20%27a%27=%27a
Server Time: Tue May 28 16:42:11 2013


Remote Port: 58141
Remote Address: 127.0.0.1
Request Method GET
Web Browser: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.10)
Gecko/20101005 Fedora/3.6.10-1.fc14 Firefox/3.6.10
Query String: txtName=&txtPasswd=
Server Time: Tue May 28 16:49:50 2013


Remote Port: 59795
Remote Address: 127.0.0.1
Request Method GET
Web Browser: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.10)
Gecko/20101005 Fedora/3.6.10-1.fc14 Firefox/3.6.10
Query String: txtName=user%27%20or%20%27a%27=%27a&txtPasswd=user
```

Snapshots (malicious input scenario)

Implementation of Prevention of XPath Injection Attack using PyBRAIN Machine Learning Library

Login Form

UserName: user' or 'a'='a

Password: ●●●●●●●●●●●●●●●

Submit | Reset

fakejsmiley fakejsmiley
fake123-456-7890 user 2
fakecmacrunfast
fakespeedsupto3

Neural Network Output

Click to view the output

```
epoch    0  total error    0.21833   avg weight    0.99168
epoch    1  total error    0.19241   avg weight    0.99171
epoch    2  total error    0.13013   avg weight    0.99189
epoch    3  total error    0.15396   avg weight    0.99243
epoch    4  total error    0.20026   avg weight    0.99314
epoch    5  total error    0.21473   avg weight    0.99399
epoch    6  total error     0.2222   avg weight    0.99529
epoch    7  total error    0.22216   avg weight    0.99623
epoch    8  total error    0.22222   avg weight      0.998
epoch    9  total error    0.22222   avg weight     1.0001
epoch   10  total error    0.22222   avg weight     1.0029
```

Analysis of Results

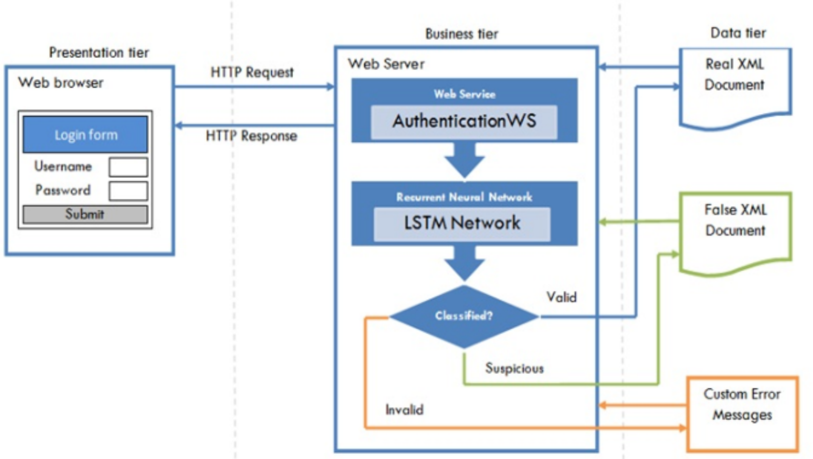
Analysis of Results

```
Login Attempt: 4

Result of Neural Network 1 (I/P: Error Code;  O/P: Class)
('Number of training patterns: ', 3)
('Input and output dimensions: ', 2, 3)
('train error:  0.00%', ',  test error:  0.00%')

Result of Neural Network 2 (I/P: Login attempt;  O/P: Class)
('Number of training patterns: ', 4)
('Input and output dimensions: ', 2, 2)
('train error:  0.00%', ',  test error:  0.00%')
```

General Architecture of the System



Attack Information

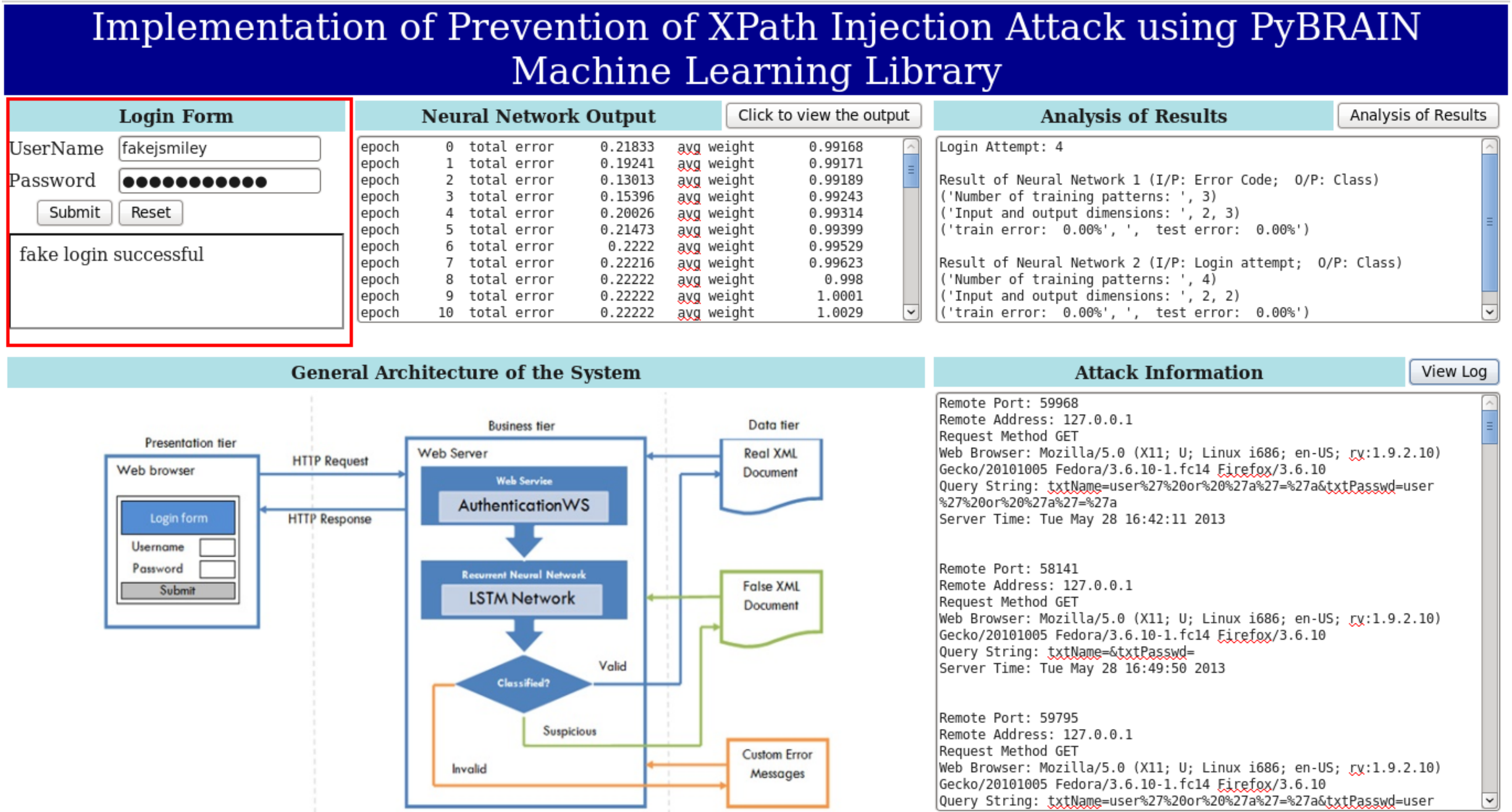
View Log

```
Remote Port: 59968
Remote Address: 127.0.0.1
Request Method GET
Web Browser: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.10)
Gecko/20101005 Fedora/3.6.10-1.fc14 Firefox/3.6.10
Query String: txtName=user%27%20or%20%27a%27=%27a&txtPasswd=user
%27%20or%20%27a%27=%27a
Server Time: Tue May 28 16:42:11 2013


Remote Port: 58141
Remote Address: 127.0.0.1
Request Method GET
Web Browser: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.10)
Gecko/20101005 Fedora/3.6.10-1.fc14 Firefox/3.6.10
Query String: txtName=&txtPasswd=
Server Time: Tue May 28 16:49:50 2013


Remote Port: 59795
Remote Address: 127.0.0.1
Request Method GET
Web Browser: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.10)
Gecko/20101005 Fedora/3.6.10-1.fc14 Firefox/3.6.10
Query String: txtName=user%27%20or%20%27a%27=%27a&txtPasswd=user
```

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

[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/

[16] matplotlib, http://matplotlib.org/contents.html

[17] https://github.com/IllusiveNetworks-Labs/WebTrap

[18] https://github.com/skrtu/DemonHunter

Thank You