Journal of Mobile Multimedia, Vol. 11, No. 3&4 (2015) 251–262

A NEURAL NETWORK BASED INTRUSION DETECTION AND USER IDENTIFICATION SYSTEM FOR TOR NETWORKS: PERFORMANCE EVALUATION FOR DIFFERENT NUMBER OF HIDDEN UNITS USING FRIEDMAN TEST

TARO ISHITAKI, TETSUYA ODA, YI LIU, DONALD ELMAZI
Graduate School of Engineering, Fukuoka Institute of Technology (FIT)
3-30-1, Wajiro-Higashi, Higashi-Ku, Fukuoka 811-0295, Japan
E-Mail: taro.ishitaki@gmail.com, oda.tetsuya.fit@gmail.com, ryuui1010@gmail.com, donald.elmazi@gmail.com

KEITA MATSUO
Fukuoka Prefectural Fukuoka Technical High School
2-19-1 Arae, Sawara-Ku, Fukuoka 814-8520, Japan
E-Mail: matuo-k7@fku.ed.jp

LEONARD BAROLLI
Department of Information and Communication Engineering,
Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-Higashi, Higashi-Ku, Fukuoka 811?0295, Japan
Email: barolli@fit.ac.jp

Due to the amount of anonymity afforded to users of the Tor infrastructure, Tor has become a useful tool for malicious users. With Tor, the users are able to compromise the non-repudiation principle of computer security. Also, the potentially hackers may launch attacks such as DDoS or identity theft behind Tor. For this reason, there are needed new systems and models to detect the intrusion in Tor networks. In this paper, we present the application of Neural Networks (NNs) and Friedman test for intrusion detection and user identification in Tor networks. We used the Back-propagation NN and constructed a Tor server, a Deep Web browser (Tor client) and a Surface Web browser. Then, the client sends the data browsing to the Tor server using the Tor network. We used Wireshark Network Analyzer to get the data and then used the Back-propagation NN to make the approximation. We present many simulation results for different number of hidden units considering Tor client and Surface Web client. The simulation results show that our simulation system has a good approximation and can be used for intrusion detection and user identification in Tor networks.

Keywords: Neural Networks, Friedman Test, User Identification, Intrusion Detection, Tor Networks, Deep Web, Hidden Unit

1 Introduction

Tor (The Onion Router) [1, 2] is an implementation of an Onion Routing network, where users expect a large degree of privacy. This privacy is reflected in the perfect forward secrecy exhibited by Tor connections such that traffic captured at any single network location during transit only uncovers the previous and next waypoints [3]. Due to the amount of anonymity afforded to users of the Tor infrastructure, Tor has become a useful tool for malicious users. With Tor, the users are able to compromise the non-repudiation principle of computer security. Also, the potentially hackers may launch attacks such as DDoS or identity theft behind Tor.

The Tor has been designed to make it possible for users to surf the Internet anonymously, so their activities and location cannot be discovered by government agencies, corporations, or anyone else. Compared with other anonymizers Tor is more popular and has more visibility in the academic and hacker communities. Tor is a low-latency, circuit-based and privacy-preserving anonymizing platform and network. It is one of several systems that have been developed to provide Internet users with a high level of privacy and anonymity in order to cope with the censorship measures taken by authorities and to protect against the constantly increasing threats to these two key security properties.

There are two main approaches to the design of Intrusion Detection Systems (IDSs). In a misuse detection based IDS, intrusions are detected by looking for activities that correspond to known signatures of intrusion or vulnerabilities. On the other hand, anomaly detection based IDS detects intrusions by searching for abnormal network traffic. The abnormal traffic pattern can be defined either as the violation of accepted thresholds for the legitimate profile developed for the normal behavior.

In [4], the authors designed and implemented TorWard, which integrates an Intrusion Detection System (IDS) at Tor exit routers for Tor malicious traffic discovery and classification. The system can avoid legal and administrative complaints and allows the investigation to be performed in a sensitive environment such as a university campus. An IDS is used to discover and classify malicious traffic. The authors performed comprehensive analysis and extensive real-world experiments to validate the feasibility and effectiveness of TorWard.

One of the most commonly used approaches in expert system based on intrusion detection is a rule-based analysis using soft computing techniques such Fuzzy Logic (FL), Artificial Neural Networks (ANNs), Probabilistic Reasoning (PR), and Genetic Algorithms (GAs). They are good approaches capable of finding patterns for abnormal and normal behavior. In some studies, the neural networks have been implemented with the capability to detect normal and attack connections [5].

In [6], a specific combination of two Neural Network (NN) learning algorithms, the Error Back-propagation and the Levenberg-Marquardt algorithm, is used to train an artificial NN to model the boundaries of the clusters of recorded normal behavior. It is shown that the training dataset, consisting of a combination of recorded normal instances and artificially generated intrusion instances, successfully guides the NN towards learning the complex and irregular cluster boundary in a multidimensional space. The performance of the system is tested on unseen network data containing various intrusion attacks [6].

In [7] is presented a NN-based intrusion detection method for the internet-based attacks on a computer network. The IDSs have been created to predict and thwart current and future attacks. The NNs are used to identify and predict unusual activities in the system. In particular, feed-forward NNs with the Back-propagation training algorithm were employed and the training and testing data were obtained from the Defense Advanced Research Projects Agency (DARPA) intrusion detection evaluation data sets. The experimental results on real-data showed promising results on detection intrusion systems using NNs.

In [8], the authors use a Back-propagation Artificial Neural Network (ANN) to learn system's behavior. The authors used the KDD'99 data set for experiments and they obtained satisfying results.

In this paper, we present the application of NNs and Friedman test for intrusion detection

and user identification in Tor networks. We used the Back-propagation NN and constructed a Tor server and a Deep Web browser (client) in our laboratory. Then, the client sends the data browsing to the Tor server using the Tor network. We used Wireshark Network Analyzer to get the data and then use the Back-propagation NN to make the approximation. We present many simulation results for different number of hidden units considering Tor client and Surface Web client.

The structure of the paper is as follows. In Section 2, we present a short description of Deep Web and Tor. In Section 3, we give an overview of ANNs. In Section 4, we present an overview of R. In Section 5, we present the proposed model. In Section 6, we give a brief introduction of Friedman test. In Section 7, we discuss the simulation results. Finally, conclusions and future work are given in Section 8.

2 Deep Web and Tor Overview

2.1 Deep Web

The Deep Web (also called the Deepnet, Invisible Web or Hidden Web) is the portion of World Wide Web content that is not indexed by standard search engines [9, 10]. Most of the Web's information is far from the search sites and standard search engines do not find it. Traditional search engines cannot see or retrieve content in the Deep Web. The portion of the Web that is indexed by standard search engines is known as the Surface Web. Now, the Deep Web is several orders of magnitude larger than the Surface Web. The most famous of the deep web browsers is called Tor.

The Deep Web is both surprising and sinister and accounts for in excess of 90% of the overall Internet [11]. The Google and other search engines deal only with the indexed surface web. The deep-dark web hosts illegal markets, such as the Silk Road, malware emporiums, illegal pornography, and covert meeting places and messaging services. The pervasiveness of the Internet provides easy access to dark-web sites from anywhere in the world. The growth of the dark web has been paralleled by an increasing number of anonymity web-overlay services, such as Tor, which allow criminals, terrorists, hackers, paedophiles and the like to shop and communicate with impunity. Law enforcement and security agencies have had only very limited success in combating and containing this dark menace.

2.2 Tor

Tor is a low-latency, circuit-based and privacy-preserving anonymizing platform and network. It is one of several systems that have been developed to provide Internet users with a high level of privacy and anonymity in order to cope with the censorship measures taken by authorities and to protect against the constantly increasing threats to these two key security properties [12, 13, 14, 15].

The Tor main design goals are to prevent attackers from linking communication partners, or from linking multiple communications to or from a single user. Tor relies on a distributed overlay network and onion routing to anonymize TCP-based applications like web browsing, secure shell, or peer-to-peer communications.

The Tor network is composed of the Tor-client, an entry/guard node, several relays and the exit node. The Tor-client is a software, installed on each Tor user's device. It enables user to create a Tor anonymizing circuit and to handle all the cryptographic keys, needed to

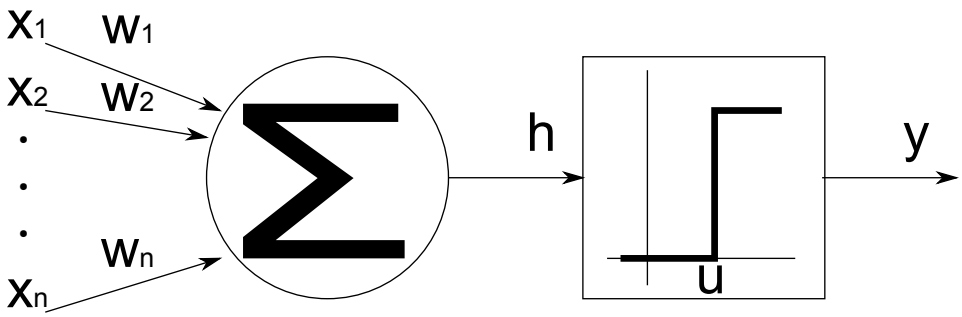


Fig. 1. A neuron model.

communicate with all nodes within the circuit. The Entry Node is the first node in the circuit that receives the client request and forwards it to the second relay in the network. The Exit Node is the last Tor-relay in the circuit. Once the connection request leaves the entry node, it will be forwarded, through relays in the circuit, all the way to the exit node. The latter receives the request and relays it to the final destination.

When a client wants to communicate with a server via Tor, he selects n nodes of the Tor system (where n is typically 3) and builds a circuit using those selected nodes. Messages are then encrypted n times using the following onion encryption scheme. The messages are first encrypted with the key shared with the last node (called the exit node of the circuit) and subsequently with the shared keys of the intermediate node. As a result of this onion routing, each intermediate node only knows its predecessor and successor, but no other nodes of the circuit. In addition, the onion encryption ensures that only the last node is able to recover the original message.

A Tor client typically uses multiple simultaneous circuits. As a result, all streams of a user are multiplexed over these circuits. For example, a BitTorrent user can use one of the circuits for his connections to the tracker and other circuits for his connections to the peers.

3 Artificial Neural Network (ANN)

3.1 Computational Models of Neurons

A computational model for an artificial neuron is shown in Fig. 1. This mathematical neuron computes a weighted sum of its n input signals, x_j, where j =1, 2, ..., n, and generates an output of 1 if this sum is above a certain threshold u [16, 17, 18]. Otherwise, an output of 0 results. Mathematically,

$$y = \theta\left(\sum_{j=1}^{n} w_j x_j - u\right), \tag{1}$$

where $\theta()$ is a unit step function at 0, and w_j, is the synapse weight associated with the jth input. For simplicity of notation, the threshold u is considered as another weight $w_0 = -u$ attached to the neuron with a constant input $x_0 = 1$. Positive weights correspond to excitatory synapses, while negative weights model inhibitory ones. In principle, suitably chosen weights let a synchronous arrangement of such neurons perform universal computations. There is

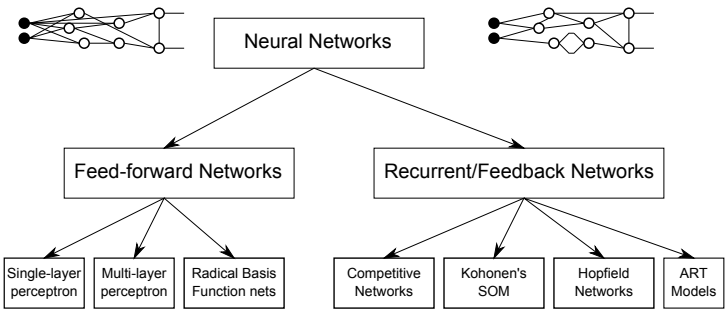


Fig. 2. A taxonomy of feed-forward and recurrent/feedback network architectures.

a crude analogy here to a biological neuron: wires and interconnections model axons and dendrites, connection weights represent synapses, and the threshold function approximates the activity in a soma. This model contains a number of simplifying assumptions that do not reflect the true behavior of biological neurons. An obvious way is to use activation functions other than the threshold function, such as piecewise linear, sigmoid, or Gaussian.

3.2 Network Architectures

The ANNs can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron inputs. Based on the connection pattern (architecture), ANNs can be grouped into two categories (see Figure 2): feed-forward networks, in which graphs have no loops and recurrent (or feedback) networks, in which loops occur because of feedback connections.

In the most common family of feed-forward networks, called multilayer perceptron, neurons are organized into layers that have unidirectional connections between them. Different connectivities yield different network behaviors. The feed-forward networks are static, because they produce only one set of output values rather than a sequence of values from a given input. The feedforward networks are memory-less in the sense that their response to an input is independent of the previous network state.

Recurrent, or feedback, networks, on the other hand, are dynamic systems. When a new input pattern is presented, the neuron outputs are computed. Because of the feedback paths, the inputs to each neuron are then modified, which leads the network to enter a new state. Different network architectures require appropriate learning algorithms.

3.3 Learning

The ability to learn is a fundamental trait of intelligence. Although a precise definition of learning is difficult to formulate, a learning process in the ANN context can be viewed as the problem of updating network architecture and connection weights so that a network can efficiently perform a specific task. The network usually must learn the connection weights from available training patterns. Performance is improved over time by iteratively updating the weights in the network. ANNs' ability to automatically learn from examples makes them attractive and exciting. Instead of following a set of rules specified by human experts, ANNs appear to learn underlying rules (like input-output relationships) from the given collection of

representative examples. This is one of the major advantages of NNs over traditional expert systems.

To understand or design a learning process, we should have a model of the environment in which a NN operates. We must know what information is available to the network. Also, we must understand how network weights are updated and which learning rules govern the updating process. A learning algorithm refers to a procedure in which learning rules are used for adjusting the weights.

There are three main learning paradigms: supervised, unsupervised, and hybrid. In supervised learning, or learning with a "teacher", the network is provided with a correct answer (output) for every input pattern. Weights are determined to allow the network to produce answers as close as possible to the known correct answers. Reinforcement learning is a variant of supervised learning in which the network is provided with only a critique on the correctness of network outputs, not the correct answers themselves. In contrast, unsupervised learning, or learning without a teacher, does not require a correct answer associated with each input pattern in the training data set. It explores the underlying structure in the data, or correlations between patterns in the data, and organizes patterns into categories from these correlations. Hybrid learning combines supervised and unsupervised learning. Part of the weights are usually determined through supervised learning, while the others are obtained through unsupervised learning.

3.4 Back-propagation Algorithm

One of the most popular ANN algorithms is Back-propagation algorithm. A Back-propagation algorithm can be broken down to four main steps. After choosing the weights of the network randomly, the back propagation algorithm is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

- Feed-forward computation,
- Back-propagation to the output layer,
- Back-propagation to the hidden layer,
- Weight updates.

The algorithm is stopped when the value of the error function has become sufficiently small.

4 The R environment

The R is an integrated suite of software facilities for data manipulation, calculation and graphical display [19]. Among other things it has:

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either directly at the computer or on hardcopy, and

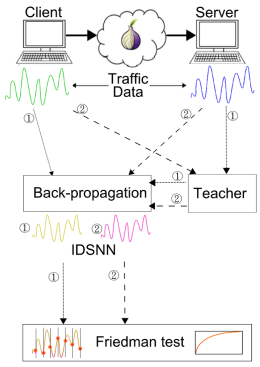


Fig. 3. Proposed system model.

- a well developed, simple and effective programming language (called 'S') which includes conditionals, loops, user defined recursive functions and input and output facilities. Indeed most of the system supplied functions are themselves written in the S language.

The term " environment "is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software. R is very much a vehicle for newly developing methods of interactive data analysis. It has developed rapidly, and has been extended by a large collection of packages. However, most programs written in R are essentially ephemeral, written for a single piece of data analysis.

Many people use R as a statistics system. The R is an environment within which many classical and modern statistical techniques have been implemented. A few of these are built into the base R environment, but many are supplied as packages.

5 Proposed Intrusion Detection and User Identification Model for Tor Networks

The proposed system model is shown in Fig. 3. We call this system: Intrusion Detection System using NN (IDSNN). We used the Back-propagation NN and constructed a Tor server and a Deep Web browser (client) in our laboratory. Then, the client sends the data browsing to the Tor server using the Tor network. We used Wireshark Network Analyzer [20] to get the data and then use the Back-propagation NN to make the approximation.

In this paper, we consider a sinusoidal function for approximation. We train the ANN and then apply the data received from Wireshark. The system runs until the number of loops is achieved.

The data of the Tor client and Surface Web client with the Tor server are compared. The system can detect the intrusion and the bad behavior user by comparing the data at the client site and server site. If the data are similar, this means that we do not have intrusion in the system and detects the bad behavior user. Otherwise, if the data are different, it can be considered that someone attacked the Tor network.

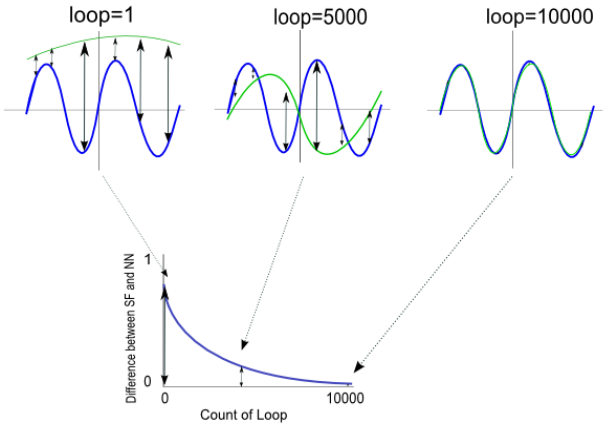


Fig. 4. Calculation method for the difference between SF and NN.

Table 1. Simulation parameters for intrusion detection.

Parameters	Values
Number of hidden layer	1
Number of hidden unit	3,6,9,12,15,18
Learning rate parameter	0.04
Count of loop	10000

6 Friedman Test

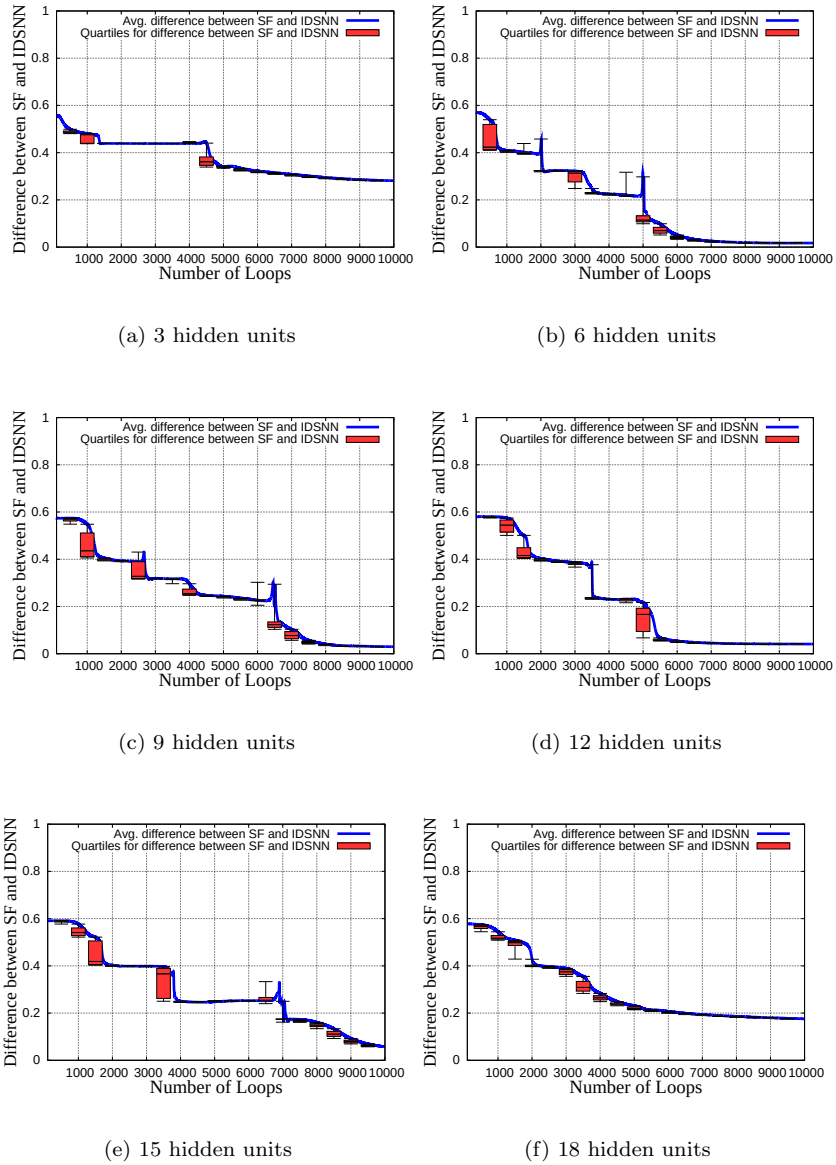
The Friedman test [21] is a nonparametric statistical test of multiple group measures. It can be used to approve the null hypothesis that the multiple group measures have the same variance to a certain required level of significance. On the other hand, failing to approve the null hypothesis shows that they have different variance values. We analyze the difference in performance between server and client considering number of packets using IDSNN and Friedman test in R. We considered as null hypothesis H_0 that there is no difference in the performance between server and client considering number of packets. As alternative hypothesis we considered H_1 that there is difference in the performance of server and client considering number of packets. The significance level in this testing hypothesis is $\alpha = 0.05$. We reject H_0 for $p > \alpha$ (p-value is the probability of obtaining a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesis is true). Further, since there is a correspondence between server and client considering number of packets using IDSNN, we used Friedman test.

7 Simulation Results

7.1 Simulation Results of Intrusion Detection in Tor Networks

The calculation method is shown in Fig. 4. We show the simulation results in Fig. 5. The simulation parameters are shown in Table 1. The traffic values are approximated by the Sinusoidal Function (SF).

We ran the simulation 10,000 times. In Fig. 5 is shown difference between SF and IDSNN



(a) 3 hidden units

(b) 6 hidden units

(c) 9 hidden units

(d) 12 hidden units

(e) 15 hidden units

(f) 18 hidden units

Fig. 5. Difference between Sin function and Back-propagation NN.

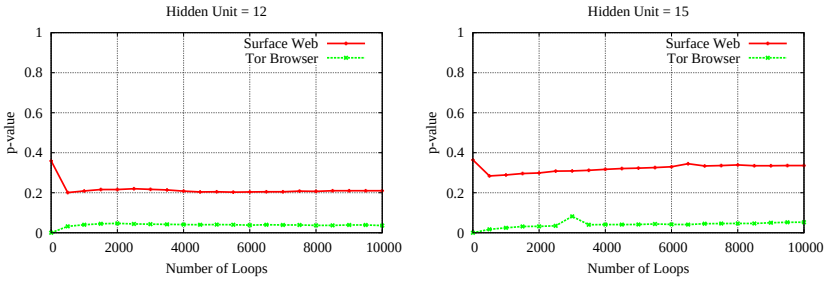
vs. the number of loops. From the results, we can see that 9, 12 and 15 hidden units have almost the same performance. But for 6 hidden units, the difference between SF and NN is the smallest.

7.2 Simulation Results of User Identification in Tor Networks

We show the simulation results in Fig. 6. The simulation parameters are shown in Table 2. We ran the simulation 10,000 times. In Fig. 6 is shown the p-value of Friedman test for

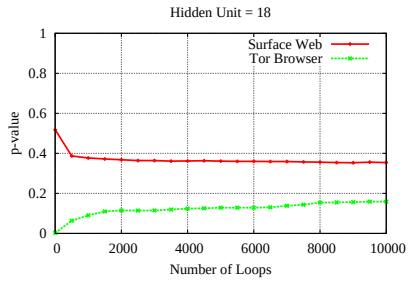
Table 2. Simulation parameters for user identification.

Parameters	Values
Number of hidden layers	1
Number of hidden units	12,15,18
Learning rate parameter	0.04
Number of loops	10000



(a) 12 hidden units

(b) 15 hidden units



(c) 18 hidden units

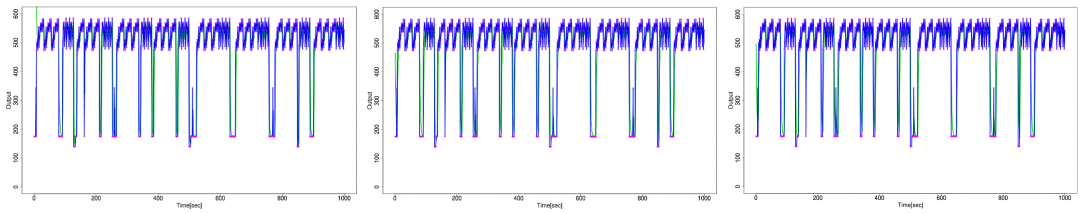
Fig. 6. The p-value of Friedman test for number of packets using IDSNN.

IDSNN considering number of packets. The p-value shows the difference between server and client. The p-values for Tor client are $p < 0.05$. In this case, we adopt H_0. For Surface Web client, we adopt H_1 since $p > 0.05$. From the results, we see that for all number of hidden units the system can identify Tor client. But for 18 hidden units, the p-value is the highest.

In Fig. 7, Fig. 8, Fig. 9, we show the initial settings, middle running and final running results, respectively. As can be seen from the simulation results, the results of Tor server are almost the same with Tor client, but they are different with Surface Web client.

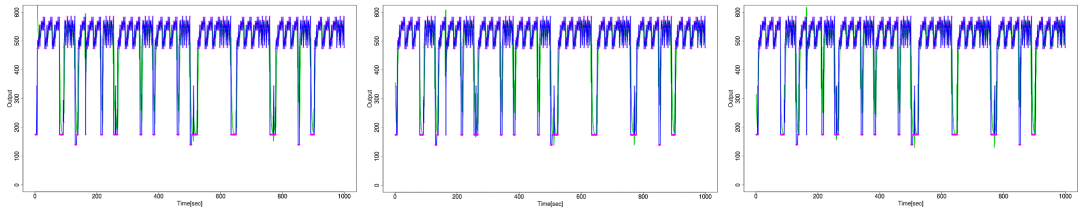
8 Conclusions

Tor network has become a useful tool for malicious users. With Tor, the users are able to compromise the non-repudiation principle of computer security. Also, the potentially hackers may launch attacks such as DDoS or identity theft behind Tor. For this reason, there are needed new systems and models to detect the intrusion in Tor networks. In this paper,



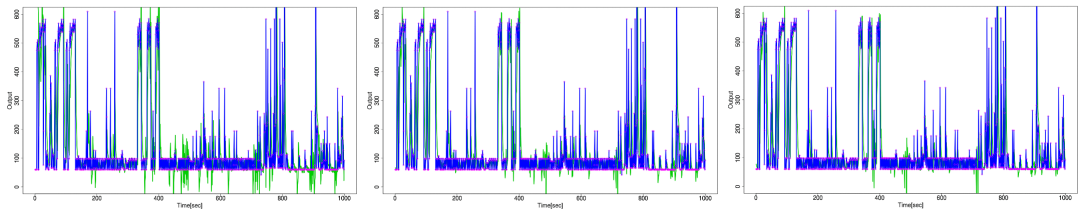
(a) Initial Setting (b) Middle Solution (c) Final Solution

Fig. 7. Simulation results of IDSNN considering Tor server.



(a) Initial Setting (b) Middle Solution (c) Final Solution

Fig. 8. Simulation results of IDSNN considering Tor client.



(a) Initial Setting (b) Middle Solution (c) Final Solution

Fig. 9. Simulation results of IDSNN considering Surface Web client.

we presented the application of NNs and Friedman test for intrusion detection and user identification in Tor networks. We used the Back-propagation NN and constructed a Tor server and a Deep Web browser. Then, the client sent the data browsing to the Tor server using the Tor network. We used Wireshark Network Analyzer to get the data and then useed the Back-propagation NN to make the approximation. The simulation results have shown that our simulation system has a good approximation and can be used for intrusion detection and user identification in Tor networks.

References

1. "Tor project web site," http://www. torproject.org/.
2. R. Dingledine, N. Mathewson, and P. Syverson, "Deploying low-latency anonymity: Design challenges and social factors," *IEEE Security Privacy*, vol. 5, no. 5, pp. 83–87, September 2007.
3. R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," *Proc. of the 13th conference on USENIX Security Symposium (SSYM-2004)*, vol. 13, p. 21, 2004.
4. Z. Ling, J. Luo, K. Wu, W. Yu, and X. Fu," *Proc. of IEEE INFOCOM 2014*, pp. 1402–1410, 2014.
5. E. K. Reddy, "Neural networks for intrusion detection and its applications," *Proc. of the World Congress on Engineering 2013 Vol. II (WCE-2013)*, July 2013.
6. O. Linda, T. Vollmer, and M. Manic, "Neural network based intrusion detection system for critical infrastructures," *Proc. of International Joint Conference on Neural Networks (IJCNN-2009)*, pp. 1827–1834, June 2009.
7. J. Shum, H. A. Malki, "Network intrusion detection system using neural networks," *Proc. of Fourth International Conference on Natural Computation (ICNC-2008)*, pp. 242–246, October 2008.
8. S. T. F. Al-Janabi, H. A. Saeed, "A neural network based anomaly intrusion detection system," *Developments in E-systems Engineering (DeSE-2011)*, pp. 221–226, December 2011.
9. Jer Lang Hong, "Deep web data extraction," *In Proc. of IEEE International Conference on Systems Man and Cybernetics (SMC-2010)*, pp. 3420–3427, October 2010.
10. M. P. Singh, "Deep web structure," *IEEE Internet Computing*, vol. 6, no. 5, pp. 4–5, 2002.
11. D. Stupples, "Security challenge of tor and the deep web," *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*, p. 14, December 2013.
12. A. Biryukov, "Trawling for tor hidden services: Detection, measurement, deanonymization," *In Proc. of IEEE Symposium on Security and Privacy (SP-2013)*, pp. 80–94, November 2013.
13. P. Dhungel, M. Steiner, I. Rimac, V. Hilt, and K. W. Ross, "Waiting for anonymity: Understanding delays in the tor overlay," *In Proc. of IEEE Tenth International Conference on Peer-to-Peer Computing (P2P-2010)*, pp. 1–4, August 2010.
14. L. Xin, W. Neng, "Design improvement for tor against low-cost traffic attack and low-resource routing attack," *In Proc. of WRI International Conference on Communications and Mobile Computing (CMC '09)*, pp. 549–554, January 2009.
15. P. Syverson, "A peel of onion," *Proc. of ACSAC-2011*, pp. 123–135, December 2011.
16. H. Jaeger, "Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach," *GMD Report 159, German National Research Center for Information Technology*, 2002.
17. S. O. Haykin, *Neural Networks and Learning Machines (3rd Edition)*. Prentice Hall, November 2008.
18. A. K. Jain, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.
19. "The r project for statistical computing," http://www.r-project.org/.
20. "Wireshark web site." http://www. wireshark.org/.
21. M. Friedman, "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.