A Comparative Study of Fuzzy Inference Systems, Neural Networks and Adaptive Neuro Fuzzy Inference Systems for Portscan Detection

M. Zubair Shafiq[1], Muddassar Farooq[1], and Syed Ali Khayam[2]

[1] Next Generation Intelligent Networks Research Center (nexGIN RC)
National University of Computer & Emerging Sciences (NUCES)
Islamabad, Pakistan
{zubair.shafiq, muddassar.farooq}@nexginrc.org
[2] NUST Institute of Information Technology (NIIT)
National University of Sciences & Technology (NUST)
Rawalpindi, Pakistan
khayam@niit.edu.pk

Abstract. Worms spread by scanning for vulnerable hosts across the Internet. In this paper we report a comparative study of three classification schemes for automated portscan detection. These schemes include a simple Fuzzy Inference System (FIS) that uses classical inductive learning, a Neural Network that uses back propagation algorithm and an Adaptive Neuro Fuzzy Inference System (ANFIS) that also employs back propagation algorithm. We carry out an unbiased evaluation of these schemes using an endpoint based traffic dataset. Our results show that ANFIS (though more complex) successfully combines the benefits of the classical FIS and Neural Network to achieve the best classification accuracy.

Keywords: Adaptive Neuro Fuzzy Inference System (ANFIS), Information Theoretic Features, Neural Networks, Portscan Detection.

1 Introduction

The number of vulnerable hosts on the Internet is increasing due to an increasing number of novice users using it [1]. An attacker can hack these vulnerable machines and use them as a potential army (zombies) for launching a Denial of Service (DoS) attack on any target host. It is not only the number of machines that is of concern, but also the time interval in which an attacker can gain access to the vulnerable machines [1]. Attackers usually accomplish the objective of infecting large number of machines in as little time as possible through portscans. A portscan is an attempt by the attacker to find out open (possibly vulnerable) ports on a victim machine. The attacker decides to invade the victim, through a vulnerable port, on the basis of the response to a portscan. These portscans are usually very fast and hence an attacker can take control of a major proportion of the vulnerable machines on the Internet in a small amount of time. The compromised machines can be used to launch DoS attacks. It is to be noted that some portscans are stealthy and slow as well. In this study, however, we only deal with fast (i.e., high-rate) portscans.

In the first phase of a DoS attack, the attacker deploys the DoS tools on the the victim machines or zombies. Worms provide an effective method to deploy DoS tools on the vulnerable hosts on the Internet automatically. Worms use random or semi-random portscans to find out vulnerable hosts across the Internet. In the second phase of DoS attack, the infected systems are used to launch a DoS attack at a specific target machine. This attack will be highly distributed due to the possible geographic spreading pattern of the hosts across the web. As a result, DoS attack will turn into, a more disruptive and difficult to counter, *Distributed Denial of Service* (DDoS) attack. Thus an attacker with the help of an intelligently written worm can very quickly gain control over millions of vulnerable systems on the Internet.

Fortunately, many existing solutions can detect and block this serious security threat. One of the most popular solution is 'firewall'. The problem with firewalls and many other similar tools is that they require a manual setting of a number of security levels (ranging from low and medium to high) which are not comprehendible to a novice user. If a user sets high security levels, this often results in the disruption of a user's activities by a high frequency of annoying pop-us and notices. This leads the user to select very low security levels, which practically makes a firewall ineffective.

The characteristics of the traffic generated by portscans is usually different from that generated by a normal user activity. This is because state-of-the-art worms spread on the principle of 'infecting maximum number of hosts in least possible time' [2]. Therefore, using certain characteristics of normal traffic of a user, we can train a classifier which will distinguish between normal traffic (due to activities of a normal user) and malicious traffic (due to portscans by a worm). Such a classifier employs some features extracted from the users' traffic to detect malicious activity.

In this paper we use two information theoretic features, namely *entropy* and *KL-divergence* of port usage, to model the network traffic behavior of normal user applications. We carry out a comparative study of the following three classifiers for the problem of automated portscan detection: 1) fuzzy rule-based system, 2) neural network, and 3) adaptive neuro fuzzy system.

Organization of the Paper. In the next section we provide a brief overview of three classifiers. In Section 3, we present the traffic features used in this paper. We describe the traffic test bed in Section 4. We will discuss the performance evaluation parameters in Section 5 and then analyze the results obtained from the experiments in Section 6.

2 A Review of Classification Schemes

In this section we present a brief overview of three classification algorithms. First is the classical inductive fuzzy rule learning classifier. Other two are neural network and ANFIS, which are bio-inspired classification algorithms.

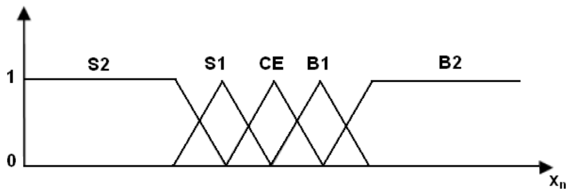


Fig. 1. Input Membership functions

2.1 Classical Inductive Fuzzy Rule Learning Classifier

In [7], the authors have given a general method to generate fuzzy rules from numerical examples for the Mamdani-type fuzzy system. In this study, 50 examples were used in the rule generation phase. This method consists of five steps:

1. In the first step, the input and output spaces are divided into fuzzy regions. This step produces initial membership functions. We divided the input space in such a manner that we get five membership functions for each input. These are denoted as S2 (Small 2), S1 (Small 1), CE (Center), B1 (Big 1) and B2 (Big 2). For symmetry, the membership functions are chosen to be isosceles triangles. Figure 1 shows the input membership functions. The output of the fuzzy classifier has only two membership functions, benign and malicious, because of its Boolean classification nature. So, the design of output membership functions does not follow procedure defined in [7].
2. The second step involves the initial generation of the fuzzy rules from given data pairs. Given the ith numerical example data (Xi$_1$, Xi$_2$, Xi$_j$, ... , Yi), the degree of membership for each input is calculated. It is possible that the input parameter has non-zero degree of membership (DOM) for two membership functions. In this case the membership function with a maximum DOM (DOM$_{max}$) is chosen. A rule is formulated as:
   `IF X1 is ‘a’ AND X2 is ‘b’ AND ... then y is ‘c’`,
   where a,b and c are the respective membership functions for each input.
3. A degree is assigned to each rule. For rule i, degree is defined as,
   `degree(i) = DOM`$_{max(1)}$ `x DOM`$_{max(2)}$ `x DOM`$_{max(i)}$ `x ... x DOM`$_{max(y)}$
4. After performing the third step, we will get a populated rule base. It is possible that more than one rule, with similar inputs, may have different outputs. This situation represents the conflict amongst the rules. In order to resolve this conflict, the rule with a maximum degree will be chosen.
5. We use centroid defuzzification technique to get a crisp output. This defuzzification technique was chosen because the output produced by it includes the balanced effect of all the inputs. The formula for calculating the centroid is given by:

$$F^{-1}_{Center\_Of\_Gravity}(\bar{A}) = \frac{\int_x \mu\bar{A}(x)xdx}{\int_x \mu\bar{A}(x)dx}$$

2.2 Neural Network Classifier with Back Propagation Learning Algorithm

Neural networks are bio-inspired paradigm which map concepts from the biological nervous systems. They consist of a network of neurons which are boolean in nature. The connections and the weight between these connections are crucial to the performance of the network. The neural network used in this study consisted of two neurons in its output layer to identify the traffic behavior as either benign or malicious. The network is a two-layer log-sigmoid/log-sigmoid network. The log-sigmoid transfer function was selected because its output range is suitable for learning to output Boolean values. The first hidden layer had 10 neurons. This number was chosen after pilot studies using different number of neurons. The standard back propagation learning algorithm was utilized by a batch training scheme in which weights are updated after a complete batch of training examples. 50 training examples were used in the training phase. More details can be found in [8].

2.3 Adaptive Neuro Fuzzy Inference System (ANFIS)

Adaptive Neuro Fuzzy Inference System (ANFIS) is a fuzzy rule based classifier in which the rules are learnt from examples that use a standard back propagation algorithm. Note that this algorithm is also used in neural network training. However, ANFIS is far more complex than the simple Mamdani-type fuzzy rule based system as explained in Section 2.1. ANFIS uses Sugeno-type fuzzy system. The subtractive clustering was used to divide the rule space. Five triangular membership functions were chosen for all inputs and output similar to the fuzzy system in Section 2.1. 50 training examples were chosen in the training phase. An interested reader can find the details of ANFIS in [9].

3 Traffic Feature Modeling Using Information Theoretic Measures

We employ information theoretic measures [6] to compare the probability distributions of a pre-sampled benign traffic profile and run-time traffic profile. It is assumed that the malicious traffic is not present while sampling the benign traffic profile. We have chosen *entropy* and *Kullback-Leibler (KL) divergence* [6] of port usage as tools to compare the benign traffic profile (collected prior to classification) with a run-time traffic profile. These measures have been used previously for network anomaly detection [3,4].

We are interested in outgoing unicast traffic to detect the malware residing on the system, which tries to propagate through portscan. We have calculated the entropy and KL using both source and destination port information. The source and destination port information is collected at a session level, where a session is defined as the bidirectional communication (mostly involving multiple packets) between two IP addresses. Entropy and KL are calculated in a time window of

15 seconds in our study. However, qualitatively similar results are obtained for other sizes of time window.

Entropy gives the spread of a probably distribution. In order to calculate entropy in a time window, let p_i be the number of times source port i was used and q_i be the number of times destination port i was used in a time window. Let p_n be the aggregate frequencies of source ports used in a particular time window n. Similarly, let q_n be the aggregate frequencies of destination ports used in a particular time window n. Mathematically, $p_n = \sum_{i=0}^{65,535} p_i$ and $q_n = \sum_{i=0}^{65,535} q_i$. The source and destination port entropies are defined as:

$$H_{source} = -\sum_{i=0}^{65,535} \frac{p_i}{p_n} \log_2 \frac{p_i}{p_n} \tag{1}$$

$$H_{destination} = -\sum_{i=0}^{65,535} \frac{q_i}{q_n} \log_2 \frac{q_i}{q_n} \tag{2}$$

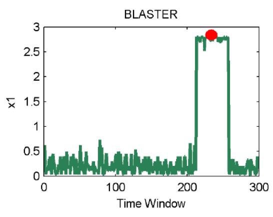
KL divergence gives the distance between two probability distributions. In this case, it will give the difference between distributions of traffic in a particular session window and benign traffic. For source and destination port KL, let $p_i^{'}$ be the frequency of source port i in the benign traffic profile and $q_i^{'}$ be the frequency of destination port i in the benign traffic profile. Moreover, p and q represent the aggregate frequency of source and destination ports in the benign profiles. Mathematically, $p = \sum_{i=0}^{65,535} p_i^{'}$ and $q = \sum_{i=0}^{65,535} q_i^{'}$. The source and destination port KL are defined as:

$$D_{source} = \sum_{i=0}^{65,535} \frac{p_i}{p_n} \log_2 \frac{p_i/p_n}{p_i^{'}/p} \tag{3}$$

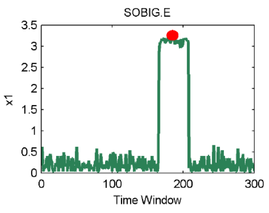
$$D_{destination} = \sum_{i=0}^{65,535} \frac{q_i}{q_n} \log_2 \frac{q_i/q_n}{q_i^{'}/q} \tag{4}$$

Since we are focusing on fast portscans, we invoke classifier only if the number of sessions per time window exceeds the `SessionThreshold`. The `SessionThreshold` in this study was set to 15 sessions per time window. This corresponds to and average rate of one session per second. The worms with the session rates lower than `SessionThreshold` are ignored. The value of `SessionThreshold` is justified since the motive of a worm is to infect large number of machines in as little time as possible.

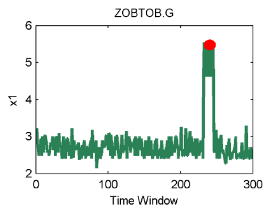
In order to produce suitable inputs for the classification systems, the means of respective information theoretic measures are used in Equations (1), (2), (3) and (4) that are calculated from the benign traffic profile. These parameters are labeled as H_{benign_source}, $H_{benign\_destination}$, D_{benign_source} and $D_{benign\_destination}$. The differences between the parameters, calculated from run-time traffic profile, and means of respective parameters calculated from benign traffic profile were



(a) Blaster Worm

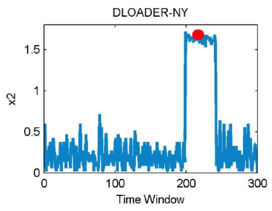


(b) SoBig.E Worm

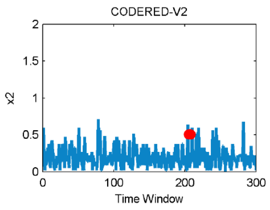


(c) Zobtob.G Worm

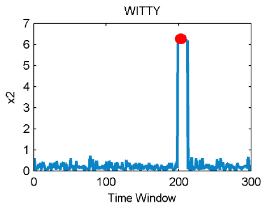
Fig. 2. Source Port Entropy for different worms



(a) Dloader-NY Worm

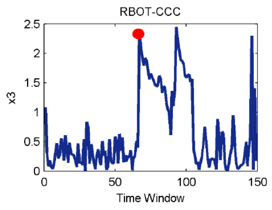


(b) Code Red v2 Worm

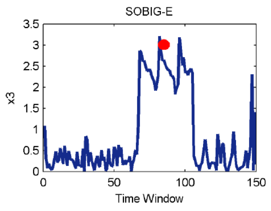


(c) Zobtob.G Worm

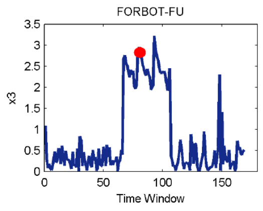
Fig. 3. Destination Port Entropy for different worms



(a) Rbot-CCC Worm



(b) Sobig-E Worm



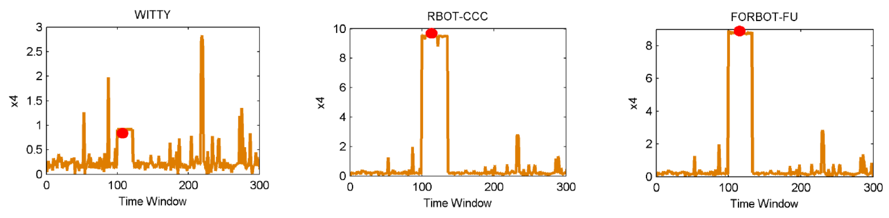
(c) Forbot-FU Worm

Fig. 4. Source Port KL for different worms

used as inputs to the classifiers. For simplicity absolute value of the difference is considered. Mathematically we can represent this as:

$$x_1 = |H_{source} - \mu_{Hbenign\_source}|$$
$$x_2 = |H_{destination} - \mu_{Hbenign\_destination}|$$
$$x_3 = |D_{source} - \mu_{Dbenign\_source}|$$
$$x_4 = |D_{destination} - \mu_{Dbenign\_destination}|$$

Figures 2, 3, 4 and 5 show a plot of these parameters for different worms. Note that the circle represents the middle of the infection period.



(a) Blaster Worm (b) SoBig.E Worm (c) Zobtob.G Worm

Fig. 5. Destination Port KL for different worms

4 Testbed Formation

In this section, we present the details of the traffic sets that were used for the comparison of the classifiers mentioned in Section 2.

4.1 Benign Traffic Set

The benign traffic data-sets were collected over the period of 12 months on a diverse set of 5 endpoints[1]. These endpoints machines were installed with Windows 2000/XP that served a variety of different types of users. Some endpoints were also shared by multiple users. Characteristics of the benign traffic set are tabulated in Table 1. Traffic data-sets were collected using `argus`, which runs as a background process storing network activity in a log file. As stated earlier, each entry in the log file corresponds to a session, where a session is defined as the bidirectional communication between two IP addresses. The entries of the log files are in the following format:

```
<session id, direction, protocol, src port, dst port, timestamp>
```

Direction is a one byte flag showing if the packets in a session are outgoing unicast, incoming unicast, outgoing broadcast, or incoming broadcast packets. We are only interested in outgoing unicast traffic. Since, the traffic sets were stored for offline analysis, every session had an associated timestamp.

4.2 Worm Set

Malicious traffic sets were collected by infecting virtual machines with different self-propagating malicious codes. Note that `CodeRedv2` and `Witty` were simulated. The traffic sets were collected using the same method as explained before in Section 4.1. These malicious traffic sets were embedded into benign traffic sets to form a test for evaluation of the designed system. The details of the malware used in this study are given in Table 2.

[1] "An endpoint is an individual computer system or device that acts as a network client and serves as a workstation or personal computing device." [5]

Table 1. Statistics of Bengin set used in this Study

Endpoint ID	Endpoint Type	Mean Session Rate (/sec)
1	Office	0.22
2	Home	1.92
3	Univ	0.19
4	Univ	0.28
5	Univ	0.52

Table 2. Statistics of Worm set used in this Study

Worm Name	Release Date	Ports Used
`Blaster`	Aug 2003	TCP 135,4444, UDP 69
`Dloader-NY`	Jul 2005	TCP 135,139
`Forbot-FU`	Sep 2005	TCP 445
`Rbot.CCC`	Aug 2005	TCP 139,445
`CodeRedv2`	Jul 2004	TCP 80
`Witty`	Mar 2004	UDP 4000
`SoBig.E`	Jun 2003	TCP 135, UDP 53
`Zobtob.G`	Jun 2003	TCP 135, 445, UDP 137

4.3 Formation of Infected Traffic Sets

Due to the university policies and the user reservations, we were not able to infect operational endpoints with worms. Therefore, we resorted to the following offline traffic mixing approach. We inserted T minutes of malicious traffic data of each worm in the benign profile of each endpoint at a random time instance. Specifically, for a given endpoint's benign profile, we first generated a random infection time t_I (with millisecond accuracy) between the endpoint's first and last session times. Given n worm sessions starting at times $t_1, ..., t_n$, where $t_n \leq T$, we created a special infected profile of each host with these sessions appearing at times $t_I + t_1, ..., t_I + t_n$. Thus in most of the cases once a worms traffic was completely inserted into a benign profile, the resultant profile contained interleaved benign and worm sessions starting at t_I and ending at $t_I + t_n$. For all worms except Witty, we used $T = 15$ minutes and to simulate the worstcase behavior of Witty, we inserted only $20,000$ scan packets (approximately 1 minute) in the infected profiles.

5 Receiver Operating Characteristic (ROC) Performance Parameters

In ROC analysis, the 2x2 confusion matrix for performance evaluation of boolean classifiers gives four possible outcomes [10]. These outcomes are True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN). The metrics considered for the evaluation of our fuzzy based classifier are False Positive Rate ($fprate$), and True Positive Rate ($tprate$). These metrics are defined as, $fprate = \frac{FP}{FP+TN}$ and $tprate = \frac{TP}{TP+FN}$.

Table 3. Results on the Infected dataset

	Inductive FIS [7]	**Neural Network [8]**	**ANFIS [9]**
Endpoint ID - 1			
tp rate	0.888	0.735	0.885
fp rate	0.037	0.062	0.043
Endpoint ID - 2			
tp rate	0.857	0.728	0.861
fp rate	0.163	0.191	0.155
Endpoint ID - 3			
tp rate	0.900	0.774	0.893
fp rate	0.100	0.116	0.121
Endpoint ID - 4			
tp rate	0.865	0.731	0.829
fp rate	0.096	0.104	0.088
Endpoint ID - 5			
tp rate	0.954	0.790	0.968
fp rate	0.160	0.211	0.114

6 Classification Results

The ROC performance evaluation parameters for all the endpoints used in this study are tabulated in Table 3. It is interesting to note that the worst results for all the classifiers were reported for the endpoint 2. Endpoint 2 was a home based endpoint with a relatively high session rate because of several multimedia and video streaming applications that were running on this endpoint. High $fprate$ is because of the high volume and the bursty nature of multimedia traffic which resembles the traffic produced by the portscan activities of worms. Note that the results of endpoint deployed at the university and office are very similar.

It is clear from the results tabulated in Table 3 that ANFIS outperforms rest of the classifiers. The results of Inductive FRBS are better than those of Neural network which shows the promise of fuzzy based schemes for modeling and representation of network traffic features. Neural network is unable to cater for the inherent fuzziness in the users' traffic patterns. ANFIS combines the advantages of the back propagation learning algorithm with the fuzzy based classifier, and as a result, we get the best detection accuracy.

7 Conclusion and Future Research

Our findings from this study are that the users' behaviors are not very crisp because of the pseudo-random nature of the users' network traffic. To cater for the inherent fuzziness in modeling of the user's network traffic trends, a fuzzy rule based system is a better approach. But the generation of fuzzy rules and membership functions is a problem. One solution is to use classical inductive fuzzy rule learning for fixed membership functions which is an empirical yet an effective way to generate fuzzy rules. Another solution is to use a neural network with a more sophisticated back propagation learning algorithm which is suitable for use with numerical data pairs for training. But a neural network based solution is unable to cater for the inherent fuzziness in the users' traffic patterns. This leads us to the following question: Is a combined approach i.e. ANFIS, which uses fuzzy rule based system along with the sophisticated back propagation learning algorithm, more effective than both of the standalone approaches?. The answer to this question is positive as per our experimental study. ANFIS is able to effectively combine the benefits from both approaches. The experimental results, for portscan detection, of ANFIS are better than both classical inductive fuzzy rule base system and neural network based classifier using the standard back propagation algorithm.

In future we wish to run the experiments on a more diverse and extensive traffic set consisting of more diverse endpoints. We also wish to extend our malware traffic repository. Further, we also plan to evaluate the performance of various machine learning algorithms such as support vector machines (SVMs) and other bio-inspired schemes such as Artificial Immune Systems (AIS).

Acknowledgements. This work is supported by the National ICT R&D Fund, Ministry of Information Technology, Government of Pakistan. The information, data, comments, and views detailed herein may not necessarily reflect the endorsements of views of the National ICT R&D Fund.

References

1. Staniford, S., Paxson, V., Weaver, N.: How to Own the Internet in Your Spare Time. In: Usenix Security Symposium (2002)
2. Moore, D., Shannon, C., Voelker, G.M., Savage, S.: Internet Quarantine: Requirements for Containing Self-Propagating Code. In: IEEE Infocom (2003)
3. Gu, Y., McCullum, A., Towsley, D.: Detecting anomalies in network traffic using maximum entropy estimation. In: ACM/Usenix IMC (2005)
4. Lakhina, A., Crovella, M., Diot, C.: Mining anomalies using traffic feature distributions. In: ACM Sigcomm (2005)
5. Endpoint Security, `http://www.endpointsecurity.org`
6. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience, Chichester (1991)
7. Wang, L.-X., Mendel, J.M.: Generating Fuzzy Rules by Learning from Examples. IEEE Transactions on Systems, Man, and Cybernetics 6(22), 1414–1427 (1992)
8. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
9. Jang, J.-S.R.: ANFIS: Adaptive-Network-Based Fuzzy Inference System. IEEE Transactions on System, Man and Cybernetics (23), 665–685 (1993)
10. T. Fawcett.: ROC Graphs: Notes and Practical Considerations for Researchers, Technical report (HPL-2003-4), HP Laboratories, Palo Alto, CA, 2003-4, USA (2003)