A Rule-Based Training for Artificial Neural Network Packet Filtering Firewall

Akharin Khunkitti
Assistant Professor, Faculty of Information Technology
KMITL – King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
e-mail: Akharin@it.kmitl.ac.th

Ponsuda Chongsujjatham
Student, Faculty of Information Technology
KMITL – King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
e-mail: 58070093@kmitl.ac.th

Abstract— **The Artificial Neural Network has been used in many network applications, including firewalls. Training process of neural network is very important to define the intelligence of the systems. Many artificial neural network firewalls used direct network packets for training process, which may be difficult to get training samples and may not follow their firewall's policies. This research work proposes a rule-based training for artificial neural network packet filtering firewall. The developed neural network model is trained by generating samples from legacy firewall ruleset. Each rule has been converted to random training samples. All firewall's rules are used to generate the training sample data, rule by rule. The accuracy results show high accuracy with some behavior studies. The number of samples per rule, number of rules and rule style, including default rule and rule-scope effects, have been studied for the best accuracy results. This study also concludes the styles of firewall ruleset for the best accuracy of the proposed system. (Abstract)**

Keywords-component; Rule-Based Training; AI; Artificial Neural Network; ANN; Packet Filtering Firewall (key words)

I. Introduction

Packet filtering firewalls are commonly used in networking environments for security purposes. All network traffics have been controlled by their rules, for permitting to pass the firewalls or have been dropped. The firewall rules must be assigned by network administrators, or network specialists. The rules control the security policies for the firewall. When there are additional policies, rules have to be applied to the firewall. The firewalls must be strictly obeyed to the rules. This make firewalls fixed to the rules only. They cannot adapt themselves from the rules, to permit or deny the traffic in smarter ways.

The Artificial Intelligence (AI) is rapidly popular in the past decade. With some helps of Neural Networks (NN), it has been applied to many applications. This also makes Artificial Neural Network (ANN) models more useful in networking applications too. The ANN has been adopted in Intrusion Detection Systems (IDS), and also in firewalls. Many research works have tried to make IDS and firewalls smarter. They tried to let the systems learned from the network traffics, which are difficult to find the required traffics, for learning. This paper applies ANN to packet filtering firewalls to make the firewall more intelligent, and can be adapted themselves by learning from firewall's rules. This will easily to teach the systems adapt to the policies, and also easily to migrate from the legacy firewalls to the AI-based firewalls.

This paper will first discuss about packet filtering firewalls and related works, in section II. It will also discuss about Artificial Neural Network in firewalls, and related works, e.g. ANN in IDS. The proposed system and method will be explained in section III. There will be a study of proposed system's behaviors, by experiments. The results will be analyzed to get the optimum results and suggestions for practical implementations. The final words will be concluded our works in section IV.

II. Packet Filtering Firewalls

A. *Legacy Packet Filtering Firewall*

The packet filtering firewalls usually have been placed between two, or more, physical or logical networks. They will control network traffics by let the packets pass through, or reject them, according to firewall's rules. Each packet will be verified to all rules, by comparing the packet's headers to rule's conditions. If it matches with a rule, firewall will do the matched-rule's action, which may be allow or deny the packet. If the packet does not match to any rules, it will be matched with the last rule, default rule, which may be explicit or implicit declared. The default rule's action may be allow, which let not-matched traffic pass through, or deny, which drop the not-matched packet. This means that the firewall's policy has been controlled by firewall's rules. All firewall's rules are assigned by network administrators or specialists, who define the firewall's policy. The policy has been translated to firewall's rules, and controlled all packets. The firewall's behavior strictly follow the rules. The legacy firewall cannot be flexible to do any actions besides rules.

B. *Artificial Neural Network Packet Filtering Firewall*

There are many research works applying Artificial Intelligence (AI) using Artificial Neural Network (ANN) to network security devices, including packet filtering firewalls, [1]-[4]. The AI using ANN also has been adopted in more research works including firewalls and Intrusion Detection Systems (IDS). The ANN is a technique to make the systems more intelligent using a network of neurons (nodes), like a human brain. For the ANN firewall, a packet, whole or partial packet, will go to input nodes of the neural network. The processing of neural network propagates values from input nodes, times by weighted values in the neural network, to next neural nodes, until reach the output nodes of the neural network. The output nodes will represent the packet classification, according to the firewall's or IDS's policy.

The weighted values in a neural network can be derived from training process, which computes the weights from known inputs and outputs. Many works use the network packet as inputs, with known classified values/outputs, for training the neural network, e.g. [1], [3], [4]. The training process is an important process to make ANN intelligent for evaluating output results. It requires sample data for teaching the neural network, to get weighted values. Many works, such as [1]-[4], used the direct network packet as input for training process. The direct network packets may be difficult to find or generate to get the best results. The training dataset can be get from archived collections of network packets, e.g. in [3]. The archived datasets are predefined sample data, which may not follow the network's policy. It's also hard to follow the defined policy with the direct packet training.

III. Rule-Based Training for ANN Firewall

All firewalls normally have their rules for their security processing. It can be easy for training the ANN firewalls using their existing firewall rules. The rule-based training can make the ANN firewalls following their policies, without finding additional training dataset. Then, we propose a method of training ANN packet filtering firewall, using rules for generating training data to the ANN model.

A. Design and Development

The overall developed system has shown in Fig.1. The system, with feed forward neural networks, has two processes, training and evaluating processes. The training process uses legacy firewall rules as input. All rules are used to generate sample packets for training the neural network model, as engine in Fig.1. The training process generates, and modifies, all necessary parameters and weight values in model, which will be used in evaluating process. After training has been completed, the system can evaluate network packets, which may be pre-processed before feeding into evaluating process. The evaluating results in our system are simple allow or deny for further firewall processing.

There are many models for neural network in choices. For the sake of simplicity and being a starting point of study, a simple feed forward popular model is chosen. So, the developed model is a kind of feed forward neural network model, Multi-Layer Perceptron (MLP), with one input layer, three hidden layers, and one output layer, as shown in Fig. 2. The input layer represents the packet's fields of 172 bits, conformed 172 input nodes. The input fields are, [Field (number of bits)];

- Direction (1)
- Interface (3)
- Source IP Address (32)
- Subnet Mask (32)
- Source Port (16)
- Destination IP Address (32)
- Subnet Mask (32)
- Destination Port (16)
- Protocol (8)

Each hidden layer composes of 250 nodes, derived from [5]-[6], for internal processing. And the output layer has 2 nodes, represented for 2 actions, allow and deny.

For developing the system, Python, Google TensorFlow v.1.13.1, Keras v.2.2.4, and Numpy v.1.16.2, [7]-[9], have been used on Microsoft Windows platform.

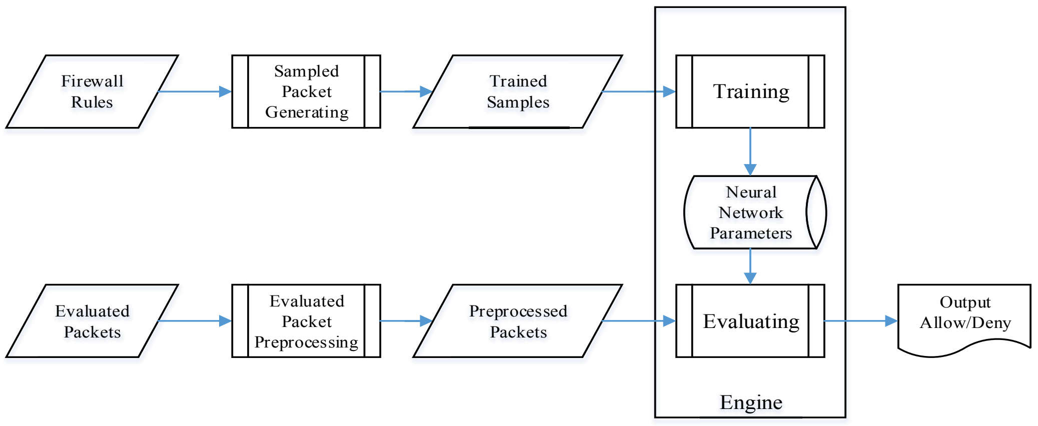

Figure 1. Overall of the ANN Packet Filtering Firewall system.

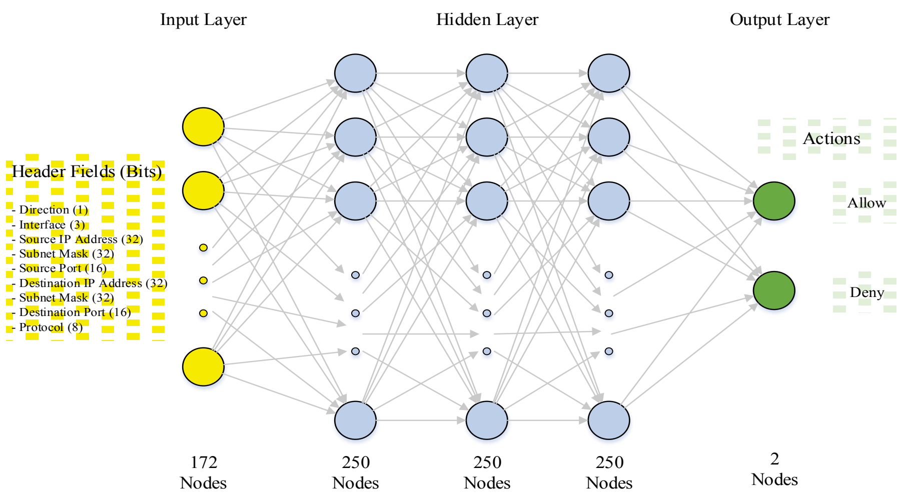

Figure 2. Neural network model for system's implementation.

B. The Rule-Based Training Algorithm

The core algorithm of proposed rule-based training for ANN firewall has shown in Fig.3. All legacy firewall rules have been read, rule by rule. Each rule is used for generating sample packets, which formed a Number of Samples per rule. Each sample packet has been generated using a random value in range of each packet's field.

The sample generating process converts each firewall rule into trained samples, depend on the number of samples per rule. So, the total number of trained samples is the total number of firewall rules, times number of samples per rule.

C. Experimental Study

The developed system has been tested. And behavioral studies have been experimented for getting the best results of Rule-Based Training for ANN Firewall.

The training process uses trained samples from the legacy firewall rules. Each rule is used to generate sample data, feeding into training process of the model. The number of samples per firewall rule has been studied to get the optimum result. If number of samples per rule is too high, training process will take more times for training process. But if it's too small, the accuracy of results will be bad. So, the optimum number has to be studied.

```
for i=1 to R /* R=Number of Rules in file */
{
  for j=1 to N /* N=Number of Samples per rule */
  {
    Generating sample for each field, Random value in each-field's range, on rule i;
  }
}
```

Figure 3. Core algorithm of Rule-Based Training for ANN Firewall.

Because the training process is derived from the classical firewall rules, the accuracy results will be affected by the styles of written rules. The number of rules, and rule-set's styles, will be studied with accuracy of the output results.

The tested evaluating packets have been classified into;

- InScope, which are packets generated within the firewall rules' scope, e.g. IP addresses and port numbers, in the rules,
- OutScope, which are packets generated outside the firewall rules' scope, outside rules, and
- MixScope, which are mix of InScope and OutScope.

The effects of rulesets, with and without default rule, are also studied.

Experiments use 4 rulesets, as shown in Fig. 4. The default rule, which will be appended as the last rule of each ruleset, labelled as WithDefault, is;

"deny any on any from any to any port any any".

```
RuleSet1:
allow in on eth0 from 192.168.0.0/16 to 161.246.34.11/24 port 80 tcp
```

```
RuleSet2:
allow in on eth0 from 192.168.0.0/16 to 161.246.34.11/24 port 80 tcp
allow in on eth0 from 100.100.0.0/16 to 161.246.34.11/24 port 80 tcp
```

```
RuleSet3:
allow in on eth0 from 192.168.0.0/16 to 161.246.34.11/24 port 80 tcp
deny in on eth0 from 192.168.0.0/16 to 161.246.34.11/24 port 22 tcp
allow in on eth0 from 100.100.0.0/16 to 161.246.34.11/24 port 80 tcp
deny in on eth0 from 100.100.0.0/16 to 161.246.34.11/24 port 22 tcp
```

```
RuleSet4:
allow in on eth0 from 192.168.0.0/16 to 161.246.34.11/24 port 80 tcp
deny in on eth0 from 192.168.0.0/16 to 161.246.34.11/24 port 80 udp
deny in on eth0 from 192.168.0.0/16 to 161.246.34.11/24 port 22 tcp
deny in on eth0 from 192.168.0.0/16 to 161.246.34.11/24 port 22 udp
allow in on eth0 from 100.100.0.0/16 to 161.246.34.11/24 port 80 tcp
deny in on eth0 from 100.100.0.0/16 to 161.246.34.11/24 port 80 udp
```

Figure 4. Rulesets using in experiments.

The experiments use epoch of 150, and batch size of 1000 for learning process. All action results are compared to original rulesets, to get the accuracy results in percentage.

Trained samples are generated using original rule sets, with varied number of samples per rule. And the evaluating packets are generated in experiments, for InScope 1106 packets, 11.06% of total, and OutScope 8896 packets, 88.96% of total. The total number of tested packets is 10000 packets, as MixScope.

The experiments are studied by using following variables; the number of samples per rule, styles of firewall rulesets, with and without default rule. All results will be compared to each scope of evaluating packets, InScope, OutScope and MixScope.

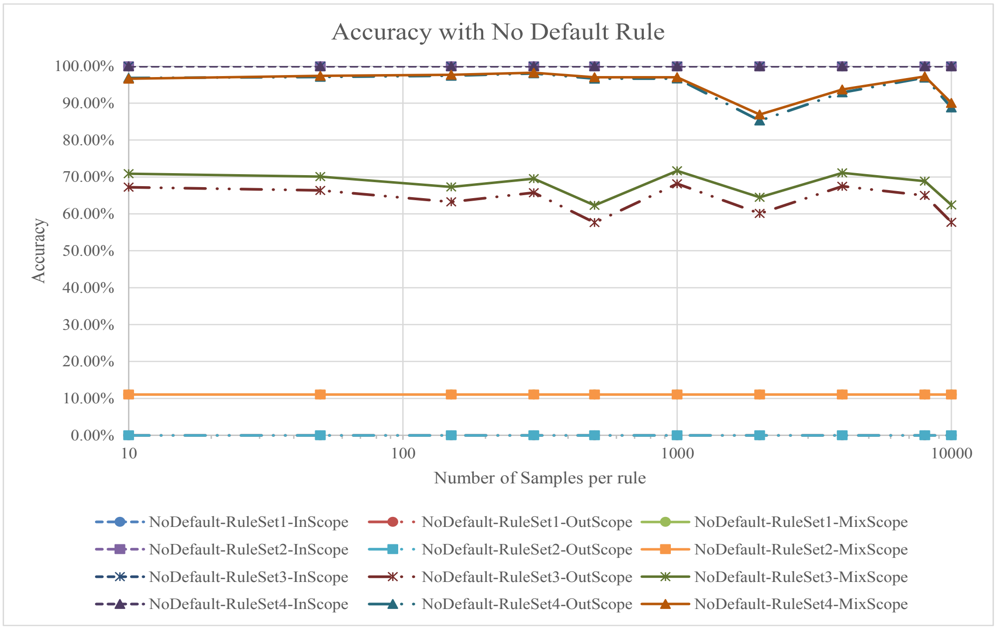

Figure 5. Experimental results of accuracy with No Default rules.

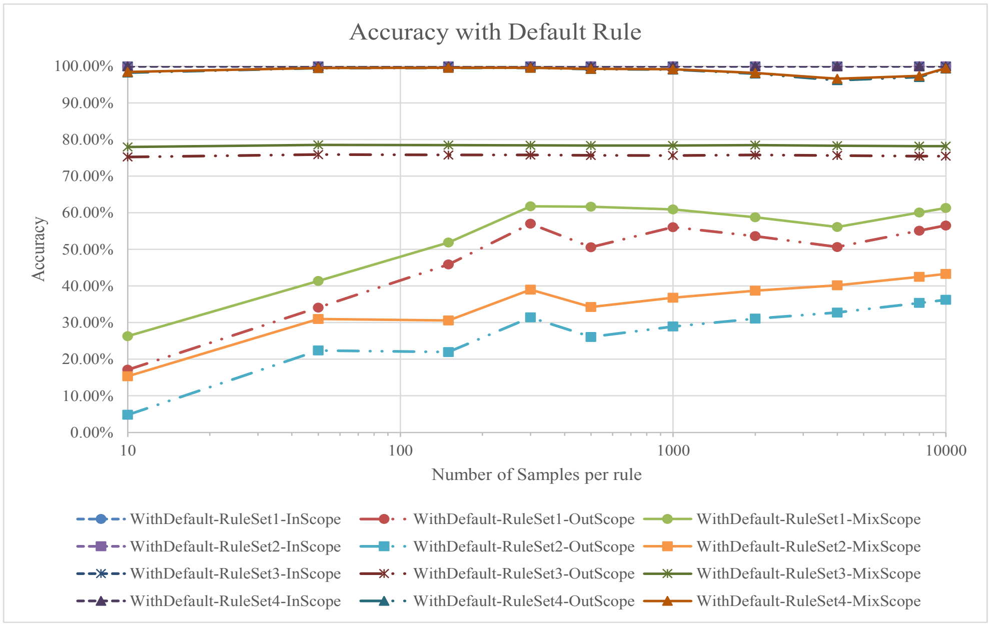

Figure 6. Experimental results of accuracy With Default rules.

D. Analysis

The experimental results are shown in Fig. 5 and Fig. 6. Graphs represent the accuracy of action results compared to the original rulesets, for each experiment. The experiments have been done for studying effects of rules' scope, default rule, rule written style, and finding the optimum number of samples per rule.

For the rules' scope effect, all evaluated InScope results have accuracy 100% or near 100%, which are very high accuracy, because the InScope packets are the same scopes of trained samples, which are generated from rules. But OutScope results, which are not trained, have very low accuracy, with some of them are zero accuracy. So, the MixScope results, which are mixed between InScope and OutScope, are in between InScope and OutScope results. This means that the model has to be trained as much as possible, with diversity, to get the best accuracy results.

It can be noticed from graphs, default rule also has effect on accuracy results. Explicit default rule in the rulesets, labelled as WithDefault rulesets, will increase the accuracy results, compared to the same NoDefault ruleset.

For number of samples per rule effect, the WithDefault-Ruleset graphs show the low accuracy with lower number of samples per rule, until the number of samples per rule reaches a certain number, the accuracy results reach a steady value. This means that increasing number of samples per rule will not gain accuracy after reaching a certain number, as we expected. Increasing the numbers wastes the times during training process. As shown in the graphs, the optimum number of samples per rule for this work is around 300 samples per rule.

The style of firewall rules also have different accuracy results. As shown in Fig. 4, RuleSet1 has only one rule, while RuleSet2, RuleSet3 and RuleSet4, have more rules and more diversity in rules, consequentially. This means that increasing the number of rules and combining between allow and deny in rulesets, will have better accuracy results. These have two reasons. First, increasing number of rules means increasing the number of trained samples, which makes the model more data to be trained. And second, combining rules will train the model for both allow and deny samples, which also makes the model more learning data for both actions. It has been shown in both Fig. 5 and Fig. 6, that RuleSet4, which has most number of rules and most variety rules, has the best accuracy results.

IV. Conclusion

This research work has proposed a rule-based training method for ANN packet filtering firewall. The trained samples have been generated from the legacy firewall rules, with number of samples per rule. Each sample is generated with random value in each field's range. This work also studied factors affected to the accuracy results, which are rule-scope of evaluated packets, default rule, written style of firewall rules, and also the optimum number of generated samples for each rule. To get the best accuracy results, a firewall ruleset, which trains the neural network model, is recommended to be written as; (1) maximized the number of rules, (2) explicit having both allow and deny rules, (3) explicit adding default rule as the last rule, and (4) explicit specify the scope of rules as many as possible, e.g. IP addresses, port numbers. The number of samples to be generated for training process, should be minimized, for reducing training time. The optimum number of samples per rule, in this work, is around 300 samples per rule. Increasing the number will waste training time without gaining accuracy results.

From this work, the legacy firewall ruleset is still important for training process with deep learning neural network firewall. It can be noticed that accuracy results are not exactly the same as legacy packet filtering firewall. The maximum accuracy for this work is about 99%. There should be other research works to study the 1% of mismatch results, which hoping to have positive results for detecting unwritten-rule threats.

References

[1] Kristian Valentin, Michal Maly. "NETWORK FIREWALL USING ARTIFICIAL NEURAL NETWORKS". Computing and Informatics, Vol. 32, Jan. 2013, pp. 1312-1327.

[2] Archana Mishra, Abhishek Agrawal, Rajeev Ranjan. "Artificial Intelligent Firewall." ACAI '11 Proceedings of the International Conference on Advances in Computing and Artificial Intelligence, 2011, pp. 205-207

[3] Devikrishna K S and Ramakrishna BB, "An Artificial Neural Network based Intrusion Detection System and Classification of Attacks", International Journal of Engineering Research and Applications (IJERA), ISSN: 2248-9622, www.ijera.com, Vol.3, Issue 4, Jul-Aug 2013, pp. 1959-1964.

[4] Michal Turcanik, "Packet Filtering by Artificial Neural Network", International Conference on Military Technologies (ICMT) 2015, 19-21 May 2015. [IEEE Xplore Digital Library, Added Date – 13 July 2015].

[5] Jeff Heaton. "The Number of Hidden Layers." Internet: https://www.heatonresearch.com/2017/06/01/hidden-layers.html, Jun. 06, 2017 [Accessed-Mar. 23, 2019].

[6] Hobs. "How to choose the number of hidden Layers and Nodes in a feedforward neural network?," USENET: https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-Layers-and-Nodes-in-a-feedforward-neural-netw, Jul. 22, 2018 [Accessed-Mar. 23, 2019].

[7] Google Inc. "Tensorflow". Internet: https://www.tensorflow.org, Jun. 14, 2019 [Accessed-Aug. 4, 2018].

[8] Keras-team. "Keras: The Python Deep Learning library". Internet: https://keras.io, Jun. 13, 2019 [Accessed-Feb. 30, 2019].

[9] NumPy Developers. "NumPy". Internet: https://www.numpy.org, Jun. 13, 2019 [Accessed-Oct. 12, 2018].