Jurnal Kejuruteraan 16 (2004) 79-95

Performance of Packet Filtering Using Back Propagation Algorithm

M. I. Buhari, M. H. Habaebi & Burhanuddin Mohd. Ali

ABSTRACT

In this paper, we analyzed the use of neural network for packet filtering. The neural network system was designed in eight ways with input to the neural network in the form of either access rules or optimized access rules or binary form of access rules or representing wildcards as 0 & 255 or combination of them. These trained neural networks were analyzed for their correctness and the performance aspects such as training time using test data. In order to further improve the security, the data related to the local usage of the network were also used to train the network. An example of implementing these trained systems in active networks packet filtering was presented.

Keywords: packet filtering, back propagation, firewall, active network, TCP

ABSTRAK

Dalam kertas kerja ini, kami menganalisis penggunaan rangkaian neural untuk penurasan paket. Sistem rangkaian neural telah direka dalam lapan kaedah dengan masukan kepada rangkaian neural dalam bentuk samada peraturan capaian atau peraturan capaian optimum atau dalam bentuk binari bagi peraturan capaian atau diwakili oleh kad liar iaitu 0 dan 255 ataupun gabungan antara mereka. Rangkaian neural yang telah dilatih ini dianalisa dari segi aspek ketepatan dan prestasinya seperti masa latihan menggunakan data ujian. Bagi tujuan meningkatkan tahap keselamatan, data yang berkaitan dengan penggunaan setempat bagi suatu rangkaian juga telah digunakan untuk melatih rangkaian tersebut. Satu contoh implementasi sistem terlatih dalam rangkaian aktif penuras paket juga telah disampaikan.

Katakunci: penurasan paket, perambatan balik, dinding api, rangkaian aktif, TCP

INTRODUCTION

The firewall has to use an IP router to control the passing of any packet from the Internet into the Intranet. Packet filtering parses the headers of the received IP packets and forwards or discards the packets according to an Access Control List (ACL) specified by a network administrator. The performance of the whole router depends on the procedure on which the rules in the ACL are applied to the packet and a decision made to allow or reject the packet. Processing all packets passing through a router degrades the packet-forwarding performance. Also, the performance of the off-the-

shelf routers is degraded in proportion to the complexity of the ACL. Hence, IP routers that forward packets more efficiently are to be explored. Currently, packet filtering is done using access control lists, which are processed sequentially.

SEQUENTIAL PARSING AND ITS DRAWBACKS

A rule in an ACL is a set of conditions that are to be satisfied to perform the specified action. If the action in the i^{th} row of the ACL is A_i, the ACL is sequentially parsed; i.e., each action is performed only when the corresponding condition is satisfied:

If (Conditions of Rule 1) then A_1
Else if (Conditions of Rule 2) then A_2
... Else if (Conditions of Rule n) then A_n

Sequential parsing causes the following problems (Miei et al. 1997):

1. As the number of rows of the rules in the ACL increases, the cost of packet filtering also increases.
2. Because a condition consists of conjunctions of parameters, disjunctive conditions must be specified in several rules. In these rules, each kind of parameter has the same value unless it specifies a disjunctive value. As a result, the same value might be applied to a packet many times.
3. Consider the conditions of the i^{th} rule and the k^{th} rule in the ACL, where $i < k$. If the Condition of the i^{th} rule is always true when the Condition of the k^{th} rule is true, the rule in the k^{th} row of the ACL is redundant because its action, A_k, is never carried out. This is similar to an infeasible path (IFP) in a procedural program and this has to be remedied.

TYPES OF ACCESS LISTS

There are two types of access lists, namely, Standard access list and Extended access list (Odom 2000). A description of them is given below.

TABLE 1. Types of access lists

Type of Access list	What can be matched
IP standard	a. Source IP address b. Portions of the source IP address, using a wildcard mask.
IP Extended	a. Source and destination IP address b. Portions of the source IP address, using a wildcard mask c. Portions of the destination IP address, using a wildcard mask d. Protocol type (TCP, UDP, ICMP, IGRP, IGMP, and others) e. Source and Destination port f. Established – matches all TCP flows except first flow g. IP Type of Service, IP precedence

RELATED WORK

As the performance of the firewall has been highly affected due to the process of sequential parsing, Miei et al. (1997) have proposed a compiler

for parallelizing IP-packet filter rules, to improve the network security and reduce the degradation in packet-forwarding performance. In the proposed method there is no need for any intermediate program. The current day packet filtering is done using sequential parsing methods (Miei et al. 1997). With the sequential parsing technique, increase in the number of rules in the ACL list will cause the parser to consume more time and become inefficient. According to Park (2002) and Sandhu (2003), the concept of usage control (UCON) method includes traditional access control, trust management and rights management techniques. Traditional access control technique focuses on all the users who utilise the server to do their job, in a closed system. Trust management technique is used to cover authorization for strangers in an open environment such as the Internet. Rights management technique is client-side oriented which deals with the rights required by the client to access a specific page.

In Osborn et al. (2000), Sandhu et al. (2000), and Sandhu (2001), the authors discuss about Role-Based Access Control (RBAC), which associates permissions and roles. Users are assigned roles based on their responsibilities and qualifications. This approach simplifies the management of permissions. Roles can be created for the various job functions in an organization and users could be assigned roles based on their responsibilities and qualifications. There are provisions for moving users from one role to another, when there is a change in job specification for a concern user. Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles as needed. As per Ferraiolo et al. (2001), hierarchical RBAC is used to define the relationship between the senior and the junior, whereby the senior roles acquire the permissions of their juniors, and junior roles acquire the user membership of their seniors.

According to Murthy et al. (1998), a firewall can be made of an external router along with a bastion host. A bastion host is the important component of a firewall that performs the tasks of user authentication, machine verification, logging of all security events to an internal host and the execution of proxy servers for all allowed services. The most serious drawback of this kind of system is that if the traffic through the host increases, the system might be overloaded. Thus Murthy et al. (1998) proposed the concept of smart filter. The smart filter reads the header of a session's first packet, compares it to the rules and, if approved, routes successive packets through a cache. While each packet files through the cache, the filter compares its header to that of the first packet to verify that it belongs to the same session. Thus, it does not need to verify each packet against the rules.

PARALLELIZING OF ACCESS RULES

If the ACL rules are executed in parallel, then the whole process could be handled within a very short time and so the time taken to process the packets can be minimized. But at the same time, there is a constraint on the conversion of the sequential parsing rules into parallel rules. This is due to the fact that the rules are highly dependent on each other and dividing them into parallel tasks is difficult. It might even end up in danger, if there is alteration of the rules without extreme care.

In order to develop a parallelized system, the rules have to be converted into independent rules. That is, in the truly independent intermediate rules, a packet matches only one particular condition and never matches any other condition. Repetition of executing the same rules again and again should be eradicated. The time needed to parallelize the above rules must be less, so that it is appropriate to incorporate such parallelism. It is necessary because whenever the rules are to be changed, the whole parallel process should be considered.

PROBLEM MOTIVATION

IP packet filtering using optimized-sequential processing, neural network and expert systems was done. In each of the above three methods proposed, the rules are optimized and hence reduce the degradation in packet filtering performance. Time is an important criterion in the IP packet filtering. From the Table 2, out of all the three methods considered, the expert system has an edge over the other methods as it processes a fewer number of rules than the other methods. It is found to be better than the optimized sequential parsing and the neural network methods of packet filtering. But, security lapse was found with both the neural network and expert system implementation. In this paper, we try to improve the performance of neural network system by applying different techniques and then take care of the security lapse also.

TABLE 2. Comparisons of proposed methods

	Sequential [Comparison]		Opt. Sequential [Comparison]		Neural Network		Expert System [Comparison]	
	Min	Max	Min	Max	Addition	Multiplication	Min	Max
Ex1	12	60	5	25	7	11	1	4
Ex2	8	24	7	21	11	15	1	7
Ex3	12	48	9	27	12	30	1	10
Ex4	12	84	10	70	13	31	1	14
Ex5	12	290	9	210	12	28	1	18

Security lapse was noted for both neural network and expert system oriented packet filtering. To solve the security lapse, we analyse the correctness of the trained neural network oriented IP packet filtering and its performance impact. As the use of access control rules alone for neural network does not provide correct output as required by the packet filtering system, different neural network systems were designed and trained in the following ways:

1. Using Access Control rules.
2. Using Access control rules and replacing wildcards with 0 and 255.
3. Using Access control rules in binary format.
4. Using Access control rules and wildcards with 0 and 255 in binary format.
5. Using Optimized Access Control rules.
6. Using Optimized Access control rules and replacing wildcards with 0 and 255.

7. Using Optimized Access control rules in binary format.
8. Using Optimized Access control rules and wildcards with 0 and 255 in binary format.

As the number of inputs and in turn the architecture (number of input and hidden neurons) of the neural network with the above-mentioned methods varies, the training process of the neural network also varies. These trained neural network systems were tested for consistency with the actual action from the ACL rules described in Tables 3, 4, 5, 6, and 7. The selected test data is used to check the correctness of the trained neural network system. The security and performance aspects of the trained system were determined based on:

1. Whether the system attained the maximum allowed error during the training process.
2. The result of the trained system is compared with the action based on the ACL rules.
3. Amount of time taken in terms number of iterations or epochs by the neural network system to train.
4. Number of input needed to train the system along with the architecture of the network.
5. Number of rules and its impact on the training of the system.
6. Use of wildcards and how neural networks is affected by that use.
7. Impact of optimization of the access rules and the conversion of the access rules into binary.

TABLE 3. Experiment 1 ACL rules

Conditions						
Packet Type	Source IPv4 address	Source TCP/UDP port	Destination IPv4 address	Destination TCP/UDP port	ACK bit	Action
TCP	*	*	202.185.*.*	*	Established	Permit
TCP	*	*	202.185.*.*	53	*	Permit
TCP	*	*	202.185.33.44	25	*	Permit
TCP	*	*	202.185.55.66	119	*	Permit
IP	*	*	*	*	*	Deny

* Not Available

TABLE 4. Experiment 2 ACL rules

Source IP address	Destination IP address	Action
10.1.2.1	10.1.1.*	Deny
10.1.2.*	10.1.3.*	Deny
*	*	Permit

TABLE 5. Experiment 3 ACL rules

Conditions						
Packet Type	Source IP address	Source TCP/UDP port	Destination IP address	Destination TCP/UDP port	ACK bit	Action
TCP	*	*	10.1.1.2	www	*	Permit
UDP	0.0.0.*	*	10.1.1.1	*	*	Deny
IP	10.1.2.*	*	10.1.3.*	*	*	Deny
IP	*	*	*	*	*	Permit

TABLE 6. Experiment 4 ACL rules

Conditions						
Packet Type	Source IP address	Source TCP/UDP port	Destination IP address	Destination TCP/UDP port	ACK bit	Action
TCP	10.1.1.2	WWW	*	*	*	Permit
UDP	10.1.1.*	*	0.0.0.*	*	*	Deny
IP	10.1.3.*	*	10.1.2.*	*	*	Deny
IP	10.1.2.*	*	10.1.3.*	*	*	Deny
IP	10.1.1.130	*	10.1.3.2	*	*	Deny
IP	10.1.1.28	*	10.1.3.2	*	*	Deny
*	*	*	*	*	*	Permit

EXPERIMENT

In an attempt to improve the performance of IP routers in forwarding packets, we propose to use the back-propagation oriented neural network algorithm to train the network to learn the ACL rules as demonstrated below.

USAGE OF BACKPROPAGATION NETWORK

The neural network algorithm proposed for this experiments is designed to be real-time because any addition to ACL rules means that the whole ACL list needs to be changed and there is need for a new training process. The network sees the N most recent patterns through a shifting time window. It learns to predict the next value of the series. First, the network is trained to the point of convergence using a set of access rules previously designed. During this process, the weights are updated from a random start configuration to the values corresponding to the desired transfer function. Then, this function is inserted online and works well without further modification.

The disadvantage of a neural network is that it takes a long time for training depending upon the various sets of patterns provided to it. In the case of general neural networks, it is possible that the already trained set of data is affected by a newly arriving training pattern. In order to avoid this kind of discrepancy, a part of the output is fed back to the input of the network.

TABLE 7. Experiment 4 ACL rules

Conditions						
Packet Type	Source IPv4 address	Source TCP/UDP port	Destination IPv4 address	Services	ACK bit	Action
*	*	*	*	finger, bootp, udp-525, ident, login	*	Deny
*	202.185.128.*	*	202.185.11.*	http, smtp	*	Deny
*	*	*	202.185.*.*	smtp, imap, pop3	*	Deny
*	*	*	*	smtp, imap, pop3	*	Permit
*	202.185.131.*	*	*	*	*	Permit
*	202.185.128.*	*	202.185.130.1	http, tcp	*	Permit
*	*	*	202.185.130.3	http, tcp	*	Permit
*	202.185.130.2	*	202.185.128.*	netbeui	*	Permit
*	*	*	202.185.130.4	ftp	*	Permit
*	202.185.128.*	*	202.185.131.178 202.185.131.179	ftp	*	Permit
*	*	*	202.186.130.1	rpc, syslog tcp	*	Permit
*	202.185.130.4	*	202.185.130.5	*	*	Permit
*	202.185.128.*	*	202.185.130.5	telnet	*	Permit
*	202.185.131.170	*	202.185.129.120	*	*	Permit
*	202.185.128.* 202.185.130.1	*	202.185.131.*	http, ftp, tcp, telnet	*	Permit
*	*	*	202.185.130.5	tcp, dns, http, nntp	*	Permit
*	202.185.128.*	*	*	tcp, http	*	Permit
*	*	*	202.185.130.1 – 202.185.130.5	http, tcp	*	Permit
*	202.185.129.*	*	*	dns, nntp, http, tcp, spool	*	Permit
*	202.185.128.*	*	202.185.131.*	snmp, icmp, echo	*	Permit

In the case of a back propagation network, the training phase needs both the input and their corresponding output. So, the network is made of varying inputs and one output. The number of inputs varies with regard to the different forms of input provided to the system, like binary or optimized. The "newff" (Feed forward back propagation) function was used to train the network and the trained network is simulated to identify the output for any specific input pattern. 'Newff' is a MATLAB oriented function which posses the training features like epochs, learning rate, momentum factor and required goal. As sample representation is as follows:

net = newff([0 3; 0 255;0 255;0 255; 0 1],[1,1],{'logsig', 'purelin'}, 'traincgb');

% 5 indicates five patterns

```
% for opt seq ex. 1
net.IW{1,1} = rands(1,5);
% first 5 is the number of columns; second 5 is the number of rows
net.LW{2,1} = rands(1,1); % five patterns with one output each

% number of epochs allowed
net.trainParam.epochs = 500;
% learning rate
net.trainParam.lr = 0.1;
% momentum factor
net.trainParam.mc = 0.5;
% goal that ought to be attained to stop the training process.
net.trainParam.goal = 0.0001;
```

If in one case, any one of the input parameters is absent then the wild-card character is used, which represents two input patterns - the lower (value as 0) and upper bound (value as 255) of the corresponding parameter. Only one output neuron is used to identify a permit or deny.

The weights are randomly chosen during initialisation of the network. The network is trained to give the same output for the two extreme values of any input, which is represented by wild-card character. This has been ascertained using the justification that the output for any intermediate input value will be the same as the output that it's got when the extreme values are used to train a system to attain a fixed value.

NEURAL NETWORK BASED IPV4 FILTERING

Originally, as per the data available from the packet, the number of input parameters is twelve. These are Packet Type (One input), Source IPv4 address (Four inputs, standing for /8, /16, /24 and /32 part), Source UDP/TCP Port (One input), Destination IPv4 address (Four inputs, standing for /8, /16, /24 and /32 part), Destination TCP/UDP Port (One input) and Acknowledgement bit (One input). These twelve input parameters are to be processed for identification, by the router.

In the case of neural networks, the wild-card character is represented by two input patterns the lower and upper bound of the corresponding parameter. Only one output neuron is used to represent the Permit or Deny operation. The network is trained to give the same output for the two extreme address values 0 and 255 and hence any value between 0 and 255 will give rise to the same output. This has been ascertained using the following mathematical justification.

In a neural network, the net input to a neuron is given by,

$$Net = w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n \qquad (1)$$

where Net is the net input to the neuron
$x_1 \ldots x_n$ are the inputs
$w_1 \ldots w_n$ are the weights between the nodes

A typical neural network diagram for Back Propagation Algorithm is shown in Figure 1.

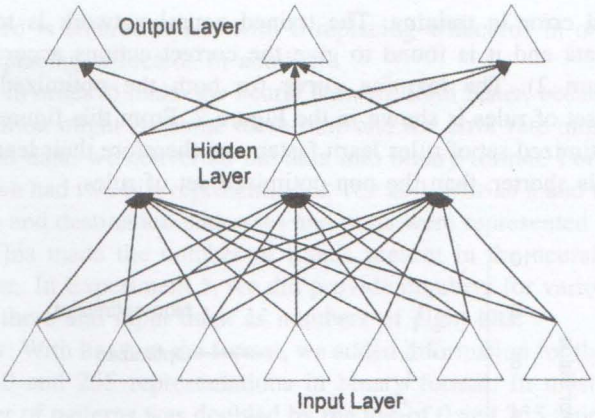

FIGURE 1. Back propagation algorithm

After optimization is applied, only five parameters are used for training a neural network. They are Packet Type, Destination IPv4 address, Destination TCP/UDP Port and ACK bit. For Packet Type, 1 represents TCP, 2 represents UDP and 3 represents IP. For Destination IPv4 address, we have two inputs standing for /16 and /32 parts of the address. For Destination TCP/UDP Port, we have only one input. For ACK bit, Established is represented as 1, wildcard "*" as 0.

For processing the optimized rules, a back propagation Neural network is trained. The Neural Network has five input neurons, two hidden neurons and one output neuron. The selection of two hidden neurons is based on the following previous works:

1. "A rule of thumb is for the size of this hidden layer to be somewhere between the input layer size and the output layer size." (Blum 1992)
2. "How large should the hidden layer be? One rule of thumb is that it should never be more than twice as large as the input layer." (Berry 1997)
3. The number of hidden neurons should be less than the number of input neurons. (Lawrence 1997)
4. The number of hidden neurons is calculated as (Kwon & Kirby 1997):

$$\textit{Number of hidden neurons} = (\textit{Number of input} + \textit{number of output})/2 \quad (2)$$

Thus with five input neurons and one output neuron, we can have three hidden neurons as per (Kwon & Kirby 1997). We preferred to have two hidden neurons after testing the neural network with different number of hidden neurons. With hidden neurons = 1, the number of epochs was 41 but the data was trained well. With hidden neurons = 2, the number of epochs was 8 with actual output as 0.9955 instead of 1 [target output]. With hidden neurons = 3, the number of epochs was 17 with actual output as 0.9928 instead of 1. With hidden neurons = 4, the number of epochs was 12 with actual output as 1.0033 instead of 1. With hidden neurons = 5, the number of epochs was 10 with actual output as 0.9884 instead of 1. From the above, it is clear that the use of two hidden neurons perform better in terms of

epochs and error in training. The trained neural network is tested using arbitrary data and it is found to give the correct actions according to the rules (Figure 2). The learning curve for both the optimized and non-optimized set of rules is shown in the Figure 2. From this figure, it is clear that the optimized set of rules learn faster, and therefore their learning curve time span is shorter, than the non-optimized set of rules.

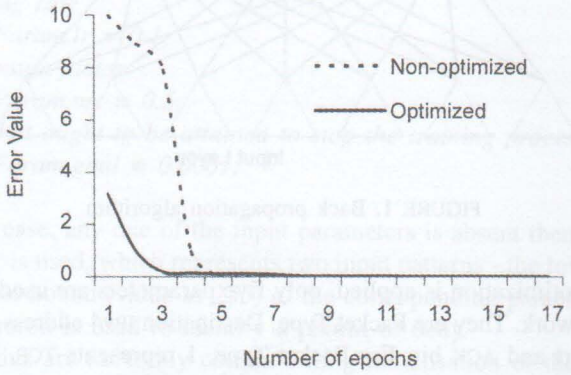

FIGURE 2. BPN Learning Chart

EXPERIMENTAL DATA AND ANALYSIS

Here, we select a set of rules that are based on the extended access list format where only the source and destination addresses are used. Five sets of data were taken among which four are shown here.

ACL RULES ORIENTED NEURAL NETWORK ANALYSIS

The neural network system was designed with the access control rules discussed above. These rules were represented into the neural network system with back-propagation network algorithm. The network with inputs and action as the output was trained for all the five experimental access rules set presented in Tables 3, 4, 5, 6, and 7. From the tables we can see that Experiments 2 and 4 alone have implicit rules (rules that posses no specific conditions but only indicate the action for those packets that don't satisfy the other access rules) at the end of the access lists. The number of rules in each access list varies from 3 to 21. These five experiment access lists are trained with eight different forms as input to the neural network system. The output of the system is either 0 or 1 indicating deny or permit respectively. According to the form of the input, the system architecture also changes as shown in Table 8. The following are the various forms of input for the neural system:

1. ACL: The access list data were input as in the access list and replacing the wildcard by 0. Here, the problem was noted in the case of totally wildcard rules like that in Experiment 2 because all the inputs are zero there.
2. ACLW: In order to cater for the range of the wildcard, we represented the wildcards with two sets of inputs as 0 and 255. Any access rule with

wildcard is entered twice, with 0 replacing wildcards in one rule and 255 replacing wildcards in another.

3. BACL: In order to make the neural network train faster, because decimal calculation might consume more time and the error rate might high for decimal data, we converted the data into binary format. For the packet type, we had two bits representing it, TCP as 1, UDP as 2 and IP as 3. The source and destination addresses and ports were represented using eight bits. This made the number of inputs present in the neural system to increase. In Experiment 5, we did provide numbers for various services given there and input them as numbers of eight bits.
4. BACLW: With BACL as the format, we added information for the wildcards using 0 and 255 representations in binary format. In most cases, the number of patterns was doubled by the use of 0 and 255 representations of the wildcards.
5. ACL, ACLW, BACL and BACLW were done with optimized rules and they are named as OACL, OACLW, BOACL and BOACLW, respectively.

The network performance was noted for how long it takes for the neural network to train and whether the training process was completed successfully. The number of epochs (iterations) indicated whether the performance goal was met or not while the architectural representation of the network was shown in Table 8. Performance goal indicates the maximum allowed error between the output generated by the neural network system and the actual ideal output. In the case of architecture, we note the number of inputs present in each pattern. This number of inputs indicates the number of input neurons in the neural network system and the number of patterns indicates the number of output neurons needed for the neural network system.

From Table 8, we can infer that the neural network system has learnt well in most cases, and the performance goal was met not only in three cases. The neural network stops the training process when the performance goal is met (which means the error between the output of the neural network and the actual output is less than the allowed error) or when the change in weights is minimum and that this change does not make the neural network learn further. From this table, we can also infer that the binary form of neural network requires more iteration to train due to the increase in number of inputs. The number of rules present in each experiment determines the number of patterns. Even though converting to binary increases the number of input nodes and with it the complexity of the network architecture, the number of epochs doesn't seem to be affected much due to the fact that the binary operations are performed faster than the decimal operations.

After training the neural network in various different forms as mentioned earlier, we tested the system with three data sets for each experiment. These data sets where selected to see the consistency of the trained neural network system with the actual results the access control rules provide. The test data set for all the experiments is shown in Tables 9 through 13.

The neural network system with all the above data set was tested for security aspects. The actual output expected for these data sets is shown in Table 14 as 'Idle Value'. The Table 14 is used to get the actual value of the neural system for various settings and compare it with the ideal value

TABLE 8. Performance comparisons

		ACL	ACLW	BACL	BACLW	OACL	OACLW	BOACL	BOACLW
Ex 1	No. of Epochs	5	18	11	33	12	17	10	43
	Goal met?						No		
	Inputs/Patterns	12/5	12/10	83/5	83/10	5/5	5/10	27/5	27/10
Ex 2	No. of Epochs	3	6	6	8	3	5	5	14
	Goal met?								
	Inputs/Patterns	8/3	8/5	64/3	64/5	7/3	7/4	56/3	56/4
Ex 3	No. of Epochs	8	25	8	20	10	72	15	18
	Goal met?								
	Inputs/Patterns	12/4	12/8	84/4	84/8	9/4	9/8	56/4	56/8
Ex 4	No. of Epochs	26	24	14	23	14	38	21	23
	Goal met?								
	Inputs/Patterns	12/7	12/13	83/7	83/13	10/7	10/13	74/7	74/13
Ex 5	No. of Epochs	27	163	49	59	25	50	53	57
	Goal met?		No				No		
	Inputs/Patterns	21/12	21/42	83/21	83/42	9/21	9/42	72/21	72/42

TABLE 9. Test Data for experiment 1

Conditions						Action
Packet Type	Source IP address	Source TCP/UDP port	Destination IP address	Destination TCP/UDP port	ACK bit	
IP	11.12.13.14	15	21.22.33.44	119	1	Deny
TCP	11.12.13.14	15	202.185.25.100	53	0	Permit
TCP	11.12.13.14	15	21.22.100.100	100	1	Permit

TABLE 10. Test Data for experiment 2

Source IP address	Destination IP address	Action
10.1.3.4	10.10.10.7	Permit
10.1.2.5	10.1.1.7	Permit
10.1.2.1	10.1.1.7	Deny

TABLE 11. Test Data for experiment 3

Conditions						Action
Packet Type	Source IP address	Source TCP/UDP port	Destination IP address	Destination TCP/UDP port	ACK bit	
TCP	10.10.10.10	10	10.1.1.2	www	0	Permit
TCP	10.10.10.10	10	10.1.1.2	www	1	Permit
IP	10.1.2.100	100	10.1.3.100	100	0	Deny

TABLE 12. Test data for experiment 4

Conditions						
Packet Type	Source IP address	Source TCP/UDP port	Destination IP address	Destination TCP/UDP port	ACK bit	Action
TCP	10.1.1.2	80	10.10.10.10	100	1	Permit
IP	10.1.3.25	100	10.1.2.25	100	0	Deny
IP	10.1.1.28	100	10.1.3.2	100	1	Deny

TABLE 13. Test data for experiment 5

Conditions						
Packet Type	Source IP address	Source TCP/UDP port	Destination IP address	Services	ACK bit	Action
TCP	202.185.128.100	100	202.185.131.100	ftp	0	Permit
TCP	202.185.128.50	50	202.185.130.5	telnet	1	Permit
TCP	202.185.128.4	50	202.185.130.5	http	0	Permit

expected. A system is said to have learnt well and considered as well suited for this operation if the actual output acquired is in close proximity of the ideal value. The output from the various neural network systems is shown in Table 14:

1. The neural network that is of ACL rules alone has not trained the system and so none of the experiments gave accurate result for all the three test data sets.
2. Experiment 2 was not properly learnt due to the following rules:
   a. Presence of very few numbers of rules in the ACL rules list and one of them is an implicit permit. Being implicit makes the neural network difficult to learn. Representing the implicit as 0 makes the neural network learn only for 0 or closer values and representing the implicit as 0 and 255 makes the neural network confuse the other set of rules. The impact of sequential set of rules also has an impact on the neural network training process.
   b. Referring to the number of epochs from Table 8, the value is less compared to other experiments. This doesn't mean that the neural network has trained faster and better. Fast learning doesn't mean that the neural network has learnt better.
3. Optimizing the ACL rules and using them to train the neural network leads to some improvement. Experiment 3 and 5 are trained using OACL. But the impact by representing 0 and 255 as wildcards to OACL is not much. The outcome of the system is the same with OACL and OACLW. So, there is no need of replacing wildcard with 0 and 255 that causes the increase in the number of patterns and make the architecture of the neural network complex.

TABLE 14. Security Comparison among neural network packet filtering systems

		Idle Value	ACL	ACLW	BACL	BACLW	OACL	OACLW	BOACL	BOACLW
Ex 1	Test Data 1	0	0.8823	0.3403	0.4189	0.8356	1.0077	1.0000	0.5381	0.7244
	Test Data 2	1	0.9942	0.9984	1.1913	0.9628	0.1107	0.8021	1.0986	1.1419
	Test Data 3	1	0.8823	1.9925	0.7438	0.5051	1.0077	0.8000	1.1508	1.1945
Ex 2	Test Data 1	1	-0.0427	-0.0215	0.0577	0.0817	0.4743	0.5788	1.6736e-005	0.1026
	Test Data 2	1	-0.0536	-0.0274	-0.0299	0.0088	-4.9951e-004	0.0467	1.8616e-005	0.0018
	Test Data 3	0	-0.0522	-4.8718e-004	0.0292	0.0061	3.4158e-005	0.0039	1.9675e-005	-5.4934e-004
Ex 3	Test Data 1	1	1.0012	0.2060	1.1292	0.4495	1.0007	1.0128	0.4626	0.8048
	Test Data 2	1	1.0012	0.1965	1.2884	0.5794	1.0007	1.0128	0.4626	0.8048
	Test Data 3	0	1.3233	-0.0071	0.0434	-0.0632	0.2597	0.0031	-0.0119	0.3286
Ex 4	Test Data 1	1	0.2538	0.0925	0.4244	0.7866	0.9913	0.9870	-0.0412	1.1817
	Test Data 2	0	0.2700	-0.0037	0.3800	0.0533	0.9913	0.6910	0.0328	-0.3673
	Test Data 3	0	1.0119	-0.0057	-0.1730	-0.2159	-0.2101	1.1868	0.0044	-0.2688
Ex 5	Test Data 1	1	0.3251	2.6991	1.1904	0.4980	0.8116	1.0363	0.9806	0.7618
	Test Data 2	1	0.9801	1.9310	1.2490	1.3889	1.0013	0.8803	1.0240	0.9234
	Test Data 3	1	1.0024	0.9749	0.8188	0.3861	1.0013	1.0253	1.0066	0.9975

4. The use of binary input for the OACL makes some difference in the output values but the permit/deny decisions remains the same.
5. The case of OACLW and ACLW for Experiment 5 is ambiguous because the system couldn't meet the performance goal. This is due to the fact that the neural network would have fallen into the local minima and so the weights cannot be further adjusted to train the network better.
6. The relationship between the number of epochs and training is normally that with the increase in the number of epochs, the system trains better. But care must be taken that the above applies only when the performance goal is met. The case of OACLW for Experiment 3 is an example.

From Table 14, none of the neural network model can identify the entire test set for all the experiments properly. Only the BOACLW could learn three of the experiments properly. Experiment 2 is not taken into consideration as it has less number of rules and one of them is of implicit type. For Experiment 1 (in the case of BOACLW) with little problem of identifying the deny option, the network has learnt better than the ACL rules. As the system is not complete, we include a next level of security check using the local data obtained from the network and also the hourly analysis of the network usage.

CONCLUSION

In this experiments, different neural network architectures based on back propagation algorithm were tested and trained with eight input sets. Those neural network architectures were analysed for the performance of the training process and the correctness of their learning. Analysis indicates that neural network could not learn all the experiments well and the presence of implicit rules for denial/permission plays a pivotal role in the training of the neural network. In order to improve further on the security aspects, we designed neural network that could be trained with local user data and the hourly hits on servers. After the training, we are able to conclude that the neural network learns better with binary input data but at the same time this binary input data increases the input nodes and thereby the complexity of the network. Improving the security aspects of the neural network with various sets of data along with the binary form affected the performance of the neural network system.

ACKNOWLEDGEMENT

We hereby wish to thank the reviewers for providing us with valuable suggestions for future work [Genetic algorithms] and also commenting on the motivation behind this work [i.e., to compare with different other packet filtering techniques].

REFERENCES

Berry, M J. A., & Linoff, G. 1997. *Data mining techniques*. New York: John Wiley & Sons.

Blum, A. 1992. *Neural networks in C++*, New York: Wiley.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. 2001. proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security* 4(8): 224-274.

Gittleson, H., Sharp, R. & Cheswick, B. 1998. *Red-hot firewalls. America's Network* **1**(102): 48-52.

Lawrence, S., Giles, C. L. & Tsoi, A. C. 1997. Lessons in neural network training: overfitting may be harder than expected. *Proceedings of the Fourteenth National Conference on Artificial Intelligence* 1: 540-545.

Miei, T., Maruyama, M., Ogura, T. & Takahashi, N. 1997. Parallelization of IP-packet filter rules. *Proceedings of Third International Conference on Algorithms and Architectures for Parallel Processing '91* 1: 381-388.

Murthy, U., Bukhres, O., Winn, W. & Vanderdez, E. 1998. Firewalls for security in wireless networks. *IEEE Proceedings of 31st Hawaii International Conference on System Sciences '98* 7: 672-680

Odom, W. 2000. *Cisco CCNA Exam #640-507 Certification Guide*. Cisco Systems.

Kwon, O. and Kirby, E. 1997. Farm Appraiser: A Neural Network for Agricultural Appraisal, Proceedings of the 1997 Annual Association for Information Systems (AIS) Conference, Indianapolis, Indiana, August 15-17, pp. 715-717.

Osborn, S., Sandhu, R. & Munawer, Q. 2000. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security* 3(2): 85-106.

Park, J. & Sandhu, R. 2002. Towards usage control models: beyond traditional access control. *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT '02)* 1: 57-64.

Provos, N. 2002. *Improving host security with system call policies*. CITI: Technical Report 02-3. Center for Information Technology Integration, University of Michigan.

Sandhu, R. 2001. Future directions in role-based access control models. *Proceedings of Mathematical Methods, Models and Architecture for Computer Networks Security* (MMM-ACNS-2001), Russia, May 21-23, [Lecture Notes in Computer Science, Springer-Verlag, Volume 2052/2001] pp. 22-26.

Sandhu, R., Ferraiolo, D. & Kuhn, R. 2000. The NIST model for role based access control: towards a unified standard. *Proceedings of 5th ACM Workshop on Role Based Access Control*, Berlin, Germany, July 26-27, pp. 47-63.

Sandhu, R. & Park, J. 2003. Usage control: a vision for next generation access control. *Proceedings of Mathematical Methods, Models and Architecture for Computer Networks Security* (MMM-ACNS-2003), Russia, September 21-23, [Lecture Notes in Computer Science, Springer-Verlag, Volume 2776/2003] pp. 17-31.

Teresa, F. L., Tamaru, A., Gilham, F., Jagannathan, R., Neumann, P. G. & Jalali, C. 1990. IDES: Progress Report. *Proceedings of the 6th Annual Computer Security Applications Conference '90* 1:273-285.

M. I. Buhari
King Fahd University of Petroleum and Minerals
Dhahran
Saudi Arabia
Email: mibuhari@ccse.kfupm.edu.sa

M. H. Habaebi
Burhanudin Mohd. Ali
Department of Computer and Communications Engineering
University Putra Malaysia
43400 UPM Serdang, Selangor D.E
Malaysia
Email: {hadi;borhan}@eng.upm.edu.my