Packet-Rule Cost Weighing Method for Best Organisation of Access Lists in Packet Filtering

Faheem Bukhatwa
Department of Computer Science
University College Dublin
Belfield, Dublin 4, Ireland. ***faheemfb@gmail.com*.**

ABSTRACT

Access list packet classifiers and filters are efficient, fast and have withstood the test of time. They also provide a good level of security. As time passed, communication is greatly expanding and the number of users continues to increase. The number of attacks from the Internet is also increasing. All this places more pressure on packet classifiers and filters to provide more filtering and greater security at higher performance levels without becoming the bottleneck. Packet classification and filtering is also becoming popular in many other areas of communication systems. Examples are: rate limiting, access control, resource reservation, routing, service differentiation, Virtual Private Networking (VPN), IP security gateways, load balancing, traffic shaping and Quality of Service (QoS). This enforces the need to search for better or more efficient methods of implementing packet filtering. Organising the rules in an access list according to their class is a way of improving performance for access list-based packet filters. One problem with this method is the large number of different permutations of the rules' classes in an access list for each individual pattern of arriving packets. In this paper we propose a method that obtains the best organisation of the classes of rules in an access list that will guarantee the best performance. This method is based on classifying the rules in the access list and obtaining the relative processing cost of each rule. By obtaining the necessary parameters for the access list and the packet stream, the average processing cost is calculated for all packets in the packet stream needed to pass through the access list in every possible permutation of the rules. The access list classes permutation that yields the lowest average packet processing time will be the best permutation possible for the particular packet stream. We are calling this method, "The Packet-Rule Cost Weighing method PRCW".

Keywords: Packet filtering, access lists, packet classification and cost weighing.

1. INTRODUCTION

In recent years, advances in computing and telecommunication technologies have expanded computer systems capabilities and greatly expanded user requirements. As a consequence, users and their organisations are becoming more and more dependent on the services provided by their systems and computer networks. Data, programs and information critical to the functioning or even survival of an organisation are kept on computer systems and exchanged over telecommunication facilities. This trend raises the need for secure systems for processing and exchanging the information.

Firewalls are used to protect networks, by being situated strategically at a single security screening station where the private network or the Intranet connects to the public Internet, see Figure 1. Incoming or outgoing packets are filtered to *allow* or *deny* each packet access to the network. Firewalls are also used to isolate and protect sub-networks. A firewall is a computer, router or other communication device that filters access to the protected network [8]. Cheswick & Bellovin [10] define a firewall as a collection of components or a system that is placed between two networks and possesses the following properties:

- All traffic from inside to outside, and vice-versa, must pass through it.
- Only authorised traffic, as defined by the local security policy, is allowed to pass through it.
- The firewall itself is immune to penetration.

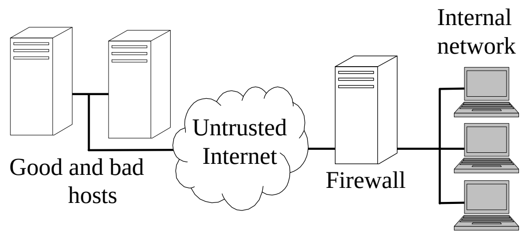


Figure 1: A typical firewall set up.

In packet filtering firewalls, packet filtering refers to the basic operation performed by the firewall to inspect the packet header, verifying any number of the fields in the packet header, i.e. the IP address, the port or both and then accepting or rejecting the packet. Filtering can be applied to incoming or outgoing packets or both. Packet filtering is transparent to the users or independent of the user's knowledge or intervention. Habtamu [5] states that firewalls of this type are cheap, simple, fast, efficient and provide a good level of security. Packet filtering has proved to be efficient and effective at improving system security [9]. Packet filters do not require client computers to be specifically configured; the packet filters do all of the work. But the cost of filtering involved may still be a significant bottleneck when a lot of inspections need to be performed on a lot of packets [1]. Packet filters typically manipulate (that is, accept or reject) packets based on a combination of some of the packet fields. A real example of a rule for a Cisco router [10] is as follows:

Access-list 101 permit TCP 20.9.17.8 0.0.0.0 , 121.11.127.20 0.0.0.0 , range 23 27

The rule indicates that any TCP protocol packet with an IP source address 20.9.17.8 and destination of IP address 121.11.127.20 is to be accepted provided the destination port address is in the range 23..27.

2. BACKGROUND OF REORGANISING ACCESS LIST RULES

All implementations of access list firewalls comprise a list of rules that are applied to inspect the incoming or outgoing packets. Most implementations of packet filtering incur lookup latency linear in the number of rules in the access list. The ultimate aim is to reduce to a minimum the processing time needed for each packet to be inspected, with a reasonable cost in memory requirements.

A “critical rule” is “that rule in an access list that identifies the packet being filtered and causes it to be accepted or refused”. There is a processing time cost added every time the packet is inspected by an individual rule until the critical rule is found (if ever). From a performance point of view, it is desirable that each packet meets that critical rule at or near the start of the list. This would reduce the time required for packet inspection and consequently improve performance. If the critical rule for a packet is met at or near the end of the list, or the rule is never met, the processing time for the packet will be very high and performance will degrade. The concept of rearranging the access list rules aims at having most packets meet their critical rule at or near the start of the list of rules [2].

It is feasible to classify access list rules into different classes based on what fields the rules inspect in a packet. For example, the rules can be divided into those that are intended to filter packets by inspecting single fields like: source address, the target address or port number or inspecting any combination of these fields. Ultimately, access list rules are first classified into different classes and secondly, the rules are arranged in the list in such a way that rules of a particular class are grouped. The class of rules is placed in some order in relation to other classes. The maximum number of unique combinations for one given number of fields (k) selected from any one given available number of fields (n) can be calculated as:

$$c_n^k = \frac{n!}{k!(n-k)!}$$

Where c is the maximum number of combinations, n is the number of fields to select from and k is the size of each combination.

Previous studies have shown that the maximum number of unique combinations (c) that can be obtained using 8 fields (n) to select from to extract 1, 3 or 4 fields (k) is 134 [2].

Consider that rules are ordered in a particular way such that all rules of a certain class, let us say class “X” are made to be located at the top of the access list. When packets start arriving, and if all arriving packets or most of them require the critical rule for acceptance or refusal to be of class “X”, then our ordering of the rules list was a successful choice and performance is expected to improve. The reason is that all or most packets need only a few rules to be checked at the start of the list before their relevant critical rule is reached, see Figure 2. While, if the list was ordered in such a way that “X” class rules are placed at the end of the list then for each packet arriving, most of the rules in the list have to be checked before eventually reaching the needed “X” class rule at the end of the list. Remember, each rule used to inspect a packet has a time cost attached to it, and the more rules the packet has to pass through the more time is wasted. The ultimate aim here is to reduce this time required for processing each packet.

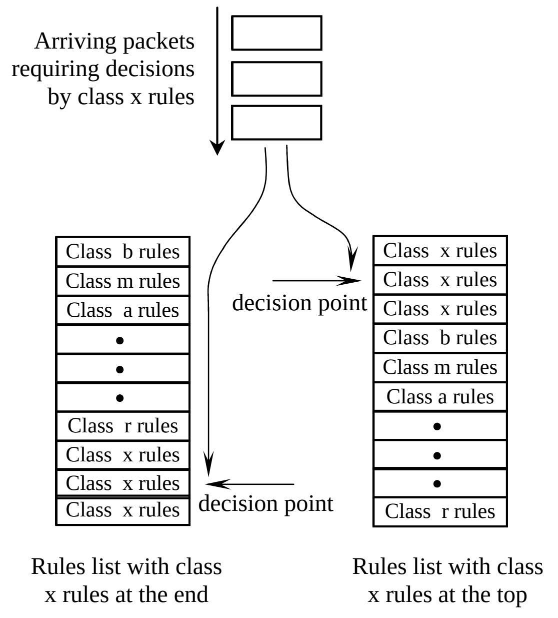


Figure 2: Effects of class arrangement of the rules in a list on processing time.

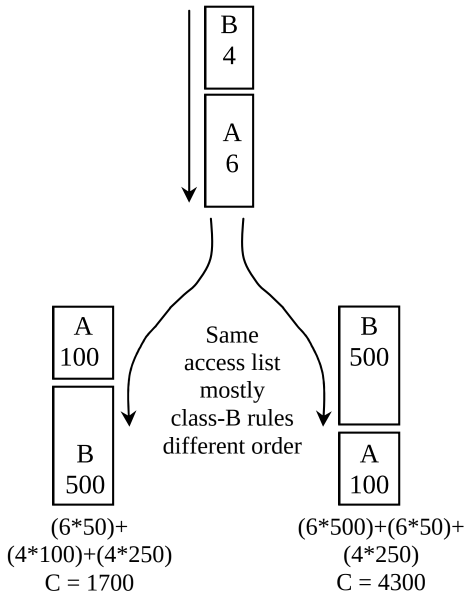
In order to arrange the access list in some order, the characteristics of the arriving packets must be known in advance. A profile of arriving packets can be developed from past observations for a network device, and then it will be possible to make a prediction about the expected pattern of these packets in the future. Profile development is possible and is easily established [4]. Classification of packets arriving in the past can help to determine expected classes and numbers of packets arriving in the future. On the other hand, real time pattern recognition can be done based on classifying arriving packets. Packet patterns can periodically be inspected to determine the most effective order of the rules list.

Changing the order of the rules within the list can in fact change the acceptance or refusal of a packet [6]. Reordering access rules can be a difficult and dangerous operation. The real difficulty is in ensuring that the list is and remains to be an accurate translation of the security policy in as far as permitting those packets intended to pass and blocking those intended to be blocked. Ordered lists require some form of validation to ensure truthful adherence with the security policy. This problem concerning the effects of ordering the list on the security policy is not the subject for discussion in this paper.

3. PACKET-RULES COST WEIGHING METHOD (PRCW)

Many of simulation experiments results obtained did not support the assumed hypothesis. That is obtaining better performance by having a particular class of rules on top of the list when filtering a packet stream most of its packets belong to the same class. Varying levels of performance was noticeable indicating that performance must be determined by more than just placing on top of the access list the rules that are dominant in the packet stream. Extending this idea with the aim to achieve higher performance levels was considered. Looking at the remaining rules in the access list, can they be arranged similarly to improve performance? If the second most common class of packets in a packet stream can be identified, then in the access list the corresponding class rules can be placed second, and so on with the rest. The idea now is to have the all access list rules arranged according to their class to match the same order of the number of packets in a packet stream. Experiments showed more cases now produced better performance but, again but not all cases. In many of the situations inspected, still lower values of processing time were observed where the organisation did not exactly match the number of packets of each class in the packet stream.

It is clear that performance is affected by how the access list rules' classes are arranged. It is suggested to look at the combination of the number of packets in a particular class in the packet stream and the processing time of each of the rules in the access list. Once the processing time for a particular class of packet passing through the list can be determined, then the processing cost for all packets in a particular class can be determined. The number of packets and the processing cost of rules taken together does more accurately influences performance. It will be referred to as the Packet-Rule Cost Weight of the class of packets reflecting the processing cost of some class of packet through rules in the list in a particular organisation.



B 4
A 6
A 100
B 500
Same access list mostly class-B rules different order
B 500
A 100
(6*50)+ (4*100)+(4*250) C = 1700
(6*500)+(6*50)+ (4*250) C = 4300

Figure 3, Better performance with class A on top of the access list.

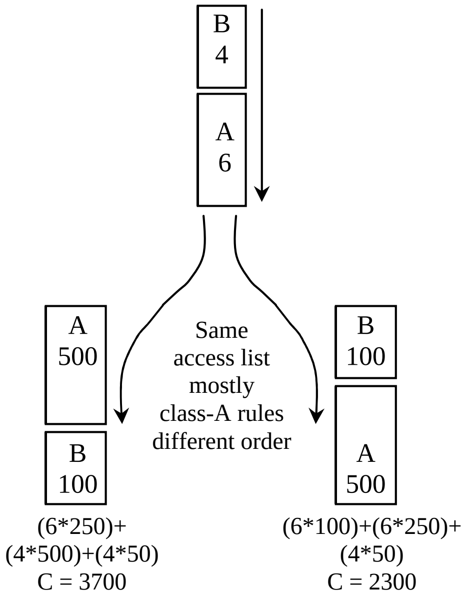
Consider a packet arriving into an access list. It is not entirely true to say that the total processing time for such a packet is alone dependent on the number of rules or the number of packets not on both numbers. The processing time of each rule must be taken into consideration for a more accurate calculation. An arriving packet getting filtered by 5 rules, which require 1 microsecond each, will have the same processing time if it was being filtered by 10 rules requiring ½ microsecond each. Therefore, the total processing time of a class of packets depends on the number of packets, number of rules as well as the processing time of each rule.

Consider a packet passing through a list of rules. If there were *m* rules of one class in the list with a processing time (or an average processing time) *r* for each rule, then the total cost of processing *C,* for a single packet passing through the rules of a particular class in the list would be calculated as: $C = m \cdot r$. If the packet has to pass through more rules belonging to different classes *s* then the total cost can be expressed as:

$$c_{total} = \sum_{s=1}^{n} m_s \cdot r_s \quad \text{where } n \text{ is the number of classes.}$$

But, for the class that contains the critical rule for the packet, processing is different. That is where the rules' class is the same as the packet's class. The critical rule may be located as the first rule in the class, last or somewhere in the middle. Let us assume that on average the critical rule will be in the middle of the class's rules. This means processing time for the class, which will accept or refuse the packet, will be presented as: $c = 0.5 m \cdot r$

Let us consider an example of a packet stream and an access list both having only two classes A and B. The packet stream consists of only 10 packets, 6 packets of class-A and 4 packets of class-B.



B 4
A 6
A 500
B 100
Same access list mostly class-A rules different order
B 100
A 500
(6*250)+ (4*500)+(4*50) C = 3700
(6*100)+(6*250)+ (4*50) C = 2300

Figure 4, Worst performance with class A on top of the access list.

For the access list, the number of rules and processing time of each has been taken into consideration to produce a total cost $\left(m_s \cdot r_s\right)$ for each of the classes A and B.

Figure 3 shows the list with the cost for classes A and B as 100 and 500 units of time respectively, while Figure 3 shows the cost for classes A and B is reversed to 500 and 100 respectively. The access lists on both Figures 3 and 4 are filtering the same packet stream. Based on what we have seen previously, to obtain the better performance it is best to place rules of class A on top of the list because there are more class A packets in the packet stream. With class A rules on top of the list, remember that class A packets passing through the list will only go through class A rules and will not get to class B rules, while class B packets will be processed by class A rules followed by class B rules. In other words, every packet will continue processing until it is processed by a rules' class similar to its class. The example in Figure 3 shows that the total cost is less if class A rules are placed on top of the list (C=1700). While Figure 4 shows the total processing cost is less when class A rules are placed at the bottom of the list (C=2300).

4. CALCULATION OF THE TOTAL COST FOR A PACKET STREAM THROUGH AN ACCESS LIST

Consider a packet stream made up of 3 classes of packets (p_1, p_2 *and* p_3) and an access list of 3 classes of rules with a calculated cost of (r_1, r_2 *and* r_3). Notice that r_x now refers to the cost for all the rules in a class rather that the cost of one rule. The maximum number of permutations for the 3 access list rules' classes would be $3! = 6$ different possibilities. The best permutation for our purpose; would be the one that would produce the lowest processing time (or least cost) with a given packet stream. The cost for each permutation is the total cost for each class of packets passing through the list, as was seen in Figures 3 and 4. With reference to Figure 5, the cost can be calculated for all permutations similar to the following formulae:

$$C_{(r1,r2,r3)} = 0.5p_1r_1 + p_2r_1 + 0.5p_2r_2 + p_3r_1 + p_3r_2 + 0.5p_3r_3$$

$$C_{(r1,r3,r2)} = 0.5p_1r_1 + p_2r_1 + p_2r_3 + 0.5p_2r_2 + p_3r_1 + 0.5p_3r_3$$

$$C_{(r2,r1,r3)} = p_1r_2 + 0.5p_1r_1 + 0.5p_2r_2 + p_3r_2 + p_3r_1 + 0.5p_3r_3$$

$$C_{(r2,r3,r1)} = p_1r_2 + p_1r_3 + 0.5p_1r_1 + 0.5p_2r_2 + p_3r_2 + 0.5p_3r_3$$

$$C_{(r3,r1,r2)} = p_1r_3 + 0.5p_1r_1 + p_2r_3 + p_2r_1 + 0.5p_2r_2 + 0.5p_3r_3$$

$$C_{(r3,r2,r1)} = p_1r_3 + p_1r_2 + 0.5p_1r_1 + p_2r_3 + 0.5p_2r_2 + 0.5p_3r_3$$

The above formulae can be simplified to extract a general format that calculates the cost for any single permutation by subtracting the cost values appearing in all of them: $0.5p_1r_1$, $0.5p_2r_2$ and $0.5p_3r_3$. Now, the general formula to calculate the cost for any number of classes of rules and packet streams for any given permutation *t* can be expressed as follows:

$$timeCost = \sum_{j=1}^{n-1} \left(r_{t_j} \cdot \sum_{z=j+1}^{n} p_z \right)$$

Where *t* is a class permutation, i.e. (r_2, r_1, r_3), *n* is the number of classes, r_{t_j} is the cost for rule class in position *j* in the permutation *t* and p_z is the number of packets of class *z*.

The second part of the algorithm actually calculates the total cost of processing the packets through the access list using each of those permutations. For this purpose two more lists of information are needed, the first is the number of packets in each class in the tested packet stream. The second is the total processing cost of rules for each class in the list. This processing cost would be simplification of two values namely the number of rules in the class multiplied by the average processing cost of the rules in that class.

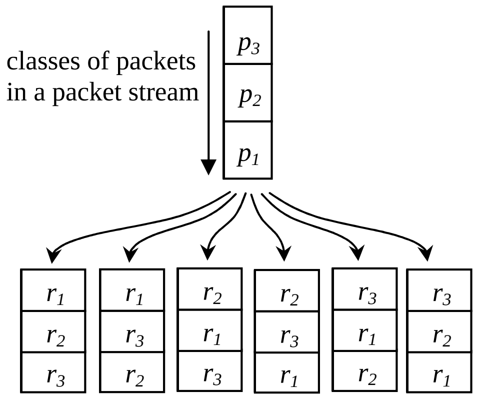


Figure 5, Calculation of cost for all different permutations.

```
minimum_cost = max_double;
// 3 arrays are assumed:
// perm_array[]                    class numbers
// pkt[]           numbers of packets in each class
//al_class_cost[]       cost of rules for each class
cost = 0;
For ( p_ind = 1; p_ind < num of classes; p_ind++)
   // repeat this for each packet class indicated
   // by  pk_index in pkt[] array
   {
      al_index = 1;
      repeat        // for each class in access list array
         {
            actual_index=perm_array[al_index]
                                   // index in the cost array
            if (pk_index == perm_array[al_index])
                // same packet class as access list class
               // half processing cost
                     cost = cost+( pkt[p_ind] *
                     al_class_cost[actual_index] / 2)
               else
                     cost = cost + ( pkt[p_ind] *
                     al_class_cost[actual_index] )
            al_index ++
         }
      until p_ind == perm_array[al_index]
           //  stop when processed access list class
           //  is same as packet class
      if (cost <= minimum_cost)
         {     print ("found a lower cost: ", cost)
               minimum_cost = cost;
               print(permutation_array)
         }
   }
```

Figure 6, calculating the processing cost for a packet stream through one permutation.

The algorithm may be expressed in two parts. The first part produces all the different permutations possible for the access list classes. This is done by recursively shifting the numbers that make up the classes through a recursive function.

The second part of the algorithm is shown in Figure 6 and it is the algorithm for calculating the processing cost for packets filtered by one permutation of an access list. The algorithm uses the arrays: *pkt[ ]* to hold the number of packets of each class in the packet stream, *al_class_cost[ ]* to hold the processing cost of each class of rules and *permutation_array[ ]* to hold the classes' permutation of the access list being tested.

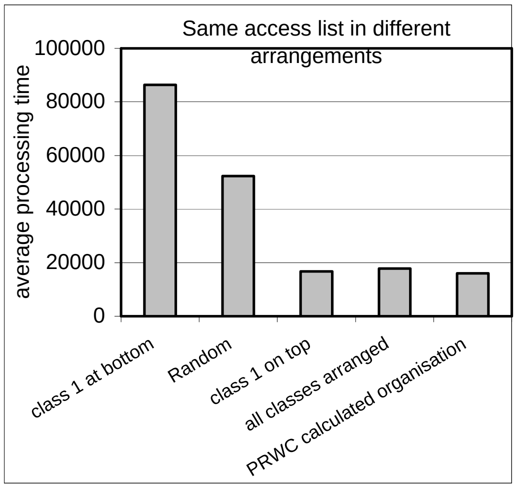


Figure 7, Performance comparison of different organisations.

Finally, Figure 7 provides a quick comparison for performance represented by the average processing time per packet for a number of organisations. The packet stream is made up of mostly class 1 packets. The access list is organised with class 1 rules at bottom and at top of the list. Class 1 at the bottom of the list gave the worst performance. The best performance obtained through the organisation based on the calculation of the processing time using the PRCW method according to the permutation with the lowest cost.

5. CONCLUSION

A key to the success of access list-based packet filtering is to improve its performance. Reorganising the access list rules based on their classifications is one way of improving performance particularly where there is a large number of rules. This performance is due the rearrangement of rules and it is relevant only to one particular characteristic of a incoming stream of packets. Finding out the best arrangement of the classes in an access list for a particular packet stream means calculation of processing costs of the packets through all possible permutations of the access list classes. Packet-Rule Cost Weighing (PRCW) is the new method that has been proposed in this paper to perform the calculation and produce the permutation that will yield the best (lowest) processing cost. This is particularly helpful when considering that the number of possible permutations of the classes in the access list can be extremely large. A 16-class access list can have more than $2*10^{13}$ different permutations.

The PRWC algorithm can be used to produce the cost for each and every permutation possible for a particular characteristic of a packet stream. It removes guessing or randomly testing the different possibilities. This algorithm has been tested and verified in thousands of simulation experiments and has been found to produce the best performance in each and every case. In this paper, a description of the approach was given, followed by how the algorithm was derived. Finally, some of the performance results of the calculations obtained by the algorithm were shown in charts.

A number of future work areas, one is the development of an automatic analyser of packet patterns arriving at a network device and producing the specifications of the characteristics of the packet stream. Another part of the work would be to analyse rules in an access list and produce the specifications of the list, that is the required characteristics of the access list. Those specifications are the input needed for the algorithm to work out the best arrangement of this list for the specific stream to produce the lowest cost. One final point representing an improvement to the algorithm is concerned with the fact that the algorithm performs the calculation for each and every permutation possible. The algorithm makes no attempt to eliminate some of the permutations that can be known in advance to produce large processing time.

6. REFERENCES

[1] Ballew, M. Scott, "**Managing IP Networks with Cisco routers**". O'Reilly, 1st edition, October 1997.

[2] F. Bukhatwa, and A. Patel, "Effects of Ordered Access Lists in Firewalls", **IADIS International Conference WWW/Internet 2003 WWW/Internet 2003**, Algarve, Portugal, 5-8 November 2003.

[3] W.R. Cheswick, and S.M.. Bellovin, **Firewalls and Internet Security, Repelling the Wily Hacker**, Addison-Wesley, 1994.

[4] P. Gupta, and N. McKeown, "Algorithms for Packet Classification", **IEEE Network,** Special Issue, vol. 15, no. 2, pp 24-32, March/April 2001.

[5] Habtamu, and Abie, "An Overview of Firewall Technologies", **Norwegian Computing Centre** January 2000.

[6] S. Hazelhurst, "Algorithms for Analysing Firewall and Router Access Lists", Techincal Report TR-Wits-Cs-1999-5, **Dept. of Computer Science, University of Witwatersand, South Africa,** July 1999.

[7] S. Hazelhurst, "A proposal for Dynamic Access Lists for TCP/IP Packet Filtering". **SAICSIT 2001. Pretoria, South Africa,** 2001.

[8] R. Schreiner, "CORBA Firewall White Papers", **TechnoSec Ltd.**, 1998. Available online at: http://heim.ifi.uio.no/~abie/fwt.pdf, [Last accessed May 2004].

[9] C. L. Schuba, and E.H. Spafford, "A Reference Model for Firewall Technology". **In Proceedings of the Thirteenth Annual Computer Security Applications Conference,** December 1997.

[10] Cheswick, W.R. and Bellovin, S.M., "**Firewalls and Internet Security, Repelling the Wily Hacker**", Addison-Wesley, 1994.