ISSN: 2321-7782 (Online)
Volume 2, Issue 6, June 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Optimization Algorithm for Packet Filtering Firewall

Tushar Subhash Pinjan[1]
Patel College of Science & Technology
Indore – India

Prof. Makrand Samvatsar[2]
Patel College of Science & Technology
Indore – India

Abstract: An optimization algorithm which optimizes the sequence of firewall rules to reduce packet matching time is presented. It has seen observed that some incoming packet can match with more than one rule. This rules are called as dependent rules and if their action differs then it is called as conflict or divergence. Our focus or attention is in this paper is on dependent rules.

This paper proposes an algorithm that is designed for divergence resolution and gives good network performance by reducing the packet matching time of the firewall. The algorithm uses the method of hashing for dividing the rule list into many equal sized sub-rule lists and resolve the divergence by the method of indexing which creates detach list for dependent rules. The performance of the algorithm has enhanced performance over other alternative algorithm in terms of packet matching time.

Keywords: Dependent rules, firewall, network performance, packet matching, conflict resolution, hashing, indexing.

I. Introduction

It has been noted that some incoming packet can match with more than one rule. This rules are called as dependent rules and if their action differ then it is called as conflict or divergence. So while designing rule list of firewall their order must get consider avoiding conflict. At the same time it is necessary to arrange rules in such way that the rule list should give good performance in terms of packet matching time. Again it is necessary to consider that the performance of packet matching time is not getting suffered as the dependency in the rule list increases.

In above papers the performance of firewall in terms of matching time of some incoming packets which are present in list below is decreases as the dependency depth increases. We had tried to overcome this problem in our paper by creating a separate index file for dependent rules .Due to this size of the main list is decreases which results in faster lookup for packet matching which improves the performance of firewall in terms of packet matching time.

We used a Windows XP operating system, 500GB, Hard disk, 4 GB Ram, LAN setup, Java programming language for coding. The techniques used are hashing and indexing for optimizing a rule list of firewall. The aim of the algorithm is to improve the performance of firewall in terms of reducing packet matching as the dependency depth and dependency ratio increases as compare to alternative approach used for firewall rule list optimization.

In our algorithm we are creating separate index file for a dependent rules. We insert all the dependent rules in a separate index file in a order as it present in a un-optimized rule list. Hence the main constraint of the algorithm is the dependent rules present in un-optimized rule list are in correct order because we are referring this order while inserting dependent rules in a index file. If the sequence of dependent rules in a un-optimized rule list which is input to our algorithm is wrong then the same order will be generated in a index file.

This causes a conflict during packet matching for such type of rules and the problem of conflict should not be removed. So the main constraint of the algorithm is rules present in an un-optimized rule list which is input to the algorithm is in correct order otherwise we will failed to achieve our goal.

II. MOTIVATION

The motivation of algorithm is based on the fact that some packet coming to the firewall can match with more than one rule which is called as dependent rule. This kind of rules present in firewall may cause divergence if their action differs hence during optimization we should have to consider a rule dependency to avoid divergence during packet matching process. Our main motivation of optimizing firewall rule list is to give good performance in terms of packet matching time even if the dependency depth and size of the rule list increases. Our algorithm is carried out in two phases. First phase is division phase and second phase is matching phase. In division phase we divide the rule list into equal size sub rule lists by using hashing. The degree of division is depending on the density on the sub rule list. More the density of the subrulelist more division is required. Here we used the concept of indexing for dependent rules. In second phase same hash key is apply on the incoming packet which gives us a subrulelist position in which lookup is made. The algorithm gives a good performance as compare to alternative algorithms in terms of packet matching time. Dependency ratio is the ratio of rules which precedes other rule as compared to total number of rules. Dependency depth is average number of rules present in dependency set.

This paper is organized as follows. Section II defines the related work for firewall rule optimization. Each has been presented their own technique for optimizing rule list by considering different factors again some are produce their own technique for conflict resolution. Section III defines problem definition and the factor which is use for comparison with previous algorithm and present the proposed technique used for rule list division. Section IV discuss the comparison with previous algorithm by showing the results of previous and proposed algorithm and section V conclude the paper and again gives the future work should be done on the related work for further improvement of performance of firewall.

III. PROGRAMMER'S DESIGN

A. Mathematical Model

Problem Statement:- The optimization problem is to reducing cost for a firewall policy consisting of N filtering rules with di as the order (depth) of rule Ri in the policy and wi is a given weight for Ri. Cost is defined as

$$Cost = \sum_{i=0}^{N-1} w_i d_i$$

Here dj is less than dk if Rk is dependent upon Rj preceding it.

Un-optimized rule list is input to the algorithm which produce optimized list which reduces a packet matching time. The motivation of algorithm is based on the fact that some packet coming to the firewall can match with more than one rule which is called as dependent rule. Such type of rules present in firewall may cause conflict if their action differs hence during optimization we should have to consider a rule dependency to avoid conflict during packet matching process. Our main motivation of optimizing firewall rule list is to give good performance in terms of packet matching time even if the dependency depth and size of the rule list increases. Our algorithm is carried out in two phases. First phase is division phase and second phase is matching phase. In division phase we divide the rule list into like size sub rule lists by using hashing. The degree of division is depending on the density on the subrulelist. More the density of the subrulelist more division is required. Here we used the concept of indexing for dependent rules. When we place in a rule in a subrulelist after applying hash key we check its dependency. If the rule is dependent on other rule then we create detach index file and store all these dependent rules in it. We give name of the index file as a reference in a action column.

In second phase same hash key is apply on the incoming packet which gives us a subrulelist position in which lookup is made. The algorithm gives a good performance as compare to alternative algorithm in terms of packet matching time. As we have given reference in action column for dependent rules, it directly goes in a index file for packet matching for such type of rule.

In our algorithm we are creating separate index file for a dependent rules. We insert all the dependent rules in a separate index file in a order as it present in a un-optimized rule list. Hence the main constraint of the algorithm is the dependent rules present in un-optimized rule list are in correct order since we are referring this order while inserting dependent rules in a index file. If the sequence of dependent rules in a un-optimized rule list which is input to our algorithm is wrong then the same order will be generated in a index file. This causes a divergence during packet matching for such type of rules and the problem of conflict should not be removed. So the main constraint of the algorithm is rules present in a un-optimized rule list which is input to the algorithm is in correct order otherwise or aim should not be achieved.

B. Optimization Algorithm

Un-optimized rule list is input to the algorithm which produce optimized list which reduces a packet matching time. The motivation of algorithm is based on the fact that some packet coming to the firewall can match with more than one rule which is called as dependent rule. Such type of rules present in firewall may cause conflict if their action differs hence during optimization we should have to consider a rule dependency to avoid conflict during packet matching process. Our main motivation of optimizing firewall rule list is to give good performance in terms of packet matching time even if the dependency depth and size of the rule list increases. Our algorithm is carried out in two phases. First phase is division phase and second phase is matching phase. In division phase we divide the rule list into many sub rule lists by using hashing. The degree of division is depending on the density on the sub rule list. More the density of the sub rule list more division is required. Here we used the concept of indexing for dependent rules.

- Phase1 algorithm is carried in following steps

1. Generate heap from Un-optimize list
2. extract the top most rule from new list till the list becomes empty
3. apply the hash key on a field and get the sub rule list position
4. Check the sub rule list is full or not
5. if the sub rule list is full apply hash key again till we get the sub rule list which is not full and get the position of sub rule list otherwise go to step 6
6. insert the rule at that sub rule list
7. after insertion check the rule dependency
8. if the rule is dependent then go to step 9 otherwise go to step 10
9. Create separate index file and insert all dependent rule in sequence in a index file. Set action column of rule in sub rule list as a name of index file. otherwise
10. Delete rule and dependent rules from new list.

- Phase2 algorithm is carried out in following steps

1. extract a required field from the packet header
2. apply hash key on that field till get the sub rule list position in which the rule will be found
3. take the action as per given in action column of matched rule in a sub rule list

As per given in figure 1 and figure 2 Our algorithm is carried out in two phases. First phase is division phase and second phase is matching phase. In division phase we divide the rule list into many sub rule lists by using hashing. The degree of division is depending on the density on the sub rule list. More the density of the sub rule list more division is required. Here we used the concept of indexing for dependent rules. When we insert a rule in a sub rule list after applying hash key we check its dependency. If the rule is dependent on other rule then we create separate index file and store all these dependent rules in it. We give name of the index file as a reference in a action column.

In second phase same hash key is apply on the incoming packet which gives us a sub rule list position in which lookup is made. The algorithm gives a good performance as compare to alternative algorithm in terms of packet matching time. As we have given reference in action column for dependent rules, it directly goes in a index file for packet matching for such type of rule.

C. Architecture

Figure 2 shows a simple data flow for matching phase which is described by following steps

1. In step 1 we take a input from a network traffic as a network packet and apply same hash key decided in division phase on a particular field which gives a sub rule list position. Five this position input to the next phase.
2. In step 2 we match packet in a sub rule list and take action accordingly. If action column contains a reference name then we will go in a next phase.
3. In the next phase we go in index file mention in action column of matched rule and match the packet in the index file and take action accordingly. As we are store dependent rules in accurate order the correct action should be performed which avoid conflict.
4. For dependent rules we create divide index file which contains related rules of the rule stored in a sub-rule list. We give the name of index file as a reference to the action column of rule stored in a sub rule list.

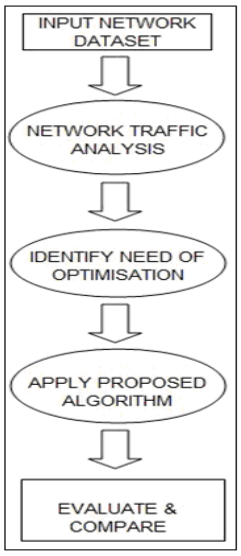

Figure 1: Data flow diagram

IV. RESULTS

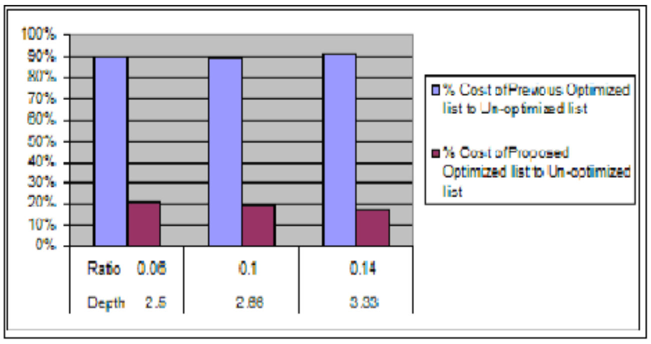

Figure 2 : Graph showing results

We shown results for 50 rules by using previous approach and proposed approach with the help of graph. Here we have calculated the cost of the optimized rule list for different dependency depths and dependency ratios. We got the 9o percent and 20 percent cost of un-optimized list by previous approach and proposed for dependency depth 2.5 and ratio 0.06.We got the 89 percent and 19 percent cost of un-optimized list by previous approach and proposed for dependency depth 2.66 and ratio 0.1.We got the 91 percent and 17 percent cost of un-optimized list by previous approach and planned approach for dependency depth 3.33 and ratio 0.14.

V. Conclusion

We conclude that the cost obtained by using our proposed approach is improved as compare to previous approach. In proposed technique we have created a many sub-rule lists of main rule list by using hashing. The same hashing concept is used during matching process hence during packet matching the lookup is done in final sub rule list which is having less size as compare to the main rule list. Hence searching for matching rule should be faster.

References

1. P. Gupta and N. McKeown, “Algorithms for packet classification,” IEEE Network Magazine, vol. 15, no. 2, pp. 24–32, March/April 2001.
2. E. Al-Shaer and H. Hamed, “Firewall policy advisor for anomaly discovery and rule editing,” in IFIP/IEEE 8th International Symposium on Integrated Network Management, March 2003, pp. 17–30.
3. T. Abbes, A. Bouhoula, and M. Rusinowitch, “An inference system for detecting firewall filtering rules anomalies,” in Proceedings of the 2008 ACM Symposium on Applied Computing, ser. SAC ’08. New York, NY, USA: ACM, March 2008, pp. 2122–2128.
4. V. Capretta, B. Stepien, A. Felty, and S. Matwin, “Formal correctness of conflict detection for firewalls,” in Proceedings of the 2007 ACM Workshop on Formal Methods in Security Engineering, ser. FMSE ’07. New York, NY, USA: ACM, November 2007, pp. 22–30.
5. A. X. Liu and M. G. Gouda, “Complete redundancy detection in firewalls,” in Data and Applications Security XIX, ser. Lecture Notes in Computer Science, S. Jajodia and D. Wijesekera, Eds. Springer Berlin / Heidelberg, 2005, vol. 3654, pp. 193–206.
6. A. X. Liu, E. Torng, and C. R. Meiners, “Firewall compressor: An algorithm for minimizing firewall policies,” in INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, April 2008, pp. 176–180.
7. H. Hamed and E. Al-Shaer, “Dynamic rule-ordering optimization for high-speed firewall filtering,” in Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ser. ASIACCS ’06. New York, NY, USA: ACM, March 2006, pp. 332– 342.
8. J. L. Garcıa-Dorado, J. A. Hern´andez, J. Aracil, J. E. L. de Vergara, F. Montserrat, E. Robles, and T. de Miguel, “On the duration and spatial characteristics of internet traffic measurement experiments,” IEEE Communications Magazine, vol. 46, no. 11, pp. 148–155, November 2008.