STATIC ANALYSIS OF ACCESS CONTROL LISTS

Tomáš Hozza
Master Degree Programme (2.), FIT BUT
E-mail: xhozza00@stud.fit.vutbr.cz

Supervised by: Ondřej Ryšavý
E-mail: rysavy@fit.vutbr.cz

Abstract: Some problems in configurations of network devices are difficult to identify. Access control lists present an important part of many configurations. Conflicts among rules of an access control list can cause holes in security policy or quality of service. In this paper we focus on identifying and classifying conflicts among rules of an access control list. Possible types of conflicts and algorithms necessary to analyze all conflicts in a single access control list of one given device are discussed. Discovering all possible types of conflicts is not a trivial task. We present optimized algorithm using tries, based on existing research by Baboescu and Varghese. We implemented tool using tries based algorithm for detecting conflicts among access control list rules of one given Cisco, HP or Juniper device.

Keywords: static analysis, configuration of network devices, access control list, security policy

1 INTRODUCTION

Errors in configuration can result in complete malfunction of network device or some particular service. The cases when network service is not working properly on first look are quite easy to identify. But sometimes an error in configuration can cause malfunction of a service only in special conditions. Such problems are hard to identify and tools such as simulation of computer networks are not always sufficient. On the other hand, tools such as static analysis can be used to identify some errors in configurations of network devices.

Access control lists (ACLs) present an important part of many configurations. Conflicts among ACL rules can cause holes in security policy or quality of service. In our work we focus on detecting conflicts among ACL rules using static analysis of a device configuration. We are able to detect all possible conflicts between every pair of rules, such as *shadowing*, *correlation*, *generalization*, *redundancy* and *superimposing* [1, 2]. Our solution is designed for five dimensional (source IP address, destination IP address, source port, destination port, network protocol) ACL rules with two possible actions (allow and deny) assuming that rules with less dimensions can be always expressed as five dimensional rules.

2 SOLUTION METHOD

The presented approach consists of two main algorithms to solve the problem of detecting and classifying conflicts among ACL rules. First algorithm detects and classifies a type of conflict between pair of rules or group of rules. For this purpose we chose algorithm based on classification by Al-Shaer [1]. Disadvantage of this algorithm is that it analyzes only pairs of ACL rules therefore analysis of whole ACL can not be done efficiently. There exist also some other algorithms that analyzes conflicts among more than two ACL rules, but do not detect all possible conflicts (mainly correlation) [3]. Our main intention is to detect all possible conflicts therefore we have chosen less efficient but complete conflict classification algorithm.

Second algorithm analyzes all possible pairs of rules in a single ACL. Analyzing conflicts among all rules in one ACL is not a trivial task. For ACLs with small number of rules there is no problem to analyze each pair of rules (time complexity $O(n^2)$ and $\Omega(n^2)$), but in case of large number of rules there is need to find more efficient solution. Baboescu and Varghese presented a method using tries which reduces number of analyzed pairs (time complexity $O(n^2 \log n)$ and $\Omega(n \log n)$) [4]. Worst case is when all rules are in conflict with each other.

2.1 AN ALGORITHM FOR COMPLETE ACL ANALYSIS

Tries based algorithm uses tries to reduce number of analyzed pairs of ACL rules assuming that all rule fields are expressed as prefixes. Fields (IP addresses, ports) are usually expressed as ranges. It is not a problem since every range can be converted to a prefix [5]. Method is based on the idea that rules are in a conflict when there is partial or complete overlap in some of their corresponding fields.

The problem of finding conflicting rules is divided in k subproblems for k-dimensional rules. It builds k one-dimensional tries associated with each field of ACL rule. Assuming we have an ACL with M rules. In the trie T_l, for every dimension $l = 1, ..., k$, each node N associated with valid prefix contains two bit vectors of size M. First bit vector (*bitVector1*) has a bit i set if and only if there is a ACL rule R_i whose field l exactly match the node N prefix. The second bit vector (*bitVector2*) at a valid prefix node N has the following value $N.bitVector2 = (\bigcup C.bitVector2) \cup N.bitVector1$, where nodes C are all immediate descendents of N that also represent valid prefix. In other words the second bit vector computes the bitmap at node N corresponding to the union of the bit vectors associated with all valid prefix nodes in the subtrie rooted at node N. Node at trie T_l is a valid prefix node if and only if there exists at least one ACL rule R_i whose field l exactly match the node prefix.

After we have constructed tries for every dimension k of ACL rules it is possible to extract M-bit conflicts bit vector (CBV) for every ACL rule R_j. CBV for rule R_j has bit i set if and only if there is an ACL rule R_i in conflict with rule R_j. CBV extraction is done for all ACL rules in all their dimensions. For every field $l = 1, ..., k$ of rule R_j we begin extraction in root of the associated trie with CBV set to zero. Subsequently as we are moving from the root to node exactly matching rule field prefix, we union our CBV with *bitVector1* at every valid prefix node. When we reached node exactly matching rule R_j field l prefix we union CBV with the node *bitVector2*. This way we extract k CBVs and intersect them to get one final CBV for rule R_j. Finally the conflict between rules R_j and R_i is classified, using conflict classification algorithm, for every bit i in final CBV which is set to one.

3 IMPLEMENTATION AND RESULTS

We implemented tool based on previously described tries based algorithm for detecting conflicts among access control list rules. Conflict pairs are determined using conflicts bit vector and classified using stated conflict classification algorithm based on classification by Al-Shaer [1]. Our intention was to implement tool usable for checking ACLs with large number of rules. To save memory during ACL analysis the *Word-Aligned Hybrid* (WAH) bitmap compression scheme [6] have been used for bit vectors implementation. This scheme offers efficient in-memory representation scheme while allowing fast bitwise logical operations (AND, OR, ...). As tool input is used a text file containing a configuration of a network device with proper configuration file syntax. Tool output containing information about detected conflicts is in form of XML file in order to allow further processing if needed. Input and output have been implemented modularly with common interfaces therefore it is possible to extend number of supported vendors and output formats in the future if needed.

Used scheme have been tested and compared with naive algorithm, which analyze every possible pair of ACL rules, to proof reduction of analysed pairs of ACL rules. For testing we used randomly generated ACLs of sizes 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000 rules. Of each

size five different ACLs have been used and results have been averaged. Testing results are shown in Figure 1.

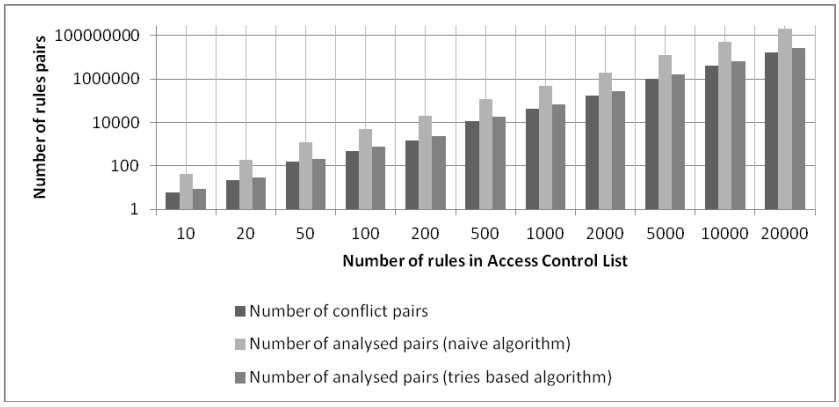

Figure 1: Results of testing naive algorithm and tries based algorithm.

4 CONCLUSIONS

In this paper we have shown possibility of using static analysis to find errors in network device configuration, especially conflicts among rules in ACL. We have implemented tool for configuration analysis of Cisco, HP and Juniper devices. Our intention was to create a tool usable for ACLs with large number of rules therefore it uses tries based algorithm. To save memory consumption during ACL analysis the *Word-Aligned Hybrid* bitmap compression scheme have been used. Implemented tool have been tested for reduction of analysed pairs of rules using randomly generated ACLs. It would be more satisfying having real devices configurations for testing.

REFERENCES

[1] E.S. Al-Shaer and H.H. Hamed. Firewall policy advisor for anomaly discovery and rule editing. In *Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on*, pages 17-30, march 2003.

[2] Zhe Chen, Shize Guo, and Rong Duan. Research on the anomaly discovering algorithm of the packet filtering rule sets. In *Pervasive Computing Signal Processing and Applications (PCSPA), 2010 First International Conference on*, pages 362-366, sept. 2010.

[3] S. Pozo, A.J. Varela-Vaca, and R.M. Gasca. A quadratic, complete, and minimal consistency diagnosis process for firewall acls. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 1037-1046, april 2010.

[4] F. Baboescu and G. Varghese. Fast and scalable conflict detection for packet classifiers. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pages 270-279, nov. 2002.

[5] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and scalable layer four switching. In *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '98, pages 191-202, New York, NY, USA, 1998. ACM.

[6] Kesheng Wu, Ekow J. Otoo, and Arie Shoshani. Optimizing bitmap indices with efficient compression. *ACM Trans. Database Syst.*, 31:1-38, March 2006.