Journal of Network and Computer Applications 113 (2018) 14–25

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A fine-grained rule partition algorithm in cloud data centers

Wei Jiang [a], Wanchun Jiang [a], Weiping Wang [a], Haodong Wang [b], Yi Pan [c], Jianxin Wang [a,*]

[a] School of Information Science and Engineering, Central South University, Changsha, 410083, China
[b] Department of Electrical Engineering and Computer Science, Cleveland State University, OH 44115, USA
[c] Department of Computer Science, Georgia State University, Atlanta, GA 30302-4110, USA

ARTICLE INFO

Keywords:
Rule partition
Fine-grained
Cloud

ABSTRACT

To better control the individual data flow in the cloud, traffic management policies are in need of increasing fine-grained rules, causing a dramatic increase in rules. The limited CPU or memory resources at the servers become the bottleneck when those policies are employed in large-scale data centers. To overcome these resource constraints, the rule partition algorithm is indispensable to decompose the rule set so that some rules can be migrated away. This paper shows that the current rule partition algorithm of vCRIB may lead to a high traffic overhead and high rule inflation in certain cases. Motivated by this observation, we propose a Fine-grained Rule Partition (FRP) algorithm. Different from vCRIB which divides the rule set based on the unit of rules within the same source address, FRP treats the individual rule as a unit, and thus reduce the traffic overhead by retaining more deny rules. In addition, FRP controls the rule inflation by optimizing the number of redirection rules as well. The simulation results confirm that FRP outperforms vCRIB, especially when the resources are severely constrained.

1. Introduction

In cloud data centers, for the better management of the data flow, network operators need to specify an increasing number of detailed policies. For instance, per-virtual machine (VM)-pair fine-grained access control rules are adopted by cloud providers to safeguard the network (Popa et al., 2010). The fine-grained bandwidth allocation policies and the rate limit policies require per-VM rules to guarantee a fair share of the bandwidth (Popa et al., 2012). This arrangement directly causes the rule explosion in large-scale data centers, quickly exhausting the CPU and memory resources.

Recent researches (Moshref et al., 2013; Kang et al., 2013; Zhang et al., 2014; Kanizo et al., 2013; Yu et al., 2010) indicate that network operators apply the rule partition algorithm to relieve the resource constraints. More specifically, the original rule set is divided into multiple smaller subsets. A part of subsets is migrated away from the original server to reduce the usage of the resources. In fact, each of the migratory subsets is replaced by a redirection rule[1] and then distributed to other devices.

It is obvious that the basic requirement of the rule partition algorithm is to guarantee the retained rules satisfy the resource constraints. Additionally, extra resources are required to store the increased rules, such as the created redirection rules to replace the migratory subsets and the replicated rules to maintain the correctness of policy (Moshref et al., 2013). Therefore, the partition algorithm is expected to minimize the increased rules. In other words, the partition algorithm has small rule inflation.

Another correlative performance indicator is traffic overhead, which refers to the unexpected traffic that is redirected due to the rule migration. We classify the rules as accept rules or deny rules. The traffic matching the accept rule is allowed to pass, while the traffic matching the deny rules is not. If a deny rule is migrated to a new device, the traffic matching this migratory deny rule first needs to travel to the new device and then be blocked. Obviously, the partition algorithm is expected to retain all deny rules at the original servers to minimize this unwanted traffic.

vCRIB (Moshref et al., 2013) is proposed to partition the rule set based on the source IP addresses of rules. A part of the rules is migrated

* Corresponding author.
E-mail addresses: jiangwei_csu@126.com (W. Jiang), jiangwc@mail.csu.edu.cn (W. Jiang), wpwang@mail.csu.edu.cn (W. Wang), hwang@eecs.csuohio.edu (H. Wang), yipan@gsu.edu (Y. Pan), jxwang@mail.csu.edu.cn (J. Wang).

[1] Rule partition algorithm creates the redirection rule to reduce the size of the original rule set by migrating the selected rule subsets. For more details, see Section 2.

https://doi.org/10.1016/j.jnca.2018.03.025
Received 8 July 2017; Received in revised form 5 March 2018; Accepted 22 March 2018
Available online 29 March 2018

away from the original servers to satisfy the resource constraints. Moreover, vCRIB decreases the rule inflation by replicating the rule that spans multiple source IP addresses, instead of splitting it into several small rules.

However, vCRIB (Moshref et al., 2013) suffers two disadvantages. First, the rule partition algorithm in vCRIB is not fine-grained enough. It's more desirable to retain the deny rules to curb the unwanted traffic. However, in vCRIB, in order to prevent a certain deny rule from being migrated, the accept rules of the same source IP address must be held. That means significant resources which have been used to accommodate more deny rules are consumed. Second, the partition algorithm of vCRIB may cause high rule inflation as it does not make the number of redirection rules optimize.

Motivated by the above observations, we propose a Fine-grained Rule Partition (FRP) algorithm to address the above issues. Instead of grouping rules by their source IP addresses as in vCRIB, FRP treats the individual rule as a unit. In this way, FRP retains as many deny rules as possible, together with a smaller traffic overhead. Moreover, with each redirection rule migrating as many accept rules as possible, FRP reduces the redirection rules and the rule inflation.

To evaluate FRP, we utilize two ACL rulesets included in the vCRIB code (Moshref, 2013), and also operate Classbench (Taylor and Turner, 2007) to generate two different rulesets as the representative of real-world access control. In addition, we validate the impacts of the different factors, such as resource constraints, the deny rule ratios, traffic localities and different types of rulesets. The simulation results confirm that FRP outperforms vCRIB in terms of the traffic overhead and the rule inflation ratio, especially when the resources are severely constrained.

This paper is organized as follows. In Section 2, we introduce the background and related work about rule partition algorithm. Our motivation is presented in Section 3. Section 4 shows the detail of FRP and Section 5 is the evaluation of FRP. We conclude the paper in Section 6.

2. Background and related work

2.1. Background

Fine-grained policies (Verma, 2002; Ferraiolo et al., 2003; Popa et al., 2010, 2012; Al-Fares et al., 2010) greatly improve cloud security, network utilization, application performance, and the fairness. However, a simple fine-grained policy may result in thousands or even millions of rules. For instance, there can be 200k rules per server for access control or fair bandwidth allocation in a data center with 100k servers and 20 VMs per server (Moshref et al., 2013). Unfortunately, the memory resources of the network interface cards, where servers' hypervisors offload rules, can merely store thousands of rules. In addition, the available servers' CPU budget for rule processing is also limited (Moshref et al., 2013). One solution proposed by vCRIB is using partition algorithm to offload a part of the rules from hypervisors to switches. Because the switches (e.g., TCAM (Rottenstreich and Tapolcai, 2017)) can also process the rules.

Fig. 1 shows a simple example of a rule partition algorithm. There are four VMs on server 1 (S1) and two VMs on server 2 (S2). The ACL rules are required to be placed on S1 to limit the unwanted network traffic generated by VM1, VM2, VM4 and VM5. Each rule consists of two parts: the action part and the predicate part. The former is to either accept or deny traffic. The latter includes a source IP address and a destination IP address. For instance, as shown in Fig. 2, the accept rule A2 with values (VM0 - VM5, VM1) represents a rule that accepts the traffic from any source IP of VM0 - VM5 to the destination IP of VM1. Besides, the rules are always assigned with priorities. When an incoming packet header matches more than one rule, the action of the rule with the highest priority is executed. If no matching is found, a default rule is performed (Ferraiolo et al., 2003).

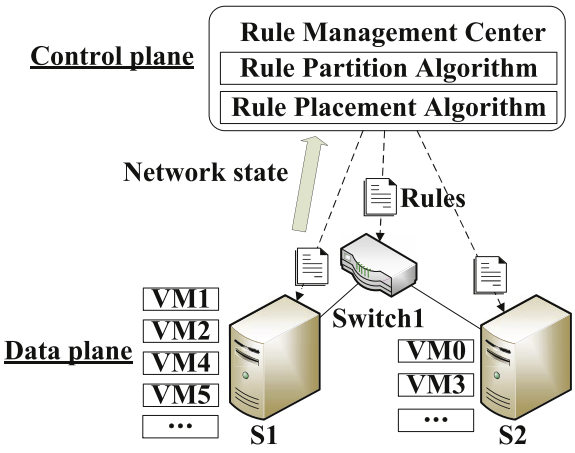

Fig. 1. An example of rule manage system.

The rule spanning multiple source IP addresses is called the multiple-span rule. For instance, A2 is a multiple-span rule which spans the source IP addresses from VM0 to VM5.

Accordingly, the rule occupying the same areas with another in the multi-dimensional space is identified as overlap. For instance, D3 overlaps with A5 as they have the common area (VM1, VM3). If all parts of one rule are within another rule in the multi-dimensional space, then this rule is inside the other rule. For instance, D4 with values (VM4, VM0) is inside the rule R' (VM4 - VM5, VM0 - VM5) as shown in Fig. 3.

Let us consider a scenario that S1 can only store eight rules and there are eleven ACL rules to be deployed on S1. The ACL rules consist of six accept rules, four deny rules, and a default rule D0 as shown in Fig. 2. In this condition, the memory constraint prohibits the deployment of all eleven ACL rules. Hence, the rule partition should be executed to reduce the size of the original rule set. According to the rule partition algorithm, a part of the rule set are selected to be migrated away so that the retained rules can be stored at S1. To maintain the same policy semantics, the migrated rules are replaced by one or multiple redirection rules.

The redirection rule is a special accept rule that all the packets falling in redirection rule can be forwarded without any modification

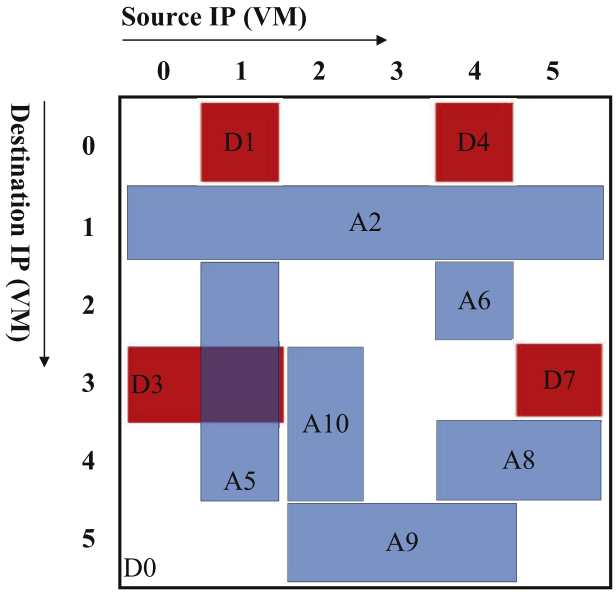

Fig. 2. The introduction of the rule set (A2, A5, A6, A8, A9 and A10 are accept rules. D1, D3, D4, and D7 are deny rules. D0 is default rule.)