Моделирование и анализ информационных систем. Т. 25, № 3 (2018), с. 251–256
Modeling and Analysis of Information Systems. Vol. 25, No 3 (2018), pp. 251–256

Программно-конфигурируемые сети

DOI: 10.18255/1818-1015-2018-3-251-256

UDC 004.415.25

A New Approach for Detecting and Resolving Anomalies in Security Policy of the External Firewall Module of the Floodlight SDN Controller

Morzhov S. V., Nikitinskiy M. A.[1]

Received December 26, 2017

Abstract. In this paper, the authors analyze the developed PreFirewall network application for the Floodlight *software defined network* (SDN) controller. This application filters rules, which are added into the firewall module of the Floodlight SDN controller in order to prevent the occurrence of anomalies among them. The rule filtering method is based on determining whether the addition of a new rule will not cause any anomalies with already added ones. If an anomaly was detected while adding the new rule, PreFirewall application should be able to resolve it and must report the detection of the anomaly.

The developed network application PreFirewall passed a number of tests. As a result of the stress testing, it was found that the time of adding a new rule, when using PreFirewall, substantially increases with increase in the number of previously processed rules. Analysis of the network application PreFirewall showed that while adding a rule (the most frequent operation), in the worst case it is necessary to compare it with all existing rules, which are stored as a two-dimensional array. Thus, the operation of adding a new rule is the most time-consuming and has the greatest impact on the performance of the network application, which leads to an increase in response time.

A possible way to of solving this problem is to select a data structure used to store the rules, in which the operation of adding a new rule would be simple. After analyzing the structure of the policy rules for the Floodlight SDN controller, the authors noted that a tree is the most adequate data structure for its storage. It provides optimization of memory used for storing the rules and, more important, it allows to achieve the constant complexity of the operation of adding a new rule and, consequently, solving the performance problem of the network application PreFirewall.

The article is published in the authors' wording.

Keywords: firewall, Floodlight, hash table, network controller, policy tree, PreFirewall, rules anomalies resolving, SDN, software-defined network

For citation: Morzhov S. V., Nikitinskiy M. A., "A New Approach for Detecting and Resolving Anomalies in Security Policy of the External Firewall Module of the Floodlight SDN Controller", *Modeling and Analysis of Information Systems*, **25**:3 (2018), 251–256.

On the authors:
Sergey V. Morzhov, orcid.org/0000-0001-6652-3574, graduate student,
P.G. Demidov Yaroslavl State University,
14 Sovetskaya str., Yaroslavl 150003, Russia, e-mail: smorzhov@gmail.com

Mikhail A. Nikitinskiy, orcid.org/0000-0001-8830-8613, system analyst, programmer,
A-Real Group, Energiya-Info Inc., 144 Soyuznaya str., Yaroslavl, 150008, Russia, e-mail: man@a-real.ru

Acknowledgments:
[1]The work was supported by RFBR, the research project № 16-07-01103a.

Introduction

The Prefirewall network application for the SDN FloodLight controller [1], proposed by the authors in the article [2], showed good results in filtering the added rules in policies rule sets. The developed algorithm prevents the occurrence of any anomalies among them. However, the load testing of the network application revealed a serious drawback in it – an increasing time the application spent to add a new rule when the number of processed rules is big enough.

To test the network application, 15 000 rules were created by the random rule generator. Such an amount of rules was proposed by one of the Russian manufacturer of UTM (*Unified threat management*) solutions, based on an analysis of the rule sets used in customer organizations with an average of 500 users. Testing was performed on a computer running Ubuntu 14.04 with Intel core i7 processor with a clock rate of 3 GHz.

Firstly, the time taken to load all generated rules directly into the external module of the Floodlight SDN controller was measured. After that, the same rules have been added to the Floodlight through PreFirewall network application, which resolves all the anomalies between them. Fig. 1 illustrates how the time taken to add each rule is growing with the growth of the number of processed and stored rules. On the *x-axis*, the scale values are from 0 to 15 000 and shows the number of installed rules. On the *y-axis*, the scale values are from 0 to 2.5 and show the rule installation time in seconds.

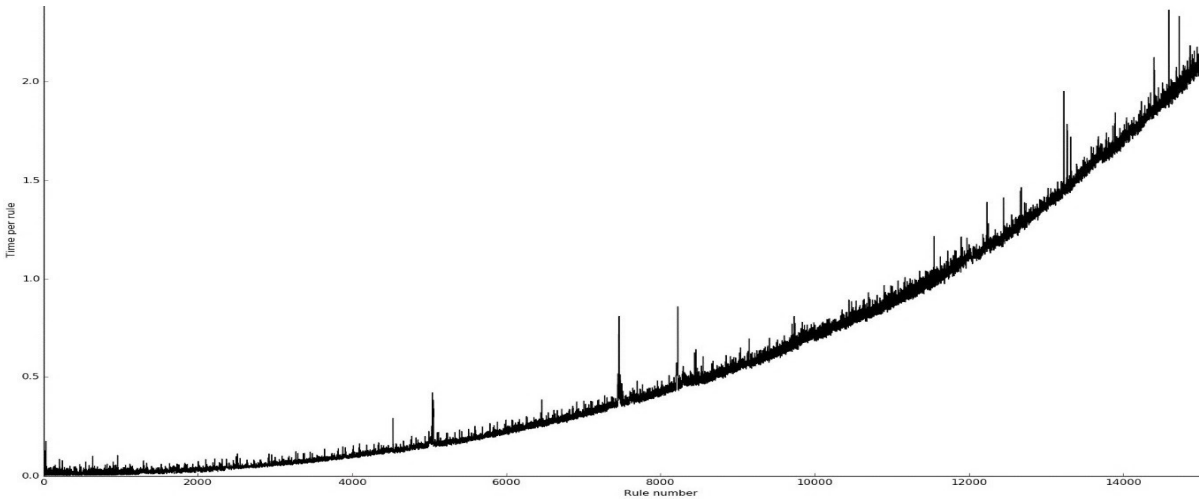

Fig 1. Adding rules to the Floodlight SDN controller through PreFirewall

The growth rate is polynomial due to the algorithm complexity [2]. Table 1 provides more detailed test results:

Taking into account the results obtained in Table 1 and analyzing possible ways of reducing the time the network application used to process a new rule, it was decided to optimize the algorithm used in PreFirewall.

Table 1. Test results

	Floodlight, s	PreFirewall + Floodlight, s
The average time of adding a new rule	0.0115	0.6333
Standard deviation	0.0073	0.6491
Total time taken to process 15 000 rules	172.055	9499.8394
The time taken to add the first rule	0,01	0,04
Rules were added (out of 15 000)	14781	11631

1. Analysis of data structures for a new algorithm

Analysis of the algorithm used in the network application PreFirewall showed that the main problem of low performance is the data structure used to store the rules. A two-dimensional array was used as the repository of rules. As a result of this, the operation of adding a new rule (one of the most frequent operations) became very slow. In the worst case, if the new rule is "good" (adding it does not cause anomalies), it should be compared with all stored rules. Thus, the addition of "good" rules is more complex than the addition of "bad" rules. Taking into account that "good" rules are added more often than "bad" rules, in order to optimize the algorithm, first of all, it is necessary to make the operation of adding "good" rules as simple as possible in terms of complexity. It is worth noting that, without introducing fundamental changes to the algorithm itself, you can achieve the task only if you store the processed rules in the form of a tree. The use of dictionaries, hash tables or databases will not yield any gain, since they are not fundamentally different from a two-dimensional array in their structure.

Table 2. Floodlight firewall rule fields

Filed name	Possible value	Description
switchid	xx:xx:xx:xx:xx:xx:xx:xx	Switch identification number
src-inport	short	Input switch port number
src-mac	xx:xx:xx:xx:xx:xx	Source MAC-address
dst-mac	xx:xx:xx:xx:xx:xx	Destination MAC-address MAC-адрес получателя
dl-type	ARP or IPv4	Protocol
src-ip	A.B.C.D/M	Source IP-address
dst-ip	A.B.C.D/M	Destination IP-address
nw-proto	TCP or UDP or ICMP	Protocol
tp-src	short	Source port number
tp-dst	short	Destination port number
priority	int	Priority of the rule (less is more important)
action	allow or deny	Allow or deny set of network flows which are match rule

So, let us take a closer look at how to introduce security policy rules in the form of a tree. Concerning OpenFlow Standard 1.0, the packet headers have 12 fields [3] (Table 2). Let us arrange the rule fields by the number of possible values that can be contained in them, in ascending order. The field priority and action will be placed at the end of this sequence, since these fields are related to the action of the rule. Thereby, the rules

tree will have a height equals to 12. We establish a one-to-one correspondence between the levels of the tree and the rules fields in such a way that the smaller the number of possible values of the field, the lower the tree level corresponding to it. Thus, we establish a one-to-one correspondence between each rule of security policy that does not contain any anomalies and tree path started at the root and ended in the leaf.

Let each vertex of the tree contain hash table (key-value pair). The key value will be the value of the rule field. The value is the address of the corresponding vertex on the level below. At level 12 of this tree, there will be leaves corresponding to action field and containing two possible values – allow or deny as keys, and the values will be nullptr. A simplified tree model is shown in Fig. 2. This tree will be called a police tree. Nodes marked with a dashed line are not included in the policy tree and are inserted into the scheme only for numbering the rules.

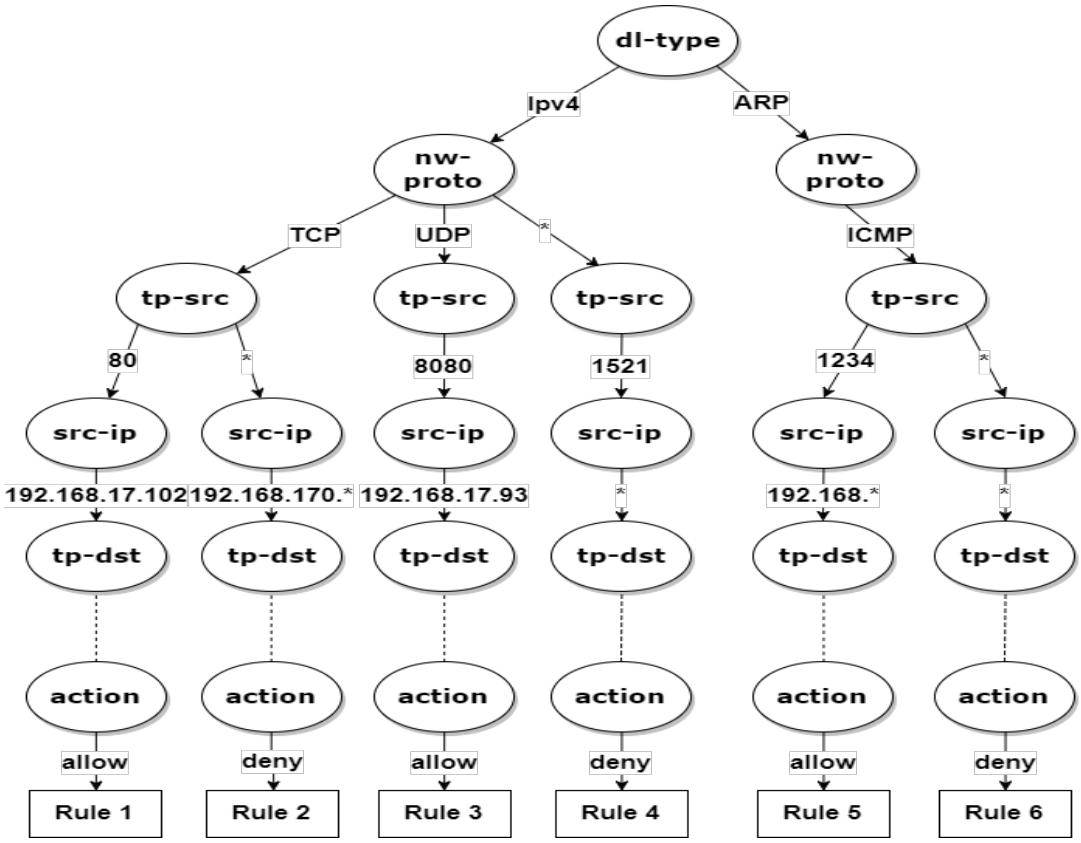

Fig 2. Policy tree example

2. Bulding police tree. Anomalies detection and resolution algorithm

Consider the algorithm for building this tree and finding anomalies among the security policy rules. The algorithm for adding a new rule to a tree is as follows:

- Discovery routine. The DiscoverAnomaly algorithm called. At this stage, it is determined whether the set of security policy rules will remain free of any anomalies,

after addition of the new rule. If so, go to stage 2, otherwise – the completion of the algorithm and showing the anomaly detection message.

- Rule insertion routine. The InsertRule algorithm called. Inserting the rule into the tree and completing the algorithm. Showing message about the insertion of the rule.

The DiscoveryRoutine algorithm is extremely simple. Each field of the new rule is compared with every field of the rules already added in the policy tree, in order to search for anomalies. Anomalies are sought on the basis of their definitions obtained earlier [2].

The decision routine is activated once all the rule fields have been checked and the action field is reached. At that point, the final anomaly state is determined and reported. If the rule action coincides with the action of another rule, an anomaly is discovered. Correlation and generalization are confirmed if the rule actions are different. If the input anomaly state is a generalization and the actions are the same, the existing rule is redundant to the new rule. Finally, if the new rule is a subset or equal to the existing rule, the new rule is redundant if their actions are the same, and is shadowed if their actions are different. If an anomaly is discovered and decided, the user is reported with the type of anomaly and the rules involved.

Thus, to determine the type of anomaly or to make sure that there are no any anomalies, it is required to check the presence of a specific value in hash table on each of the 12 levels of the policy tree. The complexity of this operation is $O(1)$ [4]. Hence, the complexity of the DiscoverAnomaly algorithm is also $O(1)$.

The InsertRule function inserts rules to the policy tree that have been tested by the DiscoverAnomaly function. The complexity of this operation is $O(1)$.

Thus, the proposed modification of the algorithm for determining anomalies have a constant time complexity.

3. Conclusion

Often, network administrators have to modify existing security policy rules, which can lead to anomalies or opening security holes. The detection of anomalies in this case becomes more difficult than detection anomalies while adding new rules, because changing the rule can potentially lead to serious changes in the policy tree as a whole. Effective restructuring of the tree, combining the rearrangement of the old rules, is a complex task. In the future, it is planned to conduct research on the development of effective algorithms for restructuring the policy tree in the case of editing existing rules, as well as carrying out stress testing of the proposed algorithm.

References

[1] *Floodlight SDN OpenFlow Controller*, https://github.com/floodlight/floodlight.

[2] Morzhov S., Alekseev I., Nikitinskiy M., "Firewall application for Floodlight SDN controller", International Siberian Conference on Control and Communications (SIBCON) (12–14 May, 2016, Moscow, Russia), 1–5.

[3] *OpenFlow Switch Specification, Version 1.4.0*, http://networkstatic.net/wp-content/uploads/2013/10/openflow-spec-v1.4.0.pdf.

[4] Cormen T., Leiserson C., Rivest R., Stein C., *Introduction to Algorithms*, 3rd., Massachusetts Institute of Technology Press, 2009, ISBN: 978-0-262-03384-8.

Моржов С. В., Никитинский М. А., "Новый подход к обнаружению и устранению аномалий в политике безопасности внешнего модуля межсетевого экрана контроллера ПКС Floodlight", *Моделирование и анализ информационных систем*, **25**:3 (2018), 251–256.

DOI: 10.18255/1818-1015-2018-3-251-256

Аннотация. Авторы рассматривают разработанное сетевое приложение PreFirewall для контроллера *программно-конфигурируемой сети* (ПКС) Floodlight. Данное сетевое приложение осуществляет фильтрацию устанавливаемых правил в модуль ядра firewall контроллера Floodlight с целью недопущения возникновения аномалий между устанавливаемыми правилами. Разработанное сетевое приложение PreFirewall прошло ряд тестов. В результате проведенного нагрузочного тестирования было установлено, что время добавления новых правил, при использовании PreFirewall, серьезно возрастает с ростом количества ранее обработанных правил. Анализ сетевого приложения PreFirewall показал, что при добавлении правила (самая частая операция), в худшем случае необходимо произвести его сравнение со всеми существующими правилами, которые хранятся в виде двумерного массива. Таким образом, операция добавления нового правила является наиболее трудоемкой и сильнее всего влияет на производительность сетевого приложения, что приводит к возрастанию времени его отклика. Одним из возможных путей решения данной проблемы является выбор такой структуры данных, используемой для хранения правил, в которой операция добавления нового правила была бы простой. В качестве такой структуры предлагается использование дерева, каждая вершина которого содержит всевозможные значения полей в устанавливаемых правилах. Данный подход обеспечивает константную сложность операции добавления нового правила и, следовательно, решает проблему производительности сетевого приложения PreFirewall. Статья публикуется в авторской редакции.

Ключевые слова: межсетевой экран, Floodlight, хеш-таблица, сетевой контроллер, дерево правил, PreFirewall, разрешение возникающих аномалий, ПКС, программно-конфигурируемая сеть

Об авторах:
Моржов Сергей Владимирович, orcid.org/0000-0001-6652-3574, магистрант,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150003 Россия, e-mail: smorzhov@gmail.com

Никитинский Михаил Александрович, orcid.org/0000-0001-8830-8613, программист-аналитик,
ООО «Энергия-Инфо», ул. Союзная, 144, г. Ярославль, 150008 Россия, e-mail: man@a-real.ru

Благодарности:
[1]Работа выполнена при финансовой поддержке РФФИ, проект № 16-07-01103а.