Net Auto-Solver: A formal approach for automatic resolution of OpenFlow anomalies

Ramtin Aryan*†, Anis Yazidi†, Adel Bouhoula‡ and Paal Einar Engelstad*†
*Department of Technology Systems, University of Oslo, Oslo, Norway
†Department of Computer Science, OsloMet – Oslo Metropolitan University, Oslo, Norway
‡ College of Graduate Studies, Arabian Gulf University, P.O. Box 26671, Kingdom of Bahrain
Email: ramtina@ifi.uio.no, anisy@oslomet.no, a.bouhoula@agu.edu.bh, paal.engelstad@its.uio.no

***Abstract*—Policy anomalies are frequent in nowadays's computer networks due to their increasing configuration complexity. Resolving policy anomalies usually requires network administrator intervention, which is a time-intensive and error-prone process. In this paper, we present *Net Auto-Solver*, a formal approach for *automatic* resolution of OpenFlow anomalies. The approach resorts to the concept of high-level policies to not only detect policy violations but also correct them on-the-fly. Our approach is fully automated and does not require interaction with the network administrator. Although there is a multitude of research works on detecting anomalies in SDN, research to correct those anomalies in an automatic manner is extremely scarce. At the heart of our approach, we propose two inference systems to perform corrective actions to the policy. We provide some experimental results involving real-life network configurations to show the performance of our approach. The first results are very promising.**

***Index Terms*—OpenFLow Anomaly, Auto-correct, Resolving Anomaly, Conflicts, Violation, SDN.**

I. Introduction

SDN anomalies were shown in the literature to be responsible for various network policy violations such data breaches, segmentation violations, and complex loops. It is not practical neither efficient to delegate the correction task to network administrators in SDN networks since human-based correction is a time-intensive process that is prone to error.

There is a handful number of works that try to automatically correct the anomalies and not only detect them. Yazidi and Bouhoula [1] suggested a semi-automated method for solving part of anomalies between firewall rules where for some anomalies automatic correction actions are performed without human interaction while for others a feedback from the network administrator is needed. Later Yazidi and Bouhoula [2] extend the latter work to a fully automated correction method for firewall anomalies by presenting the so-called Firewall Policy Query Engine (FPQE). However, the latter work is limited to non-distributed firewalls.

In addition, Zhou et al. [3] proposed an auto-correct tool, called NEAt to resolve policy conflicts before they occur as a side effect of a newly installed rule. NEAt uses a formalism for expressing high-level policies. Then NEAt tries to map and compare the network rule and configuration with the high-level policy model. As a correction action, NEAt tries to remove the policies leading the anomalies or loops. However, the proposed method for modeling the policies and conflict solving is complex and difficult for the network administrators to understand.

This paper aims to propose an automatic correction method to detect and solve the conflicts between policies. To this end, we will start first by categorizing the anomalies in data layer that cause segmentation violations or loops. Then via formal methods, we provide an inference system for resolving each anomaly category. Resolving the anomalies may cause some side effects and policy violations. So the resolution actions must be validated via the high-level policies. High-level policies describe the main constraints that the rules in the data layer should follow it, and it is assumed to be conflict-free. The conflict-freedom property can be guaranteed by the same methods as PGA [4]. By virtue of using inference systems, our method is simple and easy to comprehend by network administrators. Furthermore, we show that we can also prove its correctness.

The reminder of the paper is organized as follows. In Section II, we provide an overview over the state-of-the-art on detecting and resolving anomalies in SDN networks. Section III describes the root causes of anomalies and the correction method is discussed in Section IV. Finally, Section V we provide an implementation of our approach that demonstrates that it is applicable and check its performance.

II. Related Work

Anomaly detection and correction are of utmost importance for the network administrators. There is not many research that has focused on this problem in the context of SDN. In this section, we review some of the main research in this area.

Zhou et al. [3] present an auto-correction method that is called NEAt. NEAt controls the new rule to be updated or installed for the aim to detect eventual conflicts with the high-level policy. NEAt operates using two different correction modes: pass-through and interactive. In the pass-through mode, NEAt detects and corrects the conflict automatically. Depending on the configured mode, NEAt either suggests a solution or solves the conflict automatically on-the-fly and installs the corrected rule on the switch.

Yazidi and Bouhoula [1] present a method for correcting the anomalies within firewall rules. They use the anomaly classification defined by [5].

Kheradmand [6] presents a method that is called Anime which suggests an alternative route for bridging the forwarding gap. It introduces objective measures to apply the data mining method to the forwarding policies.

Fan et al. [7] introduce an algorithm that is called NSOA. The algorithm aims to detect an optimum alternative route after a failure occurs on a switch. NSOA tries to keep most of the network configuration unchanged and updates the affected switches.

Saâdaoui et al. [8] presented a firewall anomaly resolution tool that is called FARE. The proposed method aims to detect and correct the anomalies in single firewalls. Moreover, a new classification of anomalies across multi-firewalls is introduced.

III. Anomaly Definition

Anomalies in this paper refer to conflicts that lead to the fact that a set of packets do not follow their expected route and ending into an unintended destination. These anomalies may cause loops or other types of anomalies such as network slicing violations. In this section, we use the specific notations for the switches and rules that play an important role in the anomaly categorization and repairing process.

A. Overlap Anomaly

According to the high-level policy, the *overlap anomaly* occurs if a specific set of packet headers has the access permission to both expected and unwanted nodes. Moreover, the faulty switch has permission for forwarding the specific packet header. Equation 1 presents the formal expression of the condition of the overlap anomaly. The anomaly takes place if a set of packets (Ph_List) does not follow the expected path. Moreover, according to the high-level policies, the set of packets can reach the expected switch (S^{expt}) and unwanted switch (S^{unwt}) and the set of packets can be forwarded from the faulty switch (S^f).

$$\begin{aligned} overlapanomaly = \quad & Reach(Ph\_List, S^{unwt}) \wedge \\ & Reach(Ph\_List, S^{expt}) \wedge \\ & Forward(Ph\_List, S^f) \end{aligned} \tag{1}$$

B. Misordered Anomaly

According to the high-level policy, the *misordered anomaly* occurs if a specific set of packet headers has the access permission to the expected node while the faulty switch has permission for forwarding the specific packet header. Equation 2 presents the formal expression of the misordered anomaly condition. It shows that if a set of packets (Ph_List) does not follow the expected path, which the S^f, S^{unwt} and S^{expt} are true, and the set of packets can reach the S^{expt} and cannot reach the S^{unwt}. Moreover, the set of packet can be forwarded from the S^f. Then the *misordered anomaly* happens. We call this type of anomaly misordered since there is no conflict with the high-level policy. However, the corresponding rules have the wrong priorities.

$$\begin{aligned} misorderedanomaly = (!\, Reach(Ph\_List, S^{unwt})) \wedge \\ Reach(Ph\_List, S^{expt}) \wedge Forward(Ph\_List, S^f) \end{aligned} \tag{2}$$

IV. Repairing the Anomalies

This section, discusses four inference systems for repairing the three types of anomaly.

A. Adding a New Rule

This solution is designed for repairing a group of rules with overlap anomaly. For solving this type of anomaly, a new rule ($r_k^{sw^f}$) is added to the flow table of the faulty switch (sw^f). This rule should have a priority higher than the unwanted rule ($r_j^{sw^f}$) and the target rule ($r_i^{sw^f}$). The condition field of the new rule contains the overlap of the condition of the unwanted rule and condition of target rule ($C_k^{sw^f} = (C_i^{sw^f} \cap C_j^{sw^f}$). However, the union between the action of the unwanted rule and the target rule creates the action field of the new rule ($a_k^{sw^f} = (a_i^{sw^f} \cup a_j^{sw^f}$). The Fig. 1 presents a formal expression of the adding a new rule.

Inference System for Adding a New Rule:

$$Init: \frac{}{(C, R)} \quad where \begin{cases} C &= \text{The list of all distinct pair of rules in } R \\ R &= \text{The set of rules of the switch } sw \end{cases}$$

$$if \begin{cases} c_i \cap c_j \neq \emptyset \\ P \models Reach\,(c_i, sw^{exp}) \wedge Reach\,(c_j, sw^{unwt}) \wedge Forward(c_i, sw^f) \end{cases}$$

$$Add: \frac{((r_i, r_j) :: C, \{r_i, r_j\} \cup R)}{(C', R \cup \{r_k, r_i', r_j'\})} \quad where \begin{cases} r_i = (i, c_i, a_i) \\ r_j = (j, c_j, a_j) \\ r_i = (i, c_i/(c_i \cap c_j), a_i) \\ r_j = (j, c_j/(c_i \cap c_j), a_j) \\ C' = modify\left(C, \left[\left(r_i, r_i'\right), \left(r_j, r_j'\right)\right]\right) \\ r_k = (min\,(i, j) - 1, (c_i \cap c_j), (a_i \cup a_j)) \end{cases}$$

$$Skip: \frac{((r_i, r_j) :: C, R,)}{(C, R,)} \quad if\ no\ other\ rule\ apply\ to\ the\ (r_i, r_j)$$

$$Success: \frac{(\emptyset, R,)}{R}$$

Fig. 1

The inference system is initiated by the list of candidates C and the set of rules R that contains all rules in the network. Then the condition of the *Add* inference rule is checked for the each candidate c. The condition will be true, if there is an overlap anomaly in the pair of candidate rules. Without loss of generality, we refer to pair of candidate rules as candidate. the Then the new rule will be created and added to the rule list and the target rule r_i and the unwanted rule r_j will be modified.

If the candidate does not match with inference rule Add, the condition of the *Skip* inference rule will be checked. Skip inference rule is applied for the candidate that has no anomalies. This inference rule will remove the candidate from the list. So, this step demonstrates that the length of the candidate list will be decreased and system in finite system. When the list of candidates is empty the system will be terminated successfully.

B. Swapping Rules

This solution is useful for repairing group of rules with misordered anomaly. In this method, the priority of the unwanted rule (r_j) and the target rule (r_i) are swapped. This solution can guarantee the criteria of the high-level policy. The formal expression of the swapping solution is presented in the Fig. 2.

Inference System for Swapping Rules:

$$Init: \frac{}{(C,R)} \quad where \begin{cases} C &= \text{The list of all distinct pair of rules in } R \\ R &= \text{The set of rules of the switch } sw \end{cases}$$

$$if \begin{cases} c_i \cap c_j \neq \emptyset \\ P \models Reach\,(c_i, sw^{exp}) \wedge !\,Reach\,(c_j, sw^{exp}) \wedge Forward(c_i, sw^{f}) \end{cases}$$

$$Add: \frac{((r_i,r_j) :: C, \{r_i,r_j\} \cup R,)}{(C', R \cup \{r_i', r_j'\})} \quad where \begin{cases} r_i = (i, c_i, a_i) \\ r_j = (j, c_j, a_j) \\ r_i' = (j, c_i, a_i) \\ r_j' = (i, c_j, a_j) \\ C' = modify\left(C, \left[\left(r_i, r_i'\right), \left(r_j, r_j'\right)\right]\right) \end{cases}$$

$$Skip: \frac{((r_i,r_j) :: C, R)}{(C,R)} \quad if\ no\ other\ rule\ apply\ to\ the\ (r_i,r_j)$$

$$Success: \frac{(\emptyset, R)}{R}$$

Fig. 2

The inference system is initiated by list of candidates C, which are pairs of rules, and set of rules of all switches in the network. If the condition of the *Add* rule is correct for a candidate c, then it means there is misordered anomaly in the pair of rules. So in this step, the two rules are swapped in the rule table and the candidate is removed from C. If there is no anomaly in the pair of rules, the *Skip* rule is called. Skip rule removes the pair from the C and keep the rule table without change. At the end, if the list is empty it means there is no misordered anomalies among the rules and the system terminated successfully via the *Success* rule. As explained, length of the candidate list will be decreased and it means that the system is finite.

V. Evaluation

In this section, we evaluate the practical value of the proposed method by implementing the inference systems and testing them. The implementation uses C++ and for simulating the real-life scenario we use a topology based on a real ISP configuration from Rocketfuel dataset [9]. The test topology contains 25 switches, and each switch has 500 rules. Five different scenarios are designed for assessment and each scenario has 1, 5, 10, 15, and 20 distinct faulty rules (target rules), respectively.

A. Evaluation Metrics

We consider the processing time of creating a new rule, swapping rules, and finding the alternative route as metrics for assessing the performance of the implemented inference systems. Moreover, since the overlap and misordered anomalies take place on a single switch, there is a possibility to apply a multi-thread approach for correcting the anomalies of separate switches. Therefore, the processing time of single-thread approach is compared with the parallel approach as an evaluation metric.

B. Evaluation Results

In this section, the effect of rule number and network size on different inference systems is presented. The performance evaluation of each inference system is discussed.

1) Adding a new rule: As mentioned in Section III, if two rules have an overlap anomaly then the system should calculate the overlap section. As a next step, it creates a new rule based on the intersect part and inserts it to the flow table with a higher priority. The processing time of adding multiple rules are evaluated for both single-thread and multi-thread approaches. The result is sketched in Fig. 3.

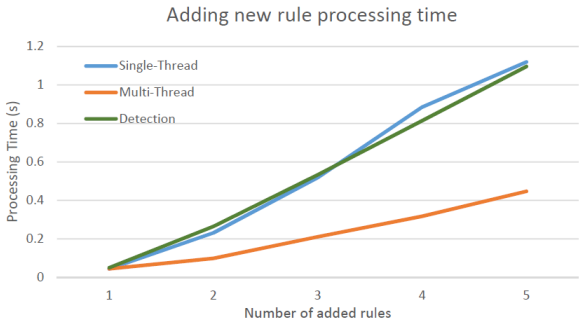

Fig. 3: Adding a new rule inference system processing time

According to Fig. 3, the processing time for single-thread approach is longer that the multi-thread approach and has almost linear behavior. However, since there is some overhead process in multi-thread method, its processing time does not follow the linear pattern.

2) Swapping two rules: The swapping action is applied if a pair of rules with a misordered anomaly. In this section, the performance of the swapping of multiple rules is evaluated for both single-thread and multi-thread approaches. The result of the assessment is presented as Fig. 4.

According to Fig. 4, the swapping process is not time-intensive. For this reason, there is not a significant difference between the single-thread and multi-thread approaches.

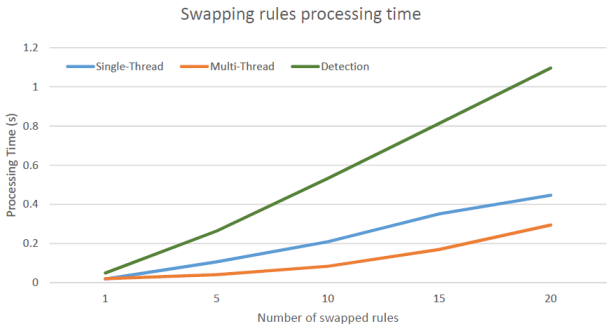

Fig. 4: Swapping rules inference system processing time

VI. Conclusion

There is a significant number of works for detecting policy conflicts in firewall rules and recently in forwarding devices in SDN. However, there is a handful number of works aspiring to correct those anomalies without human intervention.

This paper proposes the *Net Auto-Solver* method to resolve policy conflicts automatically without the need of network administrator intervention, in contrast to classical approaches that depend on prompting the network administrator for choosing corrective actions. The Net Auto-Solver uses a formal expression for modeling the high-level policy. Furthermore, via defining inference systems, our approach detects and corrects the anomalies automatically by resorting to the high-level policies. Notably, inference systems possess the advantage of being easy to comprehend by network administrators. To evaluate the performance of the Net Auto-Solver, we design a real-life scenario. Both a single-thread and multi-thread implementation of Net Auto-Solver are assessed, and the results are promising.

As future work, we would like to apply an incremental approach [10] to improve the performance of the correction process in Net Auto-Solver. Since the data plane rules are updated frequently, the correction performance is an essential factor in the efficiency of the method and thus the need for an incremental approach.

References

[1] A. Yazidi and A. Bouhoula, "On assisted packet filter conflicts resolution: An iterative relaxed approach," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, 2016, pp. 35–42.

[2] A. Bouhoula and A. Yazidi, "A security policy query engine for fully automated resolution of anomalies in firewall configurations," in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2016, pp. 76–80.

[3] W. Zhou, J. Croft, B. Liu, E. Ang, and M. Caesar, "Automatically correcting networks with neat," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 595–608.

[4] C. Prakash, J. Lee, Y. Turner, J.-M. Kang, A. Akella, S. Banerjee, C. Clark, Y. Ma, P. Sharma, and Y. Zhang, "Pga: Using graphs to express and automatically reconcile network policies," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 29–42, 2015.

[5] 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*. IEEE, 2003, pp. 17–30.

[6] A. Kheradmand, "Automatic inference of high-level network intents by mining forwarding patterns," in *Proceedings of the Symposium on SDN Research*, 2020, pp. 27–33.

[7] Z. Fan, H. Wu, J. Xu, and Y. Tang, "An optimization algorithm for spatial information network self-healing based on software defined network," in *2017 12th International Conference on Computer Science and Education (ICCSE)*. IEEE, 2017, pp. 369–374.

[8] A. Saâdaoui, N. B. Y. B. Souayeh, and A. Bouhoula, "Fare: Fdd-based firewall anomalies resolution tool," *Journal of computational science*, vol. 23, pp. 181–191, 2017.

[9] "Rocketfuel:," http://research.cs.washington.edu/networking/rocketfuel//, accessed: 2019-11-01.

[10] R. Aryan, A. Yazidi, and P. E. Engelstad, "An incremental approach for swift openflow anomaly detection," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 2018, pp. 502–510.