Journal of Computing and Security

Research Article, 2023, Volume 10, Number 1 (pp. 75–84)

http://www.jcomsec.org

Distributed Rule Anomaly Detection in SDN-based IoT: Towards a Comprehensive Approach

Rasool Kiani [a], Ali Bohlooli [a,*]

[a] *Faculty of Computer Engineering, University of Isfahan, Iran*

ARTICLE INFO.

Article history:
Received: 29 November 2022
Revised: 25 June 2023
Accepted: 17 Junly 2023
Published Online: 22 August 2023

Keywords:
Internet of Things (IoT), Software-Defined Networks (SDN), Anomaly, Rule.

ABSTRACT

The rapid increase in the number of equipment connected to different networks in the world has led to the development of diverse and new applications in the Internet of Things, which often use the current network infrastructure. In other words, force the network administrator to implement complex network policies manually. Due to this significant growth of equipment and the increase in the complexity of traditional network configuration, software-defined networks (SDN) integrate and facilitate network management by separating the control and data layers from each other and creating network rules in the data layer. For these facilities, these networks appear to be a good infrastructure for IoT networks, which will enable network programming to develop new and more efficient services to meet real needs. In addition, the variety of IoT equipment can increase complex and inconsistent network rules in SDN-based switches, making network management difficult. Accordingly, in this paper, we will try to model the behavior of anomaly rules distributed in software-defined networks such as FTD, FBF, and irrelevant anomalies that have been created by different apps in the Internet of Things. It can identify their relationship with other rules in the network and avoid registering them.

1 Introduction

The Internet of Things has been able to have a wide range of applications in areas such as medicine, economics, sports, etc., due to the managed communication established between communication equipment and devices [1]. Furthermore, the increasing growth of IoT-related equipment, devices, and various manufacturers and suppliers have complicated the proper configuration and made it difficult to maintain these networks [2]. Under such circumstances, and because of the benefits that software-defined networks offer, researchers seek to use them to configure devices used in the Internet of Things and build the infrastructure of IoT networks based on software-defined networks [3]. Software-defined networks, separate control and data layers and try to control the network centrally [4]. In these networks, the network switches that will be placed in the data layer are only responsible for sending network packets. The packets have been sent according to the tables that are completed by the network controller [5]. When a new packet is received from a node on the network in one of the software-defined switches, the packet information is sent to the network controller in the control layer [6]. It generates flow rules according to the policies and algorithms that the network administrator has already

* Corresponding author.
Email addresses: r.kiani@eng.ui.ac.ir, bohlooli@eng.ui.ac.ir
https://dx.doi.org/10.22108/JCS.2023.135769.1114
ISSN: 2322-4460

added to it [7]. These rules are sent to the data layer switches to make the appropriate decision and determine the switch's response to the received packet [8]. Software-defined networks have made it more accessible to configure network equipment by separating control and data layers, using high-level algorithms or technologies such as artificial intelligence [9]. When the Internet of Things platform is provided using software-defined networks, you can take advantage of this and experience centralized and easier management despite the various objects and programs in this network [10].

On the other hand, in software-defined networks, when the variety of programs and policies increase in the network, they will submit rules that may contradict each other resulting in a phenomenon called anomaly in flow tables [11]. These inconsistencies, which can be cases such as conflict, repetition, etc., lead to issues like reduced network quality or violation of network policies [12]. For this reason, any rules defined in the software networks when registering in the network switches must be checked for compatibility with other existing rules (absence of anomalies) to be allowed to spread at the network level.

Despite the various research that has been done in this field, discussion about detecting distributed anomalies in the whole network is a topic that has been hidden from vision. In this article, we will try to catch them by presenting a method. The major contributions of the paper can be summarized as follows:

- Mathematical modeling of rules anomalies to explain and detect more accurately,
- Presenting an algorithm to detect anomalies in a centralized flow table,
- Providing a solution for the detection of distributed rule anomalies in software-defined networks,
- Implementation of proposed algorithms and review of the obtained results.

Therefore, in the second part of this paper, we will describe the work done in the field of detecting anomalies within flow tables. The third section is devoted to the mathematical expression of the types of anomalies within a flow table and in a distributed form too. The fourth section presents two algorithms for detecting distributed and centralized anomalies. Finally, in the sixth section, the evaluation and application of the mentioned method in the practical environment will be shown.

2 Related Works

The problem of detecting anomalies in software-defined networks is an important and sometimes complex issue. It has been raised since the middle of the introduction of these networks and at the same time with the beginning of their use in practical environments. This issue became quite apparent when it was suggested that they were used in Internet of Things (IoT) platforms. For this reason, various researches have been done on the detection of anomalies in the flow tables of software-defined networks over the years;

FortNOX provides a method for authentication of various components in software-defined networks based on the NOX controller, and at the same time, prioritizes the different parts that were considered a part of the Internet of Things (IoT) [13]. The solution offered in FortNOX is a centralized solution so that the rules in FortNOX are protected [13]. When there is a conflict between the existing rules written by the network administrator in the FortNOX system and the new rules detected using an algorithm called Alias Set Rule Reduction if the inconsistent rule comes from outside FortNOX, it will be rejected, and if it is inside the FortNOX system, by looking at the priority of the current rule maker and the priority of the owner of the new rule, a rule that has a higher priority than another one will be registered [13].

VeriFlow sits between network controllers and switches and uses the network's graph to receive the rule's effects before registering the rule on the switches and creates a rule that is consistent with network policies in terms of security and privacy [14]. Although VeriFlow's primary goal is to process conflicting rules in real time, other anomalies are not considered in this study [14].

Tsogbaatar et al. propose a novel ensemble learning model for IoT anomaly detection using software-defined networks (SDN) [15]. They use a deep auto-encoder to extract handy features for stacking into an ensemble learning model [15]. The learned model is deployed in the SDN controller to detect anomalies or dynamic attacks in IoT by addressing the class imbalance problem [15].

AuYoung et al. have developed a method for detecting only conflicts by voting between different apps (application layer programs) in software-defined networks [16]. In this method, various apps vote to register or delete the inconsistent rule [16]. One of the most critical features of this method is the existence of two parameters called precision and parity for the effect of each application's vote on the fate of the rule, which can be different for each element in each period

and in relation to each packet [16]. Based on these two variables, voting is taken on the desired rule; if it reaches the quorum, it votes to register it [16].

FlowChecker is a mechanism for detecting coherence between switches and network controllers [17]. It determines the correctness of rules according to the applied protocol and resolves security problems of software-defined networks [17]. This system consists of two parts, Flow Visor and Flow Checker, in which Flow Visor is responsible for taking commands and monitoring them to send to Flow Checker. It can detect the coherence of rules between the data layer and the controller policies based on the mechanisms implemented in it [17].

The tree structure is one of the items used in the most priority-oriented problems and can implement network priorities by using its hierarchical form [18]. HFT is a method based on modeling network components in the form of a tree, which will increase the relationship between the rules by moving to the leaves of the tree [18]. When the algorithm realizes a conflict between the rules of two siblings, the strategy that the network administrator has considered to resolve it will be used [18]. In the continuing of this research, PANE corrects the detection of inconsistencies in the tree by providing a method that leads to the sharing of the tree throughout the network [18].

Anomalies in flow tables are not limited to the type of conflict and can be in other types as well. In addition to identifying conflict anomalies, Wang et al. Have attempted to model other classes of anomalies, such as redundancy or shadowing, and have used this model to identify and correct these anomalies [12].

In [19], Zhang et al. constructed the relationship tree between the rules in the flow tables. If a rule is added, deleted, or changed, the tree is searched for possible inconsistencies. In this way, in addition to saving time for all the rules in the network at the time of registering the rules, it also consumes some memory to build the relationship tree, which can also be another point of discussion.

As mentioned, there are many methods for detecting anomalies in software-defined networks that have entirely focused on detecting anomalies within a flow table and have somehow abandoned the detection of distributed anomalies within a network. In [20] preliminarily, distributed anomalies in software-defined networks were detected. An issue that this article completes and identifies distributed anomalies more comprehensively and quickly.

In [21], a technique called RuleOut is introduced to prevent SDN forwarding anomalies without using probe packets or packet traces. The approach adds unique tags to matching fields of dependent rules to ensure that a packet matches at most one rule on a switch. Rule disambiguation is implemented through source routing with efficient algorithms and optimization techniques. However, the authors do not mention how other inconsistencies are detected.

In this paper, we propose an algorithm to detect centralized and distributed rule anomalies for a given set of rules. First, try to model the types of centralized and distributed anomalies throughout the network and then use these models, to detect inconsistencies in the centralized (a flow table) and distributed (in several network switches) environment.

3 Modeling Anomalies in Flow Tables

In this section, the types of anomalies and their mathematical modeling will be introduced. In Table 1 the list of used symbols is inserted.

Table 1. List of Used Symbols.

Symbol	Description
R_x	Rule number x
m	The matching pattern of a rule
a	The action of a rule
p	Priority of a rule
$M(R_x)$	Function for getting the matching pattern of rule R_x
$A(R_x)$	Function for getting action of the rule R_x
$P(R_x)$	Function for getting priorities of rule R_x

Among the anomalies introduced, the first three types are specific to centralized flow tables, and the last two types are dedicated to anomalies distributed over the entire network.

It should be noted that in the following relations, R_x means a rule of number x, m represents the matching field, a represents the predicted operation (action), and p represents the priority of each rule, which can be defined as follows:

$$R_x = \{m, a, p\} \tag{1}$$

$$m = M(R_x), \quad a = A(R_x), \quad p = P(R_x) \tag{2}$$

3.1 Redundancy

In this type of anomaly, all the rules' components are somehow equal to each other [12]. It should be noted that equality in the match pattern means that the entire interval covered by one rule is covered by another:

$$\exists i, j : M(R_i) = M(R_j), P(R_i) = P(R_j), A(R_i) = A(R_j) \quad (3)$$

3.2 Correlation

A correlation anomaly is formed between two rules whose matching pattern is not the same but covers parts of each other [12]. Furthermore, the actions provided for each rule are different and, in some cases, contradict each other. A notable example of this anomaly is called conflict:

$$\exists i, j : M(R_i) \subset M(R_j), A(R_i) \neq A(R_j) \quad (4)$$

3.3 Generalization

In this type of anomaly, the rules are recorded in such a way that the matching patterns of one of them completely cover the other, and the predicted actions are the same for both [12]:

$$\exists i, j : M(R_i) \subset M(R_j), R_x(P) > R_y(P), A(R_i) = A(R_j) \quad (5)$$

3.4 FBF

Assume that the addition operator means the effect of all operations predicted for existing rules [22]. If the sum of all these operations is equal to the last operation, then FBF anomaly will occur:

$$\exists i_1, \ldots, i_n : \{A(R_{i_1}) + \cdots + A(R_{i_n})\} \equiv A(R_{i_n}) \quad (6)$$

3.5 FTD

This anomaly is formed if the effects of a series of rules on a packet lead to dropping it [22]:

$$\exists i_1, \ldots, i_n : \{A(R_{i_1}) + \cdots + A(R_{i_n})\} \equiv \text{DROP} \quad (7)$$

3.6 Irrelevant

An irrelevant anomaly will occur if a rule is sent for registration in a particular switch while no traffic based on it reaches the target switch. In other words, if i and j are related to each other [22]:

$$\nexists i : A(R_i) \equiv R_j \quad (8)$$

4 Proposed Method

To solve the problem studied in this paper and identify the anomalies in the flow tables, the anomalies are divided into three parts. They are described separately:

- Centralized within a table.
- Distributed throughout the network.

Algorithm. 1 shows the pseudo-code of the proposed algorithm for detecting centralized anomalies within a flow table. At first, the algorithm tries to separate the variables in the matching pattern and then uses these parameters to obtain the relationship between them. After that, the type of possible anomalies is identified based on the cases extracted from the new rule and the relationships described in the previous section.

In the second part, the proposed method identifies distributed anomalies in the software-defined network. As shown in Fig. 1, the algorithm must first discover the relationships between the switches within the network, called topology. Using the tools provided by the controller, this algorithm can be activated when the switch or a communication link is added or removed. This is the first and most important part of the algorithm due to the importance of how switches are arranged in the network and their role in identifying anomalies. Due to this high sensitivity, the network topology detection process is activated and the database of this section is updated whenever there is a change in the network topology, whether deleting or adding a communication link or a switch in the network.

In the next step, to reduce the time of reviewing the rules from each other using the method of mediation of each rule recorded in the flow table, a sample will be selected to check the relationship between the rules of network switches each other using this rule. For this purpose, in this research, by considering the beginning and end of the interval specified by the rule, the middle value of the interval is determined and by using it, we can check the reaction of other switches to the rule. For example, if the rule we are researching includes IP addresses `1.1.1.1` to `1.1.1.10`, the IP address `1.1.1.5` is considered a representative of this rule, and the response of the other switches to it is estimated. In this way, we can avoid over-reviewing the entire range specified in the law and improve the speed of review and detection of the algorithm.

In the next step, Fig. 2 and 3 show a flowchart, and Algorithm. 2 displays pseudo-code to detect distributed anomalies. According to those Figures when a new rule is registered in one of the network switches, the proposed program generates a sample packet for

Algorithm 1 Pseudo-code of the Proposed Method for Detecting Centralized Anomalies

```
procedure DETECTION(R, RuleSet)
    relation ← unknown
    anomaly ← false
    M_x ← M(RuleSet[x])
    M_y ← M(R)
    for all variables k in M do
        if M_x[k] ⊇ M_y[k] then
            if relation ∈ {"superset", "correlated"} then
                relation ← correlated
            else
                relation ← subset
            end if
        else if M_x[k] ⊂ M_y[k] then
            if relation ∈ {"subset", "correlated"} then
                relation ← correlated
            else
                relation ← superset
            end if
        else
            return Disjoint
        end if
    end for
    A_x ← A(RuleSet[x])
    A_y ← A(R)
    if A_x = A_y then
        return Redundancy
    else
        if relation = "superset" then
            return Shadowing
        else if relation = "subset" then
            return General
        else if relation = "correlated" then
            return Correlation
        end if
    end if
end procedure
```

the new rule. Then, the proposed algorithm assumes that the new rule is registered. The route of the new traffic will be determined. Specifying the route will lead to identifying the final destination and predicted operation of packets and can help detect anomalies.

If the expected response to the desired traffic on the next switch is to delete the packet, the network rules will be recorded in such a way that the corresponding packets will be destroyed after consuming the network resources. In this way, a special type of distributed inconsistency called FTD will be observed, which the algorithm expressed in Algorithm. 3 will obtain using the topology and examine the relationships of the network switches, and the result will be notified to the user or application requesting to register the rule.

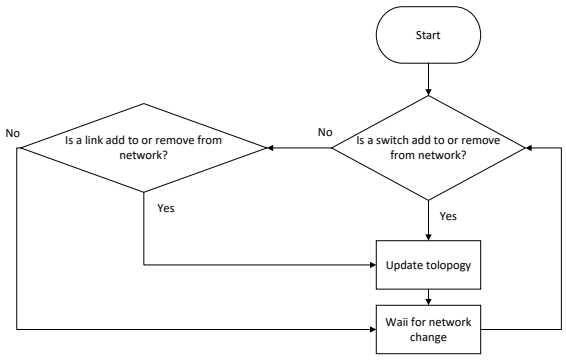

Figure 1. Topology Discovery Flowchart

Sometimes the rules of the network are set in such a way that packets generated by a host are continuously transmitted in the network loop and finally discarded when the packet TTL expires. In these cases, the proposed program will perform actions such as substituting the role of switches in the algorithm and performing the previous steps again, the address of each switch in the network that is checked during program execution is recorded. If one of the switches is seen twice, in fact, this network contains a loop that, if not identified, can lead to high network resource consumption. Furthermore, similar to the operation performed to detect FTD anomalies, this time to detect irrelevant anomalies, all the paths connected to the switch will be examined and the occurrence of the desired inconsistency will be feasible. If there is no rule in the neighboring switches of the device to send this traffic to the target switch, or in other words, it is not possible to receive the desired traffic to the switch, an irrelevant anomaly will occur.

5 Evaluation

The evaluation of the proposed algorithm is performed using the Mininet simulator and on the Linux operating system with fedora distribution and 2 GBs of main memory, which is based on the Ryu controller. To facilitate the evaluation of the proposed method, two inconsistency detection algorithms are separated and compared in a centralized and distributed flow table. First, the algorithm for detecting centralized anomalies within a flow table will be evaluated using the following method.

According to typical methods for evaluating algorithms, three sets of rules are executed as inputs in the algorithm. The processing time of the algorithm can be checked for the three worst, average, and best input states [21]. Fig. 4 shows the processing time of the algorithm for these three sets of rules mentioned in [22]. Also, to better evaluate the proposed algorithm with

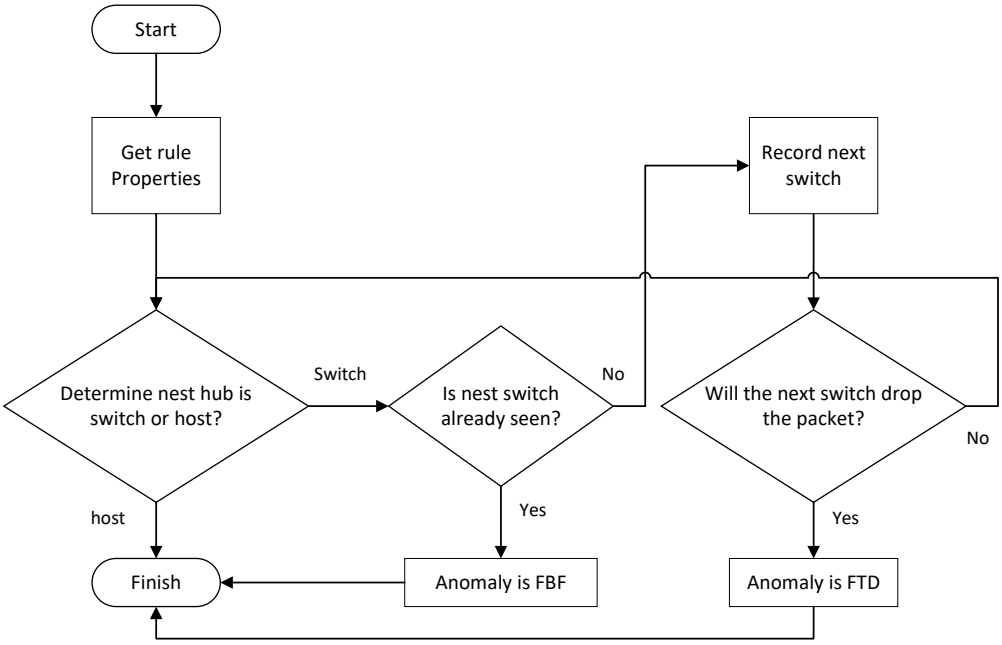

Figure 2. Flowchart of the Proposed Method for Detecting Distributed Anomalies

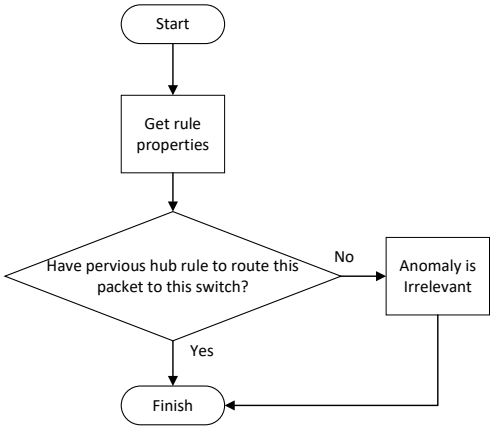

Figure 3. Flowchart of the Proposed Method for Detecting Irrelevant Anomalies

similar examples, the average state comparison diagram of this algorithm with the method proposed by Wang et al. [12] and RuleOut-Algorithm [19] is shown in Fig. 5. As can be seen in Fig. 5, in the same case between the two methods, the proposed algorithm has a shorter execution time than the Wang algorithm. In the other case, the evaluation of the detection algorithm in the distributed mode is performed by a network consisting of thirty switches and 100 hosts connected in the form of a tree similar to Fig. 6. According to Fig. 6, five switches are connected to each of the first layer switches, and finally, four hosts will be connected to each of these switches. To evaluate the method of detecting distributed anomalies, it was tried to consider rules with various path lengths and different states, Fig. 7 shows a comparison diagram of the number of existing distributed anomalies and the number of anomalies detected by the proposed algorithm, according to the use of the mediation technique discussed in the previous section, a sample packet of the rule is considered and the reaction of the other switches in the network will be examined. For this reason, some of the anomalies placed in the flow table inside the switches will not be covered, which will be reduced by using intelligence and including other parameters in the sample packet selection algorithm of the desired rule.

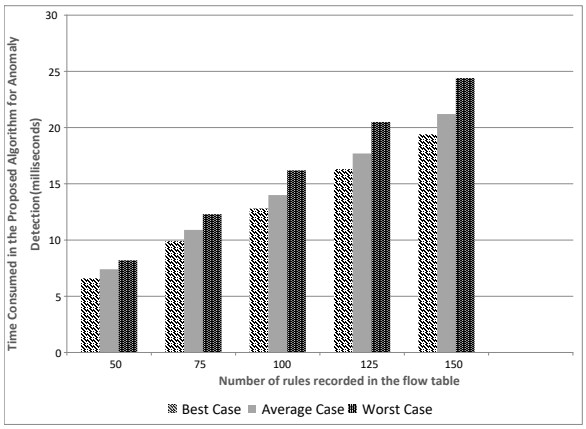

Figure 4. The Time Spent in the Proposed Algorithm in Terms of the Number of Rules Registered for Various Anomalies Cases.

Accordingly, the accuracy and recall of the proposed algorithm for the specified data set are shown in Fig. 8. Therefore, based on the conceptual analysis of the proposed algorithm in the centralized and dis-

Algorithm 2 Detection Algorithm

```
procedure DETECTION ALGORITHM(R, SwitchID)
    anomaly ← none
    CHECK(R, SwitchID)
end procedure
procedure CHECK(R, SwitchID)
    sample ← SelectASample(R)
    CHECKFBFANDFTD(sample, SwitchID)
end procedure
Global Static SetofSwitches
procedure CHECKFBFANDFTD(SamplePacket, SwitchID)
    if SwitchID is in SetofSwitches then
        anomaly ← FBF
    end if
    if A(SamplePacket) == FORWARD then
        nextDevice ← CalculateNextDevice(SamplePacket)
        if nextDevice is Switch then
            SetofSwitches.add(SwitchID)
            CHECKFBFANDFTD(SamplePacket, nextDevice)
        else
            continue
        end if
    end if
    if A(SamplePacket) == DROP then
        if SetofSwitches.contain(SwitchID) then
            anomaly ← FTD
        else
            continue
        end if
    end if
end procedure
return anomaly
```

Algorithm 3 Check Irrelevant Algorithm

```
procedure CHECKIRRELEVANT(SamplePacket, Switchid)
    anomaly ← Irrelevant
    prevDevices ← CalculatePrevDevice(SamplePacket)
    for switch in prevDevices do
        if switch has a rule to forward M to SwitchID then
            anomaly ← none
            break
        end if
    end for
    return anomaly
end procedure
```

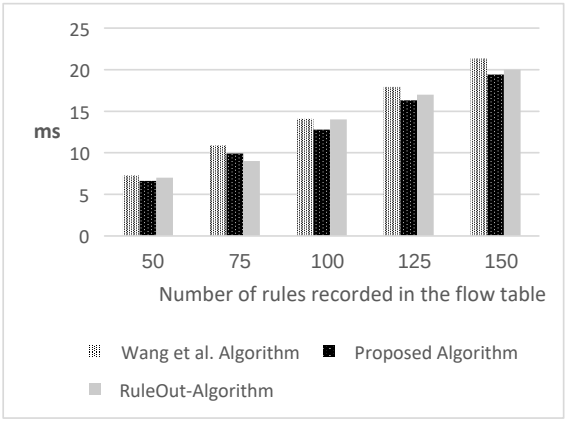

Figure 5. Comparison of the Execution Time of the Proposed Algorithm with the Algorithm of Wang et al. for a Set of an Average Number of Anomalies.

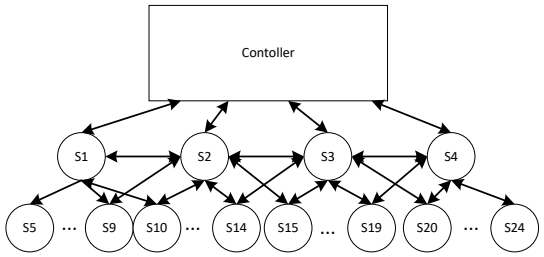

Figure 6. Network Topology to Evaluate the Second Proposed Method

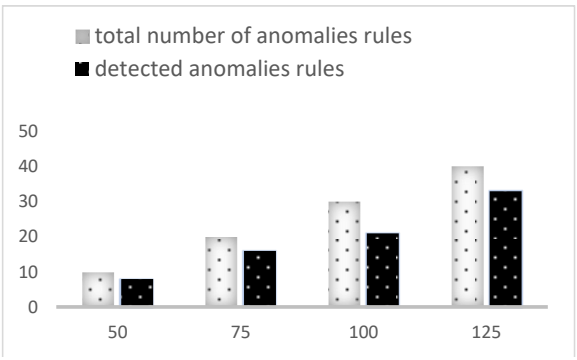

Figure 7. Comparison of Detected Distributed Anomalies with the Total Number of Distributed Network Anomalies.

tributed parts, in the worst case, we will have the time complexity of $O(n^2)$ and $O(Sn)$, respectively, where S represents the number of switches in the network and n represents the average number of rules in each switch.

A noteworthy point in evaluating this algorithm is the relationship between path lengths and detection time of the proposed algorithm. The number of anomalies in the flow tables is an issue that will have an impact far less than the length of paths predicted for packets. For this reason, in evaluating this algorithm, by increasing the number of rules in the

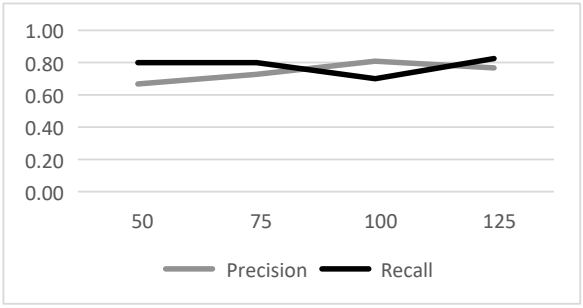

Figure 8. Accuracy and Recall of the Proposed Algorithm for the Specified Data Set with Different Numbers of Rules Submitted in Controller.

flow table, paths with longer lengths are created and sent to the proposed program. Therefore, increasing the length of traversed paths will increase the time required for processing by the proposed algorithm. Fig. 9 shows it. It should be noted that due to the recursive nature of the algorithm, the amount of time spent in the proposed algorithm depends on parameters such as the order of rules registration, the way the network is connected, how the rules are registered, etc. In addition, due to the parallel implementation of the FTD and FBF detection algorithm with the irrelevant detection algorithm and due to the longer execution time of the first two distributed anomaly detection algorithms, the entire execution time of the detection process is equal to the execution time of detecting FTD and FBF anomalies. In other words, with the addition of the irrelevant anomaly detection algorithm, almost no time will be added to the duration of the detection algorithm.

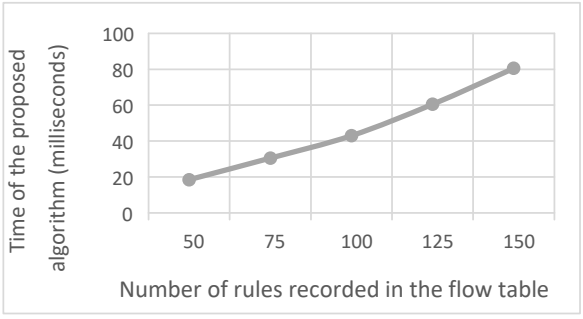

Figure 9. Consumption time of the proposed algorithm for detecting distributed anomalies.

6 Conclusions and Future Works

In this article, we have tried to identify anomalies in the Internet of Things based on software-defined networks using mathematical modeling to prevent such anomalies before entering the network and applying rules to it. First, the anomalies were expressed centrally within a flow table and distributed within several flow tables. To better understand the problem, their mathematical form was determined, and consequently, the two parts of the detection algorithm for centralized and distributed anomalies were described. Finally, a comparison of the proposed algorithm with similar examples is performed to be able to accurately assess the time required to detect centralized and distributed anomalies and determine how much it improves compared to other methods.

To determine the components of this research that can be continued in the future, two cases of processing time of the detection algorithm and how to correct the detected anomalies will be among the basic areas. In most networks today, time alone is one of the basic and important factors and the more this component reaches its minimum in optimization algorithms such as anomaly detection, can attract more and more experts. On the other hand, how to correct anomalies is another issue that, if not addressed properly, cannot complete the process of detecting anomalies.

References

[1] O. Flauzac, C. González, A. Hachani, and F. Nolot. SDN based architecture for IoT and improvement of the security. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pages 688–693. IEEE, 2015. doi:10.1109/WAINA.2015.110.

[2] L. Raju, M. Adhil, S. Logeshwaran, M. Sanjana, and V. K. Praveena. IOT based Advanced building automation and Energy Management. In *2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, pages 478–481. IEEE, 2022. doi:10.1109/AIC55036.2022.9848842.

[3] S. K. Tayyaba, M. A. Shah, O. A. Khan, and A. W. Ahmed. Software defined network (sdn) based internet of things (iot) a road ahead. In *Proceedings of the International Conference on Future Networks and Distributed Systems*, pages 1–8, 2017. doi:https://doi.org/10.1145/3102304.3102319.

[4] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh. A survey and classification of the security anomaly detection mechanisms in software defined networks. *Cluster Computing*, 24:1235–53, 2021. doi:https://doi.org/10.1007/s10586-020-03184-1.

[5] M. H. Khairi, S. H. Ariffin, N. M. Latiff, A. S. Abdullah, and M. K. Hassan. A Review of Anomaly Detection Techniques and Distributed Denial of Service (DDoS) on Software Defined Network (SDN). *Engineering, Technology and Applied Science Research*, 8(2), 2018.

[6] Y. Maleh, Y. Qasmaoui, K. El Gholami, Y. Sadqi, and S. Mounir. A comprehensive survey on SDN security: threats, mitigations, and future directions. *Journal of Reliable Intelligent Environments*, pages 1–39, 2022. doi:https://doi.org/10.1007/s40860-022-00171-8.

[7] H. Li, F. Wei, and H. Hu. Enabling dynamic network access control with anomaly-based IDS and SDN. In *Proceedings of the ACM International Workshop on Security in Software-Defined Networks and Network Function Virtualization*, pages 13–16, 2019. doi:https://doi.org/10.1145/3309194.3309199.

[8] P. Zhang, S. Xu, Z. Yang, H. Li, Q. Li, H. Wang, and C. Hu. FOCES: Detecting forwarding anomalies in software defined networks. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 830–840. IEEE, 2018. doi:DOI: 10.1109/ICDCS.2018.00085.

[9] B. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications surveys and tutorials*, 16(3):1617–34, 2014. doi:DOI: 10.1109/SURV.2014.012214.00180.

[10] A. Ahalawat, K. S. Babu, A. K. Turuk, and S. Patel. A low-rate DDoS detection and mitigation for SDN using Renyi Entropy with Packet Drop. *Journal of Information Security and Applications*, 68:103212, 2022. doi:https://doi.org/10.1016/j.jisa.2022.103212.

[11] A. H. Mohammed, R. M. Khaleefah, and I. A. Abdulateef. A review software defined networking for Internet of Things. In *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–8. IEEE, 2020. doi:10.1109/HORA49412.2020.9152862.

[12] P. Wang, L. Huang, H. Xu, B. Leng, and H. Guo. Rule anomalies detecting and resolving for software defined networks. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015. doi:10.1109/GLOCOM.2015.7417386.

[13] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu. A security enforcement kernel for OpenFlow networks. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, pages 121–126, 2012. doi:https://doi.org/10.1145/2342441.2342466.

[14] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey. Veriflow: Verifying network-wide invariants in real time. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, pages 49–54, 2012. doi:https://doi.org/10.1145/2342441.2342452.

[15] E. Tsogbaatar, M. H. Bhuyan, Y. Taenaka, D. Fall, K. Gonchigsumlaa, E. Elmroth, and Y. Kadobayashi. SDN-enabled IoT anomaly detection using ensemble learning. In *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part II 16*, pages 268–280. Springer International Publishing, 2020. doi:https://doi.org/10.1007/978-3-030-49186-4_23.

[16] A. AuYoung, Y. Ma, S. Banerjee, J. Lee, P. Sharma, Y. Turner, C. Liang, and J. C. Mogul. Democratic resolution of resource conflicts between sdn control programs. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 391–402, 2014. doi:https://doi.org/10.1145/2674005.2674992.

[17] E. Al-Shaer and S. Al-Haj. FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures. In *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*, pages 37–44, 2010. doi:https://doi.org/10.1145/1866898.1866905.

[18] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi. Hierarchical policies for software defined networks. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 37–42, 2012. doi:https://doi.org/10.1145/2342441.2342450.

[19] G. Zhang, S. Cheng, X. Song, and F. Jiang. Detecting and Resolving Flow Entries Collisions in Software Defined Networks. In *Proceedings of the 2019 3rd International Conference on Computer Science and Artificial Intelligence*, pages 245–251, 2019. doi:https://doi.org/10.1145/3374587.3374614.

[20] R. Kiani and A. Bohlooli. Distributed Rule Anomaly Detection in SDN-based IoT. In *2021 5th International Conference on Internet of Things and Applications (IoT)*, pages 1–6. IEEE, 2021. doi:10.1109/IoT52625.2021.9469714.

[21] S. Xi, K. Bu, W. Mao, X. Zhang, K. Ren, and X. Ren. RuleOut forwarding anomalies for SDN. *IEEE/ACM Transactions on Networking*, 31(1):395–407, 2022. doi:10.1109/TNET.2022.3194970.

[22] F. Valenza and M. Cheminod. An Optimized Firewall Anomaly Resolution. *J. Internet Serv. Inf. Secur.*, 10(1):22–37, 2020.

Rasool Kiani received his B.Sc. and M.Sc. degrees in computer architecture engineering at the University of Isfahan, Iran in 2015 and 2017, respectively, and was accepted as a Ph.D. candidate in the field of computer architecture engineering at the University of Isfahan, Iran in 2021. His research interests include cyber-physical Systems, the Internet of Things, and autonomous vehicles.

Ali Bohlooli received his B.Sc. and M.Sc. degrees in Computer engineering (with honors) from the Department of Electrical & Computer Engineering, Isfahan University of Technology, Iran, in 2001 and 2003, respectively. Subsequently, he received his Ph.D. degree in computer engineering from the University of Isfahan, Iran, in 2011. Presently, he is an associate professor in the Faculty of Computer Engineering, University of Isfahan, Iran. His research interests include Intelligent Cyber-Physical Systems, the Internet of Things, Embedded Systems, and IIoT Security.