DOS DETECTION USING MACHINE LEARNING AND SOFTWARE DEFINED NETWORKS

Eng. Rudolf Grežo
Faculty of Informatics and Information Technologies - University of Technology in Bratislava

rudolf.grezo@stuba.sk

Abstract: *Software Defined Networks (SDN), which are new paradigm for building networks, provide a possibility to increase network performance and security. SDN centralize network intelligence in one network node called controller and underlying infrastructure which transport data across the network consists of switches which are orchestrated via appropriate protocol by the controller. Switches are cheaper because all the protocols needed to perform routing and other services in the network are centralized in the controller. Next advantage of the centralized SDN architecture is that information about the state and the behavior of the network are available in the controller. We can use this information to make critical decisions to better utilize network resources and improve network security. At the controller we perform network traffic monitoring, analysis and management. In this work we propose to use extended measurement vector and Machine Learning (ML) model to detect Denial of Service (DoS) attacks. Controller then take actions based on the ML model output to stop or counter the attack.*

Keywords: NETWORK SECURITY, SOFTWARE DEFINED NETWORKS, NETWORK ATTACKS

1. Introduction

Big Internet Service Providers (ISPs) are constantly looking for more flexible and cheaper solutions for providing new network services or operating the existing ones. In today's networks, the number of users significantly grows. More users connected to the network may motivate more attackers to try to perform network attacks on the network or use various techniques to benefit from network vulnerabilities. In order to keep networks secure, we have to be constantly looking for new possibilities how to prevent or detect network attacks. Often, there is a need to optimize existing solutions to achieve efficient usage of present resources and possibilities to secure the network. In order to detect network attacks, we need to monitor network traffic behavior and different parameters. Existing network monitoring and management tools are often based on proprietary solutions and may not necessarily provide information useful for attack detection. Proprietary solutions may also bring network lock-in in terms of introduction of new services and applications. Our approach is based on open solutions, which on the other hand increases scalability and provide space to implement cheaper, easier to operate networks, which are also vendor independent. This work will be based on UnifyCore [1] network architecture.

1.1. Obsolete approach

Automation, programmability and security are the most needed and wanted characteristics of modern networks. Today's Internet Service Providers are still using network solutions based on proprietary devices. In these devices, control plane is represented by an operating system which has its logic and makes appropriate decisions. On the other hand, data plane is in most cases realized by ASIC (Application-Specific Integrated Circuit) which forwards traffic and cooperate with operating system of particular device. However, this approach limits network automation and programmability and brings some unnecessary configurational overhead. Each device has to be configured separately by network administrator. This may bring also potentially dangerous configuration errors which might be exploited by the attackers.

1.2. Software Defined Networking

Software Defined Networks (SDN) represent new approach for building networks [2, 3]. These networks logically centralize network intelligence into a single node called controller that controls nodes for processing user data, in UnifyCore called forwarders (switches), using a standardized interface. Fact, that switches may not have advanced intelligence directly reduces demands on their implementation and hence price. SDN also logically separates network into three segments: SDN applications, control plane and data plane [4]. The controller represents control plane and the switches, which forward the traffic in the network, represent data plane. OpenFlow is described as first and most successful protocol used for communication between control plane and data plane. In its recent specification [5] is described how to categorize ports, tables which are needed for pipeline processing and support for multiple controllers. There are also interesting extensions as TCP flags matching and others, which open new possibilities to improve network security. Securing SDN architecture is next step in overall progress to have modern, flexible, scalable and secure networks.

Result of SDN approach is not only less configuration overhead and lower price of end devices, but also that routing and statistical information about network traffic are available in the single logical entity. Based on available information, controller can make critical decisions, which can affect network behavior in order to increase security. While using SDN approach we can also analyze network traffic statistics in order to detect possible security threats and ongoing attacks.

1.3. Objective and structure

This paper is dedicated to proposal of new model for monitoring, analysis of network traffic using SDN approach with detection and prediction of network attack. Main objective of the paper is to increase security in the SDN environment, which is one of the most challenging task in developing future networks. In order to address this, we propose to use SDN approach while utilizing controller as a multipurpose entity, which analyzes traffic statistics to possibly detect or predict and mitigate ongoing network attack therefore increase the security of the network.

Paper structure is as follows: Related work section is focused on the state of art analysis, challenges and security threats in SDN and possibilities of increase security in SDN environment. Third section introduces new ideas to the existing SDN architecture, which has been developed and extended over last couple of years on the faculty. Last section summarizes the work, contribution and future work on the presented research problem.

2. Related work

Vulnerabilities in computer systems and networks might be result of poor design, incorrect configuration or implementation of specific computer system or network. Design issues come typically with lack in experience of the system designer. Root cause of such an inexperience might be unavailable documentation or simple that responsible person did not undergo the necessary training. Incorrect implementation is usually result of negligence or error during the implementation [6]. Common result of above issues is, that attackers might take advantage of these issues and might get access to the system or information, which should be unavailable to them in case of correct system operation. Correctly designed and implemented system, must not allow this situation to happen. Nevertheless, even if the best security measures are implemented in the system, there is usually some attacker, who might be able to do damage to the computer system or network, its users or owners. Therefore, improving security of the system have always been one of the top priorities. However, improving security also comes with cost of those security measures. General rule of thumb is that to overcome security measures, attacker needs to spend more resources than what is worth the information, which is protected by the security measures.

On the other hand, when the attack occurs, the computer system should be gathering information, which help to detect the ongoing attack and possibly prevent the similar attack in future. Detection of the attack is usually based on detecting anomalous sequences in the monitored data. In computer networks, abnormal behavior of traffic in the network might be detected as congestion of the network. Next type of an attack can be described as targeting single node in the network in order to use all of its resources with invalid requests, so it will not be able to handle valid request. This kind of attack is known as Denial of Service (DoS) [6]. Attacks might also be aimed to gain information for future use. Port scanning in the network might be first stage of attacking the network. Port scanning only check for vulnerabilities, which might be later used during DoS attack. Other attacks might be focused to break system passwords. Example of these attack can be brute-force attack, rainbow table attack or dictionary attack. Attacks which use newly discovered vulnerabilities in the system are called first-day attacks. These attacks are usually very hard to detect. In this work, we will focus on detection of DoS attacks based on the traffic anomalies in the computer networks.

2.1. Security in SDN, general view

As described in [7], trends in SDNs security were either to improve the security of current networks utilizing SDN [8] or to improve the security of SDN themselves [9]. In this work, we will focus on security improvements in SDN. Authors also discussed various solutions to measurement, monitoring and proposals for security improvements. Except from the discussed monitoring methods, StatSec [10] a novel approach was proposed and compared with sFlow. It appears to be efficient in terms of control plane because it uses distributed approach. However, it goes against the SDN centralized paradigm and requires intelligence to be implemented into OpenFlow switches.

Authors in [11] also first compared multiple security solutions to SDN. Next authors defined taxonomy of SDN security solutions. None of the discussed solutions provided security to all SDN layers or interfaces. Finally, they defined security threats and possible attack classes in SDN. Generally, four major challenges for securing SDN emerged:

- Securing the controller.
- Protecting the flow paradigm of the SDN.
- Securing the data plane (SDN switches).
- Hardening APIs and comm. channels.

Other part of securing the SDN network is to ensure error free configuration. In this work will assume that configuration is correct, so configuration checks are out of scope of this work. Mastering of above challenges in SDN security require further knowledge about the SDN network data plane and control plane.

2.2. Network anomalies detection

In computer networks, we use intrusion detection systems (IDSs) to detect attacker’s activities. IDS is deployed in network along with the other security measures. Deployment of IDS can be on separate host nodes in the network when it is called host-based IDS (HIDS). On the other hand, it can be deployed as a network-based IDS (NIDS). HIDS examine internal structures of computer system. NIDS on the other hand its interfaces. HIDS for example verify server logs, for unusual things as root password changes. The idea of IDS is that attacks occur less often than normal behavior. NIDS focus on detection of attacks in network traffic. Example of suspicious behavior in case of NIDS can be detection of large number of TCP connection requests to a very large number of different ports in short time. This is the behavior of aforementioned port scanning [6].

In order to have data available for NIDS, we first need to monitor the network activities. In UnifyCore TE extension [12], measurement of QoS specific parameters was done. As input parameters for NIDS different parameters will need to be monitored. Specific flow table entries must be added in order to match certain types of traffic. We will also create hierarchy of flow tables on the switches from which we will be able to get more detailed statistical information about forwarded traffic.

Parameters for malicious flow detection need to be selected based on the requirements for specific network attacks characteristics. After we will have values for these parameters, we will need to be able to detect anomalies and patterns in those data in order to detect attacks or optimize the SDN network performance. Therefore, research in machine learning might help with resolving above challenges in anomalies detection and optimization.

2.3. IDS and attack mitigation in SDN

First solution [13] utilize SDN paradigm in order to achieve more accurate view of the network. Solution evaluates sampling solutions for measuring network parameters and use SDN as possibility to measure parameters and perform anomaly detection in order to detect Distributed DoS attacks. Proposed is also detection on the edge of the network at the inter-domain connection in order to detect attacks entering the network from another one. The detection is performed on the controller. Solution proposes that packet can be dropped on the remote OpenFlow switches by the controller. This statement is not correct, as controller is only able to modify flow tables of the switches to drop specific matching packets.

Next solution [14] uses machine learning and self-created virtual environment for implementation and testing purposes. Authors utilize unsupervised machine learning method of self-organized maps (SOM) to detect anomalies. Features selected for machine learning methods are supported by OpenFlow specification. The detection method looks promising, unfortunately, evaluation does not provide any comparison with existing solutions nor the detection rate. Solution does not provide any proposal for detected attack mitigation.

Sahay et al. [15] proposed adaptive attack mitigation in ISP networks. Solution focus on DDoS attacks and try to reduce the impact of network congestion on ISP customers. Solution need high level policy definition from the administrator and then it performs management and enfnt of the policies automatically. Solution was tested and evaluated in experiment with three customers which is relatively small scope. Disadvantage of the solution might be possible conflicts and cross enforcement between customers.

Next solution discussed in this section [16] focus mainly on mitigation of the attacks. Proposed are three classes of actions to mitigate the malicious flow. Solution can either block, forward or apply QoS to the flow. Authors evaluated solution in virtual environment by blocking TCP SYN flood attack which led to fully restored network operation. Authors used heuristic approach to TCP SYN flood detection by simply evaluating the rate of TCP SYN messages. The solution is a proof of concept and brings interesting ideas into attack mitigation in SDN networks.

T. Tuan A et al. in [17, 18] in former work developed Deep neural network (DNN) model using six features to detect four categories of attacks. This network was trained by using NSL-KDD Dataset [19]. This dataset is quite old and has its own drawbacks, however it is still a good tool for comparing performance of IDS models [20]. Results for this work were average in comparison to other algorithms like Random Forest or Support Vector Machines (SVMs). In later work, authors opted for Deep Recurrent Neural Netwok (DRNN) model with Gated Recurrent Units (GRUs), which significantly improved the learning performance of the network and attack detection accuracy. According to their evaluation, GRU-RNN model have the best performance out of the compared models. With the accuracy 89% and minimalistic approach to the features selection, this result is very promising but, in our opinion, there is still room for future research.

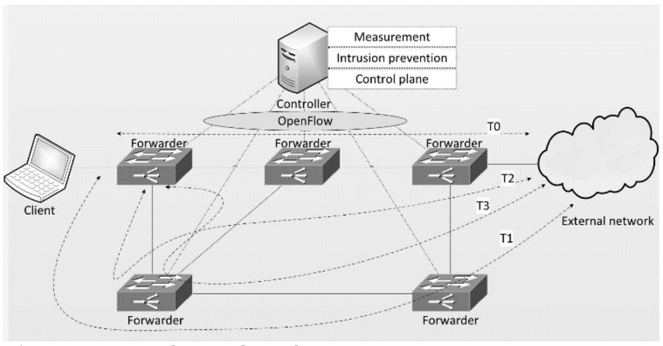

Fig. 1 *Experimental network topology*

3. *Proposed solution*

Based on the research mentioned in section Related work we will contribute with our proposal. Overall network performance including level of security can be represented by multiple metrics. These metrics may be very complex. To be able to have sufficient view on our network, we will use standard OpenFlow protocol message (ofpmp_port_stats) which can gather information about the interface counters or flow counters on the particular OpenFlow switch (forwarder). All possible measurement parameters are specified in OpenFlow specification [5]. Good Example of such parameters can be TCP flags header matching fields. Header matching fields are used in the flow table entries to match specific flows. Each flow table entry has its counters. Counters can be collected by the SDN controller per flow or per port of of the OpenFlow switch. Counters provide a lot of information about the data in the network and this information can be used to detect network attacks. As discussed in previous section, TCP SYN flood is one of the attacks which can be detected by measurement and evaluation of these counters. Current UnifyCore implementation, does not consider these counters relevant, because it uses measurements only in traffic engineering extension which has different motivation. We propose to include these counters into this measurement. These measurements will be taken at the edge of the network and will be used as an input features for our DNN which is proposed later in this paper.

Based on machine learning model and information available in the controller, we can make critical decisions to mitigate or stop the attack. Controller will then perform actions as add, modify or delete flow table entries on the switches using appropriate OpenFlow messages. Description and purpose of the aforementioned actions:

- Flow addition will be used to route suspected traffic via different paths and load-balance to reduce possible attack effect on regular traffic. This action is possible only in specific network topologies.

- Flow modification can reduce attack effectivity by using OpenFlow Meter table for shaping of the malicious traffic. This action might also divide flows to minimize impact for legitimate traffic.

- Flow deletion (drop) is the last resort applicable when attack is identified with certain probability and we need to stop it.

Next, we propose to modify and extend API, which provides measurement information about the network orchestrated by our SDN controller. This extension will include new measured counters and will improve usability of our solution. Using this API together with one of existing tools used for IDS can improve the effectiveness of network attack detection and increase detection rate. We can evaluate the accuracy of the used IDS tool together with our model and try to reduce the false positive detection rate. Controller will then choose one of the aforementioned actions based on the score, which will be calculated based on our machine learning model and the input from IDS. In order to sustain the configurability and modularity of the solution, we provide possibility to configure parameters for this score calculation via the controller API. In our architecture in Figure 1, we use Ryu [21] controller because of the OpenFlow version 1.5 support which provide better possibilities for feature selection.

Table 1: *Mapping of DNN output to Controller action*

1. Output (y_1)	2. Output (y_2)	Action
0	0	No action
0	1	Flow modification
1	0	Flow addition
1	1	Flow deletion (drop)

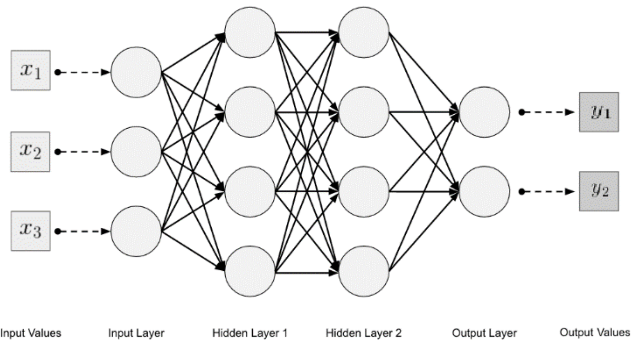

Fig. 2 *Example of the neural network model*

3.1. *Deep learning neural network model*

We propose DNN models with nine Input Values (features) on the Input Layer. We use two, four and eight Hidden Layers and two Output Values (y_1, y_2) in the Output Layer. In the Figure 2 you can see example of the simple neural network model for better illustration. Two Output Values are used to accommodate possible actions for the flow. Controller take the Output Values of the DNN model and evaluate them according to the specified mapping. In table 1 you can see the mapping of the output of the DNN to the possible actions performed by the controller with the particular flow which were described earlier. Because actual Output Layer neurons might be activated with numbers between 0 and 1, Output Values in the table might vary so we established the thresholds for the controller to evaluate the outputs. We have selected first seven features to be used as an input values from the header match fields:

1. OXM_OF_IPVX_SRC - IP source address.
2. OXM_OF_IPVX_DST - IP destination address.
3. OXM_OF_TCP_SRC - TCP source port
4. OXM_OF_TCP_DST - TCP destination port
5. OXM_OF_TCP_FLAGS - TCP flags
6. OXM_OF_UDP_SRC - UDP source port
7. OXM_OF_UDP_DST - UDP destination port
8. Duration – Duration counter of the flow entry
9. Received Bytes – bytes received per flow entry

and the last two two from the List of counters of the OpenFlow switch specification [5], resulting in nine features.

4. *Results of discussion*

We are currently in the process of evaluation of our solution. We will use standard evaluation metrics as accuracy (A), precision

(P), recall (R) and F-measure (F). Which will be calculated using these equations:

$$A = \frac{(TP + TN)}{(TP + FN + FP + TN)} \tag{1}$$

$$P = \frac{TP}{(TP + FP)} \tag{2}$$

$$R = \frac{TP}{(TP + FN)} \tag{3}$$

$$F = 2 \times \frac{P \times R}{P + R} \tag{4}$$

Where True Positive (TP) - the number of normal records correctly classified, True Negative (TN) - the number of anomaly records correctly classified, False Positive (FP) - the number of normal records incorrectly classified. False Negative (FN) - the number of anomaly record incorrectly classified.

In our solution, we also plan to use more datasets which are available from networks traffic measurements. Except the NSL-KDD one of the other examples might be UNSW-NB15 [21] dataset. These datasets contain regular traffic and also malicious traffic which is labeled. We also have unlabeled datasets from our own traffic generating experiments. As these datasets are considerably large, the challenge here is to identify the attacks (malicious traffic) in the datasets and extract the information (label them), which we will then use for training purposes of the machine learning model.

5. Conclusions

The main goal of this paper was to propose our solution of network intrusion detection and prevention system and provide new ideas regarding the presented research problem which is lack of secure solutions in SDN environment. At first, the work is focused on the analysis of existing research in the area of SDN and possible usage of SDN to increase network security. This helps to understand the challenge and possibilities in developing secure software defined network solutions. Next, we proposed new solution which includes addition of measurement, detection and prediction modules into controller. These modules will also use machine learning methods in order to detect or predict an attack. We proposed DNN approach which will detect the network attack and controller will then decide to mitigate or stop attacks. In our solution, we try to solve numerous security challenges. In future work, we will focus on detailed proposal of our secure SDN architecture and individual modules. Final proposal will be implemented into the existing SDN architecture and evaluated for its behavior and performance. Results will be evaluated and compared with other solutions. Our solution will increase security in the SDN environment, which is the main contribution of our work.

Acknowledgment

This project was partially supported by the Eset Research Centre and Scientific Grant Agency of the Slovak Republic, grant No. VEGA 1/0836/16.

References

[1] M. Nagy, I. Kotuliak, J. Skalny, M. Kalcok and T. Hirjak, “Integrating mobile openflow based network architecture with legacy infrastructure,” in Inform. and Commun. Technology. Lecture Notes in Comput. Sci., vol 9357. Springer, Cham pp. 40-49.

[2] Open Networking Found. (2012, April 13). "Software-defined networking: the new norm for networks", [Online]. Available: https://www.opennetworking.org /images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[3] T. D. Nadeau, K. Gray, "Centralized and distributed control and data planes", in SDN: Software Defined Networks, Sebastopol: O’Reilly Media, 2013, pp. 9-46.

[4] Open Networking Found. (2014, June). "SDN architecture" [Online]. Available: https://www.opennetworking.org/images/stories/doloads/sdn-resources/technical-reports/ TR_SDN_ARCH_1.0_06062014.pdf

[5] Open Networking Found. (2015, March 26). Openflow switch specification version 1.5.1 (Protocol version 0x06) [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/ openflow-switch-v1.5.1.pdf

[6] D. K. Bhattacharyya, J. K. Kalita, "Anomalies in a Network" in Network Anomaly Detection: A Machine Learning Perspective, Taylor & Francis Group, LLC, 2014, 336p.

[7] D. Kreutz et al., "Software-Defined Networking: A Comprehensive Survey, " Proc. IEEE, vol. 103, vol. 1, pp. 14–76.

[8] S. Shin et al., "Enhancing Network Security through Software Defined Networking (SDN)," 25th International Conference on Computer Communication and Networks, 2016, pp. 1-9.

[9] S. Bian, P. Zhang, Z. Yan, "A Survey on Software-Defined Networking Security," in Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications, 2016, pp. 190-198.

[10] J. Boite et al., "StateSec: Stateful Monitoring for DDoS Protection in Software Defined Networks," IEEE Conference on Network Softwarization, 2017, 9p.

[11] A. Akhunzada et al., "Securing Software Defined Networks: Taxonomy, requirements, and Open Issues," IEEE Communications Magazine, 2015 63p.

[12] R. Grežo, M. Nagy, "Network traffic measurement and management in software defined networks", in Proceedings of the 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp. 541 – 546.

[13] M. E. Ahmed, H. Kim, "DDoS Attack Mitigation in Internet of Things Using Software Defined Networking, " IEEE Third International Conference on Big Data Computing Service and Applications, 2017, pp. 1-6.

[14] D. Jankowski, M. Amanowicz, "Intrusion Detection in Software Defined Networks with Self-organized Maps," Journal of Telecommunications & Information Technology, 2015, pp. 3-9.

[15] R. Sahay et al., "Adaptive Policy-driven Attack Mitigation in SDN," in Proceedings of the 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures, 2017, pp. 1-6.

[16] P. Bull et al., "Flow Based Security for IoT Devices using an SDN Gateway," IEEE 4th International Conference on Future Internet of Things and Cloud, 2016, pp. 157-163.

[17] T. Tuan A et al., "Deep learning approach for network intrusion detection in software defined networking", In Proceeding of the International Conference on Wireless Networks and Mobile Communications (WINCOM), 2016, pp. 258-263.

[18] T. Tuan A et al., " Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks", In Proceeding of the 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), 2018, pp. 202-206.

[19] "KDD Cup 1999", [online] Available: http://kdd.ics.uci.edu/databases/kddcup99/.

[20] S. REVATHI, A. MALATHI, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection.", in International Journal of Engineering Research and Technology (IJERT), 2013, pp. 1848-1853.

[21] "Ryu", [online] Available: http://http://osrg.github.io/ryu/.

[22] M. Nour, S. Jill. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).", in Proceeding of the Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1-6.