Volume 5, Issue 6, June 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Efficient Anomaly Detection Using Adaptive Monitoring in SDN

Gagandeep Garg, Roopali Garg
Research Scholar, Dept. of IT
U.I.E.T., PU, Chandigarh, India

Abstract— Network monitoring and measurement is the key task in today's networking scenarios due to increasing low-level intrusions. With the increase in utilization of resources and wider network bandwidth gateway for intruders also enlarges. Hence to detect the anomalies entered by the intruders inside our network a better anomaly detection mechanism must need to be implemented. Also software-defined networking is a rapidly evolving networking technology which provides a centralized platform to control the network behaviour. Using SDN it is easy to monitor the network traffic as compared to the previous existing networking scenario. For detecting anomalies efficiently, some techniques already been proposed but still a dynamic and responsive mechanism is required for intrusion free network. In this paper, a work towards the improvement of existing anomaly detection algorithm for dynamic rule updation is provided with respect to flow counting of network traffic. Also, monitoring overhead must be considered while applying monitoring policies upon the network so that network's performance can also be maintained.

Keywords— Software defined networking, intrusion, abstraction, security, monitoring and measurement.

I. INTRODUCTION

Network security is the fundamental requirement in networking. For the security of network, it is required that we can efficiently measure and monitor our network so that any intrusion inside the network can be detected and handled accordingly [1]. It is very complex task to monitor the whole network with such vast increase in utilization of networks. However, software-defined networking is a new technology which provided a centralized view of the network using a SDN controller. From that centralized view of network it can be possible to monitor our network and measure its statistics. Software defined networking separates the control of network [2] i.e. control plane from the data devices (forwarding switches). It reduces the computation overhead of the networking switches and all the forwarding decisions have been made by controller at control plane. To monitor the traffic efficiently several techniques already proposed. In [3] several proposed methods to tackle such security issues are listed and compared according to their merits and demerits. SDN provides flexible open interfaces for programming the control of the network from the centralized control plane. Such open interfaces also liable to some attacks. Such scenario is implemented in [4] to perform DOS attack on the SDN. To mitigate such attacks efficient monitoring policies need to be implemented on SDN control plane. Using those monitoring policies some real time statistics must need to be captured and perform decisions according to those captured statistics.

However monitoring of every packet or bytes of data in our network is also not feasible because it will increase the monitoring overhead on the controller. It will also reduce the response time of controller which will results in poor network's performance. Hence very precise decision must need to be taken to decide which data need to be monitored. For such scenario an anomaly detection technique is provided in [5]. In this methodology a dynamic approach is followed for updating the rules to gather the statistics and detect the anomalies according to the flow count of our network traffic. Our network traffic flows are aggregated and linear prediction is used to predict the value of next flow-count dynamically. According to this predicted value decision of rule updation is applied to expand or contract the number of rules applied on our network traffic. If there is security then only there is secure communication and good output of network [6].

The rest of paper organized as follows: In II, motivation towards this adaptive anomaly detection mechanism technique is defined. In III, methodology used is explained. In IV, implementation setup used for the algorithm is discussed. In V, evaluation of obtained results is performed. Finally in V, paper is concluded by discussing the requirement and importance of adaptive anomaly detection mechanism and future work which can be done.

II. MOTIVATION

Dynamic flow path updating in SDN motivates us towards the work of adaptive anomaly detection using dynamically flow-counting. Our work also draws from the inspiration from existing methods for monitoring and detecting anomalies inside the network. In in [7] packet sampling for network traffic using port scan mechanism for each and every switch is implemented. In [8, 9] various methods different network topologies using NetFlow for anomaly detection implemented. Entropy is used as a summarization tool for classification and aggregation of traffic. Also supervised learning for detection of DOS attack is used. In [10] methods used for anomaly detection motivated us for applying statistic collection methods for monitoring. In [11] rate limiting and maximum entropy detection methods used for four prominent traffic anomaly detection algorithms. In [12] time trade-off between statics collection overhead and performance is provided. In [13] an approach towards improving the efficiently is implemented. On behalf of this given technique, in this work this redefined algorithm is further implemented with gathered statistics and evaluated with existing results.

III. METHODOLOGY FOR ADAPTIVE MONITORING

This proposed adaptive anomaly detection is able to maintain a balance between monitoring overhead of the controller and network's performance. Adaptive mechanism acts dynamically according to different patterns of data flows in our network. In this method we initially divide our network according to network prefix so that our data can be aggregated and rules can be applied on aggregated data. Those chunks aggregated according to prefixes can be further divided into smaller chunks of traffic in case of lesser network traffic. By dividing into chunks, finer granularity rules can be applied on data. From the previously accounted network data flows, a predicted range (R_L-R_U) of flow counts can be calculated using linear prediction formulas given in [5]. Then for further flows, our current flow count value will be compared to that predicted range of values. If our current values is lower than R_L then expand flag will be set on that flow. However, if value is higher than R_L but lower than R_U then contract flag is set and if value comes out higher than predicted range then an alarm will be raised.

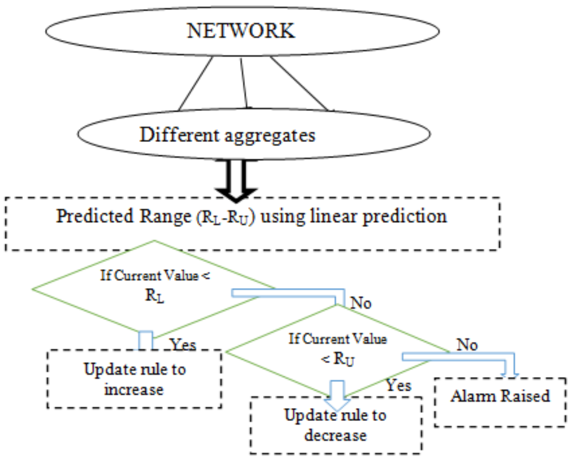


Fig. 1 Flow diagram of efficient dynamic rule updating algorithm

Including assigning these flags a dynamic rule updation decision will need to be made. If expand flag is setting then it means rules can be expanded i.e. lesser traffic then more number of rules can be applied on data so that finer granularity of rules can be applied. However if contract flag is setting on any data flow then rules need to be contracted i.e. larger amount of data present in network so lesser number of rules need to be applied to reduce the monitoring overhead. Flow diagram of this algorithm is shown in figure 1.

IV. IMPLEMENTATION

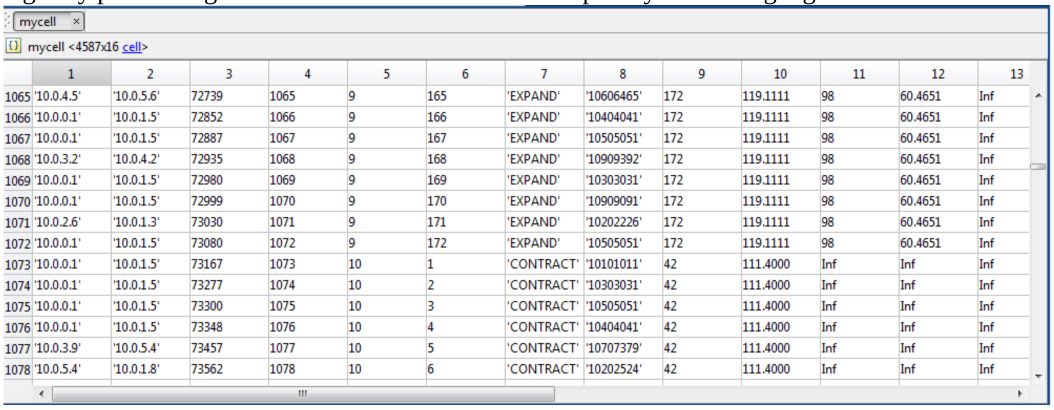
For implementation of this algorithm, a controller based simulator is created using MATLAB environment. In MATLAB different modules created for implementing the real scenario as due to large scale of network traffic, it is not feasible to capture the real network traffic aggregates. Different simulator modules created are controller, topology builder, packet generation, data propagation and entropy calculation modules. These modules perform tasks like creation of dynamic network topologies, generate different sized packets, statistics collection at the controller etc. These captured statistics can be used for dynamic calculation of predicted range of values and making decisions according to these values. Different rules [14] applied on the basis of decision are:

- Rule#1. Trace Mac address from the Source IP address.
- Rule#2. Scan the current flow to capture the throughput for a particular second.
- Rule#3. It will scan the network flows for average traffic throughput of whole network.
- Rule#4. This rule will checks the link reliability of traffic for each link.
  Link-Reliability = (100 – drop-Rate)
  *Where drop-Rate = (no. of packets * bottleneck)/100*
- Rule#5. This will find the reliability of our aggregated data up-to particular time.
  Reliability = 100– ((drop-Rate / packet-history) * 100)
- Rule#6. Alarm status will be set if count value reached above the given value.
- Rule#7. Calculate no. of anomalies detected i.e. no of times alarm is raised per 1000 packets.
- Rule#8. This will counts the no. of rules per switch.
- Rule#9. It will detect the delay in detecting the anomalies.

These rules will be applied on the basis of decision made including with setting the flags on the traffic aggregates. If setting expand flag then nine rules will be applied otherwise if setting contract flag then first six rules will be applied to reduce monitoring overhead. Different statistics captured using our simulator is shown in figure 2. Those captured statistics include the source IP-address, destination IP address, Number of bytes transferred, number of packets per flow, total number of packets transferred, flag set for each aggregate, Mac address, link reliability, aggregated throughput reliability, alarm status , rules per switch etc. here in figure 2. INF represent the null value i.e. in case of contract flag some rules are not applied hence the value of those rule set as INF. Also 13th column shown in figure shows the alarm status. It is showing all values as INF because no anomaly is detected for that given second of time for which this figure is capture.

V. RESULTS AND EVALUATION

These gathered statistics using our simulator can be used to obtain the entropy and a bar graph of entropy is plotted with respect to the prefix mask length calculated for aggregated data. This bar graph is shown in figure 3.2 In this figure red bars represent the entropy of prefix size difference of aggregated data with respect to prefix mask length and blue bars shows the volume of prefix size difference. According to this bar graph with the increase in prefix mask length entropy value decreases similar to the case given in [5] as shown if figure 3.1 taken from the existing results. Hence in comparison to these existing results our results are almost similar. As no particular parameters are given in existing dynamic rule update algorithm so exact match of values cannot be possible because these values keep on changing each time our simulation runs according to the size of packets and network traffic flows. But still flow of our results are similar to the existing results. We had obtained these results at simulation time (ST) =120 seconds. However from the existing algorithm we had reduced the step of setting flags and updating the rules dynamically on the basis of these recorded aggregates. Instead of recording the aggregates, rules are dynamically updated immediately at the time of setting the flags. By performing this we had reduced the time complexity of existing algorithm to certain extent.


mycell

mycell <4587x16 cell>

	1	2	3	4	5	6	7	8	9	10	11	12	13
1065	'10.0.4.5'	'10.0.5.6'	72739	1065	9	165	'EXPAND'	'10606465'	172	119.1111	98	60.4651	Inf
1066	'10.0.0.1'	'10.0.1.5'	72852	1066	9	166	'EXPAND'	'10404041'	172	119.1111	98	60.4651	Inf
1067	'10.0.0.1'	'10.0.1.5'	72887	1067	9	167	'EXPAND'	'10505051'	172	119.1111	98	60.4651	Inf
1068	'10.0.3.2'	'10.0.4.2'	72935	1068	9	168	'EXPAND'	'10909392'	172	119.1111	98	60.4651	Inf
1069	'10.0.0.1'	'10.0.1.5'	72980	1069	9	169	'EXPAND'	'10303031'	172	119.1111	98	60.4651	Inf
1070	'10.0.0.1'	'10.0.1.5'	72999	1070	9	170	'EXPAND'	'10909091'	172	119.1111	98	60.4651	Inf
1071	'10.0.2.6'	'10.0.1.3'	73030	1071	9	171	'EXPAND'	'10202226'	172	119.1111	98	60.4651	Inf
1072	'10.0.0.1'	'10.0.1.5'	73080	1072	9	172	'EXPAND'	'10505051'	172	119.1111	98	60.4651	Inf
1073	'10.0.0.1'	'10.0.1.5'	73167	1073	10	1	'CONTRACT'	'10101011'	42	111.4000	Inf	Inf	Inf
1074	'10.0.0.1'	'10.0.1.5'	73277	1074	10	2	'CONTRACT'	'10303031'	42	111.4000	Inf	Inf	Inf
1075	'10.0.0.1'	'10.0.1.5'	73300	1075	10	3	'CONTRACT'	'10505051'	42	111.4000	Inf	Inf	Inf
1076	'10.0.0.1'	'10.0.1.5'	73348	1076	10	4	'CONTRACT'	'10404041'	42	111.4000	Inf	Inf	Inf
1077	'10.0.3.9'	'10.0.5.4'	73457	1077	10	5	'CONTRACT'	'10707379'	42	111.4000	Inf	Inf	Inf
1078	'10.0.5.4'	'10.0.1.8'	73562	1078	10	6	'CONTRACT'	'10202524'	42	111.4000	Inf	Inf	Inf

Fig. 2 statistics collected with the simulator

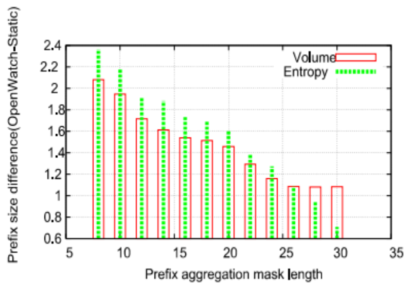


Fig. 3.1 Prefix size difference [5]

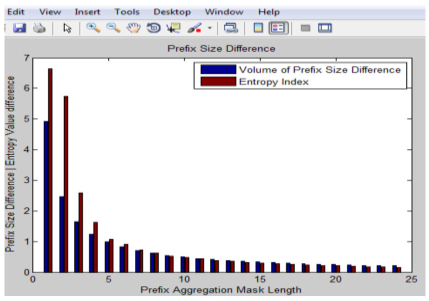


Fig. 3.2 Prefix size difference at ST=120s

Also a bar graph for delay in detection of anomalies with respect to the reporting interval is generated. Delay in detection of anomalies also kept on decreasing as with the growth of time interval volume also increases. With initial larger volume of data larger delay in detection of anomalies must be present and then delay in detection of anomalies will be reduced as per learning will be done according to detection and it will take lesser time in detection. Similarly the obtained pattern of detection delay bar graph is shown in figure 4.2 using our simulator also comes out similar corresponding to the existing graph shown in figure 4.1 [5].

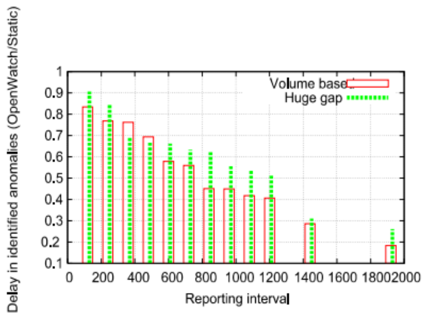


Fig. 4.1 Delay in detection [5]

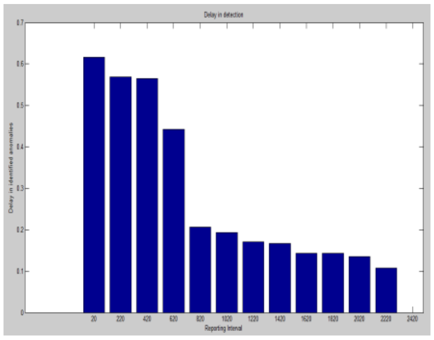


Fig. 4.2 Delay in detection

Finally in figure 5 we have shown the fraction of anomalies detected with respect to prefix aggregation mask length with a line graph. Figure 5.1 shows the statistics collected using openWatch [5] and in figure 5.2 a line graph generated using our simulator which shows the increase in the identification of anomalies as the prefix aggregation mask length increases.

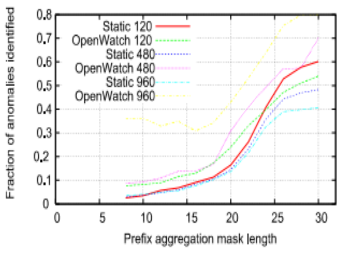


Fig. 5.1 Anomalies found with detector [5]

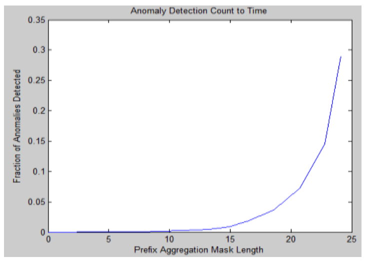


Fig. 5.2 Anomalies found with our simulator

VI. Conclusion And Future Work

Monitoring of network traffic is key for security of our network. Security of software defined networking is also a fundamental requirement for its deployment. SDN centralized control facilitates us for easy monitoring of network traffic and assist to be able to detect any sort of intrusion or anomalies in network. Heavy monitoring of traffic can also cause monitoring overhead over the controller which in turn results in poor performance. Hence a mechanism is required to maintain a proper balance between the monitoring overhead and performance of the network. Adaptive rule-updation algorithm using dynamic flow counting can able to maintain such balance by reducing the number of monitoring rules dynamically in case of heavy and legitimate traffic which in turns reduces the monitoring overhead. This given algorithm further improves the existing dynamic rule updation algorithm by reducing its time complexity and still obtaining almost similar performance results. In this work, resulting graphs are generated from the run time captured data using our simulator. Comparison of results with respect to existing results shows that flow of our graphs are similar for different data sets. In future, we consider the further improvement of this algorithm using certain parameters like reducing the size of mask length so that less computation will be made on intermediate routers and processing speed can be increased. Also values of R_L and R_U can be made dynamic according to bandwidth available for new connections so that anomalies can be detected more accurately.

Acknowledgment

I would like to give my sincere gratitude to all the friends and colleagues who are helping me to conduct this research, without which this research would be incomplete.

References

[1] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran,S. Guizani, “Securing the Software Defined Networks: Taxonomy, Requirements, and Open Issues”, *IEEE Communication Magazine*, Dec. 2014.

[2] Betts, S. Fratini, N. Davis, R. Dolin and others, *SDN Architecture*, Open Networking Foundation ONF SDN ARCH 1.0 06062014, Issue 1, June, 2014.

[3] G. Garg and R. Garg, *Review on architecture and security issues in SDN*, International Journal of Innovative Research in Computer and Communication Engineering Vol. 2, Issue 11, pp.6519-6524, November 2014.

[4] S. Shi, G. Gun, “Attacking Software-Defined Networks: A First Feasibility Study”**,** *ACM Proc. of HotSDN*, Hong Kong, China, 2013, pp. 165-166.

[5] Y. Zhang, “An adaptive flow counting method for anomaly detection in SDN”, *ACM Proc. of CoNEXT*, Santa Barbara, California, USA, December, 2013, pp. 25-30.

[6] G. Roopali, and S. Himika, *Proposed Lightweight Sybil Attack Detection Technique in MANET*, International journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Issue 5, May 2014.

[7] J. Mai, A. Sridharan, C. N. Chuah, H. Zang, and T. Ye, *Impact of Packet Sampling on Portscan Detection*, Selected Areas in Communications, IEEE Journal, Vol. 24. Issue: 12, pp: 2285 – 2298, December, 2006.

[8] P. Banford, J. Kline, D. Plonka, and A. Ron, “A signal analysis of network traffic anomalies”, *ACM Proc. of SIGCOMM IMW*’02, 2002, pp.71-82.

[9] A. Lakhina, M. Crovella, and C. Diot, “Mining anomalies using traffic feature distributions”, *ACM Proc. of SIGCOMM*, Philadeiphia Pennsylvania USA, August, 2005, pp217-228.

[10] K. Giotis, G. Androulidakis, and V. Maglaris, “Leveraging SDN for efficient anomaly detection and mitigation on legacy networks” *Proc. of EWSDN*, Budapest, Hungary, September, 2013.

[11] S. A. Mehdi, J. Khalid and S. A. Khayam, *Revisiting Traffic Anomaly Detection using Software Defined Networking*, Springer Recent Advances in Intrusion Detection, 2011.

[12] M. Moshref, M. Yu, and R. Govindan, “Resource/Accuracy Tradeoffs in Software-Defined Measurement”, *ACM Proc. of HotSDN’13*, Hong Kong, China, August 2013, pp.73-78.

[13] G. Garg and R. Garg “Detecting anomalies efficiently in SDN using adaptive mechanism”, in *IEEE ACCT2015 Rohtak*, INDIA, Feb. 2015.

[14] W. Lu and A. A. Ghorbani, *Network Anomaly Detection Based on Wavelet Analysis*, EURASIP Journal on Advances in Signal Processing, Vol. 2009, Article ID 837601, 2009.