Egyptian Informatics Journal 22 (2021) 85–90

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

A hybrid entropy-based DoS attacks detection system for software defined networks (SDN): A proposed trust mechanism

Nada M. AbdelAzim [a,b], Sherif F. Fahmy [b,*], Mohammed Ali Sobh [a], Ayman M. Bahaa Eldin [c]

[a] Faculty of Engineering, Ain Shams University, Cairo, Egypt
[b] College of Engineering, Arab Academy for Science, Technology and Maritime Transport, Sheraton, Cairo, Egypt
[c] Misr International University, on leave from Aim Shams University, Egypt

ARTICLE INFO

Article history:
Received 16 October 2019
Revised 24 March 2020
Accepted 27 April 2020
Available online 13 May 2020

Keywords:
Denial of service attacks
Entropy
Software defined networks
Stochastic distance metrics

ABSTRACT

Software defined networks are an emerging category of networks in which the data plane and control plane are separated. This separation of planes opens the door for designing sophisticated routing algorithms that would overwhelm the computing power of traditional networking nodes. In this paper, we consider the possibility of introducing node trust into the routing problem. There are many ways for measuring node trust. However, in this paper, we focus on denial of service attacks. We develop a hybrid method for detecting denial of service attacks and incorporate this information in routing decisions so that nodes that are part of a botnet can be quickly identified and excluded from the network. The proposed method is flexible enough to allow nodes that have been suspected of participating in a denial of service attack to be "rehabilitated" if they cease their malicious behavior. The technique is also able to detect the start of a second attack while another one is on-going. Our results indicate that the proposed method for detecting denial of service attacks performs better than non-hybrid techniques.

1. Introduction

Software defined networks are a relatively new development [1,2] in the field of networking that allow the data plane and the control plane to be decoupled – essentially, this means that the decision about the routing paths to be taken is performed on a machine that is separate from the one that actually routes the data packets.

This has many benefits, the two most important of which are the greater flexibility this provides when deploying a network and the fact that it is now possible to use a powerful machine to make the routing decisions, thus allowing more complex algorithms to be developed. The latter benefit, making the use of more complex algorithms possible, opens the door for designing algorithms that take factors other than simple routing cost into account. In this paper, we design one such algorithm that attempts to take node trust into account when coming up with routing decisions.

There are many different factors that can affect the level of trust of nodes in a network, in this paper, we tackle the issue of trust in a network whose members may or may not be part of a denial of service botnet. Toward that end, we design an algorithm that would first determine whether or not nodes are participating in denial of service attacks. We then use this information to come up with a trust value that we assign to nodes. The algorithm then *bans* the offending nodes for a certain period of time and notifies their administrator, thus, giving them a chance to rehabilitate. If malicious behavior is repeated, the nodes are banned indefinitely.

The rest of the paper is organized as follows, Section 2 reviews the literature, Section 3 contains a description of the proposed DoS detection solution, Section 4 contains a brief theoretical analysis of the proposed solution, Sections 5 and 6 contain the experimental setup and results, Section 7 contains a description of the proposed trust mechanism and, finally, Section 8 concludes this paper.

* Corresponding author.
E-mail addresses: nadamostafa@aast.edu (N.M. AbdelAzim), fahmy@aast.edu (S.F. Fahmy), Mohamed.sobh@eng.asu.edu.eg (M.A. Sobh), Ayman.bahaa@eng.asu.edu.eg (A.M. Bahaa Eldin).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

https://doi.org/10.1016/j.eij.2020.04.005
1110-8665/

2. Literature survey

Software defined networks are an emerging category of networks that present a new set of challenges from the security perspective. The authors in [3] provide a comprehensive overview of the security threats faced by this emerging architecture. Of these challenges, this paper focuses on denial of service attacks and how information about them can be incorporated into a trust mechanism that can inform routing decisions that occur on the network.

There are many papers that address the issue of detecting denial of service attacks. Some are based on entropy [4,5], while others use more advanced entropy based metrics to detect DoS attacks [6,7]. It should be noted that while entropy is a good metric for detecting DoS and DDoS attacks, it is possible to spoof it [8] and it is prone to false positives when the underlying behavior of the network changes [6,7].

To mitigate this, anomaly detection algorithms that take into account the underlying behavior of normal traffic have been developed [6,7]. Such techniques use several entropy-based distance metrics, including KL-divergence [9].

However, to the best of our knowledge, no work has addressed the possibility of designing a hybrid system that uses both entropy and KL-divergence to detect DoS attacks in SDNs. Also, to the best of our knowledge, there is no work that targets the need to detect a denial of service attack while another one is on-going. In this paper, we design a hybrid algorithm that merges entropy and kl-divergence to address these issues.

There are also many papers that attempt to establish trust in a network, a seminal work on establishing trust in a distributed peer to peer network is presented in [10] where the notion of **eigentrust** is presented. This work was then used in many other works [11–14] – for example, in [14] it is used for establishing trust in the TOR network. The core idea of this paper is that if nodes in a distributed system have a partial view of node behavior – i.e., most nodes interact with each other and hence have an idea about the *goodness of behavior* of other nodes based on this, necessarily limited, interaction, it is possible to aggregate this into a system-wide notion of trust.

The algorithm design is orthogonal to the metric used to judge the *goodness of behavior* of each node, thus making this technique applicable to a very wide range of scenarios. However, the underlying assumption is that all nodes interact with each other to some degree. While this assumption makes sense in, for example, distributed peer-to-peer file sharing systems [15] and onion routing [16], it doesn't hold for normal traffic patterns on a SDN.

Thus, eigentrust cannot be used as a trust mechanism for a general SDN. Fortunately, there is a node, the controller, which does have a centralized view of the traffic behavior in a SDN, and it is on this node that the algorithm presented in this paper is implemented.

Other distributed trust calculation mechanisms exist, for example, there is a whole category of agent-based algorithms that use software agents to establish trust in a distributed manner [17–19]. Like eigentrust, these approaches are useful when there is distributed trust information that needs to be used to generate a system-wide trust value for all or some nodes in the network without overloading the network with traffic.

3. Proposed DoS detection system

The proposed system relies on the fact that the source and destination IP frequency distributions of a network change when its under attack. Thus, we can detect an attack when we detect a change in either of these two distributions. We use two different techniques to detect denial of service attacks, each of which makes different assumptions about the statistical properties of the underlying network. Sections 3.1 and 3.2 contain a description of these techniques.

3.1. Entropy-based solution

The first technique relies on the entropy of source and destination IPs. The assumption is that the distribution of the frequency of occurrence of both destination and source IPs follows a uniform distribution or something very close to it when no attack is occurring. This implies that if we calculate the entropy of such a system, it is expected to be large because the distribution of IP counts is expected to be fairly random when no attack is underway. Note that the assumption of a uniform distribution for traffic in a network is common in the literature, e.g., [20]. In addition, we make this assumption because we assume that all nodes in a network are equally likely to communicate with each other – thus, the distribution of traffic in uniformly distributed.

When an attack does occur, this underlying uniform distribution is perturbed. In the case of a DoS attack, it is expected that the frequency of occurrence of the IP address of both the attacker and the node being attacked will spike, causing a significant deviation from the assumed uniform distribution of the network. Thus, the randomness of both the source and destination IP counts will decrease, leading to a drop in the entropy of both.

Thus, for the first proposed system, we detect a DoS attack as occurring when the entropy of both the source and destination IP addresses drops below a certain level. This threshold can be statically or dynamically determined. The assumption is that before the attacks occur, the entropy of the system is large because nodes communicate with each other in a random fashion. If this underlying assumption is violated, the technique is expected to detect many false positives.

For example, assume that one of the nodes transitions from normal internet browsing to, for example, streaming an online movie. This is not an attack, but is expected to cause a rise in both the destination and source IP counts that may lead to a drop in entropy in both, triggering a suspicion of an attack. Having a dynamic threshold may help resolve this, but what is really needed is a model of *normal* traffic in the system. Any deviation from this *normal* model would then be considered anomalous and hence an attack.

Since it is the distribution of the source and IP addresses that change during an attack, it is desirable to build a model of these two values that can be used as a reference point to check against. In addition, such a model should be dynamic in the sense that it allows itself to be updated as the usage of the network changes. Toward that end, we design another method for detecting DoS that takes into account past behavior of the system. This second model is described in the next subsection.

Before explaining the second method, we present the method used to calculate the entropy of IP addresses in the system. The algorithm is round-based, with each round spanning a certain duration of time. During each round, statistics about the frequency of occurrence of each IP is collected. At the end of the round, the probability of occurrence of each IP can easily be calculated by dividing the frequency of occurrenc of each IP by the total number of IPs sent in the same slot. Let us designate this probability as p_i. Given this value, it is possible to calculate the entropy, $H(x)$, of slot x using the standard equation of entropy

$$H(x) = -\sum_i p_i \times \log(p_i)$$

where the summation occurs over all the IPs in the system.

3.2. Kullback–Leibler divergence-based solution

This second model assumes that the network behavior is modeled over different time intervals, and if the behavior of the network changes from one time interval to the other, an ongoing attack is suspected. This is a reasonable assumption since network traffic will typically follow a specific pattern until an attack occurs to disrupt the normal pattern [20]. The tool used to model network behavior in any one time interval is the probability density function (PDF) of the occurrence of both the source and destination IPs. Thus, an interval is represented by two PDFs. It is fairly easy to program the controller in an SDN network to count the occurrences of destination and source IPs and create PDFs based on these numbers.

A method for computing the distance between the PDFs of any two time intervals is then needed in order to determine whether or not the *normal* behavior of the network has changed. We decided to use the Kullback-Leibler divergence [9] (or KL-divergence for short) to measure the distance between the PDFs of two successive time intervals. KL-divergence is not strictly a distance metric, it is not symmetric and does not obey the triangle property, but, for the purpose of our system, it suffices to compute the difference between the current and past source and destination PDFs.

In addition to being able to automatically adjust to changes in normal traffic in the system by modelling the underlying probability distribution of traffic flow, this method is also capable of detecting the start of a new attack while another attack is on-going. Using entropy alone for this may not yield the best result since the entropy will already be low during an attack. It is expected to become higher[1] when another attack occurs, but not by a significant amount – making it difficult to identify the beginning of a new attack.

However, if we use KL-divergence to measure the difference between probability distributions, the start of the first attack will be marked by a spike in the KL-divergence, as the algorithm measures a significant change in the probability distribution of normal traffic that transitions to the probability distribution of "attack traffic" – if we may coin a phrase.

This spike will vanish during the attack, since the distribution of the attack traffic is expected to remain unchanged during this time period. If, however, another attack occurs, the probability distribution of the underlying traffic is expected to change again, thus, leading to another spike in the KL-divergence.

Thus, it is possible for this technique to detect an attack that occurs while another is on-going more clearly than when entropy is used alone. It should be noted that the KL-divergence will also spike at the end of an attack as the underyling distribution of traffic will change in this case as well – from that representing the attack pattern to the normal pattern of network traffic.

We now present the equation for calculating the KL-divergence between two time slots. Assume that we have two PDFs for the two time slots under consideration. Let us designate these two PDFs as P and Q. The KL-divergence of Q from P, $D_{KL}(P||Q)$, can be calculated as follows:

$$D_{KL} = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

It is possible to combine both entropy and KL-divergence to come up with an accurate picture of what is happening in the network. Algorithm 1 depicts this idea in algorithmic form.

[1] This is further elaborated on in the experimental results section.

Algorithm 1 State Identification

```
1: procedure IdentifyState(En, KL)
2:   if En < th1 and KL > th2 then
3:     This is a start of an attack
4:   else if En > th1 and KL > th2 then
5:     This is end of an attack
6:   else if there is a slight rise in En, but KL > th2 then
7:     Another attack has started while the another is
     on-going
8:   else if En remains unchanged or drops slightly, but
     KL > th2 then
9:     One or more attacks are ending
10:  else
11:    No change has occurred from last status
```

In Algorithm 1, En represents the entropy of the source (or destination) IP, and KL represents the KL-divergence of two PDFs; one representing the distribution of IPs in the current time slot, and the other representing the same distribution for the previous time slot.

Both th_1 and th_2 are empirically determined thresholds that demarcate the boundary of these two variables as measured in the normal case and in the presence of an attack. Lines 2–3 and 4–5 detect the start and end of a single attack, as can be seen, when the entropy drops and the KL-divergence spikes, this is the start of an attack. The entropy drops because the uniformity of the PDF of IP occurrence is perturbed by a large spike in the source and destination IPs of the nodes involved in the attack. The KL-divergence rises because the underlying PDF is now different from the PDF of normal traffic.

Lines 6–7 and 8–9 handle the case of an attack occurring while another is in progress. This part of the algorithm requires some further explanation. When the first attack occurs in a network, the entropy drops – because what was once a uniform distribution now contains a spike that makes the occurrence of a certain IP (the IP involved in the attack) much more likely than the occurrence of any other IP. In addition, the KL-divergence spikes since the underlying distribution has changed. While the attack is ongoing, the KL-divergence goes back to normal (this technique only detects changes in the distribution of traffic and it is expected that the distribution will remain the same during the attack). The entropy will remain low during this time.

However, when a new attack occurs while the first has not terminated, two things happen. First, the traffic distribution changes which leads to a rise in the KL-divergence. This rise is not expected to be very large, but it is expected to be recognizable. It is not very large because the distribution is largely the same – the only addition is another spike at the location of the nodes involved in the new attack. Second, and this is the interesting part, the entropy is expected to rise. This occurs because we now have two nodes that should be equally probable (given the assumption that both attacks are of the same intensity), so the near certainty of the identity of the most frequent node that is associated with one attack (and that leads to a drop in entropy when one attack occurs) is now divided among two nodes – thus increasing surprise and hence entropy.

There is, thus, going to be a rise in entropy. But this rise will not be equal to the entropy before any attack occurs, because then, given the assumption of a uniform distribution of traffic, all nodes are equally likely to appear in traffic – thus the distribution is completely random, leading to large values for entropy. The larger the number of attacks occurring in the system, the larger the expected rise in entropy that will occur. This is why lines 6–7 detect a sec-

ond attack by measuring a small rise in entropy and a rise in the KL-divergence.

Lines 8–9 are the opposite, when an attack ends but others are still on-going, the entropy will only drop slightly because the degree of predictability introduced by the remaining attacks is still high. However, the KL-divergence is expected to rise as a spike is removed from the underlying PDF of traffic distribution when an attack goes away.

4. Theoretical discussion

In this section of the paper, we present some theoretical discussions pertinent to the developed algorithms. Please note that this is not a formal analysis, rather it is a semi-formal discussion that allows us to clarify our reasoning about the proposed algorithm.

We shall consider two attack models in the work, the first is one in which the attacker has full knowledge of the network and the other is in which the attacker has zero knowledge of the network. The rest of this section discusses how the proposed algorithms, and the algorithms in the literature, compare to each other.

First, we shall discuss the ability of the proposed systems to withstand attacks according to the two attack models under consideration. Then, we shall discuss the ability of our proposed algorithms to detect concurrent attacks.

4.1. Ability to withstand spoofing attacks

4.1.1. Full knowledge

In this model, we assume that the attacker has full information about the traffic patterns inside a network. Given this information, an attacker can spoof its attack packets so that they mimic normal network traffic. It was shown in [8], that if such knowledge exists, it is possible to circumvent entropy-based anomaly detection systems for DoS attacks. The attacker can generate enough packets to mask the attack or to generate enough false positives to make the anomaly detection unusable.

By monitoring the statistics of the underlying traffic in the system, it is possible for a spoofing attack to mimic normal traffic statistics and so render a system that depends solely on entropy unreliable. However, using the KL-divergence method, the attacker would not only need to record the statistics of the underlying traffic, but would actually need to build an actual model of the traffic itself – this is due to the fact that KL-divergence methods reply on the actual probability density function of traffic, rather than summary statistics.

Since it is much harder to spoof traffic to make it follow the exact PDF of normal traffic, we believe that KL-divergence methods would be more difficult to spoof compared to methods that simply relied on entropy. Spoofing summary statistics is easier than reproducing an exact PDF of normal traffic while performing an attack.

The hybrid technique proposed in this work will exhibit the robustness derived from its KL-divergence component to such spoofing attacks.

4.1.2. Zero knowledge

In this model, we assume that the attacker has no knowledge about the underlying network and is incapable of collecting enough data to form a comprehensive picture of the normal traffic patterns in the network. For such an attack scenario, it is not possible to spoof either the purely entropy-based system or the KL-divergence system.

4.2. Concurrent attacks

A key difference between the proposed hybrid system and others systems in the literature is the ability of our system to detect concurrent attacks. Systems that solely depend on entropy are not capable of detecting concurrent attacks, because they are actually expected to increase entropy a bit – a new attack will cause a second spike in the distribution of the IP addresses, thus making it more "surprising" than the single spike of a single attack scenario.

By combing both entropy and KL-divergence, we are able to both detect concurrent attacks and to mark their end as well. A detail description of this is presented in Section 6.

5. Experimental setup for DoS detection

In this section of the paper, we describe the experimental setup used to test the performance of the proposed DoS detection mechanism. The topology used to test the idea is a fully connected SDN that contains a total of 20 nodes. There is only one controller in this topology.

The controller is implemented using ryu [21]. We divide the duration of the experiment into equal time slots, during which each node communicates with a uniformly distributed random subset of other nodes. We perform two experiments in this section of the paper, the first is to see whether the proposed combined method can be used to accurately determine the occurrence of a DoS attack. The second is to determine whether the proposed method is better than using entropy only to detect the start of another DoS attack, when a previous DoS attack is on-going.

In order to test these two ideas, we design two different attack scenarios. In the first, a denial of service attack is initiated and stopped between a fixed source and destination node at uniformly distributed random times. The proposed algorithm is then tested on this topology to see if it can detect the attack. It should be noted that the random time at which an attack occurs or stops is not aligned to the beginning or end of the time slots of the experiment – this is inline with reality, attacks by malicious nodes are not typically aligned with time slots designed by system administrators to prevent them.

In the second scenario, two denial of service attacks occur at overlapping times. The purpose of this second scenario is to see whether or not the proposed technique outperforms the use of entropy alone to detect a DoS attack that begins while another attack is on-going.

6. Experimental results for DoS detection

When the first attack scenario was run, the result in Fig. 1 was obtained. We annotate the beginning and end of each attack on the figure. As can be seen, when the entropy drops and the KL-divergence spikes, we detect the beginning of an attack. When the entropy rises, and the KL-divergence spikes, we detect the end of an attack.

The entropy alone would not be enough as it would be very sensitive to the threshold chosen to detect the attack, and the kl-divergence alone would not be enough as it would not be able to differentiate between the start and end of an attack. The two combined produce more robust results. It should also be noted that when another attack occurs while another is on-going, it is expected that the entropy will increase slightly. This occurs because before the second attack, the distribution was very predictable, being dominated by the source and destination nodes involved in the first attack. When the second attack occurs, the entropy will increase slightly as this domination is now challenged

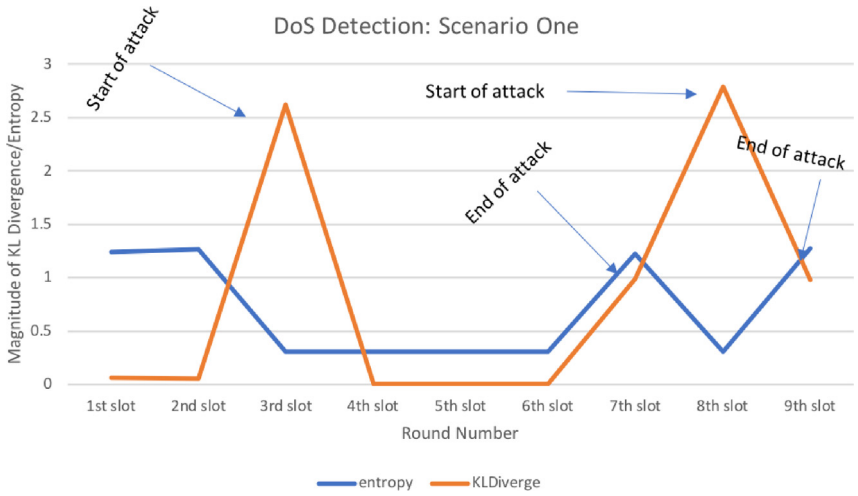

Fig. 1. DoS Detection: Scenario One.

by the traffic of the new pair of nodes involved in the second attack – i.e., the distribution now becomes less predictable and more "surprising", hence the rise in entropy.

Fig. 2 depicts the results of the second experiment conducted. A number of interesting results appear. When the second attack, attack 2, starts, two things happen: the entropy rises a little bit, and the KL-divergence also rises. This happens because, as mentioned above, the "surprising" nature of the traffic increases when another peak occurs in the source and destination PDFs around the nodes involved in the second attack. In addition, the KL-divergence increases because the probability distribution has now changed, containing two peaks instead of one.

Another interesting observation, is that when both attach 1 and 2 end in the 7th time slot, the entropy does not rise. This is probably due to the fact that the attacks end during the time slot, and at different times, this results in there being at least one attack ongoing at some point during the time slot, resulting in the low entropy, but the spike in the KL-divergence gives away the fact that the underlying distribution has changed – and hence the attacks have ended. As can be seen from Fig. 2 it would be difficult to detect the start and end of the second attack using entropy alone, thus the proposed hybrid method provides better results in this case.

7. Proposed trust framework

Now that we have designed a hybrid DoS detection system that provides superior performance to previous techniques, we shall discuss the proposed trust framework that will utilize this information to modify routing decisions in the network.

While designing the trust framework, we noted that a node may be an unwitting member of a botnet, and by banning it from the network for a given period of time – and possibly providing feedback to its administrators to take corrective actions – it can be rehabilitated into the network.

Since the proposed algorithm determines whether a node is malicious or not at the end of each round of a round-based algorithm, we decided to keep track of the number of rounds during which the node exhibits malicious behavior. Our algorithm works over a window of five rounds, and measures the number of rounds within this window that the node exhibits malicious behavior. It then calculates the percentage of the five slots during which the node was identified as malicious. This percentage is compared to two thresholds th_3 and th_4, these thresholds are the low and high marks of suspected malicious behavior respectively. They are both percentages, and indicate the number of times that a node was suspected as malicious during a test window.

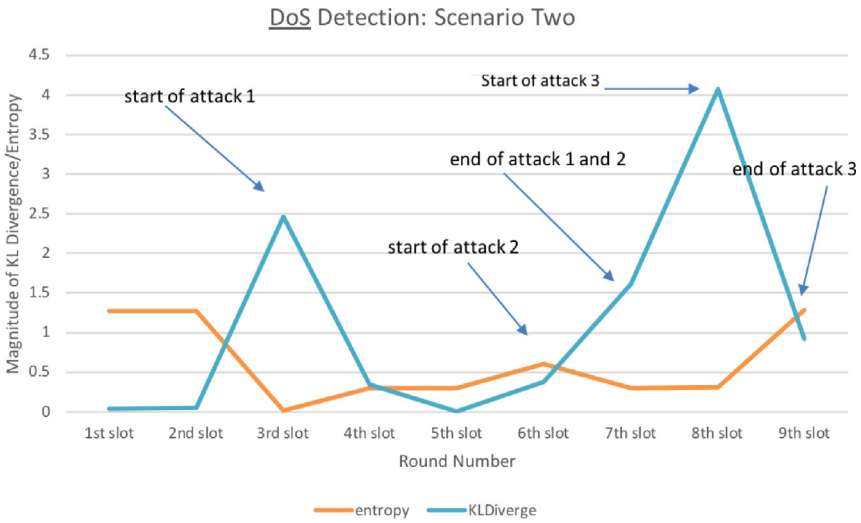

Fig. 2. DoS Detection: Scenario Two.

Therefore, the algorithm tracks the behavior of the node, if it is malicious for more than th_3%, but less than th_4% of the time after the first five time slots, it is banned for n time slots and a warning message is sent to its administrator.

After this time period, its allowed back on the network. If its behavior remains unchanged –i.e., its malicious between th_3% and th_4% of the time – then it is banned for another $n \times 2$ period of time. This continues exponentially until an administrator-set number of times, after which it is banned from the network.

Any node that is malicious for more than th_4% is immediately banned from the network. The rationale for coming up with the proposed solution is to give a node that is only "mildly misbehaving" a chance to rehabilitate itself. The number of chances given to the node is administrator controlled. If the node continues to misbehave after this period of grace, it is banned from the network. However, a node that significantly misbehaves is banned immediately.

This is a very general framework, and does not depend on the metric used to measure trust. Any value can be used as long as a decision about whether or not a node is being malicious during a time slot can be derived from the metric. It is this binary decision, malicious or not malicious, that is used as an input to this framework.

8. Conclusion and future work

This paper proposed a hybrid method for denial of service attack detection in software defined networks. Taking advantage of the increased computing power of the controller in such networks, the denial of service attack detection algorithm is deployed on the controller.

The proposed hybrid algorithm is more robust as it takes the nature of existing network traffic into account and can adapt in real-time as the nature of the traffic in a network changes. We also show that the hybrid methods is better at detecting concurrency denial of service attacks.

In addition to proposing this hybrid algorithm, we propose a trust framework for software defined networks that can be used to ban nodes that are behaving maliciously after giving them a chance for rehabilitation. The proposed framework is independent of the metric used to measure trust and so can be used with other techniques for measuring trust.

In the future, we plan to implement the proposed trust framework in a software defined network and measure its performance. We would also like to expand the study to include handling distributed denial of service attacks (DDoS) and to consider the use of machine learning in determining the thresholds used in the algorithm – they are currently set by experimentation.

We would also like to study this problem in greater detail to see if it is possible to come up with a distributed denial of service attack detection mechanism that does not rely on the centralized controller as this is a single point of failure. We would also like to consider studying the behavior of the algorithm under the full-knowledge attacker model and come up with a way to make it more robust to such attacks.

References

[1] Benzekki K, El Fergougui A, Elbelrhiti Elalaoui A. Software-defined networking (SDN): a survey. Secur Commun Networks 2016;9(18):5803–33. doi: https://doi.org/10.1002/Section 1737.

[2] Mousa M, Bahaa-ElDin AM, Sobh M. Software Defined Networking concepts and challenges. In: Proceedings of 2016 11th International Conference on Computer Engineering and Systems, ICCES, 2016. p. 79–90. doi: https://doi.org/10.1109/ICCES.2016.7821979.

[3] Elazim NMA, Sobh MA, Bahaa-Eldin AM. Software Defined Networking: Attacks and Countermeasures. In: Proceedings – 2018 13th International Conference on Computer Engineering and Systems, ICCES 2018. Institute of Electrical and Electronics Engineers Inc.; 2019. p. 555–67. doi: https://doi.org/10.1109/ICCES.2018.8639429.

[4] Mousavi SM, St-Hilaire M. Early detection of DDoS attacks against SDN controllers. In: International Conference on Computing, Networking and Communications, ICNC, 2015. p. 77–81. doi: https://doi.org/10.1109/ICCNC.2015.7069319.

[5] Singh KJ, Thongam K, De T. Entropy-based application layer DDoS attack detection using artificial neural networks. Entropy 2016;18(10):350. doi: https://doi.org/10.3390/e18100350.

[6] David J, Thomas C. Efficient DDoS flood attack detection using dynamic thresholding on flow-based network traffic. Comput Secur 2019;82:284–95. doi: https://doi.org/10.1016/j.cose.2019.01.002.

[7] Vitali D, Villani A, Spognardi A, Battistoni R, Mancini LV. DDoS Detection with Information Theory Metrics and Netflows – A Real Case. In: International Conference on Security and Cryptography (2012)., no. January 2014; 2012. p. 172–181.https://doi.org/10.5220/0004064501720181.

[8] Özçelik I, Brooks RR. Deceiving entropy based DoS detection. Comput Secur 2014;48:234–45. doi: https://doi.org/10.1016/j.cose.2014.10.013.

[9] Kullback S, Leibler RA. On Information and Sufficiency. Ann Math Stat 1951;22 (1):79–86. doi: https://doi.org/10.1214/aoms/1177729694.

[10] Kamvar SD, Schlosser MT, Garcia-Molina H. The EigenTrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th International Conference on World Wide Web, WWW 2003 (2003) 640–651. https://doi.org/10.1145/775152.775242.

[11] Abrams Z, McGrew R, Plotkin S. Keeping peers honest in EigenTrust. In: Proceedings of the 2nd Workshop on the Economics of Peer-to-Peer Systems Citeseer.

[12] Abrams Z, Mcgrew R, Plotkin S. A non-manipulable trust system based on eigentrust. ACM SIGecom Exchanges 2005;5(4):21–30.

[13] Li K, He Y, Liu X, Wang Y. Security-driven scheduling algorithms based on eigentrust in grid. In: Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05). IEEE; 2005. p. 1068–72.

[14] Aziem NMA, Fahmy SF, Hashad AI. Aggregating local metrics for global trust calculations in the TOR network. ICCTA; 2014.

[15] Gummadi KP, Dunn RJ, Saroiu S, Gribble SD, Levy HM, Zahorjan J. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. ACM SIGOPS Operating Systems Review 2003;37(5):314–29. doi: https://doi.org/10.1145/1165389.945475.

[16] Goldschlag D, Reed M, Syverson P. Onion routing for anonymous and private communications. Commun ACM 1999;42(2):39–41. URL: https://apps.dtic.mil/docs/citations/ADA465075.

[17] Abdel-Halim IT, Fahmy HMA, Bahaa-Eldin AM. Agent-based trusted on-demand routing protocol for mobile ad-hoc networks. Wireless Netw 2015;21(2):467–83. doi: https://doi.org/10.1007/s11276-014-0793-z.

[18] JingQiu, Liao LJ. Add semantic role to dependency structure language model for topic detection and tracking. In: Proceedings – SNPD 2007: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, vol. 3, 2007, pp. 517–521. https://doi.org/10.1109/SNPD.2007.122.

[19] Maximilien EM, Singh MP. Agent-based trust model involving multiple qualities. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. ACM; 2005. p. 519–26.

[20] Gallos LK, Korczyński M, Fefferman NH. Anomaly detection through information sharing under different topologies. Eurasip J Inf Secur 5. https://doi.org/10.1186/s13635-017-0056-5.

[21] Asadollahi S, Goswami B, Sameer M. Ryu controller's scalability experiment on software defined networks. In: 2018 IEEE International Conference on Current Trends in Advanced Computing, ICCTAC 2018, Institute of Electrical and Electronics Engineers Inc.; 2018, pp. 1–5.https://doi.org/10.1109/ICCTAC.2018.8370397.