Probabilistic Transition-Based Approach for Detecting Application-Layer DDoS Attacks in Encrypted Software-Defined Networks

Elena Ivannikova(✉), Mikhail Zolotukhin, and Timo Hämäläinen

Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35, (Agora), 40014 Jyväskylä, Finland
{elena.v.ivannikova,mikhail.zolotukhin,timo.hamalainen}@jyu.fi

Abstract. With the emergence of cloud computing, many attacks, including Distributed Denial-of-Service (DDoS) attacks, have changed their direction towards cloud environment. In particular, DDoS attacks have changed in scale, methods, and targets and become more complex by using advantages provided by cloud computing. Modern cloud computing environments can benefit from moving towards Software-Defined Networking (SDN) technology, which allows network engineers and administrators to respond quickly to the changing business requirements. In this paper, we propose an approach for detecting application-layer DDoS attacks in cloud environment with SDN. The algorithm is applied to statistics extracted from network flows and, therefore, is suitable for detecting attacks that utilize encrypted protocols. The proposed detection approach is comprised of the extraction of normal user behavior patterns and detection of anomalies that significantly deviate from these patterns. The algorithm is evaluated using DDoS detection system prototype. Simulation results show that intermediate application-layer DDoS attacks can be properly detected, while the number of false alarms remains low.

Keywords: DDoS attack · Anomaly detection · SDN · Clustering · Behavior pattern · Probabilistic model

1 Introduction

Distributed Denial-of-Service (DDoS) is a coordinated attack which by using multiple hosts prevents legitimate users from accessing a specific network resource, e.g. email, websites, online banking, etc. In DDoS attack, by taking control of the computer and sending a stream of packets an attacker may perform attacks to other computers by sending spam messages or huge amount of data to a website. The target server, overloaded with requests, either becomes very slow even unusable or totally crashes since it can only process a certain number of requests at once. Thus, the server becomes unavailable to the legitimate clients. Another way of the attack is sending malformed packets that cause

Z. Yan et al. (Eds.): NSS 2017, LNCS 10394, pp. 531–543, 2017.
DOI: 10.1007/978-3-319-64701-2_40

the target machine to freeze or reboot [15]. There are many other ways to deny services on the Internet [21]. DDoS attacks have become a major threat to the stability in modern high-speed networks [19]. Being hard to detect and abort in a timely fashion, these attacks can be used to disable strategic business, government, media and public utility sites prompting victims to loose productivity, revenue and reputation.

Traditional DDoS attacks are carried out at the network layer. Among them are volume-based attacks (e.g. UDP floods, ICMP floods, etc.) and protocol attacks (e.g. SYN floods, Smurf DDoS, etc.). Volume-based attacks attempt to consume the bandwidth either within the target network/service, or between the target network/service and the rest of the Internet, when protocol attacks attempt to consume actual server or intermediate communication equipment resources, such as firewalls and load balancer. Recently, these types of attacks have been well studied and various schemes for protecting network against such attacks have been reported [2,8,14]. Application-layer attack is a more advanced attack which targets vulnerabilities in operative systems and web applications. These attacks can be performed by seemingly innocent and legitimate requests from only a few attacking machines generating low traffic rate, which makes them difficult to detect and mitigate.

One of the most frequent application-layer DDoS attacks nowadays are attacks that involve the use of HTTP protocol. These attacks can be grouped into three major categories, depending on the level of sophistication [21]. *Trivial attacks*, where each bot sends a limited number of unrelated HTTP attacks towards the target site, comprise the majority of application-level DDoS attacks on the Internet. In *intermediate attacks* bots continuously generate random sequences of browser-like requests of web pages with all embedded content. Such procedure allows the attack traffic fitting better in regular human requests. *Advanced attacks* consist of a carefully chosen sequence of HTTP requests in order to better mimic the browsing behavior of regular human visitors. Advanced DDoS attacks are believed to raise popularity in the future [21].

Defending against a trivial HTTP attack does not require a complex detection system. Trivial attack can be detected by inspecting each request to determine if it comes from a legitimate user. Intermediate and advanced attacks, however, require more sophisticated techniques [21]. To name a few, paper [24] analyses intermediate application-layer DDoS attacks by defining a model of normal user behavior via a number of clustering techniques and comparing conversations against such normal patterns. Xu *et al.* [23] model user browsing behavior by random walk graph and identify attackers based on analysis of their page-request sequences. Paper [3] proposes a new clustering algorithm against HTTP-GET attacks using entropy-based clustering and Bayes factor analysis for classification of legitimate sessions. Most of the current studies devoted to HTTP-based DDoS attack detection focus on un-encrypted HTTP traffic. Nowadays many DDoS attacks are utilizing secure protocols for data encryption in the application layer of network connections making their detection more difficult. In this work, we concentrate on intermediate attacks in encrypted traffic.

Recently, cloud computing has become a strong contender to traditional on-premise implementations. The main reason is that cloud environments offer advantages such as on-demand resource availability, pay as you go billing, better hardware utilization, no in-house depreciation losses, and, no maintenance overhead [20]. Cloud resources are provided to the customers in the form of virtual machines (VMs). Cloud service provider has to guarantee the security of the machines by filtering unwanted traffic from other cloud customer networks and external hosts. Despite willing to be secured against attacks, cloud customers may wish to remain their traffic un-encrypted. Thus, the cloud service provider has to detect attacks without relying on encrypted packet payload. With the emergence of cloud computing, many attacks, including DDoS attacks, have changed their direction towards cloud environment. In particular, DDoS attacks have changed in scale, methods, and targets and become more complex by using advantages provided by cloud computing.

Modern cloud computing environments can benefit from moving towards Software-Defined Networking (SDN) technology. In SDN, the control logic is separated from individual forwarding devices, such as routers and switches, and implemented in a logically centralized controller. This allows the network control to be programmable and the underlying infrastructure to be abstracted for applications and network services. As a result, SDN allows network engineers and administrators to respond quickly to the changing business requirements by shaping traffic from the central controller without having to touch the physical switches. They use software to prioritize, redirect or block traffic either globally or in varying degrees down to individual packet levels.

There have been a number of works related to detection of network-based DDoS attacks in SDN. Phan *et al.* [18] introduce a hybrid approach based on combination of SVM and SOM [6] for flow classification in network traffic. Another work [22] suggests an attack detection system based on Bloom Filter and SDN to handle the link flooding attacks. In [11] a method based on SDN to detect DDoS attacks initiated by a larger number of bots for solving server attacks is proposed. The method uses the standard OpenFlow APIs designed for operation in general SDN environments. Other approaches related to detection of network-based DDoS attacks in SDN using machine learning techniques are described in [1,4,10]. To the best of our knowledge, there are only a few studies that try to detect application-based DDoS attacks in cloud environments with the help of SDN. Mohammadi *et al.* [16] present a software defined solution named Completely Automated DDoS Attack Mitigation Platform (CAAMP). When suspicious traffic is detected, CAAMP stores a copy of the original application on a private cloud and redirects suspicious traffic there. Thus, more time can be spent for processing suspicious traffic with no extra costs.

The aim of our research is to provide efficient and proactive solution for detecting application-layer DDoS attacks in cloud environment with the help of SDN. We propose a detection approach which is comprised of extracting normal user behavior patterns and detecting anomalies that significantly deviate from these patterns. This allows detection of attacks from legitimately connected

network machines that are accomplished by using legitimate requests. Due to operating with information extracted from packet headers, the proposed scheme can be applied in secure protocols that encrypt the data of network connections without its decrypting. In order to evaluate our scheme, we implement a DDoS detection system prototype that employs the proposed algorithm. Simulation results show that intermediate application-layer DDoS attacks can be properly detected, while the number of false alarms remains very low. Finally, not only do we provide solution for detecting application-layer DDoS attacks in SDN-driven cloud environments, but also enhance the detection algorithm proposed in previous work [25]. These enhancements include:

1. Improved performance scores (FPR, TPR, accuracy).
2. Reduced number of parameters to effectively just one, which is the cluster number parameter in the first phase training when using k-Means. The second training phase is essentially parameterless.
3. Significant reducing the amount of storage needed by the detection algorithm. That is we only need to store centroids from the clustering phase and transition/marginal probability matrices from the second phase for each sequence length plus thresholds that are found automatically. Thus, the storage complexity is quadratic in the number of possible clusters $O(k^2)$, while it was at least $O(k^4)$.

The rest of the paper is organized as follows. Section 2 briefly describes the experiment setup. Section 3 summarizes main concepts and provides theoretical background of the proposed approach. Section 4 describes the algorithm proposed in the paper. Section 4.1 explains feature extraction process, while training and detection procedures are clarified in Sects. 4.2, 4.3 and 4.4. Meanwhile, Sect. 5 is devoted to the experimental results. It describes simulation environment, data set and results of the performance tests. Finally, Sect. 6 concludes the paper and outlines future work.

2 Problem Formulation

We consider a cloud environment in which cloud customers are allowed to create private virtual networks and connect them to the existing public networks with the help of virtual routers. In addition, every customer can spawn several virtual instances in own virtual networks. Each customer operates inside one of the projects created by a system administrator for a particular set of user accounts. We assume that neither user or administrator accounts have been compromised.

Further, we assume that networking inside the cloud is carried out with the help of SDN that includes an SDN controller and several SDN forwarding devices that are designed for working with virtual instances. SDN controller and switches communicate between each other inside the cloud management network and are not available directly from the data center VMs or external hosts. Scenarios in which either the controller or one of the switches is compromised are out of scope of this paper.

We consider a cloud customer that deploys several virtual web servers inside a virtual network providing access for other cloud customers as well as external hosts. Communication between the web servers and the users is carried out with encrypted traffic. Even though the web service provider relies on the data center security defenses, it cannot allow the cloud security engineers to decrypt the network traffic since it would violate regulations on privacy along with a high risk of conflict with the web service users. For this reason, detection of DDoS attacks is assumed to be carried out on network flow level.

In this study, we assume that network flows are captured on each SDN forwarding device and sent to the controller with the help of a NetFlow or sFlow agent. The controller investigates the received flow statistics and discovers behavior patterns of normal users. Once discovered, normal behavior patterns can be used to detect DDoS attacks against the web server applications and to block traffic from malicious cloud customers or external attackers in online mode.

3 Theoretical Background

3.1 *k*-Means-Based Clustering

k-Means [12,13] is one of the most popular algorithms for cluster analysis. It aims at partitioning data points into k clusters with the parameter k fixed a priory. Given a set of points $\chi = (\mathrm{x}_1, ..., \mathrm{x}_n)$, $\mathrm{x}_i \in \mathbb{R}^m$ the algorithm starts with initializing k centroids, one for each cluster, and assigning each data point x_i to the nearest centroid. Then iteratively the algorithm recalculates the centroids and re-assigns the data points to new clusters until convergence of the algorithm. Specifically, the algorithm aims at minimizing the sum of Euclidean distances between each data point and the mean value of the cluster this point belongs to, or to find

$$\arg\min_{\mathrm{C}} \sum_{i=1}^{k} \sum_{\mathrm{x} \in \mathrm{C}_i} \|\mathrm{x} - \mu_i\|^2,$$

where $\mathrm{C} = \{\mathrm{C}_1, ..., \mathrm{C}_k\}$ are data partitions and $\mu = (\mu_1, ..., \mu_k)$ are corresponding centroids.

3.2 CURE-Based Clustering

Despite traditional clustering methods have been widely used in data analysis, they have a number of drawbacks. For example, centroid-based methods, including *k*-Means, use only one point (centroid) to represent a cluster. If a cluster is large or has an arbitrary shape, the centroids of its subclusters can be distant from each other that could cause unnecessary splitting. On the opposite edge of the spectrum, all points-based methods such as *k*-NN or kernel, use all points for cluster representation and are sensitive to outliers and even slight changes in the position of data points. Both approaches fail to work well for defining non-spherical or arbitrarily shaped clusters [5].

Clustering Using REpresentatives (CURE) [5] is a hierarchical clustering algorithm which is a compromise between centroid-based and all point-based approaches and is suitable for large scale data sets. Compared to traditional methods, this approach is less sensitive to outliers and defines well even non-spherical clusters. First, initial clusters are created by hierarchical clustering of randomly picked sample points. Next, k scattered points describing a cluster shape and extent are picked, as disperse as possible. After shrinking towards the cluster centroid by a fixed fraction α these points become representatives of the cluster. When representative points are set up for each of the initial clusters the whole data set is rescanned and each point is assigned to the closest cluster. In traditional version of CURE the closest cluster for a point is defined as the closest one among all representative points of all the clusters. We modify the original procedure of cluster assignment as follows. After clusters are found, we take all representatives and centroids and continue using them as if they were an output of a centroid-based clustering algorithm, i.e., each centroid and/or representative is thought to be a center of a cluster. Such gradation of clusters allows better capturing complexity of user behavior types.

3.3 Probabilistic Transition-Based Approach for Detecting DDoS Attacks

Let $\mathbb{C} = \{c_i | i = 1, ..., K\}$ be a set of labels. Given a sequence of labels $c = (c_1, ..., c_N) \in \mathbb{C}^N$, let $P(c_i|c_{i-1}, l = N)$ denote conditional probability of observing label c_i after c_{i-1} in a sequence of length N. Marginal probability of observing label c_i at the beginning of the sequence is denoted as $P(c_i|l = N)$. We factorize joint probability distribution over sequences of length N as the following product:

$$P(c_1, ..., c_N|l = N) = P(c_1|l = N) \times \prod_{i=2}^{N} P(c_i|c_{i-1}, l = N), \tag{1}$$

where l denotes length of the sequence. We estimate $P(c_i|c_{i-1}, l = N)$ as

$$P(c_i|c_{i-1}, l = N) \stackrel{\wedge}{=} \frac{\mathrm{n}(c_{i-1}, c_i, N)}{\mathrm{n}(c_{i-1}, N)}, \tag{2}$$

where $\mathrm{n}(c_{i-1}, c_i, N)$ denotes count of observations of pairs (c_{i-1}, c_i) in all sequences of length N over all time windows and sessions, $\mathrm{n}(c_{i-1}, N)$ denotes count of observations of label c_{i-1} in all sequences of length N over all time windows and sessions. Moreover, $P(c_i|l = N)$ is estimated as

$$P(c_i|l = N) \stackrel{\wedge}{=} \frac{\mathrm{n}(c_i, N)}{\sum_{j=1}^{K} \mathrm{n}(c_j, N)}. \tag{3}$$

Note, that in (3) we use the fact that the marginal probability of observing a label in a sequence should be equal to the marginal probability of observing the label at the beginning of a sequence since the windows are sliced arbitrarily.

During training phase we estimate conditional and marginal probabilities according to (2)–(3). Moreover, for every length of sequence N that is present in the training data we calculate minimal joint probabilities δ_N, which are further used as thresholds to examine new data for anomalies during test phase.

During test phase, we first calculate joint probability of a sequence of length N according to (1) and then compare it against a corresponding threshold value δ_N. If the sequence satisfies $P(c_1, ..., c_N | l = N) < \delta_N$ it is marked anomalous.

4 Algorithm

4.1 Feature Extraction

To detect outliers, we build a normal user behavior model. The features for building this model are extracted from a portion of network traffic at a very short time window that allows timely detection of attacks. The presented approach is based on the analysis of network traffic flows, namely, groups of IP packets with some common properties passing a monitoring point at a specified time interval. This time interval is defined to be equal to the time window. For analysis, we consider traffic flow extracted from the current time window. Furthermore, to reduce amount of data to be analyzed, we utilize aggregated traffic information by taking into account all packets of the flow transferred during previous time windows.

Next, we re-construct client to server conversations by combining the flow pairs such that the source socket of one flow equals to the destination socket of the other flow and vice versa. A conversation can be characterized by source IP, address, source port, destination IP address and destination port. For each such conversation, we extract the following information at every time interval:

1. Duration of the conversation.
2. Number of packets sent in 1 second.
3. Number of bytes sent in 1 second.
4. Average packet size.
5. Presence of packets with different TCP flags: URG, ACK, PSH, RST, SYN and FIN.

The set of features is defined by existing protocols for collecting IP traffic information such as NetFlow and sFlow. Since the values of the extracted feature vectors can have different scales, we standardize them using min-max normalization [6] by scaling to a range [0,1].

4.2 Training

We perform training using the standardized extracted features described in Sect. 4.1. First, we apply a clustering algorithm to divide the features into distinct groups representing specific classes of traffic in the network system. Thus, the algorithm discovers hidden patterns in the dataset. We assume that the traffic being clustered is mostly legitimate despite the fact it can be encrypted.

Therefore, we state that the obtained clusters describe behavior of normal users. Second, we group together conversations with the same source IP address, destination IP address and destination port extracted at a certain time interval. Such groups serve as an approximation of a user session and are analyzed separately, as other studies propose [3,23,24]. Next, we represent each session in every time window by a sequence of cluster labels obtained at the first step. Finally, from the obtained sequences we estimate conditional and marginal probabilities $P(c_i|c_{i-1}, l = N)$, $P(c_i|l = N)$ according to (2)–(3). For every sequence we calculate its probability using estimated parameters and model (1). In addition, we calculate thresholds δ_N by finding minimum among all sequence probabilities for a particular length of a sequence N.

4.3 Online Training Procedure

As behavioral patterns of users can change over time, we need to adapt our models in real-time. For adapting the clustering phase model we can use streaming k-Means algorithm. After clustering and classification have been done for a particular window t, one can update cluster centroids using the following formula:

$$\mu_i^{t+1} = \mu_i^t \cdot \delta + \sum_{\substack{\mathrm{x} \in \mathrm{C}_i^t, \\ \mathrm{x} \in \chi_{normal}}} \mathrm{x} \cdot (1 - \delta),$$

where μ_i^t is centroid of the cluster i at the time window t, C_i^t is the set of data points assigned to the cluster i, χ_{normal} is a set of data points classified as normal, and $\delta \in [0, 1]$ is a constant reflecting how fast the model has changed when a new observation emerged, i.e. for bigger δ the model changes slower. For CURE the same formula can be used, but representatives are updated instead of the cluster centroids.

To update transition probabilities from the probabilistic model dynamically, we apply the following updates that are performed for each pair of labels (c_{i-1}, c_i) from the label sequences that were classified as normal:

$$P(c_i|C_{i-1} = c_{i-1}, l = N) \leftarrow P(c_i|C_{i-1} = c_{i-1}, l = N) + \epsilon,$$
$$P(c_i|C_{i-1} \neq c_{i-1}, l = N) \leftarrow P(c_i|C_{i-1} \neq c_{i-1}, l = N) - \epsilon/(K - 1),$$

where C_{i-1} is a random variable that denotes cluster label at position $i - 1$ and ϵ represents the velocity of change of a conditional probability once a new evidence has been observed. Thus, ϵ affects how fast model is changed with respect to new data. These updates guarantee that the conditional probability remains properly normalized by adding a probability mass to the parameter that accounts for the new data and removing the same amount of probability mass from the parameters that do not correspond to the new data. Moreover, these updates implement a forgetting mechanism as the old evidence gets less and less influence on the model with time.

In order to keep thresholds δ_N up to date we propose to store top N_δ data sequences in a heap data structure with keys equal to probabilities of

the data sequences. We need to keep a separate heap for label sequences of each length. Every time the model is updated the top element with the lowest probability (equal to the current threshold δ_N) is popped out and pushed in the heap again with a new recomputed probability key. Moreover, the threshold δ_N is assigned the new value. This way threshold can either become bigger or smaller. Threshold value is also updated once a new normal data sequence gets smaller probability under the current model.

4.4 Detection

For detecting anomalies we use a model of normal user behavior obtained during training phase. First, we assign each session with a sequence of cluster numbers using clustering model from the training phase. Then, similarly to the training phase, we calculate probability of every sequence using estimated probability parameters and the model (1). The obtained probability values are compared against thresholds δ_N to decide whether the sequence is anomalous or not. If probability of a sequence is less than a threshold probability then it is marked as anomaly.

5 Algorithm Performance

5.1 Simulation Environment and Data Set

We test the attack detection algorithm proposed in this study in a virtual network environment that includes a small botnet, command and control center (C2) and a target web bank server (see Fig. 1). The target server is running in the Openstack [7] cloud environment where networks are carried out by an Opendaylight [9] integrated SDN controller and several Open vSwitches [17]. Bots and C2 are located outside the cloud. Each bot is a VM with running a special program implemented in Java, it receives commands from C2 and generates some traffic to the server. It is worth noting that all the traffic is transfered by using encrypted SSL/TLS protocol. All network flows are captured on SDN switches and sent to the controller with the help of NetFlow agents.

In order to generate a normal bank user traffic, we specify several scenarios that each bot follows when using the bank site. Each such scenario consists of several actions following each other. The list of the actions consists of logging in to the system by using the corresponding user account, checking the account balance, transferring some money to another account, checking the result of the transaction, logging out of the system, and some other actions. Each action corresponds to requesting a certain page of the bank service with all of its embedded content. Pauses between two adjacent actions are selected in a way similar to a human user behavior. For example, checking the account balance usually takes only a couple of seconds, whereas filling in information to transfer money to another account may take much longer time.

In addition to the normal traffic, we perform an intermediate DDoS attack during which several bots-attackers try to mimic the browsing behavior of regular

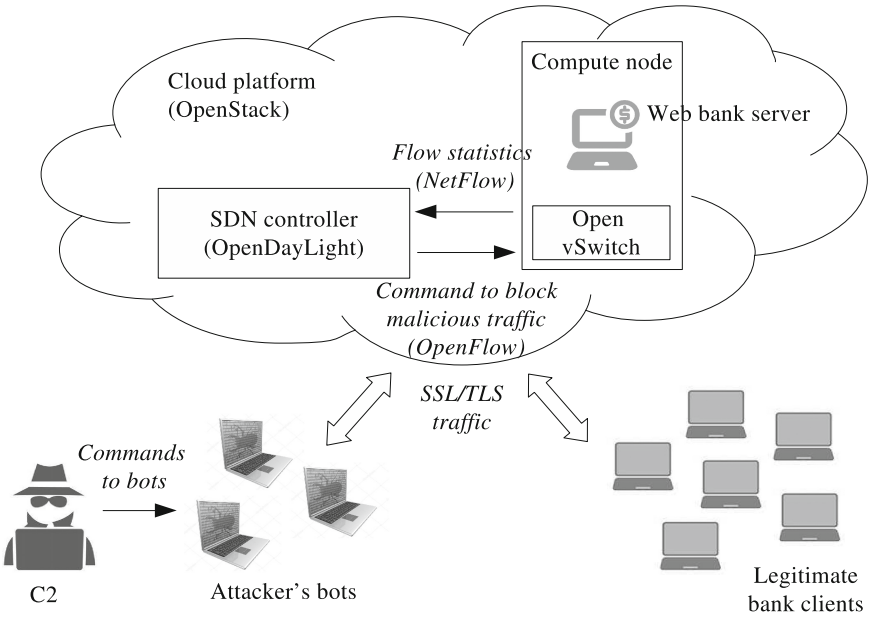

Fig. 1. Virtual network simulation environment.

users by requesting sequences of web pages with all embedded content from the service. However, unlike the normal user behavior, these sequences are not related to each other by any logic but generated randomly. We consider the case when the attacker sends traffic with about the same rate as normal users, and each attacker's connection individually looks like normal. More advanced attack scenarios are left for future works.

5.2 Results

We evaluate the proposed approach on the test set described in Sect. 5.1. We propose two methods for detecting intermediate DDoS attacks which both consist of two phases. The first method (k-Means+Prob) uses k-Means clustering in the first phase and probabilistic transition-based approach (Prob) in the second phase. The second method (CURE+Prob) applies CURE clustering in the first phase and Prob in the second phase. The algorithms have been evaluated using the detection accuracy, true positive rate (TPR) and false positive rate (FPR) performance metrics [6].

In our experiments, the time window size is set to 5 seconds, due to the nature of the data. Moreover, we are only interested in results when FPR is below 1% as the high number of false alarms is one of the most important known drawbacks of anomaly-based detection systems.

Table 1 displays accuracy of detecting intermediate DDoS attacks for the proposed detection schemes. For comparison, we also include to Table 1 performance

Table 1. Accuracy of detecting intermediate DDoS attacks

Algorithm	TPR (%)	FPR (%)	Accuracy (%)
k-Means+Prob	98.66	0	99.58
CURE+Prob	95.65	0	86.16
2-gram k-Means	95.08	0.24	86.61
3-gram k-Means	91.21	0	73.66

results of the k-Means-based data stream clustering approach proposed in [25]. The parameters of the methods are selected to maximize the detection accuracy on validation set. The best result is shown by the (k-Means+Prob) approach which outperforms other methods by 13% in terms of accuracy. Still, other methods perform relatively well reaching accuracy of 86% with FPR equaling to or near zero.

To visualize the results, we plot ROC curves in Fig. 2(a). ROC curves corresponding to the (k-Means+Prob) and (CURE+Prob) methods proposed in this paper are displayed by dashed and plain lines, correspondingly. Furthermore, by dash-dot and dotted lines we plot ROC curves for the k-Means-based data stream clustering approach proposed in [25] for 2-gram and 3-gram models, respectively. From the ROC curves one can see that (k-Means+Prob) is the only among the presented algorithms that reaches TPR of 100%. Other methods demonstrate similar performance reaching the highest TPR of around 98% at FPR near 1.5%.

In addition, for the best performing algorithm (k-Means+Prob) we plot how performance scores depend on number of clusters, which is the only parameter of this method. Figure 2(b) shows that the algorithm performs relatively well for all parameter values reaching the maximum in accuracy and TPR when the number of clusters is equal to 12. FPR, which is plotted in (100%−FPR) scale for better visual representation, remains below 1% for all parameter values.

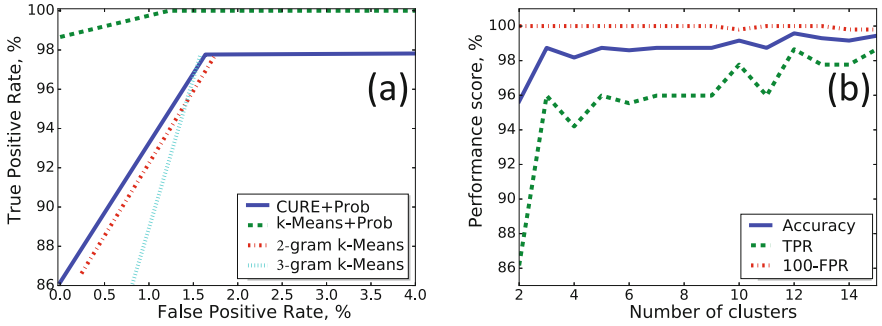

Fig. 2. (a) - ROC curves for detection of intermediate DDoS attacks, (b) - dependence of performance scores from number of clusters for the (k-Means+Prob) algorithm.

6 Conclusions and Future Work

In this work, we proposed probabilistic transition-based approach for detecting intermediate application-layer DDoS attacks in cloud environment with the use of SDN. Operating with information extracted from the packet headers makes this approach suitable for detecting DDoS attacks from encrypted traffic. We tested the proposed algorithms against other methods used for detecting application-layer DDoS attacks in encrypted networks proposed earlier in [25]. Both presented algorithms demonstrated good performance results. Moreover, (k-Means+Prob) significantly outperforms other evaluated algorithms under the condition of FPR $< 1\%$.

In the future, we plan to improve the algorithm in terms of the detection accuracy and test it with a bigger dataset. In addition, more focus will be on the simulation and detection of more advanced DDoS attacks.

Acknowledgment. This research was supported by the Nokia Foundation Scholarship funded by Nokia, Finland.

References

1. Chen, P.J., Chen, Y.W.: Implementation of SDN based network intrusion detection and prevention system. In: 2015 International Carnahan Conference on Security Technology (ICCST) (2015). https://doi.org/10.1109/CCST.2015.7389672
2. Chen, R., Wei, J.Y., Yu, H.F.: An improved grey self-organizing map based dos detection. In: IEEE Conference on Cybernetics and Intelligent Systems, pp. 497–502 (2008). https://doi.org/10.1109/ICCIS.2008.4670765
3. Chwalinski, P., Belavkin, R., Cheng, X.: Detection of application layer DDoS attacks with clustering and Bayes factors. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 156–161 (2013). https://doi.org/10.1109/SMC.2013.34
4. Dotcenko, S., Vladyko, A., Letenko, I.: A fuzzy logic-based information security management for software-defined networks. In: 16th ICACT, pp. 167–171 (2014). https://doi.org/10.1109/ICACT.2014.6778942
5. Guha, S., Rastogi, R., Shim, K.: Cure: an efficient clustering algorithm for large databases. Inf. Syst. **26**(1), 35–58 (2001). doi:10.1016/S0306-4379(01)00008-4
6. Hastie, T.J., Tibshirani, R.J., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer, New York (2009). doi:10.1007/978-0-387-84858-7
7. Jackson, K.: OpenStack Cloud Computing Cookbook. Packt Publishing, Birmingham (2012)
8. Ke-Xin, Y., Jian-qi, Z.: A novel dos detection mechanism. In: International Conference on Mechatronic Science, Electric Engineering and Computer (MEC), pp. 296–298 (2011). https://doi.org/10.1109/MEC.2011.6025459
9. Knorr, E.: Opendaylight: A big step toward the software-defined data center. InfoWorld (2013)
10. Le, A., Dinh, P., Le, H., Tran, N.C.: Flexible network-based intrusion detection and prevention system on software-defined networks. In: 2015 ACOMP, pp. 106–111 (2015). https://doi.org/10.1109/ACOMP.2015.19

11. Lim, S., Ha, J., Kim, H., Kim, Y., Yang, S.: A SDN-oriented DDoS blocking scheme for botnet-based attacks. In: 2014 6th International Conference on Ubiquitous and Future Networks (ICUFN), pp. 63–68 (2014). https://doi.org/10.1109/ICUFN.2014.6876752
12. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theor. **28**(2), 129–137 (2006). https://doi.org/10.1109/TIT.1982.1056489
13. Macqueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
14. Mills, K., Yuan, J.: Monitoring the macroscopic effect of DDoS flooding attacks. IEEE Trans. Dependable Secure Comput. **2**, 324–335 (2005). https://doi.org/10.1109/TDSC.2005.50
15. Mirkovic, J., Reiher, P.: A taxonomy of DDoS attack and DDoS defense mechanisms. SIGCOMM Comput. Commun. Rev. **34**(2), 39–53 (2004). http://doi.acm.org/10.1145/997150.997156
16. Mohammadi, N.B., Barna, C., Shtern, M., Khazaei, H., Litoiu, M.: CAAMP: completely automated DDoS attack mitigation platform in hybrid clouds. In: 12th International CNSM, pp. 136–143 (2016). https://doi.org/10.1109/CNSM.2016.7818409
17. Pfaff, B., Pettit, J., Koponen, T., Jackson, E.J., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P., Amidon, K., Casado, M.: The design and implementation of open vswitch. In: 12th USENIX Conference on Networked Systems Design and Implementation (NSDI), pp. 117–130 (2015)
18. Phan, T.V., Bao, N.K., Park, M.: A novel hybrid flow-based handler with DDoS attacks in software-defined networking. In: 2016 IEEE UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld (2016). https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0069
19. Radware: 2015–2016 global application & network security report. https://www.radware.com/newsevents/pressreleases/radwares-2015-2016-global-applications-and-network-security-report/
20. Somani, G., Gaur, M.S., Sanghi, D., Conti, M., Buyya, R.: DDoS attacks in cloud computing: issues, taxonomy, and future directions. ACM Comput. Surv. **1**(1), 1–44 (2015)
21. Stevanovic, D., Vlajic, N.: Next generation application-layer DDoS defences: applying the concepts of outlier detection in data streams with concept drift. In: 13th ICMLA, pp. 456–462 (2014). https://doi.org/10.1109/ICMLA.2014.80
22. Xiao, P., Li, Z., Qi, H., Qu, W., Yu, H.: An efficient DDoS detection with bloom filter in SDN. In: 2016 IEEE Trustcom/BigDataSE/ISPA, pp. 1–6 (2016). https://doi.org/10.1109/TrustCom.2016.0038
23. Xu, C., Zhao, G., Xie, G., Yu, S.: Detection on application layer DDoS using random walk model. In: IEEE International Conference on Communications (ICC), pp. 707–712 (2014). https://doi.org/10.1109/ICC.2014.6883402
24. Zolotukhin, M., Hämäläinen, T., Kokkonen, T., Siltanen, J.: Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic. In: 23rd ICT, pp. 1–6 (2016). https://doi.org/10.1109/ICT.2016.7500408
25. Zolotukhin, M., Kokkonen, T., Hämäläinen, T., Siltanen, J.: On application-layer DDoS attack detection in high-speed encrypted networks. Int. J. Digital Content Tech. Appl. **10**(5), 14–33 (2016)