Author(s): Zolotukhin, Mikhail; Ivannikova, Elena; Hämäläinen, Timo

Title: On Detection of Network-Based Co-residence Verification Attacks in SDN-Driven Clouds

Year: 2017

Version:

Please cite the original version:

Zolotukhin, M., Ivannikova, E., & Hämäläinen, T. (2017). On Detection of Network-Based Co-residence Verification Attacks in SDN-Driven Clouds. In O. Galinina, S. Andreev, S. Balandin, & Y. Koucheryavy (Eds.), NEW2AN 2017, ruSMART 2017, NsCC 2017 : Internet of Things, Smart Spaces, and Next Generation Networks and Systems (pp. 235-246). Springer International Publishing. Lecture Notes in Computer Science, 10531. https://doi.org/10.1007/978-3-319-67380-6_22

On Detection of Network-Based Co-Residence Verification Attacks in SDN-driven Clouds

Mikhail Zolotukhin, Elena Ivannikova, and Timo Hämäläinen

Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland
mikhail.zolotukhin@jyu.fi, elena.v.ivannikova@student.jyu.fi,
timo.hamalainen@jyu.fi

Abstract. Modern cloud environments allow users to consume computational and storage resources in the form of virtual machines. Even though machines running on the same cloud server are logically isolated from each other, a malicious customer can create various side channels to obtain sensitive information from co-located machines. In this study, we concentrate on timely detection of intentional co-residence attempts in cloud environments that utilize software-defined networking. SDN enables global visibility of the network state which allows the cloud provider to monitor and extract necessary information from each flow in every virtual network in online mode. We analyze the extracted statistics on different levels in order to find anomalous patterns. The detection results obtained show us that the co-residence verification attack can be detected with the methods that are usually employed for botnet analysis.

1 Introduction

Cloud environments allow users to consume computational and storage resources in an on-demand manner with low management overhead. As a rule, these resources are provided to the users in the form of virtual machines (VMs). In theory, VMs running on the same cloud server (i.e., co-resident VMs) are logically isolated from each other. However, in practice, a malicious customer can create various side channels to circumvent the logical isolation, and obtain sensitive information from co-resident VMs that may include machine workloads [1] and even cryptographic keys [2]. Unfortunately, a straightforward solution to this type of the attack that is eliminating all possible side channels [3] is not suitable for immediate deployment due to the required modifications to current cloud platforms [4]. For this reason, the problem of preventing malicious users from spawning their virtual instances on the same cloud server as the victim's VM is of greatest interest for modern cloud providers [2, 4] .

There are several approaches a malicious cloud customer can employ to intentionally co-locate his VMs with victim instances to run on the same physical cloud server. Probably the easiest and for this reason the most popular approach relies on networking. Co-residency verification can be carried out by measuring network round-trip time between pairs of VM instances [1], the number of network hops [5], or the network interference injected by the attacker [6].

In this study, we concentrate on the problem of timely detection of such malicious customers in cloud environments that utilize software-defined networking (SDN). SDN is a hardware independent next generation networking paradigm, which breaks the vertical integration in traditional networks to provide the flexibility to program the network through centralized network control. In SDN, the control logic is separated from individual forwarding devices, such as routers and switches, and implemented in a logically centralized controller. The separation of control and data planes in SDN enables the network control to be programmable and the underlying infrastructure to be abstracted for applications and network services. Moreover, SDN enhances network security with the centralized control of network behavior, global visibility of the network state and run-time manipulation of traffic forwarding rules. Updating security policies in SDN requires updating the security applications or adding security modules to the controller platform, rather than changing the hardware or updating its firmware [7]. For these reasons, cloud computing environments can benefit from moving towards SDN technology [8].

To the best of our knowledge, there are no any studies that try to detect intentional co-residence attempts in cloud environments with the help of SDN. However, there are various approaches for detecting different sorts of cyber attacks carried out in clouds that utilize software-defined networking. Study [9] addresses man-in-the-middle and denial of service attacks caused by address resolution protocol bug in cloud centers using SDN technology. Authors propose a detection algorithm which uses Bayesian formula to calculate the probability of a virtual instance being an attacker. In [10], a new framework for DDoS detection and mitigation using sFlow and OpenFlow is proposed. The mechanism first matches an incoming flow with a legitimate sample of traffic and then installs mitigation actions if a flow found is not lying in the bounds of legitimate traffic pattern.

The rest of the paper is organized as follows. The problem in more details is formulated in Section 2. Section 3 describes several approaches of co-resident verification attack detection. In Section 4, we evaluate the performance of the techniques proposed. Section 5 draws the conclusions and outlines future work.

2 Problem Formulation

In this section, first, we specify assumptions for the cloud environment in which we detect the attack. After that, the attack vector is described in more details. Finally, we outline our solution that relies on software-defined networking.

2.1 Cloud Environment

We consider a cloud environment that consists of one controller and several compute nodes. A cloud customer can create several virtual networks and connect them to the existing public external network with the help of virtual routers. In addition, the customer is allowed to spawn several virtual instances in his own

virtual networks. It is worth noting that each such instance is automatically assigned to a specific compute node according to some predefined allocation policy. For example, according to this policy, VMs can be concentrated to a number of compute nodes, in order to decrease the power consumption and maximize the utilization rate, or distributed across the whole data center, for the purpose of workload balance and higher reliability [4]. The allocation policy remains unknown to the customers.

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. Thus, the cloud service provider guarantees that customer networks in every project are isolated from the direct intrusion of other customers. Cloud customers can only access each other's services via external public networks. Further, we assume that the networking inside the cloud is carried out with the help of SDN. SDN controller and switches communicate between each other inside the cloud's management network and are not available directly from the data center's virtual machines or external hosts. Scenarios in which either the controller or one of the switches is compromised are out of scope of this paper.

2.2 Attack Vector

In such environment, network-based co-residence verification technique such as time-to-live (TTL) probing [5] is not working. TTL values of network packets from two VMs belonging to two different cloud customers and communicating via an external public network are reduced only by routers of this network. Since virtual SDN switches usually operate on a different layer of the ISO/OSI model, they do not alter the IP payload. Furthermore, the co-location verification technique based on measuring packet round-trip times (RTT) [1] cannot be used either. Since VMs that belong to different cloud customers can communicate only over an external network, the traffic between them goes through this network's routers. For this reason, packet round-trip time to the target VM will most likely be the same from a VM located on the same server and from a VM that is on another server of the cloud. However, for machines from the same virtual network, RTT between two co-located VMs is slightly less than between VMs located on different servers. Therefore, the attacker can use this approach if one machine from the target's network has been compromised, but, in this research, we do not take this scenario into consideration.

In this study, we focus on the process of co-residence verification that relies on the fact that co-resident VMs share the same physical network interfaces. This opens an explicitly communicative channel that can be used by the malicious customer for co-residency verification. Study [6] proposes the co-resident watermarking attack which involves launching several virtual machines and performing statistical side channel tests from them. One of these machines plays role of a master host whereas the rest are flooders and sinks. The attack begins when the master initiates a web session with the target instance. Systematically, the master iterates through its list of the flooders and commands them to start

injecting network activity into the outbound interface of their physical host machines by sending portions of traffic to sinks. In case one or several flooders are co-located with the target server, this activity creates delay in the legitimate server's flow initiated by the master.

The attacker can define whether one of his flooders is co-located with the target by measuring packet arrivals per interval over the length of the flow. After each measurement, the intervals are divided into two samples X_1 and X_0 based on the pre-negotiated co-resident activity respectively representing intervals when flooders were active and when they were not. In study [6], authors model packet arrivals by a Poisson distribution and employ the non-parametric Kolmogorov-Smirnov test for independence. The null hypothesis that the two samples are from the same distribution can be rejected with confidence α if it is bigger than α_{min} that is equal to:

$$\alpha_{min} = 2 \exp\left(-2 \frac{|X_0||X_1|}{|X_0| + |X_1|} sup(|F_1(X_1) - F_0(X_0)|)^2\right), \quad (1)$$

where $F_i(X_i)$ for $i = 0, 1$ is empirical cumulative distribution of X_i. In other words, the less α_{min} the more confident the attacker that his flooder is co-located with the target server.

2.3 SDN Solution

There are at least two mechanisms that can help cloud data centers operating in software-defined networks in mitigation of network-based co-residence attacks. First, SDN allows the cloud provider manipulate traffic forwarding rules on fly. In particular, this means that the provider can redirect traffic of customers' VMs located in the same virtual subnet flow through different virtual switches. As a result, the traffic between VMs located on the same compute node may go via a switch that is located on another node. This would make the co-location verification technique based on measuring RTT hard for the attacker to use, especially if the forwarding rules change over time.

Second, SDN enables global visibility of the network state. Each flow from every virtual network can be monitored and analyzed in online mode. In particular, network flows can be captured on each SDN forwarding device and sent to the controller with the help of some flow collector. Once these statistics have been analyzed by the controller, it commands the SDN forwarders to either block the traffic or reroute it to security middle boxes for deeper payload-based analysis.

In this study, we rely on this second advantage of SDN for timely detection of the co-residence verification attack described in the previous subsection. For this purpose, each SDN forwarding device sends to the controller statistics of all the network flows initiated by or directed to VMs located on the corresponding compute node. These statistics may include the flow source and destination IP address and port, time stamp, duration of the flow, number of packets and bytes sent and received, presence of packets with different flags for TCP flows, and probably several others. The controller investigates the flow statistics received

searching for anomalous behavior patterns of the cloud's virtual instances and external hosts. The further analysis of these patterns can help to detect potential attacks and timely mitigate them by blocking the traffic that belongs to malicious agents.

3 Co-residence Verification Detection

As can be seen from the attack vector description, the malicious cloud customer requires to spawn a large number of virtual instances on the cloud. Thus, one potential approach for co-residence verification detection would rely on the number of VMs created per a time interval by a particular customer. However, such approach cannot be scaled well on big modern cloud data centers, since there can be thousands of newly created instances that only run for a short period of time and belong to different user projects. For this reason, we try to detect malicious customers by collecting traffic in cloud's private and external public networks. The traffic is then analyzed in three different domains: flow, session and time.

3.1 Flow Domain

A flow is a group of IP packets with some common properties passing a monitoring point in a specified time interval. These common properties include transport protocol, the IP address and port of the source (client) and IP address and port of the destination (server). For each flow at each time interval, we may extract several features including the flow duration, average number of packets sent or received per second, average size of packet, and some other information. Once all relevant features have been extracted and standardized, the resulting feature vectors can be divided into several groups by applying a clustering algorithm. There are many different clustering algorithms which can be categorized based on the notation of a cluster. The most popular categories include centroid-based clustering algorithms, hierarchical clustering algorithms and density-based clustering algorithms. The flow clusters can be found during the training phase when there is only legitimate traffic in the cloud. During the detection phase, if a new feature vector does not belong to any of the clusters obtained, the corresponding flow is labeled as malicious.

3.2 Session Domain

In the next stage, we can analyze sequences of flows belonging to one user session. Such approach is often used for application-based DDoS attacks detection [11]. If packet's payload is encrypted and session ID cannot be extracted, we can group all flows which are extracted in certain time interval and have the same source IP address, destination IP address and destination port together and analyze each such group separately. We can interpret a group of such flows as a rough approximation of the user session [12].

One detection approach in the session domain is counting flows from each cluster defined by one of the clustering algorithms mentioned in the previous subsection initiated by each user during the session. If flows between a flooder and a sink are shaped to follow legitimate traffic patterns, the attacker would probably need to increase the number of flows sent by each flooder to increase delays for packets arriving at the master host, therefore, improving the co-residence verification rate. For this reason, the number of flows of at least one certain type initiated by the attacker should exceed the number of flows of this type during legitimate user sessions. This pattern can be detected with the help of one of the clustering algorithms specified above or any anomaly detection technique.

Another approach is to analyze conditional probabilities of observing a sequence of flows in a session of particular length. Given a sequence of flow labels $c_1, ..., c_N$ we can factorize joint probability distribution over sequences of length N as the following product:

$$P(c_1, ..., c_N|N) = \frac{\mathrm{n}(c_1, N)}{\sum_{j=1}^{k} \mathrm{n}(c_j, N)} \times \prod_{i=2}^{N} \frac{\mathrm{n}(c_{i-1}, c_i, N)}{\mathrm{n}(c_{i-1}, N)}, \tag{2}$$

where k is the number of flow clusters, and $\mathrm{n}(c_i, N)$ and $\mathrm{n}(c_{i-1}, c_i, N)$ denote respectively count of observations of label c_i and pairs (c_{i-1}, c_i) in all sequences of length N over all time intervals and sessions. Resulting joint probability values can be compared to each other in order to find anomalous sessions.

3.3 Time Domain

Finally, in order to detect the malicious customer, we can exploit the fact that the master's and flooder's network activity should be synchronized during the attack. Such approach is sometimes used for botnet detection [13]. Since virtual instances from the same project are able to communicate via the project's private network, it is very unlikely that there will be synchronized sessions between VMs over external public network.

In order to obtain feature vectors, we calculate Pearson correlation coefficient [14] between two different virtual instances:

$$\rho_{ij} = \frac{\sum_t (r_i(t) - \bar{r}_i)(r_j(t) - \bar{r}_j)}{\sqrt{\sum_t (r_i(t) - \bar{r}_i)^2} \sqrt{\sum_t (r_j(t) - \bar{r}_j)^2}}, \ \forall i \neq j, \tag{3}$$

where $r_i(t)$ is percentage of time interval t during which the i-th instance transmits some data and $\bar{r}_i$ is the average value of $r_i(t)$ over all time intervals by the moment the detection starts. Attacker's activity can be detected by searching for anomalously high values of ρ_{ij}, which correspond to the synchronized transmission of instances i and j.

4 Performance Evaluation

In order to evaluate the detection approach proposed, first, we briefly overview our virtual network environment used to generate network traffic. Then, we describe the attack implementation and analyze the attacker's strategy. Finally, we

present results of the detection of the attack with different approaches overviewed in this study and discuss the results obtained.

4.1 Test Environment

We test the attack detection algorithm proposed in this study in open-source software platform for cloud computing Openstack , networks in which are carried out with the help of integrated SDN controller Opendaylight and several Open vSwitches. The cloud consists of four nodes: three compute nodes and one control node that also has compute functionality. Open vSwitch is installed on each cloud's node whereas the SDN controller is located on the cloud's control node (see Figure 1).

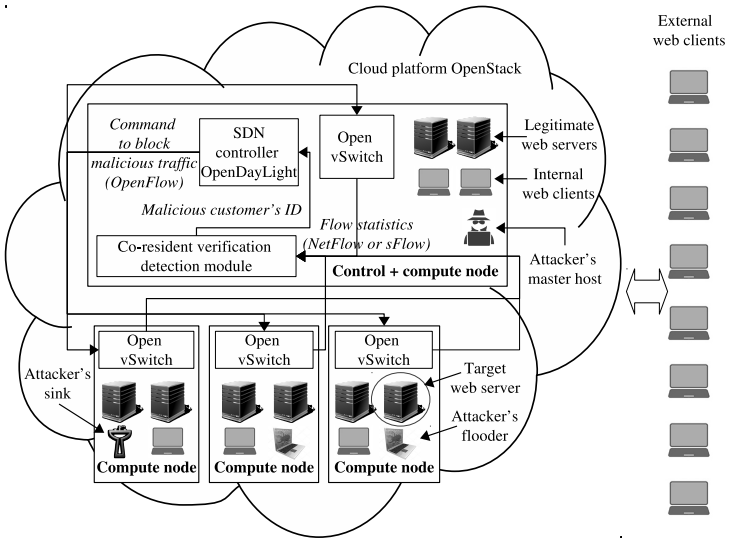


Fig. 1. Simulation environment for testing the detection algorithm.

In order to generate legitimate traffic, we spawn eight virtual web servers in the cloud and five instances that belong to different web clients. In addition, there are eight clients located outside of the cloud. Every arbitrary amount of time both internal and external clients connect to randomly selected servers and request several files from them. There are also keep-alive messages transferred time-to-time between servers and clients, and open SSH tunnels deployed in order to control VMs remotely. Statistics of all the resulting flows is recorded on switches and sent to the controller with the help of NetFlow and sFlow agents.

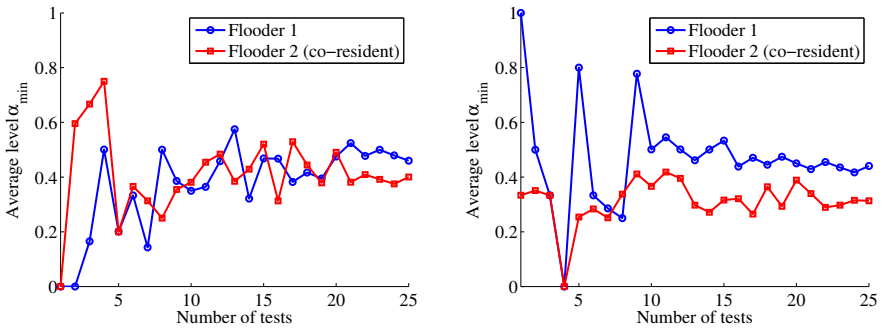
In addition to the normal traffic, we perform the co-residence verification attack against one of the web servers. For this reason, we spawn four virtual instances that belong to the attacker. These instances are supposed to be located

on different compute nodes. In practice, this can be achieved by spawning big amount of instances and removing those that are located on the same compute node. The fact that some of the attacker's instances are co-located can be checked by measuring packet RTT as these machines operate in the same virtual network. One of the resulting instances plays role of the master host, another one is the sink, whereas the rest are flooders.

4.2 Attack Analysis

During the attack, the master host first requests a file from the target server and measures packet arrivals per time interval. Next, the master commands one of the flooders to send portions of traffic to the sink, and at the same time requests one more file from the target server. After that, the procedure is repeated using another flooder. In order flooder's activity does not arouse suspicion from the cloud service provider, the traffic generated by the flooder is shaped in such a way that it mimics the traffic sent by legitimate web servers resided in the cloud. We test two co-resident attacks: low-rate and high-rate. In the low-rate attack scenario, each flooder generates the same amount of traffic flows as any of the legitimate servers, whereas during the high-rate verification the attacker increases the number of the flows generated by the flooders up to ten times.

The lowest value of minimal level α_{min} does not necessarily mean that the corresponding flooder resides on the same compute node as the target server. The reason behind this is that there is constant outbound network activity on the external network interface of every cloud's compute node which causes various delays in packet arrival times. These delays sometimes can be even bigger than delays caused by the attacker's flooders. For this reason, the attacker is supposed to run several tests for each flooder that is under his control.



(a) Low-rate co-residence verification (b) High-rate co-residence verification

Fig. 2. Dependence of minimal level of confidence α_{min} on the number of co-residence verification tests.

We assume that the attacker decides whether one of his flooder is co-resided with the target server based on the average value of minimal level of confidence α_{min} calculated during several co-residence verification tests. Figure 2 shows dependence of this average value on the number of the tests in case of the low-rate and the high-rate attack scenario. As one can see, the more tests the attacker conducts, the less the average value of α_{min} for the flooder resided on the same compute node as the target server and, therefore, the more confident the attacker is that the verification has been done properly. It is worth noting that, in the case of the high-rate co-residence verification, the difference between average values of α_{min} for the flooder that is co-resided with the target server and the flooder that is located on another cloud node can be clearly seen after few verification tests. Thus, an attacker that employs the high-rate attack requires less tests to guarantee that the verification is correct. However, the immediate drawback of the high-rate approach is the attack can be easily detected by analyzing the network traffic as shown in the next subsection.

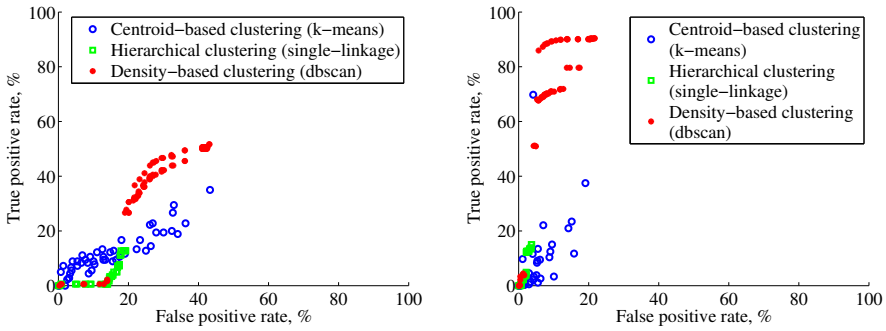
4.3 Attack Detection

First, we try to detect both versions of the attack in the flow domain. Figure 3 shows how TPR depends on FPR when detecting the attack for three different clustering approaches. We used k-means, single-linkage and dbscan as the most popular representatives of centroid-based, hierarchical and density-based clustering approaches respectively. As one can notice, all the methods fail to properly detect both the low-rate and the high-rate co-residence verification attack. In the case of low-rate verification tests, the percentage of false alarms is almost the same as the percentage of successfully detected malicious flows which results in very low accuracy values as can be seen from Table 1. The only approach that potentially can be used for the high-rate attack detection in the flow domain is density-based clustering. However, the number of false alarms when employing this approach is still high (up to 20 %) making it impractical in most of the use cases.

Table 1. Detection accuracy in the flow domain

Clustering approach	Attack type	
	Low-rate	High-rate
Centroid-based	47.21 – 73.74 %	24.56 – 78.72 %
Hierarchical	58.79 – 74.30 %	44.89 – 50.20 %
Density-based	44.69 – 74.31 %	45.30 – 86.11 %

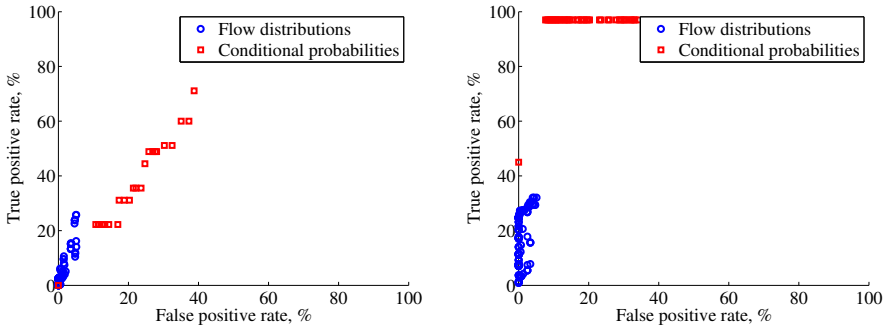
Next, we try to detect the attack in the session domain. Figure 4 shows how TPR depends on FPR when detecting the attack for two different approaches. As it can be noticed, the approach based on the analysis of flow distributions inside each session does not generate many false alarms, but at the same time it allows one to detect only 30% of attacker's sessions. On contrary, the approach based



(a) Low-rate attack detection (b) High-rate attack detection

Fig. 3. Dependence of true positive rate on false positive rate of the attack detection in the flow domain.

on calculating conditional probabilities generates lots of false alarms, however, most of the attacker's sessions can be detected in case of the high-rate attack. Detection accuracy in the both cases is still low as can be seen from Table 2.



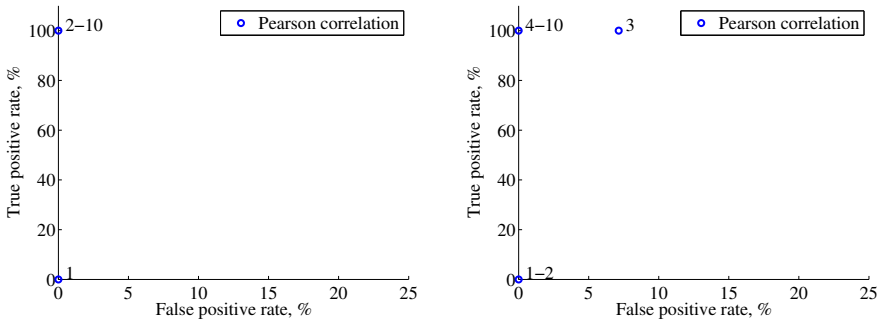
(a) Low-rate attack detection (b) High-rate attack detection

Fig. 4. Dependence of true positive rate on false positive rate of the attack detection in the session domain.

Table 2. Detection accuracy in the session domain

Detection approach	Attack type	
	Low-rate	High-rate
Flow distributions	82.44 – 86.04 %	84.44 – 89.09 %
Conditional probabilities	48.10 – 79.75 %	37.18 – 93.66 %

Finally, we detect both versions of the attack in the time domain by calculating Pearson correlation coefficients. Virtual instances with extremely high coefficient values are labeled as anomalous. Dependence of true positive rate on false positive rate of the detection is shown in Figure 5. Numbers near each point correspond to amounts of verification tests performed by the moment when the detection starts. As the number of tests increases, the detection accuracy for both versions of the attack approaches 100 % (TPR = 100 %, FPR = 0 %). Each iteration of the high-rate attack lasts longer, as a result, more legitimate VMs transmit at the same time as flooders, which may lead to few false alarms.



(a) Low-rate attack detection (b) High-rate attack detection

Fig. 5. Dependence of true positive rate on false positive rate of the attack detection in the time domain.

To summarize the results obtained, the high-rate version of co-residence verification attack can be detected on both flow and session level. This can be explained by the fact that in this case the attacker initiates many look-alike flows which results in density outliers in the space of feature vectors extracted from flows and anomalously high concentration of flows of the attacker's sessions in one cluster. However, the low-rate attack can only be detected by counting for synchronized data transmissions by virtual instances from different customers' projects.

5 Conclusion

In this study, we focused on timely detection of intentional co-residence attempts in cloud environments that utilize software-defined networking. For this purpose, we analyzed statistical features extracted from network traffic flows on three different levels, and, as a result, we were able to detect both low-rate and high-rate versions of the co-residence verification attack. In the future, we are planning to build a theoretical model to estimate the time required for a malicious customer

to successfully verify the co-residence with the target server, and the time for the cloud provider to detect the attack by analyzing anomalous network activity.

References

1. T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. Proc. of the 16th ACM conference on Computer and communications security, pp. 199–212, 2009.
2. Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang. Incentive compatible moving target defense against VM-colocation attacks in clouds. Proc. of the 27th IFIP International Information Security Conference, pp. 388–399, 2012.
3. J. Wu, L. Ding, Y. Lin, N. Min-Allah, and Y. Wang. XenPump: A New Method to Mitigate Timing Channel in Cloud Computing. Proc. of the 5th IEEE International Conference on Cloud Computing, pp. 678–685, 2012.
4. Y. Han, J. Chan. T. Alpcan, and C. Leckie. Using Virtual Machine Allocation Policies to Defend against Co-Resident Attacks in Cloud Computing. IEEE Tran. on Dependable and Secure Computing, Vol. 14, Is. 1, 2017.
5. A. Herzberg, H. Shulman, J. Ullrich, and E. Weippl. Cloudoscopy: services discovery and topology mapping. In Proceedings of the ACM workshop on Cloud computing security workshop, pp. 113–122, 2013.
6. A. Bates, B. Mood, J. Pletcher, H. Pruse, M. Valafar, and K. Butler. Detecting co-residency with active traffic analysis techniques. Proc. of the ACM Workshop on Cloud computing security workshop, pp. 1–12, 2012.
7. I. Ahmad, S. Namal, M. Ylianttila and A. Gurtov. Security in Software Defined Networks: A Survey. IEEE Communications Surveys & Tutorials, Vol. 17, Is. 4, pp. 2317–2346, 2015.
8. F. Hao, T. Lakshman, S. Mukherjee, H. Song. Secure cloud computing with a virtualized network infrastructure. Proc. of the 2nd USENIX conference on Hot topics in cloud computing, pp. 16–16, 2010.
9. H. Ma, H. Ding, Y.Yang, Z. Mi, and M. Zhang. SDN-Based ARP Attack Detection for Cloud Centers. Proc. of IEEE 12th Intl. Conf. on Ubiquitous Intelligence and Computing and IEEE 12th Intl. Conf. on Autonomic and Trusted Computing and IEEE 15th Intl. Conf. on Scalable Computing and Communications and Its Associated Workshops, pp. 1049–1054, 2015.
10. C. Buragohain and N. Medhi. FlowTrApp: An SDN based architecture for DDoS attack detection and mitigation in data centers. Proc. of the 3rd International Conference on Signal Processing and Integrated Networks, pp. 519-524, 2016.
11. C. Xu, G. Zhao, G. Xie and S. Yu. Detection on application layer DDoS using random walk model. Proc. of IEEE International Conference on Communications (ICC), pp. 707–712, 2014.
12. M. Zolotukhin, T. Hämäläinen, T. Kokkonen and J. Siltanen. Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic. Proc. of 23rd International Conference on Telecommunications (ICT), pp. 1–6, 2016.
13. M. Akiyama, T. Kawamoto; M, Shimamura; T. Yokoyama, Y. Kadobayashi, and S. Yamaguchi. A proposal of metrics for botnet detection based on its cooperative behavior. Proc. of International Symposium on Applications and the Internet Workshops, pp. 82–85, 2007.
14. J. Herlocker, J. Konstan, and J. Riedl. An Empirical Analysis of Design Choices in Neighborhood-based Collaborative Filtering Algorithms. ACM Transactions on Information Systems, Vol.5, Is. 4, pp. 287–310, 2002.