Could Network View Inconsistency Affect Virtualized Network Security Functions?

Mohamed Aslan  
Department of Systems and Computer Engineering  
Carleton University. Ottawa, ON, Canada.  
maslan@sce.carleton.ca

Ashraf Matrawy  
School of Information Technology  
Carleton University. Ottawa, ON, Canada.  
ashraf.matrawy@carleton.ca

Abstract—With SDN increasingly becoming an enabling technology for NFV in the cloud, many virtualized network functions need to monitor the network state in order to function properly. An outdated network view at the controllers can impact the performance of those virtualized network functions. In earlier work, we identified two main factors contributing to an outdated network view in the case of a load-balancer: network state collection and controllers’ state distribution. In this paper, we anticipate that the impact might be different in case of security functions. Therefore, we study the impact of an outdated network view on an anomaly-based IDS application. In particular, we investigate: (1) the impact of controllers’ state distribution on the performance of a distributed IDS in the case of a DDoS attack; and (2) the impact of network state collection on the performance of an IDS in the case of a TCP SYN flood attack. Our results showed that the outdated network view had negative impact on the IDS anomaly-detection performance in the experiments that we conducted.

Keywords—NFV, SDN, Security, IDS, Performance, Inconsistency.

I. INTRODUCTION

The Network Function Virtualization (NFV) is an emerging concept in cloud computing that enables network functions to be run virtualized as software instances on the cloud. Software-Defined Networking (SDN) is a promising network architecture, which simplifies the design of the virtualized network functions (VNF), and hence it is widely used in the cloud. In SDN, the control and data planes are decoupled. In order for applications to function properly, control-planes need to maintain a global view of the network.

Any inconsistency in the controller’s network view might have an impact on applications’ performance. In earlier work [1], we studied how the network state information collected by the controllers from the data-planes could negatively impact a load-balancer’s performance. While, Levin et al. [2] studied how the network state synchronization, in case of distributed controllers, among controllers could have a negative impact on a load-balancer’s performance. In this paper, we study the impact of network view inconsistency on a different network functions, an Intrusion Detection System (IDS).

In this paper, we make the following contributions: (1) we investigate the impact of controllers’ state distribution on the performance of a distributed anomaly-based IDS in the presence of a Distributed Denial of Service (DDoS) attack. (2) we investigate the impact of network state collection on the performance of the IDS in the presence of a TCP SYN flood.

Our results showed that the IDS function that we evaluated was negatively impacted by the outdated information. Also, the type of the attack impacted how the outdated information affected the IDS.

The rest of this paper is organized as follows: In III we provide a background on the topic and related work. The setup of the IDS experiment is presented in IV. We present two different attack scenarios and their results in V and VI. Finally VII will be our conclusion and an outline for possible foreseeable work.

II. BACKGROUND AND RELATED WORK

A. Network State Collection Vs Controllers’ State Distribution

As aforesaid, SDN controllers need to maintain an up-to-date view of the network in order for the virtualized network functions to operate properly. To alleviate scalability and reliability issues in large-scale SDN deployments, recent work [3, 4] encourages the use of physically distributed SDN controllers that are logically centralized. Therefore, controllers rely on two separate sources of state information: (1) data-planes (switches) which they control, and (2) other control-planes (controllers) in case of distributed deployments.

In case of single or distributed controller deployments, controllers need to collect network state information from the switches in a process known as network state collection. And in case of distributed controllers, those controllers also need to exchange network state information in order to keep their network view up-to-date, in a process known as controllers’ state distribution.

B. Impact of Network View Inconsistency

Together the network state collection and the controllers’ state distribution processes could lead to inconsistency in the network state information at the controllers. Such inconsistency might have an impact on applications’ performance.

In previous work [1], we showed that network state collection methods (active or passive) can have an impact on the performance of a load-balancer. Our results showed that the load-balancer’s performance was affected by the polling period of network state collection, in case of active network state collection. However, it was more resilient to the nature of the traffic load compared to passive flow-based network state collection.
Using a distributed flow-based load-balance, which employed a periodic synchronization of controllers’ state information, Levin et al. [2] demonstrated the negative impact of inconsistent network views at the controllers on the load-balancer’s performance. Two issues with Levin et al.’s load-balancer [2] were identified by Guo et al. [5]. First, short synchronization periods were required for acceptable load-balancing performance, leading to high synchronization overhead. Second, their design was vulnerable to forwarding loops due to the inconsistency of network views at the controllers.

In an attempt to mitigate the first issue with Levin et al.’s load-balancer [2], Guo et al. [5] proposed a new trigger-based controller synchronization mechanism designed specifically for load-balancers called Load Variance-based Synchronization (LVS). LVS can perform well with lower synchronization overhead by only synchronizing the controllers when the load of a specific server exceeds a certain threshold.

Subsequently, we can assume that the network state collection and the controllers’ state distribution altogether could negatively impact the performance of virtualized network functions, more specifically load-balancers. Moreover, we anticipate that the impact might be different in case of different VNFs. Hence, in this paper, we study the impact of inconsistency on a different network function rather than a load-balancer. In particular, we study the impact of inconsistent network view on a security function, i.e., an IDS.

### III. Security Network Function

The IDS in our experiment is a network anomaly-based IDS similar to that of [6], which was designed for non-SDNs. The IDS employs a clustering algorithm to learn about the behavior of network traffic and to detect the anomalous ones. Instead of using the NetFlow protocol and the classical K-Means clustering algorithm as in [6], it uses OpenFlow and employs a sequential K-Means clustering algorithm (shown in Algorithm 1). For active network state collection, our IDS implementation relies on OpenFlow STATS_REQUEST messages. In regards to controllers’ state distribution, a simple delta-consistency model (i.e., periodic synchronization) was employed (see [1] for more information).

To evaluate the impact of network view inconsistency on the IDS, we conducted two scenarios in order to simulate attacks on servers that are protected by the IDS. First, we used a set of the clients to launch a DDoS attack in [IV]. Then, we randomly used one of the clients to launch a TCP SYN flood attack on the servers. As for the traffic generation, Table 1 shows the parameters of the employed non-anomalous traffic (attack traffic will be explained in [IV-A] and [IV-A]).

#### IV. DDoS Attack Scenario

##### A. Setup

In this attack scenario, we used a number of the clients to simulate the behavior of a DDoS attack. We used a distributed (i.e., two controllers) active-IDS implementation with a fixed polling period of 2 seconds in this scenario. The network topology shown in Fig. 1. The IDS detects any anomalous behavior by monitoring the traffic arriving at the servers. We trained the IDS to expect traffic from an average of half the total number of clients at a given time. The features vector used for clustering is 3-tuple that is comprised of: (1) the total number of bytes received by a server (b), (2) the total number of packets received by a server (p), and (3) the total number of unique flows received by the server (f) per polling duration (2 sec). A weighted $(w_b = 4096, w_p = 64, w_f = 1)$ Euclidean distance was used as a distance function for the clustering algorithm. We ran the experiment for 10 runs, in each run we simulated 5 DDoS (each attack lasts 30 sec) attacks by allowing the dormant clients to send a CBR traffic to the servers.

In order to measure the performance of the IDS in this attack scenario, we use the following performance indicators: (1) the time to detect an anomalous behavior ($T_D$), (2) the precision ($P$) (shown in (1)), the recall ($R$) (shown in (2)),

---

**Algorithm 1: Using Sequential K-means Clustering at the IDS.**

| Data: $P_{(p,b,f)}$, a point in the features vector space |
| Data: $T$, total number of points |
| Data: $M$, number of clusters |
| Data: $C_i$, the centroid of the $i^{th}$ cluster, $i \in [0, M)$ |
| Data: $N_i$, number of points in the $i^{th}$ cluster |

```
begin
  $T \leftarrow 0$
  if $T < M$ then
    $C_T \leftarrow P$
    $N_T \leftarrow N_T + 1$
  else
    $i_{\text{c}} \leftarrow \text{nearest } (P, C)$
    $C_{i_{\text{c}}} \leftarrow (C_{i_{\text{c}}} \ast N_{i_{\text{c}}}) + P$
    $N_{i_{\text{c}}} \leftarrow N_{i_{\text{c}}} + 1$
    $C_{i_{\text{c}}} \leftarrow \frac{C_{i_{\text{c}}}}{N_{i_{\text{c}}}}$
  $T \leftarrow T + 1$
end

Function nearest $(P, C)$

| Data: $w_k$, weight of vector $k$, $k \in \{p,b,f\}$ |

```

---

**Table 1: Employed legitimate traffic loads and their parameters.**

| Flows rates | Poisson process |
| Flow TTL | 2 sec |
| Msps-per-flow | Poisson process |
| Msps-per-flow rates | $p_1 = 34\text{msg/s}, p_2 = 30\text{msg/s}$ |

---

[1] [2] [3] [4] [5] [6]
B. Results

Fig. 3 shows that as the synchronization period increases the true positives (TP) decreases while the false negatives (FN) increases indicating a degradation in the IDS performance. However, Fig. 4 shows the synchronization period versus the average detection time ($T_δ$). Fig. 5 shows the effect of the synchronization period on the average precision ($P_δ$), recall ($R_δ$) and accuracy ($A_δ$). The results shown in those figures do not necessarily show degradation in the IDS performance because we had to exclude those runs with infinite detection time which increases as the synchronization period increases.

\[
P_δ = \frac{TP}{TP + FP} \tag{1}
\]

\[
R_δ = \frac{TP}{TP + FN} \tag{2}
\]

\[
A_δ = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}
\]

V. TCP SYN ATTACK SCENARIO

A. Setup

In this attack scenario, we used one of the clients to launch a TCP SYN flood attack on the servers. We used a non-distributed (i.e., single controller) active-IDS implementation in this scenario. The network topology shown in Fig. 6.
IDS detects any anomalous behavior by monitoring the traffic generated by the clients. We trained the IDS to expect traffic with parameters shown in Table I. The features vector used for clustering is 3-tuple that is comprised of: (1) the number of bytes per flow duration \( b \), (2) the number of packets per flow duration \( p \), and (3) the number of unique flows from the source \( f \). A weighted \( w_b = 2046, w_p = 16, w_f = 0.25 \) Euclidean distance was used as a distance function for the clustering algorithm. We ran the experiment for 10 runs, in each run we randomly select one of the clients to launch a TCP SYN flood attack.

B. Results

In order to measure the performance of the IDS in this attack scenario, we use the time to detect an anomalous behavior \( T_d \) as our performance indicator. Fig. 7 shows the effect of the polling period on the performance of the IDS. The results show that as the polling period increases the time required to detect the anomalous behavior also increases. We believe that this is due the following reasons: (1) at the detection phase, the IDS fails to detect any attack that occurs between consequent polling periods, and (2) at the training phase, the polling period affects the number of collected data points \( i.e. \) the higher the polling period, the less the number of points collected) used during the IDS training phase, and hence affects the detection.

VI. DISCUSSION AND CONCLUSION

Previously [1], we articulated that load-balancers were affected by the network view inconsistency due to network state collection. In the case of TCP SYN flood attack \( \text{[V]} \), the IDS exhibited the same behavior as the load-balancers \( i.e. \), the higher the polling period the worse the performance (due to reasons discussed in \( \text{[V]} \)). It is also worth mentioning that we tested the effect of synchronization period in the case of a distributed IDS for the TCP SYN flood attack scenario \( \text{[V]} \). However, that effect was negligible compared to the effect of the polling period due to the localized nature of the attack \( i.e. \), an IDS can still detect the presence of the attack even without any state synchronization between the controllers. Conversely, in the case of DDoS attack \( \text{[L]} \), the synchronization period impacted the IDS (like the load-balancer) due the nature of the attack \( i.e. \), malicious clients distributed their traffic through the two domains.

In summary, this paper demonstrated the impact of network view freshness at the SDN controllers on a security network function. Our results indicated that an out-of-date network view had negative impact on the security network function we evaluated. However, the impact differed with the attack scenario. For future work, we plan to investigate how this impact could be mitigated.

ACKNOWLEDGMENT

The second author acknowledges support from the Natural Sciences and Engineering Research Council of Canada (NSERC) through the NSERC Discovery Grant program.

REFERENCES

[1] M. Aslan and A. Matrawy, “On the Impact of Network State Collection on the Performance of SDN Applications,” Communications Letters, vol. 20, no. 1, pp. 5–8, 2015.

[2] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, “Logically centralized?: state distribution trade-offs in software defined networks,” in Proc. of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 1–6.

[3] T. Koponen, M. Casado, N. Gude, J. Stirling, L. Poutievski, M. Zha, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama et al., “Onix: A distributed control platform for large-scale production networks.” in OSDI, vol. 10, 2010, pp. 1–6.

[4] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O’Connor, P. Radoslavov, W. Snow et al., “Onos: towards an open, distributed sdn os,” in Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014, pp. 1–6.

[5] Z. Guo, M. Su, Y. Xu, Z. Duan, L. Wang, S. Hui, and H. J. Chao, “Improving the performance of load balancing in software-defined networks through load variance-based synchronization,” Computer Networks, vol. 68, no. 0, pp. 95 – 109, 2014, communications and Networking in the Cloud.

[6] G. Münz, S. Li, and G. Carle, “Traffic anomaly detection using k-means clustering,” in GI/ITG Workshop MMBnet, 2007.