DFIPS: Toward Distributed Flexible Intrusion Prevention System in Software Defined Network

Xuesong Jia*, Danni Ren*, Yitao Yang*†, Huakang Li*†, Guozi Sun*†,
*College of Computer, †Institute of Computer Technology
Nanjing University of Posts and Telecommunications
Nanjing, China
1214043025@njupt.edu.cn, 1141909032@qq.com, {youngyt, huakanglee, sun}@njupt.edu.cn

***Abstract*—With the evolution of the innovative software defined network (SDN), security issues have been taken into consideration. Intrusion prevention system (IPS) has widely deployed as a crucial measure in traditional network architecture to protect network from malignity. In spite of good capability of protection, IPS is still complained in many aspects, such as fixed deployment, single-point-detection and low utilization rate. In this paper, we propose a distributed flexible intrusion prevention system in software defined network (DFIPS). Our proposed DFIPS has three main modules: a classifier, a detector pool and a control agent. The classifier is in charge of slicing traffic. The detector pool then generates several detector nodes for detecting. The control agent interacts with the classifier and the detector pool, as well as higher level SDN controller APPs and OpenFlow switches. DFIPS integrating with SDN controller can easily achieve good load balancing among DFIPSs without repetitive deployment. We evaluate the two forms of DFIPS interaction and latency to show the advantage of DFIPS. In future, we would implement a more comprehensive DFIPS emulation to prove feasibility. We believe that the proposed DFIPS will be adapted in real networks eventually.**

Keywords*—*Intrusion prevention system (IPS); Software defined network (SDN); OpenFlow

I. Introduction

Software defined network (SDN) is an innovative technology that enables a drastic change in network design and management [1]. By separating the control plane from the data plane and consolidating the control plane, SDN realizes a logically centralized and physically distributed network framework. OpenFlow [2], the most successful application program interface (API) designed for SDN, embraces the paradigm of highly programmable switch infrastructures. It enables an optimal network routing that traffic forwarding is no longer restricted to particular command on individual network devices. A global view of network makes control more flexible and centralized [3].

With the evolution of this new technology, security issues in traditional network still exist in SDN, such as suffering from Distributed Denial of Service (DDoS) attack. Besides, the logical centralization of control logic makes it easier for attackers to paralyze the whole network once they infect the controllers in SDN. So, how to protect SDN from abnormal or unpermitted flow must been taken into consideration.

Traditionally, intrusion prevention system (IPS) [4] which usually deployed at the edge of the trusted internal network, serves as the vital gate in detecting and responding to anomalies. Despite its popularity and advantage, IPS is criticized in many aspects [5], such as local concern, single-point-detection, sub-optimal utilization rate and unbalanced computing resource consuming.

In light of the problems above, we explore a different design. Instead of a inflexible deployment, we propose a distributed flexible framework of IPS in software defined network. We call it DFIPS. DFIPS is embedded in SDN controllers as an application so that it can be deployed flexibly and can have a global view of network that is convenient to supervising. DFIPS has three main modules: a classifier, a detector pool and a control agent. The proposed DFIPS also supports load balancing to achieve a global optimal utilization rate. We use network slicing and coordination work among DFIPSs to enable distributed fine-grained intrusion detection and prevention.

To summarize, the contributions of this paper are shown as follows:

- We present the detailed architecture of our proposed DFIPS (Section II) which involves: (i) three main modules presentation (the classifier, the detector pool and the control agent), (ii) load balancing and distributed processing. Besides, we also present the work procedure and give some assumptions.
- We present the initial evaluation of our design in an emulated SDN environment. We present the two forms of DFIPS interaction and latency to show the superiority of our design (Section III).
- We present a variety of outstanding related work on IPS among different areas (Section IV), before concluding in Section V.

II. Design and Architecture

In traditional network architecture, IPS runs as a standalone device and is interference in traffic forwarding. To have a global view of anomalous traffic and lower delay without decrease detecting capacity, we present IPS in this paper as an application running on the centralized controller. This means that DFIPS has access to the network: all the communication between SDN controller and switches would pass through

DOI reference number: 10.18293/SEKE2016-139

DFIPS; and DFIPS could inspect all data wanting to pass through switches.

Now, we present the detailed description of each portion in DFIPS. The structure of DFIPS is shown in Figure 1. Our proposed DFIPS has two three portions: a classifier, a detector pool and a control agent.

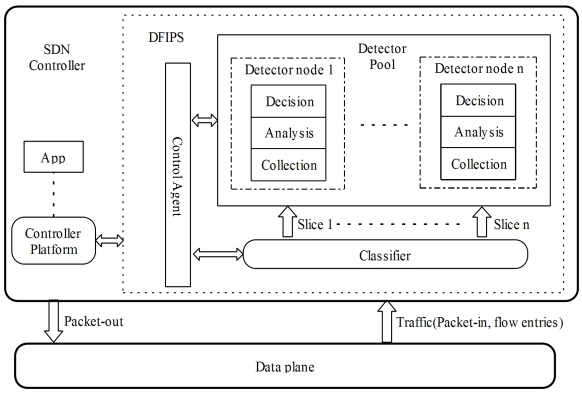


Fig. 1: The structure of DFIPS.

A. Main Modules

1) Classifier: The classifier is in charge of slicing traffic to improve resource allocation and realize a fine-grained distributed detection. We propose a modified FlowVisor [6] serving as a transparent proxy between switches and detector pool to implement network slicing. In our demonstration, a slice can be defined as (switch port, IP protocol, src/dst UDP/TCP port). Each slice is routed to the corresponding detector node independently.

2) Detection Pool: The detector pool generates a several detector nodes according to the number of slices from the classifier that each detector node has three submodule: collection submodule, analysis submodule and decision submodule. The detector node would be discarded when the slice is no longer exist.

Collection submodule keeps a record of messages that switches send to the controller. For further analyzing and queries, these records will be stored in the local database. Analysis submodule is for data analysis to find if flows are malicious, suspicious or normal via detection methods. Decision submodule has the authority to generate appropriate strategies to switches according to the analysis above. It sends information feedback to the control agent.

We propose a local database to store logs of every detection process and a remote database to share the feature library of abnormity collected from each single DFIPS in an associated region. In future, we can do date mining with semantic based algorithm [7]. Particularly, we tend to use R-trees [8] to access the feature library in remote shared database to make queries efficiently.

3) Control Agent: The control agent interacts with the classifier and the detector pool, as well as higher level SDN controller APPs and OpenFlow switches [9]. The control agent also monitors and controls the running state of the classifier and the detector pool.

B. Load Balancing and Distributed Processing

There are two cases of load balancing and distributed processing as follow:

One DFIPS faces multi-switches. A single detector in DFIPS cannot deal with massive traffic from different switches in a very short period of time. Splitting traffic into several slices can enhance the parallel processing capabilities [10]. Each slice has a unified feature, such as a particular protocol, a common port, source/destination addresses. After slicing, a number n of slices are generated and then transmitted into the detector pool where spawns a number n of detector nodes. Besides, DFIPS will analyze the relevance of suspicious traffic among different switches to achieve better intrusion prevention performance.

Multi-DFIPSs face one switch. After a switch registering to several controllers, a master controller will be chosen. Others will be slave controllers. The DFIPS running on a master controller is the master DFIPS and in active mode. The rest of DFIPSs connecting to the same switch will be in standby mode. Once the master controller is down or congestion occurs in DFIPS, slave controller will take charge and DFIPS on it will be active. Furthermore, the DFIPSs in different locations exchange their detection result to update their local feature library of suspicious traffic in order to respond to a similar attack quickly.

C. Work Procedure

When traffic arrives at border switches, DFIPS control agent modifies the flow entries to make sure that every suspicious packet is forwarded to the detector pool for inspecting. The DFIPS control agent orchestrates actions of the detector pool and the command of upper level controller APPs. A detailed process of DFIPS is shown as follow (assuming the switches have been registered to SDN controllers):

1) Firstly, slicing the traffic into several slices and forwarding the slices to detector pool in where generates corresponding number of detector nodes.
2) Collection submodule in the detector node would store the digested information of each slice.
3) Then after the slice of traffic reach the analysis submodule, the analysis submodule would inspect it with several methods, such as high speed deep packet inspection (DPI) [11] and flow level detection [12].
4) Then the decision submodule would give suggestions responding to analysis results. If an abnormity is not detected, the DFIPS control agent would control the packet forwarding to upper level controller platform for normal process. Further, if detecting an abnormity and concluding that the packet is malicious, the DFIPS control agent would send drop command of malicious packet to the related OpenFlow switches; if not, a count of this packet would be recorded as suspicious traffic and increase at a time. DFIPS control agent would modify

the flow entries to make sure that all these suspicious traffic would be forwarded to DFIPS. While the count is higher than a given threshold value, decision submodule would suggest DFIPS control agent to do an appropriate QoS (e.g. rate limitation) when forwarding the packet; otherwise, the packet would be forwarded to upper level controller platform just like normal traffic.

5) DFIPS records the detection process into the local database and updates the shared remote database for future analyzing.

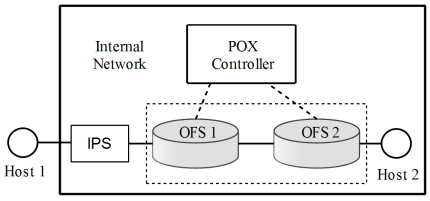
To prevent conflicting rules caused by misconfiguration, some security strategies must be taken. We tend to use FlowChecker [13] to accomplish this mission.

D. Assumptions

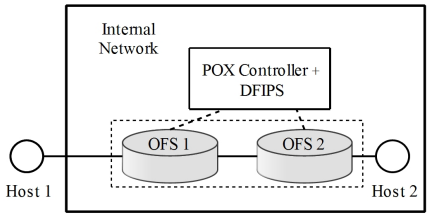
The proposed DFIPS is running as an application on centralized controllers. We assume that controllers are absolutely safe and unoccupied. In our architecture, DFIPS possesses the highest priority to conduct traffic control. It is easy to implement owing to abundant priorities (32768) available in OpenFlow. We also assume the packet loss rate and random noise are very low that can ignore.

III. Evaluation

We build a test-bed (shown in Figure 2) using POX controller [14] and Mininet [15]. In our test-bed, a modified snort [16] is utilized to simulate a detector node and IPS. In this topology, **OFS 1** and **OFS 2** are OpenFlow-enabled switches.



(a) Traffic interference simulation.



(b) Traffic monitoring simulation.

Fig. 2: Simulation diagram.

We first evaluate the two forms of DFIPS interaction to demonstrate the advantage of embedding DFIPS into SDN controller. In this simulation, we use ping to simulate the access request from the Internet to the internal network.

We ping twice (from Host 1 to Host 2) for each form of DFIPS interaction. The flow rules are not pre-configured. In traffic interference framework (Figure 2.(a)), the first ping time is 11.04ms, while the second ping time is 6.54ms. In traffic monitoring framework (Figure 2.(b)), the first ping time is 10.86ms, while the second ping time is 0.862ms.

After the first ping, a flow rule is created and stored in the OpenFlow switch. This implies the second ping time does not have flow rules setup time. So the second ping time reduces largely. The second ping time in traffic monitoring framework is much smaller than in traffic interference framework implies that integrating DFIPS into SDN controllers has lower interference to normal communications.

Then, we evaluate the latency of slicing in DFIPS with topology Figure 2.(b). In this simulation, we examine the average new flow rules setup time on two situations: with slicing and without slicing.

The simulation shows that the average latency of new flow rules setup time with slicing (4.13ms) is more small than without slicing (7.25ms). It demonstrates that slicing traffic into several detector nodes has better performance.

IV. Related Work

Now, we present some related work in intrusion prevention domain as below.

Recently, many proposed schemes have argued the novel IPS toward traditional network and software defined network. An SDN-based IPS deployment is proposed to support an unified scheduling of security applications in the whole network and load balancing among IPSs [17]. In this deployment, SDN controller manipulates IPSs in headquarters and branches to realize thread detection and load balancing between idle and busy IPSs. Different to DFIPS, this scheme cannot realize the flexible deployment.

Some work have shown the global, cooperative and distributed approaches of intrusion detection and prevention [18]. A cooperative IDS framework is addressed in [19] for cloud computing networks. In [20], an algorithm for combining observations from multiple vantage points is proposed. In [21], [22], a number of architectures that IPS/IDS works as a cluster are proposed. Take [21] as example, it proposes a general IDS architecture with splitting responsibility to other nodes by on-path distribution, replicating traffic to off-path nodes (e.g. IDS clusters) and aggregating result to split costly IDS processing. A plenty of previous work has focused on distributed security deployment. DEIDtect [23] is an elastic distributed intrusion detection framework that decouples the location of the protected network from IDS/IPS in the cloud and enterprise network. [24] shows the cooperation of active network management software and extensible hardware to stop an attack.

These proposed methods regard security facilities as middleboxes [25]. And they seek to promote system performance on the basis of the original security hardware. On the contrary, our proposed DFIPS gets rid of the hardware security device and embeds into SDN controllers. This implies that our DFIPS can easily enable flexible deployment, global view and cooperation with other DFIPSs.

A SDN-based framework MalwareMonitor [26] is proposed to realize a comprehensive, distributed malware detection for

networks by tapping into outgoing traffic and detecting it with a number of detector nodes. Contrary to focusing on outgoing traffic, we concern incoming traffic from distrustful Internet to internal network. Besides, our proposed DFIPS is more flexible particularly in deployment.

V. Conclusions and Future Work

In this paper, we present a distributed flexible intrusion prevention system in software defined network. The DFIPS we proposed decouples the location of security devices and protected points by integrating DFIPS into SDN controllers. DFIPS exploits the deployment of software defined network. One DFIPS can easily monitor several switches without repetitive deployment. This flexibility could reduce expenses largely on deployment. Further, DFIPSs in a certain region could commodiously achieve an optimal utilization rate by cooperating with each other. Slicing and virtualization enable the fine-grained detection and distributed processing.

We present the detailed description of design and implementation of each portion in the DFIPS. We evaluate the two forms of DFIPS interaction and the latency of slicing. The simulation result shows that DFIPS has lower delay and lower interference to normal communications.

In future, we would implement a more comprehensive emulation to solve other technical difficulties in DFIPS. For example, how to realize automatic network slicing and flexible detection nodes generation is a knotty problem. We also need to design a more accurate detection algorithm and a gentle controller coordination method. How to improve the performance of DFIPS is importance as well. Our ultimate goal is to get DFIPS widely deployed in real network environment.

Acknowledgments

The authors would like to thank the reviewers for their detailed reviews and constructive comments. This work is supported by the National Natural Science Foundation of China (No. 61502247, No.61502243, No.11501302), the Natural Science Fund of province (BK20140895) and the Ministry of public security key experimental open project fund (No. 2015DSJSYS001).

References

[1] N. Feamster, J. Rexford and E. Zegura, "The Road to SDN: An intellectual history of programmable networks," In ACM SIGCOMM Computer Communication Review vol. 44, no. 2, pp. 87-98, 2014.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," In ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69-74, 2008.

[3] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-defined networking: A comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, 2015.

[4] S. Vasanthi and S. Chandrasekar, "A study on network intrusion detection and prevention system current status and challenging issues," Advances in Recent Technologies in Communication and Computing (ARTCom 2011),3rd International Conference on, pp. 181-183, 2011.

[5] D. Stiawan, A. H. Abdullah and M. Y. Idris, "The trends of Intrusion Prevention System network," Education Technology and Computer (ICETC), 2010 2nd International Conference on, vol. 4, pp. 217-221, 2010.

[6] R. Sherwood, M. Chan, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T. Huang, P. Kazemian, M. Kobayashi, J. Naous, S. Seetharaman, D. Underhill, T. Yabe, K. Yap, Y. Yiakoumis, H. Zeng, G. Appenzeller, R. Johari, N. McKeown and G. Parulkar, "Carving research slices out of your production networks with OpenFlow," In ACM SIGCOMM Computer Communication Review, vol. 40, no. 1, pp. 129-130, 2010.

[7] Zheng Xu et al. Knowle: a Semantic Link Network based System for Organizing Large Scale Online News Events. Future Generation Computer Systems, 43-44, 40-50, 2015.

[8] A. Guttman, "R-trees: A dynamic index structure for spatial searching," In Proceedings of the 1984 ACM SIGMOD international conference on Management of data,pp. 47-57, 1984.

[9] P. Bosshart, G. Gibb, H. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica and M. Horowitz, "Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN," In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp. 99-110, 2013.

[10] S. Gutz, A. Story, C. Schlesinger and N. Foster, "Splendid isolation: A slice abstraction for software-defined networks," In Proceedings of the first workshop on Hot topics in software defined networks, pp. 79-84, 2012.

[11] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," In SIGCOMM '06 Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 339-350, 2006.

[12] Y. Gao, Z. Li and Y. Chen, "A DoS resilient flow-level intrusion detection approach for high-speed networks," In Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, pp. 39, 2006.

[13] A. Ehab and A. Saeed, "FlowChecker: configuration analysis and verification of federated OpenFlow infrastructures," In Proceedings of the 3rd ACM workshop on Assurable and usable security configuration, pp. 37-44, 2010.

[14] http://www.noxrepo.org/.

[15] http://mininet.org/

[16] https://www.snort.org/.

[17] L. Zhang, G. Shou, Y. Hu and Z. Guo, "Deployment of intrusion prevention system based on software defined networking," In Communication Technology (ICCT), 2013 15th IEEE International Conference on, pp. 26-31, 2013.

[18] V. Sekar, R. Krishnaswamy, A. Gupta and M. K. Reiter, "Network-wide deployment of intrusion detection and prevention systems," In Proceedings of the 6th International Conference (Co-NEXT), Article No. 18, 2010.

[19] C. Lo, C. Huang and J. Ku, "A cooperative intrusion detection system framework for cloud computing networks," In Parallel Processing Workshops (ICPPW), 2010 39th International Conference on, pp. 280-284, 2010.

[20] A. Lakhina, M. Crovella and C. Diot, "Diagnosing network-wide traffic anomalies," ACM SIGCOMM Computer Communication Review, vol 34, no. 4, pp. 219-230, 2004.

[21] V. Heorhiadi, M. K. Reiter and V. Sekar, "New opportunities for load balancing in network-wide intrusion detection systems," In Proceedings of the 8th international conference on Emerging networking experiments and technologies (CoNEXT), pp. 361-372, 2012.

[22] F. Gadaud, "NIDS architecture for clusters," In Collaborative Technologies and Systems, 2005. Proceedings of the 2005 International Symposium on, pp. 78-83, 2005.

[23] P. K. Shanmugam, N. D. Subramanyam, J. Breen, C. Roach and J. V. d. Merwe, "DEIDtect: Towards distributed elastic intrusion detection," In Proceedings of the 2014 ACM SIGCOMM workshop on Distributed cloud computing (DCC), pp. 17-24, 2014.

[24] T. Sproull and J. Lockwood, "Distributed intrusion prevention in active and extensible networks," Proceedings of the 6th IFIP TC6 international working conference on Active networks, pp. 54-65, Springer-Verlag Berlin, Heidelberg 2007.

[25] A. Gember, R. Grandl, J. Khalid and A. Akella, "Design and implementation of a framework for software-defined middlebox networking," In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp. 467-468, 2013.

[26] Z. Abaid, M. Rezvani and S. Jha, "MalwareMonitor: An SDN-based framework for securing," In Proceedings of the 2014 CoNEXT on Student Workshop (CoNEXT Student Workshop), pp. 40-42, 2014.