Towards Efficient Traffic Monitoring for Science DMZ with Side-Channel based Traffic Winnowing

Hongda Li
Clemson University
hongdal@clemson.edu

Fuqiang Zhang
Clemson University
fuqianz@clemson.edu

Lu Yu
Clemson University
lyu@clemson.edu

Jon Oakley
Clemson University
joakley@clemson.edu

Hongxin Hu
Clemson University
hongxih@clemson.edu

Richard R. Brooks
Clemson University
rrb@clemson.edu

ABSTRACT

As data-intensive science becomes the norm in many fields of science, high-performance data transfer is rapidly becoming a core scientific infrastructure requirement. To meet such a requirement, there has been a rapid growth across university campus to deploy Science DMZs. However, it is challenging to efficiently monitor the traffic in Science DMZ because traditional intrusion detection systems (IDSes) are equipped with deep packet inspection (DPI), which is resource-consuming. We propose to develop a lightweight side-channel based anomaly detection system for traffic winnowing to reduce the volume of traffic finally monitored by the IDS. We evaluate our approach based on the experiments in a Science DMZ environment. Our evaluation demonstrates that our approach can significantly reduce the resource usage in traffic monitoring for Science DMZ.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; **Network security**; *Virtualization and security*;

KEYWORDS

Network Function Virtualization; Intrusion Detection Systems; Science DMZ

ACM Reference Format:
Hongda Li, Fuqiang Zhang, Lu Yu, Jon Oakley, Hongxin Hu, and Richard R. Brooks. 2018. Towards Efficient Traffic Monitoring for Science DMZ with Side-Channel based Traffic Winnowing. In *SDN-NFV Sec'18: 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, March 19–21, 2018, Tempe, AZ, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3180465.3180474

1 INTRODUCTION

Recently, cyberinfrastructures have been advancing significantly to enable researchers to: (i) remotely access distributed computing resources and big data sets; and (ii) effectively collaborate with remote peers on a global scale [5]. As data-intensive science becomes the norm in many fields of science, high-performance data transfer is rapidly becoming a core scientific infrastructure requirement. To meet such a requirement, there has been a rapid growth across university campuses to deploy Science DMZs [6]. The Science DMZ is a high performance network environment, which is typically deployed at the edge of a university's network to support big data transfer and access to high-performance computation through very high bandwidth networks in an open environment.

However, it is challenging to monitor the traffic in the high performance network environment like Science DMZ. Traditional intrusion detection systems (IDSes) detect threats through stateful packet processing, which is resource-consuming. Though researchers have proposed to deploy IDSes with multi-thread [14] and cluster [15] architecture, it still requires numerous resources to support traffic monitoring for the high performance environment. For example, it is reported that Snort [2] can handle 800 Mbps of traffic per processor and Bro [3] is suggested to allocate one core for every 80Mbps of traffic [11].

To address the aforementioned challenge in traffic monitoring for the high performance network environment like Science DMZ, we propose to develop a lightweight side-channel based anomaly detection system for traffic winnowing that serves as a pre-filter to reduce the volume of traffic reflected to the back-end IDS instances such as Bro and Snort. The basis of this approach is the insight that attack traffic is qualitatively different from existing network traffic. Constructing attacks that have the same statistics as normal traffic is an order magnitude more difficult than constructing common network intrusions. In addition, monitoring network statistics is much less expensive than the fine-grained analysis of current IDSes.

In this paper, we construct the lightweight side-channel based anomaly detection system by monitoring the inter-packet delay of each flow. Since the Science DMZ is specific for the data transfer and are isolated from internal system, scientific applications running in the network are not likely to change. The lightweight detection system determines whether a flow is generated by a legitimate application by comparing the timing pattern of the test flow to a Hidden Markov Model (HMM) that represents the timing pattern of the legitimate traffic. If no statistically significant deviation is observed, this flow will be filtered out by the lightweight detection system and not reflected to the back-end IDS instances, otherwise the flow will be passed to the back-end IDS instances for further analysis. To achieve dynamic filtering, the lightweight detection

SDN-NFV Sec'18, March 19–21, 2018, Tempe, AZ, USA

ACM ISBN 978-1-4503-5635-0/18/03…$15.00
https://doi.org/10.1145/3180465.3180474

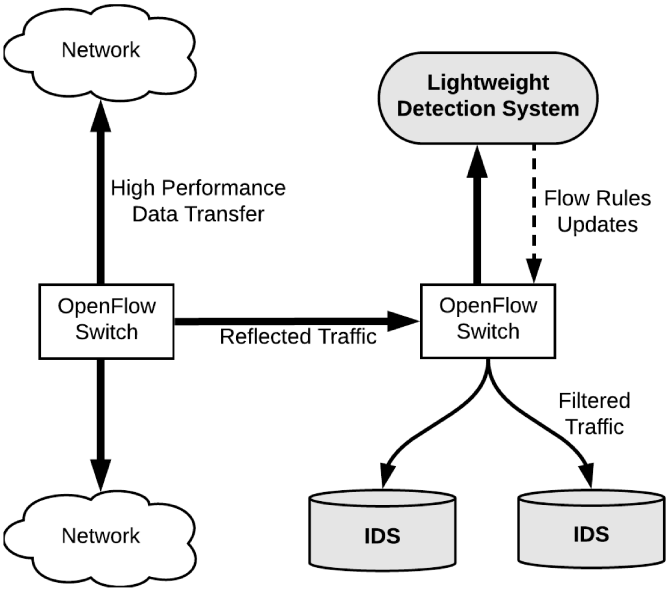

Figure 1: Overview of IDS with lightweight detection system deployed in the Science DMZ environment.

system leverages the programmability of OpenFlow switches to dynamically control the flows. We employ the Network Function Virtualization (NFV) technique to enable on-demand provisioning.

The rest of this paper is organized as follows. Section 2 describes our approach to enable the side-channel based anomaly detection system in the Science DMZ environment. We evaluate our approach in Section 3 and discuss related work in Section 4. Conclusion is draw in Section 5.

2 OUR APPRAOCH

In this section, we first give the overview of our approach, which consists of the lightweight detection system and IDS instances, then we discuss in detail how to develop the lightweight detection system.

2.1 Approach Overview

Figure 1 shows the overview of our proposed approach. The traffic in the Science DMZ is reflected to a passively monitoring system, which consists of OpenFlow switches for traffic steering and virtual machines running lightweight detection system and the IDS instances such as Bro or Snort. All the reflected traffic is delivered to the lightweight detection system, which performs side-channel based anomaly detection. At the same time, a copy of the traffic is distributed to the IDS instances. Once a flow is deemed valid (i.e., legitimate traffic) based on the analysis of the lightweight detection system, the lightweight detection system will update the flow rules in the OpenFlow switches such that the deemed valid flow will not be copied to the IDS instances for processing. This valid flow however should still be monitored by the lightweight detection system. If it is again deemed to be suspicious (i.e., potentially illegitimate), the lightweight detection system will update the flow rules in the OpenFlow switches to again copy the suspicious flow to the IDS instances for further analysis. As a result, the IDS instances are actually performing analysis on the filtered traffic, which cannot be deemed as valid flows only by the lightweight detection system.

The lightweight detection system and the IDS instances are run in virtual machines so that these functionalities can be provisioned on demand. Especially for the IDS instances, the workload of each instance can vary significantly from time to time, because the lightweight detection system might update the flow rules in the OpenFlow switches dynamically based on the detection results. To adapt to the dynamics, the lightweight detection system and the IDS instances leverage the flexibility feature of NFV. When some of the instances are overloaded, new instances will be created and some flows will be redirected to the new instances. When multiple instances are underloaded, some of them are destroyed and the corresponding traffic will be redistributed to the remaining instances. There are some existing frameworks that can support this dynamical scaling, including OpenNF [9] and Split/Merge [13]. Specific for IDSes, we propose in [20] to deploy IDSes as microservices to achieve greater flexibility and efficiency. All the instances are scaled at per-flow granularity. We do so for two reasons. First, OpenFlow switches naturally deliver the network traffic at per-flow granularity. Thus, it is straightforward to implement per-flow distribution using OpenFlow switches. Second, there is a body of work [14, 15] showing that network traffic can be efficiently distributed at per-flow granularity to multiple IDS instances for processing.

2.2 Lightweight Detection System

The lightweight detection system is a core component of our proposed approach. We first identify the requirements of the lightweight detection system and then present how we develop the lightweight detection system to meet the requirements.

We identified two key requirements for the lightweight detection system as follows.

- **Very low false negative:** The lightweight detection system should have a very low false negative (i.e., illegitimate traffic is falsely considered as legitimate traffic) rate. However, it can tolerate a relatively high false positive rate, that is, allowing legitimate traffic delivered to the IDS instances for further monitoring as long as the overall (lightweight detection system and IDS instances) system performs more efficiently than the IDS instances.
- **Efficient detection:** The lightweight detection system should consume much less resources than the IDS instances. This requirement has two respects. First, the lightweight detection system should take much less resources to process each packet than the IDS instances. Second, the lightweight detection system should filter out a large amount of traffic for the IDS instances by analyzing a small amount of traffic.

We develop the lightweight detection system based on two key insights. First, the Science DMZ resources are assumed to interact with external systems for the data transfer and are isolated from internal systems, thus the Science DMZ has limited its applications to a set of domain-specific applications. For example, Science

DMZ uses Lustre [1] or GPFS [2] as high-speed parallel file systems, GridFTP [3] or FTD [4] for data transfer, and discipline-specific tools such as XRootD [5]. This feature provides a baseline of what traffic should be deemed as legitimate. We model the timing features of the common used applications (administrators can assign which applications are commonly used) by employing Hidden Markov Models (HMM). Existing work [12, 19, 21] has shown that network protocols can be modeled by HMMs effectively. Since malicious traffic is qualitatively different from traffic generated by "known" and "valid" application protocols, the lightweight detection system considers a flow as suspicious if its timing feature does not match any known application protocols. Second, since users of Science DMZ usually transfer huge files, the size and duration of the traffic flows in the Science DMZ are usually huge. This feature provides an opportunity for the lightweight detection system to distinguish network traffic generated by known application protocols from malicious traffic, because the huge traffic flows contain sufficient information for HMM inferring and comparing. In summary, it is well suited to develop the lightweight detection system by modeling the timing feature of flows and use the timing feature to determine whether a flow is legitimate.

To satisfy the "very low false negative" requirement, we can tune the parameters of the HMMs to achieve arbitrary low false negative detection rate, while have a relatively high false positive rate. In particular, we set a very high confidence level under which should a flow be considered has the same timing feature with a known valid application protocol. If the lightweight detection system does not have sufficient confidence to claim a flow has a timing feature identical to some known application protocols, this flow will be delivered to the IDS instances for further analysis. This design ensures illegitimate traffic is not likely to be missed, thus satisfies the "very low false negative" requirement. To satisfy the "efficient detection" requirement, the lightweight detection system only checks the inter-packet delays of a flow. Each time a packet comes in, the lightweight detection system computes the timing interval between this packet and the previous packet in the same flow. This interval is then used to update the states of the HMM state machine associated to that flow. If the lightweight detection system does not yet have sufficient data to make a decision, this packet will also be sent to the IDS instances for analysis. If the HMM of a flow finally turns out to be identical with the HMM of a known application protocol, this flow is considered legitimate and does not need to be further analyzed by the IDS instances.

3 EVALUATION

We have conducted experiments in the CloudLab [1] at Clemson site, which is a part of the Science DMZ of Clemson University, to evaluate the performance of our approach. We generated traffic in the CloudLab by using the GridFTP application, which has been commonly used by Science DMZs for data transfer. We use the GridFTP traffic as our legitimate traffic data set. We also generated traffic using SCP [6], which is considered as an uncommon application on Science DMZs due to its limited transfer speed. We use the SCP traffic as illegitimate traffic data set. We implemented the lightweight detection program to collect the inter-packet delays of each flow and model the delays using HMM. More details about modeling the inter-packet delays using HMM are presented in some exiting work [12, 19, 21]. We employed Bro as the IDS instances, since it has been approved that Bro can be deployed with cluster architecture. We used Open vSwitch as the OpenFlow switches.

In the experiment, we evaluated the resource usage in two scenarios: *i)* monitoring the traffic of Science DMZ with the lightweight detection system and *ii)* monitoring the traffic of Science DMZ without lightweight detection system.

We use Equation (1) to describe the CPU usage of the whole system in scenario *i)*, where T is the average processing time of each packet [7], L is the processing time of the lightweight detection system for each packet, I is the processing time of the IDS instances, α is the ratio of packets in a flow that requires to be checked before the lightweight detection system can determine whether a flow is legitimate, β is the ratio of illegitimate flows, γ is the false positive rate of the lightweight detection system. The CPU usage of scenario *ii)* is then represented as I, since all the packets in all flows are processed by the IDS instances in this case.

$$T = L + \alpha I + (1 - \alpha)(\beta I + \gamma(1 - \beta)I) \quad (1)$$

The results of our experiments show that the lightweight detection system processes a packet in every 0.66 microsecond, while the Bro IDS instances processes a packet in average using 44.46 microseconds. In scenario *i)*, our HMM indicates that 100K packets is sufficient to infer whether a flow is identical to a known application protocol. The 100K packets in our case include 1GB of traffic. We generate both GridFTP and SCP traffic by transferring 10GB files. That means, the lightweight detection system can deem a flow is legitimate by only checking 10% of the packets in the flow. Thus, the α is 0.1 in our case. We include a half of SCP and a half of GridFTP traffic. So the β in our case is 0.5. We tune the confidence level such that for all of our date set, the lightweight detection system achieves zero false negative and 0.38 false positive. So the γ in our case is 0.38. According to Equation (1), the average processing time of each packet in scenario *i)* is 32.7 microseconds, which is 74% of the processing time of each packet in scenario *ii)*, where the time is 44.46 microseconds. In reality, the amount of legitimate traffic is far less than 50% as is set in our experiments. If we set the ratio of illegitimate traffic as 10%, according to Equation (1), the average processing time of each packet in scenario *i)* is 22.79 microseconds, which is 51% of the time in scenario *ii)*. In addition, the flows in the Science DMZ might be larger than 10GB as is set in our experiments. As the flow size increases, according to Equation (1), the average processing time of each packet in scenario *i)* decreases. This means our approach benefits more as the flow size increases.

4 RELATED WORK

Improving the processing capacity of IDSes to keep up with the pace of the growth of traffic rate has been studied a lot in the literature.

[1] Lustre: http://www.lustre.org
[2] GPFS: https://en.wikipedia.org/wiki/IBM_General_Parallel_File_System
[3] GridFTP: http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/
[4] FTD: http://monalisa.cern.ch/FDT/
[5] XRootD: http://xrootd.org/
[6] https://en.wikipedia.org/wiki/Secure_copy
[7] For the ease of evaluation, we assume each flow has the same number of packets

A body of work focuses on improving the scalability of IDSes by parallelizing IDSes with multi-thread [7], multi-core [14, 18] processing and cluster architecture [15]. Those existing work improves the capacity of the IDSes by employing more execution instances/threads. A another body of work focuses on improving the processing speed of a single IDS instance leveraging special hardware such as GPU [10, 16, 17]. Unlike our approach, which improves the capacity of IDSes by reducing the overall resource consumption of the whole system (thus with the same amount of resources, our approach gains greater capacity), those approaches enable greater capacity by exploiting more resources.

The work closest to our discussion is [8], which reduces the resource consumption by predicting the traffic patterns and selectively loading detection polices for IDSes. However, that work is limited to a single IDS instance and is not specific for high performance networks like the Science DMZs. We can employ the approach presented by this work to each of our IDS instances.

SciPass [4] presents an approach to secure the Science DMZ using OpenFlow and Bro. The authors employ an array of IDS instances to handle all flows. In contrast, our work filters out known valid flows with a lightweight detection system, reducing the number of flows being sent to the IDS instances. Our goal is to significantly reduce the resource consumption of the IDS instances.

5 CONCLUSION AND FUTURE WORK

We proposed a new approach to efficiently monitoring the traffic of Science DMZ based on side-channel features of flows. Our approach employs a lightweight detection system as a traffic filter, which significantly reduces the volume of traffic being processed by the IDS instances. We have designed and implemented a lightweight detection system based on the inter-packet timing feature. Our preliminary evaluation results demonstrated that our approach can achieve greater efficiency in CPU usage than traditional approaches.

As our future work, we will formalize the resource usage of our approach and conduct more comprehensive evaluations based on the formulas. In addition, we will include more side-channel features in the lightweight detection and employ more advanced machine learning techniques to achieve better detection accuracy, while ensuring sufficient efficiency.

ACKNOWLEDGMENTS

This work was partially supported by grants from National Science Foundation (NSF-OAC-1642143, NSF-CNS-1700499, and NSF-DGE-1723663).

REFERENCES

[1] 2015. CloudLab. http://www.cloudlab.us/. (2015).
[2] 2018. Snort. https://www.snort.org/. (2018).
[3] 2018. The Bro Network Security Monitor. https://www.bro.org/. (2018).
[4] Edward Balas and A Ragusa. 2014. SciPass: a 100Gbps capable secure Science DMZ using OpenFlow and Bro. In *Supercomputing 2014 conference (SC14)*.
[5] Prasad Calyam, Alex Berryman, Erik Saule, Hari Subramoni, Paul Schopis, Gordon Springer, Umit Catalyurek, and Dhabaleswar K Panda. 2014. Wide-area overlay networking to manage science DMZ accelerated flows. In *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE, 269–275.
[6] Eli Dart, Lauren Rotman, Brian Tierney, Mary Hester, and Jason Zurawski. 2014. The science dmz: A network design pattern for data-intensive science. *Scientific Programming* 22, 2 (2014), 173–185.
[7] Lorenzo De Carli, Robin Sommer, and Somesh Jha. 2014. Beyond pattern matching: A concurrency model for stateful deep packet inspection. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1378–1390.
[8] Holger Dreger, Anja Feldmann, Vern Paxson, and Robin Sommer. 2008. Predicting the resource consumption of network intrusion detection systems. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 135–154.
[9] Aaron Gember-Jacobson, Raajay Viswanathan, Chaithan Prakash, Robert Grandl, Junaid Khalid, Sourav Das, and Aditya Akella. 2014. OpenNF: Enabling innovation in network function control. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 163–174.
[10] Muhammad Asim Jamshed, Jihyung Lee, Sangwoo Moon, Insu Yun, Deokjin Kim, Sungryoul Lee, Yung Yi, and KyoungSoo Park. 2012. Kargus: a highly-scalable software-based intrusion detection system. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 317–328.
[11] George Khalil. 2015. Open Source IDS High Performance Shootout. https://www.sans.org/reading-room/whitepapers/intrusion/open-source-ids-high-performance-shootout-35772. (2015).
[12] C. Lu, J. M. Schwier, R. M. Craven, L. Yu, R. R. Brooks, and C. Griffin. 2013. A Normalized Statistical Metric Space for Hidden Markov Models. *IEEE Transactions on Cybernetics* 43, 3 (June 2013), 806–819. https://doi.org/10.1109/TSMCB.2012.2216872
[13] Shriram Rajagopalan, Dan Williams, Hani Jamjoom, and Andrew Warfield. 2013. Split/Merge: System Support for Elastic Execution in Virtual Middleboxes.. In *NSDI*, Vol. 13. 227–240.
[14] Robin Sommer, Vern Paxson, and Nicholas Weaver. 2009. An architecture for exploiting multi-core processors to parallelize network intrusion prevention. *Concurrency and Computation: Practice and Experience* 21, 10 (2009), 1255–1279.
[15] Matthias Vallentin, Robin Sommer, Jason Lee, Craig Leres, Vern Paxson, and Brian Tierney. 2007. The NIDS cluster: Scalable, stateful network intrusion detection on commodity hardware. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 107–126.
[16] Giorgos Vasiliadis, Spiros Antonatos, Michalis Polychronakis, Evangelos P Markatos, and Sotiris Ioannidis. 2008. Gnort: High performance network intrusion detection using graphics processors. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 116–134.
[17] Giorgos Vasiliadis, Michalis Polychronakis, and Sotiris Ioannidis. 2011. MIDeA: a multi-parallel intrusion detection architecture. In *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 297–308.
[18] Benjamin Wun, Patrick Crowley, and Arun Raghunth. 2009. Parallelization of Snort on a multi-core platform. In *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ACM, 173–174.
[19] L. Yu, J. M. Schwier, R. M. Craven, R. R. Brooks, and C. Griffin. 2013. Inferring Statistically Significant Hidden Markov Models. *IEEE Transactions on Knowledge and Data Engineering* 25, 7 (July 2013), 1548–1558. https://doi.org/10.1109/TKDE.2012.93
[20] Nuyun Zhang, Hongda Li, Hongxin Hu, and Younghee Park. 2017. Towards Effective Virtualization of Intrusion Detection Systems. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 47–50.
[21] X. Zhong, A. Ahmadi, R. Brooks, G. K. Venayagamoorthy, L. Yu, and Y. Fu. 2015. Side channel analysis of multiple PMU data in electric power systems. In *2015 Clemson University Power Systems Conference (PSC)*. 1–6. https://doi.org/10.1109/PSC.2015.7101704