MalwareMonitor: An SDN-based Framework for Securing Large Networks

Zainab Abaid[†‡], Mohsen Rezvani[†] and Sanjay Jha[†]
[‡]NICTA, Australia
[†]School of Computer Science and Engineering, UNSW Sydney, Australia
{zainaba, mrezvani,sanjay}@cse.unsw.edu.au

ABSTRACT

Large high-speed networks such as in campuses and enterprises teem with malware infections; current solutions are either incapable of coping with the high data rates, or lacking in effective and speedy threat detection and mitigation. This work presents an early architecture for MalwareMonitor, a security framework that leverages SDN technology to address these limitations. We propose *elastically* partitioning network traffic to enable distributing detection load across a range of detectors; further, a centralized SDN controller allows for network-wide threat correlation as well as speedy control of malicious flows.

1. BACKGROUND AND INTRODUCTION

Large networks such as enterprises and campuses have increasingly come to adopt network intrusion detection systems (NIDS) for mitigating malware. However many NIDS are centralized requiring deep packet inspection (DPI) [1] and hence impractical for deployment in high speed networks; others are flow-based [2], sacrificing content inspection in favor of efficiency. Holistic malware detection requires not only flow-based but also packet-level analysis, for example for signature-matching in packet payloads. The solution lies in distributed NIDS such as [10] that distribute the DPI load across multiple machines. However even these approaches are very limited in threat blocking capability. We believe that the software defined networking (SDN) pardigm is a promising solution to these limitations, as it allows for distributing network functions and flow-level contol of the network. In fact, a recent SDN security framework [8] demonstrates such flow-blocking capability; its limitation is that it does not integrate DPI.

In this work we propose leveraging SDN technology for *elastic* (requirement-dependent) load balancing by tapping into outgoing traffic and distributing it across multiple detector nodes, reducing the amount of data to be processed at each node. The detectors also implement an information sharing approach to detect coordinated malware behaviour in the network that may be missed if they were to work entirely in silos. Thus our work combines the advantage of a distributed DPI-performing NIDS with SDN, allowing for integration into existing hardware (installing OpenFlow as a firmware upgrade) and faster and more flexible threat blocking.

We now outline an early design for an implementation of our framework, MalwareMonitor.

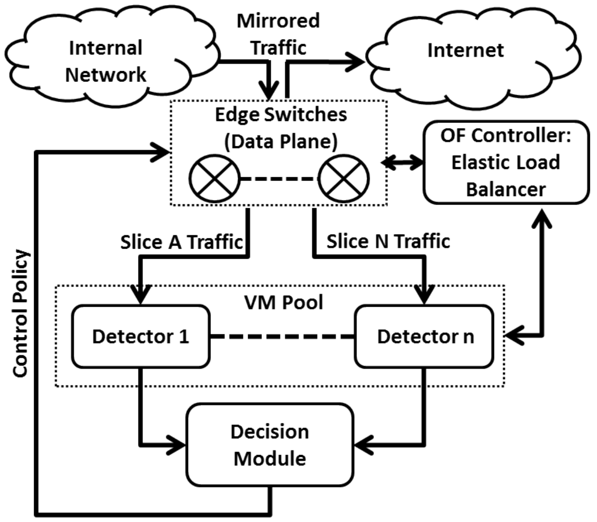

Figure 1: Architecture of MalwareMonitor.

2. MALWAREMONITOR ARCHITECTURE

2.1 Design Overview

Based on the limitations of existing approaches, we require the following from MalwareMonitor: (1) it should be capable of high speed deep packet inspection (DPI) without noticeably degrading throughput or affecting normal network operation, and (2) it should maintain a centralized view of network threats to enable observing coordinated malware such as botnets.

Our proposed architecture is shown in Figure 1. We deploy the load balancing module on an Open-Flow (OF) [4] controller on top of the data plane, comprised of OF-compliant switches at the network gateway. This module taps into outgoing traffic (similar to [10]), forwarding it to its destination but additionally slicing the network such that all packets from a given slice are mirrored to a detector node responsible only for that slice. As this detector node receives only a

CoNEXT Student Workshop'14, December 2, 2014, Sydney, Australia.
Copyright 2014 ACM 978-1-4503-3282-8/14/12 ...$15.00.
http://dx.doi.org/10.1145/2680821.2680829.

subset of traffic, it can run detection logic, involving DPI, at line-rate. It additionally computes flow level characteristics for each flow, summarizing its results as a single tuple. It sends this tuple, along with flow identification information, to a centralized decision module, which executes correlation logic and reports malicious flows. Thus we meet both our DPI and centralized view requirements. Finally, based on its detection result, the decision module (which also runs an OF controller) communicates flow rules to the data plane to block malicious flows.

2.2 Description of MalwareMonitor Modules

We now describe how we envision each module of MalwareMonitor to be implemented.

Elastic Load Balancer.

The load balancer is an SDN application running on an OpenFlow controller such as NOX [9]. It comprises a virtual machine (VM) Controller and network slicing logic. The VM controller assesses the network load and spawns a number n of machines implementing detection logic (detector nodes) from a VM pool it controls – hence the notion of elasticity. The job of the network slicing logic is to then logically partition the network into n *slices*, where a slice is defined as all traffic having a particular characteristic such as a certain port, protocol, or set of source addresses. It then creates flow rules in the data plane to forward a copy of all packets from each slice to the corresponding detector node We propose modifying FlowVisor [7] for slicing or a more flexible approach such as [3].

Detector Nodes.

Detector nodes first statefully inspect each received packet for malicious content to detect events associated with malware infections (e.g. DoS attacks or communication with malicious servers). Secondly, they compute flow statistics for each flow they receive, such as byte and packet counts or burst sizes. Many tools can accomplish malicious event detection; we propose using a ruleset available for Snort IDS [6] as part of BotHunter [1]. The second job can be accomplished by leveraging a feature of OpenFlow where switches maintain flow statistics for each flow that passes through them; thus the detector node can run an OF controller and simply pull flow statistics from the upstream edge switch. More advanced statistics can be calculated using simple utlilities (tshark, capinfos, tcpdump) available for Unix-based operating systems. The detector node then amalgamates the flow identification tuple (source/destination IP and port, and protocol) with any detected malicious activity (as an alert message) and the computed flow statistics, and sends it to the decision module.

Decision Module.

The decision module examines the flow meta-data sent by the detector nodes to compute a maliciousness score for each individual flow. However unlike the detector nodes, which looked at only one flow at a time, the decision module performs correlation across flows to discover suspicious patterns or combinations of events – for example, a distributed DoS attack, or a number of flows from different hosts connecting to a common blacklisted IP (indicating members of a botnet). Thus it assigns a maliciousness score to a flow based not only on the events that have been detected as part of it but also on its similarity with other malicious flows – this similarlity can be seen in common events occurring in a flow or similar statistical characteristics. We believe that some stealthy flows that are able to disguise themselves individually can be discovered in this manner.

We propose using a correlation algorithm such as our earlier work [5], in which we assign higher risk scores to flows that are correlated with other risky flows. While we believe this algorithm is mathematically rigorous and well-suited to the task at hand, a different algorithm can be used if needed.

We can normalize the maliciousness score of each flow to interpret it as a probability value, which we then threshold to declare infections. Finally, an OF controller within the decision module issues flow rules to block detected malicious flows at the data plane.

3. CONCLUSION AND FUTURE WORK

The framework proposed in this work is an early prototype for a comprehensive, distributed malware detection system for networks. We proposed elastically distributing network traffic to a load-dependent number of virtual detector nodes to reduce the burden on a central point, performing the most processing- and data-intensive part of the detection in this distributed fashion. A centralized decision module then combines detection summary from all nodes and generates a final maliciousness score for each network flow; high risk flows are blocked leveraging OpenFlow. Future work along this direction requires solving a number of technical challenges for each module of the framework. For instance, we need to devise a formal mechanism to decide how much load a single detector VM can take and when a new one should be spawned, perhaps drawing on approaches such as [3]. Another challenge is maintaining host and network state information at the detector and decision modules in highly dynamic environments such as a campus network, and deploying effective detection algorithms. A final and important future work direction would be evaluating this framework in a real network deployment with actual users and real world malicious data.

4. REFERENCES

[1] Guofei Gu, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, and Wenke Lee. BotHunter: Detecting malware infection through IDS-driven dialog correlation. In *USENIX Security*, volume 7, pages 1–16, 2007.

[2] Anestis Karasaridis, Brian Rexroad, and David Hoeflin. Wide-scale botnet detection and characterization. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, volume 7. Cambridge, MA, 2007.

[3] Anand Krishnamurthy, Shoban P Chandrabose, and Aaron Gember-Jacobson. Pratyaastha: an efficient elastic distributed sdn control plane. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 133–138. ACM, 2014.

[4] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[5] Mohsen Rezvani, Aleksandar Ignjatovic, Elisa Bertino, and Sanjay Jha. Provenance-aware security risk analysis for hosts and network flows. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–8. IEEE, 2014.

[6] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.

[7] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. FlowVisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[8] Seungwon Shin, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, Guofei Gu, and Mabry Tyson. FRESCO: Modular composable security services for software-defined networks. In *NDSS*, 2013.

[9] Arsalan Tavakoli, Martin Casado, Teemu Koponen, and Scott Shenker. Applying NOX to the datacenter. In *HotNets*, 2009.

[10] Matthias Vallentin, Robin Sommer, Jason Lee, Craig Leres, Vern Paxson, and Brian Tierney. The NIDS cluster: Scalable, stateful network intrusion detection on commodity hardware. In *Recent Advances in Intrusion Detection*, pages 107–126. Springer, 2007.