Fingerprinting Detection in Software Defined Networks

Evan Kenney
Electrical Engineering and Computer Science
United States Military Academy, West Point, New York
Evan.Kenney@usma.edu

December 5, 2013

Abstract

Software defined networks (SDN) provide network administrators the ability to easily control the flow of traffic within their networks by seperating the data and control planes and centralizing the flow rules in a single system. Like any new technology SDN not only makes a task easier but opens up new vulnerabilities to the system. Due to the centralization of network control a potential attacker could control an entire network from this single location. With this vulnerability it becomes beneficial for an attacker to distinguish a traditional network from a SDN. In [6] this potential is demonstrated by exploiting the delay associated with creating new flow rules in the controller. "Insert cool name here" is a program that will identify these fingerprintting attempts by keeping track of packet pairs that have only a single header difference between the pairs. "Name" was tested using a pyretic SDN and ostinato traffic generator. Trials were run with traffic based on a standard home network, and an enterprise network running mail and web servers. The results from the trials show that...

1 Introduction

Software defined networks present unique vulnerabilities due to the seperation between the data plane and the control plane. Such threats include exploiting the communication between the controller and network switches and the centralization of the control planes allows for the entire network to be from a single spot by an attacker [2]. As software defined networks become more popular, attackers will want to distinguish the difference between SDN and traditional networks prior to beginning their exploitation and attack. It then becomes beneficial for a network administrator to be able to detect such network fingerprinting attempts to prepare for and prevent future attacks. Through recent research conducted be Seungwon Shin and Guofei Gu [6] it is possible to distinguish a SDN from traditional networks based on packet return times for established flow rules and new flow rules. THe structure of reactionary SDNs pushes packets that do not currently fit a flow rule to the controller to establish a new flow rule before being forwarded on to its destination; this creares a delay in the packet flow process. If a second packet is sent through with matching headers it will match the new rule that was generated from the last packet, allowing it to flow without the additional time delay. Shin and Gu show that as this process is repeated changing a single value within the packet header, the differences between the return times can indicate the delay from flow rule generation and identify the target as a node within a software defined network.

There are many current solutions to the need for intrusion detection systems deployed on networks today. These intrusion detection systems can be broken down into two main categories of systems [4]. The first is the Network Intrusion Detection System. These types of systems are employed at the network edge so that it is able to identify malicious traffic for the entire subnet. The other type of system is the Network Node Intrusion Detection System which function in the same way as the NIDS however it is placed on the end nodes of the network to iden-

tify malicious traffic going to a single host. Both of these systems are inadequate for identifying the Software Defined Network Scanner described by [6]. These systems utilize five main techniques to mitigate attack techniques [1]. The first is simple pattern matching, which examines the sequence of bytes contained within a packet. The second is Session Aware Pattern Matching. This takes simple pattern matching a step further by maintaining state information that allows the system the ability to interpret a complete conversation across multiple packets. These forms of pattern matching however are inadequate to identifying the scanning of SDNs. Another detection method is examining the context based signatures of a packet. Each form of network traffic has its own inherent vulnerabilities. By understanding the context of the conversation between machines the IDS can adjust the information that it looks for. This cannot effectively determine a SDN scan however because in the scan the packets themselves are the malicious part, not the data contained within them. This scan does not require any data to be included in the packets only single changes in the headers. The final two can be considered together for this overview. The first is Heuristic Analysis and the second is Traffic Anomaly analysis. Heuristic Analysis employs logical algorithms that determine the type of traffic traversing the network. With these algorithms developing a standard base overtime they identify deviations from the standard network statistics. Similarly Traffic Anomaly Analysis identifies deviations from a baseline; however these deviations are based on thresholds set on network traffic. These two methods also fall short in detecting the SDN scanner based on scope alone. These two methods require larger volumes of network traffic to be effective and the SDN scanner only requires two packets to be sent at a time.

In order to identify a scanning attempt on a SDN "name" will utilize the need for duplicate packets to test for the delay of adding flow rules. In order to effectively detect this attack you must identify a series of duplicate packets with single header field changes. The challenge that arises is distinguishing these packets from normal network traffic and minimizing the state requirements of the machine. When packets come through the system they are held for x seconds if no duplicate packets are received the record will be discarded. If a duplicate is received it are flagged and added to a watchlist. As more duplicates pass through the network and are appended to the watch list they are compared to the headers of the previous entries. When y pairs of packets are detected with a single change in the header a new rule will be created blocking traffic from that source address.

2 "Name"

There are three main parts of the "name" program. The first involves detecting the duplicate packet pairs that could be used to identify the delay found in SDN architecture. Once these duplicates are identified they are saved for comparison to future duplicate pairs. The second part of the program compares the duplicates for similarities. If there is only a single difference the source is flagged. After y matches to a similar pattern a rule is generated in the controller blocking future connection attempts from that source.

2.1 Packet Detection

Considerations for detection: what is normal network traffic? How long between packets and the possibilty of duplicates being sent? The manner in which a connection is broken (RST or FIN or timeout). Amount of packets system can feasibly hold without losing performance.

2.2 Packet Comparison

Does it have to be only a single difference between the packets?

2.3 Rule Generation

How long do rules remain in place? Leads to a potential DoS by intentionally having addresses identified as scanning attempts?

3 "Name" Evaluation

Model benign network traffic from DARPA data outlined in "Toward Comprehensive Traffic Generation for Online IDS Evaluation" Sommers.

4 Related Work

4.1 Detection

As SDN gains in popularity more people begin working on the problem of securing the networks against attackers. Pyretic is modular SDN framework that allows for multiple policies to process a single packet "citeMonsanto". This modular structure provides a level of abstraction that allows a network administrator the ability to create robust controller applications at a higher level language. The modular design of the framework provides a scalable network easily adapted and changed to include the addition of firewalls and other security programs. The "name" program fits within this structure as a seperate module in the network examining packets. "Name" utilizes the higher-level language of pyretic to implement its detection heuristic for fingerprinting attempts on the SDN. Another modular SDN framework comes from Shin et al. [5] and the development of FRESCO. FRESCO is a SDN security application development framework to simplify the development and deployment of openFlow security applications. In their work they outline several features of FRESCO that include scan detectors and BotMiner. The FRESCO framework allows developers to prototype security modules and create new security applications. Research done by Mehdi et al [3] has implemented several algorithims for effectively monitoring for network traffic anomalies in SDN. The methods that they explored were Threshold Random Walk with Credit Based Rate Limiting, Rate Limiting, Maximum Entropy Detection, and NETAD. In their comparison they implemented these strategies using NOX in C++. This research showed that it is possible to detect traffic anomalies produced by TCP Port Scans with both high and low flow rates as well as both TCP and UDP floods. Where these detection methods focus on scanning end hosts on a network to create a network map or identify potential vulnerabilities on an end host, "Name" focuses on the detection of the network itself. "Name" uses concepts from these detection algorithms to develop a new detection program to identify malicous attempy to fingerprint a network as a SDN.

4.2 Evaluation

To test the "Name" it is not enough to simply send the malicous packets that constitute the fingerprinting attempt as in normal network traffic these must be identified through the benign traffic. The benign traffic that is utilized is generated by "citeOstinato" based on the DARPA benign network traffic data. Using the generated traffic flow patterns from this software we are able to simulate a functioning network and identify the few packets in a stream of many that represent an attempt to identify the network as a SDN.

5 Future Work

References

[1] Corporate Headquarters. *Server Farm Security in the Business Ready Data Center Architecture*, chapter Cisco Network-Based Intrusion Detection - Functionalities and Configuration. Cisco Systems, 2006.

[2] Diego Kreutz, Fernando Ramos, and Paulo Verissimo. Towards secure and dependable software-defined networks. In *SIGCOMM Hong Kong*, 2013.

[3] Syed Akbar Mehdi, Junaid Khalid, and Syed Ali Khayam. Revisiting traffic anomaly detection using software defined networking. Master's thesis, National University of Science and Technology Islamabad, Pakistan, 2013.

[4] SANS Institute InfoSec Reading Room. *Understanding Intrusion Detection Systems*, 2001.

[5] Seugwon Shin, Phillip Porras, Vinod Yegneswaran, Martin Fong, Guofei Gu, and Mabry Tyson. Fresco: Modular composable security services for software-defined networks. In *SIGCOMM Hong Kong*, 2013.

[6] Seungwon Shin and Guofei Gu. Attacking software-defined networks: A first feasibility study. In *SIGCOMM Hong Kong*, 2013.