Towards a Reliable SDN Firewall

Hongxin Hu[†], Gail-Joon Ahn[‡], Wonkyu Han[‡] and Ziming Zhao[‡]
[†]Delaware State University, [‡]Arizona State University

1 Introduction

One primary goal of Software-Defined Networking (SDN) is to enable various network applications, which are basically network services, to run on the controller to manage the network directly by configuring packet-handling mechanisms in underlying devices. Consequently, when enterprises adopt OpenFlow for their networks, it is virtually inevitable that legacy security appliances, such as firewalls, have to be migrated to OpenFlow-based networks by re-designing and implementing them as compatible security applications. However, our study reveals that OpenFlow not only presents tremendous opportunities to networking, but also brings great challenges for building SDN firewalls as follows:

- **Examining Dynamic Network Policy Updates:** In an OpenFlow network, network states are dynamically updated and configurations are frequently changed. Thus, simply checking *flow packet violation* by monitoring packet-in behaviors in a firewall application is not effective, since *flow policy violation*, which indicates exiting flow policies violate the firewall policy induced by the changes of network states and configurations such as updating flow entries and firewall rules, should be detected and resolved in real time as well.
- **Checking Indirect Security Violations:** OpenFlow allows various `Set-Field` actions that can dynamically change the packet headers. *Adversaries* could take advantage of this feature to strategically enable flow rules that would evade security mechanisms (e.g. firewalls). In addition, flow rules may overlap each other in a flow table, indicating intra-table dependency of flow rules [3]. The rules in a firewall policy may also overlap each other [9]. These rule dependencies could also be leveraged by *malicious* OpenFlow applications to bypass firewalls.
- **Architecture Option:** *A centralized SDN firewall* can immediately enforce updated rules in the firewall policy to check security violations. However, when a flow policy violation is detected, it can only reject the new flow policy or flush resident flow policy that causes the violation. If only *partial* packets matching a flow policy violate the firewall policy, eliminating the flow policy may drop legal traffic. *A distributed SDN firewall* may directly resolve flow policy violations by propagating and enforcing the firewall policy at each *individual* entry (ingress switch) of the flow in the network. However, a distributed firewall needs a complicated revocation and repropagation mechanism [2] to handle dynamic policy updates.
- **Stateful Monitoring:** Currently, OpenFlow only provides very limited access to packet-level information in the controller [7]. In addition, the OpenFlow forwarding plane is almost stateless and unable to actively monitor flow status without the involvement of the controller [8]. Therefore, it is challenging to fully support stateful packet inspection in SDN firewalls.

The built-in firewall application in Floodlight [1] is limited to check flow packet violations and incapable of examining flow policy violations caused by dynamic network policy updates. VeriFlow [4] and NetPlumber [3] can be potentially used to detect SDN firewall policy violations, but they could not provide automatic, effective and real-time violation resolution. Pyretic's composition operators [5] could potentially resolve *direct* policy conflicts, but it cannot discover and resolve *indirect* security violations due to the lack of a flow tracking capability. FortNOX [6] attempts to identify *indirect* security violations, but it only conducts *pairwise* conflict analysis without considering *rule dependencies* within flow tables and firewall policies. In summary, there is a strong need to undertake the aforementioned challenges and limitations. This work attempts to seek a reliable network management for OpenFlow-based SDN environments to address such an emerging demand.

2 Approach Overview

The goal of this work is to design and develop *a systematic solution for building reliable firewalls that enable effective network-wide access control in SDNs*. We propose a comprehensive framework called FLOWGUARD to facilitate accurate detection as well as flexible resolution of firewall policy violations in dynamic OpenFlow networks along with a variety of toolkits for visualization, optimization, migration, and integration of SDN firewalls.

Violation Detection in SDN firewalls: The violation detection approach in FLOWGUARD detects violations by examining flow path space against firewall authorization space, and is capable of tracking flow paths in the entire network with respect to dynamic packet modifications, and checking rule dependencies in both flow tables and firewall policies.

To support network-wide access control in an OpenFlow network, an SDN firewall needs to not only check violations at the ingress switch of each flow, but also track the flow path and then clearly identify both the *original* source and *final* destination of each flow in the network. As a preliminary solution, we leverage NetPlumber [3] as a baseline for building the flow tracking mechanism, because it offers several features that can fulfill some of our requirements for tracking flows, such as support for arbitrary header modifications, automatic rule dependency detection, and real-time response. To calculate flow path space, we

abstract fields, which are needed for checking firewall policy violations, from the pattern expression of a flow rule to represent the space of corresponding flow path. In addition, we reorganize these fields with a (*source address*, *destination address*) pair to specify a *flow path space*. Then, we define three kinds of spaces for representing a flow path space: (1) *Incoming Space* represents *original* header spaces of packets that can pass through the flow path; (2) *Outgoing Space* represents *final* header spaces of packets after the packets pass through the flow path; and (3) *Tracked Space* represents *original source* address and *final destination* address of header spaces of packets that can pass through the flow path. For accurately detecting firewall policy violations, the dependency relations between "allow" rules and "deny" rules in the firewall policy should be decoupled. We propose a concept of *firewall authorization space*, which represents a collection of all packets either allowed or denied by the firewall rules. We then introduce a space partition approach, which represents rules with *header space* and performs various set operations on rules, to convert a list of firewall rules into two *disjoint* authorization sub-spaces, *denied authorization space* and *allowed authorization space*.

Once the space of a flow path and the firewall authorization space of the firewall policy are calculated, we identify violations through checking the *tracked* space of the flow path against the *denied* authorization space of the firewall policy. If these two spaces overlap each other, we call the overlapping space as the violated space, which indicates a firewall policy violation. There are two kinds of violations: (1) *Entire Violation* (the denied authorization space includes the whole tracked space); and (2) *Partial Violation* (the denied authorization space partially includes the tracked space).

Violation Resolution in SDN firewalls: For resolving various firewall policy violations in real time, we introduce a flexible and effective violation resolution mechanism. Figure 1 demonstrates how FLOWGUARD adopts various violation resolution strategies to resolve different firewall policy violations in terms of new flows and various update operations on both flow and firewall policies in OpenFlow-based networks. Since a firewall application can directly reject the new flows which violate the firewall policy, it would be straightforward to resolve *flow packet violations*. There are four resolution strategies used for handling *flow policy violations*: (1) *Dependency Breaking*. When a new flow policy is being added into the network switches, this single flow policy may not violate the firewall policy. However, the rules in this new flow policy may overlap with the rules of other existing flow policies. Since rule dependencies could cause unexpected changes in packet headers of flows, they may lead to new firewall policy violations. Note that such kind of violations can be also incurred by other changes of network states, such as modifying flow entries and updating firewall rules. We introduce two alternative mechanisms, *Flow Rerouting* and *Flow Tagging*, for breaking rule dependencies that cause violations; (2) *Update Rejecting*. Adding a new flow policy, or changing or deleting an existing flow policy, can cause new *entire* violation(s). Applying this strategy, the update operation is rejected directly; (3) *Flow Removing*. When updating (adding, changing, or deleting) a rule(s) in the firewall policy, the firewall application examines all existing flow policies applying the updated rule(s) and detect new *entire* violation(s). Using this strategy, all flow entries associated with a flow path, which entirely violates the firewall policy, are removed from the network switches; and (4) *Packet Blocking*. For any *partial* violation detected by the firewall application, this strategy can be applied. There may exist two ways to block packets of a flow. First, if the flow is a new flow, the firewall application only needs to block it in the ingress switch of the flow. Second, if the flow is an old flow, the firewall application needs to block the packets in both ingress and egress switches.

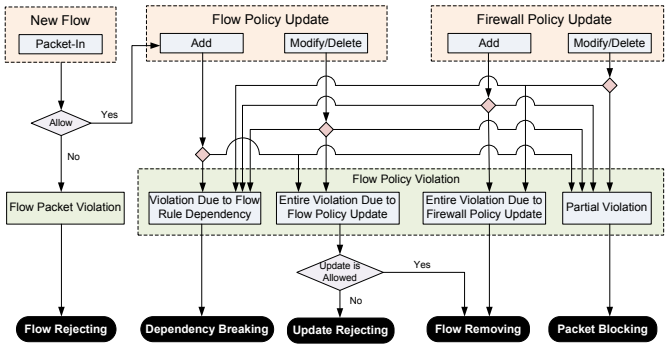

Figure 1: FLOWGUARD violation resolution mechanism.

3 Initial Results and Future Work

We have implemented a preliminary prototype of FLOWGUARD in Floodlight. Our experimental results indicated FLOWGUARD effectively addresses violations in a real-world network topology, and produces manageable performance overhead with *real-time* violation detection and resolution. As our on-going work, we are currently exploring various toolkits for supporting visualization, optimization, migration, and integration of SDN firewalls in the FLOWGUARD framework. We will also study a more sophisticated flow tracking mechanism in the FLOWGUARD framework. We will further develop and integrate stateful packet inspection and analysis modules in the FLOWGUARD framework to support the implementation of stateful SDN firewalls. Furthermore, we plan to integrate our conflict detection and resolution solution into popular SDN controllers to build a robust security enforcement kernel for the controllers.

References

[1] Floodlight: Open SDN Controller. `http://www.projectfloodlight.org`.
[2] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. In *Proceedings of CCS*, 2000.
[3] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte. Real time network policy checking using header space analysis. In *Proceedings of NSDI*, 2013.
[4] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey. Veriflow: verifying network-wide invariants in real time. In *Proceedings of NSDI*, 2013.
[5] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker. Composing software-defined networks. In *Proceedings of NSDI*, 2013.
[6] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu. A security enforcement kernel for openflow networks. In *Proceedings of HotSDN*, 2012.
[7] S. Shirali-Shahreza and Y. Ganjali. Flexam: Flexible sampling extension for monitoring and security applications in openflow. In *Proceedings of HotSDN*, 2013.
[8] H. Song. Protocol oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane. In *Proceedings of HotSDN*, 2013.
[9] L. Yuan, H. Chen, J. Mai, C. Chuah, Z. Su, P. Mohapatra, and C. Davis. Fireman: A toolkit for firewall modeling and analysis. In *Proceedings of IEEE S&P*, 2006.