Challenges and Preparedness of SDN-based Firewalls

Vaibhav Hemant Dixit, Sukwha Kyung,
Ziming Zhao, Adam Doupé, Yan Shoshitaishvili and Gail-Joon Ahn
Arizona State University, Tempe, AZ, USA
Email: {vdixit2, skyung1, zmzhao, doupe, yans, gahn1}@asu.edu

ABSTRACT

Software-Defined Network (SDN) is a novel architecture created to address the issues of traditional and vertically integrated networks. To increase cost-effectiveness and enable logical control, SDN provides high programmability and centralized view of the network through separation of network traffic delivery (the "data plane") from network configuration (the "control plane"). SDN controllers and related protocols are rapidly evolving to address the demands for scaling in complex enterprise networks. Because of the evolution of modern SDN technologies, production networks employing SDN are prone to several security vulnerabilities. The rate at which SDN frameworks are evolving continues to overtake attempts to address their security issues.

According to our study, existing defense mechanisms, particularly SDN-based firewalls, face new and SDN-specific challenges in successfully enforcing security policies in the underlying network. In this paper, we identify problems associated with SDN-based firewalls, such as ambiguous flow path calculations and poor scalability in large networks. We survey existing SDN-based firewall designs and their shortcomings in protecting a dynamically scaling network like a data center. We extend our study by evaluating one such SDN-specific security solution called FlowGuard, and identifying new attack vectors and vulnerabilities. We also present corresponding threat detection techniques and respective mitigation strategies.

ACM Reference Format:
Vaibhav Hemant Dixit, Sukwha Kyung, Ziming Zhao, Adam Doupé, Yan Shoshitaishvili and Gail-Joon Ahn. 2018. Challenges and Preparedness of SDN-based Firewalls. 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, 6 pages. https://doi.org/10.1145/3180465.3180468

1 INTRODUCTION

In traditional networks, both control and data planes are tightly integrated in physical devices. To specify routing policies in traditional networks, network administrators must maintain forwarding rules individually in all switches and routers in the network. In contrast, SDN has brought significant changes to how networks function by decoupling the control plane from data plane. The decoupling abstracts the higher level functionality and moves the intelligence of network configuration to a centralized controller. This innovation has influenced both industries and academic institutions to persistently work towards adaptation and evolution of SDN. The two main advantages of SDN (central programmability and visibility) tremendously improve cost-effectiveness and ease of maintenance in these complex networks.

The security of conventional networks is often dependent on legacy firewalls and intrusion detection systems (IDS). However, both of these security mechanisms are ill-suited for protecting SDN environments. Traditional firewall functions are based on static rule set, which lacks fine granularity. Filtering the traffic is performed by matching every packet against *allow* or *deny* policies, using only source, destination IP addresses and TCP/UDP ports. Thus, the existing static and coarsely grained security mechanisms lack adaptivity and scalability. Consequently, it is critical that an SDN-based firewall solution leverages the functions of controller to enforce the security of SDN.

Recently, there have been numerous studies to address security concerns that are specific to SDN. Some of these work propose SDN-based firewalls that work in collaboration with controller and provide a centralized security framework [5, 7, 13]. However, the feasibility of these firewall solutions on enterprise networks is uncertain as these systems are often evaluated in simulated environments, such as mininet [2], which are inadequate to demonstrate their efficiency in real world networks. Thus, it is critical to review current proposed SDN-based firewall solutions, discover the shortcomings, and identify ways to improve them for SDN deployment in real-world networks.

In this paper, we revisit the existing SDN-based firewall designs to assess their readiness in protecting large-scale networks. We identify vulnerabilities in these existing designs for detecting security policy conflicts in an SDN. Along with surveying shortcomings of the existing work, we also revamp one of the accepted firewall designs, called *FlowGuard* [7], as our case study. We identify shortcomings of the current *FlowGuard* design and propose improved measures for better performance and accurate conflict resolution. We also observe that assessment of SDN security solution on a simulated environment does not guarantee its credibility in a real world network. Therefore, in our evaluation, we use an enterprise ready controller, OpenDayLight [3] and incorporate ScienceDMZ [6] network at Arizona State University, as the target underlying topology.

2 BACKGROUND

2.1 OpenFlow

The decoupled control plane logic in SDN makes the forwarding decisions and enforces policies. This information sharing between SDN controller and switches takes place using APIs provisioned by OpenFlow protocol. OpenFlow enabled switches maintain the

SDN-NFV Sec'18, March 19–21, 2018, Tempe, AZ, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5635-0/18/03…$15.00
https://doi.org/10.1145/3180465.3180468

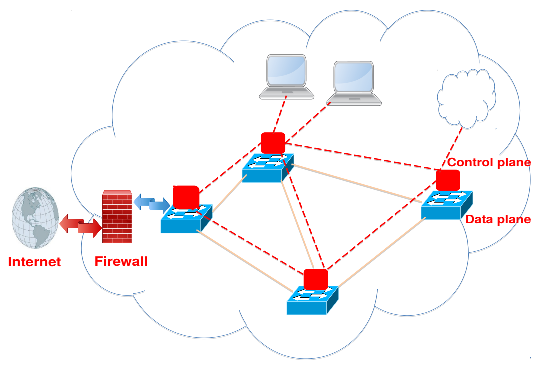


Figure 1: Firewall in a `traditional` network.

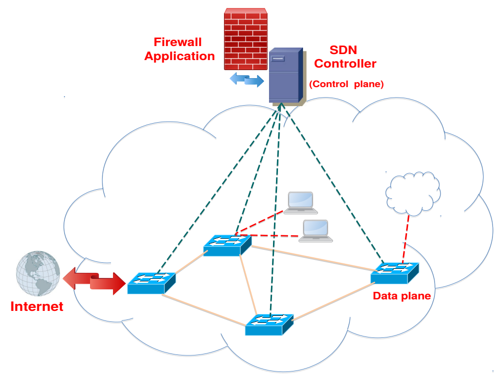


Figure 2: Firewall in an `SDN` network.

flow routing policies in flow tables. A typical rule present in a flow-table has 3 different field sets for packet handling: Match, Action and Statistics. The match set allows packet filtering based on header fields. After a successful match, the packet undergoes respective actions. As per the OpenFlow standard, the first packet of a network flow is typically sent to the controller, which then inspects the packet headers and decides actions to be taken for rest of the flow. This action can be either to *drop* or *forward*. The central visibility in SDN is of great advantage as controller can extrapolate the future impact of traffic on any other node in the network.

2.2 SDN Controller

In an SDN, a controller behaves much like a kernel of network operating system. Hardware based services of traditional networks like firewall and load balancer run as software applications within a controller. SDN-Controllers are capable of much more than a traditional control plane (in which a particular node remains oblivious of network activity on any other node in the network). Some light weighted controllers are used for academic testing and prototyping, whereas enterprise-ready ones are complicated in their design and functionality. Due to the lack of implementation standards, these controllers are implemented differently and often leave security loopholes in their implementation and design. Two of the most popular SDN controllers on the market are OpenDayLight (ODL) and Open Network Operating System (ONOS). Both are open-source projects hosted by The Linux Foundation and differ greatly in their internal designs and implementation.

2.3 Firewalls

A conventional firewall, as shown in Figure 1, is a system that protects trusted internal networks from the untrusted and unsecured outside world, such as the Internet. Firewalls filter and scan the traffic in and out of the network. In the SDN paradigm, the positioning, functionality and capabilities of a firewall assume multiple adaptations. Generally, they can be seen as an application running in the control plane, as shown in Figure 2.

Security policies (such as Access Control Lists) for a network are defined centrally at the controller. The firewall converts these policies into flow rules which are installed in the network by a flow programming application. The placement of firewall logic on the control plane necessitates a novel design, tailored to protect SDN. Firewall with a centralized view of a network can functionally inspect local traffic to detect internal policy violations. Additionally, an SDN-based firewall can provide resolutions for the conflicts at runtime. Unlike traditional firewalls, which filter and scan every packet coming and going through it, an SDN-based firewall analyzes the first packet of each flow and installs rules for the rest of that flow in relevant switches. Moreover, since the network programmability is achieved from a software, multiple privileged sources (applications, users and administrator) can asynchronously install the routing rules and policies. Thus, SDN-based firewalls should implement *concurrent* software listeners to get updates about policy and flow modifications done on the network and verify the impact of such changes to prevent any violations.

3 POTENTIAL OF SDN-BASED FIREWALL

As described in Section 2.3, an SDN-based firewall has a potential to be a holistic network security solution by leveraging centralization, high scalability and abstraction at the control plane. To harness these advanced capabilities, we identify seven vital features that a firewall should support to leverage the advanced network capabilities enabled by SDN. We then survey existing techniques in SDN-based firewalls to evaluate their readiness for securing an SDN network.

(1) **Centralized Policy Enforcement:** The basic functionality of any firewall is to enforce security policies in a network. Thus, an SDN using traditional firewall hardware as a node in the network is not harnessing the advanced functions enabled by SDN: centralization and high programmability. Centrally-specified policies can range from having fine granularity to being largely coarse. To enforce security on a network, the firewall converts these policies into flow rules which later get installed in the network.
(2) **Centralized Flow Tracking** (end to end flow control)**:** The rules installed in the network direct packets belonging to a particular flow from source to destination. Thus, any path taken by the packets belonging to a flow, can contain multiple flow rules from one or more switches. To effectively enforce the policy, SDN-based firewall keeps track of the flow path space. Merely confirming the policy on ingress and

Table 1: Comparison of SDN based Firewalls

Firewall	Controller	Centralized Flow Tracking	Centralized Conflict Detection	Multi-Tenant support	Auto Priority handling	Violation Resolution	Concurrent updates	Stateful	Year
Ethane[1] [5]	Ethane	×	✓	×	×	×	×	×	2007
FortNOX [13]	NOX	×	✓	×	✓	×	×	×	2012
FlowGuard [7]	FloodLight	✓	✓	×	×	✓	×	×	2014
FW over SDN [15]	POX	×	✓	×	×	×	×	×	2014
SE-FloodLight[2] [12]	FloodLight	×	✓	×	✓	✓	×	×	2015
AuthFlow [11]	POX	×	✓	×	×	×	×	×	2016
Reactive stateful FW [16]	RYU	×	✓	×	×	×	×	✓	2016

egress port of a switch cannot guarantee a successful conflict detection, let alone resolution. Furthermore, building a flow path space by leveraging centralization guarantees to take into account the changes that OpenFlow allows in header fields of an in-route network flow.

(3) **Conflict Resolution:** Upon detecting a violation in the rules when a new policy or a flow rule is being added, a comprehensive firewall should provide a conflict resolution too. Different instances, FlowGuard [7] and SE-Floodlight [12], prove that just allowing or denying a network update is not an efficient firewall design. If there exists a path in the network without a conflicting entity, the firewall should dynamically perform resolution to give an updated path. Similarly, the new update may only partially cause a violation. In this case, the firewall should dissect the update into allowed/denied sets and process the allowed set. Therefore, a comprehensive SDN-based firewall must support dynamic conflict resolutions.

(4) **Automatic Priority Handling:** Different sources (for e.g., SDN controllers, applications, and administrator) update the security policies in a network. Assigning an authorization level to each source is important to prevent low priority updates from overriding crucial network policies. Moreover, an intelligent firewall should process these authorization issues automatically and provide a robust user experience.

(5) **Multi Tenant Support:** A complex network, such as datacenters, can contain more than one sub-networks and multiple tenants for different services. Handling multiple tenants is not a special concern in a traditional networks as different firewalls are dedicated to each tenant network. However, in SDN, the controller has a centralized view of entire network topology and the firewall is responsible for centralizing the enforcement of security policies. SDN-based firewall should be capable of generating a distinction between subnetworks even if their address ranges overlap.

(6) **Scalability and Concurrency:** In a dynamic and scalable network, updates of security policies are not always sequential. Multiple user threads or applications running inside an SDN controller make concurrent updates to modify firewall policy or flow policy. In case of conflicting updates, lack of handling of concurrency could lead to low priority rules being handled before overlapping higher priority rules. The issue with concurrent updates becomes a problem when firewall applications are deployed in enterprise networks, where there are multiple pipelines which access and update the same configuration data stores.

(7) **Stateful Support:** Maintaining the states of active connections gives a definite advantage to the reliability of a firewall. Connection states can be obtained using information from OpenFlow communication. Information about the connection states should be formulated into time-bound flow rules for a network, providing finer granularity in firewall rules and a more accurate violation detection mechanism.

4 SURVEY OF SDN-BASED FIREWALLS

Security in SDN has become a popular research topic these days. However, this focus arose almost a decade after the inception of Software Defined Networking. The very first work on security of SDN [5] focused only on implementing access control lists within the SDN controller. Moreover, every work on SDN-based firewall focuses on certain aspect of SDN security but misses other important measures as shown in Table 1. Apart from the slow progress in SDN-security, there are other factors responsible for the absence of a comprehensive SDN-based firewall solution. First, there are no operation and design standards for SDN controllers. Since an SDN-based firewall is a software application running inside the controller, unless there exist norms for the design of an SDN controller, an agnostic SDN-based firewall is a distant dream. On the contrary, there exists a well maintained standard for the OpenFlow protocol [4]. It is intensely adapting to the new requirements. Significant changes in each of its revision demand correspondent changes in the firewall design as well. Finally, the existing SDN-based firewalls are developed and tested on simulated environments. Therefore, the efficacy of a firewall in an enterprise sized network often remains undetermined. Due to such factors, SDN-based firewalls become obsolete soon after they are proposed.

The earliest SDN-based security solution, **Ethane** [5] accompanied the very pioneering work on SDN architecture. It showcased that security policies, instead of being defined individually on network entities, can be centrally enforced. No mitigation or threat prevention measures were discussed as the focus was on network programmability. With a lack of standard OpenFlow protocol, the policy enforcement was done only on a per-node basis via the

[1]ETHANE is a SDN architecture with inbuilt security

[2]Conflict resolution is not done for policy violations on the network but between competing controller applications

controller. With increasing adaptation of SDN, security loopholes regularly surfaced, with various work focusing on attack detection and mitigation in SDN [14].

In **FortNOX** [13], security kernel has been proposed for the NOX controller. FortNOX firewall prevents unauthorized rule installations by assigning control plane applications and users to an authorization level. For a new policy, the decision to reject the update depends upon the installer's authorization level. FortNOX, however, is not a comprehensive firewall solution. It lacks accuracy, as the rule conflict analysis is done in alias sets. The pairwise comparison ignores the rule dependencies and thus, provides an inaccurate result in complex networks with interdependent flow rules. It also lacks an ability to provide a fine-grained conflict resolution. FortNOX is designed as an extension for the NOX controller, which is significantly different from enterprise controllers, and does not handle the challenges that arise when several sources race to install and modify rules concurrently.

An SDN-based firewall **FLOWGUARD** [7] is proposed as a prototype solution built on FloodLight [1] controller. To detect flow policy violations, the firewall pulls network topology information from the data plane. It leverages NetPlumber [8] library to build a plumbing graph (logical snapshot of a network). Using a tracked space of a flow in the network comprising of initial source and final destination addresses, it centrally validates the security policies in entire network and provides dynamic resolutions.

To demonstrate a software based firewall solution, a simple SDN-based firewall application [15] is proposed and deployed on the POX OpenFlow-based controller. This is an experimental project to prove the redundancy of hardware-based firewalls in an SDN network. Only the usability of the firewall is emphasized by providing a command-based user interface. Security-specific features such as flow path space calculation and violation resolution are not given much importance.

SE-FloodLight [12], an extension of FortNOX, is proposed as a security enforcement kernel for the FloodLight controller. SE-Floodlight provides resolutions to conflicts that surface when multiple applications are deployed in the same network by pre-signing an application's class, which gets digitally verified by firewall at runtime. However, traffic engineering (such as tracking the path of a flow to detect violations) is not done. Consequently, complex issues which arise due to indirect and partial conflicts will go undetected.

An access control mechanism based on authentication at layer-2 is proposed in**AuthFlow** [11]. To prevent the address spoofing and communication overhead, the security mechanism is placed in data link layer. The approach however conflicts with the principle *software-based* model of SDN by employing decentralized security techniques. Realizing a need of an extra hardware (radius server) further increases the operational and configuration costs.

A reactive SDN-based firewall [16] is proposed as a stateful solution. Policy enforcement is done based on the state of active connections in a network, mined in a local state table. Yet, the new generation requirements of the firewall (dynamic resolution and end to end flow path calculations) are not dealt with.

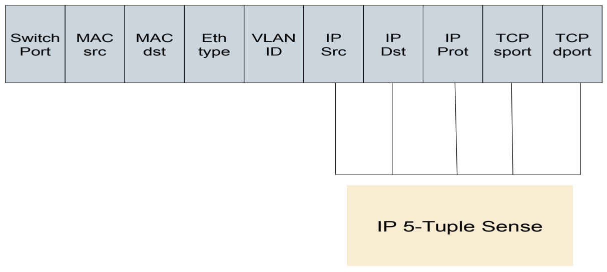


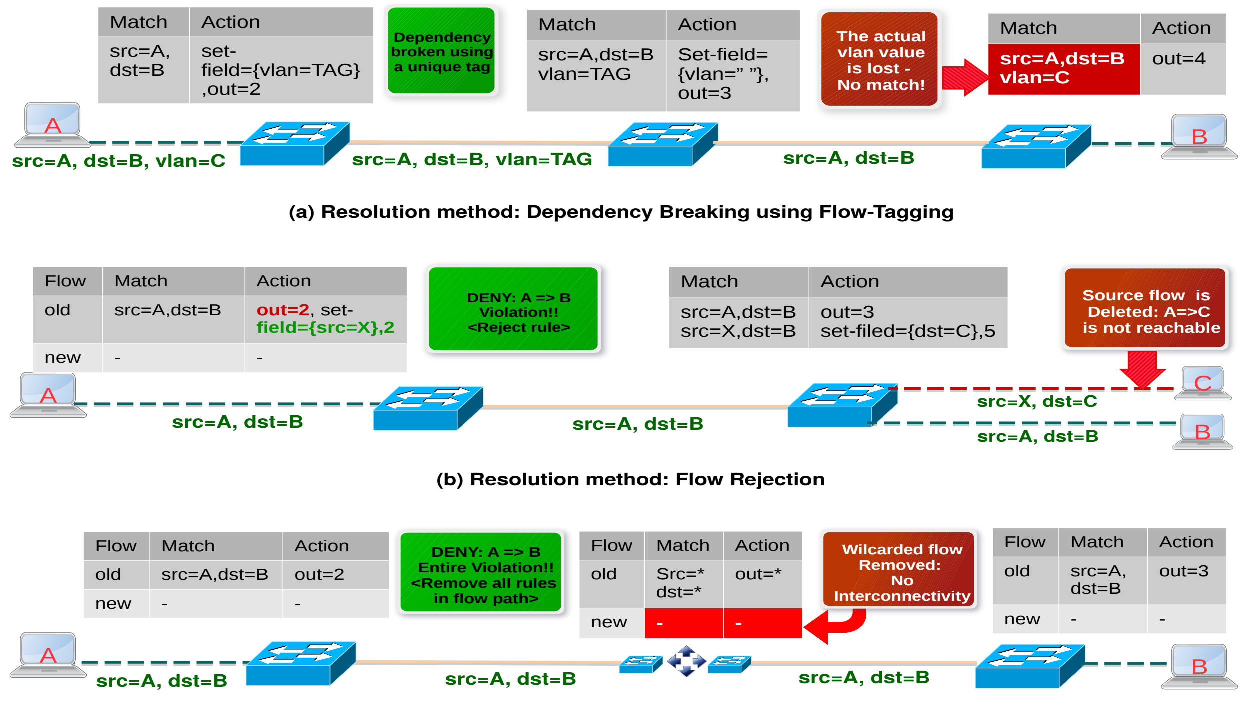
Figure 3: OpenFlow `Ten-Tuple` format

5 FIREWALL CASE STUDY: FLOWGUARD

To discover the challenges faced by SDN-based firewalls, we deploy one of the existing solutions, FlowGuard, on a real and traffic-rich network topology (ScienceDMZ), which is managed by a production ready controller (OpenDayLight). We examine the number of SDN specific features (centralization, scalibiltiy, abstraction and dynamic responsiveness) the existing solutions harness and select FlowGuard for an extended evaluation. Our decision of selecting FlowGuard as a representative study of the challenges faced by SDN-based firewalls is based on the quantified scale (Table 1) on which we rate the existing solutions. First, to detect policy conflicts, FlowGuard leverages centralization by taking into account the entire flow path space, unlike other solutions which inspect packet headers at ingress and egress ports only. Second, FlowGuard is relatively more agnostic in nature than other solutions as it uses open-source Header Space Analysis framework [9] to calculate rule interdependencies. In our experiments, we find multiple issues that FlowGuard faces when built on the latest infrastructure and tested on the actual complex network. These findings, along with possible mitigation approaches, are discussed below:

(1) **Ambiguous Flow Path Space:** We find that the IP-5 Tuple based approach used for conflict detection (See Section 4) is prone to ambiguous flow path calculations. A firewall with a centralized view of two or more different tenant sub-networks cannot successfully distinguish one tenant from others based solely on IP 5-Tuples. In a tenant-based network (for e.g., in data center), multiple disconnected tenants use similar network configurations. This leads to hosts having identical IP 5-tuple addresses in two disconnected sub-nets. On an OpenStack cloud with multiple different tenants, FlowGuard produces conflicting flow path spaces which further leads to undesired resolutions. In order to prevent the conflict, we propose to use a more fine-grained, OpenFlow 10-Tuple, as shown in Figure 3. Apart from the existing fields in an IP 5-tuple sense, an OpenFlow 10-tuple also includes a physical layer-2 port, protocol type, vlan id and ethernet type. With this fine-grained set of header fields, it is not possible to have duplicate route identification. In addition, we can also prevent the ambiguity in flow path space calculation.

(2) **Conflicting Priority Handling:** In a conventional firewall, priority of the rules are implicit based on their ordering. In



(a) Resolution method: Dependency Breaking using Flow-Tagging

(b) Resolution method: Flow Rejection

(c) Resolution method: Flow Removal

Figure 4: Challenges faced by conflict resolution mechanisms in FlowGuard

an SDN-based firewall, the firewall rules and flow rules can maintain their own set of priorities. FlowGuard divides the firewall rule space into two disjoint sets of *allow* and *deny* rules. However, this partitioning is performed either by considering implicit rule priorities (like in *iptables*) or not considering them at all. Thus, the absence of an intelligent firewall decision logic (logically capable of assigning role-based priorities) can violate the administrators actual intent of policy enforcement. This can be addressed by taking the priorities of policies in account from multiple sources, including IDS, feedback module and network administrator.

(3) **Coarse Conflict Resolution:** Upon successful detection of policy conflicts, an SDN-based firewall should also be able to dynamically resolve the violations between flow and firewall policies. Different conflict resolution methods have been proposed in FlowGuard [7], depending on the violation. In our deployment, we discover that although these proposed approaches solve policy conflicts successfully, they come at the cost of network's availability and accuracy as shown in Figure 4. The new or modified rules prevent unauthorized traffic in the network, but in some cases, also deny communication between authorized hosts:

Challenge 1. Different flow rules matching the same header fields can have inter-dependency issues. This can lead to communication between unauthorized hosts as mentioned in FlowGuard [7]. An approach to break the inter-dependency is by tagging the header fields, and thus creating distinct network flows. This approach, however, leads to the loss of critical information from the original packet headers. For example, vlan-tagging is used wherein extra tag action is added in a dependent flow. In our testing of multi tenant networks, *vlan* field plays an important role in steering traffic between different subnets. By tagging *vlan* field with a hard coded value, the dependency is broken but the subsequent flow rules which are intended to match on the vlan field, malfunction. Any flow rule which is not part of dependency process (but is using the tagged field to process the packet further in the network) may fail to match flow in which packet headers are now changed.

Challenge 2. When a firewall detects that there is an *entire violation* (i.e., the flow path space completely violates the security policy) in the existing flow path, all the rules present in the flow path are removed from corresponding switches [7]. The tracked path, comprising of only source address from the incoming packets and the destination address from the outgoing packets is matched against the firewall rule policies. In our testing, we find that this approach is prone to loopholes and comes at the cost of network's availability. For example, if a violating tracked space consists of rules from 5 switches, and rules in the third switch (intermediate node) are only wildcarded rules, all the rules including the wildcared rule are deemed incorrect and removed. It leads to denial of any authorized communication that the wildcarded rule was previously allowing in the network. To prevent resolutions from losing availability of the network, higher-priority rules with finer granularity should be installed. The wildcarded flows require special attention as they are responsible for both allowed and denied traffic passing through them.

Challenge 3. In the new OpenFlow protocol, actions per match in a flow rule can be chained. It means when a packet of a flow is matched by a flow rule in a switch, the same packet can undergo multiple relevant actions. Upon detecting a violation in a flow rule, deleting the entire flow rule deletes all the violating *and* non violating actions within it. Therefore, deletion of a flow rule requires careful examination. The action set within a rule can only partially violate the policy, just as entire flow rule can have partial violations.

(4) **Performance Issue:** The response time of FlowGuard substantially increases as the network scales. This is because the conflict detection algorithm is based on dynamically propagating a sample flow in the logical snapshot (plumbing graph) of a network. Such a graph can potentially have more than 100 nodes for a network of just 3 switches due to the fact that a node in a plumbing graph is a **match and action set** present in a flow rule. Various chained action sets in each flow, and several such flows, make for a complicated plumbing graph. When we tested the firewall on a convoluted topology of flow rules, the violation detection reached an order of seconds. This time is large enough for an unauthorized communication to take place before a conflict is successfully resolved by a firewall.
As an improvement, we propose to maintain a ***reachability map*** which is updated during the initialization process of a firewall. For future updates, full propagation of a sample packet from source to destination can be avoided. For this, we tested multiple hosts connected via 5 OpenFlow switches in a linear topology. When a new flow rule update for an intermediate switch in the flow path is made, a propagation check (for policy enforcement) from first node to the last was ignored. Instead, with the information that 2/3 of the graph edges do not undergo a change, the propagation check happened only for the overlapping nodes in the graph. Thus, the performance is improved by reducing the propagation time in proportion to number of overlapping edges.

(5) **Disregard for concurrent updates:** In our testing, we found issues when multiple threads make concurrent updates on firewall policy or flow policy. In case of conflicting updates, lack of concurrency handling leads to low priority rules with a different action on same match being handled before the higher priority rule. For example, if an administrator installs a new flow rule in the network but a countermeasure engine (at the same time) installs a conflicting action on the same match, the decision of the asynchronous firewall depends on which thread is executed first. The conflict detection and resolution algorithm should provide thread-safe security enforcement and take into account role-based access control. This can be achieved by using a privilege score assigned to each authorized application and user as an input to the algorithm resolving simultaneous action conflicts.

6 CONCLUSION AND FUTURE WORK

In this work we have juxtaposed existing SDN-based firewalls against each other to inspect their readiness to be deployed in enterprise and large scale networks. We have identified various metrics that an SDN-based firewall solution should address. Seven different firewalls are then compared and evaluated against these metrics. As a case study, we have deployed FlowGuard [7] on ScienceDMZ network and discovered underlying vulnerabilities in protecting a large scale, complex network. The challenges are individually discussed and possible mitigation measures are proposed.

With the advent of SDN, attacks can be fine-tuned to target different layers of SDN [10]. We want to extend the firewall to incorporate advanced features which protect the SDN network from various attacks targeting these layers. We plan to introduce an agnostic and comprehensive firewall solution of our own with such advanced capabilities to protect an SDN network from intrusions before their occurrence.

ACKNOWLEDGMENTS

This work was partially supported by grants from National Science Foundation (NSF-ACI-1642031) and a grant from the Center for Cybersecurity and Digital Forensics at Arizona State University.

REFERENCES

[1] Floodlight: SDN Controller. http://www.projectfloodlight.org
[2] Mininet: An Instant Virtual Network on your Laptop (or other PC). mininet.org
[3] OpenDayLight. https://www.opendaylight.org/
[4] OpenFlow Switch Specification 1.4.0. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf
[5] M. Casado, M. J. Freedman, J. Pettit, J. Luo, and N. McKeown. 2007. Ethane: Taking Control of the Enterprise. In *Proceedings of the ACM SIGCOMM, August, 2007, Kyoto, Japan.* (2007).
[6] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski. 2013. The Science DMZ: A network design pattern for data-intensive science. In *2013 SC - International Conference for High Performance Computing, Networking, Storage and Analysis.*
[7] H. Hu, W. Han, G. J. Ahn, and Z. Zhao. 2014. FLOWGUARD: Building Robust Firewalls for Software-Defined Networks. In *Proceedings of the Third ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking* (Aug, 2014).
[8] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte. 2013. Real Time Network Policy Checking using Header Space Analysis. In *10th USENIX Symposium on Networked Systems Design and Implementation* (2013).
[9] P. Kazemian, G. Varghese, and N. McKeown. 2012. Header Space Analysis: Static Checking for networks. In *10th USENIX Symposium on Networked Systems Design and Implementation* (2012).
[10] D. Kreutz, F. M. V. Ramos, and P. Verissimo. 2013. Towards Secure and Dependable Software-Defined Networks. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking* (2013).
[11] D. M. F. Mattos, L. H. G. Ferraz, and O. C. Duarte. 2016. AuthFlow: Authentication and Access Control Mechanism for Software Defined Networking. In *Annals of Telecommunications December 2016, Volume 71* (2016).
[12] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran. 2015. Securing the Software-Defined Network Control Layer. (2015). http://www.csl.sri.com/users/porras/SE-Floodlight.pdf
[13] P. Porras, S. Shin, V. Ygneswaran, M. Fong, M. Tyson, and G. Gu. 2012. A Security Enforcement Kernel for OpenFlow Networks. In *Proceedings of the HotSDN, August 13, 2012, Helsinki, Finland.* (2012).
[14] S. Scott-Hayward, S. Natarajan, and S. Sezer. 2016. A Survey of Security in Software Defined Networks. In *IEEE Communications Surveys and Tutorials* (2016).
[15] M. Suh, S. H. Park, B. Lee, and S. Yang. 2014. Building Firewall over the Software-Defined Network Controller. In *Proceedings of 2014 16th International Conference on Advanced Communication Technology (ICACT)* (2014).
[16] S. Zerkane, D. Espes, P. L. Parc, and F. Cuppens. 2016. Software Defined Networking Reactive Stateful Firewall. In *The 11th International Conference on Risks and Security of Internet and Systems* (2016).