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).