Poster: OFX: Enabling OpenFlow Extensions for Switch-Level Security Applications

John Sonchack
University of Pennsylvania
jsonch@cis.upenn.edu

Adam J. Aviv
United States Naval Academy
aviv@usna.edu

Eric Keller
University of Colorado, Boulder
eric.keller@colorado.edu

Jonathan M. Smith
University of Pennsylvania
jms@cis.upenn.edu

ABSTRACT

Network Security applications that run on Software Defined Networks (SDNs) often need to analyze and process traffic in advanced ways. Existing approaches to adding such functionality to SDNs suffer from either poor performance, or poor deployability. In this paper, we propose and benchmark OFX: an OpenFlow extension framework that provides a better tradeoff between performance and deployability for SDN security applications by allowing them to dynamically install software modules onto network switches.

1. INTRODUCTION

SDNs are a promising deployment target for network security applications [6, 7, 4]. However, network security applications often need to analyze and process traffic in more advanced ways than SDN data planes (*i.e.* switch forwarding logic) allow. OpenFlow, the de-facto SDN standard, has many noted data plane limitations [5, 9], for example.

One common approach to overcoming data plane limitations is to add functionality as a library on the centralized network controller, such as Fresco [8] has proposed (Figure 1 illustrates this approach). However, the path between a switch's forwarding logic and its interface to the controller has bottlenecks, which add latency and limits the bandwidth of traffic that is processed with the custom functionality. The alternate approach is to embed the new functionality into the data plane of custom designed switches, as illustrated in Figure 2. Avant- Guard [9] takes this custom switch approach, which allows for new functionality to be applied to every packet at high speeds, but presents a significant deployment challenge requiring network operators to deploy new switches.

Many SDN security applications would be better served by an approach that takes a middle ground between performance and deployability, where their custom functionality was implemented as software to provide easy deployability, but ran on network switches and was tightly coupled to the data plane, to provide sufficient performance. Modern network switches have the general purpose hardware necessary to perform local packet processing in software, and even run operating systems with standard kernels, such as Open Network Linux [2]. These resources are largely untapped for security applications; they are mostly used to provide simple, hard coded functions such as generating ARP responses.

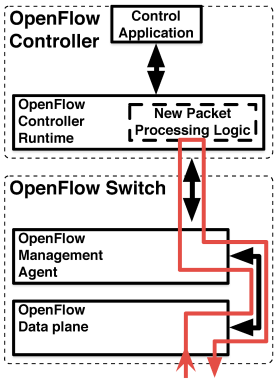
Ideally, a SDN security application would be able to dynamically install software functions onto network switches, using a framework that was integrated into the centralized control platform.With this approach, a SDN security application could leverage the network wide view that SDN control platforms provide to intelligently select which packets to process with the software functionality it installed onto the switches. For example, a security application could install a software module onto ingress switches that analyzes the first few packets of each new TCP connection to determine whether or each TCP connection has completed its handshake. As we discuss in Sections 3 and 4, this functionality can protect other control applications from SYN flood denial-of-service attacks. Previously systems that protected against this class of threats, such as AvantGuard [9], required custom switches and could not be deployed onto existing OpenFlow hardware.

In this paper, we introduce OFX (for OpenFlow eXtensions), a framework that meets these design goals. OFX, depicted in Figure 3, allows an OpenFlow control application to easily load stack-independent data plane extension modules into OFX software agents running on dedicated OpenFlow switches [3] that are already widely deployed. We show how OFX extension modules can add security functionality to the network, inspired by AvantGuard [9], but without requiring customized data plane hardware. In experiments, our OFX extension module allowed networks using OpenFlow switches with standard hardware data planes to withstand over two orders of magnitude more attack traffic.

2. OFX ARCHITECTURE

The OFX framework has four main components (see Figure 3). First the *OFX extension modules* specify the new data plane functionality. The *OFX library* provides control applications with an interface to the functionality defined in a module and handles all module related controller to switch communication, including loading the module onto switches. When the controller sends a message defined in an OFX extension module to the switch, the *OFX Switch Agent* receives it and calls the message handler from the appropriate extension module. The OFX Switch Agent maintains connections to the controller, the standard OpenFlow Management Agent that

CCS'15, October 12–16, 2015, Denver, Colorado, USA.
ACM 978-1-4503-3832-5/15/10.
DOI: http://dx.doi.org/10.1145/2810103.2810120 .



Packet paths drawn in red.

Figure 1: Adding functionality in the controller.

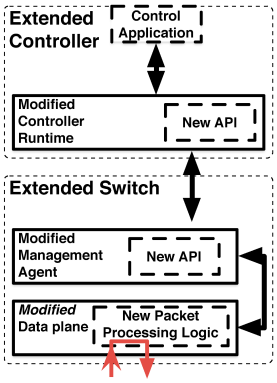


Figure 2: Adding functionality in switch data planes.

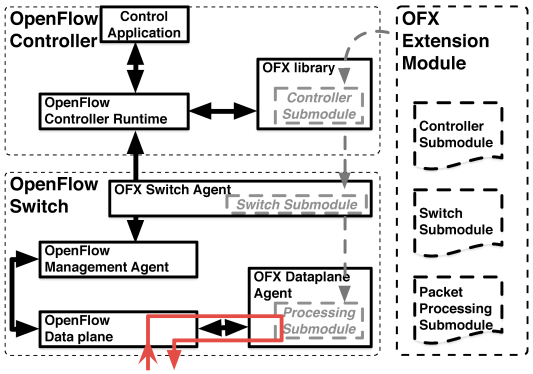


Figure 3: Adding data plane functionality with OFX, which loads extension modules onto agents that run on a switch's general purpose hardware.

runs on the switch, and the *OFX Data Plane Agent*. This component of OFX maintains a lower layer socket based connection to the data plane, and does any packet processing required by OFX modules.

To develop an OFX module that can be used in an OpenFlow security application, a developer needs to write:

Message Definitions both for controller to switch messages (*i.e.* to invoke the new functionality), as well as switch to controller messages (*i.e.* to provide replies to the controller).

Message Handlers that are triggered when a module message arrives at the switch or controller.

Packet Processing Functions that contain logic that needs to run efficiently in the data plane of the switches.

Redirection Rules that specify which traffic flows should be directed to the packet processing functions, and what should happen to packets after they are returned. A module can dynamically generate, install, and remove these rules.

An OpenFlow security application loads an OFX module onto its network's switches by calling a function in the OFX library that sends the module to each OFX Switch Agent running in the network. Once a switch agent recieves the OFX module, it registers all of the module's message handlers and loads the packet processing functions into the OFX Data Plane Agent. When a module installs an OpenFlow rule to redirect traffic to a packet processing function, OFX augments the rule with instructions to tag the packets with the unique ID of the extension module that made the request. The Data Plane Agent parses the tag to determine which packet processing functions to apply. After processing, the OFX Data Plane Agent sends the packet back to the standard OpenFlow data plane, where another redirection rule determines what happens to it.

3. AN OFX SECURITY MODULE

We implemented an OFX security extension module that contains security functions similar to AvantGuard [9]. Unlike AvantGuard, which defined new switch and controller specification, the OFX security module *does not* require implementation changes to any part of the OpenFlow stack and can be deployed on existing OpenFlow switches.

Push-Based Alerts allow a switch to signal its controller whenever the packet- or byte-rate of a flow exceedes a threshold. In contrast, OpenFlow requires a controller to poll all the switches on its network for these statistics. After loading the module, the control application can send a switch an **install alert message**, which specifies the traffic flow that should be monitored, and the threshold that the controller wants to be notified about. Upon recieving this message, the module component loaded in the OFX switch agent installs a rule onto the switch's OpenFlow data plane that counts, but does not modify, traffic matching the flow specified by the controller, and then spawns a thread to periodically query the data plane for statistics about the installed rule. When the thread detects that the threshold has been exceeded, it sends a message to the controller.

TCP Handshake Validation protects an OpenFlow controller that installs routes for each TCP flow from Syn flood attacks, by ensuring that a TCP connection has completed its handshake before informing the controller about it. When the controller enables TCP handshake validation, the security module component on the switch installs low priority rules into its OpenFlow data plane that redirects TCP traffic that would otherwise be sent to the controller (*i.e.* packets that only match the default send to controller rule) to the OFX Data Plane Agent instead. The module loads a function into the data plane agent that checks packets against TCP connection records. If the packet belongs to an unestablished connection, the function redirects the packet to its destination MAC address, so the connection can complete. If the packet belongs to an established connection, the function redirects the packet to the controller, which can then install a higher priority route for the new TCP flow that will bypass the OFX module's low priority rule.

4. EVALUATION

We evaluated a prototype implementation of OFX and the OFX security modules. We implemented the OFX library as a python module that can be imported by the Ryu OpenFlow controller; the OFX Switch Agent as a stand-alone Python process; and the OFX Data Plane Agent as a stand alone C process. The Data Plane Agent connects to its switch's data plane using a memory mapped socket.

We tested OFX using the following machines: a Pica8 3290 OpenFlow switch with a harware data plane; a quad core Intel i7 controller with 4GB of ram; and two traffic generation machines with dual-core Intel Core-2-Duos and 2GB RAM. As a control ap-

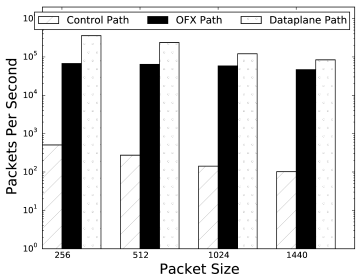


Figure 4: Packet size vs packets per second using different paths.

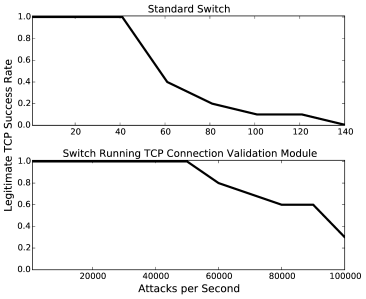


Figure 5: Attack packets per second vs TCP connection rate.

Statistic	Control Path	OFX Path	Data Path
Min Latency	3.604 ms	0.251 ms	0.169 ms
Avg Latency	4.039 ms	0.31 ms	0.232 ms
Max latency	8.08 ms	0.405 ms	0.292 ms
Max TCP Throughput	1.2 Mbps	584 Mbps	847 Mbps
UDP Drop % @ 5MBPS	72 %	0 %	0%
UDP Drop % @ 50MBPS	-	0.13 %	0%
UDP Drop % @ 500MBPS	-	3.6%	0%

Figure 6: Traffic statistics for flows travelling through different paths on our testbed.

plication, we ran the reference OpenVswitch test controller (version 2.3), in a hub mode that responded to every packet-in message from a switch with an instruction to flood the packet. The connections between all machines were with Gigabit ethernet.

OFX Throughput We first measured the throughput of OFX for packet processing with a null OFX module that loads each packet into memory, but does no processing. Figure 4 displays results which compares the maximum number of ping packets one traffic generation host can send to the other using the hping3 [1] tool. Different paths through the network were measured and Table 6 summarizes the statistics as measured using iperf. For all measurements, the OFX path performed several orders of magnitude better than the control path. The main bottleneck for both the control path and OFX path was the CPU usage of the switch, though OFX was much more efficient.

OFX Security Module The push-based alerts of our OFX Security Module reduced control traffic as expected and did not impact the data plane performance. More interestingly, was the the performance of the TCP Connection Validation.

We connected an attack host to our testbed that sent a flood of TCP syn packets to the controller, and then measured the likelihood of the traffic generation hosts establishing a TCP connection during the flood. [1] For this experiment, we replaced the reference controller with a TCP learning control application running on Ryu which installs a flow on the switch whenever it receives a packet from a previously unseen TCP stream.

Figure 5 shows the percentage of 20 TCP connection attempts that succeeded, for trials in which the flood rate varied. Without the handshake validator, the attack packets quickly overloaded the switch's ability to send packets to the controller, preventing the legitimate connections from being established. Running the OFX module allowed the network to be resistant to attacks several orders of magnitude larger. To our knowledge, this is the first defense against SYN flood attacks that runs directly on an SDN built with standard OpenFlow switches.

5. CONCLUSIONS AND FUTURE WORK

Software Defined Networking has much to offer security applications. However, current SDN security applications must choose between either *deployability* and *performance*. OFX is a framework for extending the functionality of network switches with modules that run on their general purpose hardware, and provides a better trade off: it allows SDN security applications to achieve *sufficient* performance for the their tasks without sacrificing *any* deployability. As future work, we plan to continue development, port OFX to other hardware switches and control platforms, investigate higher performance data plane integration techniques, and use OFX to develop more novel security applications.

[1]The TCP timeout window was set to 6 seconds.

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant Nos. 1406177, 1406225, and 1406192. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

6. REFERENCES

[1] Hping 3. http://www.hping.org/hping3.html, 2014.

[2] Open network linux. http://opennetlinux.org, 2014.

[3] White box switchs. http://www.whiteboxswitch.com, 2015.

[4] R. Braga, E. Mota, and A. Passito. Lightweight ddos flooding attack detection using nox/openflow. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 408–415. IEEE, 2010.

[5] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. DevoFlow: Scaling Flow Management for High-performance Networks. In *Proc. SIGCOMM*, 2011.

[6] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Openflow random host mutation: transparent moving target defense using software defined networking. In *Proc. Workshop on Hot topics in software defined networks (HotSDN)*, 2012.

[7] S. A. Mehdi, J. Khalid, and S. A. Khayam. Revisiting traffic anomaly detection using software defined networking. In *Recent Advances in Intrusion Detection*, pages 161–180. Springer, 2011.

[8] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson. Fresco: Modular composable security services for software-defined networks. In *Proc. Network and Distributed System Security Symposium (NDSS)*, February 2013.

[9] S. Shin, V. Yegneswaran, P. Porras, and G. Gu. AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-defined Networks. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2013.