OpenFlow Vulnerability Assessment

Kevin Benton
School of Informatics and Computing
Indiana University
Bloomington, Indiana, USA
KTBenton@Indiana.edu

L. Jean Camp
School of Informatics and Computing
Indiana University
Bloomington, Indiana, USA
LJCamp@Indiana.edu

Chris Small
School of Informatics and Computing
Indiana University
Bloomington, Indiana, USA
ChSmall@Indiana.edu

ABSTRACT

We provide a brief overview of the vulnerabilities present in the OpenFlow protocol as it is currently deployed by hardware and software vendors. We identify a widespread failure to adopt TLS for the OpenFlow control channel by both controller and switch vendors, leaving OpenFlow vulnerable to man-in-the-middle attacks. We also highlight the classes of vulnerabilities that emerge from the separation and centralization of the control plane in OpenFlow network designs. Finally, we offer suggestions for future work to address these vulnerabilities in a systematic fashion.

Categories and Subject Descriptors

C.2.6 [**Computer-Communication Networks**]: Networks

Keywords

Security, Software-Defined Networking, OpenFlow

1. INTRODUCTION AND RELATED WORK

OpenFlow offers powerful network customization by separating the data flow from the network control plane. The controller configures the data plane behavior of switches by installing rules using the OpenFlow protocol. Controllers can install rules in response to received traffic (i.e. "reactively") or they can install rules immediately to handle all traffic (i.e. "proactively"). The communications between the controller and switches are carried over a TCP connection that can optionally be protected by TLS with mutually authenticated certificates signed by a private key corresponding to a mutually trusted public key. The TCP connection is initiated by the switch with an operator-configured setting that specifies the IP address and TCP port of the controller.

Creating secure networks using OpenFlow is an active research area. *FRESCO* [7] provides a programming framework to execute and link together security related applications. *FortNOX* [5] extends the *NOX* controller to deal with conflicting OpenFlow applications by adding role-based constraints to the permitted rules that an OpenFlow application can send to switches. In a similar context, *FlowVisor* [6] acts as a mediator between controllers and switches to apply limitations to the rules created by controllers. We believe our work is the first to focus on vulnerabilities that emerge from OpenFlow network designs and vulnerabilities within the protocol.

HotSDN'13, August 16, 2013, Hong Kong, China.
ACM 978-1-4503-2178-5/13/08.

2. VULNERABILITY ANALYSIS

The original OpenFlow specification required the control channel between the controllers and switches to be protected using TLS. However, the subsequent specifications up to the current one (v1.3.0), make TLS optional. TLS has a higher technical barrier (than plaintext) for operators due to the steps required to configure it correctly, which include the following: generating a site-wide certificate, generating controller certificates, generating switch certificates, signing the certificates with the site-wide private key, and installing the correct keys and certificates into all of the devices. Comparatively, the plaintext operation only requires a single configuration parameter (the controller address). This creates an incentive for network administrators to forego TLS.

Due to the rapidly evolving nature of OpenFlow, many vendors of both switches and controllers have not fully implemented the specification. Specifically, most have skipped the TLS portion entirely. There is a noticeable lack of support from both software and hardware vendors, which consequently deters both sides from spending significant effort implementing it. The NEC IP8800 is the only hardware switch we identified with TLS support. The only controller with full TLS support was Open vSwitch. When referring to TLS support, one of the developers for the *Floodlight* controller wrote, "it would be pretty trivial to add it if there was sufficient interest", arguing that the lack of demand for the feature was due to limited support from the switches [1].

Man-in-the-middle Attacks The lack of TLS leaves an avenue for adversaries to infiltrate OpenFlow networks and remain largely undetected. While this may be infeasible for some physically secure networks, it becomes a greater concern in campus-style and remote-office deployments in which switches are deployed to locations which are easier for adversaries to access. Note that OpenFlow traffic may also be carried by an adversarial or less competent ISP.

The risks posed by a successful man-in-the-middle attack in an in-band (i.e., links carry both data and OpenFlow traffic) managed OpenFlow network are arguably worse than in current networks. In regular networks, an attacker has to wait until an operator logs into each switch management interface using an insecure protocol (e.g. Telnet, SNMPv2) to capture credentials. However, due to the constant connectivity and lack of authentication in the plaintext OpenFlow

TCP control channel, an attacker can immediately seize full control of any down-stream switches and execute very fine-grained eavesdropping attacks that would be difficult to detect. While this type of attack was technically achievable before, OpenFlow eliminates both the heterogeneity in network management interfaces and the time-consuming requirement of capturing management credentials.

Listener Mode This risk of adversarial rule modification is compounded by a "listener mode" supported by many switches, in which they accept unauthenticated connections to a configured TCP port from any network source [2]. These externally initiated connections can be used to write rules to switches and read information from them to allow for easy debugging. However, it eliminates the requirement for an attacker to conduct a man-in-the-middle attack to take over a network. By simply discovering a switch configured with a passive listening port, the attacker can dump the flows for reconnaissance and then insert rules to hijack downstream switches, capture traffic, and/or configure the switch to act as a proxy for further attacks.

Switch Authentication Even when TLS is implemented, failure to implement switch authentication in the controller (e.g. NOX) can allow an attacker to perform network reconnaissance by observing how the controller responds to different packets. Depending on the topology discovery logic in the controller, an attacker could potentially take down portions of the network or receive sensitive traffic by falsifying attached host information.

Flow Table Verification A full TLS implementation could increase security of the messages between switches and controllers; however, it wouldn't help detect switches that erroneously alter rules. Also, tracking the state changes of the flow-table for each switch by recording all of the *flow-removed* messages generated by switches requires extra logic on the controller, especially in the case of temporary network outage recovery. This mismatch between the controller's idea of the network's rule-state and the actual rule-state can lead to an access-control failure, a network outage, or other unexpected behavior. Currently, the only way to verify the rules is by dumping and inspecting the flow tables from each switch. This can be quite computationally costly for the switches and the controller(s).

Denial of Service Risks By centralizing the control plane, a new point of failure is introduced into the network. This can be mitigated by the use of multiple controllers (e.g. *Onix* [4]), but without careful rule design, controllers can be exposed to DoS attacks. In current network devices, some edge-case packets will have to be to be processed by the control plane. However, in OpenFlow, poor rule design can lead to saturating volumes of controller queries, affecting all switches that rely on that controller.

The majority of these DoS risks impact networks that use reactive rules. Networks based on proactive rule insertion do not have the same exposure as long as no traffic that can generate arbitrary *Packet-In* events. Yet these switches remain vulnerable to a DoS caused by excessive flow modifications from the controller. Special care must be taken by application developers to avoid conditions that cause excessive *Flow-Mod* messages. The OpenFlow 1.3 specification [3] suggests policing packets destined to the controller; however, it indicates that it is outside the scope of the specification. It provides no guidance for rate limiting messages to the controller, nor rule-insertions to the switches.

Today network devices have specialized firmware designed to deal with the known vulnerabilities of the protocols they support. For example, most modern enterprise Ethernet switches have code that offers ARP poisoning protection, DHCP snooping prevention, broadcast/multicast rate limiting, and MAC address limits for ports. In OpenFlow networks, all of these basic protections must be provided by controller. This places the burden of implementing complex security protections on the developers of the OpenFlow applications, who may be unaware of the existence of these attacks.

Controller Vulnerabilities By placing OpenFlow applications that perform deep packet inspection and conversation reconstruction on the host responsible for the control of the entire network, application isolation becomes an integral part of network security. Without it, compromising one OpenFlow application could lead to adversarial control of the entire network. Even attempting to parse complex protocols can lead to vulnerabilities, as demonstrated by the long list of Wireshark dissector security advisories[1].

3. CONCLUSIONS AND FUTURE WORK

OpenFlow risks repeating the errors of other insecure network management protocols (e.g. Telnet, SNMPv2, TFTP) where physical security was initially the only security and adoption of transport security lagged. The potential of OpenFlow includes controlling enterprise switches over the Internet to manage branch-office networks or to offer "network-management-as-a-service" to others. Without wide-spread adoption of strong protocol security, OpenFlow will be unable to expand into those roles.

Acknowledgments[2]

4. REFERENCES

[1] Floodlight-developers mailing list: Tls support. `https://groups.google.com/a/openflowhub.org/forum/?fromgroups=#!topic/floodlight-dev/hwPdcZOZhog`, Jan 2012.

[2] Hp switch software - openflow supplement. `http://h20000.www2.hp.com/bc/docs/support/SupportManual/c03170243/c03170243.pdf`, Feb 2012.

[3] Openflow switch specification: Version 1.3.0. `https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf`, Jun 2012.

[4] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, et al. Onix: A distributed control platform for large-scale production networks. *OSDI, Oct*, 2010.

[5] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu. A security enforcement kernel for openflow networks. In *Proceedings of the first workshop on Hot topics in software defined networks*, HotSDN '12, pages 121–126, New York, NY, USA, 2012. ACM.

[6] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[7] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson. Fresco: Modular composable security services for software-defined networks. *Internet Society NDSS (Feb. 2013). To appear*, 2013.

[1]http://www.wireshark.org/security/

[2]This material is based upon work supported, in part, by DARPA FA8750-13-2-0023. 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 DoD or Indiana University.