Towards Robust Trust Models for Software Defined Networks

Christopher C. Lamb
Department of Electrical and Computer Engineering
The University of New Mexico

May 12, 2014

Abstract

Software defined networks (SDNs) are becoming more popular in industry, though currently still only deployed by very technically-savvy organizations. Nevertheless, as the advantages of using SDN become more clear, future adoption promises to be high, with all network equipment vendors quickly moving to deploy products providing SDN capabilities. This impending wider adoption demands that security implications within SDN be more clearly understood. Today, mechanisms through which vendors can provide enhanced integrity and availability as well as agent-centric authentication and non-repudiation are poorly understood and have yet to be thoroughly investigated. In this paper, we present our current work outlining how we can define trust in SDN and what trust in SDN means for various operational components. We also address what the operational characteristics that impact trust propagation are, and present promising approaches to managing trust within SDN as an extension of these definitions and attributes.

1 Introduction

Clearly, Software Defined Networks (SDNs) are here to stay. The specific technologies are still in question, in that the community has yet to decide if OpenFlow will be the most common southbound protocol, or if it will be supplanted by some other alternative [Enn+11]. Nevertheless, intense industry involvement in SDN technologies and techniques makes it clear that organizations will be adopting SDN in the future, whether they would like to or not [Ope]. In order to effectively deploy SDN systems, we need to have a clear understanding of how we can secure them. To begin to secure SDN, we need to establish a more general picture of how trust propagates through SDN systems so we can more clearly envision how we can take advantage of hardened or redundant systems to enhance the security posture of deployed systems. The way we extend trust to system components in operational systems is key to defining and clarifying overall security postures in SDN architectures.

This paper represents our work in progress toward defining a rigorous trust model for SDN systems. As of today, we have framed the problem and defined a general SDN control architectural model over which we will begin to apply mathematical trust models. This paper will describe this reference model for the SDN control plane, highlight the key attributes of realistic SDN control planes a trust model must handle, and describe promising approaches to mathematically describing trust in SDN. We clost the paper with references to related work and our future plans in this area.

2 Trust in SDN

In order to frame the discussion of trust appropriately, we first propose a common model for the SDN control plane. This model allows us to define the common elements we need to examine, to describe the trust relationships, and describe precisely why these components are forced to trust other elements in the overall trust model. Here, we are going to limit our taxonomy to objects and classes based in OpenFlow inspired SDNs.

When modern SDN with OpenFlow was first developed, it consisted of a two layers — a controller layer

sending messages to a programmable switching layer. This provided separation between the control and data forwarding planes, enabling new capabilities for network innovation. In this early model, controllers maintained local databases of network state or local context to make packet forwarding decisions. This model was certainly useful and provided new capabilities previously unable to be deployed. Systems were able to take advantage of fine-grained system and user authentication and authorization, for example, at lower network levels than previously attempted. These initial configurations used the OpenFlow protocol to send information between controllers and switches, resulting in a two layer system [**ORIGINAL_OF_WORK**].

Over time, researchers began to incorporate other external elements, primarily to provide context information with respect to the current network. These data repositories could provide controllers with much more information about the current state of a given network. This information would then be used to make routing decisions. It could furthermore be shared between multiple geographically distributed controllers, providing more global insight into current operational state while empowering local controllers to appropriately optimize their local environments [**CONTROLLER_PLACEMENT**; **LOCAL_ROUTING_OPT**].

Shortly thereafter, due to the combined drives to both provide support for multiple protocols for switch management and to enable more established network vendors to claim a more competitive position in the burgeoning SDN market, software engineers began to propose architectures similar to what we see today in OpenDaylight. These systems began the current trend toward northbound and southbound interface separation, where the southbound interfaces used various protocols to communicate with network forwarding hardware while northbound interfaces enabled applications to submit control preferences, requests, or commands to the overall network management system [**BIG_NETWORK_CONTROLLER**; Ope].

2.1 A Common Structural Trust Model

2.2 Differentiating Attrubutes of Software Defined Networks

2.3 Promising Approaches to Trust Management

3 Related Work

4 Conclusions

References

[Enn+11] R. Enns et al. *Network Configuration Protocol (NETCONF)*. IETF 6241. Fremont, CA, USA: Internet Engineering Task Force, 2011.

[Ope] *OpenDaylight | A Linux Foundation Collaborative Project*. May 2014. URL: `http://www.opendaylight.org/`.