Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 18, 2013

S. Hartman
M. Wasserman
Painless Security
D. Zhang
Huawei Technologies co. ltd

October 15, 2012

Security Requirements in the Software Defined Networking Model
draft-hartman-sdnsec-requirements-00

Abstract

Software defined/driven networks provide new dimensions of flexibility in network design. This document analyzes security requirements as we design protocols to support multiple network applications on an SDN in an open manner.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as

described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Moving Beyond a Single Application 3
2.1. Class 1: Network Sensitive Applications 3
2.2. Class 2: Services for the Network 4
2.3. Class 3: Packaged Network Services 5
3. Authentication, Authorization and Multiple Organizations . . . 6
4. Security Requirements 8
4.1. Nested Application Security 9
5. Security Considerations 9
6. IANA Considerations 10
7. Informative References 10
Authors' Addresses 10

1. Introduction

This document analyzes the security of SDN architectures as we work to build SDN frameworks supporting multiple applications at the same time. The assumption of this protocol is that protocols like Openflow will be used between a SDN controller and switches. However this document assumes that there will be additional protocols between controllers and between controllers and applications. That is the focus for the current analysis.

2. Moving Beyond a Single Application

Openflow defines a protocol between a physical switch and a controller. Several factors motivate a layer between the controller and applications. For example [I-D.nadeau-sdn-problem-statement] discusses a model where managed service providers (MSPs) provide networking services to applications. This model involves the following attributes that significantly impact SDN security analysis:

- An application in one organization may use an MSP in another
- MSPs may be nested; one MSP may use the services of another
- Privacy concerns may limit what information should be exposed
- Applications require significant authorization and policy

The remainder of this section examines a few classes of applications in order to identify characteristics of SDN use cases that affect security.

2.1. Class 1: Network Sensitive Applications

Some applications require particular characteristics from the network. For example an application might need access to ports in a particular isolation domain/vlan. An application might require a path with particular characteristics. An application might require that traffic stay within a certain jurisdiction, travel only over certain equipment, or similar constraints. An application might want to monitor the cost of certain traffic or flows. And application might require that flows be discarded to mitigate a DOS attack or accepted in order to run a service.

Applications in this class will typically wish to provision aspects of the network using some API. They may wish to collect information from the network for monitoring, auditing or accounting.

Applications may not be aware of each others' requests. the SDN controller needs to make sure that one application does not negatively interact with another. Isolation of applications may be a security requirement. In this case the controller needs to make sure that even a malicious application cannot interact badly with another.

In most cases authorization will be important. The network may have resources that are not appropriate for one application or another and may wish to enforce authorization between them.

Multiple organizations may be involved within providing network services to such an application. For example, an application may request a network connection within a particular isolation domain. However that isolation domain may include resources within an enterprise, a cloud provider and a transit network between the enterprise and cloud provider. Authorization and authentication are likely to be handled by proxies in at least the simpler cases. For example, an application in the enterprise network requests access to the application domain from some controller in the enterprise. That controller has necessary credentials to request access from the transit network and cloud provider. Authentication and authorization may be more complex when an application in the cloud environment requests access to the isolation domain. Does it contact the enterprise controller or does it contact a resource in the cloud that has sufficient privilege to grant access to the enterprise aspect of the isolation domain. Debugging of multi-organization environments is facilitated by exposing information about all the environments. For example it is likely that for monitoring and debugging purposes enterprise applications would want visibility into the cloud environment and the parts of the transit network that the enterprise is allowed to see. Obviously the transit network and cloud provider would want to limit visibility into their internal structure but would want to make available information about resources controlled by that customer. for example the cloud provider would typically allow customers to see their own instances and virtual networks. Going through a proxy to authenticate requests for debugging and monitoring may not be ideal; it may be desirable for the application to collect debugging and monitoring information directly from the transit network and cloud provider. If so, the authentication and authorization needs to be flexible enough to permit this.

2.2. Class 2: Services for the Network

An application may provide a service such as a firewall, content inspection or intrusion detection to the entire network. In this case, the security model is similar to the security model between the SDN controller and switch; there is one key exception that will be discussed shortly. The primary role of the separation between the

SDN controller and application for this class of applications is to permit multiple applications to co-exist. So, isolation of resources is still important.

The security model for this class of application is similar to one of the common models for routing protocols and network management. Strong defense against outside attacks is required. It would be a significant attack if an attacker could impersonate an authorized application and gain the ability to reconfigure or monitor the network. However, inside attacks are not generally considered in scope for the threat analysis.

One key consideration with this type of security model is protecting the boundary between inside and outside and supporting multiple zones of trust. As an example, a border firewall application does not need the ability to reconfigure the interior of the network or to examine traffic inside interior isolation domains not destined for the border of the network. So, even in this model authorizing what resources an application is permitted to see and manipulate is important. This contrasts somewhat with the security model between the controller and switch, where the controller is permitted to manipulate all OpenFlow resources on the switch.

2.3. Class 3: Packaged Network Services

This class combines the previous two classes. Consider an application of class 1 that wishes for all traffic leaving a certain isolation domain to pass through a particular border firewall service. In effect an application is requesting an instantiation of another application be created as a virtual element in the network. This class of application permits abstraction and re-use of network applications. There is obvious value in the cloud space. However even within an organization, configuration re-use may have significant value.

To discuss the security we will say that an outer application nests a nested application within the network. The nested application may involve network resources (virtual or real) as well as compute and other resources.

This class involves classic security concerns such as authentication and authorization. What applications is the outer application permitted to nest? How is auditing and accounting handled? The IETF SDN architecture needs to permit these questions to be answered in a secure manner.

There may be multiple instances of a nested application, nested into either the same or different outer applications. There are

significant issues related to the sharing of information between instances of a nested application. For example, it is quite common for e-mail filtering services to collect information from multiple customers in order to better detect unwanted e-mail. However leaking proprietary information from one customer to another would be undesirable. To a large extent appropriate information sharing is a matter for application design and is out of scope for the SDN architecture. However the SDN architecture needs to support tracking of instance-specific information as well as global information and needs to facilitate design of applications that supports isolation of instances. This parallels virtual computing architectures, where the decision about how to split a problem is application specific, but the virtualization platform provides facilities to share information in a controlled manner and to manage large numbers of instances of applications.

Authentication may be more complex in the nested application situation. The permissions that a nested application has will depend on which outer application it is working on-behalf of; the authentication approach will need to account for this.

Multiple organizations may be involved with this class of application. As an example, the nested application may be provided by an MSP. Also, the nested application may use an MSP in providing the application.

Balancing which resources are visible to the outer application will be tricky. As with class 1 applications, debugging argues for visibility where possible. however, resources may be shared between instances of a nested application. Also, visibility into the nested application might provide proprietary information or provide an attacker with potential advantages. For example, understanding what flow filters were in place on a firewall application might expose information that would be valuable in getting around the firewall.

There's a similar concern with the nested application's visibility into the outer application. That visibility can be valuable for optimization and debugging. However, it may provide proprietary information or otherwise compromise the privacy goals of the outer application.

3. Authentication, Authorization and Multiple Organizations

In looking at needs of the various classes of applications, support for applications and network resources spanning multiple organizations appeared as a requirement multiple times. This requirement is interesting from a security standpoint. This section

explores how authentication and authorization can be handled between organizations.

One approach is for an organization to proxy connections to other organizations. The controller in one organization has credentials on behalf of the organization that it uses with other organizations. The local application talks to the local controller. When the controller realizes that it needs resources from another organization it talks to that organization's controller. This approach has been used fairly effectively in SIP [RFC3261] and other protocols with trusted intermediates. A significant advantage of this approach is that it permits the organization to enforce organization-level policy. It also permits the organization to hide information. For example, consider the class 1 example where the enterprise's isolation domain is split between a cloud, transit network and domain inside the enterprise. That split might be unimportant to the application; the organization's controller might try and present a unified network to applications. In such a case a proxy approach can work well.

The proxy approach has some disadvantages. There is no end-to-end security including integrity or data-origin authentication. The proxy becomes a very sensitive target. Because of the lack of end-to-end integrity, if the proxy is compromised, then the attacker can impersonate other network resources from the view of elements behind the proxy. The proxy may make deploying new features more difficult. To the extent that the proxy needs to understand a new feature before it is used, the proxy makes it more expensive to deploy the feature.

Another approach is for applications to directly have credentials in another organization. For example, this might work if an application wishes to use a particular external MSP. The advantage of this approach is that it provides flexibility to the application. The disadvantage is that it leaves open the question of how the two organizations' resources are glued together to form a consistent network. In some cases the application could manage this. In other cases, for example where changes are required in flow tables based on dynamic responses from the other organization, this may not be practical. Other disadvantages include increased complexity and difficulty in enforcing organization-level policy.

A third approach is to use some sort of federated/delegated authentication approach such as oauth [I-D.ietf-oauth-v2] or ABFAB [I-D.ietf-abfab-arch] to permit applications to obtain credentials that can be used in other organizations. This sort of delegation and federation could facilitate the following use cases:

- o Permit applications to obtain credentials for debugging and monitoring while attaching constraints to those credentials limiting resources the application can see

- o Preset credentials so applications can talk to foreign controllers; for example the cloud application talking to the enterprise controller to gain access to the enterprise isolation domain

- o Permit nested applications to be delegated access to some outer application resources

- o Grant outer applications access to nested application resources for debugging, monitoring or application-specific reasons

Another concern when multiple organizations are involved is auditing and accountability. Digital signatures and other mechanisms can be used to provide end-to-end accountability. However, this needs to be balanced against the need to hide information. It seems like the SDN use case may be one where end-to-end accountability is rarely an option.

4. Security Requirements

This section captures a variety of security requirements for layers on top of SDN controllers. These requirements are based on the discussion of potential applications as well as multi-organization considerations.

REQ1: Authentication is REQUIRED to the controller. Authentication SHOULD support existing credentials that are likely to be used in the datacenter.

REQ2: The interface to the SDN controller MUST support authorizing specific network resources to applications and manipulating the authorizations of applications.

REQ3: The SDN controller MUST provide facilities to isolate one application from another. XXX more discussion of this

REQ 4: The SDN controller interface MUST support a controller acting as a proxy on behalf of applications.

REQ 4a: The SDN interface SHOULD support a way of associating an audit ID or other tracking ID so that requests can be correlated with an original application when a proxy acts on behalf of an application.

REQ 5: The SDN controller interface MUST provide mechanisms for operators and applications to enforce privacy.

REQ 6: The SDN controller interface MUST support delegating access to a subset of resources; as part of delegation new authorization and privacy constraints MAY be supplied. This supports the security needs of the debugging use case, aspects of the nested application use case, and facilitates other inter-organization uses.

4.1. Nested Application Security

This section captures requirements for nested application support.

REQ N1: The SDN controller interface MUST support controlling authorization for what nested applications an outer application can nest.

REQ N2: The controller MUST separate authorizations held by one instance of a nested application from authorizations help by other instances of the same nested application. This is more about defense from bugs and operational mistakes than maintaining isolation of authorizations even if the nested application tries to circumvent the authorizations. However, it should be possible to write a nested application that maintains isolation sufficiently that compromise of one instance of the application will not lead to compromise of other instances. Obviously doing so places constraints on what resources need to be duplicated between instances of an application.

REQ N3: The SDN controller interface SHOULD provide outer applications a way to learn a nested application's policy for sharing information between instances.

REQ N4: Nested applications MUST be able to authenticate on behalf of a specific outer application. This facilitates authorization, accounting and auditing.

REQ N5: Nested applications MUST be able to specify privacy policy for what resources are visible to the outer application.

REQ N6: Outer applications MUST be able to specify privacy policy and authorizations with regard to what outer resources the nested application can interact with.

5. Security Considerations

This document provides discussion of the security implications of SDN architectures supporting multiple applications.

6. IANA Considerations

IANA is requested to make the internet secure. Note to someone: re-word this section before IANA reviews it.

7. Informative References

[I-D.ietf-abfab-arch]
Howlett, J., Hartman, S., Tschofenig, H., Lear, E., and J. Schaad, "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture", draft-ietf-abfab-arch-03 (work in progress), July 2012.

[I-D.ietf-oauth-v2]
Hardt, D., "The OAuth 2.0 Authorization Framework", draft-ietf-oauth-v2-31 (work in progress), August 2012.

[I-D.nadeau-sdn-problem-statement]
Nadeau, T. and P. Pan, "Software Driven Networks Problem Statement", draft-nadeau-sdn-problem-statement-01 (work in progress), October 2011.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

Authors' Addresses

Sam Hartman
Painless Security

Email: hartmans-ietf@mit.edu

Margaret
Painless Security

Email: mrw@painless-security.com

Dacheng Zhang
Huawei Technologies co. ltd
Huawei Building No.3 Xinxi Rd., Shang-Di Information Industrial Base Hai-Dian District, Beijing
China

Email: zhangdacheng@huawei.com