Access Control for SDN Controllers

Felix Klaedtke Ghassan O. Karame Roberto Bifulco Heng Cui
NEC Laboratories Europe, Heidelberg, Germany
firstname.lastname@neclab.eu

ABSTRACT

Based on the OpenFlow model, we propose an access control scheme for SDN controllers. Our scheme accounts for the different network resources, multiple security requirements, conflicts originating from the reconfiguration of network components, and the delegation of access permissions.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*network management*; D.4.6 [**Operating Systems**]: Security and Protection—*access control*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

software-defined networking; access control; reference monitor

1. INTRODUCTION

In SDN, network applications, e.g., for managing and analyzing the network, run atop of a logically centralized controller. However, existing controllers do not expose a common interface for applications and applications need therefore to be redeveloped for different controllers. To overcome this limitation, organizations—such as the ONF—started defining and standardizing a north-bound API, i.e., an interface provided by a controller for interacting with applications [4]. The reliance on a north-bound API is expected to shape the large-scale deployment of SDN, where multiple (third-party) applications interface with the same controller, providing better resource utilization and management.

A common north-bound API and the resulting deployment of various third-party applications poses new security threats in SDN. Indeed, "malicious" applications can leverage such an API and infiltrate the SDN network. These threats become even more evident when network resources are shared among user groups, divisions, or even other companies. In such a setting, the network owner "leases" network slices, which are administrated by the leasing tenants, which in turn install their own, possibly third-party, applications atop of the network owner's controller. These different tenants usually share network resources and at the same time might have competing objectives.

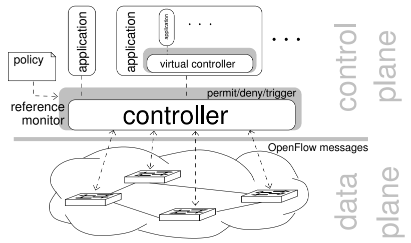

Figure 1: Security-enhanced SDN architecture

Restricting the access of the network users (i.e., the network owner and the leasing tenants) and the applications to the network components would reduce the network's attack surface and protect the network users' resources. The main issues for an access control scheme for SDN controllers are as follows.

(1) *Separation.* A network comprises various components. The scheme needs to logically separate them to be able to differentiate who can access them and how.
(2) *Enforcement.* The network users have their own security requirements with respect to accessing the network components. The scheme needs to support mechanisms to express and enforce these requirements.
(3) *Conflicts.* The network components are shared among network users. The scheme needs to resolve conflicts between the users, e.g., about the components' configuration.
(4) *Delegation.* Network components are owned, leased, or sub-leased. The scheme needs to support the delegation of permissions for accessing the components.

We propose an access control scheme for SDN controllers based on OpenFlow that—to the best of our knowledge—is the first that addresses (1) to (4). In particular, our proposal is more comprehensive than previously proposed schemes, e.g., the ones in [1–3, 5, 6]. Our scheme controls access to the network resources at the control plane but as close as possible to the data plane (see Fig. 1). By doing so, our access control scheme has a complete view over the components at the data plane and their configurations, which simplifies policy enforcement with conflict resolution. Overall, our work adds fundamental security concepts to controllers, which are essential for the development of mature SDN network operating system for multi-application/multi-tenant settings.

2. ACCESS CONTROL SCHEME

Subjects and Objects. As in most access control schemes, the entities of our access control scheme are classified as

HotSDN'14, August 22, 2014, Chicago, Illinois, USA.
ACM 978-1-4503-2989-7/14/08.
http://dx.doi.org/10.1145/2620728.2620773.

subjects (the network users) and *objects* (the network components). A network application runs with the permissions of the subject who executes it. Examples of objects are flow rules and the ports of a switch. Note that instead of treating a switch as a single object, we consider its components as separate objects. This allows one to differentiate between the different ways to interact with switches. Furthermore, instead of flow tables, we use so-called flow spaces, which resemble directory trees in an operating system, where the flow rules correspond to files and the flow spaces correspond to subdirectories. However, flow spaces can overlap and a flow rule can be contained in multiple flow spaces. Flow spaces allow one to equip a subject with different permissions to different kinds of network traffic. They only exist on the control plane and are managed by the controller.

Permissions. Similar to read and write permissions in an operating system for accessing the content of files, we have permissions for reading statistics (stat), requesting information about an object (config_read) and modifying an object's state (config_mod), and subscription permissions (subscr). For instance: (a) a subject who has the config_read permission for a flow space can obtain a list of all the entries of that flow space and the config_read permission is necessary to read a flow rule's match set and its priority, and (b) the subscr permission for a port allows a subject to subscribe an application, which is then triggered whenever the status of the port changes. The involved OpenFlow messages for the config_mod permission are, e.g, OFPT_PORT_MOD for ports and OFPT_FLOW_MOD with parameter OFPFC_MODIFY for flow rules. Instead of having a separate permission for each message type, we only have these four permissions for accessing the objects. This makes the scheme independent from the actual version of the OpenFlow standard. Another advantage is that policies remain manageable since a policy only assigns a few permissions to subjects.

We remark that the permissions not only restrict the OpenFlow messages that an application can send to the data plane; they also concern, e.g., the information the controller is allowed to provide to subjects. Also note that reply messages sent by a switch are received by the application that has sent the corresponding request message. For requesting statistics, the two message types OFPT_STATS_REQUEST and OFPT_STATS_REPLY is an example of such a pair of request and reply messages. Error messages (i.e., messages of type OFPT_ERROR) are treated as reply messages. An application does not need any permission to receive reply messages.

In addition to the OpenFlow attributes (priority, timeout values, etc.) for flow rules, our scheme handles attributes that relate flow rules to other objects and restrict their modifications. Two such attributes are no_overwrite and cascade. The attribute no_overwrite prevents the installation of overlapping flow rules with a higher priority. The cascade attribute relates flow rules installed at different switches among each other. For example, the deletion of one flow rule in a cascaded flow forces the deletion of all its flow rules. The switches do not need to be aware of these additional attributes. The reference monitor, which is the controller component that enforces the given policy (see Fig. 1), checks, before the controller reconfigures a switch's flow table, whether the conditions specified by the attributes are fulfilled.

Policies. Our access control scheme comprises a mandatory access control component and a discretionary one. The first component is managed by the network administrator; the second component allows subjects to delegate permissions to other subjects, similar to the Unix file system. For illustration, consider the following example. Assume a subject has the config_mod permission in a flow space. With this permission, it can create subspaces and install flow rules within that flow space. The new objects are owned by the subject. By the discretionary access control component, the object owner can, e.g., assign config_read permissions to other subjects.

Multiple applications can be subscribed to an event and their replies might conflict with each other. For example, for a packet-in event (i.e., an OpenFlow message of type OFPT_PACKET_IN), one application wants to install flow rules for the packet and another wants to drop it. For conflict resolution, our scheme uses a dynamic approach (as opposed to a static one, which would not permit the subscription of contradicting applications in the first place). We prioritize the applications' replies and only carry out the ones that are not in conflict with a reply of a higher priority. The priorities for the applications are determined by exploiting the hierarchical structure of the flow spaces by either prioritizing the most global flow spaces or the most local ones in which applications are triggered. The application with the highest priority can be efficiently determined by traversing the flow space hierarchy. Furthermore, by specifying an application's replies when subscribing it, we can avoid to trigger applications that contradict the one with the highest priority, thereby saving computational resources at the control plane.

3. CONCLUSION

The proposed access control scheme is low-level in the sense that it is close to the south-bound API of the controller, which interfaces directly with network components using OpenFlow. The reasons for this design choice are: (a) the OpenFlow standard is already well established and (b) other network abstractions are not yet that well understood or defined. In fact, we expect that future north-bound APIs in SDN will support multiple different abstractions of the network at the control plane. Furthermore, any such interface will be built on top of the interface provided by OpenFlow that directly interacts with the network components. We claim that since, e.g., conflicts are easier to detect and resolve close to the south-bound interface, access control at higher layers will utilize our access control scheme and complement it.

4. REFERENCES

[1] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi. Participatory networking: An API for application control of SDNs. In *SIGCOMM 2013*.

[2] M. Monaco, O. Micher, and E. Keller. Applying operating system principles to SDN controller design. In *HotNets 2013*.

[3] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Du. A security enforcement kernel for OpenFlow networks. In *HotSDN 2012*.

[4] S. Raza and D. Lenrow. North bound interface working group charter. Open Networking Foundation, 2013.

[5] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the production network be the testbed? In *OSDI 2010*.

[6] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang. Towards a secure controller platform for OpenFlow applications. In *HotSDN 2013*.