Enhancing SDN Security by Device Fingerprinting

Nicholas Gray, Thomas Zinner, Phuoc Tran-Gia
University of Würzburg, Germany
Email: {nicholas.gray|zinner|trangia}@informatik.uni-wuerzburg.de

***Abstract*—Software-defined Networking (SDN) provides an increased flexibility and cost savings by separating the data from the control plane. Despite these benefits, this separation also results in a greater attack surface as new devices and protocols are deployed. OpenFlow is one of these protocols and enables the communication between the switch and the controller. Ideally this connection takes place over an encrypted TLS channel, but as this feature is marked optional, it is not supported by all devices. This allows an attacker to eavesdrop and alter the communication, hence resulting in a comprised network. In this work, we demonstrate a new approach for authentication based on device fingerprinting to enhance the security in scenarios, where cryptographic mechanisms are unavailable.**

I. Introduction

The ability to dynamically react to changing network requirements, not only improves the flexibility but also results in cost savings. One concept providing a high adaptability is Software-defined Networking (SDN) [1]. Here, the control plane is decoupled from the data plane and outsourced to a programmable controller. The communication between the controller and the data plane, which remains in the SDN-enabled switch is governed by protocols such as OpenFlow [2].

Despite the advantages of this separation, the introduction of newly created protocols and networking devices also adds to the overall attack surface of the network. As the controller configures a switch upon its first connection and may reconfigure the switch if network requirements change, an attacker impersonating a switch is able to learn vital information concerning the network topology. Hence, a proper authentication of legit switches has to be guaranteed.

In OpenFlow the communication between the controller and the switch is ideally established over a TLS [3] encrypted connection, which enables an end-to-end verification of the participating devices. However, during the migration from OpenFlow 1.0 to 1.1 the requirement of an encrypted communication channel has been marked as optional. Hence, several products lack this functionality as stated in [4].

In this demonstration, we show a novel approach to ensure the authentication of devices using fingerprinting, which applies in scenarios where cryptographic mechanisms are unavailable. For this, we implement a module for the ONOS [5] controller, which extracts the required information for the generation of the device fingerprint as well as prevents unauthenticated access.

II. Fingerprinting Module

The fingerprinting module is written as plugin for the ONOS OpenFlow controller. Its main responsibilities are the creation and management of device fingerprints as well as authenticating connecting switches. In the following we describe these functionalities in more detail.

A. Fingerprint Generation

The generation of the device fingerprint is based on the observation that the OpenFlow switch specification differentiates between required and optional features. Thus, the vendor is responsible to define which of these features are supported by their products. Therefore, devices may differ in their provided capabilities and as no concrete implementation requirements are given, even common functionalities may differ concerning their processing times. Based on these discrepancies it is possible to identify a device.

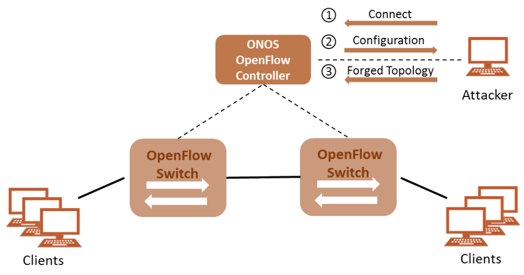
During the process of creating the fingerprint, the module takes several attributes of a device into account, which can be categorized by static or dynamic features. Whereas we define static features as all items which can be directly polled such as the number of ports, we define dynamic features as all indicators, which are computed during the direct interaction with the switch such as timings of specific operations. The final fingerprint is then composed as vector containing the results of all tested features.

The computation of the fingerprint is triggered upon reception of the first message of a newly connecting switch.As all static features could be obtained and replayed by an attacker which is able to eavesdrop on a legit switch connection, the module additionally starts benchmarking the device for specific operations and records the timings. As these timings are conducted during direct interaction with the device, they are harder to reproduce and add an extra layer of security.

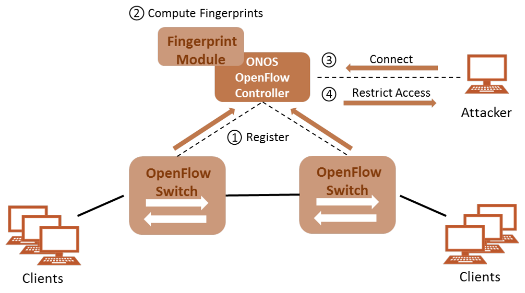
One candidate for such a dynamic feature, is provided by the investigations of [6]. Here, the authors evaluate the time span required for the switch to process a varying number of flow_mod messages. Their results show that different switches can be easily clustered by their processing times. The same procedure has been implemented by the fingerprinting module which repeats the measurements several times to accommodate for timing differentiations. In addition to the timings of the individual runs, the average, minimum and maximum as well as the standard derivation is computed for future reference during the authentication procedure.

B. Registering a new switch

Whenever a new switch is deployed, the switch has to be registered at the fingerprinting module to create a database of trusted devices. For this, the switch needs to be configured to connect to the OpenFlow controller running the fingerprinting



(a) Scenario I: Depicting the successful attack.



(b) Scenario II: Illustrating the mitigation of the attack.

module. The module then automatically determines the connection attempt of an unauthorized device and lists every device in a graphical view to the network operator, where the new switch can be approved. This triggers the generation of the fingerprint and adds it to the list of trusted devices.

C. Authentication

Upon connection of a device the fingerprinting module is notified by the ONOS framework and directly initiates a block. This prevents other ONOS plugins to use the device until the authentication process is completed. Furthermore, the computation of the fingerprint is started. Once the computation is completed the result is checked against the list of registered fingerprints. If a match is returned, the initial block is removed and the device can be used as intended. However, if the authentication fails, the device is remains isolated and is reported to the network operator. In addition, all incoming messages from the device are dropped to prevent further resource consumption and attack surface.

III. Demonstration

In our demonstration we show how an attacker can impersonate a switch to obtain and alter information regarding the network topology and how this attack is mitigated without the need of TLS by using the fingerprinting module.

The demonstration setup consists of an OpenFlow controller running ONOS and the fingerprinting application as well as two OpenFlow-enabled switches for managing the data plane. Furthermore, several virtual machines are used to propagate the network and one notebook resembles the attacker.

The benefits of the fingerprinting module are demonstrated in two scenarios which are illustrated in Figure 1a and 1b. In the first scenario, the OpenFlow controller is not running the fingerprinting module and thus the attacker is able to connect to the controller by impersonating a legit switch. As no authentication is required the attacker is provided with vital information as part of the initial configuration, which is sent by the controller. Furthermore, the attacker actively alters the network topology by sending forged packets to the controller, resulting in a compromised network.

In the second scenario, we demonstrate how the prior attack is successfully mitigated by activating the fingerprinting module. As illustrated in Figure 1b, the two legit switches are registered at the module and their fingerprint is generated. Once the registration is complete, both switches are able to connect to the controller and the network functions as intended. The previous attack is then repeated. In contrast to the first scenario the attack no longer succeeds as the fingerprint which is generated for the attacker differs from the entries of trusted devices. Hence, the connection to the controller fails and the security of the network is improved.

IV. Conclusion & Outlook

In this work, we demonstrate the computation of fingerprints of OpenFlow switches and use this information to enhance the security in scenarios where authentication based on cryptographic mechanisms is unavailable. For this, we determined a set of features based on static capabilities as well as dynamic attributes of OpenFlow-enabled switches to identify different products. This procedure is then incorporated in a novel authentication mechanism implemented as proof of concept for the ONOS controller.

Future extensions of our work aim for incorporating more features for the generation of the fingerprint as well as determining a minimized set of features which uniquely identify a specific switch. Hereby, we strive to reduce the connection setup times and the number of possible false positives.

References

[1] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for sdn," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 210–217, 2014.

[2] Open Networking Foundation, "Openflow switch specification," https://www.opennetworking.org/, called on Jan. 10, 2017.

[3] The Internet Engineering Task Force, "The transport layer security protocol version 1.2," https://tools.ietf.org/html/rfc5246, called on Jan. 10, 2017.

[4] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 151–152.

[5] ON.LAB, "Onos - the open network operating system," http://onosproject.org/, called on Jan. 10, 2017.

[6] A. Nguyen-Ngoc, S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, and M. Jarschel, "Performance evaluation mechanisms for flowmod message processing in openflow switches," in *IEEE Sixth International Conference on Communications and Electronics*, 7 2016.

This work has been performed in the framework of the SARDINE project and is funded by the BMBF (Project ID 16KIS0261). The authors alone are responsible for the content of the paper. The authors want to thank Michael Ott for his programming work.