Demonstrating State-based Security Protection Mechanisms in Software Defined Networks

Demonstrating State-based Security Protection Mechanisms in Software Defined Networks

Thianantha Arumugam and Sandra Scott-Hayward
Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast, BT3 9DT, N. Ireland
Email: t.arumugam@qub.ac.uk, s.scott-hayward@qub.ac.uk

I. Introduction

Software Defined Networking (SDN) has been well-adopted on a local scale; within organizations such as Google, Facebook, and Microsoft, and by telecommunications operators such as AT&T, Deutsche Telekom, and SK Telecom. The benefits of SDN to implement changes quickly and thereby accelerate service provisioning and reduce operational costs have been well documented [1]. From a security perspective, the global network view and programmability of the data plane fundamental to SDN support enhanced network security services. This is achieved by the ability to export network traffic statistics from the data plane (*collection*), analyse these via SDN controller security applications (*analysis*), and re-programme the network with the appropriate remediation mechanism in response to the network threat (*control*). The security opportunity with SDN has been of interest to both the academic and industry security research communities for a number of years. For example, Radware's DefenseFlow [2], released in 2013, was one of the first SDN-based Distributed Denial-of-Service (DDoS) attack defence systems. With the drive to deploy SDN on a global scale, security is even more important today.

However, despite the potential with SDN/NFV (Network Functions Virtualization) for automated and adaptive network security services e.g. [3], [4], a clear limitation on the performance of these network security solutions is the control communication introduced in the process. For efficient, real-time attack detection and protection, the latency to exchange data between the network elements and a remote SDN controller is impractical. This also assumes that the control channel is available for the required communication. The vulnerability of the control channel to communication overload and the ability for an attacker to exploit this has been highlighted in [5]. Finally, linked to the performance challenge is the issue of scalability. There is a limit on the number of switches that a controller can support, which is linked to the controller flow processing capacity and the specific network configuration and operation. The introduction of security applications and their associated processing requirements adds a further load.

Given the performance and scalability challenges linked to the controller interaction, a reasonable solution would be to provision these network security services in the data plane i.e. directly on the switches. This immediately introduces a different challenge. The current concept of SDN and the OpenFlow switch specification [6], is of stateless switches. As shown in Figure 1, the OpenFlow protocol matches packets on L2-L4 fields, while for network security functions, information on the flow state is required. This could be, for example, to detect a TCP SYN Flood attack, for which information on the state of the network connection is required, or to provide network security functions such as port-knocking to add a layer of authentication. It should be noted that in OF Switch specification v1.5, L2-L4 support was extended to include L4 TCP flag matching, which would enable TCP connection state identification. Unfortunately, OpenFlow support in hardware switches does not directly follow the specification update.

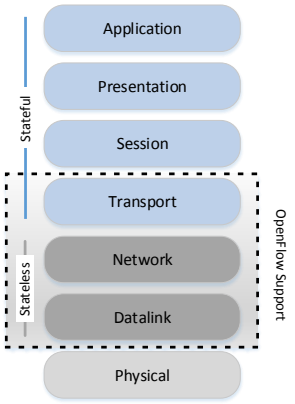

Fig. 1. OpenFlow relationship with the OSI model

A number of switch-level stateful data-plane approaches have been proposed [7]–[9]. The connection tracking module from Linux has been added to OpenvSwitch to support stateful tracking of flows [7]. OpenFlow extensions have been introduced to support this. SDPA [8] proposes a Match-State-Action paradigm by introducing three additional tables; State table, State Transition table and Action table. Although SDPA provides a switch-level design to detect and prevent, for example, Denial-of-Service (DoS) attacks, continuous interaction with the controller is required to initiate the state tables and to create and update state entries. OpenState [9] uses two distinct tables; the State table and the eXtended Finite State Machine (XFSM) table, along with packet handling mechanisms to support stateful packet forwarding inside OpenFlow-enabled switches. The port knocking application and a DDoS application have been demonstrated leveraging the OpenState features [10].

These state table mechanisms enable some traditional network attack protections. However, SDN-specific attacks include the threat of a compromised application or malicious network elements. As a result, the SDN is exposed to threats such as exfiltration, bypassing specific network components, eavesdropping and man-in the-middle attacks [11]. These threats arise from the implementation of SDN control functions such as link reconfiguration and switch identification, the protection from which have not been considered in [7]–[9].

The OpenState framework is leveraged in this work to introduce state-based SDN security protection mechanisms. Several extensions are required for this design and these are outlined in Section II based on description of a SDN configuration-based attack. The demonstration of the attack protection mechanism is described in Section III.

II. State-based Security Protection Mechanisms

A. SDN Configuration-based Attack

As described in Section I, SDN configuration-based attacks target the topology and path update network control functions. The normal host mobility process is illustrated in Figure 2 with H1 relocating from port 1 to port 6. When H1 is disconnected from port 1 and connected to port 6, the switch identifies that the host has moved (1) and a relocation message is sent to the controller (2). In response, the controller updates the flow rules corresponding to H1 in switch SW1 to reflect the new network connection (3), and H1 traffic is processed via port 6 (4). A malicious host can use this host relocation process to send a fake relocation message to the controller triggering network reconfiguration. A successful attack can isolate a legitimate host from the network.

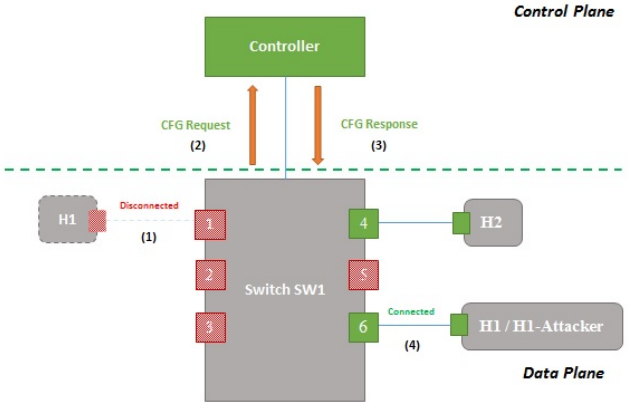

Fig. 2. Configuration update/attack - No Protection Mechanism

B. SDN Configuration (CFG) Security Protection Mechanism

The configuration attack is initiated from the data plane but the configuration update originates from the control plane. We therefore propose to isolate the SDN controller (control path) and introduce intelligence to the data path to self-monitor relocation messages. There are two functions introduced to support this process; the *Host Connection State Table*, and the *Configuration Request Module (CRM)*.

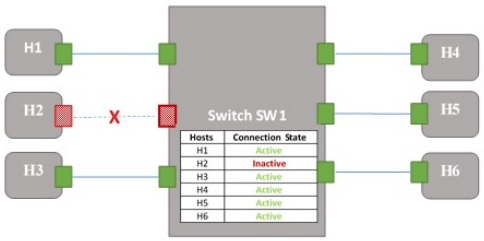

Fig. 3. Host Connection States

1) Host Connection State Table: The connection state table to monitor host-switch connections is shown in Figure 3. This captures the active/inactive connection state. In order to reach the active state, an authentication process is required to gain full access to the routing logic specified by the switch flow tables. Based on the OpenFlow match-action paradigm, an L2-4 packet-based authentication process is proposed. The state machine describing the connection and authentication logic is illustrated in Figure 4.

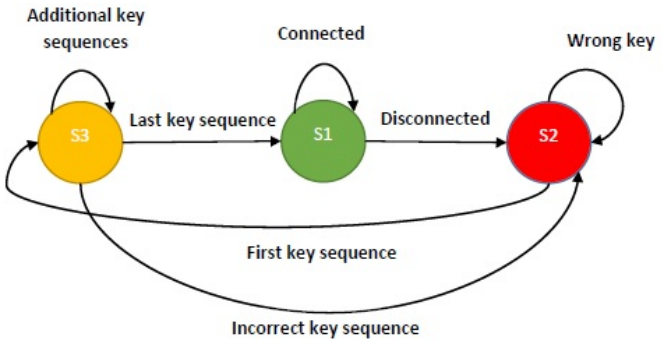

Fig. 4. Connection Authentication State Machine

In state S1, host H1 is connected to switch SW1, as shown in Figure 3. In this state, H1 is *active* and normal network operation can proceed. Host H2 is disconnected from switch SW1, as shown in Figure 3, so that the connection state of H2 is *inactive*, and H2 is in state S2 in Figure 4. From state S2, in order to reconnect to the network via switch SW1, H2 must present a sequence of authentication messages e.g. key sequence '1' to 'n'. When in state S2, if the first key sequence is correct, H2 moves from state S2 to S3. From state S3, further correct sequences will be processed until the last correct sequence is reached and H2 is restored to state S1 with the link activated. However, at any point in the sequence, if the wrong sequence is detected by SW1, the connection state for H2 will revert to state S2.

Note: An authentication key sequence can be shared with devices at the point of registration to the network. In the complete CFG security solution, secrecy is not required. However, this mechanism could also be used for PHY security in a controlled environment to protect against a compromised device attempting to connect to the switch. In such a scenario, the key sequence is secret and programmed by a network operator to a controlled set of devices.

2) Configuration Request Module: The layered CFG security protection mechanism is illustrated in Figures 5 and 6. In the proposed architecture for CFG security, a relocation message from the switch (1) will be processed by a configuration request module (CRM) that determines whether the message is genuine and valid, or not (2). The CRM will make this decision based on the host connection state (3). Following the FSM in Figure 4, if a relocation message is received and the host is in state S1, the message will be treated as malicious and dropped by the CRM, as shown in Figure 5. The fake H1 connection to port 6 will be prevented. However, if a relocation message is received while the host is in either state S2 or S3, the message will be considered to be valid and the CRM will pass the request to the controller, as shown in Figure 6.

A request for relocation under an invalid connection state can also be treated as a first alert to identify a compromised switch. The CRM can send an alert to the network administrator for further action.

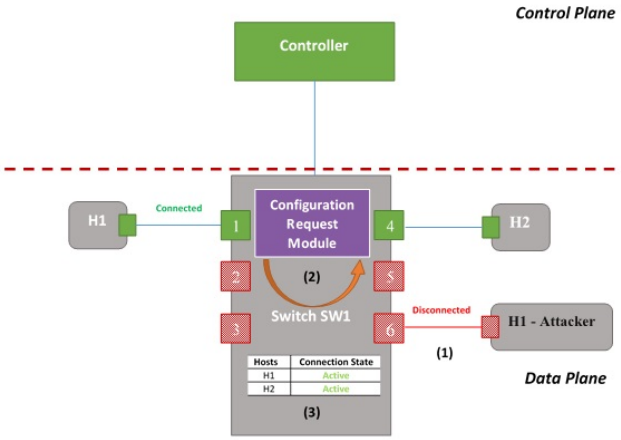

Fig. 5. CFG Security Protection Mechanism - Attack Scenario

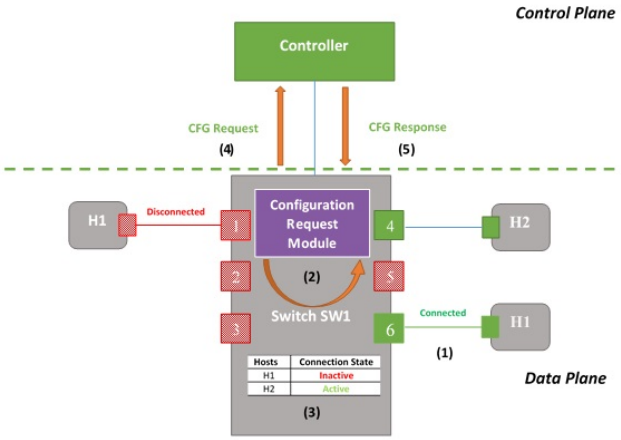

Fig. 6. CFG Security Protection Mechanism - Update Scenario

III. Demonstration

We use a Mininet 2.2.1 VM on Ubuntu 14.04.1 (64-bit) to demonstrate the security mechanisms presented in Section II. A single switch, multiple host topology is created using the CPqD/ofsoftswitch13, which has been adapted to implement OpenState [10]. The SDN Controller is Ryu version 3.29, which has also been modified to support OpenState.

a) No CFG Security: The configuration-based attack is first demonstrated with no protection mechanism in place. The network is initialized with the endpoints connected (H1 connected to port 1). H1-Attacker then fakes a connection to port 6. With no protection in place, this relocation request will be processed as normal and traffic destined for H1 will be directed to port 6. The legitimate H1 connection is effectively disconnected from the network, as shown in Figure 2.

b) CFG Security - Legitimate relocation: Once the network is initialized with the endpoints connected and in *active* state, host H1 disconnects from port 1 and connects to port 6 (see Figure 6). This represents a legitimate connection. On disconnection, the H1 entry in the connection state table will transition to *inactive*. On H1 connection to port 6, a configuration request message is sent to the CRM. The CRM then checks the connection states of the end points of the request. As the connection state is *inactive*, the CRM will pass the relocation message to the controller.

c) CFG Security - Malicious connection: The network is re-initialized. With H1's connection *active* on port 1, an attacker spoofing H1 now attempts to connect to port 6. This represents a malicious connection. The configuration request message for H1-Attacker is sent to the CRM. The CRM checks the connection states of the end points of the request and identifies that there is a current *active* H1 connection. The CRM will drop this configuration request from the switch and the H1-attacker will not be allowed to connect to the network.

References

[1] "AT&T: SDN is Slashing Provisioning Cycle Times by up to 95%." [Online]. Available: http://www.lightreading.com/carrier-sdn/sdn-architectures/atandt-sdn-is-slashing-provisioning-cycle-times-by-up-to-95-/d/d-id/717582

[2] "Radware DefenseFlow." [Online]. Available: http://www.radware.com/Products/DefenseFlow/

[3] T. Yu, S. K. Fayaz, M. Collins, V. Sekar, and S. Seshan, "Psi: Precise security instrumentation for enterprise networks," in *Proc. NDSS*, 2017.

[4] T. Koulouris, M. Casassa Mont, and S. Arnell, "SDN4S: Software Defined Networking for Security," 2017. [Online]. Available: https://www.labs.hpe.com/publications/HPE-2017-07

[5] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For*. IEEE, 2013, pp. 1–7.

[6] "OpenFlow Switch Specification Version 1.5.1," Open Networking Foundation. [Online]. Available: https://www.opennetworking.org/sdn-resources/technical-library

[7] J. Pettit and T. Graf, "Stateful connection tracking & Stateful NAT," OpenvSwitch, 2014. [Online]. Available: http://openvswitch.org/support/ovscon2014/17/1030-conntrack_nat.pdf

[8] S. Zhu, J. Bi, C. Sun, C. Wu, and H. Hu, "SDPA: Enhancing stateful forwarding for software-defined networking," in *Network Protocols (ICNP), 2015 IEEE 23rd International Conference on*. IEEE, 2015, pp. 323–333.

[9] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "OpenState: programming platform-independent stateful openflow applications inside the switch," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 44–51, 2014.

[10] "OpenState SDN Project." [Online]. Available: http://openstate-sdn.org/

[11] K. Thimmaraju, L. Schiff, and S. Schmid, "Outsmarting network security with SDN teleportation," in *Security and Privacy (EuroS&P), 2017 IEEE European Symposium on*. IEEE, 2017, pp. 563–578.