Abstract—Dynamic network security services have been proposed exploiting the benefits of Software Defined Networking (SDN) and Network Functions Virtualization (NFV) technologies. However, many of these services rely on controller interaction, which presents a performance and scalability challenge, and a threat vector. To overcome the performance issue, stateful data-plane designs have been proposed. Unfortunately, these solutions do not offer protection from attacks that exploit the SDN implementation of network functions such as topology and path update, or services such as the Address Resolution Protocol (ARP). In this work, we propose state-based SDN security protection mechanisms. Our stateful security data plane solution, OFMTL-SEC, is designed to provide protection against attacks on SDN and traditional network services. Specifically, we present a novel data plane protection against configuration-based attacks in SDN and against ARP spoofing. OFMTL-SEC is compared with the state-of-the-art solutions and offers increased security to SDNs with negligible performance impact.

I. INTRODUCTION

Software Defined Networking (SDN) turned 10 years old in 2018. Over these 10 years, significant research and development has been undertaken to enable adoption and integration of SDN across network operations, from enterprise to telecommunications providers. The benefits of SDN to implement changes quickly and thereby accelerate service provisioning and reduce operational costs have been well documented [1]. In recent years, a focus on security in SDN has developed. This stems from the enhanced network security services based on the SDN global network view and programmability of the data plane [2]. 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. For efficient, real-time attack detection and protection, the latency to exchange data between network elements and a remote SDN controller (particularly in a large-scale network) is impractical. This also assumes availability of the control channel. The vulnerability of the control channel to communication overload and the ability for an attacker to exploit this has been highlighted in [2].

In this work, we consider two specific network attacks; the first is an SDN-specific attack related to the SDN topology and path update network functions, and the second is the Address Resolution Protocol (ARP) spoofing attack that targets both traditional and software-defined networks.

The SDN configuration-based attack leverages the message exchange between the network element and the SDN controller when a topology update is processed. For example, the topology management service in the SDN controller handles host discovery. When a new host transmits data on the network, in reactive mode with no flow rule to match this packet, a packet_in message will be generated at the switch and sent to the controller. The controller then learns and updates the host information (i.e. MAC address, inport) based on the packet_in message and can generate flow_mod messages to the switch to direct traffic to/from this host. As there is no check on the legitimacy of the host identifier information, it is possible for a malicious attacker to spoof the host identifier information and “poison” the network topology, as described in [5]. This message exchange mechanism also enables control channel denial-of-service (DoS) attacks through flooding packet_in messages [2].

Due to the insecurity of ARP, ARP spoofing is often the first step in a more sophisticated network attack. ARP is used to map IP addresses to MAC addresses. It is a stateless protocol such that each request or reply is treated independently. As a result, it is possible to “poison” the ARP cache in hosts by creating a false IP-MAC address binding. This arises because there is no mechanism to authenticate the sender of an ARP request/reply message or to check the integrity or validity of provided information. The two main approaches to mitigate ARP spoofing/poisoning in SDN are: (1) The SDN Controller implements proxy ARP maintaining a single master ARP table (IP-MAC address mapping database) for the entire network. This enables a legitimacy check but the controller communication for each ARP message introduces delay. Furthermore, any ARP spoofing attack can target any victim on the SDN (rather than within a broadcast domain as would be the case for a traditional network), and (2) As an improvement to controller proxy ARP, SecureBinder is a controller application introducing stronger binding between network identifiers [6]. In an OpenFlow multi-table implementation, SecureBinder uses Table 0 to direct all identifier binding broadcast traffic e.g. ARP, DHCP, DNS to the controller where it is processed by the SecureBinder application. All other network applications use Tables 1+. While securing the MAC-IP binding, this solution still requires controller communication for all ARP traffic.

In contrast, with our OpenFlow Multi-Table Security solution (OFMTL-SEC), we enable protection against the SDN configuration and ARP spoofing attacks without reliance on SDN controller communication. OFMTL-SEC is a data plane security solution leveraging the OpenFlow multi-
mentation and the stateful data plane, OpenState [7]. This removes the latency in the protection mechanism offering both security and performance in the network. The main contributions of this work are:

- A novel data plane protection against configuration-based attacks in SDN;
- A novel data plane protection against ARP spoofing;
- Implementation of these security protection mechanisms using OpenState [7].

The paper is laid out as follows: Related work is discussed in Section II. The attack model is introduced in Section III. The design of the OFMTL-SEC solutions is presented in Section IV with the implementation described in Section V. In Section VI, the performance of the OFMTL-SEC solution is evaluated and discussed. Section VII concludes the paper.

II. RELATED WORK

A. Stateful data plane designs

OpenFlow (OF) [8] is the most popular protocol for communication between the data and control planes in SDN. In the original OF switch specification [9], matching is based on L2-L4 packet header fields offering stateless traffic management. In subsequent developments, (OF Switch specification v1.5), L2-L4 support was extended to include L4 TCP flag matching, which would enable TCP connection state identification. However, this capability is not supported by all OF devices.

Due to the identified limitations of the original OF specification, a number of stateful data-plane designs have been proposed [10]–[12]. The connection tracking module from Linux has been added to OpenvSwitch to support stateful tracking of flows [10] and OF extensions have been introduced to support this. SDPA [11] 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 [12] 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 OF-enabled switches. The port knocking application and a DDoS application have been demonstrated leveraging the OpenState features [7].

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 (MITM) [13]. These threats arise from the implementation of SDN control functions such as link reconfiguration and switch identification, for which protection is not considered in [10]–[12].

With the evolution towards stateful flow processing, a number of vulnerabilities of stateful SDN data planes have been identified in [14]; unbounded flow state memory allocation, triggerable CPU intensive operations, lack of authentication mechanisms in the data plane, and lack of a central data plane state management. Of particular interest for the OFMTL-SEC solution is unbounded flow state memory allocation, which is the issue of a large volume of flow state information to be stored on the switch that could lead to switch memory saturation attack. This is addressed in the OFMTL-SEC design and discussed in Section IV.

B. SDN attacks and protection mechanisms

The vulnerabilities of the SDN architecture and SDN-specific attacks have been described by various researchers [2], [13], [15]. In [15], the authors describe attacks on the network topology and forwarding functionality. Their solution, SPHINX, enables detection of attacks in SDNs based on monitoring and validation of OF control messages. Detection of ARP poisoning, topology manipulation, and control and data plane DoS attacks are demonstrated. The solution relies on SPHINX monitoring the control communication, which means that it is limited to a reactive SDN environment and is deployed at the control plane.

In [5], the authors present a solution called TopoGuard, with an extension to the OF controller to provide automatic and real-time detection of network topology poisoning attacks. The objective is to fix the OF controller security omissions that can lead to host location hijack and link fabrication attacks. The solution is a Topology Update Checker that automatically validates the update of network topology, and is dependent on the information provided by Port Manager and Host Tracker controller applications. This dependency is critical as it requires the SDN controller to evaluate relocation messages and determine their legitimacy. Furthermore, it does not verify legitimacy of the source MAC address so that the network is still vulnerable to attacks based on MAC spoofing. Recently, TopoGuard+ [16] has been presented with extensions to TopoGuard to defend against two new attacks; port probing and port amnesia. Both TopoGuard and SPHINX were identified as being vulnerable to these attacks. The connection authentication in OFMTL-SEC provides protection against these attacks.

PacketChecker [17] proposes an improvement over TopoGuard by binding the switch port with the host’s MAC address in the SDN controller. This offers a solution to the packet injection attack through verification of packet-in messages. However, similar to TopoGuard, PacketChecker processes the packet at the controller to compare MAC address, Switch Datapath ID and Import information. Upon malicious packet-in message detection, a flow rule is inserted to the data plane to drop subsequent packets, which is intended to limit the overhead on the control channel.

SecureBinder [6] was identified in Section I as a solution to prevent identifier binding attacks (e.g. ARP spoofing) in SDN. Based on the alerts generated by a false binding attempt, SecureBinder also offers protection against the port probing attack described in [16]. However, the solution is implemented as a controller application, which introduces latency due to
Table I: Comparison of related work

<table>
<thead>
<tr>
<th></th>
<th>Stateful DP</th>
<th>SDN Mode</th>
<th>Host Location Hijack Protection</th>
<th>ARP Spoofing Protection</th>
<th>Control Plane Saturation Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>TopoGuard [5]/TopoGuard+ [16]</td>
<td>✗</td>
<td>Reactive</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>PacketChecker [17]</td>
<td>✗</td>
<td>Reactive</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>SPHINX [15]</td>
<td>✗</td>
<td>Reactive</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LineSwitch [18]</td>
<td>✓</td>
<td>Reactive/Proactive</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>OFMTL-SEC</td>
<td>✓</td>
<td>Reactive/Proactive</td>
<td>✓ (uses TopoGuard)</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

the control communication. Similarly, SARP_NAT [19] is a control plane based solution to ARP spoofing by sanitizing the ARP requests and checking the legitimacy of ARP replies.

More generally, DoS on the control channel, also known as the control plane saturation attack, has been highlighted in [18], [20]. LineSwitch [18] uses proxying and blacklisting to prevent the attack traffic from reaching the control plane and avoid the issue of buffer saturation introduced by switch proxy solutions such as AVANT-GUARD [20], which implements a SYN proxy mechanism in each OF switch. Similar to LineSwitch, OFMTL-SEC removes the possibility for an attacker to use a spoofed network address and is also designed to avoid the introduction of a further vulnerability due to switch resource usage (i.e. flow table overflow).

The capabilities of OFMTL-SEC in comparison with the state of the art solutions are shown in Table I.

III. ATTACK MODEL

In this section, we present the attack models for the SDN configuration-based and ARP spoofing attacks.

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 1 with H1 relocating from port 1 to port 6. When H1 is disconnected from port 1 (1) and connected to port 6, the switch receives the first packet from H1 on port 6 and a packet_in 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).

In the host location hijacking attack, a malicious host exploits this process to send a fake relocation message to the controller triggering network reconfiguration. The fake relocation message is a spoofed packet (or series of packets) that contains the victim’s MAC address as their Ethernet source address. The aim is to confuse the SDN controller into thinking that the victim has moved to the attacker’s location so that the attacker will receive traffic destined to the victim. A successful attack can isolate a legitimate host from the network while repeated triggering of the relocation process can lead to DoS on the control channel (control plane saturation attack).

B. ARP Spoofing attack

There are three main methods for ARP spoofing, each of which aims to falsify the MAC-IP binding:

1) Gratuitous ARP request with the attacker’s own Sender Hardware Address (SHA), and the victim’s Sender Protocol Address (SPA), and Target Protocol Address (TPA). Each recipient caches a mapping of the attacker’s SHA with the victim’s SPA so that subsequent traffic addressed to the victim will be sent to the attacker.

2) Gratuitous ARP reply with the victim’s SPA and the attacker’s TPA. Each recipient will update its ARP cache with the victim’s SPA and attacker’s SHA.

3) Malicious Reply to Legitimate Request with the victim’s SPA and the attacker’s TPA. Each recipient will update its ARP cache with the victim’s SPA and attacker’s SHA.

As previously noted, the ARP spoofing attack is a precursor to further network attacks e.g. MITM, access control bypassing etc. Some SDN specific attacks leveraging ARP cache poisoning are described in [21].

IV. SOLUTION DESIGN

For the two attack types described in Section III, we describe the OFMTL-SEC solutions.

A. CFGSec (Configuration Security)

As described in Section III, the normal SDN host mobility process involves controller communication, which enables the SDN configuration-based attacks. The objective of the state-based security solution is to remove the load on the SDN controller. We isolate the SDN controller (control path) and introduce intelligence to the data path to self-monitor relocation messages. The only time that the controller interacts
with the switch is at the initial setup phase when the switch connects to the controller to launch the relevant security application (e.g. CFGSec.py RYU app) and programme the required rules to the switch. Following this initialization, the switch self-manages all configuration/relocation requests locally without further controller intervention. This is separate to the regular traffic processing, which may require controller communication e.g. new flow learning based on switch flow table miss. We describe the configuration attack protection as CFGSec, and introduce two functions; the Host Connection State Table, and the Configuration Request Module (CRM).

1) Host Connection State Table: The connection state table to monitor host-switch connections is shown in Figure 2.

![Figure 2: Host Connection States](image)

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 OF match-action paradigm, an L2-4 packet-based authentication process is proposed. The connection and authentication logic is illustrated in Figure 3.

![Figure 3: Connection Authentication State Machine](image)

In state S1, host H1 is connected to switch SW1, as shown in Figure 2. In this state, H1 is active and normal network operation can proceed. Host H2 is disconnected from switch SW1, as shown in Figure 2, so that the connection state of H2 is inactive, and H2 is in state S2 in Figure 3. 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 CFGSec 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 4 and 5. In CFGSec, a relocation packet_in 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 3, 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 4. 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/S3, the message will be considered valid and the CRM will pass it to the controller, as shown in Figure 5. A request for relocation from 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.

![Figure 4: CFGSec - Attack Scenario](image)

![Figure 5: CFGSec - Update Scenario](image)
B. ARPSec (ARP Security)

Based on the mechanism used in the CFGSec solution described in Section IV-A, a state-based ARP security solution has been developed to prevent ARP based attacks on devices connected to it (e.g. gratuitous ARP request/reply). The host connection authentication process described in Section IV-A1 provides the first layer of the ARPSec security implementation. Once authenticated, when an end host tries to communicate with another host via the switch, it will be allowed but ARP spoofing attempts will be prevented. The second layer of protection for ARP attacks is handled by the ARP_CRM agent. This is an extended version of the CRM (Section IV-A2).

The ARP_CRM checks for authentication of the end point before allowing any ARP based activity on the switch. An authenticated endpoint with legitimate IP, MAC and input port information will be allowed to perform ARP request/reply and subsequently will also be allowed to send traffic via the switch. In the example shown in Figure 6, H1 is not allowed to send ARPs as it is an unauthorized endpoint while H2 is allowed as it has a valid connection and also legitimate MAC, IP and input port combinations.

The IP, MAC and input port information is recorded at initial host connection and stored until the host disconnects from the switch. In the event of an ARP spoofing attack, the unauthorized attempt to send ARPs is detected. For example, referring to Figure 6, if an attempt is made to send ARPs from port 6 with H2’s credentials, the ARP_CRM will check its record of MAC, IP and input port combinations. As H2 with MAC2 and IP2 is recorded in the ARP_CRM as connected and active on port 4, transmission from the spoofed IP and MAC addresses at port 6 will be disallowed. The same IP and MAC will be allowed to send ARP messages by disconnecting at port 4 and reconnecting at port 6 following the authentication sequence described in Section IV-A1. This host integrity check provides protection against ARP spoofing based on gratuitous requests and replies and spoofed replies.

V. IMPLEMENTATION DESIGN

As identified in Section II, OpenState [7] is a stateful data plane implementation for OF-enabled switches. The OpenState framework is leveraged in this work to introduce the state-based security protection mechanisms. CFGSec/ARPSec processing is illustrated in Figure 7. The packet arriving at the switch is first processed by the CRM to determine if there is a record of the MAC/IP/Port information. If a MAC/ARP spoofing attempt is detected, the packet will be dropped. If the packet is determined to be legitimate, it is then processed in the first pipeline stage at Table 0. A new connection will be in state 0 and the packet will be processed according to the specified authentication sequence (stateful processing as described in Section IV). Once authentication is complete, the connection state for traffic from this source will be set to 4 and the packet will proceed to Table 1 for further processing/forwarding. Traffic arriving in Table 0 for which the connection state is 4 will be passed directly to Table 1 for further processing/forwarding.

VI. EVALUATION

This section details the results of the performance evaluation for CFGSec and ARPSec. To enable comparison with prior works, results are provided for host join latency, new flow setup latency, and controller load. TopoGuard [5] and PacketChecker [17] were identified in Section II as alternative solutions to CFGSec. As TopoGuard specifically aims to protect against the host location hijack attack and the source code is available, we compare TopoGuard with CFGSec. Although there is no direct comparison for ARPSec, the closest alternative is SecureBinder [6]. As the source code is not available, we compare based on the published results.

A. Test Platform

The test platform is an Ubuntu 14.04.1 (64-bit) VM running Mininet 2.2.1. We configure a single switch, multiple host topology using CPqD/ofsoftswitch13, which has previously been adapted to implement OpenState [7]. The SDN Controller is Ryu version 3.29, which has also been modified to support OpenState.
Table II: Host Join Latency

<table>
<thead>
<tr>
<th></th>
<th>OpenState</th>
<th>TopoGuard</th>
<th>CFGSec</th>
<th>ARPSec</th>
<th>SecureBinder*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Host</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. join time (s)</td>
<td>1.0045</td>
<td>1.3785</td>
<td>4.0128</td>
<td>4.0142</td>
<td>3.5050</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.0006</td>
<td>0.0946</td>
<td>0.0008</td>
<td>0.0021</td>
<td>-</td>
</tr>
<tr>
<td>Multiple Hosts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg. join time (s)</td>
<td>1.0134</td>
<td>1.3828</td>
<td>4.0136</td>
<td>4.0143</td>
<td>-</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.0012</td>
<td>0.1169</td>
<td>0.0017</td>
<td>0.0019</td>
<td>-</td>
</tr>
</tbody>
</table>

Table III: New Flow Latency

<table>
<thead>
<tr>
<th></th>
<th>w/o OpenState</th>
<th>TopoGuard</th>
<th>CFGSec</th>
<th>ARPSec</th>
<th>SecureBinder*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. latency (s)</td>
<td>0.0086</td>
<td>1.5145</td>
<td>0.0135</td>
<td>0.0142</td>
<td>0.0060</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.0007</td>
<td>0.0987</td>
<td>0.0012</td>
<td>0.0009</td>
<td>-</td>
</tr>
</tbody>
</table>

*Results reported in [6] based on Ubuntu 14.04.4 VM running Mininet 2.2.1, Open vSwitch 2.4.0 and ONOS 1.5.1.

B. Performance Evaluation

1) Host Join Latency: The host join latency is a measurement of the time taken for a host to join the network calculated from the time that a host sends its first packet until the insertion of the first flow rule in the switch flow table. The authentication sequence described in Section IV-A1 introduces additional processing that effects the host join latency. The host join latency measurements for a single and multiple hosts are provided in Table II.

As illustrated by the results in Table II, the impact of the CRM/ARP on host join latency is minimal. The four-fold increase in join time using CFGSec/ARPSec is due to the number of authentication stages (4) introduced in the design. Each packet used for authentication and transition to the next state(s) takes approximately 1 s to process. It is anticipated that this authentication sequence would be configured dependent on the environment i.e. a more complex sequence for a more secure network deployment.

2) New Flow Latency: This is a measurement of latency for new flows to be added and used in the switch. It is calculated from the time that a host sends its first packet through insertion of the flow rule in the switch and receipt of that packet at the destination. The measurement is based on a burst of five packets once the hosts have already joined the network i.e. pre-authenticated endpoints. For a baseline comparison, CFGSec/ARPSec are compared with an implementation of the CPqD switch connected to the Ryu controller without implementing OpenState. The results are presented in Table III. There is a small increase in new flow latency when using OpenState (CFGSec/ARPSec). However, it is a negligible impact on the setup time for new flows.

Comparing TopoGuard with CFGSec, it is clear that the SDN controller communication increases the host join and new flow latencies. As the host join latency results for CFGSec include the authentication sequence, the direct comparison here is between the new flow latency results (Table III). The average new flow setup latency with CFGSec is 0.0135 s compared with 1.5145 s for TopoGuard. Furthermore, the performance of TopoGuard is expected to be significantly slower in the event of a remote controller with greater distance between the SDN controller and the switch. In contrast, CFGSec operates with the CRM directly at the switch.

In terms of ARPSec, the latency introduced for host join and new flow processing is minimal; approx. 5.6 ms for new flow latency compared with the stateless solution. The host join latency for SecureBinder is 3.5 s compared to 4.014 s including authentication of 4 s for ARPSec. The new flow latency for SecureBinder is 6 ms compared to 14 ms for ARPSec. The small performance penalty for ARPSec new flow latency is offset by the control plane saturation protection. These results demonstrate the efficiency of protecting against the configuration and ARP spoofing attacks at the data plane.

3) Switch Load: Flow table occupancy is a standard measure of SDN switch load. Therefore, we analyse the number of rules required for CFGSec/ARPSec. In the developed applications, four states were used to authenticate endpoints. The complexity and number of authentication steps (i.e. key sequence length) determine the number of flow table rules that will be required by the application. Only one heartbeat rule is required in the flow table to service all the heartbeat

Figure 8: No. of Switch Flow Rules for CFGSec/ARPSec
requirements in the state table. Figure 8 shows the number of authentication rules and heartbeat rules (HB) required as a function of the number of hosts and the complexity of the authentication. The number of additional rules per switch is: 

\[ \text{Length of authentication sequence} \times \text{edge ports} + 1 \times \text{HB} \]

The switch load measurement is not applicable to TopoGuard as a control plane solution. However, SecureBinder proposes a similar calculation with additional rules per switch of: 

\[ 26 + 13 \times \text{edge ports} + \text{internal ports} \]

For example, for a 48-port edge switch, 638 flow rules are required. For ARPSec, only 193 flow rules are required.

VII. CONCLUSIONS

The security solutions presented in this paper have been motivated by the new network architecture introduced by SDN and the vulnerabilities identified in this architecture with respect to the control-data plane communication. OFMTL-SEC enables security protection at the switch level without the requirement for controller intervention. In comparison to alternative approaches, OFMTL-SEC has a minimal impact on network processing performance and provides protection against a range of attacks without exposing the network to additional threat e.g. control plane saturation. Both the SDN configuration-based attacks and traditional ARP spoofing/poisoning attacks present the first step in a multitude of network attacks. The ability to protect against these threats offers strong protection to the network on which the OFMTL-SEC solutions are deployed. Future work includes the multi-switch implementation of OFMTL-SEC to protect against attacks across multiple switches, and the evaluation of other security mechanisms leveraging the OFMTL-SEC design.

ACKNOWLEDGEMENT

This work has been supported by Huawei Technologies, German Research Center.

REFERENCES


