Received: 13 August 2019 | Revised: 15 September 2019 | Accepted: 16 September 2019

DOI: 10.1002/itl2.131

USE CASE LETTER

WILEY

Teaching software-defined network security through malicious tenant detection

Farzaneh Abazari[1] | Flavio Esposito[2] | Hassan Takabi[1] | Hamid Hosseinvand[3] | Tommaso Pecorella[4]

[1]Department of Computer Science and Engineering, University of North Texas, Texas,

[2]Computer Science Department, Saint Louis University, Missouri

[3]Technical and Engineering Department, Shahed University, Tehran, Iran

[4]Department of Information Engineering, University of Firenze, Florence, Italy

Correspondence
Farzaneh Abazari, Department of Computer Science and Engineering, University of North Texas, TX 77030.
Email: farzaneh.abazari@unt.edu

Software-Defined Networking (SDN) has radically changed how we manage our network, increasing flexibility, and enabling network programmability. Providing security for tenants is one of the most significant issues in SDN. In this paper, our goal is to introduce with an educational use case, four types of attack vectors associated with malicious tenants that seriously challenge the SDN foundation. We also provide suggestions to teach how to investigate the mitigation of these techniques against the attacks. Our solution, to be tested in the classroom, introduce an approach that utilizes several concepts, such as a topological graph to detect malicious tenants. Unlike other methods to teach how to secure SDNs, our proposed approach teaches how to detect and secure a system without requiring any changes to the underlying protocols.

KEYWORDS

education, malicious tenant, OpenFlow, SDN security, topology poisoning

1 | INTRODUCTION

Software-Defined Networks (SDN) provide a new approach to manage networks. Controllers are the operating systems in SDN, that make network control more programmable and dynamic. Being the OS of the network, an SDN controller provides users a great tool to design and control the underlying network.[1,2]

Security is one of the critical aspects of networking in SDN. Modern architectures in SDN introduces new potential security vulnerabilities in networking.[3]

By surveying the recent literature, we identified three main reasons that make SDN vulnerable to malicious tenants:[4]

(a) SDNs do not have any built-in security mechanism to prevent malicious switches or tenants from packet spoofing. (b) Unlike a traditional network, SDN switches are considered to be dumb forwarding entities that forward packets based on the installed rules on controllers. So many traditional attacks, including ARP poisoning and LLDP spoofing, could be performed on vanilla SDN environments. (c) Software switches such as Open vSwitches are more attractive targets to even naive attackers than hardware switches since it is harder to compromise hardware switches to modify routing rules.

Studying SDN security threats and countermeasures will help students understand why modern networking mechanisms use the SDN paradigm, despite their vulnerabilities, and how such vulnerabilities may lead to other severe damages.

Besides, using SDN vulnerabilities as case studies, students can learn the principles of secure design of SDN applications. However, designing practical case studies that are engaging and that meet course educational objectives requires significant time investments. In this paper, we help towards this goal by presenting a few case studies to demonstrate the vulnerabilities of the SDN and the effect of malicious tenants. The only requirement is Mininet as a lightweight virtualization/container-based emulator. Malicious tenants are considered to generate packets. Meanwhile, various tools are used to perform attacks such as DoS and MITM. Malicious Tenant floods the network with the spoofed traffic from the victim tenant. They can also launch packet injection and topology poisoning as well. Besides, they intercept traffic intended for another tenant. A malicious host can continuously send forged ICMP scanning to form a DoS attack.

We believe the most similar work is by SR et al. in,[5] where the authors employed three different approaches to expose SDN to experimenters while achieving isolation and fair sharing goals. We are different as our goal is to evaluate SDN security by showcasing several experiments. To do so, we present a few attacks that can be performed by a malicious tenant by exploiting vulnerabilities in the OpenFlow protocol or the SDN platform within the network operating system. We demonstrate the feasibility of these attacks in Mininet, a network emulation environment based on namespaces and the Linux traffic control command tc. Finally, we introduce a new approach to detect malicious tenants based on their behavior on the network.

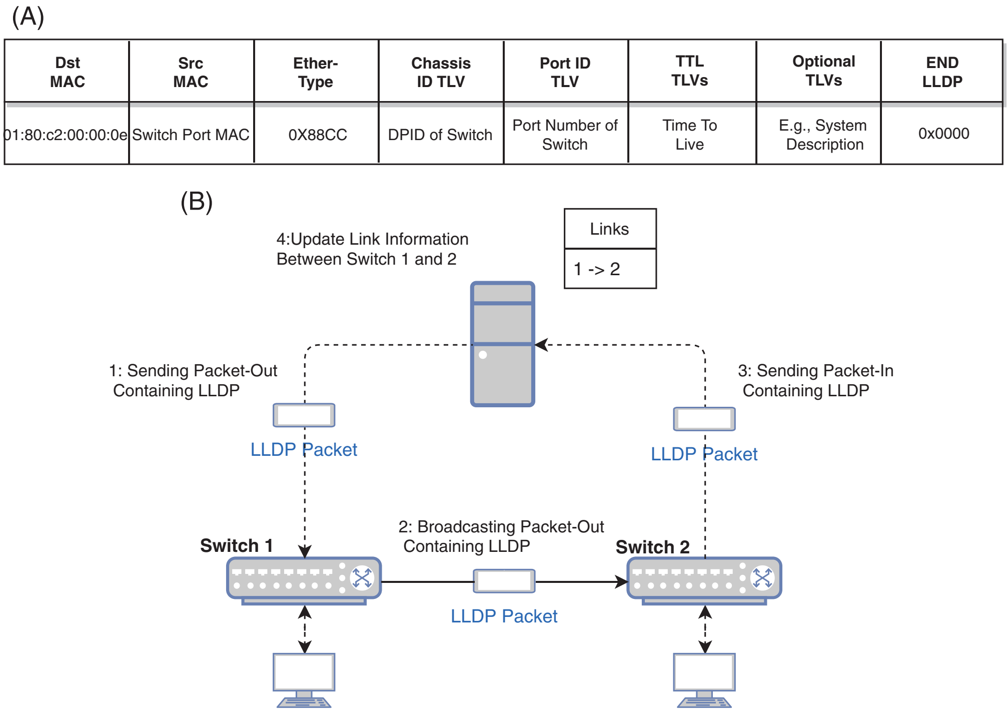
The rest of the paper is organized as follows. An overview of the concepts in SDN is discussed in Section 2. We then explore our considered threat model followed by our experiments on Section 3. Section 4 introduces the proposed approach, and Section 5 concludes the work.

2 | OPENFLOW AND LINK DISCOVERY PROCEDURE BACKGROUND

SDNs propose a promising architecture where the control and data planes are separated. In this section, we explain a few underlying concepts and introduce some definitions.

OpenFlow, as an SDN protocol, offers developers control of the network software. It creates a standard, network-accessible interface to control the data-plane of network equipment. Using OpenFlow, control-plane logic can be moved from individual network devices to a centralized controller. Network protocol changes, traffic engineering requirements, as well as other updates, can be accomplished by reconfiguring the controller instead of upgrading or replacing network hardware.

In order to handle the dynamic nature of tenants in SDN, many controllers often implement Host Tracking Services (HTS) and Link Discovery Services (LDS). HTS keeps a tenant profile that includes a MAC address, an IP address, some location information, and VLAN ID, which is updated dynamically. LDS dynamically discover topology and changes. In contrast with a traditional network, the link service in an SDN network is special due to its logically centralized controller. This service is implemented inside OpenFlow controllers and uses Open Flow Discovery Protocol (OFDP), which transfers LLDP (Link Layer Discovery Protocol) packets through SDN switches. So LLDP packets are used to detect switch-to-switch links (Figure 1A). The process is repeated for each port of all the switches in the network. The whole link discovery procedure is performed periodically at fixed intervals.



(A)

Dst MAC	Src MAC	Ether-Type	Chassis ID TLV	Port ID TLV	TTL TLVs	Optional TLVs	END LLDP
01:80:c2:00:00:0e	Switch Port MAC	0X88CC	DPID of Switch	Port Number of Switch	Time To Live	E.g., System Description	0x0000

(B)
4:Update Link Information Between Switch 1 and 2
Links
1 -> 2
1: Sending Packet-Out Containing LLDP
3: Sending Packet-In Containing LLDP
LLDP Packet
LLDP Packet
2: Broadcasting Packet-Out Containing LLDP
Switch 1
Switch 2
LLDP Packet

FIGURE 1 Link layer discovery protocol. A, Packet. B, Protocol

Figure 1B depicts the link discovery procedure in an SDN. First, the OpenFlow controller creates a single LLDP "Packet-Out" packet for each port on switch *A*. The controller sends these messages to switch *A* with the payload of a controller-specific LLDP packet. The payload of the LLDP packet contains DPID and the output port of switch *A*. When receiving the LLDP packet, Switch *A* broadcasts it to all of its ports. When switch *B* receives the packet, it reports the incoming LLDP packet to the controller with the ingress Port ID and DPID of Switch *B* via a "Packet-In" message. Because all switches have a pre-installed rule in their flow table that forwards any LLDP packet received from any port, except for the controller port, to the controller via an OpenFlow "Packet-In" message. Upon receiving the "Packet-In" message from Switch *B*, the OpenFlow controller can detect a link from switch *A* to Switch *B*.

3 | THREAT MODEL

Malicious tenants in SDN can be used to compromise the whole network. In this section, we introduce the possible threats of a malicious tenant in SDN that we have considered.

3.1 | Topology poisoning attack

First, we analyze a topology poisoning attack.[6] This attack can be performed on an SDN controller to destroy its view of the network by using detrimental "Packet-In" messages sent by the switches. These malicious "Packet-In" messages could be generated by malicious tenants, which can send spoofed LLDP messages to affect connectivity across links between the switches. Since there is no built-in mechanism in the SDN controller to ensure the integrity of LLDP packets, attackers can perform topology poisoning efficiently.[3] In the following, we discuss two scenarios that lead to fake topology attacks: (a) Fabricated LLDP Injection and (b) LLDP Replay. These attacks affect the operation of network applications that are running in the controller, such as packet routing, network virtualization, and mobility tracking.

3.1.1 | Fabricated LLDP injection

In this scenario, an attacker generates intended fake LLDP packets into an OpenFlow network to announce a dummy internal link between two SDN switches. Since the malicious tenant is connected to the switch, by monitoring the traffic from OpenFlow switch, and decoding the LLDP packets, the attacker can extract packet fields. Then, the attacker can capture the specific format of the LLDP packet, because each OpenFlow controller leverages specific syntax and TLVs for verification. Moreover, due to the open-source nature of most controllers, the attacker can find out the format of the LLDP packet. As soon as the attacker captures the LLDP packet and decodes it, he can modify the DPID and port number field of the LLDP packet, and launch the Link Fabrication Attack. Figure 2A shows fake LLDP injection attack process. At the end of the attack, the controller view of the network has changed: a new fake link is added to the links table.

3.1.2 | LLDP replay attack

Instead of injecting manipulated LLDP packets, the attacker can replay a captured traffic through the network, in order to forge target switch without any modification. The result of this attack is a fake topology view in the OpenFlow controller. A fake internal link will be formed between two SDN switches (Figure 2B).

Typical non-SDN attacks can also be performed within an SDN-managed network. For example, malicious tenants and switches can mount DoS attacks by flooding the network with traffic to random tenants to exhaust its resources, switches and

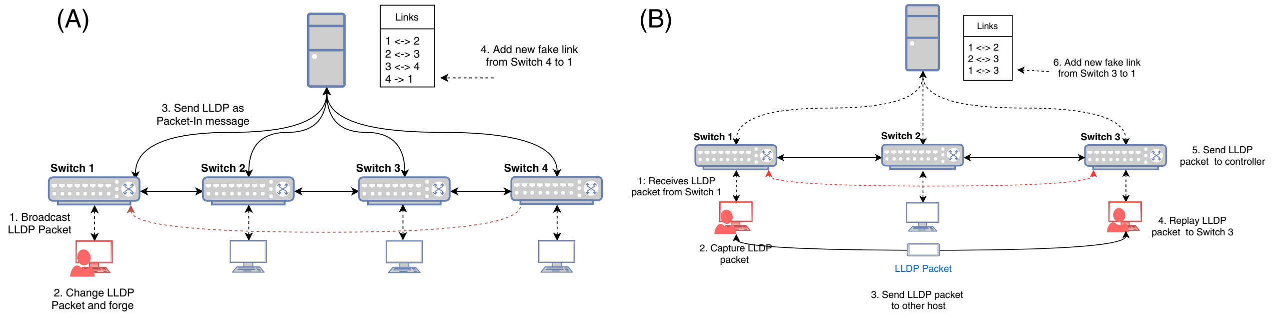

(A)
Links
1 <-> 2
2 <-> 3
3 <-> 4
4 -> 1
4. Add new fake link from Switch 4 to 1
3. Send LLDP as Packet-In message
Switch 1
Switch 2
Switch 3
Switch 4
1. Broadcast LLDP Packet
2. Change LLDP Packet and forge
(B)
Links
1 <-> 2
2 <-> 3
1 <-> 3
6. Add new fake link from Switch 3 to 1
Switch 1
Switch 2
Switch 3
5. Send LLDP packet to controller
1: Receives LLDP packet from Switch 1
4. Replay LLDP packet to Switch 3
2. Capture LLDP packet
LLDP Packet
3. Send LLDP packet to other host

FIGURE 2 Topology poisoning attack. A, Fabricated LLDP injection attack in SDN. B, LLDP replay attack in SDN

SDN controller, thereby affecting forwarding in the data plane. Moreover, a man-in-the-middle (MITM) attack can be performed by sniffing traffic to discover IP addresses on the network, and then perform impersonification of the gateway or victim MAC address by using ARP spoofing.

3.2 | Use case experiment

To propose a method in enhancing the security of SDN, first, we deploy four attacks in Mininet that are performed by malicious tenants to exploit SDN vulnerabilities. In each scenario, first, we build a custom topology with Mininet as a lightweight virtualization/container-based emulator. Next, we set up a Floodlight controller and consider a malicious tenant that can generate packets and send them to the network. In each scenario, various tools can be used to perform the attack, such as Wireshark, Scapy, Ettercap, Ping, and sFlow-RT. Each scenario is described in a separated section. We denote with h a Host/Tenant, and we use S to denote a Switch.

3.2.1 | Fabricated LLDP injection

In this scenario, we deploy fake LLDP attacks in an SDN network. First, we start Wireshark in h_1 then capture the LLDP packets that are sent by switch S_1 and analysis the packet to retrieve LLDP packet components like TLVs and chassis ID. Second, we create an LLDP packet and set the Src Mac, Chassis ID, and Port ID based on the switch S_4 information with the Scapy library and send the forged packet to h_1.

When the attack occurred, a new link will appear between switch S_1 and S_4. Since those ports on the switches that were connected to the tenants, reconnect to other ports in other switches, h_1 and h_2 lose their connection. Floodlight's log after the attack shows an update in attachment between (Switch ID = 4, Port ID = 2) and (Switch ID = 1, Port ID = 1).

3.2.2 | LLDP replay attack

To successfully launch an LLDP replay attack, the attacker first needs to find at least two tenants; a connection test can achieve that. In the attack scenario, we capture LLDP traffic in h_1, replay the captured traffic in another tenant ($h2$) and view the poisoned topology in Floodlight.

3.2.3 | MITM attack

In this scenario, we use Ettercap as a comprehensive tool for MITM attacks to launch a MITM in SDN that causes a malicious tenant to sniff victim tenant traffic. First we open $h1$ terminal and start Ettercap on *h1*. From the Ettercap menu, we scan for tenants. Add victim tenants *h2* and *h3* in tenants list as *Target 1* and *Target 2*. We choose *Arp poisoning* from the MITM tab, then check sniff remote connections and start sniffing. Then we launch the DVWA website on *h3* and enter username and password to log in. Ettercap outputs the username and password.

3.2.4 | DoS attack

In this scenario, we use the Ping command to launch a DoS attack in SDN that causes a decrease in victim tenant performance. SFlow-RT, an analytic software can monitor the traffic between tenants. We start a web server on host *h1* and make a HTTP request from host *h2*. Then we create a sFlow agent to capture the traffic. Finally, we launch the attack by generating a flood directed to *h1*. The total transferred bytes are always less than 1 K during the experiment. However, the total transferred bytes are over 300 K when *h3* runs "*ping -f 10.0.0.1*" command.

4 | PROPOSED EDUCATIONAL APPROACH

Our proposed educational approach involves showing students how to prevent the attacks mentioned above with a methodology based on three steps. First, students should monitor all the controller communication and identify relevant OpenFlow, ARP, scanning, and other relevant packets. Due to the absence of encryption in the control plane, it is possible to capture control and data plane traffic. In the second step, students should analyze this information separately based on their protocol types. For example, OpenFlow packets can be used to build a topological network graph, or ARP packets can be used by Active ARP inspection[7] to detect ARP poisoning attacks. The topology graph approximates the actual network among tenants based on their

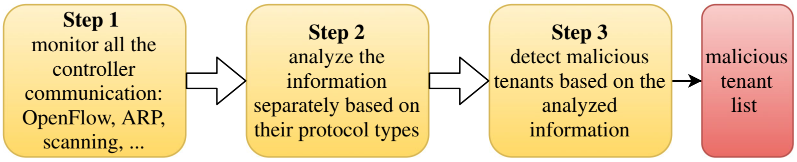

Step 1
monitor all the controller communication: OpenFlow, ARP, scanning, ...
Step 2
analyze the information separately based on their protocol types
Step 3
detect malicious tenants based on the analyzed information
malicious tenant list

FIGURE 3 Malicious tenant detector architecture

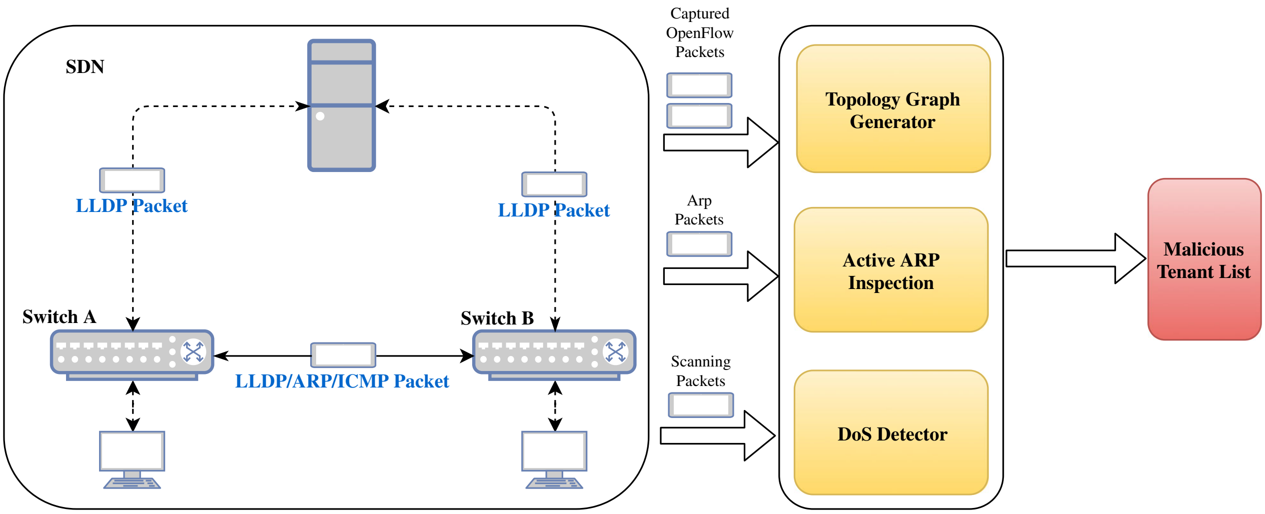


FIGURE 4 Malicious tenant detector architecture

TABLE 1 Summary of the related literature comparing response methods

References	Method	Modification	TP	DoS	MITM
2	Sphinx: Build incremental flow graphs	Yes	✓	—	—
6	Monitor LLDP packets and topology update	No	✓	—	—
8	Considering LLTP switch-to-switch time	Yes	—	—	✓
9	LLDP Packet Authentication	Yes	✓	—	—
10	Using stealthy probing-based verification	No	✓	✓	✓
11	Add encrypted timestamps to LLDP	Yes	✓	—	—
12	Present secure and efficient OFDP	Yes	✓	—	—

communication flow. We analyze specific OpenFlow control messages to build this topology graphs. This graph is continuously updated for changes. A simple program in the first step can passively capture OpenFlow messages, ARP packets, and other relevant protocols. Then, a parser extracts high-level information related to the topology graph, ARP inspection, and scanning attacks. In the last step, students can write code or use visual inspection to detect suspicious tenants. For example, to detect malicious tenants based on the topology graph, students should look for a unidirectional edge in the graph. The tenant who is attached to a switch with a unidirectional edge is suspicious. The output of our method is a Malicious Tenant List. A response module can manage and quarantine malicious tenants in the entire network by changing the flow table strategy.[13,14] Figure 3 depicts the steps and Figure 4 illustrates the components of the proposed approach. We investigate previous approaches used in detecting or mitigating the mentioned threats. Table 1 compares different approaches in detecting malicious tenants.

5 | CONCLUSION AND FUTURE WORK

In this paper, we performed an analysis of vulnerabilities and threats in OpenFlow protocol and SDN infrastructure that could lead to serious attacks. Besides, we introduced four types of attack vectors associated with malicious tenants that seriously challenge the SDN foundation. Then the mitigation techniques against malicious tenants are investigated. Finally, we introduced an approach that utilized the topology graph, ARP inspection, and DoS detector to detect malicious tenant. Despite other methods such as *TopoGuard*,[6] our approach does not need any changes to underlying protocols.

We believe that with our simple approach, students could understand with simple hands-on experiments why SDN is vulnerable, how such vulnerabilities lead to other severe damages, and why many security mechanisms are needed.

ORCID

Farzaneh Abazari https://orcid.org/0000-0002-2139-5684

REFERENCES

1. Mousavi SM, St-Hilaire M. Early detection of ddos attacks against sdn controllers. In Computing, Networking and Communications (ICNC), 2015 International Conference on, pp. 77–81, IEEE; 2015.
2. Dhawan M, Poddar R, Mahajan K, Mann V. *Sphinx: detecting security attacks in software-defined networks*. Proceedings of the 22nd Annual Network and Distributed System Security Symposium *NDSS. Internet Society*; 2015.
3. Khan S, Gani A, Wahab AWA, Guizani M, Khan MK. Topology discovery in software defined networks: threats, taxonomy, and state-of-the-art. *IEEE Communications Surveys & Tutorials*. 2017;*19*(1):303-324.
4. Benton K, Camp LJ, and Small C, Openflow vulnerability assessment. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 151–152, ACM, 2013.
5. Sivaramakrishnan SR, Mikovic J, Kannan PG, Choon CM, Sklower K. Enabling sdn experimentation in network testbeds. In *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pages 7–12. ACM, 2017.
6. Hong S, Xu L, Wang H, and Gu G. Poisoning network visibility in software-defined networks: new attacks and countermeasures. In Proceedings of the 22th Annual Network and Distributed System Security Symposium NDSS, 2015.
7. Xia J, Cai Z, Hu G, Xu M. An active defense solution for Arp spoofing in openflow network. *Chinese Journal of Electronics*. 2019;*28*(1):172-178.
8. Nguyen T-H, Yoo M. A hybrid prevention method for eavesdropping attack by link spoofing in software-defined internet of things controllers. *International Journal of Distributed Sensor Networks*. 2017;*13*(11):1–9.
9. Alharbi T, Portmann M, Pakzad F. The (in) security of topology discovery in openflow-based software defined network. *International Journal of Network Security & its Applications*. 2018;*10*(3):1–16.
10. Alimohammadifar A. Verifying Network Topology in Software Defined Networks Using Stealthy Probing-based Verification (SPV). PhD thesis, Concordia University; 2018.
11. Skowyra R, Xu L, Gu G et al. Effective topology tampering attacks and defenses in software-defined networks. In 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 374–385, Luxembourg: IEEE, 2018.
12. Azzouni A, Boutaba R, Trang NTM, Pujolle G. softdp: secure and efficient topology discovery protocol for sdn. *arXiv preprint arXiv:1705.04527*, 2017.
13. Liu L, Ko RK, Ren G, Xu X. Malware propagation and prevention model for time-varying community networks within software defined networks. *Security and Communication Networks*. 2017;*2017*:2910310:1–2910310:8.
14. Jero S, Bu X, Nita-Rotaru C, Okhravi H, Skowyra R, Fahmy S. *Beads: Automated attack discovery in openflow-based sdn systems*. In: Dacier M, Bailey M, Polychronakis M, Antonakakis M, eds. *International Symposium on Research in Attacks, Intrusions, and Defenses*. Atlanta, GA: Springer; 2017:311-333.

How to cite this article: Abazari F, Esposito F, Takabi H, Hosseinvand H, Pecorella T. Teaching software-defined network security through malicious tenant detection. *Internet Technology Letters*. 2019;2:e131. https://doi.org/10.1002/itl2.131