Towards 5G Intrusion Detection Scenarios with OMNeT++

Katina Kralevska, Michele Garau, Mathias Førland, and Danilo Gligoroski

Department of Information Security and Communication Technology
NTNU – Norwegian University of Science and Technology, Trondheim, Norway
{katinak, michele.garau}@ntnu.no, mathiaskfoerland@gmail.com, danilog@ntnu.no

Abstract

We implement an intrusion detection application to investigate the security capabilities of Software Defined Networking (SDN) in a 5G-like environment under Distributed Denial-of-Service (DDoS) attacks. The simulation environment is created in OMNeT++ with a novel integration of two OMNeT++ extension libraries, SimuLTE and OpenFlow OMNeT++ Suite. The 5G-like environment enables vast and diverse testing of 5G topologies, as well as performance analysis of SDN security applications with various detection and mitigation methods. We analyze distributed synchronize (SYN) flood attack performed by compromised nodes. We report our findings about the sensitivity and the specificity of detection and mitigation of SYN flood for different number of attack and benign nodes.

1 Introduction

Two main transformations lead the transition towards the fifth generation mobile network (5G): (i) evolutionary transformation, where higher data rates, capacity and connectivity will be offered by integrating existing and new radio access technologies, and (ii) revolutionary transformation, where heterogeneous services with diverse requirements will be provided over the same infrastructure. Software-Defined Networking (SDN) and Network Function Virtualization (NFV) are the key enabling technologies, in particular, for the second transformation. SDN separates the control plane from the data plane, thus, it enables a centralized control over the network via a SDN controller. NFV is highly complementary to SDN, and it implements network services, that are traditionally run on proprietary dedicated hardware, as network functions on off-the-shelf hardware.

While SDN has been already used to increase link utilization [5], leveraging SDN and NFV for security purposes is still in an early stage. In a security-context, SDN has some useful attributes. The SDN controller has a global view of the network, centralized control and provides programmability of the network elements. It monitors and gathers network statistics through the southbound API. The controller can use its global view of the network to identify malicious patterns, i.e. intrusion detection system (IDS), and its centralized control to respond to events by updating flow tables in the network elements to filter out the traffic or redirect it to an intrusion prevention system (IPS). Each early threat detection at any network location and quick response at run-time is especially important in 5G networks where many use-cases and users will be served simultaneously over the same infrastructure.

The authors in [1] identified eight security challenges in 5G. Two challenges are about the threat of Denial of Service (DoS) attacks, where both the infrastructure and end-user devices are vulnerable. Applying SDN for detection and mitigation of different versions of TCP SYN Scanning attack was proposed in [3]. The authors of [10] implemented four types of security functions with SDN: in-line mode (e.g. firewalls and IPS), passive mode (e.g. IDS), network anomaly detection (e.g. scan and DDoS detector), and advanced security functions (e.g. stateful firewall and reflector networks). Reference [6] proposed running intrusion detection as a service (DaaS) with SDN for anomaly detection system in mobile network operators. The proposed architecture contains several DaaS nodes which analyze every received packet and notify the SDN application when a malicious packet is detected, but it does not specify the detection algorithm used by the DaaS nodes.

In this article, we investigate the security capabilities of SDN in a 5G-like environment by implementing a SDN Distributed Denial-of-Service (DDoS) defense application in a simulation environment where malicious traffic is detected and mitigated. The 5G-like environment is created in OMNeT++ simulator with a novel integration of two OMNeT++ extension libraries SimuLTE [9] and OpenFlow OMNeT++ Suite [4]. In recent years, OMNeT++ [8] has become the reference framework for performing simulations for communication networks due to its peculiar characteristics, such as a free and open source code, a wide list of available libraries, an extensive documentation and an easy-to-use graphical run-time environment. The environment enables vast and diverse testing of 5G topologies, as well as performance analysis of SDN security applications with various detection and mitigation methods. We test the performance against a distributed synchronize (SYN) flood attack performed by compromised users.

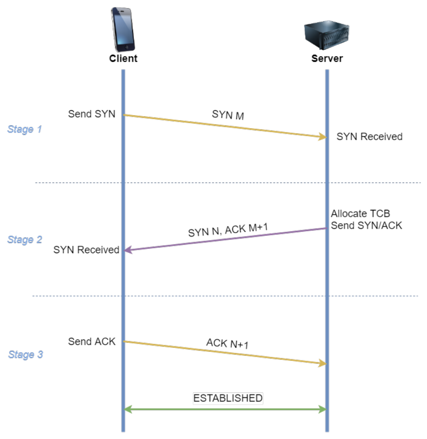
2 SDN based DDoS Defense

As the most frequent Internet attack, Denial-of-Service (DoS) attack presents a realistic threat to 5G. It aims to prevent a targeted network resource from answering and serving requests from legitimate users by exhausting the network resources with requests from an attacker. DDoS attack uses several compromised systems in a coordinated fashion to decrease or hinder the availability of a network service. The collection of compromised systems, called bots or zombies, used in the attack are often referred to as a botnet and are usually distributed globally.

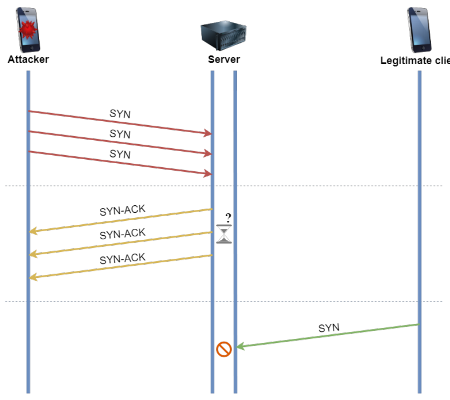
The programmable nature that SDN brings in 5G has some security challenges. For instance, if an attacker gets an access to the controller, then the whole network or part of the network that is controlled by that controller can be monitored and exploited by the attacker. On the other hand, the controller can protect the network from DDoS attack by running detection and mitigation techniques. There are many versions and ways to perform DDoS attack and in this work we focus on the SYN flood attack.

2.1 SYN Flood attack

SYN Flood attack is a protocol exploitation attack that takes advantage of the vulnerability in the second stage of three-way handshake process in Transmission Control Protocol (TCP) given in Figure 1a. The vulnerability of the TCP protocol is that the receiver, i.e. the server, of TCP SYN request creates a half-open connection by initiating Transmission Control Block (TCB) to uniquely identify the connection, and it binds resources on the server to be used when the connection is established. Figure 1b illustrates how an attacker exploits the three-way handshake in a SYN flood attack. The attacker sends a flood of SYN requests to the server, and it makes the server to bind all of its resources and to become unavailable for legitimate



(a)



(b)

Figure 1: (a) TCP three-way handshake, and (b) Simplified SYN Flood attack.

users. Even if the server restarts or frees up all of its resources, the intensity and the length of the SYN flood attack may cause all of the resources to be instantly starved again. The attack can be made even more efficient by exploiting the time that the server waits for an ACK response from a client. The attacker can choose to not reply to all of the SYN-ACK packets from the server or to send a SYN request with a spoofed source IP address, causing the server to send SYN-ACK packet to an IP address which will not reply as it did not send the original SYN request.

2.2 Detection and mitigation techniques

There are several DDoS detection techniques such as entropy-based, machine learning, traffic pattern analysis, and connection rate-based [2]. We use the connection rate-based detection method, where the detection is based on the assumption that malicious hosts have a higher connection attempt rate than benign hosts. In order to detect when the number of SYN packets increases, the controller includes a threshold $\mathbf{T}^*$ that the switch is instructed to compare with its counter value for each SYN flow entry. The value of $\mathbf{T}^*$ is calculated as $\mathbf{T}^* = \frac{\mathbf{T}}{\mathbf{t_i}}$ where $\mathbf{t_i}$ is the flow entry idle timeout.

Once a DDoS attack is detected, mitigation techniques like dropping packets, blocking ports, or redirection of traffic can be easily performed using SDN. SDN can enforce these techniques by sending flow-mod messages to its switches and install new or edit existing flow entries. The implemented logic for the mitigation technique is explained in the next Section.

2.3 Evaluation metrics

The application behaves as a binary classifier as it classifies the packets into two groups: malicious or benign. We use sensitivity and specificity analysis [7] to evaluate the performance of the implemented SDN application by building a confusion matrix. Sensitivity or True Positive Rate (TPR) is a measure of how good the application detects malicious packets and is calculated with Eq. (1) where True Positive (TP) is the number of packets that have been correctly registered as malicious by the switch and therefore dropped and False Negative (FN) is the number of packets that are incorrectly registered as benign by the switch and, therefore, forwarded instead of being dropped. Specificity or True Negative Rate (TNR) is the measure of

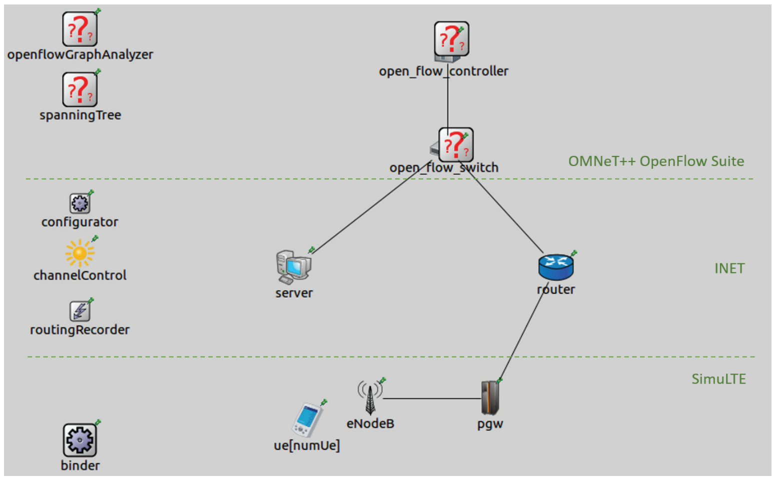


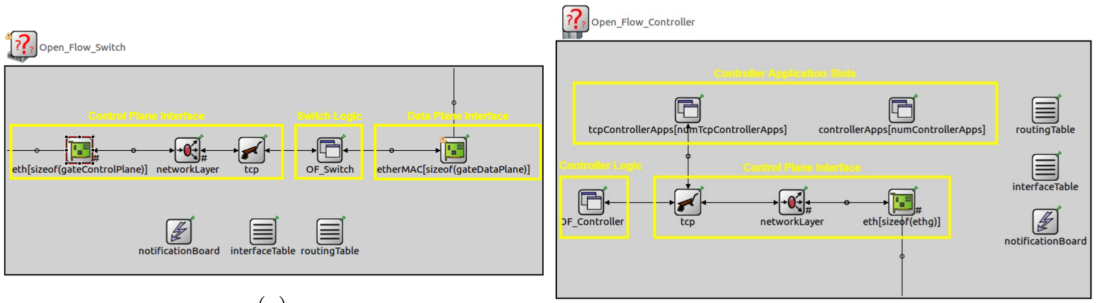
Figure 2: Proposed 5G-like architecture based on OMNeT++ libraries.

how good the application identifies legitimate traffic, defined with Eq. (2), where True Negative (TN) is the number of packets that have been correctly registered as benign by the switch, and therefore forwarded through a port, and False Positive (FP) is the number of packets that have been incorrectly registered as malicious by the switch, and are therefore dropped instead of being forwarded.

$$TPR = \frac{TP}{TP + FN} \quad (1) \qquad\qquad TNR = \frac{TN}{TN + FP} \quad (2)$$

3 IDS OMNeT++ Implementation

To provide OMNeT++ with capabilities of simulating SDN in a 5G-like environment, the third party extension OpenFlow OMNeT++ Suite has been chosen as a set of libraries for SDN-based traffic routing and SimuLTE as a modular system-level simulator for LTE-A networks. OpenFlow OMNeT++ Suite [4] is a library that includes OMNeT++ modules for SDN switch and SDN controller, and it implements the OpenFlow protocol for communication between the switches and the controller. SimuLTE [9] is a simulator based on OMNeT++ that allows simulating LTE and LTE-A wireless networks. It implements a full protocol stack according to LTE standards, and it provides modules for evolved NodeB (eNB), User Equipment (UE), Packet Data Network Gateway (PGW), as well as a realistic representation of scheduling and physical layer. Both libraries are based on the INET library, which provides the most common Internet protocols and support for mobility and wireless technologies. Thus, it allows establishing a flexible and mixed scenario with LTE and SDN as parts of a wider communication network. The working build was installed on Ubuntu 16.04 Operating System, composed of OMNeT++ 4.6, INET 2.5, simuLTE v0.9.1 and OpenFlow OMNeT++ Suite. Figure 2 shows our proposed 5G-like architecture. The topology consists of seven node types: `ue`, `eNodeB`, and `pgw` are modules imported by the SimuLTE library; `open_flow_switch` and `open_flow_controller` are modules taken from the OpenFlow OMNeT++ Suite library; and the server is a generic Internet server that answers requests from UE connected to eNB. In order to interconnect the SimuLTE and OpenFlow OMNeT++ Suite, a standard router was introduced. This topology slightly differs from an ideal 5G scenario, where the eNB would be directly connected to the SDN switch, with



(a) (b)

Figure 3: OMNeT++ modules of: (a) SDN switch and (b) SDN controller.

the LTE protocol processing function being hosted in a remote/cloud server. Yet, all the traffic from UEs to the server flows through the SDN switch, allowing a complete monitoring of the traffic by the SDN controller. This topology can be considered as a valid simplification that does not affect the quality of the results.

The SDN switch and controller represent the core of IDS. Figure 3 shows the OMNeT++ modules of SDN switch and controller. The traffic from the LTE UE to the server is routed through the SDN switch. Through the SDN controller, the SDN switch allows traffic monitoring and it performs mitigation actions in case of an intrusion detection.

A SDN DDoS Defence controller application has been developed. The application controls the behaviour of the `open_flow_controller` by specifying the routing logic, how to handle packet_in messages from the `open_flow_switch` and how to construct flow_mod and packet_out messages. It is implemented as an altered version of the OpenFlow controller application `LearningSwitch`, which is included in the OpenFlow OMNeT++ Suite package. The application functions in two stages, detection phase and mitigation phase.

The `open_flow_controller` keeps an overview of the connection attempt rate of all active hosts in the network. Instead of keeping a table with the state of all SYN attempts for each IP address in the network, the controller separates TCP SYN packets from other packets.

Our simplified DDoS detection and mitigation techniques are given in Algorithm 1 together with the C++ scripts of the OpenFlow OMNeT++ Suite that have been changed. The matching is done on the fields for Ethernet type, incoming port, source MAC address, destination MAC address and TCP SYN flag, while special flow entries are made for any packet that has the TCP SYN flag raised. These special flow entries include additional match fields of the ACK flag, IP source address and IP destination address. The ACK flag is included to separate SYN messages from SYN-ACK messages. The IP source and destination addresses are included to identify the host that sends an abnormally high number of TCP SYN messages to a specific target (lines 1 to 3 on Algorithm 1). For each SYN packet, if the pair (IPsource, IPdestination) is not already categorized as malicious (line 5), then the switch compares the threshold value $\mathbf{T}^*$ with its counter value (line 9). Once the threshold of a flow entry is surpassed, the `open_flow_switch` is instructed to transmit packet_in that contains the triggering packet and a unique value for the field packet_in_reason to the `open_flow_controller`. The controller application replies to the special packet_in with flow_mod message instructing the switch to change the action of the respective flow entry to drop (line 10), as well as a packet_out message instructing the switch to drop the triggering packet (line 11). Any SYN packet that matches this flow entry is dropped once it enters the switch, mitigating the exhausting attack on the server. Note that this method

Algorithm 1 DDoS attack detection and mitigation

```
Input: Switch OF_SW receives data packet PKT
Output: OF_SW manages PKT according to DDoS detection rules
 1: OF_SW matches PKT fields                                    ▷ LearningSwitch.cc
 2: if TCP SYN flag is raised then                              ▷ LearningSwitch.cc
 3:     Check ACK flag, IP source, IP destination, $t_i$, $t_h$      ▷ LearningSwitch.cc
 4:     if not SYN-ACK then                                     ▷ Flow_Table.cc
 5:         if (IPsource, IPdest) $\leftarrow$ Table_DDoS then          ▷ Flow_Table.cc
 6:             Drop PKT                                        ▷ Flow_Table.cc
 7:         else
 8:             Counter++ $\rightarrow$ SYN (IPsource, IPdestination, $t_i$, $t_h$)   ▷ Flow_Table.cc
 9:             if Counter > T* then                            ▷ Flow_Table.cc
10:                 (IPsource, IPdestination) $\rightarrow$ Table_DDoS          ▷ Flow_Table.cc
11:                 Drop PKT
12:             else
13:                 if $t_i$, $t_h$, expired then
14:                     Drop PKT
15:                 else
16:                     Forward PKT from IPsource to IPdestination
17:                     Reset Counter of (Ipsource, IPdestination) to 0   ▷ Flow_Table.cc
18:                 end if
19:             end if
20:         end if
21:     end if
22: end if
```

can be bypassed easily by an attacker that uses spoofed IP addresses, as the detection method relies heavily on packet IP address field, but this is beyond the scope of this experiment.

The OMNeT++ framework allows the creation and transmission of signals during a simulation for statistics purposes. Four signals have been implemented in the code to register respectively TP, TN, FP, and FN that are used for calculating the sensitivity and specificity.

4 Results and Analysis

The topology used in the experiment is based on the architecture depicted in Figure 2, where two types of User Equipments (UEs) are considered: `ue` nodes, which represent the legitimate users that generate legitimate traffic towards the server; `ueMal` nodes, which represent the attackers coordinated in order to reproduce a behaviour similar to bots in a botnet. Each UE node creates a new TCP connection with the server at randomly chosen time with a uniform probability distribution within every 10s window, i.e., 1-10s, 11-20s, 21-30s, 31-40s, and 41-50s, and transmits data with randomly chosen byte size (with uniform distribution between 100 bytes and 2000 bytes). The data transmission start time is randomly and uniformly distributed. The `ueMal` nodes transmit TCP SYN messages to the server without replying to the SYN ACK in order to create half-open connection and exhaust the server's resources. Each `ueMal` node transmits a wave of 10 SYN messages approximately every 3s for a total duration of 29s. The first wave is transmitted after approximately 2s. This equates to 10 waves in a flood attack, where a wave lasts for 1s. The simulation runtime was set to 50s to be able to test a fully

Table 1: The parameters used for the simulation.

Parameter	Value	
	Case 1	Case 2
Number of benign UEs	20	0, 10, ..., 70
Number of malicious UEs	0, 10, ..., 70	20
Simulation runtime	50s	
Packet rate pr UE	0.1 data packets per second (pps)	
Number of malicious UEs	40	
Packet rate pr malicious UE	3.5 pps	
SYN-RECEIVED Timer	75s	
SYN-ACK Retransmissions	Loop: 3s - 6s - 12s	
Flow entry timeout	10s	
Detection threshold $\mathbf{T}^*$	0.5 pps	
TCP Algorithm	TCP Reno	

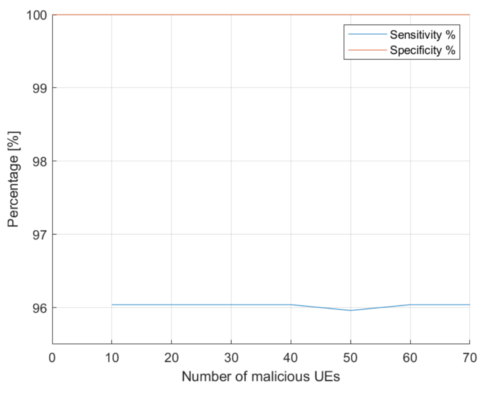
active SYN flood attack against the controller application. To explore how the developed SDN DDoS application performs against SYN flood attacks, two experiments with parameters given in Table 1 are conducted and repeated eight times:

- **Case 1**: The number of malicious UEs varies, the number of benign UEs is constant;
- **Case 2**: The number of benign UEs varies, and the number of malicious UEs is constant.

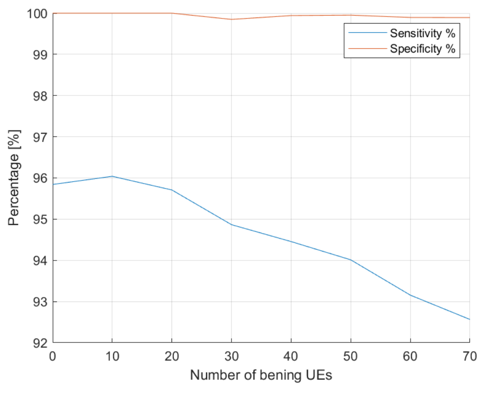
Figure 4a shows the sensitivity and specificity for a varying number of malicious UEs (Case 1), while Figure 4b presents the same metrics for different number of benign UEs (Case 2). The results for Case 1 show that the sensitivity and the specificity remain the same for any number of malicious UEs for the considered range. The sensitivity value is around ∼96%, and the specificity value is constant 100%. The explanation is that the OpenFlow enabled switch creates a flow table entry for each malicious UE, but real-life switches do not have unlimited memory for flow table entries. Thus, in the simulation, the maximum number of malicious nodes does not surpass the limit of flow entries in the switch, and therefore the sensitivity and the specificity are not affected by it. The results for Case 2 indicate a continuous decline in the detection application's sensitivity as the number of benign UEs increases. Note that the controller achieves a specificity value even with 0 benign UEs in the experiment. This is caused by the legitimate SYN-ACK messages generated by the server as a response to the malicious SYN messages received from the attack nodes. The number of malicious nodes does not impact the performance of the application, which implies that the controller application could scale well in terms of the number of attack nodes, given that the OpenFlow-enabled switch has sufficient memory for flow tables. The sensitivity performance of the application seems to suffer when the number of benign nodes (and, by extension, network traffic) increases while the malicious traffic remains the same. The reason for the decrease in performance can be the increased transmission delay in the network caused by the increased network traffic, which causes a greater delay between malicious SYN packets and thereby a lower attack rate.

5 Conclusion

We presented a novel integration of OMNeT++ extension libraries, INET, SimuLTE and Open-Flow OMNeT++ Suite, for implementing SDN DDoS Intrusion Detection application in a 5G-like scenario. The presented results showed that the application mitigates effectively SYN flood



(a)



(b)

Figure 4: The performance of the SDN application in regards to sensitivity and specificity when: (a) varying the number of malicious UEs; (b) varying the number of benign UEs.

attacks and provides a proof of concept for SDN as a security component. The implemented framework is promising for further tests and comparisons of different SDN-based algorithms for detecting and mitigating malicious attacks in 5G-like scenarios.

References

[1] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtov. Overview of 5G Security Challenges and Solutions. *IEEE Communications Standards Magazine*, 2(1):36–43, 2018.

[2] Narmeen Zakaria Bawany, Jawwad A. Shamsi, and Khaled Salah. Ddos attack detection and mitigation using sdn: Methods, practices, and solutions. *Arabian Journal for Science and Engineering*, 42(2):425–441, Feb 2017.

[3] K. Cabaj, M. Gregorczyk, W. Mazurczyk, P. Nowakowski, and P. Żórawski. SDN-based Mitigation of Scanning Attacks for the 5G Internet of Radio Light System. In *Proceedings of the 13th Int. Conf. on Availability, Reliability and Security*, ARES 2018, pages 49:1–49:10. ACM, 2018.

[4] N. Gray, T. Zinner, S. Gebert, and P. Tran-Gia. Simulation framework for distributed SDN-controller architectures in OMNeT++. In *Int. Conf. on Mobile Netws. and Mgmt.*, pages 3–18. Springer, 2016.

[5] S. Jain et al. B4: Experience with a globally-deployed software defined WAN. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 3–14. ACM, 2013.

[6] M. Monshizadeh, V. Khatri, and R. Kantola. Detection as a service: an SDN application. In *19th Int. Conf. on Advanced Comm. Tech. (ICACT)*, pages 285–290. IEEE, 2017.

[7] David MW Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation, school of informatics and engineering, flinders university, adelaide, australia. Technical report, TR SIE-07-001, Journal of Machine Learning Technologies 2: 1 pp. 37-63, 2007.

[8] A. Varga and R. Hornig. An overview of the OMNeT++ simulation environment. In *Proc. of 1st Int. conf. on Sim. tools and tech. for comm., networks and systems & wksh*, page 60. ICST, 2008.

[9] A. Virdis, G. Stea, and G. Nardini. SimuLTE-A modular system-level simulator for LTE/LTE-A networks based on OMNeT++. In *4th Int. Conf. On Simulation And Modeling Methodologies, Techs And Apps (SIMULTECH)*, pages 59–70. IEEE, 2014.

[10] C. Yoon et al. Enabling security functions with SDN: A feasibility study. *Comp. Netws.*, 85:19 – 35, 2015.