Preventing DDoS with SDN in 5G

Mathias Kjolleberg Forland, Katina Kralevska, Michele Garau, and Danilo Gligoroski
Department of Information Security and Communication Technology, NTNU, Norway
Email: mathiaskfoerland@gmail.com, {katinak, michele.garau, danilog}@ntnu.no

Abstract—In this paper we set a 5G-like environment using the OMNeT++ Discrete Event Simulator with integrated extension libraries INET, SimuLTE, and OpenFlow OMNeT++ Suite, and we develop a Software-Defined Networking (SDN) Distributed Denial-of-Service (DDoS) defense controller application for detecting and mitigating distributed SYN flood attack. The developed application serves as a Proof of Concept for the potentials of using SDN as a security component in 5G. The performance of the controller application is evaluated by a sensitivity and specificity analysis, and it is validated through numerous experiments where parameters such as attack rate, detection threshold, flow entry timeout, and the number of malicious and benign nodes are varied. The results show that SDN developed applications can be indeed used as security components in 5G.

Index Terms—5G security, Software Defined Networking (SDN), DDoS, SYN flood attack

I. Introduction

The fifth generation (5G) of mobile communications systems is expected to be commercially launched in 2020, and it is anticipated as a disruptive technology that provides among other things critical infrastructure use cases like autonomous vehicles and smart cities. Two main transformations lead the transition towards 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) [1] and Network Function Virtualization (NFV) [2] are the key enabling technologies of 5G (in particular, for the second transformation) as they introduce the concepts of separation of data and control planes and implementation of network services as network functions on off-the-shelf hardware. In this way, SDN reduces the (re)configuration complexity and promotes flexibility and customization of the network, while NFV makes proprietary dedicated hardware obsolete.

While SDN has been already used to increase link utilization, for instance in Google's B4 network [3], leveraging SDN and NFV for security purposes is still in an early stage [4], [5]. Some crucial security issues have been identified for 5G in [6], including the increased threat of Distributed Denial-of-Service (DDoS) attacks as the number of connected devices is expected to grow 10 to 100 times in 5G. 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, and it communicates with the network elements with OpenFlow protocol. In order to prevent from security attacks, the programmable software oriented environment allows most network functions to be implemented as SDN applications. 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 concept of Security-as-a-Service (SECaaS) was initially introduced in [4] where cloud providers provide security mechanisms, and reference [7] further applies it to 5G verticals. Detection as a Service (DaaS) for detecting anomalies in the network traffic was proposed in [8]. The SDN 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. Reference [5] applied SDN for detection and mitigation of different versions of Transmission Control Protocol (TCP) synchronize (SYN) scanning attack. There the SDN controller tracks the state of each TCP connection that gets attempted through the switch by receiving a copy of every incoming packet by the switch. The TCP connection state information is used to count the number of unsuccessful connection attempts by a host. If the number of unsuccessful connection attempts during a predefined window of time exceeds a preset threshold, then the source IP address of the attempts gets banned for a certain time by the controller adding a flow entry in the switch that drops all packets from that IP source address.

In this article, we first build a simulation of a 5G-like environment in OMNeT++ discrete event simulator [9] and its INET library with a novel integration of two extension libraries: SimuLTE [10] and OpenFlow OMNeT++ Suite [11]. Our 5G-like environment enables testing of diverse 5G topologies, as well as performance analysis of SDN security applications with various detection and

978-1-7281-0960-2/19/$31.00 ©2019 IEEE

mitigation methods. Then we develop one SDN controller application for a defense from a specific DDoS attack, and report our findings for its rate of malicious traffic detection and mitigation. More specifically, we test the performance against a distributed synchronize (SYN) flood attack performed by compromised users. We present results of numerous experiments with varying parameters such as: the number of attack nodes, the number of benign nodes, the detection threshold, the flow entry timeout and the attack rate. The results are quantified and validated using an evaluation method.

The remainder of the paper is organized as follows. Section II gives details on the OMNeT++ simulation. Section III explains SYN Flood DDoS attack, the simulation scenario, the implemented detection and mitigation techniques, and the metrics used for their evaluation. In Section IV, we discuss the simulation results, and Section V concludes the paper.

II. IDS OMNeT++ Implementation

In order to create a simulation framework where SDN-based 5G-like scenarios can be reproduced and analyzed, different third party OMNeT++ extensions have been merged, namely OpenFlow OMNeT++ Suite, SimuLTE and INET. OpenFlow OMNeT++ Suite [11] is a library that includes OMNeT++ modules for SDN switch and SDN controller, and implements the OpenFlow protocol for the communication between the switches and the controller. In addition to the common single controller approach, the library also allows composing horizontally and hierarchically distributed control architectures. SimuLTE [10] 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 provides modules for eNode-B (eNB), User Equipment (UE), Packet Data Network Gateway (PDN-GW), 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, 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. For guaranteeing a complete compatibility of the mentioned libraries, the following versions were chosen: OMNeT++ 4.6, INET 2.5, SimuLTE v0.9.1 and OpenFlow OMNeT++ Suite.

Fig. 1 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. The server is a generic Internet server that answers requests from UE connected to the LTE base station 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

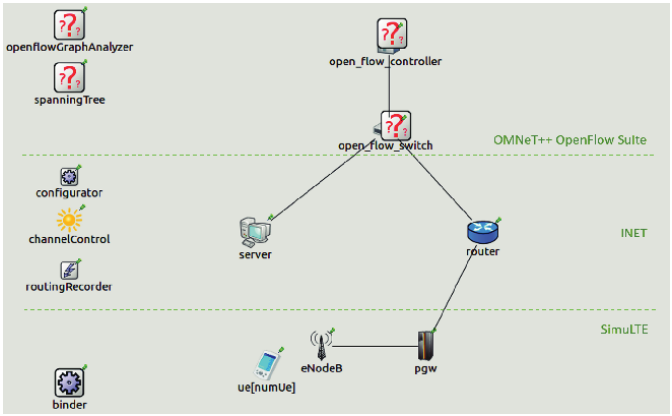

Fig. 1. Proposed 5G-like architecture based on OMNeT++ libraries.

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 switches and the controller represent the core of IDS. The traffic from the LTE UE to the server is routed through the SDN switch. Through the SDN controller, the SDN switch allows monitoring the traffic and, in case of intrusion detection, performs mitigation actions.

To the best of our knowledge, this is the first combination of SimuLTE and OpenFlow OMNeT++ Suite packages in OMNeT++ and therefore, this platform development represents a novel contribution. By integrating the two extension frameworks, we are able to create a simulation environment that facilitates testing of close-to 5G scenarios. The modularity of OMNeT++ enables the creation of various 5G-like topologies which can be used to experiment with SDN controller applications in a single or multiple controller architecture. OMNeT++ provides tools for collecting statistics during simulation, which can be utilized to evaluate the performance of detection algorithms and mitigation methods.

III. DDoS attacks in 5G

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 resource with requests from the attacker. DDoS attack uses several compromised systems in a coordinated fashion to decrease or hinder the availability of 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.

A. 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) [12]. The vulnerability of the TCP protocol is that the receiver, i.e. the server, of a 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. 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 to connect with legitimate users. Even if the server restarts or frees up all 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 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.

B. Simulation setup

The general parameters used in the experiment are specified in Table I. The simulation runtime is set to 50s in order to limit the real runtime of the experiment and still be able to test a fully active SYN flood attack against the controller application. The maximum number of simultaneously connected and active users per eNB could be from 60 to 100 devices. We assume that the majority of the connected devices to the eNB is corrupted, therefore 20 benign UEs and 40 malicious UEs. The packet rate of both the benign UEs and malicious UEs is not set to necessarily mimic real-life values, but to have a proportionate relationship where the malicious packet rate would be drastically higher than the benign packet rate. The default value set in OpenFlow OMNeT++ Suite is used for the SYN-RECEIVED timer and SYN-ACK Retransmissions, which is set according to [13]. If an acknowledgement (ACK) is received before the timer runs out, the connection is completed. If not, the connection attempt is aborted and the reserved resources for the connection are released. The default value set in OpenFlow OMNeT++ Suite for the idle timeout of flow entries is used. The detection threshold is set according to the used packet rate generated by the UEs. Depending on the generated events and included nodes, the actual simulation time of a conducted experiment ranges from 15 to 60 minutes.

C. Generating legitimate traffic

The experiment includes 20 UE nodes to represent legitimate users that generate legitimate traffic. Each UE node creates a new TCP connection at a time chosen uniformly random within every 10s window, i.e., 1-10s, 11-20s, 21-30s, 31-40s, and 41-50s, with the server and transmits data with a randomly chosen byte size in a uniform distribution between 100 bytes and 2000 bytes.

TABLE I
Parameters used for the simulation.

Parameter	Value
Simulation runtime	50s
Number of UEs	20
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	0.5 pps
TCP Algorithm	TCP Reno

The data transmission start time is also chosen randomly in uniform distribution.

D. Attack scenario

The TCP SYN Flood attack is performed by 40 ueMal nodes which represent the attackers. The attack nodes have identical behaviour to imitate bots in a botnet. The attack is designed as a DDoS attack targeted at the server. The ueMal nodes transmits TCP SYN messages to the server without replying to the TCP SYN-ACK response in order to create half-open connections and exhaust the server's resources. Each ueMal node transmits a "wave" of 10 SYN messages every 3s for a total duration of 29s. The first wave is transmitted after 2s in simulation time. This equates to 10 waves in a flood attack, where a wave lasts for 1s.

E. SDN Controller application

To evaluate the security capabilities of SDN, a SDN DDoS Defence controller application has been developed for this experiment. The application determines 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 the Flow_Mod and Packet_Out messages. It is implemented as an altered version of the OpenFlow Controller application "LearningSwitch" which is included in the OpenFlowOMNeT-Suite package. The application functions in two stages, with a detection phase and a mitigation phase given in Algorithm 1.

1) Detection method: The detection method is a simple version of a connection rate-based detection method, where detection is based on the assumption that malicious hosts have a higher connection attempt rate than benign hosts. To utilize the shared characteristics of the attacker hosts, the open_flow_controller needs to keep an overview of the connection attempt rate of all active hosts in the network.

Instead of the controller keeping a table with the state of all SYN attempts for each IP address in the network, it separates normal packets from TCP SYN packets. Flow entries for normal flows are installed with match fields for Ethernet type, incoming port, source MAC address,

```
OF_SW matches PKT fields;
if TCP SYN flag is raised then
    Check ACK flag, IP source, IP destination, t_i,
     t_h;
    if not SYN-ACK then
        if (IPsource, IPdest) ← Table_DDoS then
            Drop PKT;
        else
            Counter++ → SYN (IPsource,
             IPdestination, t_i, t_h);
            if Counter > T then
                (IPsource, IPdestination) →
                 Table_DDoS;
                Drop PKT;
            else
                Forward PKT from IPsource to
                 IPdestination;
                Reset Counter of (IPsource,
                 IPdestination) to 0;
            end
        end
    end
end
```

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.

destination MAC address and TCP SYN flag, while special flow entries are made for any packet that has the TCP SYN flag raised. These SYN flow entries include the 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 hosts that send an abnormally high number of TCP SYN messages to a specific target.

In each flow entry for SYN packets, the controller includes a threshold value **T** that the switch is instructed to compare with its counter value for each SYN flow entry. The threshold value is a number set dependent on the flow entry idle timeout $\mathbf{t_i}$ and hard timeout $\mathbf{t_h}$. All flows have an idle timeout of 10s as the default value and do not utilize $\mathbf{t_h}$ as it is not implemented in the used version OpenFlow OMNeT++ Suite. The idle timeout is reset each time a new packet matches a flow entry as long as the timeout has not expired. That means that the threshold potentially can be surpassed even with a low packet rate if the idle timeout is reset right before it expires every time.

The detection method does not distinguish between completed connection attempts and failed connection attempts mainly to simplify the solution, but also because of the assumption that the malicious connection attempt rate is much higher than the legitimate connection attempt rate, and therefore the completed connection attempts can be considered as negligible in this experiment.

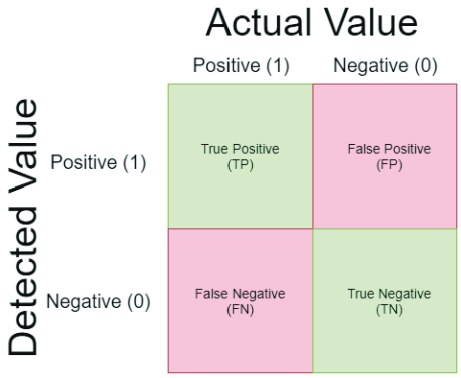

Fig. 2. Contingency table or confusion matrix.

2) Mitigation method: Once the threshold of a flow entry is surpassed, the open_flow_switch is instructed to transmit a Packet_In that contains the triggering packet and unique value for the field packet_in_reason to the open_flow_controller. The controller replies to the special Packet_In with a Flow_Mod message instructing the switch to change the action of the respective flow entry to "drop", as well as a Packet_Out message instructing the switch to drop the triggering packet. Any SYN packet that matches this flow entry is dropped once it enters the switch, mitigating the exhausting attack on the server. Note: This method can be bypassed easily by an attacker that uses spoofed IP addresses, as the detection method relies heavily on the packets' IP address field, but this is beyond the scope of the experiment. The experiment is designed to provide a Proof of Concept (POC), knowing that the application only functions against a specific attack. The POC implies that more intricate and improved detection and mitigation methods can be implemented in SDN, using other logic, analysis, match fields and actions than the current method.

F. 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 [14] to evaluate the performance of the implemented SDN application by building a confusion matrix (shown in Fig. 2).

Sensitivity, or True Positive Rate (TPR), is a measure of how good the application detects malicious packets and is calculated in 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 how good the application identifies legitimate traffic, defined in 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 forwarded.

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

IV. Results and Analysis

The initial results were produced using the parameters showed in Table III-B, and had a sensitivity of 96% and a maximal performance of 100%. The sensitivity of 96% is a promising initial result and indicates a good performance in this environment, where we assume that the undetected 4% represents the first malicious packets of the attack that go undetected until the detection packet rate threshold is reached. The specificity performance of 100% can be explained by the relatively low packet rates of the benign UEs, which averages 0.1 pps and consequently do not surpass the maximum packet rate detection threshold of 0.5 pps or the minimum packet rate threshold as the simulation lasts 50s, before the threshold can potentially be surpassed. To further explore how the SDN DDoS Defense application performs against SYN flood attacks and to explain the initial results, five additional experiments have been conducted by varying the following parameters individually: 1. Number of malicious UEs; 2. Number of benign UEs; 3. Flow entry timeout; 4. Malicious packet rate; 5. Detection threshold.

A. Performance when varying the number of malicious nodes

In this experiment, the initial parameters presented in Table I are used for the simulation, except for the number of malicious UEs. The experiment is repeated 8 times with the number of attack nodes starting at 0 and increasing by 10 nodes for each repetition until it reaches 70 attack nodes in the network. The reason for testing with the minimum value of 0 is to evaluate the specificity performance of the detection application in a network with only legitimate traffic. The maximum parameter value of 70 malicious nodes is chosen to evaluate how the application performs in an environment with an overwhelming amount of generated malicious traffic compared to legitimate traffic. The increased value of 10 malicious nodes for each repetition allows us to evaluate how the number of malicious nodes impacts the performance indicators as it grows.

Fig. 3 show that both sensitivity and specificity remain the same for any number of malicious UEs. The specificity value is at ∼96%, and the specificity value is at 100%. The OpenFlow enabled switch creates a flow table entry for each malicious UE which can affect the performance of the switch as real-life switches do not have unlimited memory for flow table entries. In the simulation environment, the maximum number of malicious nodes used in these experiments does not surpass the limit of flow entries the switch can have, and therefore the sensitivity and specificity are not affected by it.

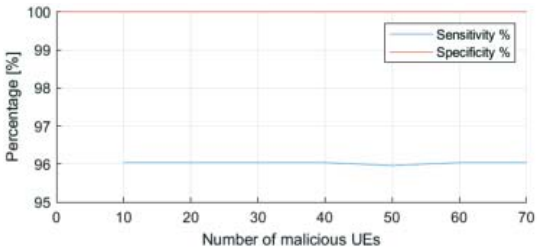

Fig. 3. Sensitivity and specificity for different number of malicious UEs.

B. Performance when varying the number of benign nodes

To evaluate the impact of the relationship between legitimate traffic and malicious traffic on the detection application, an experiment is repeated eight times where the number of benign UE nodes increases by 10 nodes per repetition in the range of 0 to 70 nodes. The results presented in Fig. 4 indicate a continuous decline in the detection of the 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 to generate legitimate traffic. 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.

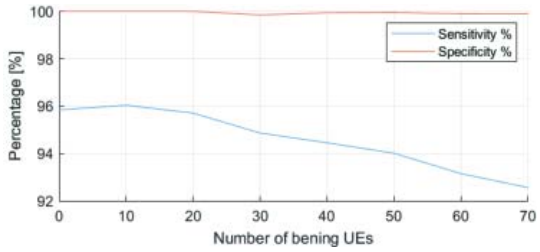

Fig. 4. Sensitivity and specificity for different number of benign UEs.

C. Performance when varying the flow entry timeout

The SDN DDoS Defense application relies on flow entries to mitigate attack traffic from nodes marked as malicious, and therefore the impact of the idle timeout of the flow entries should be evaluated as the timeout dictates how long a flow entry remains active. The results of ten simulations where the flow entry idle timeout is set to 1, 2, 4, 6, 8, 10, 20, 30, 40, and 50s respectively is presented as a graph in Fig. 5.

When the flow entry timeout is set to 2s or lower, the results show a sensitivity of ∼64%, while the sensitivity value is ∼96% for timeout values higher than 4s. The results can be explained by the frequency of packet transmissions by the malicious nodes being set to every 2s in the simulation. The waiting time of 2s before a new wave of SYN packets allows the flow entry timeout to run out and causes it to be deleted from the flow tables

of the OpenFlow enabled switch, causing the malicious traffic to reach the server again. As each malicious node transmits ten SYN packets in each wave, the first four malicious packets of the wave are judged as legitimate by the switch as the detection threshold is not surpassed yet. The fifth packet surpasses the detection threshold and causes itself and the remaining five packets to be marked as malicious and to be dropped at the switch. The performance in regards to specificity decreases from ∼100% at flow entry timeout values of 10s and lower, to ∼98% at timeout values of 20s and higher.

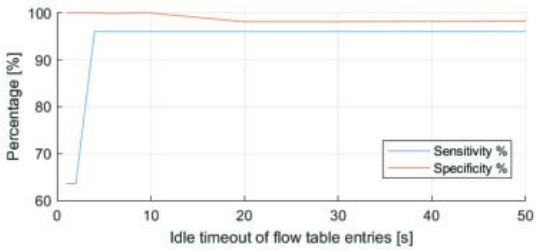

Fig. 5. Sensitivity and specificity for different values of the idle timeouts of the flow entries.

D. Performance when varying the detection threshold

The impact of the detection threshold value is evaluated by conducting an experiment that is repeated seven times with the maximum packet rate threshold starting at 0.1 pps, then from 0.5 pps to 3 pps, where the parameter is increased by 0.5 pps for each repetition. Fig. 6 shows a linear decrease of sensitivity as the threshold increases. Naturally, the number of false negatives increases as the threshold is lowered because more packets are required to surpass the threshold, allowing a larger number of malicious packets to reach the server before the attack node is detected and its traffic is dropped at the OpenFlow enabled switch. With a maximum packet rate threshold of 0.1 pps, the specificity is at ∼94%, but it stays at 100% from 0.5 pps and higher. A low threshold value makes the SDN controller to mark mistakenly legitimate nodes as malicious with a spike SYN message rate higher than the threshold value.

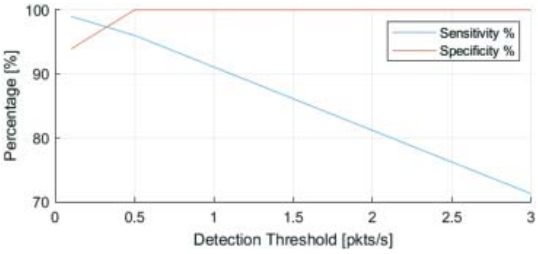

Fig. 6. Sensitivity and specificity for different packet rate thresholds for detection.

E. Performance when varying the malicious packet rate

The last two experiments are repeated four times with the attack rate per wave ranging from 5 packets to 20 packets, with an increase of 5 packets per repetition. The first experiment has a detection threshold value set to 5 packets, and the second experiment has the detection threshold value set to 30 packets.

The results of the first experiment, presented in Fig. 7, show an increase in the specificity of the detection method as the packet rate per wave increases. A detection threshold value of 5 packets gives a maximum packet rate threshold of 0.5 pps, making the first four SYN packets of the attack to be registered as false negatives, while the following packets are registered as true positives. When the attack rate per wave is increased, the number of true positives increases while the number of false negatives remains the same. The sensitivity given with Eq. (1) shows that if the number of true positives increases, then the sensitivity increases as well. Note that the sensitivity does not reach 100% as long as the maximum packet rate threshold is over 1 pps.

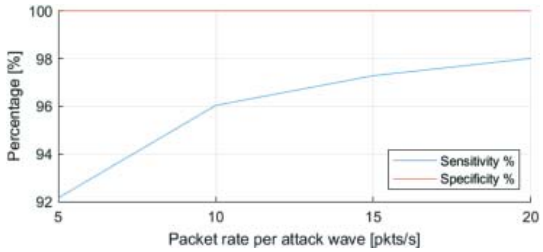

Fig. 7. Sensitivity and specificity with a packet rate threshold of 5pps for detection and varying the malicious packet rate.

The results of the second experiment, presented in Fig. 8, reaffirm the results of the first experiment by showing a similar growth in the sensitivity as the packet rate per attack wave increases.

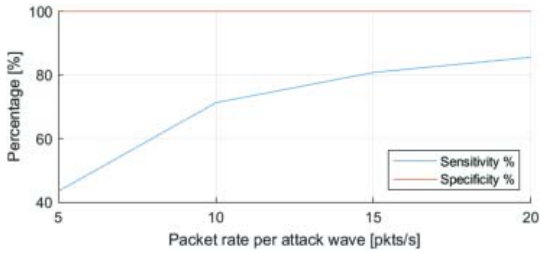

Fig. 8. Sensitivity and specificity with a packet rate threshold of 30pps for detection and varying the malicious packet rate.

V. Conclusion

In this article we presented a novel integration of OMNeT++ extention libraries, namely INET, SimuLTE and OpenFlow OMNeT++ Suite, for implementing an SDN DDoS Intrusion Detection application in a 5G-like scenario. The presented results showed that the SDN

DDoS defense controller application mitigates SYN flood attacks effectively and provided a proof of concept for SDN as a security component. The implemented simulation framework appears promising in order to further test and compare different SDN-based algorithms and topologies for detecting and mitigating malicious attacks in 5G-like scenarios.

References

[1] L. Yang, R. Dantu, T. Anderson, and R. Gopal, "Forwarding and control element separation (ForCES) framework," Tech. Rep., 2004.

[2] G. ETSI, "001:" Network Functions Virtualisation (NFV)," Architectural framework, 2013.

[3] S. Jain et al., "B4: Experience with a globally-deployed software defined WAN," in ACM SIGCOMM Computer Communication Review, vol. 43, no. 4. ACM, 2013, pp. 3–14.

[4] V. Varadharajan and U. Tupakula, "Security as a Service Model for Cloud Environment," IEEE Transactions on Network and Service Management, vol. 11, no. 1, pp. 60–75, March 2014.

[5] 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 Proc. of Int. Conf. on Availability, Reliability and Security, ser. ARES 2018. ACM, 2018, pp. 49:1–49:10.

[6] I. Ahmad, T. Kumar, M. Liyanage, J. Okwuibe, M. Ylianttila, and A. Gurtov, "Overview of 5G Security Challenges and Solutions," IEEE Communications Standards Magazine, vol. 2, no. 1, pp. 36–43, 2018.

[7] Y. Khettab, M. Bagaa, D. L. C. Dutra, T. Taleb, and N. Toumi, "Virtual security as a service for 5G verticals," in IEEE Wireless Comm. and Networking Conf. (WCNC), April 2018, pp. 1–6.

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

[9] 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. ICST, 2008, p. 60.

[10] 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). IEEE, 2014, pp. 59–70.

[11] 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. Springer, 2016, pp. 3–18.

[12] W. M. Eddy, "Defenses against tcp syn flooding attacks," The Internet Protocol Journal, vol. 9, no. 4, pp. 2–16, 2006.

[13] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on tcp," in Proc. IEEE Symp. on Security and Privacy, May 1997, pp. 208–223.

[14] D. M. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation, school of informatics and engineering, flinders university, adelaide, australia," TR SIE-07-001, Journal of Machine Learning Tech. 2: 1 pp. 37-63, Tech. Rep., 2007.