# A Method of Network Workload Generation for Evaluation of Intrusion Detection Systems in SDN Environment

Damian Jankowski
Institute of Telecommunication, Faculty of Electronics
Military University of Technology
Warsaw, Poland
Damian.Jankowski@wat.edu.pl

Marek Amanowicz
Institute of Telecommunication, Faculty of Electronics
Military University of Technology
Warsaw, Poland
Marek.Amanowicz@wat.edu.pl

***Abstract*—Software defined networks create new opportunities for an implementation of the intrusion detection and protection methods. Therefore, special data collections called datasets are necessary for the development, testing and evaluation of such mechanisms. For the SDN environment, there are no prepared datasets that could be used directly to develop IDS methods. These sets contain tuples with features, which represent the activities performed in the IT system. In the presented approach, normal and malicious traffic are generated in the SDN virtual environment. The workload for the dataset is generated in a random manner. The proposed method enables the generation of flows, which can be used for the evaluation of various SDN-based intrusion detection methods.**

***Keywords— IDS dataset; SDN simulation; attack generation; virtual testbed; Mininet; Opendaylight***

## I. INTRODUCTION

Software defined networks (SDN) is an approach that allows an IT Engineer to implement and control network functionality through software solutions. It is possible to deploy new services and applications in a virtual environment. The basic idea of the SDN technology is the separation of the data plane from the control plane. In contrary to classical network solutions, network devices are controlled by SDN controllers in a centralised manner. It enables for the configuration and programming from a central interface to match the service requirements in a distributed network. Moreover, centralised logic and management allows for comprehensive monitoring of the network. [1]

Software-defined network creates new opportunities for the implementation of intrusion detection systems (IDS) and intrusion protection systems (IPS). Moreover, it allows for integration of threat detection methods with the SDN environment. Due to the openness of platforms supporting the SDN technologies, it is possible to use existing mechanisms and protocols, or to implement new approaches. [2] An important factor associated with intrusion detection is the possibility of aggregating statistics from network devices and forwarding them to the controller. The collected parameters can be used as features for intrusion detection mechanisms. SDN architecture creates new opportunities to increase the level of security, especially in the context of the detection of unauthorised activities. The intrusion detection and protection technologies, integrated with the SDN environment and based on native SDN functionality, can provide an additional element of security, besides the classic IDS and IPS. [3]

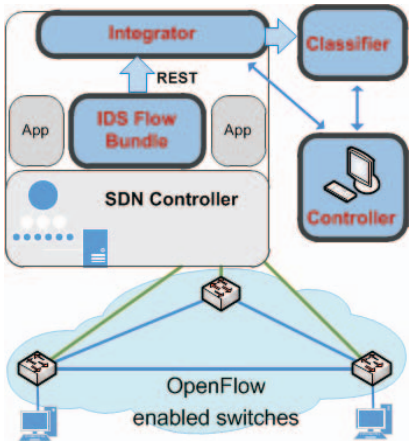


Fig. 1. SDN-based intrusion detection architecture

We propose a concept of using the native features of the SDN technology for monitoring and detection of unauthorised activities in SDN data flows. The system consists of 4 main modules, as presented in Fig. 1.The Flow Bundle Module is responsible for extracting traffic statistics from data flows. The Integrator is responsible for the collection and processing of traffic statistics and for traffic feature generation. The outcomes of this process are passed to the machine learning-based Classifier, which is responsible for the detection of unauthorised activities in the data plane. The controller supervises operations performed by the modules and processes the results of traffic classification for the visualisation. In order to conduct tests and a verification of our concept, it is necessary to obtain a set of network workload test data. Sample datasets for classical IDS are KDD99 Cup , NSL-KDD [4] and Kyoto Dataset [5]. Such sets contain tuples labelled with features, which represent activities performed in IT system. These sets can be used in the process of developing and

researching machine learning based IDS. These datasets are especially useful for the processes of learning, testing and validation. [6] Thanks to the common dataset, it is possible to compare the performance of different methods [7]. Unfortunately, there are no prepared datasets that could be used directly to evaluate the IDS on SDN controller platform, which are based on the classification of flow parameters. For generation of such datasets, various approaches can be applied.

- Conversion of existing datasets features to the corresponding SDN features [8].
- Collection of benign and malicious traffic from the real SDN environment. Features of malicious traffic can be collected by honeypots or IDS in the SDN network.
- Generation of synthetic traffic in the SDN data plane by the simulation or emulation environment.
- Combination of the above approaches, for instance, benign traffic can be collected from a real network and malicious activities can be adopted from an existing dataset, or generated in the emulation process [9].

In this paper, we presented the method that covers the method of malicious and normal traffic generation in the emulation environment. This approach ensures high flexibility in composition with a of variety network services. Moreover, new classes of traffic can be easily added. The types of normal and unauthorised activity that will be created can be defined and adjusted. Contrary to capturing the real traffic, it is possible to determine the course of the simulation. The drawback of the proposed concept is the difficulty in the simulation of a complex network environment. There are solutions and tools that generate application traffic with suitable statistical parameters. These solutions provide mock servers or clients among which is the generated traffic. [10] However, for these tools, it is difficult to generate malicious traffic. In order to ensure a realistic course of the attack, malicious actions should be carried out on the real and running services. Therefore, mock clients should initiate traffic to the real services, in accordance with the required statistical parameters. Another solution is the integration of mock client and servers for normal traffic, with malicious client and servers for attack.

This paper is organised as follows. Chapter II describes the virtual testbed used to generate traffic. The third chapter introduces the principles and methods of generating defined traffic classes. Chapter IV presents the examples of results obtained with the data set. Finally, Chapter V presents our conclusions.

## II. Virtual Testbed

### A. Architecture

The presented virtual environment enables generation of the SDN network traffic. The Mininet platform is used as a network emulator. The server side is emulated by Metasploitable 2 virtual machines with the Ubuntu operating system. Vulnerabilities in services and operating systems, default passwords and misconfigurations are intentionally left on the server environment. Simultaneously, the clients generate requests to the server. At the same time, the malicious host performs unauthorised activities directed to servers by using attack tools. The course of the emulation is automated by Python scripts. Generated traffic is probed through the measurement module. The servers reside on separate virtual machines and clients are virtualised at the level of Mininet OS. At this stage of the implementation, the SDN network has a star topology with a single switch. The architecture of the testbed is shown in the Figure 1.

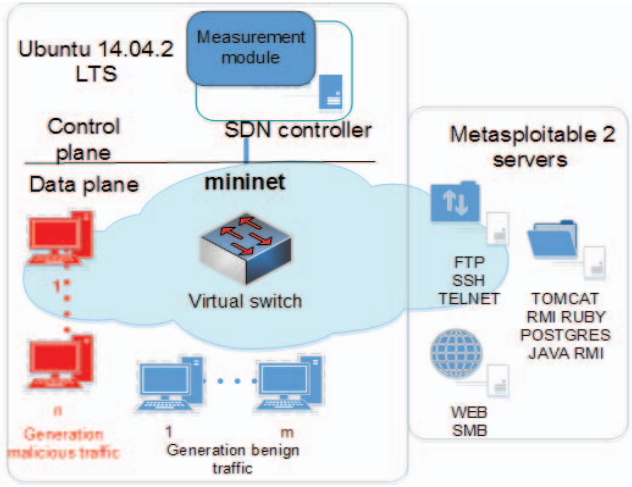


Fig. 2. Architecture of virtual environment

### B. SDN Emulator

Mininet is developed in Python and C. In the connection with the Opendaylight controller functionality, it is possible to model the SDN environment with various topologies. Network configuration can be done by the Linux command line console or by using scripts in Python. Mininet uses process-based virtualisation to run up to 4096 hosts and switches on a single OS kernel. Linux kernel on version 2.2.26 and above supports network namespaces - a kind of lightweight virtualisation feature. It provides individual processes with separate network interfaces, routing tables, and ARP tables. Mininet allows for the creation of a kernel or user-space OpenFlow switches, controllers to control the switches, and hosts to communicate over the simulated network. Mininet connects switches and hosts using virtual Ethernet pairs. [11] The Mininet platform is characterised by a favourable features, facilitate the design of the research testbed.

- Flexibility – additional functionality and new features can be set by programming languages, with additional modules. The external packages or tool can be also used.
- Concurrency – the processes and threads in simulation can run simultaneously. For example, the hosts generate requests independently.
- Scalability – the environment is able to simulate a small autonomic network as well as a large topology with network devices on the virtual operating system.
- Realistic - the Mininet provide real time behaviour with a high degree of confidence, so applications and

protocols stacks should be usable without any code modification. Moreover, there is a possibility to connect a virtual emulator with real SDN networks.

- Applicability- implementations conducted in prototypes should also be usable in real networks based on hardware without any changes in source codes. [12]

### C. Method of Measurement

The presented approach assumes that the features of network workload are measured and generated by the native mechanisms of the SDN technology. The module works on the SDN Opendaylight controller as an OSGi bundle. [12] The flows are matched by the following parameters:

- destination IP address,
- source IP address,
- destination port of transport layer,
- source port of TCP/UDP layer,
- protocols – ARP, IP, TCP, UDP or unknown.

For each flow, an idle timeout parameter is set. It defines the period after the entries are deleted from the flow table. In addition, an identification number is assigned to the flow. The proposed way of matching traffic, distinguishes flows in the context of different port numbers and IP address. However, the proposed method may have a potential performance disadvantages because of the fine granulated flows. For instance, the specified network activity may consist of multiple flows. The number of these flows can be very large in the network environment with intensive traffic and various services. Consequently, SDN controllers have to process large amounts of flows. It is possible to measure the parameters and define relationships between flows at the transport layer connection. The flows are refreshed within a specified period of time. An important component of this mechanism is a REST client. This module communicates with the Opendaylight server. The following parameters permit to change the sampling resolution:

- time period between queries (in the REST client),
- time period between refresh of array status and statistical parameters defined configuration flow controller SDN.

For the collected flow, a set of basic parameters is determined in the REST module. A determination of additional features can be performed during tests or after the measurement process on the basis of tuples with the basic features. All collected features of flows are stored as rows in the external file or database.

The presented virtual environment allows to collect features, which can be used as an input vector for the IDS mechanisms. For each collected flow, a set of parameters is determined. This approach is modelled on the flow based IDS and it is adopted in order to take advantages of the SDN capabilities. The dataset is represented as a set of tuples $<X, d>$, where $X$ is the feature vector and d is the label of classes.

In the presented approach, there are two ways of determining from OpenFlow statistics and fields:

- basic features,
- additional features.

The basic features are native OpenFlow attributes and statistics, for instance: total bytes or packets count, duration time, IP addresses, ports numbers and transportation layer protocols. On the basis of such parameters, it is possible to generate additional metrics. The additional features reflect a relationship between flows or flow parameters:

- flows from one host with different port,
- single flows count to host,
- connections count from different host on the same port,
- increase the number of flows with different ports.

Moreover, there are two ways to measure the features.

- *Time-varying features* - the changes of flow statistics over the time. The values of flows parameters are sampled with specified period t, $X_i$ $(x_1, x_2, \ldots, x_n)$ at $i$ time of sampling. For example, there are defined multiple samples of the measured *byte count*, for each flow. The samples reflect changes in the number of bytes as a function of time
- *Statistical features* - values of the features is determined at the end of flows. The vector of features $X_s$ $(x_1, x_2, \ldots, x_n)$ includes the maximum or final value of features from all $X_i$ vectors. These individual values can be probed when the flow is deleted after idle timeout. For instance, there is one assigned *byte count* parameter for each flow. This reflects the total number of bytes transmitted in the flow.

## III. Traffic Generation

### A. Assumptions

The dataset should not exhibit any unintended properties, which could have an impact on the intrusion detection process. This should provide a clearer picture of the real effects for normal traffic and attacks over the network. For this reason, it is necessary for network activities to look and behave as realistically as possible. This should include both normal and anomalous traffic. [14] [15] In the process of implementing scripts for the test environment, the following assumptions are established:

- Normal and anomaly traffic is generated on separated hosts.
- Normal and anomaly traffic is simultaneously.
- Time between traffic initiations is undetermined due to selected probability distributions.
- Client hosts initiate connections to the servers.
- Client hosts do not communicate with each other.

TABLE I. PARAMETERS OF GENERATED TRAFFIC CLASSES

| Classes of traffic | Description of activities | Tools for traffic generation | Parameters of distribution | Subclass count | Statistics of the basic features μ - mean, σ - deviation | | | Number of instances |
|---|---|---|---|---|---|---|---|---|
| | | | | | Packet count | Byte count | Flow duration [s] | |
| Normal | Traffic between clients and servers | Clients and servers of following services FTP, SSH, SMB, Apache, Web, Tomcat, RMI Ruby, Java RMI, Postgres, Telnet | k = 1<br>λ = 3 | 20 | μ = 74,16<br>σ = 1025,32 | μ = 128046<br>σ = 2150267 | μ = 6,50<br>σ = 28,81 | 26916 |
| Probe | Port probe, vulnerability scan, version scan | Metasploit, Nmap | μ = 4000<br>σ = 200 | 7 | μ = 0,07<br>σ = 0,50 | μ = 6,06<br>σ = 61,55 | μ = 2,30<br>σ = 1,60 | 16946 |
| R2L | Password crack | Metasploit, Hydra | μ = 600<br>σ = 200 | 6 | μ = 6,56<br>σ = 5,09 | μ = 603,80<br>σ = 702,72 | μ = 7,12<br>σ = 3,82 | 2331 |
| DoS | Denial of service attacks | Metasploit, Hping3, Nping | μ =1200<br>σ= 200 | 6 | μ = 18,34<br>σ = 89,85 | μ = 21127<br>σ = 171170 | μ = 4,29<br>σ = 3,21 | 1002 |
| U2R | Remote exploits, backdoors | Metasploit | μ = 120<br>σ = 40 | 10 | μ = 4,80<br>σ = 15,538 | μ = 262,40<br>σ = 651,04 | μ = 1,65<br>σ = 0,50 | 358 |

In the proposed approach the server side provide services which are presented in Tab I. It can provide variety of network application traffic. The traffic is randomly initiated by client hosts. On each client host, the infinite loop is established. Firstly, the time between requests is determined by stochastic distribution. After that, requests to the server are generated by Python instructions. Five classes of traffic are generated in presented testbed: normal, DoS, probe, U2R, R2L. For the normal traffic, the number of host requests per time unit is defined by Poisson distribution:

$$f(k,\lambda)=\frac{\lambda^k e^{-\lambda}}{k!} \tag{1}$$

where: λ – variance, k = 0, 1, 2, … - support.
The unauthorised activities are generated with time intervals, which are determined by Gaussian distribution:

$$f(x\mid\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{2}$$

where: μ – mean, σ – standard deviation.
Stochastic parameters of distributions are defined before the simulation process (see Tab I). Each class of traffic has subset of activities, randomly selected by uniform distribution (see Tab. I). In normal traffic, this subset defines the services, which are used for generation. In addition to this, the parameters of the connection to the specific subclass are determined by a uniform distribution. For instance, the size of file download from server or the number of SQL queries

.

### B. *Malicious Traffic Generation*

In order to maximise the realism of the attacks, malicious activity are conducted using special tools (see Tab I), which ensure an adequate level of realism of the generated attacks. Each class of malicious traffic has subclasses, which define the detailed course of attacks, types of attack tools or exploits. The DoS class represents the Denial of Service attacks. In the virtual testbed, these activities are performed against the network or server resources. For instance, the malicious hosts perform a flooding attack on the SDN network by Nping tool with specific parameters. These events have an impact both on the SDN controller performance and the available data plane resources, and can cause delays in processes of matching flows. The probe class includes attacks that are intended to obtain information about the object of attack. These attacks involve performing a port, version, services or vulnerability scan. Such malicious activities give information to the intruders about the potential targets of the attacks. The scans are performed by Metasploit or Nmap tools, and indicate a potential attack surfaces. This kind of activity may be a preliminary phase of the main attack, i.e. DoS or buffer overflow.

The U2R class includes network activities related with the back doors and remote exploitation attacks. The attacks are performed by Metasploit Framework scripts and commands. These malicious activities are carried out against the vulnerable services which are used in normal traffic. Therefore, these attacks are characterised by a high degree of similarity to the short duration normal traffic. The course of the U2R attacks is shown in Fig 2. Firstly, the malicious requests prepared by Metasploit are sent to the vulnerable service. The payload contains the exploit and shellcode for the specified service. After the exploiting operation, the malicious host gains access to the shell with root privileges. At next step, the malicious host establish the connection to the exploited service, with the reverse shell and execute a few Linux commands. The type of exploit and detailed course of the attack may vary for individual services. For instance, before exploit steps, the user login may take place. Each stage of the attack is reflected in SDN fine granulated flows

The R2L class includes the credentials guessing and the unauthorised access to IT accounts. The password guessing is conducted in the form of dictionary attack. The potential

passwords and logins are stored in external file. The malicious hosts try to authorise with parameters from lists of credentials. Sequential requests are sent to the service. When the authorisation succeeds, the corresponding credentials are stored. These activities generate moderate number of flows.

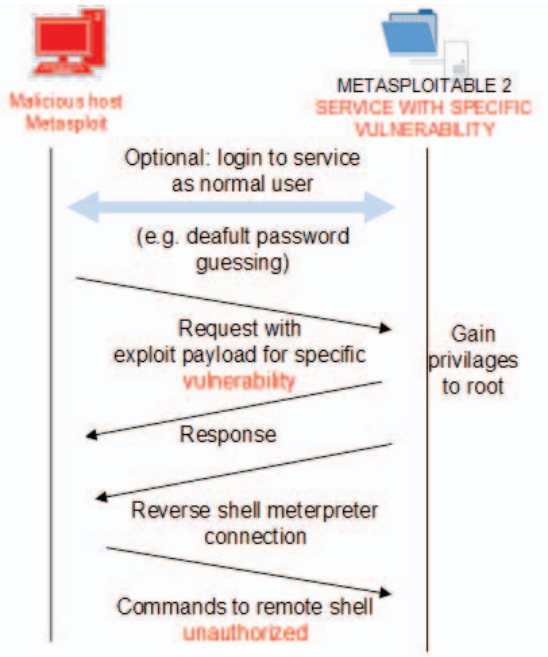


Fig. 3. Stages of generic U2R attack performed in testbed

### C. *Traffic Labelling*

In the presented approach, priori to classification it is necessary to assign labels to suitable network activities. The dataset with the prepared labels eliminates the process of manual labelling. [14] Labelling can be carried out on the basis of IP addresses, with assumption that classes of traffic are generated from specific hosts. Particular hosts generate only one class of network activities. For instance, the collected dataset rows can be label as follow:

- 10.0.0.1 – 10.0.0.20 – normal traffic
- 10.0.0.21-10.0.0.25 – DoS attack
- 10.0.0.26 – 10.0.0.30 – probe attack
- 10.0.0.31 – 10.0.0.35 – U2R attack
- 10.0.0.36 – 10.0.0.40 - R2L attack

Unfortunately, this creates a high correlation between class and IP address. Therefore, to avoid this inconvenience, the part IP addresses of malicious hosts can be changed to IP of benign hosts, on stochastic or determined manner. After this process it is possible generation of additional features taking into account changes in IP addresses. The artificial post-capture trace insertion will negatively affect the data set and introduce possible inconsistencies in the final dataset [14]. However, this change of IP address will not have a negative impact on the dataset.

## IV. The Examples of The Network Worload Generation

The exemplary parameters for 24-hour emulation time are shown in the Tab I. The generated dataset has unbalanced classes. The disproportion in the number of class instances is the result of measurement method. The least likely generated unauthorised class is the probe, however due to the fine granulated flows programming, this is the most numerous class of the unauthorised actions. In contrast to that, the relatively often generated U2R class is the least numerous. The attributes are generated from fine the granulated flows, and port scanning increases the number of short-duration flows on different ports. The issue of unbalanced classes is important, because some methods of classification can characterise low performance in the case of unbalanced dataset.

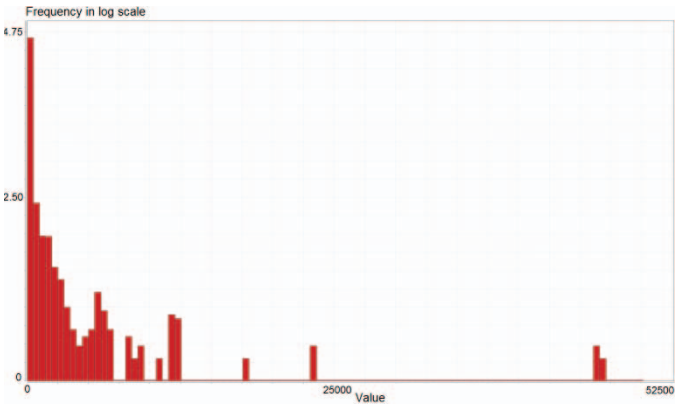


Fig. 4. Histogram of packet count in flows

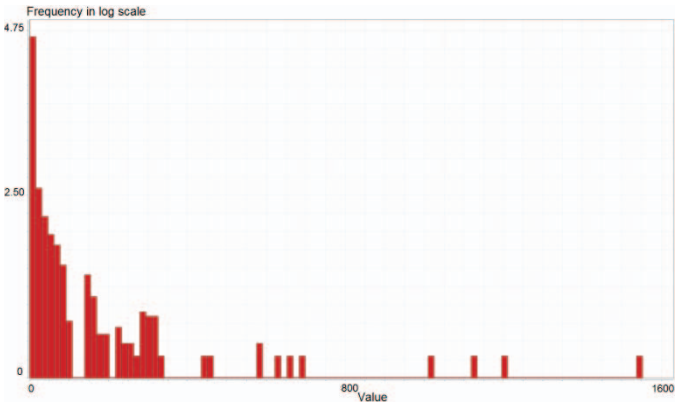


Fig. 5. Histogram of flows duration

Fig. 3 and 4 present the histograms of the basic features for the collected dataset. The frequency is in the logarithmic scale. The distribution histogram shows a strong dominance of the flow with a small count of packets and a short duration. This could be an effect of the generation method, where short time and small packet count flows are the most common. Moreover, fine granulated flow programming has influence on the shape of the presented histograms.

Fig. 5 and 6 show the scatter plots of basic and additional features. The features are normalised in the range (0 … 1). It needs to be highlighted that the scope of the *Flow duration* axis is from 0 to 0.05. The presented graphs show that the basic characteristics (*Flow duration* and *Destination port*) give worse discrimination characteristics. The variant with the additional feature (*connection to host count*) provides a better distinguishability of classes. The values of feature *Duration of flow* are strongly grouped in the range 0 ... 0.01. In Figure 6, the features show patterns over all of the range. The results indicate that additional features can provide better separation of classes than the basic features. What is more, the presented results show that the classes of the data set are not well separated. Consequently, the process of classification is not trivial for the intrusion detection mechanisms.

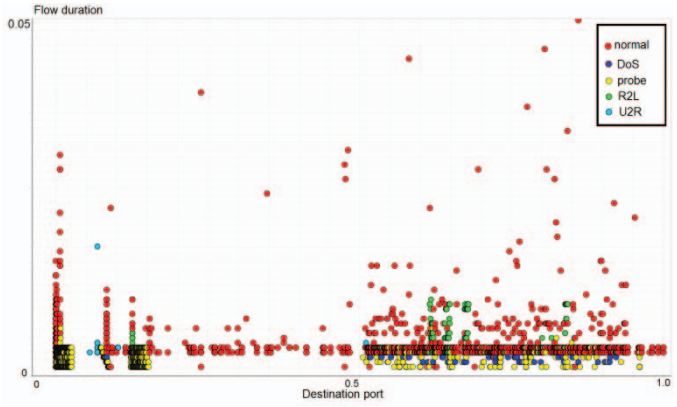


Fig. 6. Part of scatter plot of *Destination port* and *Flow duration* features

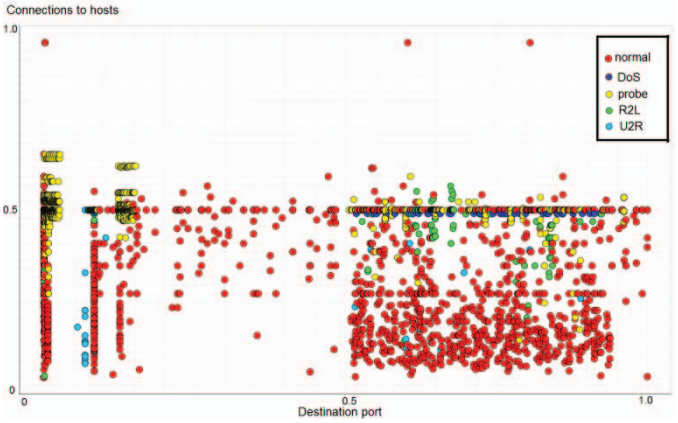


Fig. 7. Scatter plot of *Destination port* and *Connections to host* features

The time-varying features can be extracted by SDN native mechanisms. The waveforms in Figures 7 and 8 present changes of the flows features in the function of time. The presented graphs show significant differences between the classes of traffic. Contrary to normal traffic, the R2L attacks consist of series of peaks values. They represent attempts to login in a specific time stamp. Other properties are characterised by exemplary waveforms of U2R attacks. The values of probes and duration are smaller than equivalent parameters of other classes. The features of the waveforms can be used as an input vector for the classification methods. The distinctive features of time-varying parameters can be determined using transformations e.g.: Fourier [16], Wavelet [17] or Hilbert-Huang transformations [18].

The features can be determined on the basis of the first packet transmitted to the controller, however, at this stage of research, this mechanism is not implemented. The approach of analysing the header and contents of the first package may have a positive influence on the detection performance of attacks on specific groups. However, it is necessary to improve the performance of methods without packets inspection because malicious payloads (i.e. exploits and shellcode) in packets can be obfuscated. Therefore, with the development of the presented method, it will be evaluated which feature is more significant.

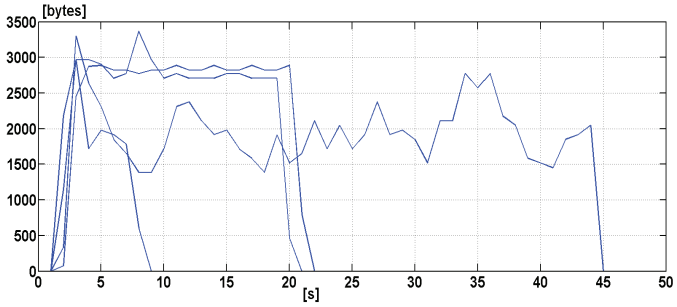


Fig. 8. Normal traffic waveforms *(bytes per seconds)*

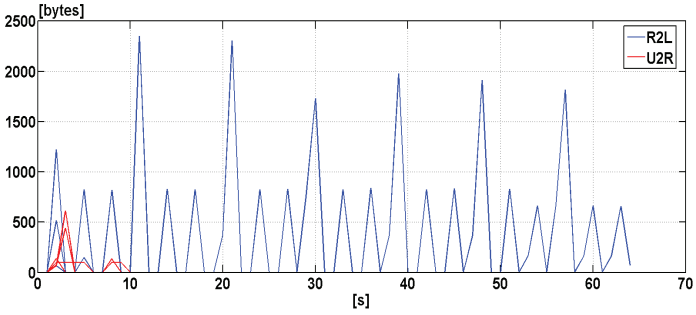


Fig. 9. U2R and R2L attack waveforms *(bytes per seconds)*

## V. Conclusions and Future Works

In the paper, we presented the concept of dataset generation in the SDN data plane. The proposed method includes basic ways for the synthesis of application layer traffic and can be expanded, depending on the specific needs. The statistical properties of the generated data set require an implementation of advanced classification techniques for the identification of an individual traffic class. This means that the presented method of the SDN dataset generation can be used for the evaluation of selected machine learning methods and their applicability for SDN-based intrusion detection. In future works, we plan to develop more a advanced and adequate stochastic model of both normal and malicious traffic generation. This would require a more careful examination of the behaviour of intruders in the SDN environment, for instance, by using honeypots. Another important issue refers to the implementation of a wider variety, especially malicious network activities targeted at the SDN control plane. This

would require an elaboration of specific flow generators, combining both normal and hostile traffic.

## REFERENCES


[1] F.M.V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, S. Uhlig, Software-Defined Networking: A Comprehensive Survey, Proceedings of the IEEE, Proceedings of the IEEE, 2015

[2] Sandra Scott-Hayward, Sriram Natarajan, Sakir Sezer Member, A Survey of Security in Software Defined Networks, IEEE Communication Surveys & Tutorials, 2015

[3] Changhoon Yoon, Taejune Park,Seungsoo Lee,Heedo Kang,Seungwon Shin,Zonghua Zhang, Enabling security functions with SDN: A feasibility study, Computer Networks 85, 2015, pp. 19–35

[4] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani, A Detailed Analysis of the KDD CUP 99 Data Set,

[5] Jungsuk Song, Hiroki Takakura, Yasuo Okabe, Masashi Eto, Daisuke Inoue, Koji Nakao, Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation, EuroSys 2011, pp. 29-36

[6] Sumouli Choudhury, Anirban Bhowal, Comparative Analysis of Machine Learning Algorithms along with Classifiers for Network Intrusion Detection, ICSTM, 2015, pp.89-95

[7] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, Wei-Yang Lin, Intrusion detection by machine learning: A review, Expert Systems with Applications, 2009

[8] R.Sathya, R.Thangarajan, Efficient Anomaly Detection And Mitigation In Software Defined Networking Environment, ICECS, 2015

[9] K. Giotis, C. Argyropoulos, G. Androulidakis , D. Kalogeras, V. Maglaris, Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments , Computer Networks 62 (2014) 122–136

[10] A. Botta, A. Dainotti, A. Pescapè, A tool for the generation of realistic network workload for emerging networking scenarios Computer Networks (Elsevier), 2012, Volume 56, Issue 15, pp 3531-3547

[11] Mininet, An Instant Virtual Network on your Laptop (or other PC), [Online]. Available: http://minimet.org

[12] OpenDaylight Platform [Online]. Available: https://www.opendaylight.org/

[13] Faris Keti, Shavan Askar, Emulation of Software Defined Networks Using Mininet in Different Simulation Environments, 6th International Conference on Intelligent Systems, Modelling and Simulation, 2015

[14] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, Ali A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Computers & Security 31, 2012, pp. 357 - 374

[15] VanLoi Cao, Van Thuy Hoang, Quang Uy Nguyen, A Scheme for Building A Dataset for Intrusion Detection Systems, Third World Congress on Information and Communication Technologies, 2013

[16] Enkhbold Chimetseren , Keisuke Iwai , Hidema Tanaka and Takakazu Kurokawa, A Study of IDS using Discrete Fourier Transformn, International Conference on Advanced Technologies for Communiations, 2014

[17] Chin-Tser Huang, Sachin Thareja, Yong-June Shin, Wavelet-based Real Time Detection of Network Traffic Anomalies, IEEE, 2006

[18] Weihong Wan, Kai Pei, Xiaogang Jin, Using Hilbert-Huang Transform to Characterize Intrusions in Computer Networks, Third International Conference on Natural Computation, 2007