Kalpa Publications in Computing

Volume 14, 2023, Pages 96–99

Proceedings of V XoveTIC Conference. XoveTIC 2022

kalpa Computing

Annotated Dataset for Anomaly Detection in a SDN infraestructure

Anxo Otero Dans and Víctor Manuel Carneiro Díaz

CITIC - Department of Computer Science and Information Technologies , University of A Coruña, 15071 A Coruña, Spain
{anxo.oterod, victor.carneiro}@udc.es

Abstract

Software-Defined Network (SDN) is an emerging architecture which objective is to reduce the limitations of traditional IP networks by decoupling the network tasks performed on each device in certain planes by controlling and managing the whole network from a centralized location. However, this centralization also introduces new inefficiencies and vulnerabilities, such as those related to southbound and northbound controller interfaces, which often negatively affect security. In the past years, Machine Learning (ML) techniques have been implemented in SDN architectures to protect networks and solve security problems but sometimes it is difficult to obtain the right characteristics in real time. In this paper, we introduce a flow-based anomaly detection system in which the controller itself is in charge of receiving, analyzing and classifying the traffic by extracting a group of flow features.

1 Introduction

In traditional networks there are two main planes; the control plane, which is responsible for filling the routing table, drawing the network topology and thus enabling the data plane functions; the second is known as the data plane, which handles the functions of forwarding packets from one interface to another relying on the control plane logic.

Software-defined networking (SDN) has brought a new paradigm in which the control plane and the data plane are decoupled allowing the centralization of network logic and topology through a device called controller which is responsible for taking all decisions in the network while the rest of the network devices become simple packet forwarders whose behaviour can be modified through a new standard protocol called OpenFlow (OF). This centralization provides greater cost efficiency and easier maintenance of the devices that make up the network topology.

Despite the advantages provided, SDNs are susceptible to new types of threats and for this reason, there has been plenty of studies related with SDN security over the past years. Early work on SDN security [1] focused solely on the implementation of access control lists within the SDN controller missing a lot of important aspects. Another solutions are the SDN-based firewall like [4] [9] [2] which focuses on creating a centralized firewall leveraging the controller that enables programmability and keeps track of the flow path space, however, these approaches

do not take into account a large number of flow characteristics that can influence anomaly detection.

In terms of security, conventional networks usually rely on firewalls and intrusion detection systems (IDS) which sometimes may lead to a harder maintenance of the network security. In order to create an effiecient anomaly detection system it is very important to collect information and extract features, and as mentioned above, SDN has great capabilities for information gathering due to the centralization of the network logic thanks to the controller which is able to obtain real-time information and monitor the network from a global perspective by making requests to the OpenFlow switches without the knowledge of the network hosts, and with this information we can classify traffic flows that can later be used to create classification models and detect anomalies in the network. While it is true that some of the current controllers on the market offer functionalities for collecting traffic information, this information is often insufficient and incomplete.

In this paper we introduce a flow-based anomaly detection system in which the network itself, throughout the controller is in charge of detecting anomalous traffic by receiving, analyzing and classifying traffic without a third-party software. Due to the possibility of self-development, the application is not only able to obtain the data requested to the OpenFlow switches but it is also able to obtain features taking into account the context and it is also able to obtain mathematical statistics such as mean, variance, maximum and minimum values of different features.

Therefore, the final objective is the use of different Machine Learning (ML) and Deep Learning (DL) techniques based on the collection of network flow features thanks to the created capture system for anomalous detection.

2 Traffic capture Process

In this section, we propose a system to collect, process and select flow features for traffic classification in SDN with the objective of creating a set of features classified into different types of traffic flows. To create this system, the used controller is ONOS (Open Network Operating System) which is an open source project that provides a web GUI and an API [7] for developing ONOS-based applications. To provide an example of use of this API, in [8] the creators build an application for ONOS controller which creates a simple firewall that blocks more than two pings in a row.

There are several data capture techniques for downstream feature extraction. In [3] they create a network topology in which some hosts generate anomalous traffic and perform known attacks. The hosts themselves are individually responsible for capturing the traffic with the Tcpdump tool and saving it in PCAP files for later processing with the CICFlowMeter tool [5]. However, this approach does not take advantage of the controller for the information gathering and does not make real time decisions.

2.1 Scenario

The fact of using OF switches such as Open vSwitch (OVS) allows the controller to obtain the information directly with a global view of the network and as previously mentioned, the controller is in charge of all the control logic and the one in charge of deciding which actions can be performed and which cannot.

These OF switches have a flow table with inputs and when a switch receives a new flow, it looks for an entry in the table that matches that flow and if it does not match, a PACKET_IN message with some information is sent to the controller. The ONOS controller is able to obtain

certain data from the packets, such as the source ip, destination ip, source port, destination port or protocol. However, these characteristics are very insufficient when developing a feature-based system for anomaly detection.

As mentioned above, ONOS allows the addition of extra modules through ONOS applications. By default, there are a number of ONOS applications that can be enabled or disabled via the GUI or the driver CLI. In the same way, ONOS allows the creation of custom applications through its Java API that provides great programmability for topology functionality.

Therefore, in the proposed scenario, the OF switches are in charge of receiving the packets from the hosts and sending this information directly to the controller which will be in charge of receiving all the information and processing it to extract the indicated features thanks to the created ONOS-based application to subsequently, through machine learning techniques, make decisions about the network topology in a global way taking into account the traffic generated and received by all the hosts.

2.2 Proposed architecture

The proposed architecture is based on [3] in which a series of virtual machines are deployed simulating different agents for different kind of traffic generation, but with certain differences such as the data capture module or the configurations of the different applications.

Four virtual machines are created in a VMware Workstation environment. The first one is an Ubuntu 16.04 LTS machine in which the ONOS driver will be installed through the open source platform Docker which provides an easy deployment of the SDN. This VM will be in charge of controlling the rest of the network through the controller and it will also contain the data capture and processing module which means that it will be the core of the system. The second VM will be a Kali Linux machine that will be in charge of simulating the different attacks against the other hosts in the network using the wide variety of offensive security tools that are pre-installed in the OS. The third VM will be a Metasploitable 2.0 machine which is a host vulnerable to all kinds of attacks and will serve to make effective most of the attacks performed by the Kali linux VM. Finally, the last VM will be a pre-packaged Ubuntu 16.04 machine containing Mininet [6], the OpenFlow binaries and the necessary tools to configure it. Mininet allows us to create a realistic virtual network, running real kernel, switch and application code on a single machine and this helps us create a greater variety of traffic simulation.

2.3 Implementation of traffic capture and feature extraction system

The application for the traffic capture and feature extraction was developed entirely in Java, as it is the default language for the ONOS API. This application works as follows:

The first step is to create an ONOS application and syncronize it with the controller, then it is important to tell this ONOS application that a packet processor is going to be created, this processor will be in charge of doing a first packet filtering, that is to say, it will separate TCP, UDP or ICMP traffic depending on the payload content found in the IP header of the packet.

It is important to analyze traffic in context and not in a single direction, which is why traffic is grouped into flows. Because of this, the next step is to obtain raw data from the filtered packets and then create a unique key for each detected flow. This means that, once a packet is received from the controller, the application creates a key and stores the flow in a key-value mapping for a set period of time, if the controller receives another packet corresponding to a flow stored within that period of time, this packet is added to the context of that flow.

Once the system is grouping these flows and classifying them roughly, the application is in charge of calling a function that obtains from these flows a great variety of characteris-

tics, specifically 75 characteristics. These flow characteristics are grouped into seven groups: Network-based attributes, Packet-based attributes, Interarrival Times attributes, Flow timers attributes, Flag-based attributes, Flow-based attributes and Subflow-based attributes.

Finally, with all these features, the controller is able to obtain a large amount of information from all network flows and put them in context in order to fine-tune the anomaly detection system using machine learning or deep learning techniques.

3 Conclusions and Future Work

SDN is a novelty for conventional networks and brings with it a lot of advantages, however, the technology is fairly new and this means that the documentation of most driver software is incomplete or its compatibility with other software is poor, which led to great difficulties in creating the traffic capture module for feature extraction. Despite the difficulties, the system created has been able to capture the traffic generated in the test scenario and has been efficient with flow feature extraction and traffic classification. This system is also transparent to potential attackers as the entire operation is performed from the controller. This system will serve to solve one of the problems related to the availability of datasets in the SDN environment by helping researchers to be able to create datasets in a simpler way taking into account almost any type of traffic.

As future work, the elaboration and capture of different types of traffic with anomalies that have not been contemplated in this work is proposed. In addition, in our case we propose the implementation of response code to these traffic anomalies detected thanks to the programmability of the ONOS controller.

References

[1] Martin Casado, Michael J Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. Ethane: Taking control of the enterprise. https://dl.acm.org/doi/10.1145/1282427.1282382, 2007.

[2] Vaibhav Hemant Dixit, Sukwha Kyung, Ziming Zhao, Adam Doupé, Yan Shoshitaishvili, and Gail-Joon Ahn. Challenges and preparedness of sdn-based firewalls. https://dl.acm.org/doi/proceedings/10.1145/3180465, 2018.

[3] Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D Jurcut. Insdn: A novel sdn intrusion dataset. https://ieeexplore.ieee.org/document/9187858, 2020.

[4] Hongxin Hu, Wonkyu Han, Gail-Joon Ahn, and Ziming Zhao. Flowguard: building robust firewalls for software-defined networks. https://dl.acm.org/doi/proceedings/10.1145/2620728, 2014.

[5] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of tor traffic using time based features. https://www.researchgate.net/publication/314521450_Characterization_of_Tor_Traffic_using_Time_based_Features, 2017.

[6] Mininet. Mininet vm. http://mininet.org/download/, 2022.

[7] ONOS Project. Onos java api (2.4.0). http://api.onosproject.org/2.4.0/apidocs/index.html, 2022.

[8] Nan Haymarn Oo and Aung Htein Maw. Firewall application for onos sdn controller. https://meral.edu.mm/records/4989, 2017.

[9] Phillip A Porras, Steven Cheung, Martin W Fong, Keith Skinner, and Vinod Yegneswaran. Securing the software defined network control layer. https://www.ndss-symposium.org/ndss2015/ndss-2015-programme/securing-software-defined-network-control-layer/, 2015.