Towards Prediction of Security Attacks on Software Defined Networks: A Big Data Analytic Approach

by

Emre Unal, B.S.

A Thesis

In

Computer Science

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCES

Approved

Rattikorn Hewett, Ph.D.
Chair of Committee

Susan Mengel, Ph.D.

Abdul Serwadda, Ph.D.

Mark Sheridan
Dean of the Graduate School

August, 2019

ACKNOWLEDGMENTS

I would first like to express my sincere gratitude to my thesis advisor, Department Chair, Dr. Rattikorn Hewett, of the Computer Science at Texas Tech University for the continuous support of my thesis study and research. Prof. Hewett's office was always open whenever I ran into a trouble spot or had a question about my research, writing or personal. She consistently allowed this paper to be my own work, but also steered me in the right the direction whenever she thought I needed it. I couldn't have imagined having a better advisor and mentor for my Master of Science study.

I would also like to thank and express my profound gratitude to my parents and to my sister for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

ABSTRACT

Cyber-physical systems (CPS) tightly integrate physical and computing processes by monitoring and control data interacting between them via underlying networks. Software Defined Network (SDN) Technology has increasingly become essential in many advanced computer networks, including those in modern CPS, to provide flexible and agile network development. Despite many benefits that SDN offers, malicious attacks that can eventually prevent network services are unavoidable. Among the most predominant attacks on SDN controller layer, Link Discovery Attack and ARP (Address Resolution Protocol) Spoofing Attack are fundamental in that they are the gateways of many other SDN threats and attacks. To defend these attacks, most existing techniques either rely on relatively complex data validation techniques or use thresholds that can be subjective and unable to detect more than one type of attacks at a time if one deciding factor is used. While Big data technology, particularly machine learning, has been widely used for intrusion/anomaly detection, little has been done in SDN. This paper explores how well this technology can be used to predict these SDN attacks. By employing typical machine learning algorithms on simulated data of routing in SDN when attacks occur, preliminary results, obtained from four machine learning models, show the average area under ROC curve of over 96% and 92% for sample size 50,970 (12 switches) and 60,000 (20 switches), respectively. Further experiments show near-linear scaling in training time for the best performing algorithm when sample size grows up to 100,000.

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

MOTIVATION

There has been enormous development of information technology and its applications in the past two decades. with the rise of the Internet usage in industry, business and societies that impact our everyday living and human life, building scalable, reliable and robust networks becomes one of the most important tasks for network development. In today's world, Cyber-physical systems (CPSs) are everywhere from infrastructures such as transportation and power grid systems to automated operations in industry to services in business organizations. CPSs are systems that tightly integrate physical and computing processes by monitoring and control data interacting between them via underlying networks. Sensed data from physical systems are used by cyber systems to compute and determine appropriate action or parameter controls of the physical system components that in turn generate resulting behavioral data as a feedback to the cyber systems. The scalability and complexity of CPS compound the challenges of traditional networks to meet a huge demand for rapid development of flexible and reliable networks. To overcome these challenges, Software Defined Network (SDN) technology has been proposed [10, 11, 13]. It has increasingly become essential in many advanced computer networks, including those underlying networks in modern CPS, to provide flexible and agile network development. By separating functions of control and data (i.e., control plane and data plane), one can program to specify SDN controller to direct switches to deliver network services easily. SDN provides benefits in network programmability, customizability, development flexibility, and lower processing expenses. These mechanisms help to improve the network productivity as a whole and also allow better network management. Despite valuable advantages, like other networking architectures, SDN is unavoidably susceptible to security threats and attacks [2, 5, 8, 17, 18].

There are many attacks (e.g., flow table saturation attack, control plane saturation attack, Topology poisoning attack, spoofing attack, denial-of-service attack,

eavesdropping attack) on network infrastructures today which not only compromises the availability of networks, hosts and services, but also the confidentiality and integrity of the network data. Many of these attacks are easy to launch (e.g., spoofing attack [1, 3, 12, 16]) and difficult to trace back to identify attackers [18]. Most importantly, some attacks can lead to DoS (Denial of Service) attack [4, 14, 20] causing devastating consequences of loss of services. Because SDN also includes some functions of traditional networking, many of SDN security attacks also occur in traditional network architectures [28].

This paper focuses on two attacks of SDN, namely Link Discovery Attack and ARP (Address Resolution Protocol) Spoofing Attack. In general, the former is SDN-specific and the latter is not (except when it occurs in the SDN-specific architectural components (e.g., links interacting with the SDN controller) In Link Discovery Attack, an attacker aims to deceive the SDN controller's perception by fabricating/ forging a new link in the network that does not exist [7, 9, 19]. By exposing the traffic that traverses over this malicious link to interception and manipulation, the attacker can obtain the network information and take control over the traffic [7]. On the other hand, in ARP Spoofing Attack [1, 3, 12, 16], an attacker aims to pretend to be static gateway of a network by spoofing the SPA (Sender Protocol Address) and SHA (Sender Hardware Address) in the ARP message to create an invalid address mapping [3]. Because ARP treats each request or reply independently, a host can readily accept information from ARP replies without a corresponding request. ARP does not have mechanisms to authenticate the sender, replies, or integrity of information [1]. Thus, an attacker can reply with a broadcast to the network with its MAC address as the physical address of the gateway. In this paper, we consider ARP spoofing on the communication links between SDN controller and switches. Both of these attacks are SDN-specific and serious attacks as they are the front door of many other network threats and attacks, e.g., man-in-the-middle and denial of service (DoS) attacks.

To defend these attacks, many techniques have been proposed [1, 3, 9, 12, 16, 19]. However, most either rely on relatively complex validation techniques (e.g., ARP's Network Address Translation [3]) or use thresholds (e.g., [1, 19]) that can be

subjective and unable to detect more than one type of attacks at a time. As Big data technology, particularly machine learning, has been widely used for intrusion/anomaly detection in many CPS, however, little has been done in SDN. Our research aims to explore how well Big data technology can be used to predict these SDN attacks. In particular, a set of typical machine learning algorithms [22] is applied to simulated data of SDN routing communication to predict the presence of each type of these attacks.

The rest of the paper is organized as follows. Chapter II describes related work and Chapter III presents overview of SDN and the two SDN attacks. Chapter IV discusses the prediction of SDN-specific attacks followed by Chapter V giving experiments and the results. Chapter VI concludes the paper and discusses future work [28].

CHAPTER II

RELATED WORK

Much research has studied SDN security threats at various architectural layers: data layer, controller layer, and application(s) layer [2, 5, 8, 17, 18]. Link discovery attacks are described in [6, 8, 17] along with other network topology poisoning. Hong et al. [6] proposed a security extension to the SDN controller to provide a real-time detection of the network topology misuse. Hu et al. [8] proposed a new architecture which is supporting packet data scan detection to provide security services. However, this approach requires additional implementation and enormous modifications in the network infrastructure. Smyth et al. [19] presents a technique to detect Link Fabrication Attack by observing if the packet traffic exceeds normal threshold. For example, Wang et al. [20] uses transfer time to detect fabricated links and man-in-the-middle attacks. Wang designed an efficient tree data structure to store matching rules for every incoming packet. Each level in the tree accounted to a specific bit in a forwarding rule. Wang's approach has limitations regarding to memory allocation since the depth or size of tree may get very large. While these approaches work reasonably well, its accuracy depends on the set threshold that tends to be subjective. Unlike these approaches, our machine learning approach does not assume a pre-determined threshold.

ARP Spoofing attacks have been studied in both traditional LAN networks and SDN. Several techniques [1, 3, 12, 16] have been proposed to prevent ARP spoofing (e.g., ARP authentication, compare with trusted database, configure ARP entries manually, etc. [1]). Alharbi et al. [3] presents a mechanism to prevent ARP spoofing by making sure that the potentially spoofed information is kept away from any hosts and thereby prevents ARP cache poisoning. For reply based attack, the mechanism will only accept ARP replies for which it has a corresponding request. Abdelsalam et al. [1] does port level ARP packet count monitoring to detect large volume of incoming malicious packets and stops them. Although this solution and his approach is fast, reliable and tested for different scenarios, it requires threshold to help make decisions. Ramachandran et al. [12] injects ARP request and TCP SYN packets to the

network on the purpose of investigating possible inconsistencies. None of the above techniques make use of behavioral data of SDN executions to build a predictive model of each kind of attack as our proposed approach.

While machine learning has been widely applied to anomaly detection and intrusion detection in traditional networks, at the time of this writing machine learning applications to SDN attacks are few or only mentioned with no empirical studies. This technique distinguishes certain network parameters as features and depending on the features' values, classifies the network's status as either under attack or under no attack [28].

CHAPTER III

BACKGROUND

This chapter describes an overview of SDN in Chapter I and the two SDN-specific attacks in Chapter II.

Software Defined Networking (SDN)

SDN [10, 11] is an emerging architecture that can be viewed in three different planes (or layers). SDN breaks the vertical assimilation by separating the control logic (control plane) from the underlying routers and switches (data plane) that forward the traffic. This feature enables the network control to have an ability to directly programmable and the underlying infrastructure to be virtualized for any applications and network services. Fundamental basic structure of this architecture can be seen in Fig. 1.

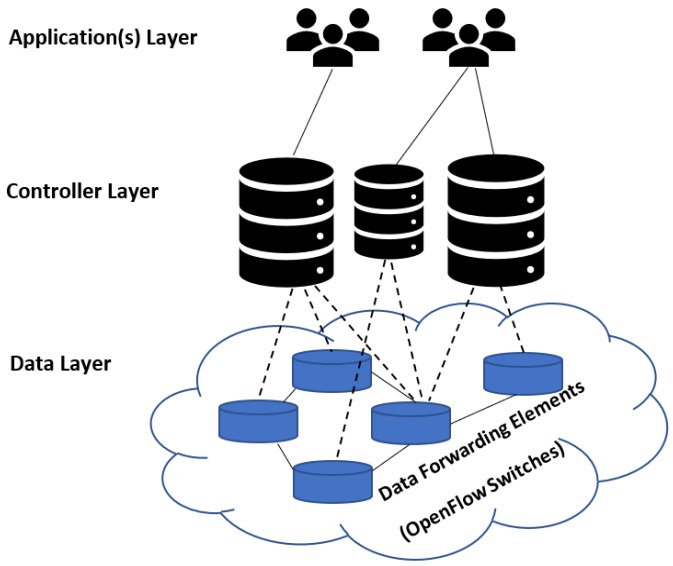

Figure 1. SDN Architecture.

The data layer consists of forwarding devices which are responsible for forwarding data, including physical and virtual switches. The switches offer the view of the programmable flow tables that define an action for each packet related to a

specific flow. The control logic which is moved to an external entity, called the SDN controller lies in the control plane of the architecture. The controller is a programmable software platform which has the complete view of the data layer and it has the ability to make optimal routing decisions. In this way, controller improves the network visibility. It exercises the direct control over the data plane through some well-defined application programming interfaces (API) based on a logically centralized, abstract view of the network. The network is programmable through software applications situated in the application plane which are running on top of the control plane. This plane has a set of applications that implement some network control functions like routing, load balancers, fault-tolerance, recovery, etc.

The separation of the control plane and the data plane can be comprehended by means of a well-defined programming interface between switches and the SDN controller. The controller has direct control access on the data plane elements through a well-defined application programming interface (API), as depicted in Fig. 1, known as Southbound API. The most notable of such API is OpenFlow. It is one of most popular standard protocol that is used for managing the communication between the data layer and controller through the Southbound channel.

The OpenFlow switches has got one or more tables which has the packet-handling rules known as flow tables. These rules direct the devices and the routers regarding the traffic flow. The network managers use these flow tables to modify the layout of the network and the traffic flows.

The Northbound API, similar to Southbound API, is the interface which represents the communication between control layer and application layer takes place. This Northbound interface abstracts the low-level instruction sets used by southbound interfaces to program forwarding devices [28].

SDN-specific Attacks

SDN attacks can be initiated from malicious management applications, malicious actions in the controller, or from compromised network entities, namely hosts and switches. Since the centralized controller makes SDN architecture unique

and powerful to offer its benefits in network management and development. For this reason, in this research we focus on security attacks that are associated with the controller plane.

Link Discovery Attacks

One of the key functionalities of the control plane is the network topology service. This service manages and updates the topological information and provides information to the application layer services like routing, security services, network management, policy implementation, etc. From this point forth, any intrusion, poisoning or, malfunction on this service would directly impact the other dependent services in the network since it would mislead the information of network topology [28].

Link discovery is an important process for SDN where the controller uses OpenFlow Discovery Protocol, which leverages the Link Layer Discovery Protocol (LLDP) packet, to dynamically detect direct links between adjacent switches. Network topology information will be provided at the end of the link discovery process. During this process, by manipulating the propagation of LLDP packets, an attacker may attempt to create fake links. We call this malicious act the Link Discovery Attack (sometimes it is also referred as Link Fabrication Attack or Link-Layer Discovery Protocol Attack or LLDP Attack). The attacker has two options to do that; either relay the genuine packet received from the one target switch to another or generate the fake LLDP packets and send them to the target switches. In both cases the controller thinks that a there exists a link between two switches, but actually there is no link between them. Eventually, any traffic that is routed through those false links will either let the attacker spy on the traffic by going through those malicious switches, or the traffic will get dropped [19]. See more details in [7, 9, 19, 28].

ARP Spoofing Attacks

ARP (Address Resolution Protocol) is a process that provides a service to resolve the mapping of data layer addresses like, IP addresses to link layer addresses and MAC addresses. To find the MAC addresses, a node in the network broadcasts an

ARP requests with the Sender Hardware Address (SHA) field set to its own MAC address and the Sender Protocol Address (SPA) field set to its own IP address, while the Target Protocol Address (TPA) set to the IP address of the target node and the Target Hardware Address (THA) is initialized to a dummy value. Each recipient node in the network will check if the TPA value in the request matches its own IP address, and if so, will respond with an ARP Reply message. As a result, ARP process will provide resolved mapping of the data layer addresses [28].

Spoofing attack occurs when an attacker impersonates another user's identity on a network. By this way, the attacker can steal other user's data or bypass the access control. ARP is a stateless protocol, which implies that it does not retain any information of any requests or replies. Since ARP treats each request or reply independently, a host can easily accept information from ARP replies without a corresponding request. ARP does not have mechanisms to authenticate the sender, replies, or integrity of information [1]. Thus, it makes it easy for an attacker to poison a host's ARP cache with a false IP-MAC address mapping. See more details in [1, 3, 12, 16, 28].

CHAPTER IV

PREDICTION OF SDN-SPECIFIC ATTACKS

Proposed Approach

Using a big data analytic approach ultimately involves applying an analytic algorithm in a distributed parallel computing framework in order to cope with enormous amount of data that do not fit in one machine. A variety of such frameworks exist with different computing paradigms (e.g., MapReduce, Storm, Spark, etc.). However, before dealing with large-scale data, we want to identify and show the effective techniques for smaller data set size.

As a preliminary study, we propose an approach to apply machine-learning algorithms to build a predictive model for predicting the presence of attacks. In particular, we apply seven popular algorithms using different forms of models, namely, Regression, BayesNet, Decision Tree, Decision Table, Random Forest, Random Tree, and Naïve Bayes. Regression classifier is based on a mathematical model in statistics where classifier is built to recognize one class at a time. BayesNet is a probabilistic model based on bayes rule using conditional probabilities. BayesNet is in the same family of Naive bayes classifier and Bayes Network learning algorithm. Decision tree is one of the most popular machine learning algorithms because of its easy to understand tree model and its ability to produce models with high accuracy for most data. Decision table can be viewed similar to decision tree learning where the resulting model is a table and each row represents a decision rule or a path in the tree. Random Forest is mostly used for classification with the aid of construction of decision trees by randomizing the combination of attributes. Random Tree also creates multiple trees but it considers K number of random selected attributes. Naïve Bayes is usually used for learning task that based on "Bayes" theorem. More details of these machine-learning techniques can be found in [22, 28]

Most anomaly or intrusion detection techniques for networks rely on monitoring critical measures of the communication (e.g., number of packets transmitted).

Abnormal number of this measure is observed. If the difference between normal and abnormal numbers is greater than some specified threshold then potential attack is identified. While this technique is simple, it is also problematic since determining the threshold can be subjective. Furthermore, if only one performance measure is used, the threshold-based technique is unable to detect more than one type of attacks at a time. Our approach aims to overcome this limitation and at the same time to obtain increased accuracy compared to that of threshold method. With the use of machine learning our approach can predict multiple classes and as a result distinguish different types of attacks to occur. Next chapter deals with how we acquire data for use in our experiments [28].

Machine Learning Algorithms

Machine learning is an application which grants ability to the computer program to learn from training dataset. Algorithms uses training data sets in order to make prediction or decisions. Training dataset is usually a record of events that happened in the past. Thus, the quality of the historical training data set can be accounted as a major fact for the prediction of the data. Machine learning have traditionally used in classification problems. However, machine learning techniques are improved and now it is used in many areas such as e-mail spam filtering, image recognition, data mining, recommendation systems etc. [27].

Decision Tree (J48)

J48 is a developed version of C4.5 that produces decision trees. J48 is a standard algorithm that is very common in the field of practical machine learning problems [24]. Most significant disadvantage of J48 is that, run-time complexity of the learner can't be greater than the number of attributes. Depth of the tree is related to tree size hence the size of dataset [25]. So, the size of J48 trees increases linearly with the number of dataset instances which is the main reason to have lower performance on noisy datasets.

Regression

A regression scheme that employs any classifier on a copy of the data that has the class attribute discretized. The predicted value is the expected value of the mean class value for each discretized interval (based on the predicted probabilities for each interval). This class now also supports conditional density estimation by building a univariate density estimator from the target values in the training data, weighted by the class probabilities [22].

BayesNet

BayesNet algorithm is commonly used to solve problem of intrusion detection in combination with statistical techniques [26]. It provides data structures (network structure, conditional probability distributions etc.) and facilities common to Bayes Network learning algorithms such as K2 and B [22].

Naïve Bayes

Naïve Bayes is usually used for learning task, where a training set with target class is provided [26]. This learner based on the "Bayes" theorem of probability to predict the class of unknown data instances. Naïve Bayes classifier assumes that one attribute is not related to the other attributes in the class. Simply, Naïve Bayes algorithm assumes all set of predictors are completely independent.

Random Forest

Random forest algorithm uses decision trees for classification. The basis of random forest learner is the creation of various decision trees by randomizing the combination of variables. Each generated decision tree makes its own prediction. Decision trees which made the correct prediction reinforces each other while other decision trees which made the false prediction are nullified. Since unnecessary decision trees are nullified, it makes random forest algorithm strong to deal the overfitting problem.

Random Tree

Random Tree is a supervised classifier and it constructs a tree in the consideration of K number of randomly chosen attributes at each node. It doesn't

perform any pruning. Random Tree learner can be used for both classification or regression problems.

Decision Table

Decision tables are classification models used for prediction. Every decision tree can be represented as a decision table. The structure of the decision tree is similar to dimensional stacking [27].

Data Simulation

Most research studies in SDN architecture use a simulation of network and controller for evaluation. Here we use simulated data from Mininet [5] for emulating virtual SDN/OpenFlow networks, and POX, a Python-based controller for software-defined networking. The forwarding devices are Open vSwitch. We configured the Mininet with a specific network topology. The malicious switches and hosts vary depending on the test being run. The speed of the communication links was set to 1 Gbps. We assume that the links have enough bandwidth and the network has no traffic congestion. When the network functions normally, Transmission Control Protocol (TCP) packets are sent from the source host to the destination host at the transfer rate of 520 pps. The data packets for the attacks have been simulated using Scapy library [23], which is commonly used for communication packets. In this paper, our simulation considers only cases when the network has a single attack at a time. The attack can be either the Link Discovery Attack (Link fabrication, LLDP attack) or the ARP spoofing attack.

In the Link Discovery Attack, an attacker's goal is to give the controller's a perception of link that does not exist. In general, the controller discovers "Link" in the topology if there is an evidence of a packet sent between the two nodes. This evidence is shown by the recipient's node sending a message to the controller notifying a packet being sent from a sender's ID and address. Thus, we can simulate a "fake" link between two nodes by injecting a falsify message to the controller notifying a packet being sent from either a non-existing sender's ID or from a legitimate sender's ID/address but with a non-existing/unused port.

In the ARP spoofing attack, an attacker wants to spoof information of the target's node or poison the target node's ARP table that contains IP/MAC addresses of all nodes in the network. The attacker can use Ettercap tool [14] to scan through the network to identify vulnerable node to compromise. From the compromised node, the attacker sends a message to a target's node pretending to be a legitimate node in the network. This way, the target's node leaks its information by sending packets to the attacker (through the compromised node) thinking that he is legitimate. Thus, we can simulate the ARP attack by sending a packet to a target's node with a falsify header's information (e.g., fake IP/MAC address). We tested the simulated attack mechanism by Wireshark [19], a packet analyzer tool to see the ARP table content.

The simulation was run on HP v7x machine having Intel(R) Core(TM) i7-5600U CPU at 2.60 GHz, and 8GB of RAM, running 64 bit Windows 10 Pro [28].

CHAPTER V

EXPERIMENTS

Data Sets and Descriptions

Table I, gives a detailed information and description of the attributes that we simulated on the purpose of predicting the network attacks in all of our experiments. Each data instance (row) provides information about a specific node in the network during each of its communication to and from controller and adjacent nodes during a packet transfer between destination and source nodes [28].

Table I. Data Descriptions

No.	Attribute	Description (Data Type)
1	NodeID	Id of the node (numeric)
2	NodeAddress	Address of the node (numeric)
3	NodePort	Port of the node (discrete: 1 to 4)
4	C-Address	Address of the controller (numeric)
5	C-Port	Port of the controller (discrete: 1 to 4)
6	BandwidthUsed	Utilized bandwidth between this node to one of its adjacent nodes (numeric)
7	Packet-Rcvd	# packets received by this node from all incoming links (discrete: 1 to 1112)
8	Packet-Sent	# packets sent by this node to all outgoing links (discrete: 1 to 1648)
9	Transmitted_bytes	Total # of transmitted bytes of this node both sent and received (numeric: 1 to 113382)
10	Rcvd_bytes	Total # of received bytes for this node from all incoming links (numeric: 1 to 75408)
11	Rcvd_pkt_in_Cntr	Controller's total # of received packets (numeric: 1 to 1059)

Table I. Continued

12	Sent_pkt_out_Cntr	Controller's total # of sent packets (numeric 1 to 1648)
13	Trans_bytes_Cntr	Controller's total # of transmitted packets (numeric: 1to 46144)
14	Class	Status of the network (No attack, LLDP attack, ARP attack)

As shown in Table I, not all of these data are necessary to be predictive factors of the classification model for attack prediction (e.g., NodeID, C-Address). Though, we include them for completeness of the information about a particular node. First step of data analytic is to clean the data set based on domain knowledge. However, in this work, we skip this step for simplicity and in order to show the strength of machine learning algorithms [28].

Table II. Dataset Sizes in Two Network Topologies

Network Size	# Nodes	# No Attack Instances	#ARP Attack Instances	#LLDP Attack Instances	Dataset Size
Small	13	10,270	20,291	20,409	50,970
Medium	21	20,000	20,000	20,000	60,000

Combining all simulated data of each case for each network topologies, we obtained two data sets whose characteristics and size of data set can be summarized in Table II [28].

Experimental Setup

Our experiments are divided into two sections: 1) simulating SDN-specific attacks on both small and mid-size network and 2) experiments on scalability where we measure the training time for each machine learning algorithm. To obtain data relevant to SDN-specific attacks, we simulated data transmission from a source host to a destination host in SDN network with no attack and with each type of attack (i.e., LLDP link fabrication attack and ARP spoofing attack). For each attack in each network topology, we simulated the attack on two specific network locations, first of

them is just next to the source and second of them is near the destination. Recall that we only consider when only one single attack occurs at a time during an attack period. The details of these attacks will be described later. Also, data for experiments on scalability is gathered by simulating the mid-size network for longer time which consist of no attack and with each type of attack (LLDP link fabrication and ARP spoofing attack.

By using Weka data mining tool [22], the experiments were run on Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz (4 Core CPUs and 4 Logical CPUs) with 2.5GHz, and 8192MB RAM on Windows 10 OS. Below gives results obtained from each scenario.

Experiments on Network Attack Detection

Small Network

A topology of a small network with three hosts (X, Y, Z) and 10 switches is shown in Figure 2.

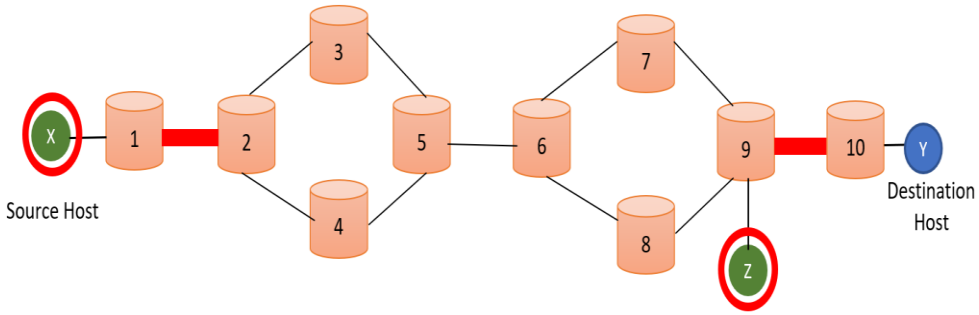

Figure 2. Small Network of 13 Nodes

For LLDP attack, by compromising host X, an attacker aims to create a forged link between switches 1-2, giving a link fabrication attack. We also simulated another link fabrication attack between switches 9-10 at a different time. For ARP spoofing attack, X and Z are victim hosts for spoofing. The attacks are indicated as thick lines for two events of link discovery attacks and circles for two events of spoofing attacks.

Mid-size Network

A topology of a mid-size network with three hosts (X, Y, Z) and 18 switches is shown in Figure 3.

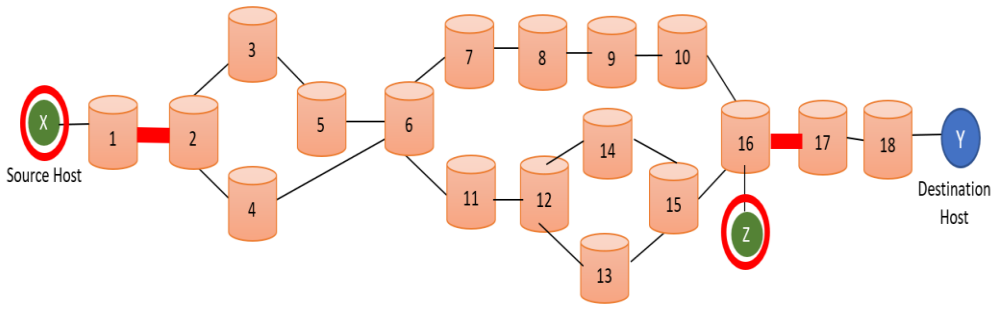

Figure 3. Mid-size Network of 21 Nodes

In this scenario, for LLDP attack, by compromising host X, an attacker aims to create a forged link between switches 1-2, giving a link fabrication attack. At a different time, host Z was compromised and fake link between switches 16-17 was created. For ARP spoofing attack, X and Z are victim hosts for spoofing. The attacks are indicated as thick lines for two events of link discovery attacks and circles for two events of ARP spoofing attacks [28].

Experiments on Scalability

Besides predicting the presence of attack, as data become larger, other concern is how well these machine learning algorithms scale in terms of their training time. We investigate this by measuring the training time of each of these algorithms as the training set size grows.

Experimental Results

Small Network

Data is randomly shuffled for each experiment and then the data are split into training (66%) and testing data sets for the rest of the data. Table III shows average results, obtained by each of the machine learning algorithms, over 10 experiments.

Table III. Comparison Results on Average for Small Network

Algorithm	Accuracy	TP Rate	FP Rate	ROC
Regression	99.87	0.98	0.001	0.98
Decision Tree	99.97	0.98	0.001	0.99
BayesNet	99.77	0.97	0.002	0.97
DecisionTable	99.72	0.96	0.002	0.96
RandomTree	99.71	0.97	0.002	0.96
RandomForest	99.71	0.96	0.002	0.96
NaiveBayes	54.25	0.54	0.132	0.85

As shown in Table III, apart from NaiveBayes machine learner, all algorithms obtained over 99% of accuracy with low false positive and high ROC of over 0.95 in each of the machine learning models. Because of excellent results, we wonder if the results are over-fitting to the sample even though our sample size for this small network has 50,970 instances. To investigate further, we conduct experiment on an increased size of the network.

Mid-size Network

Experiments were run for same machine-learning algorithms that has been used for small network topology. Table IV shows average results for mid-size network, obtained by each of the machine learning algorithms, over 10 experiments as the same way for small network experiments.

Table IV. Comparison Results in Mid-size Network

Algorithm	Accuracy	TP Rate	FP Rate	ROC
Regression	79.96	0.800	0.100	0.952
Decision Tree	79.10	0.791	0.104	0.943
BayesNet	71.23	0.712	0.143	0.922
DecisionTable	76.96	0.770	0.115	0.939
Random Tree	76.48	0.766	0.118	0.892
Random Forest	76.53	0.765	0.117	0.916
NaiveBayes	49.36	0.490	0.250	0.792

As shown in Table IV, Regression obtained the highest accuracy followed by Decision Tree with no significant differences. They are top two performers that obtained accuracy of about 79%, low false positive of 0.1 and high ROC of 0.9. While BayesNet and Decision Table do not receive accuracy as high, their performances are relative good. Due to limitation of Naïve Bayes as mentioned earlier in Chapter IV, naïve bayes showed the lowest performance since the algorithm assumes all predictors (features) are independent. Random Tree and Random Forest decision tree algorithms performance resulted in average success compared to other algorithms [28].

In general, the results of the mid-size network show that performance of predictive models are sensitive to size and structure of the network. Regression's and Decision Tree's performance are superior to BayesNet and Decision Table. However, the general results by ROC of both network topologies are close to each other with ROC of small network slightly better. ROC is a better measure for quality of predictive models because it takes not only accuracy but also false positives into account [28].

Scalability

As the prediction of network attack is an essential matter, at the same time, it is also critical to see how well these techniques perform as the data set is increased. For this reason, we furthered our analysis on scalability with the top 4 performer of network attack detection algorithms.

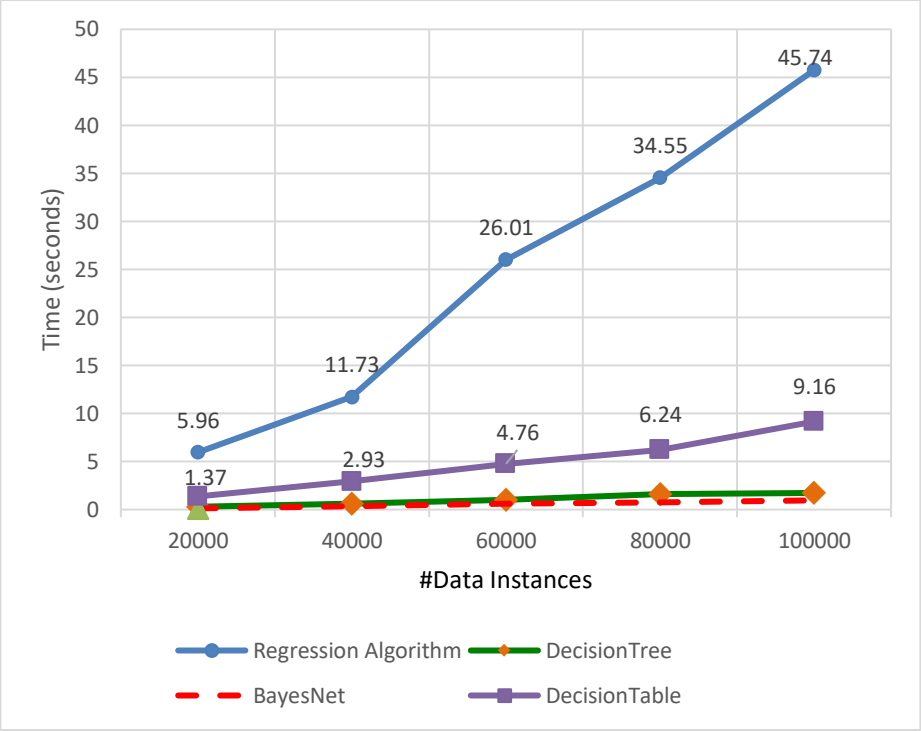

Figure 4. Training Times vs Training Set Sizes

As shown in Figure 4, the training time of each of the machine learning algorithms is near linear with different rates. The Regression technique takes most time while the Decision Tree and BayesNet are top two performers. These preliminary results are encouraging. Detailed information of the output for each algorithm is shown in the Appendix A.

CHAPTER VI

CONCLUSION AND FUTURE WORK

In this paper, security challenges in SDN network have been addressed. In order to improve the network attack detection rate, we propose an approach to predict attacks in the SDN networks by applying machine learning techniques instead of using the traditional technique with threshold values, which tend to be problematic due to dynamic environments of SDN. We have analyzed and investigated various machine learning techniques which can be used to predict the presence of attack. Our proposed method not only predicts the presence of attack but also attack type by a predictive model, which represents the behaviors of the network, particularly under ARP attack, LLDP Attack, or no attack [28].

We concluded that, performance of the applied machine learning algorithms is sensitive to size and design of the network topology. We also showed how our proposed method performed under various data sizes with the use of different machine learning techniques.

However, pinpointing the location of the attack is as important as predicting the existence of attack. We note during experiments that our proposed approach has potential to locate the origin of the attack. The future work aims to identify methodology to determine the location of these attacks as well as applying machine learning algorithms in distributed and parallel computing environments for large-scale systems [28].

Other future work aims to get rid of our limitation on the number of attacks that occurs at a time. We would like to consider multiple same type of SDN-specific attacks on different hosts at once. Moreover, we also would like to see the performance of our proposed approach on different type of SDN-specific attacks on different hosts at the same time.

Finally, we also think that some modifications on the network topology may be helpful for our approach to have better results. Thus, our future work also aims to investigate and find a structural design pattern that may help our approach to perform

more efficiently both in the terms of scalability and accuracy to detect network attacks.

BIBLIOGRAPHY

[1] AbdelSalam, A.M., A. El-Sisi, and V. Reddy, "Mitigating ARP Spoofing Attacks in Software-Defined Networks," in Proceedings of the International conference on Computer Theory and Applications (ICCTA), 2016.

[2] Ahmad, I., S. Namal, M. Ylianttila, and A. Gurtov, "Security in Software Defined Networks: A Survey", IEEE Communications Surveys & Tutorials, Volume: 17, Issue: 4, pp: 2317 – 2346, 2015.

[3] Alharbi, T., D. Durando, F. Pakzad and M. Portmann, "Securing ARP in Software Defined Networks," in Proceedings of IEEE 41st Conference on Local Computer Networks (LCN), pp.523-526, 2016.

[4] Ashraf, J. and S. Latif, "Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques," in Proceedings of Software Engineering Conference (NSEC), 2014 National, pp. 55–60, 2014.

[5] Eskca, B., O. Abuzaghleh, P. Joshi, S. Bondugula, T. Nakayama, and A. Sultana, "Software Defined Networks Security: An Analysis of Issues and Solutions," International Journal of Scientific & Engineering Research, Volume 6, Issue 5, pp.1270-1275, 2015.

[6] Fontes, R., S. Afzal, S. Brito, M. Santos and C. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," in Proceedings of 11th International Conference on Network and Service Management (CNSM), 2015, pp. 384-389. MININET

[7] Hong, S., Xu, L., Wang, H., and Gu, G., "Poisoning network visibility in software-defined networks: New attacks and countermeasures", Network and Distributed System Security (NDSS) Symposium, USENIX, 2015.

[8] Hu, Z., Wang, M., Yan, X., Yin, Y. and Luo, Z., "A comprehensive security architecture for SDN", Intelligence in Next Generation Networks (ICIN), 18th International Conference on (pp. 30-37), IEEE, 2015.

[9] Khan, S., Gani, A., Wahab, A. W. A., Guizani, M., & Khan, M. K., "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," IEEE Communications Surveys & Tutorials, 19(1), 303-324, 2017.

[10] Kim, H. and Feamster, N., "Improving network management with software defined networking," IEEE Communications Magazine, 51(2), pp.114-119, 2013.

[11] Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S. and Uhlig, S., "Software-defined networking: A comprehensive survey", in Proceedings of the IEEE, 103(1), pp.14-76, 2015.

[12] Masoud, M. Z., Y. Jaradat, and I. Jannoud. "On preventing ARP poisoning attack utilizing Software Defined Network (SDN) paradigm." In Proceedings of Conference on Applied Electrical Engineering and Computing Technologies (AEECT), IEEE, 2015.

[13] Mendiola, A., Astorga, J., Jacob, E. and Higuero, M., "A survey on the contributions of software-defined networking to traffic engineering," IEEE Communications Surveys & Tutorials, 19(2), pp.918-953, 2017.

[14] Mousavi, S., St-Hilaire Marc. "Early Detection of DDoS Attack Against Software Defined Network Controllers," Journal of Network and Systems Management, Vol.26(3), pp.573-591, 2018.

[15] Ornaghi, A. and M. Valleri, "Ettercap," available at URL: http://ettercap.github.io/ettercap. October, 2018.

[16] Ramachandran, V. and S. Nandi, "Detecting ARP Spoofing: An Active Technique", ICISS 2005: Information Systems Security, pp 239-250, 2005.

[17] Scott-Hayward, S., O'Callaghan, G. and Sezer, S., "SDN security: A survey," Future Networks and Services (SDN4FNS), IEEE SDN, pp. 1-7, 2015.

[18] Shin, S. and G. Gu, "Attacking software-defined networks: A first feasibility study," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13), ACM, 2013.

[19] Smyth, D., S. McSweeney, D. O'Shea and V. Cionca, "Detecting Link Fabrication Attacks in Software-Defined Networks", in Proceedings of 26th International Conference on Computer Communication and Networks (ICCCN), 2017.

[20] Wang, H., L. Xu, and G. Gu, "Floodguard: A dos attack prevention extension in software-defined networks," in Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'15), June 2015.

[21] WireShark, "WireShark Official Documentation," available at URL: https://www.wireshark. org/docs, October, 2018.

[22] Witten, I.H., Frank, E., Hall, M.A. and Pal, C.J., Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2016.

[23] Sacpy, available at URL: P. Biondi, http://www.secdev.org/projects/scapy/

[24] Witten, I.H., Frank, E., Trigg L., Hall, M.A., Holmes G., and Cunningham S., Weka: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kauffman, 1999.

[25] Bhargava N., Sharma G., Bhargava R., Mathuria M., “Decision Tree Analysis on J48 Algorithm for Data Mining”, in International Journal of Advanced Research in Computer Science and Software Engineering, June 2013.

[26] Ashraf, Javed & Latif, Seemab. (2014). Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques. 55-60. 10.1109/NSEC.2014.6998241.

[27] B. G. Becker, "Visualizing decision table classifiers," *Proceedings IEEE Symposium on Information Visualization (Cat. No.98TB100258)*, Research Triangle, CA, USA, 1998, pp. 102-105.
doi: 10.1109/INFVIS.1998.729565

[28] E. Unal, S. Sen-Baidya and R. Hewett, "Towards Prediction of Security Attacks on Software Defined Networks: A Big Data Analytic Approach," *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 4582-4588.

APPENDIX A

CLASSIFIER OUTPUTS FOR 100000 DATA INSTANCES

A.1 REGRESSION RUN ON 100000 DATA INSTANCES

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.meta.ClassificationViaRegression -W weka.classifiers.trees.M5P -- -M 4.0
Relation:     DataSet_100000
Instances:    100000
Attributes:   14
              node(id)
              node(address)
              node(port)
              cntr(address)
              cntr(port)
              utilized_bw
              rcvd_pkt
              snt_pkt
              transmitted_byte
              rcvd_bytes
              cntr_pkt_out
              cntr_pkt_in
              cntr_transmitted_byte
              Class
Test mode:    split 66.0% train, remainder test

Number of Rules : 792



Time taken to build model: 51.89 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.18 seconds

=== Summary ===

Correctly Classified Instances       28928               85.0824 %
Incorrectly Classified Instances      5072               14.9176 %
Kappa statistic                          0.7554
Mean absolute error                      0.1409
Root mean squared error                  0.2621
Relative absolute error                 33.8263 %
Root relative squared error             57.426  %
Total Number of Instances            34000

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.956    0.163    0.855      0.956    0.902      0.798    0.971     0.971     No Attack
               0.720    0.038    0.865      0.720    0.786      0.728    0.958     0.904     AttackARP
               0.772    0.053    0.829      0.772    0.799      0.737    0.956     0.898     AttackLLDP
Weighted Avg.  0.851    0.104    0.851      0.851    0.848      0.765    0.964     0.936

=== Confusion Matrix ===

     a     b     c   <-- classified as
 16263   413   343 |     a = No Attack
  1391  6163  1001 |     b = AttackARP
  1376   548  6502 |     c = AttackLLDP
```

A.2 DECISION TREE RUN ON 100000 DATA INSTANCES

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     DataSet_100000
Instances:    100000
Attributes:   14
              node(id)
              node(address)
              node(port)
              cntr(address)
              cntr(port)
              utilized_bw
              rcvd_pkt
              snt_pkt
              transmitted_byte
              rcvd_bytes
              cntr_pkt_out
              cntr_pkt_in
              cntr_transmitted_byte
              Class
Test mode:    split 66.0% train, remainder test

Number of Leaves  :     320

Size of the tree :      586


Time taken to build model: 1.88 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.07 seconds

=== Summary ===

Correctly Classified Instances       28517               83.8735 %
Incorrectly Classified Instances      5483               16.1265 %
Kappa statistic                          0.736
Mean absolute error                      0.1446
Root mean squared error                  0.2718
Relative absolute error                 34.7025 %
Root relative squared error             59.5669 %
Total Number of Instances            34000

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.948    0.164    0.853      0.948    0.898      0.789    0.963     0.958     No Attack
               0.694    0.038    0.859      0.694    0.768      0.707    0.944     0.883     AttackARP
               0.765    0.068    0.788      0.765    0.777      0.705    0.948     0.882     AttackLLDP
Weighted Avg.  0.839    0.108    0.839      0.839    0.835      0.748    0.954     0.920

=== Confusion Matrix ===

     a     b     c   <-- classified as
 16136   382   501 |     a = No Attack
  1391  5935  1229 |     b = AttackARP
  1391   589  6446 |     c = AttackLLDP
```

A.3 BAYESNET RUN ON 100000 DATA INSTANCES

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.K2
Relation:     DataSet_100000
Instances:    100000
Attributes:   14
              node(id)
              node(address)
              node(port)
              cntr(address)
              cntr(port)
              utilized_bw
              rcvd_pkt
              snt_pkt
              transmitted_byte
              rcvd_bytes
              cntr_pkt_out
              cntr_pkt_in
              cntr_transmitted_byte
              Class
Test mode:    split 66.0% train, remainder test


Time taken to build model: 0.85 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.12 seconds

=== Summary ===

Correctly Classified Instances       25063               73.7147 %
Incorrectly Classified Instances      8937               26.2853 %
Kappa statistic                          0.595
Mean absolute error                      0.1792
Root mean squared error                  0.3281
Relative absolute error                 43.0068 %
Root relative squared error             71.8952 %
Total Number of Instances            34000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.735    0.074    0.908      0.735    0.812      0.673    0.952     0.953     No Attack
                 0.628    0.077    0.733      0.628    0.677      0.582    0.924     0.836     AttackARP
                 0.853    0.223    0.557      0.853    0.674      0.560    0.927     0.848     AttackLLDP
Weighted Avg.    0.737    0.112    0.777      0.737    0.744      0.622    0.939     0.897

=== Confusion Matrix ===

     a     b     c   <-- classified as
 12503  1349  3167 |     a = No Attack
   634  5376  2545 |     b = AttackARP
   629   613  7184 |     c = AttackLLDP
```

A.4 DECISION TABLE RUN ON 100000 DATA INSTANCES

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.rules.DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D 1 -N 5"
Relation:     DataSet_100000
Instances:    100000
Attributes:   14
              node(id)
              node(address)
              node(port)
              cntr(address)
              cntr(port)
              utilized_bw
              rcvd_pkt
              snt_pkt
              transmitted_byte
              rcvd_bytes
              cntr_pkt_out
              cntr_pkt_in
              cntr_transmitted_byte
              Class
Test mode:    split 66.0% train, remainder test
```

```
Decision Table:

Number of training instances: 100000
Number of Rules : 655
Non matches covered by Majority class.
        Best first.
        Start set: no attributes
        Search direction: forward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 80
        Merit of best subset found:   79.473
Evaluation (for feature selection): CV (leave one out)
Feature set: 2,10,12,14

Time taken to build model: 9.81 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.16 seconds

=== Summary ===

Correctly Classified Instances       27336               80.4    %
Incorrectly Classified Instances      6664               19.6    %
Kappa statistic                          0.6748
Mean absolute error                      0.1639
Root mean squared error                  0.2828
Relative absolute error                 39.3463 %
Root relative squared error             61.9678 %
Total Number of Instances            34000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.939    0.237    0.799      0.939    0.863      0.713    0.960     0.961     No Attack
                 0.657    0.058    0.792      0.657    0.718      0.640    0.939     0.859     AttackARP
                 0.681    0.046    0.830      0.681    0.749      0.682    0.941     0.865     AttackLLDP
Weighted Avg.    0.804    0.144    0.805      0.804    0.798      0.687    0.950     0.912

=== Confusion Matrix ===

     a     b     c   <-- classified as
 15977   720   322 |     a = No Attack
  2086  5618   851 |     b = AttackARP
  1934   751  5741 |     c = AttackLLDP
```