Predicting Network Attack Patterns in SDN using Machine Learning Approach

Saurav Nanda*, Faheem Zafari*, Casimer DeCusatis†, Eric Wedaa‡ and Baijian Yang*
*Deparment of Computer and Information Technology
Purdue University, West Lafayette, IN 47906
Email: {nandas,faheem0,byang}@purdue.edu
†School of Computer Science and Mathematics
Marist College, Poughkeepsie, NY 12601
Email: casimer.decusatis@marist.edu
‡Information Technology
Marist College, Poughkeepsie, NY 12601
Email: eric.wedaa@marist.edu

***Abstract*—An experimental setup of 32 honeypots reported $17M$ login attempts originating from 112 different countries and over 6000 distinct source IP addresses. Due to decoupled control and data plane, Software Defined Networks (SDN) can handle these increasing number of attacks by blocking those network connections at the switch level. However, the challenge lies in defining the set of rules on the SDN controller to block malicious network connections. Historical network attack data can be used to automatically identify and block the malicious connections. There are a few existing open-source software tools to monitor and limit the number of login attempts per source IP address one-by-one. However, these solutions cannot efficiently act against a chain of attacks that comprises multiple IP addresses used by each attacker. In this paper, we propose using machine learning algorithms, trained on historical network attack data, to identify the potential malicious connections and potential attack destinations. We use four widely-known machine learning algorithms: *C4.5, Bayesian Network (BayesNet), Decision Table (DT), and Naive-Bayes* to predict the host that will be attacked based on the historical data. Experimental results show that average prediction accuracy of 91.68% is attained using Bayesian Networks.**

***Index Terms*—Network Attack; Machine Learning; Honeypots; SDN**

I. Introduction

The drastic increase in the number of devices connected to the Internet has resulted in a number of useful solutions in different fields such as agriculture, health care, industry, commerce. Such a huge increase in demand for connectivity has challenged the traditional network architectures. To account for the challenges, Software Defined Network (SDN) architecture was proposed that decouples the traditional user plane and control plane [1], [2]. The advantage of such an architecture is that it improves the overall network efficiency and allows better network management [2]. While, such an architecture has valuable advantages, it is also prone to number of threats such as security attacks.

An attacker can perform attacks, such as Secure Shell (SSH) bruteforce attack, on the SDN controller that can result in serious security threats. Even if the network administrator identifies a potential attack and attacker, it may not be possible to efficiently account for simultaneous attacks in real time. Therefore, there is a need for specific security rules that are usually implemented on the SDN controller similar to firewall rules. However, defining these rules can be a daunting task, since the goal of such rules is to restrict the access of the malicious nodes or attackers while allowing smooth access to the normal users.

Abdou et al. [3] found that the malicious users have specific characteristics that can be used to differentiate between the attackers and legitimate users. Patterns such as coordinated attacks and sharing of password dictionary are common among attackers. Different techniques, including machine learning, can be used to identify such patterns. Machine learning based approaches have shown significant potential in classification of the users [4]. In this paper, we apply four different machine learning algorithms: *C4.5, BayesNet (BN), Decision Table (DT), and Naive-Bayes (NB)* to predict the potential vulnerable host that might be attacked, using historical network attack data. We use data from the *LongTail* project [5] to train different Machine Learning (ML) models to predict the potential host that can be attacked. Leveraging the ML-algorithm output, we can define security rules on the SDN controller to restrict the access of potential attackers by blocking the entire subnet. The main contributions of this paper are:

- To the best of our knowledge, our work is the first paper that leverages Machine Learning approach for defining security rules on the SDN controller
- We compare and evaluate the performance of four widely used ML algorithms. However, our goal is to show the viability of ML approach in SDN security rather than highlighting the four algorithms used.
- We show that even a small probability of attack, obtained through ML approach, has significant effect on the SDN security.

The paper is structured as follows: Section II discusses related work and presents an insight into the ML algorithms used in this work. Section III presents our methodology. Section 4 presents the experimental results and a discussion on our findings. Section V presents the conclusions.

II. Related Work

In this section, we present a review of relevant literature related to SDNs and Machine Learning (ML). We also present an insight into ML algorithms used in this paper.

A. Literature Review

Ashraf et al. [2] described machine learning techniques that could be used to handle intrusions and Distributed Denial of Service (DDoS) attacks in Software Defined Networks (SDNs). The paper discussed neural networks, Bayesian networks, Fuzzy logic, genetic algorithm and support vector machine and their application in SDN anomaly detection. The paper presents a detailed discussion on the advantages and disadvantages of these approaches for anomaly detection. Ali et al. [6] present a detailed survey on leveraging SDNs to secure networks, and propose the use of SDNs for security as a service. The survey lists a number of different challenges and solutions that have been proposed in the literature to account for network threats. Astuto et al. [7] also present a survey on programmable networks with emphasis on SDNs. The paper provides a discussion on the evolution of programmable networks and highlights the SDN architecture. The paper also describes various alternative solutions to the OpenFlow standard and testing of SDN protocols. Hu et al. [8] present a survey of SDN from OpenFlow perspective and highlight the basic concept, applications, and security aspects of OpenFlow. Abdou et al. [9] present an analysis of the automated SSH brute force attacks. The paper used the data from the *LongTail* project [5] to thoroughly analyze the attacker's behavior and the dynamics of the attacks including the password dictionary sharing and coordinated attacks. The result of the analysis can be leveraged to provide recommendations to the network administrator and SSH users. Sommer [10] discusses different anomaly detection mechanisms in SDN such as k-Nearest Neighbors (kNN), Bayesian Networks, Support Vector Machines, and Expectation Maximization. The author elaborates different attack scenarios, and their implementation in terms of SDN applications. Qazi et al. [4] proposed a innovative framework, called *Atlas*, that leverages application-awareness in SDN, and is efficient for $L2/3/4$-based policy enforcement. *Atlas* uses a machine learning approach, $C5.0$ classifier, to classify the traffic in SDN, and collects ground truth data using crowd-sourcing approach to integrate with the centralized control of SDN's data reporting system. Their proposed system is capable of detecting a mobile application with fine-granularity, and achieved an average accuracy of 94% for the top 40 android applications.

Kim et al. [11] provide a generic discussion on SDN with particular emphasis on the present issues in the network configuration and management systems, and propose different mechanisms to improve network management. They focused on the three major challenges of network management: capability of frequent changes to network conditions and state, high level language support for network configuration, and better interface and control to conduct network troubleshooting. Drutskoy et al. [12] present a system on top of SDN layer known as FlowN that leverages programmable control on network switches, and allows different tenants to define their own rules for routing and control. FlowN has the capability of providing individual address space, topology and control logic to their tenants. They also take advantage of databases to scale the mapping between physical and virtual networks. Eskca et al. [1] present a detailed discussion on security aspect of SDN. The paper discusses a number of techniques and approaches including machine learning that can be used to address the security challenges. Jain et al. [13] present a novel system called B4, which is a private Wide Area Network (WAN) that connects Google's data centers around the world. The proposed approach has unique features such as high bandwidth for reasonable number of data centers, dynamic traffic demand that aims to maximize the average bandwidth, and thorough control over the edge servers. Due to the aforementioned characteristics, B4 leverages SDNs improved control over the networks switches resulting in near 100% link utilization.

In contrast with the prior work, we leverage machine learning algorithms to predict the networks that might be attacked and limit the access of malicious users. The proposed approach can be leveraged by SDN controller to define security rules that aims to prevent potential attacks by blocking an entire subnet, rather than the traditional approach of blocking individual IPs. The motive behind blocking the entire subnet is that most of the attackers have been found to use multiple IP addresses within the same subnet range.

B. Machine Learning Algorithms

Machine learning algorithms have been widely used for a number of classification and prediction problems [14] and have provided accurate results. Below we describe some of the widely used ML algorithms.

1) C4.5 Decision Tree: C4.5 Decision tree is widely used for inductive inference [15]. In C4.5, the discrete-valued functions are approximated and decision tree is used for representing the learned function. C4.5 is based on heuristic hill climbing, and carries out non-backtracking search throughout all possible decision trees. In C4.5, the data is partitioned into subgroups recursively. C4.5 is robust to highly noisy data and is preferred for learning various disjunctive expressions. As discussed in [15], the main learning steps of C4.5 DT are

- An attribute is selected, based on which a logical test is formulated.
- Each test outcome is used as a branch and subset of the training data that satisfies the outcome is moved to the corresponding child node.
- The process is run recursively on all the child nodes.
- A leaf is declared as a node based on specific termination rule.

C4.5 is suitable for large datasets. Since SDN related data is huge in size, C4.5 DT is a viable option [2]. A detailed discussion on C4.5 DT can be found in [15], [16].

2) Bayesian Network: Bayesian Network or Bayes Net encodes probabilistic relationships among different variables of interest [2]. It consists of a number of variables and set of edges between the variables, resulting in an acyclic graph [17]. Every node in the graph represents the random variable and a directed edge from one variable to another. Every variable in the Bayesian network is independent of the non-descendants. Bayes Net has been used as classifier and if trained properly, can result in highly accurate classifications. A detailed discussion on Bayesian Network can be found in [16], [18], [19].

3) Naive-Bayes: Naive-Bayes uses Bayesian theory that predicts the type of the unknown samples based on prior probability using the training samples. The Bayesian classification model relies on statistical analysis and Bayesian theory that consists of the Bayesian learning [20], [21]. Bayesian learning uses the prior and posterior probability in combination and uses it to find the posterior probability as per the supplied information and data samples [22]. The Naive-Bayesian algorithm operates by segregating the training set into an attribute vector and a decision variable. The algorithm also assumes that every member of the attribute vector independently acts on the decision variables. A detailed discussion on Naive-Bayes can be found in [22].

4) Decision Table: Decision Table (DT) is used to organize and document logic in a way that assists in easy inspection [23]. DT helps in representing the machine learning output as the input [24], and involves selecting some of the data attributes. They also assist in evaluating different set of rules for ambiguities and redundancy [25]. A detailed discussion, which is beyond the scope of this paper, can be found in [23]–[25].

III. Methodology

We use machine learning (ML) algorithms to predict potential target host attacks based on the historical network attack data for SDN. We use four different algorithms: C4.5, BayesNet, DT, and Naive-Bayes [26] for predicting the host that would be attacked, and compare their performance in terms of accuracy. Figure 1 shows our ML-based architecture for defining security rules on SDN controller. Historical data is used to train a model. The trained model is then used for predicting the potential attacks on different hosts using real-time network data, and specific rules are set on the SDN controller to block the potential attacker. The underlying principles of our proposed approach consist of a) Using historical data to train the ML-based models, and b) Using the trained model to identify the potentially vulnerable hosts and define the security rules in the SDN controller as per the prediction output of the Machine learning algorithm. Below we discuss each one of them in detail

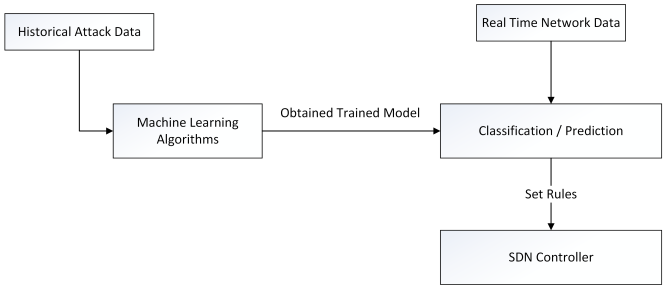

Fig. 1: Machine learning based architecture for defining security rules on SDN controller

A. Use historical data to train the ML-based models

To obtain accurate classifiers to identify potential vulnerable hosts, historical data is required to train the ML-based models. The training helps the model learn and obtain better results. Our goal is to use the historical attack pattern to identify which host will be attacked by a potential attacker. Based on attacker IP, we predict the potential host that can be attacked. These predictions can be used to define security rules on SDN controller to ascertain the network security. Rather than restricting the access of a single IP, we also propose blocking the entire subnet to prevent the future attacks from the same attacker, attacking through a different IP in the same subnet.

B. Use the trained model to identify potentially vulnerable host

Once the model is trained, it can be used to identify potential hosts that can be attacked by an IP. During the testing phase, we used our trained model to predict the attacked host based on the attacker's IP. If the attacker actually attacked a host as predicted by the ML algorithm, it means that the model is accurate. The accuracy of the models is calculated using Equation 1.

$$Accuracy = \frac{Number\ of\ correctly\ predicted\ attacks}{Total\ number\ of\ attacks} \times 100 \tag{1}$$

While testing, we also chose a threshold level α percentage as the minimum probability required to consider any host as vulnerable. We altered the values of α to evaluate its effect on the classification accuracy. Algorithm 1 summarizes our proposed approach.

Algorithm 1 Machine Learning based predictor for SDN network attacks

```
procedure ML-BASED ATTACK PREDICTOR
    Chose the Machine Learning Algorithm
    Train the ML algorithm using historical data
    if The trained model predicts an attack on a host by an IP then
        Update the SDN controller rules to block the IP subnet
    else Allow the IP to access the resources
```

TABLE I: Details of the dataset

Dataset	**Size**	**Format**
1	278,598 (With Chinese attack data)	<attacker IP> <attacked host> <number of attempts in an attack> <timestamp>
2	187,488 (Without Chinese attack data)	<attacker IP> <attacked host> <number of attempts in an attack> <timestamp>
2	91,110 (Only Chinese attack data)	<attacker IP> <attacked host> <number of attempts in an attack> <timestamp>

TABLE II: Prediction accuracy of different ML algorithms for the dataset 1 in different training/testing split scenarios and threshold (α)

Algorithm	**Split**	α**=10%**	α**=5%**	α**=1%**	α**=0%**
C4.5	30-70	78.62	85.54	87.95	88.22
C4.5	40-60	80.13	86.72	88.94	89.2
C4.5	50-50	80.96	87.5	89.68	89.91
C4.5	60-40	81.88	88.47	90.61	90.8
C4.5	70-30	82.3	89.1	91.2	91.4
NB	30-70	67.7	79.32	96.86	99.52
NB	40-60	68.62	80.24	96.87	99.52
NB	50-50	69	80.65	96.87	99.51
NB	60-40	69.66	81.2	96.99	99.5
NB	70-30	69.63	81.48	97.07	99.51
BN	30-70	73.56	86.63	98	99.84
BN	40-60	74.89	87.49	98.03	99.87
BN	50-50	75.31	87.99	98.12	99.87
BN	60-40	76.53	88.6	98.12	99.89
BN	70-30	76.92	88.96	98.18	99.89
DT	30-70	70.36	83.08	99.46	99.99
DT	40-60	71.18	83.99	99.5	99.99
DT	50-50	72.48	85.23	99.57	99.99
DT	60-40	72.04	84.73	99.59	99.99
DT	70-30	73.34	85.74	99.64	99.99

TABLE III: Prediction accuracy of different ML algorithms for the dataset 2 in different training/testing split scenarios and threshold (α)

Algorithm	**Split**	α**=10%**	α**=5%**	α**=1%**	α**=0%**
C4.5	30-70	84.17	86.78	90.98	91.17
C4.5	40-60	84.51	86.86	90.84	90.99
C4.5	50-50	77.87	86.08	89.86	90.23
C4.5	60-40	78.63	86.29	89.9	90.24
C4.5	70-30	78.93	86.34	89.78	90.1
NB	30-70	72.11	82.27	97.74	99.6
NB	40-60	73.07	82.81	98.08	99.61
NB	50-50	73.66	83.27	98.13	99.62
NB	60-40	74.35	83.66	98.36	99.62
NB	70-30	74.34	83.79	98.28	99.59
BN	30-70	80.09	90.22	98.2	99.93
BN	40-60	80.65	90.53	98.5	99.93
BN	50-50	81.12	90.96	98.64	99.92
BN	60-40	81.7	91.31	98.65	99.94
BN	70-30	81.67	91.68	98.73	99.94
DT	30-70	66.26	83.48	99.16	99.98
DT	40-60	66.5	84.39	99.12	99.98
DT	50-50	67.35	84.61	99.19	99.98
DT	60-40	67.6	83.55	99.21	99.98
DT	70-30	68.06	83.98	99.25	99.98

TABLE IV: Prediction accuracy of different ML algorithms for the dataset 3 in different training/testing split scenarios and threshold (α)

Algorithm	**Split**	α**=10%**	α**=5%**	α**=1%**	α**=0%**
C4.5	30-70	79.83	82.54	83.74	83.88
C4.5	40-60	81.36	83.75	84.9	85
C4.5	50-50	82.45	84.79	85.85	85.94
C4.5	60-40	83.61	85.81	86.82	86.91
C4.5	70-30	84.06	86.29	87.13	87.21
NB	30-70	72.27	82.87	95.37	99.6
NB	40-60	73.31	83.51	95.25	99.62
NB	50-50	73.78	84.3	95.43	99.65
NB	60-40	74.68	84.86	95.31	99.67
NB	70-30	75.1	85.48	95.4	99.68
BN	30-70	78.73	88.77	97.77	99.83
BN	40-60	79.77	89.56	97.99	99.88
BN	50-50	80.42	89.77	98.08	99.92
BN	60-40	81.51	90.26	98.28	99.93
BN	70-30	82.14	90.55	98.43	99.93
DT	30-70	67.69	82.76	99.76	99.99
DT	40-60	69.89	84.44	99.85	99.99
DT	50-50	71.25	85.12	99.82	99.99
DT	60-40	72.16	86.28	99.84	99.99
DT	70-30	73.65	87.43	99.85	99.99

IV. Results and Discussions

To evaluate the accuracy of the used ML algorithms, we use a machine learning tool called WEKA [27]. For training purpose, we used a public dataset from "*LongTail*" [28], an open source project by Marist College that records SSH brute force attacks on the honeypots deployed for this purpose. LongTail also performs a statistical analysis by recording IP addresses used, accounts, passwords, and account-password pairs, and analyze attack patterns for similarity and number of times used. As of now 32 different honeypots are used to collect the data. Table I presents a detailed description of data. There are three different datasets used. Datasets 1 contain the entire data available at [28], dataset 2 does not contain the attack records from China, and dataset 3 contains the record of attacks only from China. Using these three datasets, we trained our ML models and then tested them to identify potential vulnerable hosts. The datasets were split in 30/70, 40/60, 50/50, 60/40, and 70/30 ratio for training and testing purposes. We also altered the α threshold described in Section III.

Tables II, III, and IV show the prediction accuracy of different ML algorithms, for different data sets, training/testing split ratio and the α threshold. For a particular instance, the highest accuracy of 99.99% is obtained with *Decision Table* for dataset 1 and α 0% as shown in Table II. It is evident that the choice of α, the ML algorithm and the training/testing split ratio significantly affects the prediction accuracy. Below, we discuss the results of these tables in detail with respect to different parameters.

A. *Effect of α on prediction accuracy*

The results in Tables II, III, and IV highlight the fact that the α threshold drastically affects the prediction accuracy of the attacked host. The average prediction accuracy attained with different values of α is given in the Table V. It is evident that

TABLE V: Effect of α on average prediction accuracy

α (%)	Avg. Prediction Accuracy
0	97.06
1	95.78
5	85.74
10	75.59

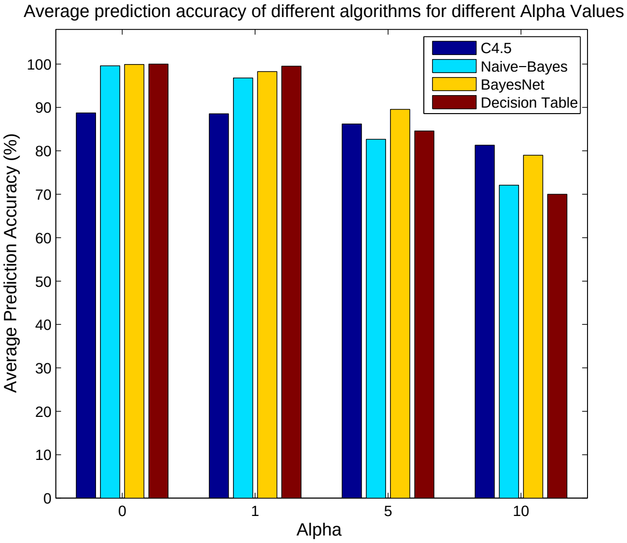

Fig. 2: The average prediction accuracy of different algorithms for different values of α

the increase in the value of α affects the prediction accuracy. The increase in α from 0-10% reduces the prediction accuracy by 21.47%. The result is significant as it shows that even a very small probability of attack on any particular host should be deemed significant and not ignored. Therefore, the attacker IP should be blocked and SDN controller rules must be modified to accommodate for better network security on vulnerable hosts.

B. Effect of the ML algorithm

Figure 2 provides an insight into the effect of ML algorithm on the prediction accuracy. Table VI provides the average prediction accuracy for the four different algorithms, and it can be seen the that highest average prediction accuracy of 91.68% is attained with BayesNet.

C. Effect of training/testing split ratio

The training/testing split ratio also affects the prediction accuracy as shown in Table VII. However, with the nature of the data obtained from the “LongTail” project [28], the change in the training/testing split ratio did not bring significant change in the prediction accuracy for a specific value of α.

TABLE VI: Average Prediction Accuracy for Different ML algorithms

Algorithms	Avg. Accuracy
C4.5	86.19
Nave-Bayes	87.78
BayesNet	91.68
Decision Table	88.52

TABLE VII: Effect of training/testing split ratio on average prediction accuracy for different α

Split Ratio	10	5	1	0
30/70	74.28	84.52	95.42	96.80
40/60	75.32	85.36	95.66	96.97
50/50	75.47	85.86	95.77	97.04
60/40	76.20	86.25	95.97	97.21
70/30	76.68	86.74	96.08	97.27

TABLE VIII: Effect of dataset on average prediction accuracy for different α

Dataset	10	5	1	0	Avg.
1	74.26	85.13	96.06	97.32	88.19
2	75.63	86.14	96.53	97.52	88.96
3	76.88	85.96	94.74	96.33	88.47

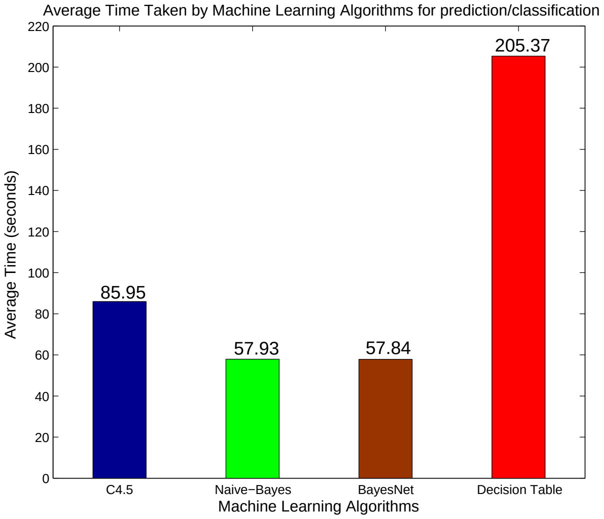

Fig. 3: Average time taken by different machine learning algorithms for predicting attacks on vulnerable hosts

D. Effect of the Dataset

The dataset also plays an important role in the prediction accuracy. The higher the variance in data, the higher will be the chances of false prediction. Since dataset 2 did not have the entries from the Chinese attackers, the variation in the dataset was much lesser than dataset 1 and dataset 3. This is why, as seen in Table VIII, the average prediction accuracy is higher for dataset 2 when compared with dataset 1 and dataset 2.

The results in Tables II, III, IV, V, VII and VIII show that ML algorithms can accurately predict the vulnerable host. Such accurate prediction can then be leveraged by SDN controller to block the potential attacker from attacking the network and help protect different hosts. Figure 3 compares different algorithms in terms of total time required for training and testing the machine learning model. From the Figure 3, it is also evident that BayesNet performs better than the other three algorithms used, in terms of time as well average prediction accuracy (see Table VI).

V. Conclusion

In this paper, we used machine learning algorithm to predict the vulnerable host in SDN network that is highly likely to be attacked. Leveraging the prediction output of machine learning algorithms, the security rules for SDN controller can be defined that will prevent malicious users from accessing the network. Experimental results showed that machine learning algorithms can help in defining security rules for SDN controller by accurately predicting the potential vulnerable host. An average prediction accuracy of 91.68% was achieved with Bayesian Network which indicates that out of the total 278,598 attacks, Bayesian network was able to accurately predict 254,834 attacks. Also, the decline in the prediction accuracy with the increase in the threshold α indicated that even a small probability of attack should not be ignored and security rules on the SDN controller must be accordingly modified to account for the potential threat.

References

[1] E. B. Eskca, O. Abuzaghleh, P. Joshi, S. Bondugula, T. Nakayama, and A. Sultana, “Software Defined Networks Security: An Analysis of Issues and Solutions,”

[2] J. Ashraf and S. Latif, “Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques,” in *Software Engineering Conference (NSEC), 2014 National*, pp. 55–60, Nov. 2014.

[3] A. Abdou, D. Barrera, and P. C. van Oorschot, “What Lies Beneath? Analyzing Automated SSH Bruteforce Attacks,”

[4] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, “Application-awareness in SDN,” in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 487–488, ACM, 2013.

[5] LongTail, “LongTail Log Analysis.” http://longtail.it.marist.edu/honey/. [Online; accessed 21-Mar-2016].

[6] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, “A survey of securing networks using software defined networking,” *Reliability, IEEE Transactions on*, vol. 64, no. 3, pp. 1086–1097, 2015.

[7] B. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, 2014.

[8] F. Hu, Q. Hao, and K. Bao, “A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation,” *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[9] A. Abdou, D. Barrera, and P. C. van Oorschot, “What lies beneath? analyzing automated ssh bruteforce attacks,”

[10] V. Sommer, “Anamoly Detection in SDN Control Plane,” Master’s thesis, Technical University of Munich, Munich, Germany, 2014.

[11] H. Kim and N. Feamster, “Improving network management with software defined networking,” *IEEE Communications Magazine*, vol. 51, pp. 114–119, Feb. 2013.

[12] D. Drutskoy, E. Keller, and J. Rexford, “Scalable network virtualization in software-defined networks,” *Internet Computing, IEEE*, vol. 17, no. 2, pp. 20–27, 2013.

[13] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hlzle, S. Stuart, and A. Vahdat, “B4: Experience with a Globally-deployed Software Defined Wan,” in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM ’13, (New York, NY, USA), pp. 3–14, ACM, 2013.

[14] G. M. Khan, S. Khan, and F. Ullah, “Short-term daily peak load forecasting using fast learning neural network,” in *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pp. 843–848, IEEE, 2011.

[15] K. Polat and S. Güneş, “A novel hybrid intelligent method based on c4. 5 decision tree classifier and one-against-all approach for multi-class classification problems,” *Expert Systems with Applications*, vol. 36, no. 2, pp. 1587–1592, 2009.

[16] M. A. Friedl and C. E. Brodley, “Decision tree classification of land cover from remotely sensed data,” *Remote sensing of environment*, vol. 61, no. 3, pp. 399–409, 1997.

[17] V. Muralidharan and V. Sugumaran, “A comparative study of naïve bayes classifier and bayes net classifier for fault diagnosis of monoblock centrifugal pump using wavelet analysis,” *Applied Soft Computing*, vol. 12, no. 8, pp. 2023–2029, 2012.

[18] J. Cheng and R. Greiner, “Comparing bayesian network classifiers,” in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 101–108, Morgan Kaufmann Publishers Inc., 1999.

[19] F. V. Jensen, *An introduction to Bayesian networks*, vol. 210. UCL press London, 1996.

[20] W.-t. Wu, W.-z. Jin, and P.-q. Lin, “Research on choice of travel mode model based on naive bayesian method,” in *Business Management and Electronic Information (BMEI), 2011 International Conference on*, vol. 2, pp. 439–444, IEEE, 2011.

[21] Y. Wang, J. Hodges, and B. Tang, “Classification of web documents using a naive bayes method,” in *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, pp. 560–564, IEEE, 2003.

[22] T. Mitchell, “Machine learning. 2003.”

[23] B. J. Cragun and H. J. Steudel, “A decision-table-based processor for checking completeness and consistency in rule-based expert systems,” *International Journal of Man-Machine Studies*, vol. 26, no. 5, pp. 633–648, 1987.

[24] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[25] R. Kohavi, “The power of decision tables,” in *Machine Learning: ECML-95*, pp. 174–189, Springer, 1995.

[26] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, “Supervised machine learning: A review of classification techniques,” 2007.

[27] WEKA, “Machine Learning Group: University of Waikato.” http://www.cs.waikato.ac.nz/ml/weka/downloading.html. [Online; accessed 21-Mar-2016].

[28] LongTail, “LongTail Log Analysis Dashboard.” http://longtail.it.marist.edu/honey/dashboard.shtml. [Online; accessed 22-April-2016].