Enabling Dynamic Network Access Control with Anomaly-based IDS and SDN

Hongda Li*
Clemson University
hongdal@clemson.edu

Feng Wei*
Clemson University
wei8@clemson.edu

Hongxin Hu
Clemson University
hongxih@clemson.edu

ABSTRACT

In the Software Defined Networking (SDN) and Network Function Virtualization (NFV) era, it is critical to enable dynamic network access control. Traditionally, network access control policies are statically predefined as router entries or firewall rules. SDN enables more flexibility by re-actively installing flow rules into the switches to achieve dynamic network access control. However, SDN is limited in capturing network anomalies, which are usually important signs of security threats. In this paper, we propose to employ anomaly-based Intrusion Detection System (IDS) to capture network anomalies and generate SDN flow rules to enable dynamic network access control. We gain the knowledge of network anomalies from anomaly-based IDS by training an interpretable model to explain its outcome. Based on the explanation, we derive access control policies. We demonstrate the feasibility of our approach by explaining the outcome of an anomaly-based IDS built upon a Recurrent Neural Network (RNN) and generating SDN flow rules based on our explanation.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; • **Networks → Programmable networks**.

KEYWORDS

IDS; SDN; Dynamic Access Control

ACM Reference Format:
Hongda Li, Feng Wei, and Hongxin Hu. 2019. Enabling Dynamic Network Access Control with Anomaly-based IDS and SDN. In *ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization (SDN-NFVSec '19), March 27, 2019, Richardson, TX, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3309194.3309199

1 INTRODUCTION

The emerging SDN and NFV network paradigms introduce significant flexibility to the network, which calls for a more flexible way to enforce access control policies on the network (a.k.a., dynamic network access control). A significant body of work has made efforts to enable dynamic access control using SDN, such as SDN firewall [15], virtual firewall [11], and general network control framework [8]. However, prior work assumes the access control policies are defined by the network administrators based on their knowledge about the network. Those access control policies fail to consider any emerging network anomalies – network activities that have not been seen before. Those anomalies are especially dangerous because they are likely to associate with unknown vulnerabilities in the network or zero-day security threats.

Anomaly-based IDS is known to be powerful because of its ability to uncover novel security threats. However, instead of specifically pinpointing what threats are detected, anomaly-based IDS only outputs the level of deviation, which is obscure and has little help to generate access control policies. To fully utilize the outcome of anomaly-based IDS for dynamic network access control, we must bridge the semantic gap between the outcome of anomaly-based IDS and the access control policies we want to generate.

The heart of anomaly-based IDS is to model the normal pattern of a network. Those models are usually implemented via machine learning approaches [20, 21, 23]. Fortunately, there exists a large body of recent work that aims to explain machine learning models [7, 10, 13, 14, 17]. There are two major mechanisms that have been used to explain machine learning models. The *whitebox* mechanism assumes the internal characteristics of a model are available and augments the model with the ability to produce explanations of each prediction. The *blackbox* mechanism treats the model as a black box and explains the outcome through approximating the decision boundary around the outcome. By explaining a machine learning model, one can get better understanding of the whole model (global explanation) or gain knowledge about the reason behind the decision made by a model on a specific input (local explanation). Machine learning model explanation seems to be a promising solution that can help us to generate network access control policies from an anomaly-based IDS.

In this paper, we consider anomaly-based IDS as a blackbox model that can predict whether an input is a normal instance or a deviation. Given an input, we explain the outcome of this blackbox model if the outcome indicates a deviation. Then a network access control policy is generated according to the explanation of that outcome. To explain the outcome of a blackbox model, we generate the samples that are "close" to the given input. Then we use regression model with fused lasso to approximate the local decision boundary of the blackbox model around the given input as described in LEMNA [14]. The explanation consists of the scores that indicate how significantly each individual feature contributes to the prediction against the given input. The features are then sorted based on their importance. We select top-k features to aid generating network access control policies, where k is a hyper-parameter.

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SDN-NFVSec '19, March 27, 2019, Richardson, TX, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6179-8/19/03…$15.00
https://doi.org/10.1145/3309194.3309199

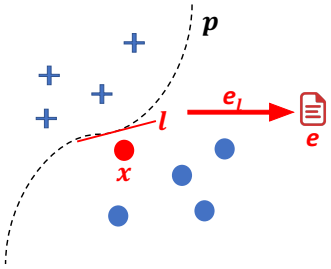

Figure 1: To obtain an explanation e of the outcome of a predictive model p against a given input x, a model l is developed to imitate the local prediction of p around x and an explanation logic e_l is used to reason over l and x.

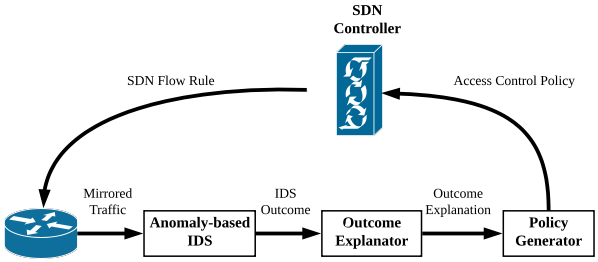

Figure 2: A High-level view of our overall approach that achieves dynamic nework access control.

The contributions of this paper are as follows.

- We investigate the approach to explaining the outcome of anomaly-based IDS through local approximation of decision boundary of the anomaly-based intrusion detection model.
- We demonstrate the feasibility of our approach with a case study through generating a network access control policy based on the outcome explanation of an anomaly-based IDS.

The rest of the paper is organized as follows. In Section 2, we provide the background and describe the details of our technical approach. In Section 3, a case study is presented to demonstrate the feasibility of our proposed approach. we conclude and discuss our future work in Section 4.

2 METHODOLOGY

2.1 Background

Formally, a predictive model for anomaly-based IDS is a function $p : X^{(m)} \rightarrow Y$, which takes an instance $x \in X^m$ with m features as input and outputs a score $y \in Y$ indicating the deviation between the input instance and a normal instance. Conventionally, $p(x) = y$ denotes that y is an outcome of the prediction of p corresponding to the input x. The task of explaining an outcome of a predictive model can be defined as follows. Given an input x and a predictive model p, we need to find an interpretable model l that imitates the prediction of p locally against x. Then, an explanation e is obtained through some explanation logic e_l reasoning over x and l. Figure 1 illustrates each concept in the aforementioned process.

There are two key components in the above process: the interpretable model and the explanation logic. There is a set of models, such as *decision tree* [12, 16] and *linear models* [14, 17], believed to be easily understandable and interpretable for humans. For explanation logic, there exist three major techniques: *saliency map* [22, 24], *decision rules* [18], and *feature importance* [14, 17].

In this work, we employ linear models to approximate the local decision boundary of the original predictive and obtain the explanation through feature importance.

2.2 Challenge and Technical Overview

Despite SDN introduces great programmability to the network, it is still limited in considering network anomalies when controlling the network. This limitation is caused fundamentally by the fact that the SDN controller is not designed to analyze the network traffic in depth like IDS.

To overcome this shortcoming, an anomaly-based IDS is introduced to operate on the data plane to directly analyze the traffic in depth and report any network anomalies to aid the SDN controller controlling the network. Deriving access control policies from the results of a signature-based IDS [1, 2] could be straight forward since the signatures are working in an *if-match-then-action* manner [19], which can be easily mapped to an access control policy. However, there exist several limitations of signature-based IDS, including being unable to detect zero-day security threats and suffering from signature base explosion.

In the following, we achieve dynamic network access control through the combination of SDN and anomaly-based IDS. Figure 2 depicts the high-level idea of our overall approach. We mirror a copy of network traffic from the SDN switch to an *anomaly-based IDS*. If the outcome of the anomaly-based IDS indicates an anomaly, it is sent to the *outcome explanator* to get explained. The explanation of the outcome is then processed by the *policy generator* to generate the access control policy. The generated access control policy can be sent to the SDN controller via a network control framework [8]. The SDN controller finally installs SDN flow rules into the SDN switch according to the access control policy.

2.3 Technical Details

Existing work has already demonstrated how to develop anomaly-based IDS using Deep Neural Network (DNN) [20, 21, 23], and a network control framework has been presented in [8] to allow IDS communicate with the SDN controller. Therefore, our focus is the outcome explanation and policy generation.

Anomaly-based IDS Outcome Explanation. As is shown in Figure 1, outcome explanation consists of two major steps: *i)* training a model to approximate the local decision boundary of the target predictive model; and *ii)* reasoning on the trained model and the given input based on some explanation logic.

To approximate the local decision boundary of the target predictive model, we first synthesize a set of data samples around x as described in LEMNA [14] and utilize the target predictive model to predict the label for each synthetic data sample. Then we can use those synthetic data samples to train a linear regression model l defined as:

$$l(\mathbf{x}) = \boldsymbol{\alpha}\mathbf{x} + \epsilon \quad (1)$$

where ϵ is the error term, and $\mathbf{x}$ is a synthetic data sample $(x_1, x_2, ..., x_M)^T$ containing M features. The vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_M)$ contains the coefficients of the linear model. However, a standard linear regression model cannot approximate a local decision boundary correctly for an anomaly-based IDS because the anomaly-based IDS makes decision considering dependencies across multiple features. We introduce the fused lasso, a penalty term that can capture dependencies between features. The loss function L of the linear regression model can be defined with fused lasso as:

$$L(l(\mathbf{x}), y) = \sum_{i=1}^{N} \parallel \boldsymbol{\alpha}\mathbf{x_i} - y_i \parallel \tag{2}$$

$$\text{subject to} \quad \sum_{j=2}^{M} \parallel \alpha_j - \alpha_{j-1} \parallel \leq S \tag{3}$$

where $\mathbf{x_i}$ is the i-th synthetic data sample represented as a feature vector. N is the number of synthetic data samples. y_i is the label of $\mathbf{x_i}$. $\parallel \cdot \parallel$ is the L2-norm distance between the model prediction and the true label. S is a hyper-parameter controlling the threshold of dissimilarity of the coefficients assigned to adjacent features. Minimizing Equation (2) under the constrain of Equation (3), we get the linear regression model l that approximate the decision boundary of anomaly-based IDS around the given input x.

After we have developed a linear regression model that approximates the local decision boundary of anomaly-based IDS, we can use the coefficients of the linear model to derive the explanation. Assigning larger coefficient to a feature when the linear regression model is trained means this feature contributes greater to the prediction of the linear model, thus has a greater impact to the decision made by anomaly-based IDS. Therefore, the importance of each feature provides an explanation to the outcome of anomaly-based IDS corresponding to a specific input.

Access Control Policy Generation. A network access control policy consists of two components: *filters* and *actions*. It can be defined as <*filters, actions*>. The *filters* are responsible for selecting which network entity, such as host, flow, connection, and packet, the access control policy is applied to. The *actions* define the actions that will be taken against the entity selected by the *filters*. For simplicity, let's assume the *actions* include *allow* or *deny*. Note that one can easily extend the *actions* by adding more action types.

To generate a network access control policy, one needs to fill the two components. The *filters* are filled with the values of the corresponding fields of the input instance, such as *source IP*, *destination IP*, *source port*, and *destination port*. In addition, some fields of the *filters* are obtained according to the importance of the fields in the input instance. For example, if the important features indicate that the SYN flag is a major cause of the anomaly, then the SYN flag is set in the *filters* of the access control policy. If some features that are not so important to the outcome, we may remove those fields from the *filters*. For example, if the explanation believes that the protocol type has low importance to the anomaly, we will remove the protocol type, such as TCP, UDP, and ICMP, from the *filters* using a wildcard instead. The *actions* indicate what actions will be taken to the network traffic selected by the *filters*.

No.	Features	Types	No.	Features	Types
1	duration	cont.	22	is_guest_login	cont.
2	protocol_type	symb.	23	count	cont.
3	service	symb.	24	srv_count	cont.
4	flag	symb.	25	serror_rate	cont.
5	src_bytes	cont.	26	srv_serror_rate	cont.
6	dst_bytes	cont.	27	rerror_rate	cont.
7	land	cont.	28	srv_rerror_rate	cont.
8	wrong_fragment	cont.	29	same_srv_rate	cont.
9	urgent	cont.	30	diff_srv_rate	cont.
10	hot	cont.	31	srv_diff_host_rate	cont.
11	num_failed_logins	cont.	32	dst_host_count	cont.
12	logged_in	cont.	33	dst_host_srv_count	cont.
13	num_compromised	cont.	34	dst_host_same_srv_count	cont.
14	root_shell	cont.	35	dst_host_diff_srv_rate	cont.
15	su_attempted	cont.	36	dst_host_same_src_port_rate	cont.
16	num_root	cont.	37	dst_host_srv_diff_host_rate	cont.
17	num_file_creations	cont.	38	dst_host_serror_rate	cont.
18	num_shells	cont.	39	dst_host_srv_serror_rate	cont.
19	num_access_files	cont.	40	dst_host_rerror_rate	cont.
20	num_outbound_cmds	cont.	41	dst_host_srv_rerror_rate	cont.
21	is_host_login	cont.			

Table 1: Features of NSL-KDD dataset

3 CASE STUDY

3.1 Anomaly-based IDS

We follow [23] to build an anomaly-based IDS with RNN and use *NSL-KDD* [5] as the benchmark dataset. Compared with the original *KDD* dataset [3], *NSL-KDD* has solved two major issues. The first issue is the huge number of redundant records, which make the classifier favor more frequent records and ignore the infrequent records. The second issue is that *NSL-KDD* labels the records in *KDD* dataset with different difficulty levels. For each record there are 41 features as shown in Table 1. We use *one-hot encoding* to convert the three symbolic features into numeric form, because our RNN model accepts a numeric vector as its input. After the preprocessing we convert 41 dimensional records into 122 dimensional vectors.

We use *Keras* [4] to build our *RNN* model, with *TensorFlow* [6] as backend. We employ *SimpleRNN* to construct the hidden layers and add one *Normalization* layer to the model. After training and testing we get an anomaly-based IDS with sufficient detection accuracy.

3.2 Outcome Explanation

We follow [14] to construct an interpretable model and obtain an explanation to illustrate why each record is classified as normal or abnormal.

In our experiment we use *Neptune* attack [9] as a test case. This attack is also known as a half opened *TCP SYN* attack. The purpose of this attack is to exhaust the *TCP* server resources and reject any new connections from normal *TCP* clients. The *Neptune* records used as given inputs in our experiment are shown below.

Record1: $(0, tcp, private, S0, ..., 255, 20, 0.08, 0.07, 0, 0, 1, 1, 0, 0)$

Record2: $(0, tcp, imap4, REJ, ..., 255, 17, 0.07, 0.07, 0, 0, 0, 0, 1, 1)$

There are two major steps in our experiment. First we generate a set of record samples around the given input and use those records and their classification labels to train an interpretable model. Then the interpretable model assigns different important scores to all 41 features according to their coefficients. In Table 2, features with different important scores are marked with different colors. We choose the top-4 most important features as our explanation. The

Feature	dur.	proto.	service	flag	...	dsthost_serror_rate*	dsthost_srv_serror_rate*	dsthost_rerror_rate*	dsthost_srv_rerror_rate*
Record1	0	tcp	private	S0	...	1	1	0	0
Record2	0	tcp	imap4	REJ	...	0	0	1	1

Table 2: Explanation for Neptune SYN-FLOOD attack. The outcome explanation marks the most important feature as red, followed by orange, yellow, lime, cyan. The top-4 features are marked with*.

(dst_host_serror_rate) and *(dst_host_srv_serror_rate)* features represent the percentage of connections that have SYN errors. The *(dst_host_rerror_rate)* and *(dst_host_srv_rerror_rate)* features represent the percentage of connections that have REJECT errors. In reality, *Neptune* is a typical *SYN FLOOD* attack, which is usually fingerprinted by both SYN and REJECT errors. This knowledge is identical to our explanation, thus validates that our anomaly-based IDS is trustworthy and the explanation is correct.

3.3 Policy Generation

From the above explanation we can see that the key features of those abnormal records are highly related to the *SYN* field. To prevent this type of attacks we can generate a policy with deny action for those instances with *SYN* features. Each record in the *NSL-KDD* dataset represents one connection. Each connection can be identified by the source and destination IPs, ports and protocol types [1]. Below is an access control policy generated according to our explanation that drops the network packets related to the *Neptune* attack.

<filters=(src_ip=192.168.1.2, dst_ip=192.168.1.3, ip_proto=6, tcp_flags=0x02), actions=(drop)>

The source IP, destination IP, and the protocol type can be derived from the abnormal connection record. Worth noting, the *tcp_flags* field in the *filters* is set to 0x02 (*SYN* flag set) because the features included in the explanation are all related to *SYN*. This policy will be sent to the SDN controller, which then generates and installs corresponding SDN flow rules into the SDN switch to achieve dynamic network access control.

4 CONCLUSION AND FUTURE WORK

This paper introduces a method to explain the outcome of anomaly-based IDS via an interpretable model and generate the network access control policies based on the explanation. In our case study, we show that the proposed method produces trustworthy outcome and is useful to generate policies for dynamic network access control.

In our future work, we will investigate better explanation approaches to handle the dependency among multiple records to improve our explanation accuracy. In addition, the policy generation process should be formalized to enable full automation. We will also test on real-world network traffic considering different attack types.

ACKNOWLEDGMENTS

This work was partially supported by grants from National Science Foundation (NSF-OAC-1642143, NSF-CNS-1700499, and NSF-DGE-1723663).

[1] *NSL-KDD* dataset does not include the IPs in each record, but in practice the IDS can easily capture the IP information and embed it into the outcome.

REFERENCES

[1] 2018. Snort Network Intrusion Detection & Prevention System. https://snort.org/.
[2] 2018. Suricata IDS. https://suricata-ids.org/.
[3] 2019. KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[4] 2019. Keras: The Python Deep Learning library. https://keras.io/.
[5] 2019. NSL-KDD dataset. https://www.unb.ca/cic/datasets/nsl.html.
[6] 2019. TensorFlow. https://www.tensorflow.org/.
[7] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160.
[8] Johanna Amann and Robin Sommer. 2015. Providing Dynamic Control to Passive Network Security Monitoring. In *Proceedings of the 18th International Symposium on Research in Attacks, Intrusions, and Defenses-Volume 9404*. Springer-Verlag New York, Inc., 133–152.
[9] Rocky KC Chang. 2002. Defending against flooding-based distributed denial-of-service attacks: a tutorial. *IEEE communications magazine* 40, 10 (2002), 42–51.
[10] Zhiyong Cheng, Xiaojun Chang, Lei Zhu, Rose C Kanjirathinkal, and Mohan Kankanhalli. 2019. MMALFM: Explainable recommendation by leveraging reviews and images. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 16.
[11] Juan Deng and Hongda Li. 2017. On the Safety and Efficiency of Virtual Firewall Elasticity Control. In *24th Network and Distributed System Security Symposium (NDSS 2017)*.
[12] Alex A Freitas. 2014. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter* 15, 1 (2014), 1–10.
[13] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. 2018. Ai 2: Safety and robustness certification of neural networks with abstract interpretation. In *Security and Privacy (SP), 2018 IEEE Symposium on*.
[14] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. 2018. Lemna: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 364–379.
[15] Hongxin Hu, Wonkyu Han, Gail-Joon Ahn, and Ziming Zhao. 2014. FLOWGUARD: building robust firewalls for software-defined networks. In *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 97–102.
[16] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. 2011. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems* 51, 1 (2011), 141–154.
[17] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 1135–1144.
[18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*.
[19] Martin Roesch and Chris Green. 2016. Snort Users Manual 2.9. 8.2.
[20] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. 2018. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2, 1 (2018), 41–50.
[21] Tuan A Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. 2018. Deep recurrent neural network for intrusion detection in sdn-based networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 202–206.
[22] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*. 2048–2057.
[23] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. 2017. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* 5 (2017), 21954–21961.
[24] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2921–2929.