Research and development automatically generate detection rules for IDS based on machine learning technology

Le Hai Viet, Nguyen Huy Trung, Tran Duc Thang

***Abstract*—Nowadays, there have been many signature-based intrusion detection systems deployed and widely used. These systems are capable of detecting known attacks with low false alarm rates, fast detection times, and little system resource requirements. However, these systems are less effective against new attacks that are not included in the ruleset. In addition, recent studies provide a new approach to the problem of detecting unknown types of network attacks based on machine learning and deep learning. However, this new approach requires a lot of resources, processing time and has a high false alarm rate. Therefore, it is necessary to find a solution that combines the advantages of the two approaches above in the problem of detecting network attacks. In this paper, the authors present a method to automatically generate network attack detection rules for the IDS system based on the results of training machine learning models. Through testing, the author proves that the system that automatically generates network attack detection rules for IDS based on machine learning meets the requirements of increasing the ability to detect new types of attacks, ensuring automatic effective updates of new signs of network attacks.**

***Tóm tắt*—Ngày nay, đã có nhiều hệ thống phát hiện xâm nhập dựa trên chữ ký được triển khai và sử dụng rộng rãi. Các hệ thống này có khả năng phát hiện các cuộc tấn công đã biết với tỷ lệ báo động giả thấp, thời gian phát hiện nhanh và yêu cầu ít tài nguyên hệ thống. Tuy nhiên, các hệ thống này kém hiệu quả khi chống lại các cuộc tấn công mới không có trong tập luật. Các nghiên cứu gần đây cung cấp một cách tiếp cận mới cho vấn đề phát hiện các kiểu tấn công mạng mới dựa trên học máy và học sâu. Tuy nhiên, cách tiếp cận này đòi hỏi nhiều tài nguyên, thời gian xử lý. Vì vậy, cần tìm ra giải pháp kết hợp ưu điểm của hai cách tiếp cận trên trong bài toán phát hiện tấn công mạng. Trong bài báo này, nhóm tác giả trình bày phương pháp tự động sinh luật phát hiện tấn công mạng cho hệ thống phát hiện xâm nhập dựa trên kết quả huấn luyện mô hình học máy. Qua thử nghiệm, tác giả chứng minh rằng phương pháp này đáp ứng yêu cầu tăng khả năng phát hiện chính xác các kiểu tấn công mới, đảm bảo tự động cập nhật hiệu quả các dấu hiệu tấn công mạng mới vào tập luật.**

Keywords*—Machine Learning; Network Security; Intrusion Detection System.

Từ khóa*—Học máy; An ninh mạng; Hệ thống phát hiện xâm nhập.

I. INTRODUCTION

Nowadays, detecting network attacks is key in ensuring the information security of businesses and organizations. Intrusion Detection Systems (IDS) such as SolarWind, Snort, Suricata, Qradar,... have received much attention from researchers and IT technicians. Along with IDS, the application of machine learning algorithms to detect network attacks has also been studied and applied effectively in recent years. Thereby helping to classify learned patterns, predicting new attack patterns well. This promises to bring an effective approach to the attack detection problem. There are many papers and research projects in this field. We also surveyed IDS development studies applying machine learning such as [1]–[4]. In addition, the authors Jayveer Singh and Manisha J. Nene have evaluated the pros and cons of some machine learning algorithms in detecting network attacks [5]. However, studies on the combination of network attack detection systems and the results of classification of machine learning models are few, have many

This manuscript is received on November 20, 2021. It is commented on November 25, 2021 and accepted on November 26, 2021 by the first reviewer. It is commented on November 25, 2021 and accepted on November 30, 2021 by the second reviewer.

limitations in terms of execution time, are up-to-date, and require many system resources.

Therefore, we propose a way to combine the results of machine learning model training at a server. We converted these training results into rules used in rule-based intrusion detection systems. The main contributions of this research effort are to provide a method that automatically generates detection rules for IDS based on machine learning technology.

The rest of this paper is structured as follows. Section II illustrates background and related work. Then, Section III provides the proposed method. Section IV implements and evaluates the proposed method. Section V discusses the result and provides the futures plan of this work. Finally, Section VI concludes this work.

II. Background and related work

This section provides relevant background information on Intrusion Detection Systems, and related work on detection network attacks based on machine learning.

A. An Intrusion Detection System

An Intrusion Detection System (IDS) is a software or hardware system that automates the process of monitoring events occurring in a computer system or network, analyzing them to find and warn of signs of a security problem [6].

In recent years, researchers have used many techniques to design intrusion detection systems. However, there are some problems with current intrusion detection systems such as [1]:

- High false-positive rate: Many false alarms and inaccurate network attack identification. By lowering the thresholds to reduce false alarms, it increases the number of network attacks that go undetected as false negatives.
- High rate of false negatives: Some network intrusions remain undetected in some systems which means that IDS cannot detect all these intrusions.
- Lack of efficiency: IDS is often required to evaluate events in real-time. This requirement is difficult to meet when a system is faced with a very large number of events that often occur in present networks. As a result, host-based IDSs often slow down the system and network-based IDSs drop network packets that they don't have time to process.

To better understand the above statement, we will analyze the advantages and disadvantages of intrusion detection systems. From there, we propose a solution to address these problems. Intrusion detection systems can be classified based on monitoring data sources or based on intrusion detection methods. According to the first classification, intrusion detection systems are divided into two types: Network-based IDS (NIDS) and Host-based IDS (HIDS).

Network-based IDS: NIDS is a stand-alone system that identifies unauthorized access by examining information flows on the network. NIDS accesses the network flow by connecting to the hub, and switches configured with port mirroring or a network tap to capture packets, analyze the packet's contents. From the analysis results, NIDS generates alarm of detection attacks.

Host-based IDS: In contrast to NIDS, HIDS is usually a piece of software that runs on workstations (hosts) to monitor all activities on these terminals. This system analyzes the information obtained within the system, so it provides a comprehensive analysis mechanism for activities and accurately detects attack components.

According to the second classification, intrusion detection systems are divided into two types: Signature-based IDS and Anomaly-based IDS.

Signature-based IDS: Signature-based IDS works based on signatures of network attacks. Those signatures (signature patterns) can be information about known suspicious connections. The system will model the signatures of known attacks and by comparing the information of incoming packets with these signals to detect suspicious activities and alert the system. The advantage of this system is that it is very effective in detecting known attacks with low false alarms rates. However, the main disadvantage of the system is that it can only detect known attacks, the need to regularly update the characteristics (signature samples) of new attacks. Besides, the detection time increases as the ruleset grows [2].

Anomaly-based IDS: The idea of this approach comes from the assumption that "the signature of attacks is different from that of the considered normal network states". At that time, the detection will take place through two stages: the training phase (training phase) and the detection phase (detection phase). At the training phase, a profile of normal operations (standard parameters) is built. Then, at the detection phase, it will match observations (packets) with records, thereby identifying anomalies. The advantage of this system is that it is effective in detecting unknown risks. However, due to the complexity in the machine learning algorithms applied to detect anomalous data, these systems require a lot of resources and processing time [2]. Currently, this approach is attracting a lot of attention from researchers to optimize this operating model.

As mentioned above, Signature-based IDS can accurately and quickly detect forms of network attacks. However, this IDS is strongly dependent on the quality of regularly updated signature samples. However, most of these signature samples (which often exist as rulesets) are written based on the experience and personal analysis of experts' security. Therefore, an optimal solution is needed to apply the self-learning ability of the computer to automatically generate rule sets for Signature-based IDS to reduce the forecasting time and processing resources while ensuring the accuracy in detecting network attacks.

B. Machine learning

In recent years, artificial intelligence, and more specifically, machine learning (ML) has emerged as evidence of the fourth industrial revolution. This field trends all over the world, every year there are thousands of scientific articles on this field. A series of applications using ML are born in all areas of life, from computer science to less related disciplines such as physics, chemistry, medicine, economics, and politics. Machine learning is a subset of artificial intelligence that gives computers the ability to learn without being explicitly programmed. Machine learning focuses on developing computer programs that can change when exposed to new data [7]. Machine learning is a key component of the burgeoning field of data science. Through the use of statistical methods, algorithms and computers can make classifications or predictions based on learned data, contributing to motivating developers and data analysts to make decisions more exactly.

Currently, many machine learning algorithms are researched, developed, and improved. However, based on the learning style, machine learning algorithms can be divided into the following main groups: Supervised Learning, Unsupervised Learning, Semi-Supervised Learning, Reinforcement Learning.

C. Related work using machine learning in detection network attack

Nowadays, there are many studies applying machine learning algorithms in the problem of detecting network attacks, specifically:

Doshi [8] proposes a NIDS model to detect DDoS attacks in network data streams connecting IoT devices. By using a gateway router or other intermediate network devices (middleboxes), Doshi can detect the source of DDoS attacks from a local network using machine learning algorithms (including KN, LSVM, DT, RF, NN) with more than 98% accuracy. Doshi recommends using network flow characteristics such as Packet Size, Interval between Packets, Transmission Protocol, Network Bandwidth, Destination IP Address.

Indre [9] presents a solution to combine features extracted from packets of different hosts with similar proportions of REJ, SYN, ACK flags, and connection state. Indre focuses on data analysis from URI and RESTful methods. These proposed features are based on the features of the KDD99 dataset [30]. The test results show that the proposed feature set is used in combination with the Passive-Aggressive Classifier [10] achieved an accuracy of 98.4%. However, the KDD99 dataset mainly collects data about computer network attacks since 1999 [11]. At this point, the IoT Botnet has not yet emerged. Therefore, the identification of IoT Botnet based on the characteristics of the KDD99 dataset is incomplete and inaccurate.

Alrashdi [3] proposed to use the UNSW-NB15 [12] dataset to build a more efficient IoT Botnet detection NIDS in smart cities, replacing the outdated KDD99 dataset. Alrashdi uses the Random Forest algorithm with 12 features selected from the UNSWNB15 dataset to achieve an accuracy of 99.34%. These selected features include: srcip, dstip, dur, dsport, ctdst-src-ltm, ct-rsv-dst, ct-dst-ltm, ct-srcltm, ct-src-dport-ltm, dbytes, proto , is-ftp-login.

Breitenbacher [4] proposed HIDS as HadesIoT IDS, which requires low power for IoT devices. Hades-IoT IDS uses system calls data to detect malicious behavior of VP-Nfilter and IoTReaper malware on 7 IoT devices, including routers and IP cameras. The accuracy of Hades-IoT IDS is 100% according to the author, but this test result is only applied on limited data sets (2 types of IoT devices with 7 device models and two malicious code samples). little to prove the above result). Features used in Hades-IoT IDs include names, addresses, and parameters of system calls.

Prokofiev [13] presents a NIDS model using Logistic Regression for IoT devices with an accuracy of 97.3%. To create a logistic regression model, the network traffic characteristics were selected including destination port, source port, the total number of packets requested, average time interval between requests, number of requests on ports other, average packet size, delta value for packet size, average packet entropy.

Wu [14] uses Sandbox IoTBox [15] to execute IoT Botnet templates and collects more than 3 million malicious session log files. These logs store shell commands that the attacker sends to the Bot. Wu proposed a model that learns the sequential sequences of these shell commands to detect IoT Botnet with about 90% accuracy. The limitation of Wu's proposed model is that the use of training datasets with large deviations (300 log Gayfgt, 51 log nttpd, 2430 log Zorro) leads to inaccurate prediction models.

In summary, the solutions for applying machine learning in network attack detection surveyed above require a lot of resources, processing time and have a high false alarm rate. Therefore, a solution that combines advantages is needed points of rule-based IDS systems and machine learning-based network attack detection models. In the next section, the author presents a proposed model to address the above problem.

III. PROPOSED METHOD

A. Overview

The proposed method is illustrated in Fig. 1, including the following four main steps:

- Preprocess data (1): the system receives and preprocesses input data sets (datasets) containing information about new network attack patterns before training machine learning models.
- Train machine learning model (2): The system uses machine learning algorithms commonly used in network attack detection, then trains those algorithms with the required dataset.
- Extract weights of the trained model (3): The weights of the trained model are extracted to generate the rules for IDS.
- Automatically generate rules for IDS (4): The algorithm automatically builds matching rules for IDS according to the weights extracted from the trained machine learning model.

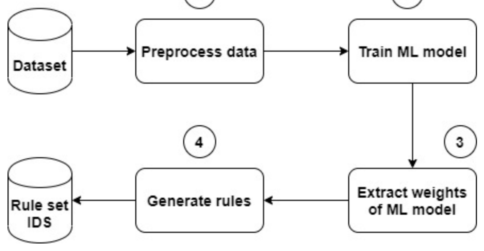


Fig. 1. Overview of the proposed method.

B. Preprocessing data

To preprocess the input data as network data streams, we use the CICFlowMeter tool [16] to extract the desired features. CICFlowMeter is a network traffic flow generator which has been written in Java and offers more flexibility in terms of choosing the features you want to calculate, adding new ones, and having a better control of the duration of the flow timeout. It generates Bidirectional Flows (Biflow), where the first packet determines the forward (source to destination) and backward (destination to source) directions, hence the 83 statistical features such as Duration, Number of packets, Number of bytes, Length of packets, etc. are also calculated separately in the forward and reverse direction.

The output of the application is in CSV file format with six columns labeled for each flow, namely FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol with more than 80 network traffic features. The CICFlowMeter-V3 can extract more than 80 features which are listed in the Fig. 2.

C. Training machine learning model

Many popular machine learning algorithms have been applied in network attack detection as evaluated above. However, within the scope of this paper, we will focus on learning and using the decision tree algorithm (DT) in building experimental models and evaluating efficiency. The decision tree algorithm was chosen for the following reasons:

- The output of the algorithm is the thresholds of each classification criterion, which are suitable for the forms of network attack detection rules of IDS.
- Decision tree architecture has a close relationship between nodes, easy to convert into logical propositions corresponding to rules for IDS.

The decision tree is one of the forms of visual data description, easy to understand for users. The structure of a decision tree consists of nodes and branches. The bottom node is called the leaf node. In the data classification model, the leaf node is the values of the class labels (referred to as labels for short). The nodes other than the leaf node are called child nodes. These are also attributes of the data set. These attributes must be different from the class attributes. Each branch of the tree that originates from a certain node corresponds to a comparison based on the domain of that node. The first node is called the root node of the tree.

With the criterion of building decision trees becoming increasingly simple, with high classification accuracy, low cost, and scalability, many authors have produced more and more optimal algorithms. Some popular algorithms are used a lot today such as ID3 (Interactive Dichotomizer 3), J48, 4.5, CART (Classification And Regression Tree), CLS (Concept learning System). Choosing which algorithm to have high classification efficiency depends on a lot of factors, of which the data structure greatly affects the results of the algorithms. For example, the ID3 and CART algorithms give very high classification efficiency for quantitative values while algorithms like J48, C4.5 are more efficient for qualitative values.

D. Extracting weights of the trained model

The training result of the machine learning model based on the DT algorithm is a binary tree with corresponding weights at the nodes. The weight values corresponding to each of these nodes will be extracted from the model training results and stored in a file for easy rule generation for IDS.

E. Automatically generation rules for IDS

The automatically generating network attack detection rules for IDS is an important function of the method and the main contribution of this paper. From the training results of the model, we will extract features that are decisive in detecting network attacks on the system. The automatic rule generation function will automatically take those features, write rules according to the IDS rule structure, and automatically update the IDS ruleset database that has been created and stored in the system.

IV. EVALUATION AND EXPERIMENTAL RESULTS

A. Dataset

In this paper, we train the machine learning model with the CSE-CIC-IDS-2018 dataset [17]. This is a collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). A description of this dataset is presented in Table I.

TABLE I. DESCRIPTION OF DATASET

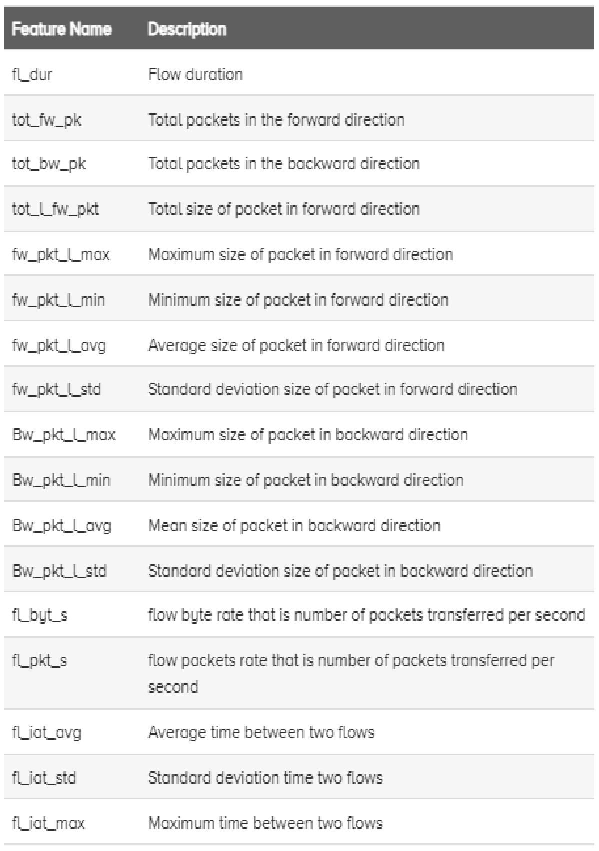
Attack	Duration	Attacker	Victim
Bruteforce attack	One day	Kali linux	Ubuntu 16.4 (Web Server)
DoS attack	One day	Kali linux	Ubuntu 16.4 (Apache)
DoS attack	One day	Kali linux	Ubuntu 12.04 (Open SSL)
Web attack	Two days	Kali linux	Ubuntu 16.4 (Web Server)
Infiltration attack	Two days	Kali linux	Windows Vista and Macintosh
Botnet attack	One day	Kali linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)
DDoS + PortScan	Two days	Kali linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)

B. Evaluation environment

The system was tested on a computer using Intel Core i5 4200U processor 1.8GHz up to 3.1GHz 4MB L2 cache, 8GB RAM with libraries that support machine learning processing based on Python language such as Numpy. , Pandas, Sklearn, Matplotlib in Ubuntu 20.04 LTS operating system environment. In addition, the test system also installs the IDS Snort software to test the effectiveness of the automatically generated rule in detecting network attacks.

C. Experimental results

Input data preprocessing: With the CSE-CIC-2018 Dataset, 80 network flow features are extracted with the help of the CICFlowMeter-V3 tool. Fig. 2 depicts the features extracted from the network stream data from the Dataset.

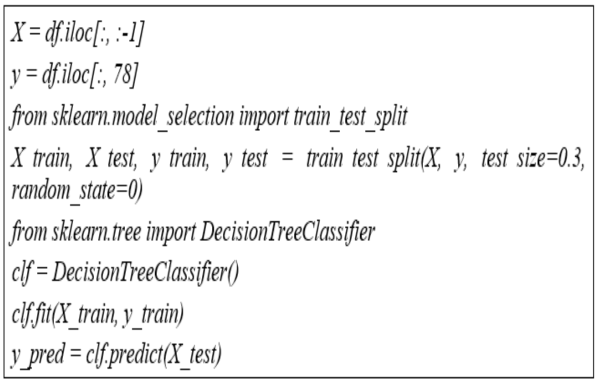


Feature Name	Description
fl_dur	Flow duration
tot_fw_pk	Total packets in the forward direction
tot_bw_pk	Total packets in the backward direction
tot_l_fw_pkt	Total size of packet in forward direction
fw_pkt_l_max	Maximum size of packet in forward direction
fw_pkt_l_min	Minimum size of packet in forward direction
fw_pkt_l_avg	Average size of packet in forward direction
fw_pkt_l_std	Standard deviation size of packet in forward direction
Bw_pkt_l_max	Maximum size of packet in backward direction
Bw_pkt_l_min	Minimum size of packet in backward direction
Bw_pkt_l_avg	Mean size of packet in backward direction
Bw_pkt_l_std	Standard deviation size of packet in backward direction
fl_byt_s	flow byte rate that is number of packets transferred per second
fl_pkt_s	flow packets rate that is number of packets transferred per second
fl_iat_avg	Average time between two flows
fl_iat_std	Standard deviation time two flows
fl_iat_max	Maximum time between two flows

Fig. 2. Features extracted from Dataset.

Machine learning model training: After receiving and preprocessing the input dataset, the system will use previously-stored machine learning algorithms to train the model. In our system, we use Decision Tree Classifiers algorithm from the Sklearn library with Python language. The model training is done with the source code as shown in Fig. 3. After training, we get the weight of the model (here we get the decision tree of the model). Visual decision trees will help users easily observe and evaluate how the algorithm works. To perform the output function, the system uses python's Graphviz library to output the decision tree as an image as shown in Fig. 4.

Whatever machine learning model is used, accuracy is always a top concern. Therefore, the system after training will assess the accuracy and efficiency of the model through some common evaluation parameters for machine learning algorithms (illustrated as shown in Fig. 5-6).



```
X = df.iloc[:, :-1]
y = df.iloc[:, 78]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

Fig. 3. Trainning our model.

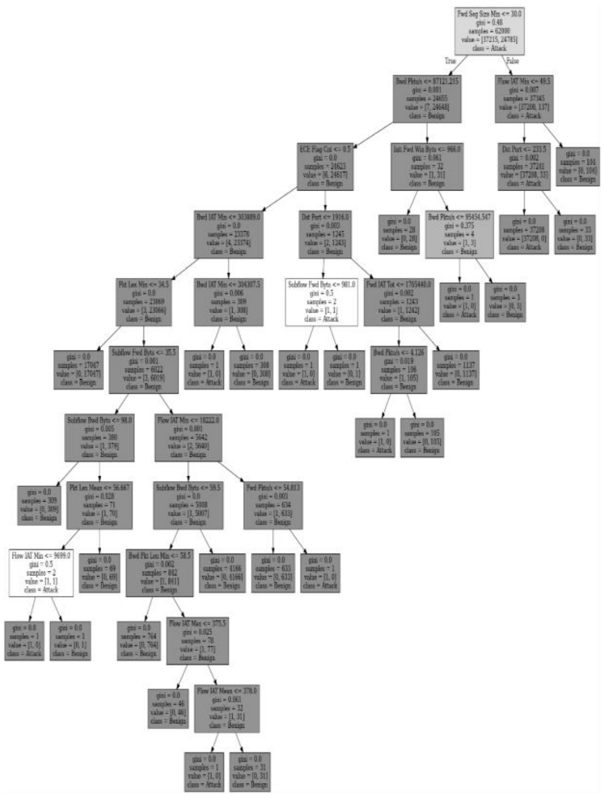
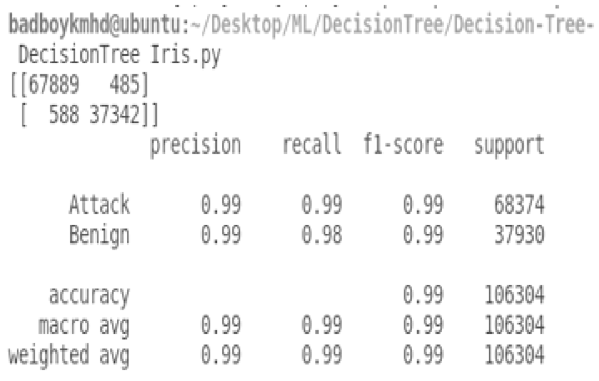


Fig. 4. Result of the trained decision tree.



```
badboykmhd@ubuntu:~/Desktop/ML/DecisionTree/Decision-Tree-
 DecisionTree Iris.py
[[67889   485]
 [  588 37342]]
              precision    recall  f1-score   support

      Attack       0.99      0.99      0.99     68374
      Benign       0.99      0.98      0.99     37930

    accuracy                           0.99    106304
   macro avg       0.99      0.99      0.99    106304
weighted avg       0.99      0.99      0.99    106304
```

Fig. 5. Evaluate the trained machine learning model.

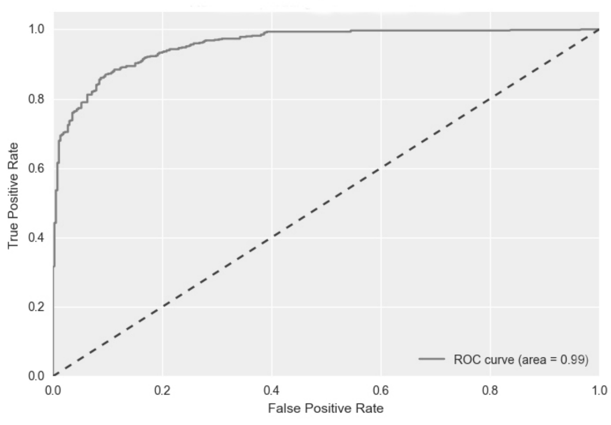
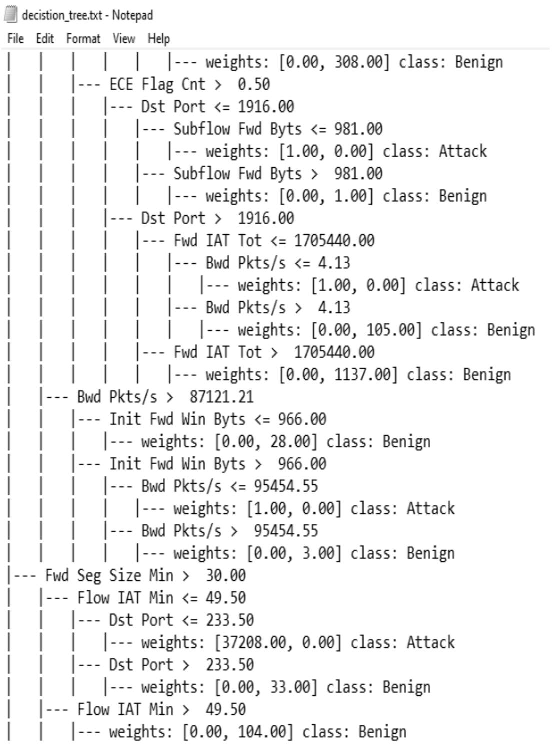


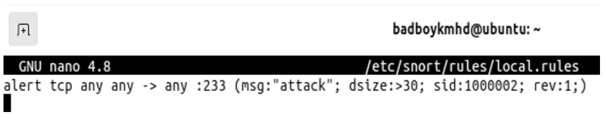
Fig. 6. An ROC curve.

Extract machine learning model training results: Machine learning model training results are node weights stored as a text file as input to the rule generation algorithm for IDS. The weights of the extracted machine learning model are illustrated as shown in Fig. 7.



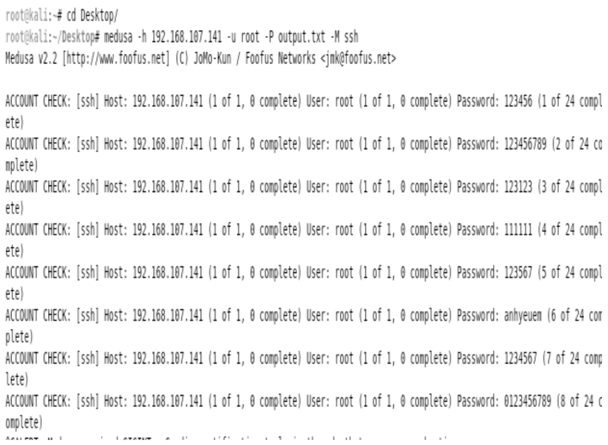
```
decision_tree.txt - Notepad
File  Edit  Format  View  Help
|   |   |   |   |   |--- weights: [0.00, 308.00] class: Benign
|   |   |--- ECE Flag Cnt >  0.50
|   |   |   |--- Dst Port <= 1916.00
|   |   |   |   |--- Subflow Fwd Byts <= 981.00
|   |   |   |   |   |--- weights: [1.00, 0.00] class: Attack
|   |   |   |   |--- Subflow Fwd Byts >  981.00
|   |   |   |   |   |--- weights: [0.00, 1.00] class: Benign
|   |   |   |--- Dst Port >  1916.00
|   |   |   |   |--- Fwd IAT Tot <= 1705440.00
|   |   |   |   |   |--- Bwd Pkts/s <= 4.13
|   |   |   |   |   |   |--- weights: [1.00, 0.00] class: Attack
|   |   |   |   |   |--- Bwd Pkts/s >  4.13
|   |   |   |   |   |   |--- weights: [0.00, 105.00] class: Benign
|   |   |   |   |--- Fwd IAT Tot >  1705440.00
|   |   |   |   |   |--- weights: [0.00, 1137.00] class: Benign
|   |--- Bwd Pkts/s >  87121.21
|   |   |--- Init Fwd Win Byts <= 966.00
|   |   |   |--- weights: [0.00, 28.00] class: Benign
|   |   |--- Init Fwd Win Byts >  966.00
|   |   |   |--- Bwd Pkts/s <= 95454.55
|   |   |   |   |--- weights: [1.00, 0.00] class: Attack
|   |   |   |--- Bwd Pkts/s >  95454.55
|   |   |   |   |--- weights: [0.00, 3.00] class: Benign
|--- Fwd Seg Size Min >  30.00
|   |--- Flow IAT Min <= 49.50
|   |   |--- Dst Port <= 233.50
|   |   |   |--- weights: [37208.00, 0.00] class: Attack
|   |   |--- Dst Port >  233.50
|   |   |   |--- weights: [0.00, 33.00] class: Benign
|   |--- Flow IAT Min >  49.50
|   |   |--- weights: [0.00, 104.00] class: Benign
```

Fig. 7. Model weights extracted..



```
badboykmhd@ubuntu: ~
 GNU nano 4.8                       /etc/snort/rules/local.rules
alert tcp any any -> any :233 (msg:"attack"; dsize:>30; sid:1000002; rev:1;)
```

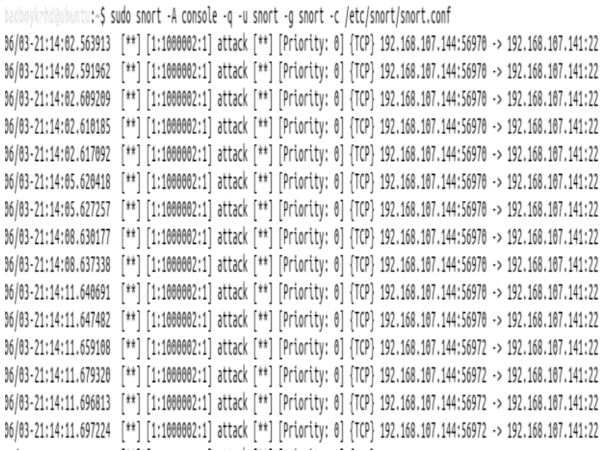
Fig. 8. Rule automatically generated for IDS Snort.



```
root@kali:~# cd Desktop/
root@kali:~/Desktop# medusa -h 192.168.107.141 -u root -P output.txt -M ssh
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 192.168.107.141 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: 123456 (1 of 24 compl
ete)
ACCOUNT CHECK: [ssh] Host: 192.168.107.141 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: 123456789 (2 of 24 co
mplete)
ACCOUNT CHECK: [ssh] Host: 192.168.107.141 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: 123123 (3 of 24 compl
ete)
ACCOUNT CHECK: [ssh] Host: 192.168.107.141 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: 111111 (4 of 24 compl
ete)
ACCOUNT CHECK: [ssh] Host: 192.168.107.141 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: 123567 (5 of 24 compl
ete)
ACCOUNT CHECK: [ssh] Host: 192.168.107.141 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: anhyeuem (6 of 24 com
plete)
ACCOUNT CHECK: [ssh] Host: 192.168.107.141 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: 1234567 (7 of 24 comp
lete)
ACCOUNT CHECK: [ssh] Host: 192.168.107.141 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: 0123456789 (8 of 24 c
omplete)
```

Fig. 9. Demo Brute-Force SSH attack.



```
badboykmhd@ubuntu:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf
06/03-21:14:02.563913  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:02.591962  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:02.609209  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:02.610185  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:02.617092  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:05.620418  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:05.627257  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:08.630177  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:08.637338  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:11.640691  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:11.647482  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56970 -> 192.168.107.141:22
06/03-21:14:11.659108  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56972 -> 192.168.107.141:22
06/03-21:14:11.679320  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56972 -> 192.168.107.141:22
06/03-21:14:11.696813  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56972 -> 192.168.107.141:22
06/03-21:14:11.697224  [**] [1:1000002:1] attack [**] [Priority: 0] {TCP} 192.168.107.144:56972 -> 192.168.107.141:22
```

Fig. 10. IDS Snort issues an attack detection alert.

Generate network attack detection rules for IDS: Experiment to extract feature parameters in a branch of the decision tree, put the parameters into Snort's rules and write them into the ruleset. Here, the feature is taken from the weight file decision_tree.txt, the new rule is written to Snort's rule set stored at /etc/snort/rules/local rules of the system using Ubuntu 20.04 (software IDS Snort is already installed on this system). The automatic generation of network attack detection rules is done through the use of a python script with the functions of opening files, extracting parameters, and writing to Snort's rule files. To be able to use this function, the user needs to have administrative rights (root) of the system. We tested the functionality with the parameters FwdSegSizeMin (incoming packet size) and DstPort (destination port number) to generate the corresponding Snort rule. The result of automatic rule generation for IDS Snort is illustrated in Fig. 8.

D. Evaluation of our rule for network attack detection

To evaluate the generated rule, we use a test scenario as follows:

- Step 1: Prepare two virtual machines to use as the victim machine and the attacker machine respectively. The victim machine uses the Ubuntu 20.04 operating system with the IDS Snort system installed, with the IP address 192.168.107.141. The attacker machine installs the Kali Linux operating system, with the IP address 192.168.107.144.
- Step 2: At the victim's machine, update our rules into IDS Snort's rule database.
- Step 3: At the victim machine, start IDS Snort. Snort has not issued any warnings yet.
- Step 4: At the attacker machine we will perform a network attack. Here we test the Brute-Force SSH attack to detect the root password of the victim machine based on an existing password dictionary file. To perform this type of attack, we use the tool BruteForce Medusa (Illustrated in Fig. 9).
- Step 5: Monitor IDS Snort's network attack detection alerts.

The results of the test scenario show that the Snort IDS system, after updating the proposed autogenerated rule, was able to detect the Brute-Force SSH attack that was not previously in the signature sample database. The result of IDS Snort's network attack detection alert is shown in Fig. 10.

V. DISCUSSION AND FUTURE WORK

With the experiment results mentioned above, we have proven the effectiveness of the proposed method. The proposed method is capable of receiving up-to-date information security knowledge sources published worldwide (in the form of datasets), and automatically processing and generating rules for detecting network attacks for IDS (eg IDS Snort). This proposed method can use any best ML model with a decision tree algorithm such as

ID3, J48, C4.5. The generated rule can work with the IDS system and gives an alert when there is a form of network attack present in the Dataset.

However, the proposed method has some limitations. Specifically, the proposed method has not fully exploited the output features of the machine learning model to include the content of the rule for the IDS. Therefore, the automatically generated rule is still simple, not tight with many parameters to accurately detect different types of network attacks. The cause of this problem is that there is not a tight match between the features extracted from the Dataset and the parameters in the supporting IDS rule structure. In future work, we will continue to research to overcome this limitation.

Conclusion

In this paper, we present a method to automatically generate network attack detection rules for IDS systems based on the results of training machine learning models. With the test results, we prove that the proposed method meets the requirements of increasing the ability to detect new types of attacks, ensuring effective automatic updating of new signs of network attack. In the future, the proposed method will be further studied to perfect in ensuring network security.

Acknownledgment

This research has been funded by the Research Project: VAST01.06/21-22, Vietnam Academy of Science and Technology.

References

[1] M. N. Mohammed and N. Sulaiman, "Intrusion detection system based on SVM for WLAN," *Procedia Technol.*, vol. 1, pp. 313–317, 2012.

[2] H. Tanaka, "Effectiveness and weakness of quantified/automated anomaly based IDs," *Int. J. Netw. Secur. Its Appl. IJNSA Vol*, vol. 9, 2017.

[3] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "AD-IoT: anomaly detection of IoT cyberattacks in smart city using machine learning," 2019, pp. 0305–0310.

[4] D. Breitenbacher, I. Homoliak, Y. L. Aung, N. O. Tippenhauer, and Y. Elovici, "HADES-IoT: A Practical Host-Based Anomaly Detection System for IoT Devices," 2019, pp. 479–484.

[5] J. Singh and M. J. Nene, "A survey on machine learning techniques for intrusion detection systems," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 2, no. 11, pp. 4349–4355, 2013.

[6] F. Sabahi and A. Movaghar, "Intrusion detection: A survey," in *2008 Third International Conference on Systems and Networks Communications*, 2008, pp. 23–26.

[7] J. M. Helm *et al.*, "Machine learning and artificial intelligence: definitions, applications, and future directions," *Curr. Rev. Musculoskelet. Med.*, vol. 13, no. 1, pp. 69–76, 2020.

[8] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," 2018, pp. 29–35.

[9] I. Indre and C. Lemnaru, "Detection and prevention system against cyber attacks and botnet malware for information systems and Internet of Things," 2016, pp. 175–182.

[10] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, no. Mar, pp. 551–585, 2006.

[11] T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," 2017, pp. 1881–1886.

[12] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015, pp. 1–6.

[13] A. O. Prokofiev, Y. S. Smirnova, and V. A. Surov, "A method to detect Internet of Things botnets," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Moscow, Jan. 2018, pp. 105–108. doi: 10.1109/EIConRus.2018.8317041.

[14] C.-J. Wu, Y. Tie, K. Yoshioka, and T. Matsumoto, "IoT malware behavior analysis and classification using text mining algorithm," *Comput. Secur. Symp. 2016*, Oct. 2016.

[15] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: A novel honeypot for revealing current iot threats," *J. Inf. Process.*, vol. 24, no. 3, pp. 522–533, 2016.

[16] L. Hieu, *cicflowmeter: CICFlowMeter V3 Python Implementation*. Accessed: Mar. 03, 2021. [Online]. Available: https://gitlab.com/hieulw/cicflowmeter

[17] R. I. Farhan, A. T. Maolood, and N. Hassan, "Performance analysis of flow-based attacks detection on CSE-CIC-IDS2018 dataset using deep learning," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 20, no. 3, pp. 1413–1418, 2020.

ABOUT THE AUTHORS

Le Hai Viet

Workplace: Department of Information Security, People’s Security Academy, Hanoi, Vietnam.

Email: haiviet.lepsa@gmail.com

Education: He received a master’s degree in Computer systems and networking at Irkutsk National Research Technical University, Irkutsk, Russia (2014). Currently, a PhD student at the University of Science and Technology, VAST, Hanoi, Vietnam.

Recent research direction: Machine learning, Network Security; Intrusion Detection System, Internet of Things.

Nguyen Huy Trung

Workplace: Department of Information Security, People’s Security Academy, Hanoi, Vietnam.

Email: huytrung.nguyen.hvan@gmail.com

Education: He received a doctoral degree in Information system at the University of Science and Technology, VAST, Hanoi, Vietnam (2020).

Recent research direction: Machine learning, Network Security; Malware analysis.

Tran Duc Thang

Workplace: Institute of Information Technology, VAST, Hanoi, Vietnam.

Email: thang@ioit.ac.vn

Education: He received a BS degree from Hanoi University of Science and Technology, in 1996, and MSc from National Tsing Hua University, Taiwan, in 2008.

Recent research direction: Network security; Wireless broadband; Big data; Cloud computing and Internet of Things.