A Data-Driven Network Intrusion Detection Model Based on Host Clustering and Integrated Learning: A Case Study on Botnet Detection

Lena Ara and Xiao Luo(✉)

School of Engineering and Technology, Indiana University-Purdue University Indianapolis, Indianapolis 46202, USA
leara@iu.edu, luo25@iupui.edu

Abstract. The traditional machine learning based network intrusion detection system (NIDS) is based on training a model using known network traffic for selected attacks, and testing it on the unknown network traffic of the same attacks. Evaluating machine learning based IDS with new attack traffic that is different from that training set is rare. With a large amount of network traffic generated every second, it is tough to gather all the traffic then train a model. In this research, we designed and developed an intrusion detection model by treating a network as a community. Based on the traffic behaviors, we first developed a host clustering algorithm to group the hosts into clusters and unique hosts. Then, we developed an integrated learning algorithm to integrate model-based learning derived from host clusters, and instance-based learning obtained from individual hosts. We evaluated the intrusion detection model on the CTU-13 data set which is a botnet attack data set. The results show that our model is more robust and effective for network intrusion detection and gains an average 100% detection rate, with a 0% false positive rate on detecting known attack traffic, and 98.2% detection rate on identifying new Botnet attack traffic.

Keywords: Network intrusion detection · Network flow · Integrated machine learning · Host clustering · Botnet detection

1 Introduction

Conducting intrusion detection and prevention based on network traffic while protecting data privacy is critical, but challenging. Since many hosts can generate a large volume of network traffic in a second, one challenge in the network intrusion detection is to efficiently model the traffic behaviors of the whole network in real-time. Although machine learning and AI algorithms have shown effectiveness for network intrusion detection in recent years [1,10,13,16,24], research rarely discusses the robustness of the learning models to new intrusion traffic.

Based on the recent research in adversarial machine learning [6,11,17,20,21], the model-based learning algorithms or models are vulnerable to the adversarial activities of the machine learning algorithms, and not resilient against being attacked or poisoned by adversarial samples. Although instance-based learning algorithms are relatively robust against adversarial machine learning, the computational cost of instance-based learning is high when the number of data instances in the training set is huge. Some research applies ensemble learning to intrusion detection [12,23,26]. Ensemble learning is a process that strategically generates and combines multiple models or algorithms to detect anomalous network traffic. In this research, we explored a new IDS by integrating model-based and instance-based learning in a distributed manner. This new IDS model is more robust when preventing adversarial actions given that it is not based on one single training set.

The new IDS explored here is based on treating a network as a community. In this community, there are host clusters and individual hosts based on traffic behaviors. The host clustering algorithm is developed to identify the host clusters based on the network flows. The individual hosts are those who do not belong to any clusters. Model-based learning algorithms can be applied to the hosts within host clusters, whereas, instance-based learning algorithms can be applied to the individual hosts. The intrusion detection model integrates learning models derived from host clusters and individual hosts using the majority vote mechanism. To integrate different model-based learning for host clusters, we first transferred trained learning models into decision regions, then, merged the sets of decision regions recursively through applying a feature ranking algorithm. Our experimental results show that the developed IDS model works better than four traditional machine learning models – Random Forest (RF), Naive Bayes (NB), Logistic Regression (LR) and Convolutional Neural Networks (CNN) on detecting known and new Botnet attack traffic.

In summary, the contributions of this paper are as follows:

- We designed and developed a host clustering algorithm based on a cross-testing mechanism. This clustering algorithm measures the similarity between the network flows of hosts and makes use of the machine learning algorithm to summarize the characteristics of the network traffic.
- We designed and developed an integration algorithm for model-based learning by transferring the trained models into decision regions and merging the decision regions of different learning models. A feature ranking algorithm is used to select essential network features to generate and merge the decision regions recursively.
- We compared the effectiveness of our model against four traditional models by using a Botnet data set. The evaluation results show the effectiveness of our intrusion detection model is better than the conventional models and achieves on average a 100% detection rate, with 0% false positive rate, and 98.2% detection rates in detecting unknown Botnet attack traffic.

2 Related Research

Host Clustering. Research in the literature explored host clustering or classification based on the network traffic or host roles. Wei and colleagues studied host clustering based on traffic behavior profiling [29]. Xu and colleagues also researched host clustering [30]. This research included analyzing the behavior of the end hosts in the same network prefixes by considering both host and network level information. Other research used supervised learning algorithms to classify hosts based on their roles in the network [14]. Hosts that were clients were classified to one category, whereas hosts that were email-servers were classified to another category. Many derived features based on the network flows, such as the number of unique host system ports and the number of unique protocols, were used to identify the roles of the hosts. It is computationally costly. Host clustering or classification purely based on the traffic behaviors, such as the incoming and outgoing network flows, has rarely been investigated. Also, very little research has been done on how to measure the similarity between two hosts based on the network traffic. In this research, we explored a host clustering algorithm based on a cross-testing mechanism and maximal clique detection on a graph.

Learning Model Integration. Research has been done on integrating different learning models for data mining or knowledge discovery applications [5,15]. One of the popular approaches is ensemble learning. Ensemble learning can improve classification or prediction performance of network anomaly detection [12,23,26]. Most of the existing ensemble learning model integrates model-based learning through majority vote. Very few investigate a strategy to integrate both model-based and instance-based learning for network intrusion detection. Compared to model-based learning, instance-based learning can work as a specialized detector to identify specific anomalous traffic.

To the best of our knowledge, this research is the first to investigate a strategy to ensemble model-based and instance-based learning by first modeling the network traffic, then integrating the learning models in a distributed manner.

3 System Framework

Figure 1 demonstrates the overall system framework of the intrusion detection model. Given n hosts within a network, first, the host clustering algorithm is used to identify m host clusters based on the similarity of the network traffic flows. One host can belong to one or more host clusters. After determining the host clusters, model-based learning algorithms are applied to the network traffic of each host in the host clusters to generate learning models. The rest of the hosts that are not in the host clusters are used to create instance-based learning models using their specific traffic. For each host cluster, since the network traffic of the hosts are very similar to each other, we further integrated the learning models of the hosts in the same cluster by applying the integration algorithm developed in this research. Lastly, we ensembled the model-based learning models with the instance-based models through a majority vote mechanism. A network flow is classified as an attack if more than half of these learning models predict it as an attack.

Payload information in the network flows is not included. The network flow features used in this research are: duration of the connection in seconds, the type of the protocol (TCP, UDP, ICMP), the port of the connection destination, the port of the connection source, the type of service from destination to source, the type of Service from source to destination, the total bytes of the flow, total packets of the flow, the direction of the flow.

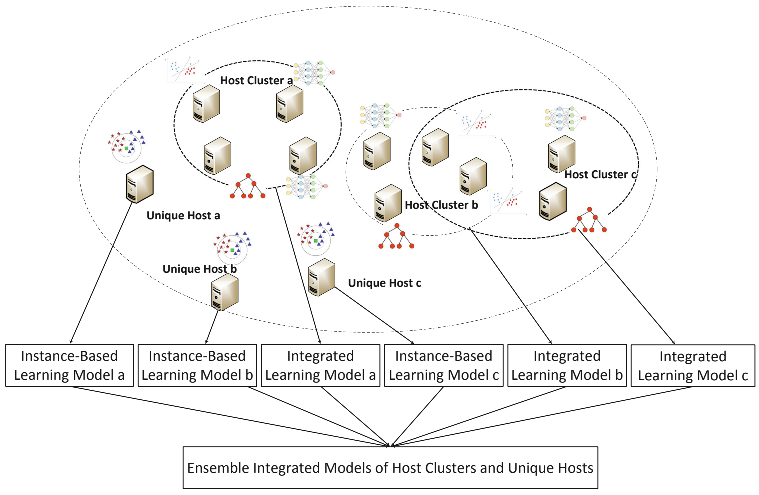
Host Cluster a
Host Cluster b
Host Cluster c
Unique Host a
Unique Host b
Unique Host c
Instance-Based Learning Model a
Instance-Based Learning Model b
Integrated Learning Model a
Instance-Based Learning Model c
Integrated Learning Model b
Integrated Learning Model c
Ensemble Integrated Models of Host Clusters and Unique Hosts

Fig. 1. System Framework of the Intrusion Detection Model

4 Cross-Testing and Host Clustering

The host clustering groups the hosts based on analyzing their traffic patterns. It is not feasible to compare a large amount of network traffic generated by two hosts in a short period. The host clustering algorithm developed in this research is based on an assumption: The network traffic behavior (normal vs. attack) can be summarized and inferred by a trained learning model generated from a learning algorithm. So, if the trained learning models using network flows of two hosts are similar, these two hosts can be clustered into one cluster. We used a cross-testing approach to evaluate the similarity of two learning models. The cross-testing approach works as follows: if a learning model generated by the network flows of host i can detect attack network traffic of host j, and vice versa, the network traffic patterns of both hosts are very similar, and they are in the same cluster. Different learning algorithms take advantage of different optimization functions or processes to generate learning models. We hypothesize that if two hosts are similar enough to each other, the cross-testing needs to be deployed on more than one machine learning algorithm.

The traditional evaluation metrics, such as F1-measure, false positive, and detection rate, can be used for cross-testing. Threshold (θ) can be used to determine the confidence level of similarity. In this research, F1-measure is used to evaluate the similarity between the hosts. The confidence level of similarity is defined as $\theta = 0.9$. When θ is set to be a high value, it means the model created on one host can detect most of the intrusions of the other host with low false alarm rate. Two hosts are similar only if F1-measure values of cross-testing are above 0.9. The cross-testing evaluation results based on a learning algorithm can be stored in a matrix ($CrossT$), as Eq. 1. The $e_{i,j}$ represents the cross-testing that learning model has generated on host i, and tested using network traffic of host j. The $e_{i,i}$ represents the cross-testing results of host i using its network traffic for both training and testing. If m learning algorithms are used for cross-testing, m cross-testing matrices can be aggregated by averaging them into one. If some learning algorithms are treated as more critical than others, a weighted average can be taken, as Eq. 2.

$$CrossT = \begin{pmatrix} 1 & e_{1,2} & \cdots & e_{1,n} \\ e_{2,1} & 1 & \cdots & e_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ e_{n,1} & e_{n,2} & \cdots & 1 \end{pmatrix} \tag{1}$$

$$AverageCrossT = \sum_{i=1}^{m} w_i CrossT_i, \qquad \sum_{i=1}^{m} w_i = 1 \tag{2}$$

Algorithm 1. Cluster-Identification

```
Input: $R$, $Hosts$, $X$, $Edges$
Output: $Cluster$
  $X = \emptyset$
  $R = \emptyset$
  if $(Hosts = \emptyset) \&\& (X = \emptyset)$ then
    $Cluster = R$
  end if
  for each host $v$ in $Hosts$ do
    $N(v)$ = Neighbor Set of $v$ based on $Edges$
    $R$ = Cluster-Identification($R \cup \{v\}, Hosts \cap N(v), X \cap N(v), Edges$)
    $Hosts = Hosts \setminus v$
    $X = X \cup v$
  end for
```

After constructing the cross-testing matrix, an undirected graph is derived to generate host clusters. An edge (i, j) exists only when $e_{i,j}$ and $e_{j,i}$ are both above the threshold θ. The hosts within the network are treated as nodes in an undirected graph, whereas, the edges between the hosts demonstrate the similarities between the hosts. Algorithm 1 is developed to identify the host clusters in a network. Algorithm 1 is based on the maximum clique identification algorithm developed by Östergård [19].

The following example demonstrates the process of host clustering. There are four hosts within this example network; the cross-testing matrix is calculated as Eq. 3. The derived undirected graph and the identified host cluster are presented in Fig. 2.

$$CrossT = \begin{pmatrix} 1 & 0.9 & 0.5 & 0.9 \\ 0.9 & 1 & 0.9 & 0.9 \\ 0.8 & 0.9 & 1 & 0.2 \\ 0.9 & 1 & 0.7 & 1 \end{pmatrix} \tag{3}$$

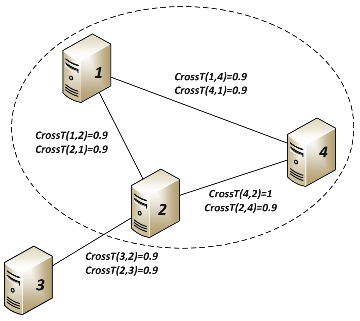

Fig. 2. Host Clustering based on the Cross-Testing Matrix

5 Network Intrusion Detection Model

After host clustering, the intrusion detection model is built by first integrating learning models in host clusters, and then ensembling with the learning models generated by the individual hosts.

5.1 Integrate Learning Models in Host Clusters

In this research, we integrated the learning models in host clusters by combining the decision regions generated by the learning models. Since the hosts within a cluster behave similarly to each other from the network traffic point of view, it is valid to assume that the decision regions generated by the learning model of each host are similar enough to be combined. Different from the tensor-based approach explored by Zhang et al. [32], the integration of the decision regions relies on a recursive feature-ranking process. Figure 3 demonstrates combining decision regions of two hosts: A and B using two network flow features – port number and duration. Figure 4 demonstrates integrated decision regions. In reality, more than two network flow features are used for intrusion detection, although it is not easy to depict in two-dimensional space.

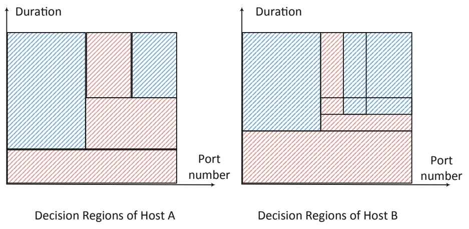

Fig. 3. Decision Regions of Hosts

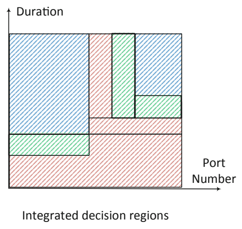

Fig. 4. Integrated Decision Regions

In this research, we explored an algorithm to combine decision regions (shown as Fig. 4) through applying a recursive feature-ranking process [4]. First, the chi-squared (χ^2) feature ranking method [18] is used to identify the top-ranked feature. The calculation of χ^2 is given as Formula 4, where f denotes a network flow feature; $|T|$ means the total number of network flows; C_j means normal or attack. Combined decision boundaries are first constructed based on one top-ranked feature and the decision regions of each host. Within the integrated decision boundaries, the feature ranking method is used again to select the top-ranked feature among the rest of the features and construct decision boundaries. This recursive process continues till all features or top n features are used.

$$\chi^2(f, C_j) = \frac{|T|(P(f, C_j)P(\overline{f}, \overline{C_j}) - P(\overline{f}, C_j)P(f, \overline{C_j}))^2}{P(f)P(\overline{f})P(C_j)P(\overline{C_j})} \tag{4}$$

Algorithm 2 demonstrates the generation of integrated decision regions (InteDecisionRegs), where n is the pre-defined top features to generate decision regions, F_0 is the top-ranked feature in feature list F. The function $CrtDecisionRegs$ is used to create the decision regions based on the decision regions of each host ($HostDecRegs$), the new feature F_0, and the existing integrated decision regions.

5.2 Ensemble Learning Models of Host Clusters and Unique Hosts

Within a network, it is normal that some hosts have their unique traffic behaviors, which are not similar to those of the rest of the hosts. In our IDS model, while generating the integrated models on the host clusters, instance-based learning models are produced on the unique hosts at the same time. Specifically, k-Nearest-Neighbor (kNN) [2] is used to create instance-based learning models on the individual hosts. In our experiments, k is set to 2.

Algorithm 2. Algorithm-for-Combining-Decision-Regions

```
Input: $F$, $n$, $HostDecRegs$
Output: $InteDecisionRegs$
  $i \leftarrow 1$
  $F \leftarrow chi^2$(F)
  while $(F! = \emptyset)\&\&i \leq n$ do
    $InteDecisionRegs$=$CrtDecisionRegs(F_0, HostDecRegs, InteDecisionRegs)$
    $F = F \setminus F_0$
    $i = i + 1$
    Algorithm-for-Combining-Decision-Regions($F, N, HostDecRegs$)
  end while
  return $InteDecisionRegs$
```

As demonstrated in Fig. 1, we ensemble the learning models generated from the host clusters with the instance-based models produced from the individual hosts by applying the majority vote mechanism. A network flow is classified as an attack, if more than half of these models predict it as an attack.

6 Experimental Results and Analysis

6.1 Data Set

This intrusion detection model is evaluated on a publicly available Botnet data set – CTU-13 data set [8].

CTU-13 has 13 different data sets, and the network flows are labeled as Background, Botnet, C&C Channels and Normal. In our experiments, we used data sets 10 and 13. Table 1 demonstrates the different Botnet attack scenarios in each data set. In this research, one objective is to evaluate the robustness of our model on detecting new attacks. So, each data set is used to generate an intrusion detection model. The testing phase includes using the other data set, which contains new attacks that are not in the training data. The model training and test set up is as follows: 70% of data from each data set is to generate the models, 30% of each data set and 100% of the other data set are used to test the models.

Table 1. Type of Botnet in Each Data Set

Data Set	Type of Botnet
CTU-13 - 10	IRC, UDP DDoS, CTU compiled and controlled Botnet
CTU-13 - 13	SPAM, Port Scan, HTTP Botnet

In our research, the traffic is separated based on the hosts. The network flows of each data set are grouped according to the destination IP addresses. Each unique IP address is treated as a host in the network. Only the hosts that contain both normal and Botnet flows are kept to build the intrusion detection model. In the end, there are 16 hosts in data set 10 and 18 hosts in data set 13.

6.2 Host Clustering Results

In this research, two learning algorithms – Decision Tree [22] and Support Vector Machine [7] – are used to generate the cross-testing matrices. An equally weighted average is taken to integrate the two cross-testing matrices for Decision Tree and Support Vector Machine respectively. Based on cross-testing and host clustering algorithms described in Sect. 4, Figs. 5 and 7 present the weighted average of the cross-testing matrices of F1-measures. Figures 6 and 8 present the undirected graphs generated based on these matrices. After applying the host clustering algorithm, three host clusters are identified for data set 10, and two host clusters are identified for data set 13.

$$\left(\begin{array}{cccccccccccccccc}
1 & 0.06 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.5 & 1 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.5 & 0.5 & 0.4 & 0.5 & 0.4 & 0.5 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.1 & 0.4 & 0.6 & 0.6 & 0.6 & 1 & 1 & 1 & 0.6 & 1 & 0.9 & 1 & 0.9 & 1 & 0.9 & 0.9 \\
0.4 & 0.5 & 0.6 & 0.8 & 0.8 & 0.6 & 0.7 & 1 & 0.7 & 0.6 & 0.8 & 0.6 & 0.8 & 0.8 & 0.7 & 0.8 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.1 & 0.4 & 0.6 & 0.6 & 0.5 & 1 & 1 & 1 & 0.6 & 1 & 1 & 1 & 0.9 & 1 & 0.9 & 0.8 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.1 & 0.4 & 0.6 & 0.6 & 0.5 & 1 & 0.6 & 1 & 0.7 & 1 & 0.7 & 1 & 1 & 0.8 & 1 & 0.7 \\
0.9 & 0.2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right)$$

Fig. 5. Cross-testing Matrix of Data Set 10

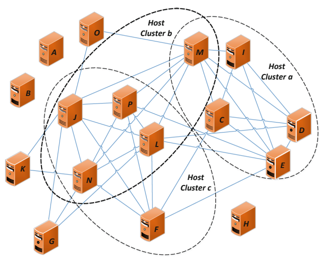

Fig. 6. Host Clustering of Data Set 10

$$\begin{pmatrix}
1 & 0.5 & 0.5 & 0.3 & 0.2 & 0.5 & 0.4 & 0.8 & 1 & 1 & 0.4 & 1 & 0.8 & 1 & 0.4 & 1 & 1 & 1 \\
0.9 & 1 & 1 & 1 & 1 & 0.7 & 0.7 & 0.8 & 0.8 & 0.7 & 0.7 & 0.7 & 0.8 & 0.8 & 0.8 & 0.6 & 0.6 & 1 \\
0.9 & 1 & 1 & 1 & 1 & 0.7 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.4 & 0.2 & 0.2 & 1 & 0.4 & 0.2 & 0.4 & 0.2 & 0.4 & 0.3 & 0.4 & 0.4 & 0.4 & 0.3 & 0.4 & 0.4 & 0.4 & 0.3 \\
0.2 & 0.8 & 0.7 & 0.7 & 1 & 0.4 & 0.7 & 0.7 & 0.2 & 0.4 & 0.5 & 0.3 & 0.6 & 0.4 & 0.8 & 0.1 & 0 & 0.1 \\
0.2 & 0.4 & 0.3 & 0.4 & 0.5 & 1 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0.1 \\
0.9 & 0.9 & 0.9 & 1 & 0.6 & 0.7 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.9 & 0.9 & 0.8 & 0.6 & 0.5 & 0.7 & 1 & 1 & 1 & 1 & 0.4 & 1 & 1 & 1 & 0.8 & 1 & 1 & 1 \\
0.9 & 0.9 & 0.8 & 0.6 & 0.5 & 0.7 & 1 & 1 & 1 & 1 & 0.4 & 1 & 1 & 1 & 0.8 & 1 & 1 & 1 \\
0.2 & 0.2 & 0.3 & 0.1 & 0 & 0.4 & 0.4 & 0.7 & 0.5 & 1 & 0.4 & 1 & 0.4 & 0.6 & 0.4 & 0.1 & 0 & 0.4 \\
0.2 & 0.8 & 0.7 & 0 & 0.3 & 0.6 & 0.4 & 0.4 & 0.2 & 0.5 & 1 & 0.3 & 0.4 & 0.4 & 0.4 & 0 & 0 & 1 \\
0.2 & 0.2 & 0.3 & 0.1 & 0 & 0.4 & 0.4 & 0.7 & 0.5 & 0.8 & 0.4 & 1 & 0.4 & 0.4 & 0.4 & 0.1 & 0 & 0.4 \\
0.9 & 0.9 & 0.9 & 1 & 0.6 & 0.7 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.2 & 0.2 & 0.3 & 0.1 & 0 & 0.4 & 0.4 & 0.7 & 0.5 & 1 & 0.4 & 1 & 0.4 & 1 & 0.4 & 0.1 & 0 & 0.4 \\
0.9 & 1 & 1 & 1 & 1 & 0.7 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.9 & 0.9 & 0.8 & 0.8 & 0.6 & 0.7 & 1 & 1 & 1 & 1 & 0.4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0.9 & 0.9 & 0.8 & 0.6 & 0.5 & 0.7 & 1 & 1 & 1 & 1 & 0.4 & 1 & 1 & 1 & 0.8 & 1 & 1 & 1 \\
0.2 & 0.7 & 0.7 & 0 & 0.2 & 0.4 & 0.3 & 0.5 & 0.1 & 0.3 & 0.3 & 0.1 & 0.5 & 0.3 & 0.2 & 0 & 0 & 1
\end{pmatrix}$$

Fig. 7. Cross-testing Matrix of Data Set 13

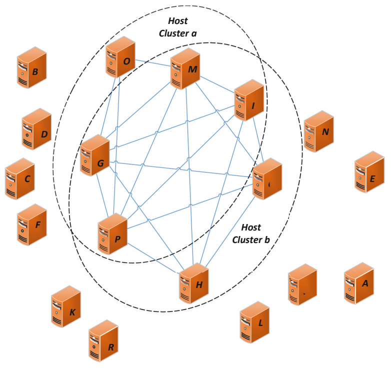

Fig. 8. Host Clustering of Data Set 13

6.3 Performance of the Intrusion Detection Model

After the host clusters are identified, intrusion detection models are built for each data set respectively. First, integrated learning models are built for the host clusters based on the algorithm of combining decision regions. Meanwhile, the instance-based learning model – kNN is created on the individual hosts that are not in the host clusters.

In this research, we compared our intrusion detection model with four other traditional detection models that have been used for network intrusion detection: Random Forest (RF) [25,31], Naive Bayes (NB) [3,9], Logistic Regression (LR) [25,28] and Convolutional Neural Networks (CNN) [27]. The RF was set to use a maximum of 100 trees and split points chosen from a random selection of 3 features. The Gaussian Probability Density Function was used for NB. The number of hidden layers of CNN was 5, and the number of neurons for each layer was set to be 20, 15, 25, 15, and 20, respectively. The metrics - false positive rate (FPR) and detection rate (DR) were used to evaluate the results. FPR and DR are calculated using Eqs. 5 and 6.

$$FPR = \frac{Number\ of\ Normal\ Detected\ as\ Attack}{Total\ Number\ of\ Normal\ Connections} \tag{5}$$

$$DR = \frac{Number\ of\ Detected\ Attacks}{Total\ Number\ of\ Attack\ Connections} \tag{6}$$

The models generated based on the training data are evaluated on the test data of the same data set, as well on the data of the other data set. We evaluated the model performance on each host to see whether the models are robust enough for some hosts that have very minimum traffic.

Table 2. Test Results on Data Set 10 using Learning Model of Data Set 10

Hosts		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Average
RF	DR	1	0.78	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.986
	FPR	0	0	0.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0.001
LG	DR	0	0.99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.061
	FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NB	DR	0	0.95	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.059
	FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CNN	DR	1	0.78	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.986
	FPR	0	0	0.02	0	0	0	0	0	0	0	0	0	0	0	0	0	0.001
Our Model	DR	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	FPR	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Table 3. Test Results on Data Set 13 using Learning Model of Data Set 10

Hosts		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	Ave.
RF	DR	0.81	0.68	0.65	0	0.23	1	0	0.34	0	1	0	1	0	1	0	0	0	0	0.37
	FPR	0.04	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0.03
LG	DR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	FPR	0	0	0	0	0	0	0	0.4	0	0	0	0	0	0	0	0	0	0	0.02
NB	DR	0	0.08	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01
	FPR	0	0	0.05	0	0	0	0	0.47	0	0.09	0.06	0.04	0.05	0	0	0	0	0	0.04
CNN	DR	0.99	0.37	0.21	0	0.2	0.4	1	0.34	1	1	0	1	0.5	0	0.5	0	0	1	0.47
	FPR	0	0	0	0	0	0	0	0	0	0	0	0.02	0	0	0	0	0	0	0.001
Our Model	DR	**1**	**0.85**	**1**	**0.6**	**0.73**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0.95**
	FPR	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Table 2 demonstrates the results of training and testing the learning model using network traffic of data set 10. The LR and NB both do not work well on this data set. Other than Host B, the detection rates on the rest of the hosts are all 0. The RF and the CNN perform better. Other than Host B, the detection rates on the rest of the hosts are all 1, and the false positive rates on all hosts are 0 or close to 0. Our model works better than all the other learning models and

Table 4. Test Results on Data Set 13 using Learning Model of Data Set 13

Hosts		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	Ave.
RF	DR	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	FPR	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
LG	DR	0.99	0.62	0.39	0.9	0.65	0.8	0	0.34	0	1	0	1	0	1	0	1	1	1	0.59
	FPR	0	0	0	0	0	0.5	0.2	0.16	0.07	0.19	0.13	0.16	0.21	0.03	0.35	0.09	0	0	0.12
NB	DR	0.99	0.43	0.3	0	0.13	0.8	0	0	0	1	0	1	0	1	0	0	0	1	0.37
	FPR	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0.03
CNN	DR	1	1	1	1	1	1	1	1	1	1	0.5	1	1	1	1	1	1	1	0.97
	FPR	0	0	0.05	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0.03
Our Model	DR	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	FPR	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Table 5. Test Results on Data Set 10 using Learning Model of Data Set 13

Hosts		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Ave.
RF	DR	1	**0.78**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0.99**
	FPR	0	0.9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0.05**
LG	DR	1	0.09	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.95
	FPR	0	0.9	0.46	0.04	0.31	0.21	0.34	0.3	0.26	0.14	0.14	0.35	0.14	0.18	0.41	0.07	0.24
NB	DR	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.94
	FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CNN	DR	1	0.77	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0.98**
	FPR	0	1	0.04	0	0.2	0.02	0.03	0	0	0	0	0	0	0.02	0.2	0	0.08
Our Model	DR	**1**	0.69	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0.98
	FPR	**0**	0.9	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0.05**

gains detection rates of 1 and false positive rates of 0 on all of the hosts. In our model, because the Host B is not within any of the host clusters, instance-based learning, kNN, is used. This demonstrates that instance-based learning works better if the network traffic behavior of a host is unique.

Table 3 shows the testing results of the learning model generated from data set 10 but tested on data set 13. These results demonstrate the robustness of traditional models and our model on analyzing new Botnet attack traffic. RF and CNN both work well on some of the hosts of data set 13. For hosts A, G, I, and R, CNN works better than the RF. CNN gains detection rates close to 1 and false positive rates of 0. For hosts B, C and N, RF achieves higher detection rates. Our model outperforms all the other models on all hosts. Especially for hosts D, K, P and Q, both RF and CNN cannot detect any Botnet traffic on those hosts, whereas our model gains a detection rate of 0.6 on host D and a detection rate of 1 on hosts K, P and Q. The false positive rates for these four hosts are also as low as 0. Our model detects all Botnets on hosts G, H, I, M, P and Q in cluster b of data set 13 (shown in Fig. 8). This confirms that the traffic similarity between these hosts is very high. As long as the model works on one of the hosts, it works on the rest of the hosts in a host cluster.

Table 4 demonstrates the results of training and testing the learning model using network traffic of data set 13. LG and NB perform worse than the other three models. Our model and RF both achieve high performance with detection rates of 1 and false positive rates of 0 on all the hosts. Although CNN performs worse than our model and RF on hosts C, F, and K, it performs well on the rest of the hosts.

Table 5 shows the results of the learning model generated from data set 13 but tested on data set 10. The detection rates of all models on all hosts other than Host B are 1. The false positive rates of RF, CNN and our model are all close to 0 except Host B. Overall, LR and NB perform worse than the other three models. RF performs slightly better than CNN and our model on Host B.

To compare the overall performance of the four detection models based on the traditional machine learning or AI algorithms, we took the average of all hosts of the two data sets. Figures 9 and 10 show the comparison of the detection rates and false positive rates respectively. Our model has the highest detection rate and lowest false positive rate comparing to the other traditional models.

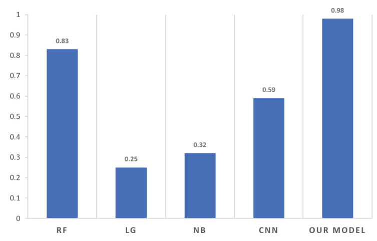

Fig. 9. Comparison of Average DR

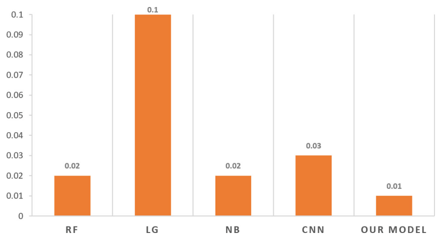

Fig. 10. Comparison of Average FPR

7 Conclusions and Future Work

In this research, we designed and developed an innovative network intrusion detection model by first identifying the host clusters and then integrating the learning models of the host clusters and unique hosts. Different from the traditional learning models, our model doesn't require gathering all the network traffic to train a centralized learning model. Instead, the training can happen in parallel on all the hosts. To integrate the learning models within the clusters, we proposed a method to merge the decision regions in a recursive way based on a feature ranking algorithm. The achieved detection rates and false positive rates demonstrate that our model gains better detection rates and false positive rates on both existing and new network flow patterns.

The current work uses F1-measure and threshold 0.9 for creating the host clusters, it doesn't require the traffic behaviors of two hosts are exact the same with regards to intrusion detection. We will consider other metrics and thresholds in the future. In this research, supervised learning algorithms were used. We also plan to extend this intrusion detection model by applying an unsupervised learning algorithm in the future.

References

1. Akramifard, H., Khanli, L.M., ABalafar, M., Davtalab, R.: Intrusion detection in the cloud environment using multi-level fuzzy neural networks. In: Proceedings of International Conference on Security and Management, pp. 152–159 (2015). https://doi.org/10.1109/CSE.2015.26
2. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**(3), 175–185 (1992)
3. Amor, N.B., Benferhat, S., Elouedi, Z.: Naive bayes vs. decision trees in intrusion detection systems. In: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 420–424. ACM (2004)
4. Andrzejak, A., Langner, F., Zabala, S.: Interpretable models from distributed data via merging of decision trees. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 1–9. IEEE (2013)
5. Chong, M., Abraham, A., Paprzycki, M.: Traffic accident analysis using machine learning paradigms. Informatica **29**(1) (2005)
6. Clements, J., Yang, Y., Sharma, A., Hu, H., Lao, Y.: Rallying adversarial techniques against deep learning for network security. arXiv preprint arXiv:1903.11688 (2019)
7. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
8. García, S., Grill, M., Stiborek, J., Zunino, A.: An empirical comparison of botnet detection methods. Comput. Secur. **45**, 100–123 (2014)
9. Hasan, M.A.M., Nasser, M., Pal, B., Ahmad, S.: Support vector machine and random forest modeling for intrusion detection system (IDS). J. Intell. Learn. Syst. Appl. **6**(01), 45 (2014)
10. Huang, H., Al-Azzawi, H., Brani, H.: Network traffic anomaly detection. arXiv preprint arXiv:1402.0856 (2014)
11. Huster, T.P., Chiang, C.Y.J., Chadha, R., Swami, A.: Towards the development of robust deep neural networks in adversarial settings. In: MILCOM 2018–2018 IEEE Military Communications Conference (MILCOM), pp. 419–424. IEEE (2018)
12. Kim, G., Lee, S., Kim, S.: A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Exp. Syst. Appl. **41**(4), 1690–1700 (2014)
13. Le, D.C., Zincir-Heywood, A.N., Heywood, M.I.: Data analytics on network traffic flows for botnet behaviour detection. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–7. IEEE (2016)
14. Li, B., Gunes, M.H., Bebis, G., Springer, J.: A supervised machine learning approach to classify host roles on line using sflow. In: Proceedings of the First Edition Workshop on High Performance and Programmable Networking, pp. 53–60. ACM (2013)
15. Lin, W.Y., Hu, Y.H., Tsai, C.F.: Machine learning in financial crisis prediction: a survey. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **42**(4), 421–436 (2012)
16. Sheikhan, M., Jadidi, Z., Farrokhi, A.: Intrusion detection using reduced-size rnn based on feature grouping, neural computing and applications. Neural Comput. Appl. **21**(6), 1185–1190 (2012)
17. Martinez, E.E.B., Oh, B., Li, F., Luo, X.: Evading deep neural network and random forest classifiers by generating adversarial samples. In: Zincir-Heywood, N., Bonfante, G., Debbabi, M., Garcia-Alfaro, J. (eds.) FPS 2018. LNCS, vol. 11358, pp. 143–155. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18419-3_10

18. Moh'd A Mesleh, A.: Chi square feature extraction based Svms Arabic language text categorization system. J. Comput. Sci. **3**(6), 430–435 (2007)
19. Östergård, P.R.: A fast algorithm for the maximum clique problem. Disc. Appl. Math. **120**(1–3), 197–207 (2002)
20. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519. ACM (2017)
21. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 372–387. IEEE (2016)
22. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
23. Reddy, R.R., Ramadevi, Y., Sunitha, K.: Real time anomaly detection using ensembles. In: 2014 International Conference on Information Science and Applications (ICISA), pp. 1–4. IEEE (2014)
24. Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top. Computat. Intell. **2**(1), 41–50 (2018)
25. Tsai, C.F., Hsu, Y.F., Lin, C.Y., Lin, W.Y.: Intrusion detection by machine learning: A review. Exp. Syst. Appl. **36**(10), 11994–12000 (2009)
26. Vanerio, J., Casas, P.: Ensemble-learning approaches for network security and anomaly detection. In: Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, pp. 1–6. ACM (2017)
27. Vinayakumar, R., Soman, K., Poornachandran, P.: Applying convolutional neural network for network intrusion detection. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1222–1228. IEEE (2017)
28. Wang, Y.: A multinomial logistic regression modeling approach for anomaly intrusion detection. Comput. Secur. **24**(8), 662–674 (2005)
29. Wei, S., Mirkovic, J., Kissel, E.: Profiling and clustering internet hosts. DMIN **6**, 269–75 (2006)
30. Xu, K., Wang, F., Gu, L.: Network-aware behavior clustering of internet end hosts. In: 2011 Proceedings of the IEEE INFOCOM, pp. 2078–2086. IEEE (2011)
31. Zhang, J., Zulkernine, M., Haque, A.: Random-forests-based network intrusion detection systems. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **38**(5), 649–659 (2008)
32. Zhang, S., Yang, L.T., Kuang, L., Feng, J., Chen, J., Piuri, V.: A tensor-based forensics framework for virtualized network functions in the internet of things: Utilizing tensor algebra in facilitating more efficient network forensic investigations. IEEE Consum. Electron. Mag. **8**(3), 23–27 (2019)