ADSIM: Network Anomaly Detection via Similarity-aware Heterogeneous Ensemble Learning

Wenqi Chen*, Zhiliang Wang*‡, Ying Zhong*, Dongqi Han*, Chenxin Duan*, Xia Yin†‡, Jiahai Yang*‡, Xingang Shi*‡
*Institute for Network Sciences and Cyberspace, Tsinghua University
†Department of Computer Science and Technology, Tsinghua University
‡Beijing National Research Center for Information Science and Technology, China
chenwq19@mails.tsinghua.edu.cn, wzl@cernet.edu.cn, {{zhongy18, handq19, dcx19}@mails., yxia@}tsinghua.edu.cn, {yang, shixg}@cernet.edu.cn

***Abstract*—The last decade has seen increasing application of machine learning to various tasks, including network anomaly detection. But anomaly detection methods using a single machine learning algorithm often fail to perform well, since network traffic can have complex and changeable patterns. Therefore, many solutions based on ensemble learning have been proposed to address this problem. However, previous studies have a essential drawback that they overlook the similarity between the weak classifiers, which may degrade the detection ability of the model. What's more, prior work use offline and supervised algorithms, which means a large amount of memory and reliable labels are necessary during the training period.**

In this paper, we propose ADSIM, an online, unsupervised, and similarity-aware network anomaly detection algorithm based on ensemble learning. In the training phase, ADSIM incrementally maintains a distance matrix to record the similarity between the classifiers and uses hierarchy clustering to group similar classifiers. In the detecting phase, each cluster will be assigned a weight based on the consistency of the classifier outputs within it. We evaluate ADSIM on two datasets, MAWILab and CIC-IDS-2017, and the results show that ADSIM can accurately detect various anomalies and outperforms state-of-the-art ensemble learning methods.

***Index Terms*—Network Traffic Anomaly Detection, Ensemble Learning, Clustering**

I. INTRODUCTION

The variety and scale of cyber attacks have been increasing in recent years, causing serious loss and adverse impact. Therefore, network intrusion detection has become rather important. Network intrusion detection consists of two categories [1], [2]: signature-based and anomaly-based detection. Signature-based detection uses specific rules to match attack behaviors, which takes great effort to design the detection mechanism case by case, and cannot cope with zero-day attacks. As a result, the other type, namely anomaly-based detection, has attracted more interest from researchers. Anomaly-based detection aims to construct profiles of normal traffic, and traffic whose pattern differs from the profiles would be identified as anomalies. With the development of machine learning, more and more machine learning algorithms are being applied in this field [3], [4].

However, when a single machine learning algorithm is used for network traffic anomaly detection, it often fails to obtain ideal results [5]. The main reason is that network traffic can have complex patterns, which are hard to be learned with only one classifier. To address this problem, many ensemble learning based detection algorithms have been proposed so far, i.e., combine multiple algorithms (weak classifiers) with some strategies to enhance the detection ability of the model [5]–[10]. However, most of them are not practical for the following reasons:

Supervised Learning. Most existing methods require labels to calculate metrics (e.g. accuracy) or identify the incorrectly classified samples in the training period [7]–[9], but labeling data is expensive and time-consuming. Furthermore, it's hard to include all kinds of malicious traffic in the training data, so the trained models cannot detect all attacks.

Offline Training. Many prior work retrain the classifiers with the entire dataset in each iteration [7]–[9], thus storing it in memory and loading it repeatedly, which is space-consuming. What's more, these methods become unscalable when the size of the training dataset becomes larger.

Not Similarity-aware. Most existing work neglect the similarity between the weak classifiers [7]–[10], which may degrade the detection capability of the model. For example, Weighted Voting assigns weights to the weak classifiers according to their detection performance (e.g., accuracy) and calculates the weighted average as the final result [8]. If there are many similar weak classifiers, it's likely that they all misclassify a sample while they possess a large weight in the voting stage, which may lead to more false/missed alarms.

In this paper, we propose ADSIM, a network anomaly detection algorithm based on ensemble learning: in the training phase, a distance matrix of the weak classifiers is maintained in an online way, and a low-complexity clustering algorithm is used to group similar classifiers; in the detecting phase, each cluster is assigned a weight according to the consistency of the classifier outputs within it. ADSIM is online, unsupervised and similarity-aware, and our evaluation shows that ADSIM can accurately detect various network anomalies.

II. RELATED WORK

Existing ensemble learning algorithms can be divided into two classes, homogeneous ensemble learning and heterogeneous ensemble learning [11]. **Homogeneous ensemble learning** comprises two common methods: Boosting and Bagging. Boosting repeatedly assigns weights to the samples and retrains the weak classifiers to make them focus more on the misclassified samples [12]. Bagging randomly extracts

978-3-903176-32-4

feature/sample subsets of the dataset to train different weak classifiers to enhance model diversity [13], [14]. **Heterogeneous ensemble learning** mainly consists of two different categories: Stacking and Voting. Stacking uses a series of weak classifiers as first-level classifiers, and a second-level classifier makes the final decision based on the detection results of the first-level classifiers [15]. Voting uses the weighted average of the results from the weak classifiers as the final result and the weight can be assigned in different ways [11]. All of the above methods have been widely used in network anomaly detection [7], [8], [16]–[20], but they either need expensive labels for supervised training or takes up a lot of memory for offline processing, and more importantly, the similarity between the classifiers is not considered.

In consideration of the issues in related studies, we propose ADSIM, which seeks to combine the results of the weak classifiers in an online and unsupervised manner, and recognize the similarity of the weak classifiers and eliminate its negative effects.

III. ADSIM DESIGN

A. Motivating Example

This section introduces a simple example to illustrate the intuition behind ADSIM. As Fig. 1 shows, five weak classifiers, denoted by A-E respectively, are used for network anomaly detection, and the table shows their detection results for the packets (malicious packets are in red color and 1 means producing an alarm). It can be seen that classifier A,B,C are very similar because they have the same detection results for these packets. Prior studies have overlooked this similarity and treat them as three different classifiers. Taking the most common method majority voting for example [5], [8], A,B,C all fail to identify the last packet as an anomaly while they possess larger weight, so the last packet will still be considered benign in the voting stage, resulting in a missed alarm.

To eliminate this effect, one solution is to find similar classifiers and treat them as a whole, which is exactly the intuition of ADSIM. ADSIM recognizes similar classifiers based on previous detection results, groups them into the same clusters, and assigns a weight to each cluster based on the consistency of the results. For this example, ADSIM gets three clusters $[A, B, C]$, $[D]$, and $[E]$, and the last packet would be successfully identified as an anomaly because $[D]$ and $[E]$ hold heavier weight now. What's more, we design ADSIM to be online and unsupervised to make it more practical.

B. Overview

The workflow of ADSIM is shown in Fig. 2. Suppose at timestamp t, the m-th packet arrives, and n weak classifiers $c_1, c_2, ..., c_n$ are used.

1) Feature Extraction: An online feature extractor is used to get the feature vector $\boldsymbol{f}$ of the packet. Some existing feature extraction algorithms can be used here (e.g. AfterImage [3], CICFlowMeter [21]).

2) Anomaly Detection: The anomaly detector receives $\boldsymbol{f}$, and operates differently in the training and detecting phase.

- **In the training phase**:
  - **a) Weak classifiers Detection**: The classifiers detect the new packet based on f, getting the detection result vector $\vec{\boldsymbol{x_m}} = [x_m^{(1)}, \ldots, x_m^{(n)}]$, where the i-th dimension represents the detection result of the i-th classifier.
  - **b) Updating the distance matrix**: According to $\vec{\boldsymbol{x_m}}$, a distance matrix $\mathbf{D} = [D_{i,j}]$ will be maintained in an online manner, where $D_{i,j}$ represents the distance between the i-th and the j-th classifier.
  - **c) Clustering the classifiers**: At the end of the training phase, the classifiers will be divided into k clusters $[N_1, N_2, ..., N_k]$ according to $\mathbf{D}$.
- **In the detecting phase**:
  - **a) Weight Decision**: Each cluster will be assigned a weight based on the consistency of its internal classifier outputs, and the weighted average of the cluster results is the final anomaly score.

More detailed description of distance counting, classifier clustering, and weight decision will be discussed as follows.

C. Distance Between Classifiers

When two weak classifiers are used to detect the same set of packets, their results can be viewed as two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. By calculating the correlation distance between these two vectors, the similarity between the corresponding weak classifiers can be known. The correlation distance is defined as:

$$d_{cor} = 1 - \frac{(\boldsymbol{u} - \overline{u}) \cdot (\boldsymbol{v} - \overline{v})}{\|(\boldsymbol{u} - \overline{u})\|_2 \|(\boldsymbol{v} - \overline{v})\|_2}$$

However, if this formula is directly used, all the detection results ($\boldsymbol{u}$ and $\boldsymbol{v}$) need to be stored in memory, which is space-consuming. To this end, we propose the following algorithm to incrementally maintain the distance matrix:

First, the Linear sum $\vec{\boldsymbol{L}}$ can be maintained in an online manner:

$$\vec{\boldsymbol{L_m}} = (L_m^{(i)}) = (\sum_{j=1}^{m} (x_j^{(i)})) = (L_{m-1}^{(i)} + x_m^{(i)})$$

Based on the Linear Sum, the Residual Linear Sum $\vec{\boldsymbol{R_m}} = (R_{m-1}^{(i)} + x_m^{(i)} - \frac{L_m^{(i)}}{m})$, the Residual Squared Sum $\vec{\boldsymbol{RS_m}} = (RS_{m-1}^{(i)} + (x_m^{(i)} - \frac{L_m^{(i)}}{m})^2)$ and the Partial Correlation Matrix $\mathbf{PC_m} = [PC_{m-1}(i,j) + (x_m^{(i)} - \frac{L_m^{(i)}}{m})(x_m^{(j)} - \frac{L_m^{(j)}}{m})]$ can be maintained in the same way.

So at any moment, if the training phase is finished, the distance matrix can be obtained from the above variables:

$$\mathbf{D} = [D_{i,j}] = [1 - \frac{PC(i,j)}{\sqrt{RS^{(i)}}\sqrt{RS^{(j)}}}]$$

D. Clustering Weak Classifiers

Based on the distance matrix $\mathbf{D}$, a hierarchical clustering algorithm can be used to cluster the classifiers. The process consists of two steps, namely Linkage and Division, which are briefly described below.

Linkage takes the weak classifiers set $c = [c_1, ..., c_n]$ and the distance matrix $\mathbf{D}$ as input. Initially, each classifier is

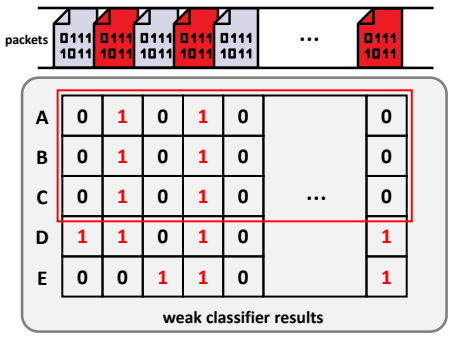


Fig. 1. A motivating example.

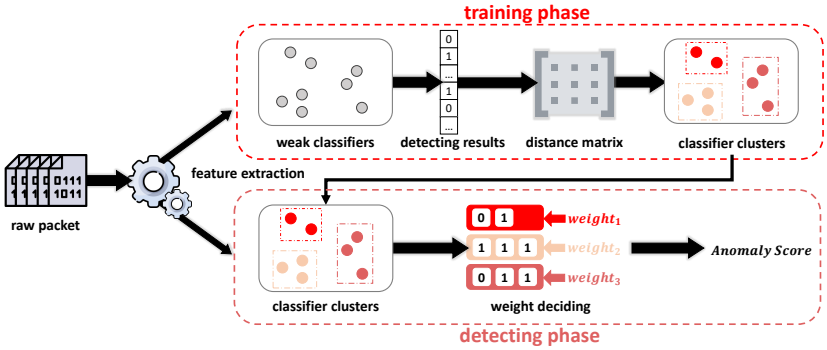


Fig. 2. The workflow of ADSIM.

treated as a cluster; in each iteration, the nearest cluster pair (N_i, N_j) are chosen and merged into a new cluster, and its distances to other clusters are computed by **Average Linkage** method, which is defined as:

$$dist(N_i, N_j) = \frac{1}{|N_i||N_j|} \sum_{c_i \in N_i} \sum_{c_j \in N_j} d(c_i, c_j)$$

The iteration stops when there is only one cluster left, and the remaining cluster can be represented as a dendrogram.

Division divides the dendrogram into final clusters: First, the root node of the dendrogram is broken and the tree is divided into two subtrees. Then Division is executed recursively over the subtrees until the termination condition is satisfied (e.g., cluster size, distance threshold, etc.). In ADSIM, the recursion stops when all classifier distances in the subtree are less than $\phi = 0.1$. After Division, each subtree is a classifier cluster.

E. Weight Decision

In the detecting phase, a weight will be assigned to each cluster according to the consistency of the classifier outputs within it: First, a confidence p is assigned to each weak classifier. If the labels are available, the confidence can be determined by the performance of the classifier (e.g. accuracy, etc.), otherwise it can be set to some common values like 0.5. Then if a cluster N_i contains n_i classifiers and m_i of them classify a packet as an anomaly, then the confidence of this cluster can be calculated as:

$$p_i = 1 - \binom{n_i}{m_i}(1-p)^{m_i} p^{n_i - m_i}$$

Finally, the anomaly score can be calculated by the weighted average of the detection results of the clusters:

$$Score = \frac{1}{k} \sum_{i=1}^{k} p_i \cdot Score_i$$

IV. Evaluation

A. Datasets

CIC-IDS-2017 Dataset. CIC-IDS-2017 dataset is built by injecting attack traffic into normal background traffic [22]. The benign background traffic is generated by B-Profiling system to characterize the behavior of 25 users, and the attacks were implemented consecutively from Tuesday to Friday, which contain various anomalies (e.g., Brute Force, DDoS/DoS, Web Attack, Botnet, etc.).

MAWILab Dataset. MAWILab dataset is built by capturing real network traffic traces on a backbone link between Japan and the United States [23]. Based on the collected traffic, MAWILab project uses a combination of four anomaly detectors (Hough, Gamma, KL, PCA) to label the data. The traffic identified as anomalies are further classified into different taxonomies (e.g., DoS, port scan, etc.).

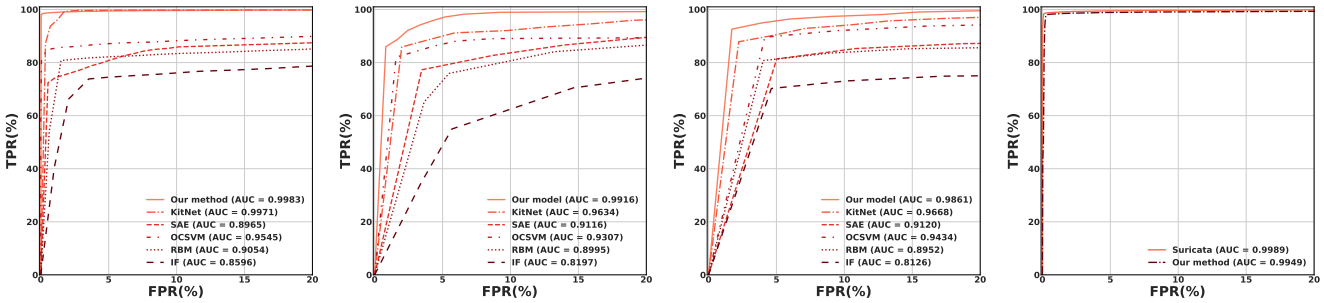
In CIC-IDS-2017 dataset, we choose four attacks, DDoS, HeartBleed, Botnet and Port Scan for experiments. As for MAWILab dataset, we select the traffic on 6/1/2019 as a test point. As traffic in CIC-IDS-2017 dataset have relatively simple patterns compared to MAIWLab, making it easier to analyze the detection results, in the experiments on CIC-IDS-2017 dataset, we analyze the performance improvement of ADSIM over the weak classifiers, and MAWILab dataset is used to compare our algorithm with the baseline methods.

B. Experiment Setup

Feature Extractor. The feature extractor in our experiments is AfterImage [3]. AfterImage retrieves over 100 statistics from the meta-data of packets in an online manner, which perfectly portray the volume information (e.g., bandwidth, packet rate, jitter, etc.) of different streams. For more details about AfterImage, we refer the readers to [3].

Weak Classifiers. The weak classifiers include OCSVM [24], Isolation Forest [25], RBM (Restricted Boltzmann Machine) [26], SAE (Sparse Autoencoder) [27], KitNet [3]. We also use Suricata [28] for signature-based detection.

Baseline Methods. We compare ADSIM with some state-of-the-art anomaly detection algorithms based on ensemble learning. We choose XGBoost [29] and Random Forest [14] as the representative algorithms for homogeneous ensmeble learning, and for heterogeneous ensemble learning algorithms, we choose MVexp [8] and HELAD [10] as baseline methods. MVexp uses weighted voting and assigns weight to each cluster based on its accuracy, HELAD combines Autoencoder and LSTM for anomaly detection. All these methods have shown good performance in network anomaly detection.



(a) DDoS detection result (b) Botnet detection result (c) Port Scan detection result (d) HeartBleed detection result

Fig. 3. ROC curves on CIC-IDS-2017 dataset.

TABLE I
RESULTS ON CIC-IDS-2017 DATASET.

Method	DDoS		HeartBleed		Botnet		Port Scan		Overall	
	F-Score	AUC	F-Score	AUC	F-Score	AUC	F-Score	AUC	F-Score	AUC
IF	0.841	0.860	-	-	0.722	0.820	0.792	0.813	0.669	0.596
OCSVM	0.917	0.954	-	-	0.882	0.931	0.892	0.943	0.697	0.683
SAE	0.854	0.896	-	-	0.862	0.912	0.859	0.912	0.691	0.605
KitNet	0.990	0.997	-	-	0.935	0.963	0.949	0.967	0.668	0.632
RBM	0.786	0.905	-	-	0.844	0.899	0.838	0.895	0.760	0.607
Suricata	-	-	0.994	0.999	-	-	-	-	0.668	0.632
ADSIM	**0.995**	**0.998**	**0.986**	**0.995**	**0.964**	**0.992**	**0.972**	**0.986**	**0.988**	**0.997**

TABLE II
RESULTS ON MAWILAB DATASET.

	Precision	Recall	F-Score	AUC
XGBoost	0.766	0.774	0.770	0.865
Random Forest	0.816	0.779	0.797	0.863
MVexp	0.836	0.832	0.834	0.884
HELAD	0.842	0.846	0.844	0.907
ADSIM	**0.865**	**0.867**	**0.866**	**0.931**

Evaluation Metrics The evaluation metrics include Precision, Recall, F-Score and AUC. Based on the classification result, Precision, Recall, F-Score can be calculated as follows: $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$, $F = \frac{2PR}{P+R}$, and AUC is the area under the ROC curve (the curve of recall varying with FPR).

C. Results and Findings

ADSIM combines the advantages of the weak classifiers. The results on CIC-IDS-2017 dataset are shown in Table I. It can be seen in the overall results that ADSIM has significant improvement over the weak classifiers, with F-Score and AUC approaching 98%. The separated results can better illustrate the reason of this improvement: the anomaly-based methods can effectively detect DDoS/Botnet/Port Scan attacks, but cannot deal with attacks like HeartBleed, and it's the opposite for signature-based methods like Suricata. Consequently, the weak classifiers fail to identify all of these attacks, resulting in their poor performance. ADSIM combines the advantages of the weak classifier, which enable it to cope various attacks, thus contributing to its performance improvement. The corresponding ROC curves are plotted in Fig. 3, which shows in detail the improvement of ADSIM over the weak classifiers on these attacks respectively.

ADSIM outperforms the current state-of-the-art ensemble learning methods. Table II shows the results of ADSIM and the baseline methods on MAWILab dataset. It can be seen that ADSIM has better detection ability than the state-of-the-art ensemble learning methods, with F-Score improved by at least 4% and AUC improved by at least 3%. Especially, the performance improvement of ADSIM over MVexp is a strong indication that our similarity-aware ensemble learning scheme effectively enhances the detection ability of the model. The ROC curves are plotted in Fig. 4, which also shows the performance improvement of ADSIM over the baseline methods.

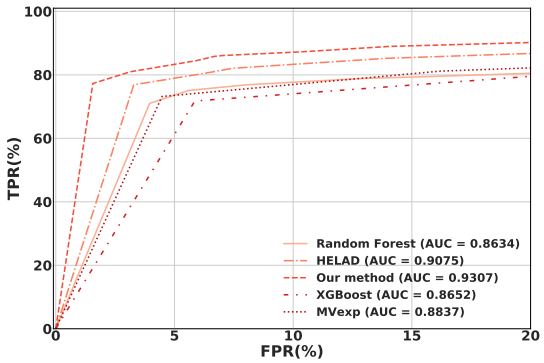


Fig. 4. ROC curves on MAWILab dataset.

V. CONCLUSION

In this paper, we introduce ADSIM, an online, unsupervised, and similarity-aware ensemble learning based network anomaly detection algorithm. ADSIM incrementally maintains a distance matrix of the weak classifiers, divides similar weak classifiers into the same clusters, and assigns a weight to each cluster based on the consistency of the classifier outputs within it. Experiment results show that ADSIM can accurately detect various attacks and outperforms state-of-the-art ensemble learning methods.

REFERENCES

[1] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[2] Nour Moustafa, Jiankun Hu, and Jill Slay. A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications*, pages 33–55, 2019.

[3] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. *Network and Distributed System Security Symposium*, 2018.

[4] Mingyi Zhu, Kejiang Ye, Yang Wang, and Cheng-Zhong Xu. A deep learning approach for network anomaly detection based on amf-lstm. In *International Conference on Network and Parallel Computing*, pages 137–141. Springer, 2018.

[5] Gianluigi Folino and Pietro Sabatino. Ensemble based collaborative and distributed intrusion detection systems: A survey. *Journal of Network and Computer Applications*, pages 1–16, 2016.

[6] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. A survey of intrusion detection systems based on ensemble and hybrid classifiers. *Comput. Secur.*, pages 135–152, 2017.

[7] Jing Gao et al. Consensus extraction from heterogeneous detectors to improve performance over network traffic anomaly detection. In *2011 Proceedings IEEE INFOCOM*, pages 181–185. IEEE, 2011.

[8] Juan Vanerio and Pedro Casas. Ensemble-learning approaches for network security and anomaly detection. pages 1–6. ACM, 2017.

[9] Sahil Garg, Amritpal Singh, Shalini Batra, Neeraj Kumar, and Mohammad S Obaidat. Enclass: Ensemble-based classification model for network anomaly detection in massive datasets. In *IEEE Global Communications Conference (GlobeCOM)*, pages 1–7. IEEE, 2017.

[10] Ying Zhong, Wenqi Chen, Zhiliang Wang, Yifan Chen, Kai Wang, Yahui Li, Xia Yin, Xingang Shi, Jiahai Yang, and Keqin Li. Helad: A novel network anomaly detection model based on heterogeneous ensemble learning. *Computer Networks*, 2019.

[11] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall, 2012.

[12] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, pages 148–156. Citeseer, 1996.

[13] Leo Breiman. Bagging predictors. *Machine learning*, pages 123–140, 1996.

[14] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, pages 278–282. IEEE, 1995.

[15] David H Wolpert. Stacked generalization. *Neural networks*, pages 241–259, 1992.

[16] Parag Verma, Shayan Anwar, Shadab Khan, and Sunil B Mane. Network intrusion detection using clustering and gradient boosting. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2018.

[17] Kamaldeep Singh et al. Big data analytics framework for peer-to-peer botnet detection using random forests. *Information Sciences*, pages 488–497, 2014.

[18] A.P.F. Chan et al. Comparison of different fusion approaches for network intrusion detection using ensemble of rbfnn. *International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 3846–3851, 2005.

[19] Iwan Syarif, Ed Zaluska, Adam Prugel-Bennett, and Gary Wills. Application of bagging, boosting and stacking to intrusion detection. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 593–602. Springer, 2012.

[20] N.F. Haq, A.R. Onik, and F.M. Shah. An ensemble framework of anomaly detection using hybridized feature selection approach (hfsa). In *Proceedings of SAI Intelligent Systems Conference (IntelliSys)*, pages 989–995, 2015.

[21] Arash Habibi Lashkari et al. Characterization of tor traffic using time based features. In *International Conference on Information Systems Security and Privacy (ICISSP)*, pages 253–262, 2017.

[22] Iman Sharafaldin, Amirhossein Gharib, Arash Habibi Lashkari, and Ali A Ghorbani. Towards a reliable intrusion detection benchmark dataset. *Software Networking*, pages 177–200, 2018.

[23] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2010.

[24] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134, 2016.

[25] Fei Tony Liu et al. Isolation forest. In *IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[26] Ugo Fiore, Francesco Palmieri, Aniello Castiglione, and Alfredo De Santis. Network anomaly detection with the restricted boltzmann machine. *Neurocomputing*, pages 13–23, 2013.

[27] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 2011.

[28] Suricata: Open source ids/ips/nsm engine. https://suricata-ids.org/. 2 2019. (Accessed on 2/12/2019).

[29] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.