IoT Malware Detection based on Latent Representation

1st Cuong Nguyen Van
Le Quy Don Technical University
HaNoi, VietNam
cuongnguyen.lqd@gmail.com

2rd Viet Anh Phan
Le Quy Don Technical University
HaNoi, VietNam
anhpv@mta.edu.vn

3nd Van Loi Cao
Le Quy Don Technical University
HaNoi, VietNam
loi.cao@lqdtu.edu.vn

4nd Khanh Duy Tung Nguyen
Le Quy Don Technical University
HaNoi, VietNam
tungkhanhmta@gmail.com

***Abstract*—This paper proposes a new approach for IoT malware detection system based on the analysis of IoT network traffic features. First, we use an autoencoder network to gather latent presentation of the input data. This is followed by a classifier to identify whether an IoT network traffic is malware or benign. We carry out a comprehensive comparison of different input feature sets and figure out that using latent representation is more effective than the original features. This proves that autoencoder network can compress the IoT network traffic features and keep only the most meaningful features. The model latent representation and classifies IoT malware and benign with high performance. Another finding is that our trained model can detect new types of abnormal IoT network traffics which do not appear in the training process.**

***Index Terms*—IoT Malware, Malware detection, anomaly detection, Autoencoder(AEs)**

I. Introduction

Nowadays Internet of Things (IoT) systems have had important applications in a wide range of areas such as smart home, elder care, infrastructure, and so on. The vision of the IoT describes a future where many everyday objects are interconnected through the global network. They collect and share data with each other to allow widespread monitoring, analyzation, optimization, and control [1]. It is expected that 50 billion devices will be available by 2020 [2]. However, various studies have revealed that IoT devices and their software are plagued with weaknesses [3] and vulnerabilities [4], [5].

Since 2008, cyber-criminals have created malware to attack IoT-devices, such as routers and other network equipments. Several prominent cyber attacks happening in recent years, attackers have used malware to gain access to such poorly protected devices and perform distributed denial of service (DDoS) attacks, expose users' private data, steal identities, or just cause inconvenience. This is somewhat confirmed by the Mirai's infamous attack and source-code release in 2016, leaving over 1.2 million IoT devices infected with malware. Figure 1 shows top 10 IoT threat verdicts in the first half of 2019.

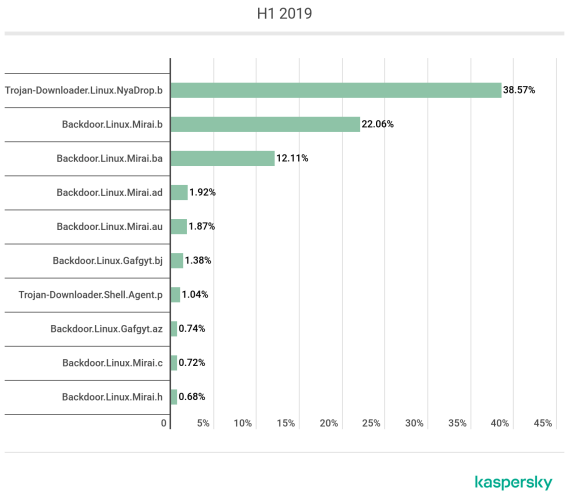

Figure 1. TOP 10 IoT threat verdicts, first half of 2019

To reduce damage from malware infection by protecting IoT devices from new and variant malware attacks, studies on security issues for IoT devices and the detection of IoT malware have been conducted. Granjal et al. [6] focused on analyzing extant protocols and mechanisms to secure communications for the IoT. Costin et al. [7] presented a comprehensive survey and analysis of all currently known IoT malware classes, without discussing IoT malware detection approaches. Felt et al. [8] reviewed the 46 pieces of mobile malware in the wild and collected dataset to evaluate the effectiveness of mobile malware identification and prevention methods. Additionally, a recent survey by Imran Makhdoom et al. [9] is quite comprehensive when presenting security issues and risks of threats to IoT devices. They emphasized that inherent safety provided by the communication protocols does not guard against harmful IoT malware and node compromise attacks.

This paper presents a new approach to IoT malware detec-

978-1-7281-4510-5/20/$31.00 ©2020 IEEE

tion system based on analysis on IoT network traffic features. The method used for abnormal detection is autoencoders (AEs). This design was named as "autoencoder" by Japkowicz et al. [10], who applied it for novelty detection in 1995. An autoencoder is a neural network which learns to reconstruct its input at the output layer. A narrow middle layer compresses redundancies in the input data while non-redundant information remains [10]. This effect resembles a non-linear PCA. We used AEs as building blocks in deep neural networks [11], and after training, the output layer is discarded, and the hidden layer is used as a new feature representation.

In summary, the main contributions of this research include:

- First, we present the data reduction method used by AEs model to detect abnormal IoT malware data and compare with other data reduction methods (PCA, tSNE) to evaluate the effectiveness.
- Second, we use several datasets as training data and categorize them on these datasets. Then evaluate the results, the ability to detect other abnormal IoT malware data in the remaining datasets.

II. Related Work

Recently, autoencoders (AEs) have been used as a feature representation learners in hybrid anomaly detection models [12]–[18]. In such hybrid anomaly detection models, the middle hidden layer of trained autoencoders is employed as a new feature representation for enhancing the performance of traditional anomaly detection methods, such as distance-based/density-based anomaly detection techniques [12]. The latent feature representation can be learned in supervised learning [18], semi-supervised learning [12]–[14] and unsupervised learning [15], [16]. Once an AE has been trained on training data (normal data, both labelled normal and anomalous data, or unlabelled data), its decoder is discarded, the encoder is used as a feature learner for the following anomaly detection method in a hybrid model. The central idea is that the encoder of a trained AE can represent the original input data into a feature space which can be lower dimension, and discover more robust features distinguishing normal behaviors from anomalies.

Erfani et al. [16] employed a deep belief network (DBN) for constructing a robust feature representation and for one-class classifications (OCCs), such as One-class Support Vector Machine (OCSVM) and Support Vector Data Description (SVDD). The main objective is to solve the problem of high-dimensional anomaly detection. Firstly, the DBN was pre-trained in the greedy layer-wise fashion by stacking Restricted Boltzmann Machines (RBMs) trained in unsupervised manner. OCCs such as OCSVM and SVDD were then stacked on top of the DBN. This hybrid can inherit the strengths of high decision classification accuracy from these OCCs and feature representation from DBNs. Eight high-dimensional UCI datasets were employed to assessed the structure, and the experimental results show that the hybrid model often outperform stand-alone OCSVM and SVDD.

Recently Cao et al. [12] introduced two regularized AEs, namely SAE and DVAE for capture the normal behaviors of network data. These regularizers AEs are attempted to put normal data towards a small region at the origin of the latent feature space, which can result in reserving the rest of the space for anomalies occurring in the future. In this work, the authors assumed that only normal samples are available for training. These regularized AEs were designed to overcome the problem of identifying anomalies in high-dimensional network data.The latent representation of SAE and DVAE was then used for enhancing simple one-class classifiers. The proposed models were testified on eight well-known network datasets. The experimental results confirmed that their models not only can yield a better performance, but also are less sensitive on a wide range of parameter settings in comparison to stand-alone OCCs, and those of other feature representation methods.

Alternatively, in supervised manner, Vu et al. [18] proposed Multi-distribution VAE (MVAE) to represent normal data and anomalous data into two different regions in the latent feature space of VAE. Originally, variational autoencoders (VAEs) learn to map input data into a standard Gaussian distribution $\mathcal{N}(0, 1)$ in its middle hidden layer. In MVAE, the class labels, normal and anomaly, are incorporated into the loss function of VAE to force normal data and anomalous data into two different regions. These regions have the same Gaussian distribution shape with $\sigma = 1$, but different mean values. The proposed model was evaluated on two publicly network security datasets, and it produces promising performance.

In this work, we attempt to construct a latent representation for identifying IoT malware in supervised manner. This means that we first train autoencoders on unlabelled data (both benign, malware,and C&C) to map the original data into a low dimensional feature space. Supervised learning methods are then stacked on top of the AE encode. The main objective is to investigate the behaviors of the AE latent representation according to the ratio between benign, malware and C&C in the training data.

III. Proposed Approach

Our proposed approach for IoT malware detection consists of two phases: Training phase and Inference phase (as shown in Figure 2). In the training phase, an autoencoder is trained in unsupervised manner to construct a latent representation in the middle hidden layer of the AE. Following this, the latent feature space is employed to represent labelled data for effectively building multi-class classifiers for identifying IoT-based malware. In the inference phase, the querying data is passed to the encoder part of the AE. The output of the encoder is fed into the classifiers for detecting whether the querying data is malware or benign. More details are presented in the subsection III-A and III-B

A. Constructing Latent Representation

In this subsection, we show how to construct a latent representation of AEs which facilitates binary classifiers in

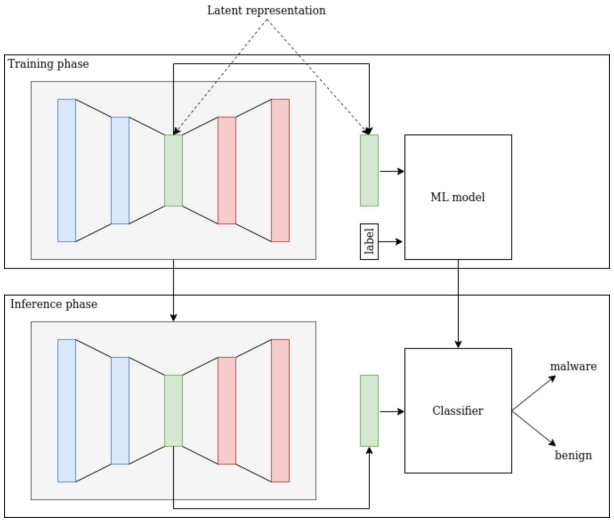

Figure 2. IoT malware detection process

distinguishing IoT-based malware from benign. Ordinary autoencoders consist of two parts: encoder (called f_θ) and decoder (called g_ϕ), where θ and ϕ are the parameters of the encoder and decoder respectively. AEs are aimed to learn reconstructing the input data at the output layer. In other words, the encoder compresses the input data X into a much lower latent feature space, $Z = f_\theta(X)$. Inversely, the decoder attempts to reconstruct the original input data from the latent data Z, $X = g_\phi(Z)$. The parameters θ and ϕ are learned during the training process by minimizing the discrepancy between the original input and its reconstruction, called reconstruction error (RE). The training algorithm of AEs can be shown in Algorithm 1.

Algorithm 1 Training autoencoder algorithm

INPUT: Dataset $X = x_1, x_2 \ldots x_N$
OUTPUT: Parameters θ, ϕ

θ, $\phi \leftarrow$ Initialize parameters
repeat
$RE = \frac{1}{N}\sum_{i=1}^{N} \|x_i - f_\theta(g_\phi(x_i))\|^2$
θ, $\phi \leftarrow$ Update parameters using gradients of RE
until convergence of parameters θ, ϕ

Our idea is to train an AE in unsupervised manner to obtain a latent representation. This means that, a training data set consisting of both malware and benign (without labels) is used for training the AE in unsupervised manner. Once the training process completed, the encoder, f_θ, is employed to extract meaningful features from the input features. Suppose we have a training data set:

$$D = (x_i, y_i), i = \overline{1, N}$$

where N is the number of data samples, and y_i is the label of data. Latent data of each n_i can be easily obtained by using the encoder f_θ: $z_i = f_\theta(x_i)$. Thus, we can have data set

$$D' = (z_i, y_i), i = \overline{1, N}$$

Now, we can use the data set D' for construct binary classifiers instead of the original training data set D.

The latent representation is a compressed version of the original feature space, which can represent the input data in lower dimension, and reserve meaningful information of the data. Therefore, the latent representation can help binary classification methods, such as Decision Tree, K-nearest Neighbors, and Support Vector Machine, efficiently learn to capture the characteristics of benign as well as malware. Thus binary classifiers operated on the latent representation may perform better than that on the original feature space.

We also evaluated our proposed model in comparison to other feature reduction methods, such as Principal Component Analysis (PCA) [19] and t-Distributed Stochastic Neighbor Embedding (tSNE) [20]. More details of the comparison will be discuss in the experiments, and the Results and Discussion sections.

B. Training Classifiers

The subsection describes how to construct binary classifiers on the latent representation. As mentioned above, we train binary classification algorithms on the latent data (D') instead of on the original training data set (D). In this paper, three popular classification methods, namely Support vector machine [21], K-nearest Neighbors [22], and Decision Tree [23], are employed for identifying IoT-based malware. We use D' to build three binary classifiers. We then construct a hybrid between the encoder (f_θ) and each of these classifiers. For querying stage, the testing data is passed the encoder, and the output of encoder is fed into these classifiers. The hybrid will assign a label (either benign or malware) to each of querying samples. The detail of the implementation is described in Section IV. We evaluate our proposed model on publicity IoT dataset with 23 scenarios. The experimental results and discussion are presented in Section IV and V.

IV. Experiments

A. Dataset

The dataset namely IoT-23 is published by Stratosphere Laboratory [24]. IoT-23 is a new dataset of network traffic from Internet of Things (IoT) devices. It has 20 malware captures executed in IoT devices, and 3 captures for benign IoT devices traffic. The malicious captures were generated by executing malware of Mirai, Torii, Hide and Seek, Hajime and others in Raspberry Pi. The benign captures are the traffics from three different IoT devices including a Philips HUE smart LED lamp, an Amazon Echo home intelligent personal assistant and a Somfy Smart Door Lock.

Table I shows the instance number for each malicious scenario. Each data sample is represented by 23 features extracted from a network flow in which the flow generated by the malware is labelled as abnormal. It can be seen in Table I, many scenarios are unbalanced such as D03 D04, D05, D07, D08, D10, D11, D12, D14, D15, D16, D17, and D19 whereby abnormal flows are minorities.

Table I
TWENTY MALICIOUS SCENARIOS

DataSet	Malware	Normal	Abnormal	DataSet	Malware	Normal	Abnormal
D01	Hide and Seek	4151	14110	**D11**	Mirai	67332	1560870
D02	Muhstik	3519	16631	**D12**	Okiru	600	6263
D03	Linux.Mirai	1097	29323	**D13**	IRCBot	4065	1866
D04	Hakai	317	1741	**D14**	Trojan	2093	6
D05	Linux.Hajime	14963	237	**D15**	Mirai	609	69853
D06	Kenjiro	4483	6567	**D16**	Mirai	124	18
D07	Torii	1222	9	**D17**	Mirai	527	40201
D08	Torii	946	9	**D18**	Mirai	1969	5594
D09	Kenjiro	9809	4705	**D19**	Mirai	198	2402
D10	Mirai	687	4001	**D20**	Hide and Seek	191	108

Three categorical features including protocol, service, and connection state, are transformed into one-hot-encoding. In the experiments, thirteen features are selected and normalized into [-1, 1] to build malware detection models. Each scenario is split into three sets of training, validation, and test by the ratio of 3:1:1.

B. Experimental Settings

We conducted the experiments with different scenarios to confirm two points: 1) the ability to generate sophisticated features of the proposed method, 2) the consistent behavior of malware among different devices. **Scenario 1**. All training sets from D01 - D20 are gathered to train the autoencoder by the unsupervised manner. In this step, label information is ignore. We then apply the encoder to generate the latent representations for feature vectors.

The quality of the feature generator was assessed by the performance of several machine learning models such as support vector machines (SVM), k-nearest neighbors (kNN), and decision tree (DT) on original data, and new representations generated by the proposed model, PCA, and tSNE.

Scenario 2. The research question is that whether or not malware behavior is the same in different IoT devices. To verified the question, we merge training sets of D01 - D03 into a single one to build prediction models. After that, we run the prediction on the test sets.

For the implementation, we use Keras framework [25] with Tensorflow backend for the autoencoder, scikit-learn library [26] for PCA, tSNE, support vector machines, random forest, and decision tree.

The autoencoder network is comprised of nine fully connected layers with the symmetric structure. The neurons of the layers from the input to the output are 32-24-12-6-3-6-12-24-32. The middle layer is latent representations with the size of 3. This means that the feature vectors are reduced to the dimension of 3. To find the best configurations for PCA, tSNE, SVM, and DT, we apply the grid search algorithm.

V. RESULTS AND DISCUSSION

A. The quality of latent representations of the autoencoder

Tables II and III compare the methods according AUC and F1 on unbalanced and balanced datasets, respectively. Generally, the proposed method can produced sophisticated representations for the data. As the results, it achieves the best results on most of datasets except D10, D11, D14, D19 for the unbalanced and D01 for the balanced. Table II shows that the unbalanced data problem is challenging for any methods, especially for the datasets with just a few samples of minority classes like D07 and D14. Thus, we will focus to analyze the quality of the classifiers on the balanced datasets.

Although the AE can reduce the feature dimension significantly, the data characteristics are remained. In the comparison with the original features, AE boosts the AUC and F1 of the classifiers on all balanced datasets notably. For example, DT's performance on D06 is improved 2.94% of AUC and 1.12% of F1; on D09 the improvement is 3.48 % of AUC and 0.32% of F1. On unbalanced datsets, the AE also remains the classifiers' performance . Interestingly, the latent representations of AE just has the dimension of 3, while that of the original is 32.

Comparing with other dimension reduction algorithms like PCA and TSNE, AE has retained meaningful properties of the data. As seen in Table III, The DT classifiers with features generated by AE achieve the higher AUC and F1 than those with features produced by PCA and tSNE. This phenomenon is also kept in the cases of kNN and SVM classifiers.

B. The ability to detect unknown malware

Our purpose is to verify the similarity of the behavior among malware and among devices. Table IV presents the performance comparison. In AE-3 columns, the classifiers are trained on the training sets of D01-D03, and tested on the test sets of the corresponding datasets. From the results, we can see that malware in different IoT malware has shown similar behavior. Although, just training with the malware of Hide and Seek (D01), Muhstik (D02) and Linux.Mirai (D03), the classifiers can detect other malware types with a high accuracy. The performance even is better than the original data.

Table II
THE COMPARISON OF METHODS ACCORDING TO AUC, F1 ON UNBALANCED DATA.

Dataset	Algorithm	DT				kNN				SVM			
	Method	None	PCA	tSNE	AE	None	PCA	tSNE	AE	None	PCA	tSNE	AE
D03	AUC	99.55	99.55	99.51	99.55	99.55	99.55	**99.77**	99.55	99.1	98.99	99.71	98.85
	F1	**99.98**	**99.98**	99.95	**99.98**	99.97	99.97	99.97	99.97	99.97	99.97	99.97	99.97
D04	AUC	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	F1	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	99.85	99.85	**100**	**100**
D05	AUC	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	F1	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
D07	AUC	99.8	99.8	50	**100**	99.59	99.8	99.8	**100**	**100**	99.59	99.59	**100**
	F1	66.67	66.67	0	**100**	66.67	33.33	66.67	**100**	66.67	66.67	66.67	**100**
D08	AUC	99.73	99.73	**100**	99.73	99.82	99.82	99.56	99.56	**100**	99.65	**100**	**100**
	F1	85.71	85.71	**100**	85.71	85.71	85.71	85.71	85.71	85.71	85.71	85.71	**100**
D10	AUC	98.38	98.81	97.94	98.37	98.42	98.42	98.43	96.5	99.24	**99.25**	99.19	97.48
	F1	99.57	99.38	99.32	99.07	**99.69**	99.57	**99.69**	99.38	**99.69**	99.63	**99.69**	99.32
D11	AUC	**100**	98.36	99.86	99.46	99.9	99.86	99.73	99.46	**100**	99.8	99.96	99.96
	F1	99.99	99.96	**100**	99.99	99.98	99.98	**100**	99.97	99.99	99.99	99.99	99.99
D12	AUC	**100**	99.59	**100**	99.59	**100**	**100**	**100**	**100**	**100**	99.93	**100**	**100**
	F1	**100**	99.96	**100**	99.96	**100**	99.96	99.96	99.96	**100**	99.96	**100**	99.96
D14	AUC	**100**	**100**	50	50	**100**	**100**	50	50	**100**	0	93.45	**100**
	F1	**100**	**100**	0	0	0	0	0	0	0	0	0	0
D15	AUC	95.62	96.76	94.32	96.48	97.36	97.36	97.36	97.36	**99.87**	99.86	99.62	99.77
	F1	99.89	**99.91**	99.87	**99.91**	99.9	**99.91**	99.9	**99.91**	**99.91**	99.9	**99.91**	**99.91**
D16	AUC	**100**	**100**	96.3	**100**	**100**	**100**	**100**	**100**	**100**	**100**	0	**100**
	F1	**100**	**100**	66.67	**100**	**100**	**100**	**100**	**100**	**100**	**100**	0	**100**
D17	AUC	99.54	99.54	96.78	**100**	99.54	99.54	99.54	98.62	99.08	94.08	99.39	**100**
	F1	**99.99**	**99.99**	99.94	99.98	**99.99**	**99.99**	**99.99**	99.98	**99.99**	99.95	**99.99**	99.98
D19	AUC	99.9	**100**	94.34	99.9	**100**	**100**	**100**	**100**	99.79	97.92	**100**	99.89
	F1	99.9	**100**	99.49	99.9	**100**	99.9	**100**	99.9	99.9	99.59	**100**	99.9

Table III
THE COMPARISON OF METHODS ACCORDING TO AUC, F1 ON BALANCED DATA.

Algorithm		DT				kNN				SVM			
	Method	None	PCA	tSNE	AE	None	PCA	tSNE	AE	None	PCA	tSNE	AE
D01	AUC	91.09	93.31	81.22	91.31	91.47	91.21	91.05	91.73	**95.36**	94.63	93.93	94.76
	F1	93.59	94.61	90.59	93.92	93.35	93.27	93.41	94.4	**94.86**	94.7	94.6	94.81
D02	AUC	99.78	**100**	99.87	**100**	**100**	**100**	**100**	**100**	**100**	99.86	**100**	**100**
	F1	99.96	**100**	99.93	**100**	**100**	**100**	**100**	**100**	**100**	99.99	**100**	**100**
D06	AUC	91.95	91.09	91.19	94.89	98.15	98.15	98.09	98.09	97.63	97.68	**98.26**	97.33
	F1	92.78	92.9	93.16	93.9	93.97	93.97	94.02	93.77	92.93	92.1	**94.04**	92.93
D09	AUC	85.2	83.76	74.6	**88.68**	86.63	86.82	86.54	87.9	86.42	80.7	88.19	84.37
	F1	73.32	74.42	64.85	73.64	71.83	72.11	72.03	**74.52**	74.32	74.04	74.11	74.08
D13	AUC	**100**	**100**	99.15	**100**	99.94	**100**	99.94	**100**	**100**	98.72	99.93	**100**
	F1	**100**	**100**	98.77	**100**	99.73	99.73	99.73	99.86	**100**	89.55	99.86	**100**
D18	AUC	99.72	**100**	99.69	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	F1	99.91	**100**	99.56	**100**	99.91	**100**	**100**	**100**	99.78	99.47	**100**	99.78
D20	AUC	95.24	90.23	78.39	95.24	95.24	94.93	93.1	97.62	99.39	**100**	10.38	**100**
	F1	**95**	89.47	72.22	**95**	89.47	90	75.68	89.47	87.18	89.47	0	89.47

Specifically, for known malware on D01, D02 and D20, AE-3 are on the top due to the use of the bigger training dataset. On the remaining datasets (D06, D09, D13 and D18), although malware is knownn for AE-3 and known for others, AE-3 outperforms the original significantly. For example, on D09, the DT classifier is improved 2.69% of AUC and 0.56% of F1; the kNN is improved 0.65% of AUC and 2.64%.

VI. CONCLUSION

In this work, we leverage the autoencoder to produce sophisticated representations for IoT malware detection. Our experiments on a real dataset of IoT malware have confirmed two important points 1) the autoencoder is beneficial to compress IoT malware features and 2) our method can detect the unknown malware efficiently.

ACKNOWLEDGEMENTS

This research is funded by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2018.306.

Table IV
THE PERFORMANCE OF THE CLASSIFIERS TRAINED ON DATASETS OF D01-D03 WITH THE FEATURES GENERATED BY AE.

Dataset/	Algorithm	DT			kNN			SVM		
Malware	Method	None	AE	AE-3	None	AE	AE-3	None	AE	AE-3
D01	AUC	91.09	91.31	90.51	91.47	91.73	91.79	95.36	94.76	95.09
Hide and Seek	F1	93.59	93.92	93.24	93.35	94.4	94.3	94.86	94.81	94.77
D02	AUC	99.78	100	100	100	100	100	100	100	100
Muhstik	F1	99.96	100	100	100	100	100	100	100	99.99
D06	AUC	91.95	94.89	94.99	98.15	98.09	97.84	97.63	97.33	97.45
Kenjiro	F1	92.78	93.9	93.24	93.97	93.77	93.98	92.93	92.93	92.74
D09	AUC	85.2	88.68	87.89	86.63	87.9	87.28	86.42	84.37	84.17
Kenjiro	F1	73.32	73.64	73.88	71.83	74.52	74.47	74.32	74.08	74.04
D13	AUC	100	100	99.98	99.94	100	99.87	100	100	98.98
IRCBot	F1	100	100	99.73	99.73	99.86	99.46	100	100	99.04
D18	AUC	99.72	100	90.51	100	100	91.79	100	100	95.09
Mirai	F1	99.91	100	93.24	99.91	100	94.3	99.78	99.78	94.77
D20	AUC	95.24	95.24	96.34	95.24	97.62	97.07	99.39	100	95.12
Hide and Seek	F1	95	95	95.24	89.47	89.47	92.68	87.18	89.47	92.68

REFERENCES

[1] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad hoc networks*, 10(7):1497–1516, 2012.

[2] Dave Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011.

[3] Andrei Costin, Jonas Zaddach, Aurélien Francillon, and Davide Balzarotti. A large-scale analysis of the security of embedded firmwares. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 95–110, 2014.

[4] Andrei Costin, Apostolis Zarras, and Aurélien Francillon. Automated dynamic firmware analysis at scale: a case study on embedded web interfaces. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 437–448, 2016.

[5] Ang Cui and Salvatore J Stolfo. A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 97–106, 2010.

[6] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Communications Surveys & Tutorials*, 17(3):1294–1312, 2015.

[7] Andrei Costin and Jonas Zaddach. Iot malware: Comprehensive survey, analysis framework and case studies. *BlackHat USA*, 2018.

[8] Adrienne Porter Felt, Matthew Finifter, Erika Chin, Steve Hanna, and David Wagner. A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 3–14, 2011.

[9] Imran Makhdoom, Mehran Abolhasan, Justin Lipman, Ren Ping Liu, and Wei Ni. Anatomy of threats to the internet of things. *IEEE Communications Surveys & Tutorials*, 21(2):1636–1675, 2018.

[10] Nathalie Japkowicz, Catherine Myers, Mark Gluck, et al. A novelty detection approach to classification. In *IJCAI*, volume 1, pages 518–523. Citeseer, 1995.

[11] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[12] Van Loi Cao, Miguel Nicolau, James McDermott, et al. Learning neural representations for network anomaly detection. *IEEE transactions on cybernetics*, 49(8):3074–3087, 2018.

[13] Van Loi Cao, Miguel Nicolau, and James McDermott. A hybrid autoencoder and density estimation model for anomaly detection. In *International Conference on Parallel Problem Solving from Nature*, pages 717–726. Springer, 2016.

[14] Thanh Cong Bui, Minh Hoang, Quang Uy Nguyen, et al. A clustering-based shrink autoencoder for detecting anomalies in intrusion detection systems. In *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–5. IEEE, 2019.

[15] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.

[16] 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.

[17] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. Auto-encoder based data clustering. In *Iberoamerican congress on pattern recognition*, pages 117–124. Springer, 2013.

[18] Ly Vu, Van Loi Cao, Quang Uy Nguyen, Diep N Nguyen, Dinh Thai Hoang, and Eryk Dutkiewicz. Learning latent distribution for distinguishing network traffic in intrusion detection system. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[19] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[20] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[21] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[22] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[23] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[24] Stratosphere. Stratosphere laboratory datasets, 2015. Retrieved March 13, 2020, from https://www.stratosphereips.org/datasets-overview.

[25] François Chollet et al. Keras. https://keras.io, 2015.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.