Summer 2022

NetSec: Real-time and Scalable Malware Traffic Detection within IoT Networks

Ethan Weitkamp
Elizabethtown College, weitkampe@etown.edu

Yusuke Satani
Elizabethtown College, sataniy@etown.edu

Peilong Li
Elizabethtown College, lip@etown.edu

Jingwen Wang
Elizabethtown College, wangjingwen@etown.edu

NetSec: Real-time and Scalable Malware Traffic Detection within IoT Networks

Ethan Weitkamp*, Yusuke Satani*, Peilong Li, Jingwen Wang
Department of Computer Science
Elizabethtown College
{weitkampe, sataniy, lip, wangjingwen}@etown.edu

***Abstract*—Detecting malicious network traffic in real time has become a crucial requirement at smart communities for elderly care and medical facilities with the prevalence of Internet-of-things (IoT) devices. Existing machine learning based solutions for network traffic malware detection often fail to scale with the exponential increase of IoT devices at the facility and to detect malicious traffic with desirable low latency. In this paper we seek to fill the gap by designing a scalable end-to-end network traffic analyzing system that permits real-time malware detection. By leveraging distributed systems such as Apache Kafka and Apache Spark, the system has demonstrated scalable performance as the number of IoT devices grow. Using Intel's oneAPI software stack for both machine learning and deep learning models, the model inference speed is boosted by three-fold.**

***Index Terms*—Malware Detection; Machine Learning; Internet of Things (IoT); Feature Engineering and Analysis; Spark Streaming; OneDNN**

I. Introduction

Since the dawn of computing, there have been challenges with various types of malware threats attacking the computer machines and Internet of Thing (IoT) devices. These challenges have gotten worse both vertically (i.e. numbers and volumes) and horizontally(i.e. types and functionality) due to the evolution of technologies [1]. Vertically, signs currently point out that malware attacks on IoT devices are increasing. Two Zscaler studies show that malware attacks on IoT devices connected to corporate networks increased by 700% in 2022 during the pandemic [2]. Horizontally, it is possible to create smart and sophisticated malware because of the Internet, social networks, smartphones, IoT devices, and so on.

The state-of-the-art research focuses on the application of machine learning techniques for malware detection due to its ability to keep pace with malware evolution. Machine learning techniques have shown promising results from the literature [3][4][5] on malware detection targeted on various executable formats such as Windows Portable Executable (PE) and Android DEX (APK) .

With the implementation of encryption on websites, manipulated data can be very hard to track accurately. According to Google, 97% of all websites are encrypted [6] as of July 2020. The number of daily issued certificates keeps increasing linearly and is currently sitting at 1.5 million [7]. With this knowledge, it is more important than ever to prevent further attacks from occurring on encrypted traffic.

Although this problem seems complex in nature, many researches have advanced their methods for detection of malicious network traffic in order to prevent further attacks. Previous attempts to detect such traffic have been successful, but oftentimes take too much time for detection and are impractical for analyzing the large influx of data which has been made readily available. Alongside these complications, many researchers tend to use different datasets when comparing their newly created models. This confusion leads to different results when training and testing models. In the past, it has been difficult for researches to find appropriate, and complete datasets; This is where the common ground of network traffic analysis comes in.

Within this project we use the NetML and CICIDS2017 datasets which are publicly available and widely used by the network detection community to compare the results of their models. We also work toward improving previous detection rates by implementing Intel's DAAL and OpenVino to both Machine Learning and Deep Learning models. Our goals that we wish to accomplish are as follows. 1) We first implemented traditional Machine Learning and Deep Learning Models in order to get baseline scores 2) We then seek to accelerate the traditional Machine Learning and Deep Learning models with the use of Intel's DAAL and OpenVino. 3) Lastly, we conduct a thorough evaluation of multiple variables in order to determine improvement.

The organization of this paper is as follows. Section §II talks about all the previous works that have been accomplished by previous researchers. Section §III depicts the overall design of OpenVino and DAAL. Section §IV we talk about performance evaluation. Lastly, in Section §V we conclude the paper.

II. Related Works

The existing Deep Learning approaches either consider a limited number of attacks or use outdated datasets. Moreover, most of the previous work is specific to a particular task and suffers from under-fitting because of limited training samples. Table I presents a thorough literature

review of the state-of-art research works on malicious traffic detection within IoT networks.

To fill in the research gap, Sahu et al. [8] present a security framework for IoT attack detection using a hybrid Deep Learning model with two stages. Specifically, a CNN model first learns the features from the IoT network traffic, then the feature representation generated from previous step is used as the input of an LSTM model for attacks detection. However, whenever the network is scaled up by adding additional IoT devices, then an additional CNN module should be accompanied with the connected network switch.

In the previous work to develop an effective solution to detect and identify the stage of malware attacks using machine learning, Kaluphahana pointed out the two main challenges. The first one is the limitation of a traditional detection solution. The conventional solution depends on one network premises; this cannot detect attacks that originated at different network premises. Another problem is there is the case that devices might still operate for a long time, even after infection. To solve these problems, they proposed Adept, a security framework that detects bots attack and classifies them into attack stages across space and time. Firstly, Adept checks IoT traffic locally, respecting the normal profiles and generate alerts when the traffic does not match the profiles. Secondly, the security manager extract patterns of attack stages. Finally, Adept use alert-level and pattern-level information to classify the type and stages of attacks into categorical classification by using machine learning models, k-nearest neighbor (k-NN), random forest (RF), and support vector machine (SVM). In the machine learning section, they tried three solutions, Baseline, adept-v1, and adept-v2, and two sets of features, Alert Based Features and Pattern-Based Features. Alert-based features come from the alerts that are gathered at the security manager and have "Source and destination IP addresses,""direction"," protocol," "connection size," and "connection count" in both inward and outward directions. Pattern-based features are gathered at the second level of Adept, and their features are "IP and Port Orientations," Average Packet Size(inward and outward)"," Support,"" Source-to-Destination and Destination-to-Source Ratios"," port per IP"," Unique Attributes in Pattern", and "Unique Entities in Wild Cards". The first two solutions only use Alert Based Features.

The baseline model executes classification prior to FIM, a data mining technique that gets repeating patterns across a given set, and the Adept-v1 model runs models after FIM treatment. Compared to the first two solutions, the Adept-v2 model uses both Alert-based features and pattern-based features after FIM. This experiment shows that the Baseline model is slightly superior to the Adept-v1 model in all machine learning models. The author points out this is because of the decrease in sample size due to FIM. Moreover, the Adept-v2 model always shows the best score in all situations. In terms of machine learning techniques, SVM is a little inferior to others. In addition to those results, the experiment under the subnet-dependent case always provides better results than in the subnet-independent case.

Table II shows the best score of different machine learning and deep learning model in classifying large-scale offensive accesses on IoTs systems. The methods used vary from study to study, with some studies using only

TABLE I
LITERATURE REVIEW

[9]	Researchers have focused on developing ways to protect IoT devices by providing deep learning capabilities, frameworks, evaluation metrics, and big data technologies.	Their research covers lots of categories, but there is some lack of systemic literature review(SLR). Researchers do not set up questions they want to answer at the end of papers, and they need to emphasize using proper datasets because some datasets are old for applying machine learning models. These gaps occur because they focus on using machine learning and disregard ML in their papers.
[10]	The research aims for privacy and secure communication has provided papers about IoT security.	Researchers need to assess the quality of each researched paper and compare previous studies, but their primary interest is protecting IoT systems from attacks.
[11]	The survey by using ML and DL methods to improve security systems in IoT devices have covered confidentiality and integrity.	Their survey provides comprehensive conclusions. However, it lacks technical and scientific discoveries. In addition, the possibility b that IoT devices become part of a malicious attack and make an offensive attack also needs to be covered.
[12]	The research has presented an analysis of NIDS by using ML and DL, including a brief dataset.	The systematic literature review does not focus on IoT systems, detecting large-scale attacks, and classifying attack types.
[13]	The survey has provided specific details about IoT IDS, destination strategy, and security threats.	This research comes from a general survey. Therefore, the information about ML and DL solutions for protecting the IoT environment from large-scale attacks.
[14]	The researcher's scope of privacy protectionhas reached data protection, malware detection, and ML implementation challenges.	This research sticks to ML-based studies, so DL-based solutions are needed to be developed.
[15]	The research has supplied a detailed SLR review focusing on authentication, access control, and data protection.	This investigation lacks machine learning and deep learning models.
[16]	This research paper with focusing on machine learning and authentication is provided.	This investigation lacks machine learning and deep learning models.
[17]	Researchers have provided detailed research about IoT system protection, advantages, disadvantages, open issues, and future trends.	This research does not cover the detection of large-scale attacks, and analysis should be done related to the main topic. The main topic of this SLR is a survey performed on signature-based and anomaly-based IDS.
[18]	The researchers have presented a detailed study on security and privacy in IoT systems and also show different security challenges.	This research shows depth details on data privacy and challenges, but machine learning and deep learning methods should be reviewed in the paper.

TABLE II
MODEL ACCURACY IN SURVEYS

model	name model	AC	PR	RE	F1
1.	ANN	99.74	95.99	100	97.95
	SVM	99.86	91.98	100	95.82
2.	LSTM		99.98	100	99.99
	SVM		88.18	45.43	59.97

deep learning and others using only machine learning. Also, results are sometimes stated in writing rather than numerically because the researchers did not calculate classification accuracy.

In the survey in "HTTP Botnet Detection in IoT Devices using Network Traffic Analysis," researchers studied an algorithm for detecting IoT botnets in IoT devices. They employed a Deep Learning model called Artificial Neural Network (ANN) and machine learning called Support Vector Machine (SVM). In terms of accuracy score, the SVM score is 99.86 and superior to the ANN model(99.74).On the other hand, the ANN model scores 95.99 in precision ratio, and the score of the SVM model is 91.98. Therefore ANN is a better solution than SVM.

In another survey titled "A Long Short-Term Memory Enabled Framework for DDoS Detection", a new detection way named Long-Short-Term Memory("LSTM") is developed, and the feature of this model is to determine whether an attack or a benign attack can be detected from a relatively small amount of data. The conventional SVM model using machine learning has an accuracy score of 88.18, but the recall and F1 scores are below 50. On the other hand, the LSTM model using deep learning has an accuracy score of 99.98, and both recall and F1 are close to 100.

III. DESIGN

In this section, we present the design of our project by describing each step in our data pipeline including an overview, data collection and generation, offline model creation, and online model inference.

A. Overview to the Data Pipeline

The overall process can be seen in the following figure 1.

As displayed in figure 1, the data originates from IOT devices on a local network. These IOT Devices send their network traffic through a smart gateway that contains the next steps in the pipeline. On the smart gateway the network traffic is first sniffed by a program that then generates PCAP files from the given network traffic. These PCAP files are then sent to a Kafka producer, which then sends the data to a Kafka topic. The Kafka producer is important because it allows for scalability when adding more IoT devices to a network. Spark streaming is then used to ingest the data from the Kafka topic. Again, Spark streaming is used for the goal of keeping the entire pipeline scalable with as many IoT devices as possible.

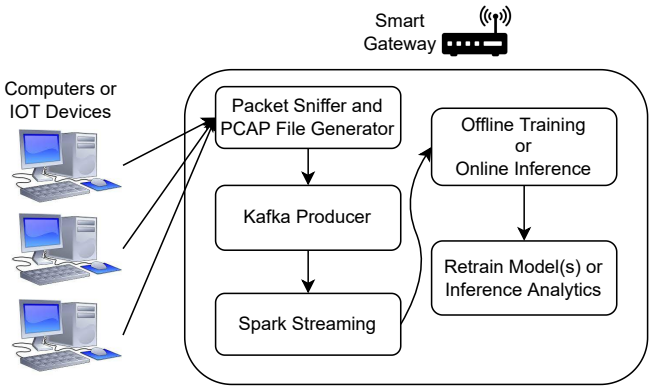

Fig. 1. Overall Design Pipeline

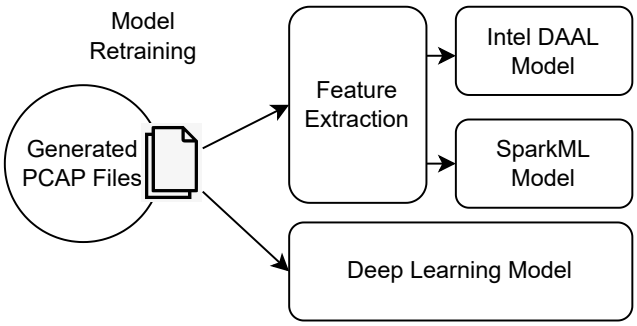

Fig. 2. Model Training Process

From there there the data can be used for retraining the ML or DL model(s) and/or using previously trained ML or DL models to make a malicious or benign inference on that network traffic.

B. Data Generation

The retraining process is shown in figure 2.

In order to train our different types of models we require a starting dataset and a way to retrain the model on new data to adapt to the new types of attacks. Whenever PCAP files are generated from the packet sniffer, they can be used later on to retrain the models with the actual network traffic happening on the network. The starting dataset we used to train the original models is the IoT-23 Dataset. We tested model accuracies for machine learning and deep learning models using different subsets of features provided by the IoT-23 Dataset and decided to use most of the features except for the host and recipient IP address and ports. We decided to remove these since they did not increase model accuracy, as they should not correlate with a malicious or benign attack and would only over fit out models. After the base models are trained using the existing dataset they can be re-trained as desired using the newly generated PCAP files. In the case of the machine learning models, the desired features will have to be extracted from the PCAP files before retraining along with any other data preprocessing steps.

C. Offline Model Training

As time goes on, new malicious attacks are being found and designed. What has been secure in the past must adapt in order to remain secure today and into the future. This is still true even in the case of machine learning. Our predictive models must be trained on more current data in order to maintain its security. Our design to support this is to have a built in way to retrain our different types of models with our own generated data. Data coming though the pipeline is saved in a way that our models can easily be retrained with later on. This means we would not have to rely on any existing datasets to keep our models up to date. We decided to test out five different machine learning models and five deep learning models. The machine learning model classifiers include random forest, decision tree, logistic regression, linear SVC, and Gaussian-NB. The deep learning models include artificial neural networks (ANN), convolutional neural networks (CNN), two dimensional CNN, long short term memory (LSTM), and a combination of CNN and LSTM. Each different model has its own strengths and reasons why we decided to test it that will be detailed in the next subsections.

D. Online Inference

In the overall pipeline after network traffic is converted into PCAP files and eventually picked up by Spark streaming, our previously offline trained model(s) can make an inference on that data. As mentioned previously, we have trained five types of machine learning models and five types of deep learning models.

Based on the timestamp of the data when it enters the Spark streaming step of the pipeline, all new data will be evaluated by any one of the already trained models. The machine learning models tend to have longer preprocessing times since feature extraction is required, but the inference/prediction time is quick as the models are not that complex. On the other hand, the deep learning models tend to have less preprocessing time, but a much longer inference time due to their complexity. Their complexity also means that they tend to be more accurate on complex data like ours.

Currently when an inference is made on network traffic, that is the end of the pipeline, but the inferences could then be used in any way desired.

E. Machine Learning and Feature Selection

Dataset features The IoT23 dataset has 1) timestamp; 2) unique identifier; 3) Originating endpoint's IP address; 4) Originating endpoint's TCP/UDP port; 5) Responding endpoint's IP address; 6)Responding endpoint's TCP/UDP port; 7) Transport layer protocol of connection; 8) service; 9) duration; 10) Originator payload bytes; 11) Responder payload bytes; 12) Connection state; 13) local_orig; 14) local_resp; 15) missed_bytes; 16) history; 17) Number of ORIG packets; 18) Number of ORIG IP bytes; 19) Number of RESP packets; 20) Number of RESP IP bytes; 21) tunnel_parents; 22) label; 23) detailed-label.

In the research, we applied two feature selections to machine learning models: 1) the `Full Feature Set`; and 2) the `De-identified Feature Set`. The first set includes every feature except timestamp and unique identifier. The De-identified feature set drops the originating endpoint's TCP/UDP IP address and port, and the responding endpoint's TCP/UDP IP address and port. The reason for removing them is that these information tends to mislead the machine learning models so that the models are biased in classifying the malicious traffic by identifying the IP addresses and ports.

We trained five machine learning models. 1) Random-Forest, 2) Decision-Trees, 3) Logistic-Regression, 4) Linear-SVC, 5) Gaussian-NB.

F. Deep Learning Models

On top of the previously mentioned machine learning models, we also trained five deep learning models. Those models included artificial neural network (ANN), convolutional neural network (CNN), long short-term memory (LSTM), 1D convolutional Neural Network (1DCNN), and lastly 2D convolutional Neural Network (2DCNN). The models also have a similar data preparation and preprocessing steps to the machine learning models, but provides much better accuracy in general.

Looking into the hyperparameters of our deep learning models, the activation function that we used for all of the ANN and CNN models is ReLU. The only one that is different is the LSTM model which uses tanh as the activation function and sigmoid as the recurrent activation function. For layers, the ANN only uses one layer, while all of the other models are using 2 layers. Our other model hyperparameters can be seen in the Table 1 below.

TABLE III
Deep Learning Hyper-Parameters

Parameter	Value
learning_rate	1e-3
decay_rate	1e-5
dropout_rate	0.5
dense_units	128
n_batch	100
n_epoch	1
filters	filters
kernel_size	4
strides	1
CNN_layers	2
clf_reg	1e-5

IV. Evaluation

This section of the paper evaluates the importance of our research. Specifically, we will firstly explain the datasets used throughout this research. Then we will talk about the specs used to train and test machine learning and deep learning models.

A. Datasets

The dataset mainly used for the research is the Aposemat IoT-23 [19] dataset. The IoT-23 dataset has 20 malware captures executed in IoT devices and three captures for benign IoT device traffic. Twenty malware files have the name of the malware sample executed in each scenario.

B. System Specifications

Throughout this research our data was being collected with a computer who's specs can be seen in Table IV. We used built in Python tools in order to collect our data, like training time, inference time, accuracy, and CPU usage.

TABLE IV
Experiment Platform Specification

Item	Specification
CPU	Intel i7 7700k
GPU	GTX 1080ti with 11Gb DDR5
Memory	32 Gb DDR4 @ 3600MHz
Storage	2Tb SSD
Software	Intel DAAL v2020.1
Host OS	Windows 10

C. Model Accuracy Comparison

We calculated the prediction accuracy from machine learning and deep learning models. In the machine learning inference section, we manipulate five machine learning models(Random-forest, Decision-Trees, Logistic-Regression, Linear-SVC, and Gaussian-NB) and two feature sets, the Full feature set, and the De-identified feature set. Figure 3 shows the outcomes from machine learning models. In addition, Figure 4 shows the outcomes from five deep learning models, Artificial Neural Network, Convolutional Neural Network, Convolutional Neural Network_2D, Long-Short-Term-Memory, and a combination of Convolutional Neural Network and Long-Short-Term-Memory.

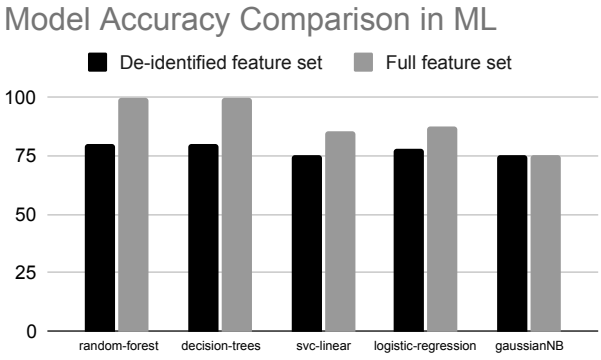

Fig. 3. Model Accuracy Comparison in ML

In the machine learning experiments, generally full feature set marked a higher score. The first two models, random-forest and decision-trees, almost reached 100%.

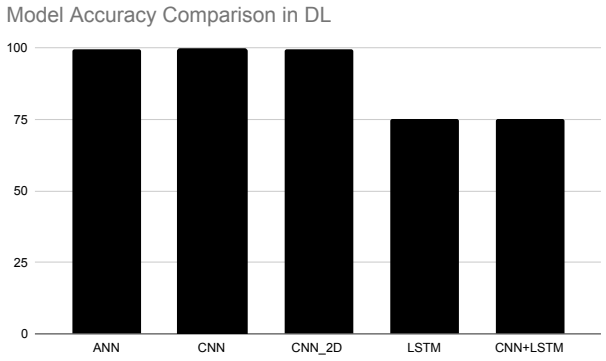

Fig. 4. Model Accuracy Comparison in DL

TABLE V
Inference Time for Machine Learning Models on Two Different Feature Sets

	De-identified feature set (s)	Full feature set (s)
Random-Forest	33.1795034408569	34.1987118721008
Decision-Trees	10.403615951538	10.7832884788513
Logistic-Regression	10.3037991523742	10.7082350254058
SVC-Linear	10.2995085716247	10.6838743686676
Gaussian-NB	10.4795570373535	10.8659715652465

The score of the De-identified feature set are 75% on the whole.

In the deep learning section, the first three models were marked as almost 100%. LSTM and LSTM+CNN, on the other hand, scored about 75%.

D. Inference Time Comparison

We measured the time of average time it takes for inference per each line from CSV files. As same with before experiment, we tested machine learning and deep learning. Figure 5 Shows the results from machine learning models, and Figure 6 Shows the results from deep learning models

This experiment shows that the random-forest model takes longer than other models. The reason seems that random-forest runs decision trees in parallel. The other models make inference per line around 2.6ms. In addition, the full feature set requires a little longer time to predict than the De-identified feature set.

This experiment shows that all deep learning models makes inference much faster than machine learning models. LSTM models take a little longer than other models, but only 0.4 ms per line

TABLE VI
Inference Time for Deep Learning Models

	Time per row (ms)
ANN	0.0563
CNN	0.1175
CNN2D	0.1015
LSTM	0.3778
CNN+LSTM	0.2258

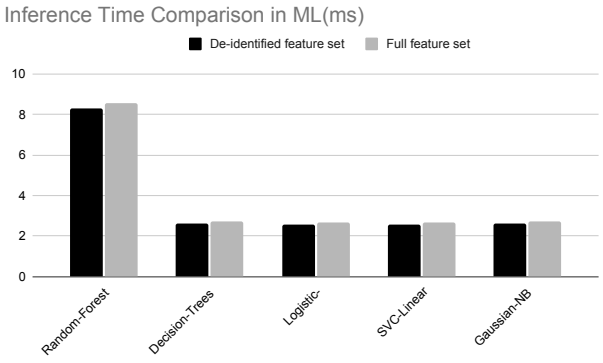

Fig. 5. Inference Time Comparison in ML (ms)

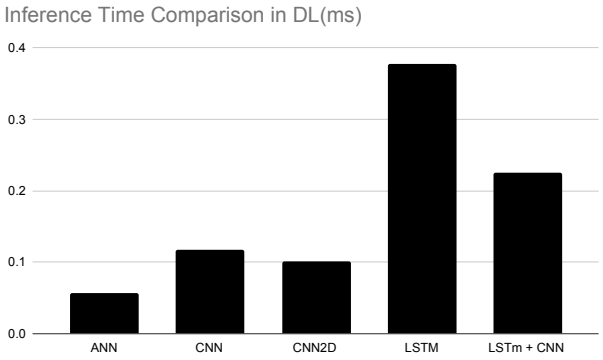

Fig. 6. Inference Time Comparison in DL (ms)

E. Scalability

Another key component of our research is scalability. We measured the change of inference time depending on the number of supporting devices. In the test, we added the number of devices from one to nine. Figure 7 Shows the outcome of the trial.

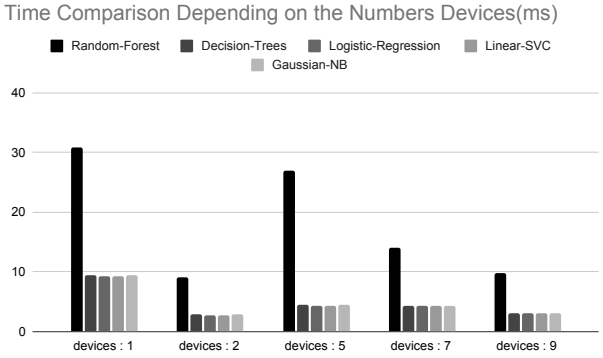

Fig. 7. Time Comparison Depending on the Numbers Devices(ms)

The experiment shows the scalability of our model. Generally, the inference time per line decreases as the number of connected devices gradually increases. Random-forest takes longer to predict than other models, same with the previous experiment.

V. Conclusion

In this project we achieve our goals of real-time and scalable malicious network traffic through the use of a Kafka producer and Spark streaming. Our models were trained using a publicly available dataset, IoT-23, which contains millions of network flows of information regarding malicious and benign network traffic. After performing prepossessing on the data, we can make inferences on this data using one of the many different models we have created. This process could be automated on a smart gateway to implement live-fed processed data into these models in order to create live detection over a local network.

We have made the source code for this project open-source for future research and is available to access through GitHub: https://github.com/BlueJayADAL/NetSec.

Acknowledgment

This work is supported in part by a grant from Intel Corporation, and a grant from Summer Scholarship, Creative Arts and Research Projects (SCARP) Program of Elizabethtown College.

References

[1] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, p. 102526, 2020.

[2] "Zscaler study confirms iot devices are a major source of security compromise, reinforces need for zero trust security." [Online]. Available: https://ir.zscaler.com/news-releases/news-release-details/zscaler-study-confirms-iot-devices-are-major-source-security

[3] C. Jindal, C. Salls, H. Aghakhani, K. Long, C. Kruegel, and G. Vigna, "Neurlux: dynamic malware analysis without feature engineering," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 444–455.

[4] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46 717–46 738, 2019.

[5] Y. Fang, Y. Gao, F. Jing, and L. Zhang, "Android malware familial classification based on dex file section features," *IEEE Access*, vol. 8, pp. 10 614–10 627, 2020.

[6] Google, LLC, "Https encryption on the web," 2020. [Online]. Available: https://transparencyreport.google.com/https/overview

[7] Let's Encrypt, "Let's encrypt stats," 2019. [Online]. Available: https://letsencrypt.org/stats/

[8] A. K. Sahu, S. Sharma, M. Tanveer, and R. Raja, "Internet of things attack detection using hybrid deep learning model," *Computer Communications*, vol. 176, pp. 146–154, 8 2021.

[9] M. A. Amanullah, R. A. A. Habeeb, F. H. Nasaruddin, A. Gani, E. Ahmed, A. S. M. Nainar, N. M. Akim, and M. Imran, "Deep learning and big data technologies for iot security," *Computer Communications*, vol. 151, pp. 495–517, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419315361

[10] T. Rajmohan, P. H. Nguyen, and N. Ferry, "Research landscape of patterns and architectures for iot security: A systematic review," in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2020, pp. 463–470.

[11] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (iot) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.

[12] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4150

[13] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804517300802

[14] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "Iot security techniques based on machine learning," 2018. [Online]. Available: https://arxiv.org/abs/1801.06275

[15] E. L. C. Macedo, E. A. R. de Oliveira, F. H. Silva, R. R. Mello, F. M. G. França, F. C. Delicato, J. F. de Rezende, and L. F. M. de Moraes, "On the security aspects of internet of things: A systematic literature review," *Journal of Communications and Networks*, vol. 21, no. 5, pp. 444–457, 2019.

[16] D. Androcec and N. Vrcek, "Machine learning for the internet of things security: A systematic review," 01 2018, pp. 563–570.

[17] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, "Intrusion detection systems in the internet of things: A comprehensive investigation," *Computer Networks*, vol. 160, pp. 165–191, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619306267

[18] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of things security: A top-down survey," *Computer Networks*, vol. 141, pp. 199–221, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618301208

[19] S. Garcia, A. Parmisano, and M. J. Erquiaga, "Iot-23: A labeled dataset with malicious and benign iot network traffic (version 1.0.0) [data set]." [Online]. Available: https://www.stratosphereips.org/datasets-iot23