**1 Introduction**

In the burgeoning field of the Internet of Things (IoT), the increase of interconnected devices has dramatically changed the way people interact with modern technology. It offers marked improvements in efficiency and utility in daily life. But IoT is not limited to private consumption, its deployment in industrial settings is also redefining manufacturing, logistics, and energy distribution realms. The Industrial IoT (IIoT), typified by sensors and automation, is aimed at enhancing operational effectiveness, predictive maintenance, and live monitoring across diverse sectors. Despite these advancements, the escalation in IoT and IIoT deployments is matched by an upsurge in cybersecurity vulnerabilities, such as IoT malware. These threats bear significant implications, with the potential to disrupt not only consumer devices but also vital industrial operations. A security breach could have dire outcomes, ranging from personal data exposure to the crippling of fundamental infrastructure, even escalating to widespread, coordinated attacks on crucial systems.

IoT malware presents distinct challenges that stem from the heterogeneity and widespread presence of devices, from household gadgets to advanced industrial controls. The design of many IoT devices, especially those in industrial contexts, emphasizes utility over security, rendering them susceptible to assorted cyber threats. For example, the infamous IoT Botnet Malware Mirai, heavily relies on the default system password of these Linux based IoT devices for self-propagation. Thus, the detection of IoT malware needs to be addressed immediately. Robust, efficient detection methods are essential to ensure the integrity, confidentiality, and availability of IoT infrastructures.

In tackling such security concerns, the application of unsupervised learning stands out as a feasible technique. Contrary to conventional training on labeled datasets, unsupervised learning utilizes the vast, unlabeled network data generated by IoT systems in their normal operation. This strategy affords several benefits in the IoT sphere. Primarily, the immense and diverse data generated by innumerable devices are not labeled, labeling these data for supervised learning can be slow and resource intensive. Additionally, IoT's dynamic environment and evolving malware methodologies quickly outdated static, labeled datasets. In contrast, unsupervised learning can nimbly accommodate shifts and trends within real-time data streams, offering a more robust and adaptive solution for malware detection.

In this study, we examine the nuances of IoT malware data and the function of detection frameworks in protecting IoT devices. We evaluate the pros and cons for unsupervised learning techniques and illustrate their potential in meeting the complex requirements of IoT security in a world of growing connectivity. We aim to contribute in this study as follows:

* Using Unlabeled Data: We propose anomaly detection techniques for detecting IoT malware, which capitalizes on the unlabeled data from IoT devices, mitigate the restrictions associated with traditional labeled data practices.
* Exploring on Positive Training: Our proposed methodology concentrates on training with non-malicious (positive) data, which simplifies the training process and simulates the data from real world environments which malicious data are not always accessible.
* Prioritizing Deployment Compatibility: Our design strategy emphasizes compatibility and easy deployment on a wide array of IoT devices, irrespective of their computational limitations, to guarantee broad-based efficacy without compromising device performance.

**2 Related Work**

**2.1 Unsupervised learning**

Most previous studies have traditionally relied on supervised learning methods, necessitating well-labeled datasets for training to build hybrid models for static or dynamic analysis. This conventional approach, while beneficial in structured environments, encounters significant challenges in the IoT context, where labeling vast amounts of data is impractical due to their sheer volume and heterogeneity. Only a limited number of studies have ventured into exploring the use of unlabeled data which is our study aims to address because there remains a significant research void in comprehensively employing these unsupervised methods for IoT malware detection that becomes more important given the practical difficulties in obtaining labeled data in real-world settings.

(J. Jeon, J. H. Park and Y. -S. Jeong)[1] has performed a dynamic analysis for IoT malware detection using CNN in a virtual environment, which detects malware by integrating feature selection and classification phases into a single phase. Eventually, they achieved a 99.28% accuracy using 1000 malware samples and 401 benign samples. The model they proposed proven to be effective in a controlled environment, however, in real life scenario, the network traffic data could be way more than this in a short time range. To detect the malware in real time, analyzing and labeling the data is also not an option. In our study, the dataset we use is also labeled, but the training will be unsupervised and there will be over 10 million samples used for training to make the model robust.

**2.2 One Class Classification**

Detecting malware is essentially an anomaly detection problem, however most previous studies have been using both malicious and benign data for training which is logical because the results will be more accurate. However, in real world situations, malicious data are not always accessible especially for a real time malware detection system, malicious data are most likely not available, the training must be done only based on the normal benign data. Our study, therefore, delves into an underexplored territory in IoT cybersecurity research — the exclusive use of unlabeled positive data for training detection models. ~~While some studies have hinted at the feasibility of training on non-malicious data, a systematic and focused exploration in this area is scarce.~~ We aim to build on the limited existing research and advance the field by proposing methodologies that are specifically tailored to exploit unlabeled data, ensuring both the practicality and effectiveness of malware detection in diverse IoT settings. This focus on unlabeled data represents a significant departure from traditional methods and positions our study as a crucial contribution to the evolving landscape of IoT cybersecurity.

**3 Data**

**3.1 Dataset**

The dataset used in this project is called IoT-23[1]. This dataset is released in January 2020 and compiled by the Stratosphere Laboratory at CTU University in the Czech Republic and funded by Avast Software. This dataset comprises 23 different network traffic captures (scenarios), 20 are from IoT devices infected with malware, each labeled with the malware name, and 3 from benign IoT devices including a Philips HUE smart lamp, an Amazon Echo, and a Somfy smart door lock. These captures were made on real hardware devices in a controlled environment with normal internet access. The dataset includes detailed information such as README.md files, original pcap files, Zeek conn.log.labeled files with network flow and behavior labels, and other analysis files. The coon.log.labeled files which are the connection logs based on NetFlow analyzed by Zeek. Different than network traffic in the pcap files, NetFlow is a network protocol developed by Cisco for collecting IP traffic information and monitoring network flow. Cisco standard NetFlow version 5 defines a flow as a unidirectional sequence of packets that all share seven values which define a unique key for the flow. The seven attributes are: ingress interface (SNMP ifIndex), source IP address, destination IP address, IP protocol number, source port for UDP or TCP (0 for other protocols), destination port for UDP or TCP (type and code for ICMP, or 0 for other protocols), and IP Type of Service[2].

The following table shows the information of the 20 malware infected captures, capture-7-1 and capture-36-1 are being used because these 2 captures contain a decent number of benign samples without being too big. Some other captures are either too big or do not contain enough benign samples. And the 3 benign captures provided in this dataset only have a few thousand samples which is too small compared to over 20 million malicious samples, so the benign samples also must be extracted from the infected captures. Capture-7-1 is the capture infected by Linux version of Mirai botnet, it contains 75,955 benign samples and 11,378,759 malicious samples. Capture-36-1 is the capture infected by Okiru botnet, which is also a variant of Mirai, it contains 2,663 benign samples and 13,642,435 malicious samples. After extracting the benign samples there are total of 78,618 benign samples.

**3.2 Preprocessing**

Both captures are combined and processed together to ensure that all the features are consistent, different labels are assigned to samples from different captures, so they can be easily split later. The conn.log.labeled file can be treated as a csv file that has 21 features. 20 features are the top 20 features shown in image 1. And the last feature is the detailed label. The original detailed label specified the type of malware attacks of each sample, however in this project the label will be binary, which is benign or malicious, the detailed label will not be used. Time stamp and uid are also not used as features so they are removed. Features local\_orig and local\_resp are completely empty, services contain over 99% missing values so they will also be removed. In this rest 15 features, there are 7 numeric features and 8 categorical features, and the categorical features will be encoded into numeric, so the algorithm can process.

Except connection state and connection history which are a bit complicated as image 1 shows, the rest categorical features are IP addresses and port numbers. For the port numbers, I categorize them into 3 labels which are well-known ports (0-1023), registered ports (1024-49151), and dynamic/private ports (49152-65535). And for IP addresses, since there are no IPv6 addresses in these 2 datasets I used, each address is split into 4 numbers as 4 new numeric features. According to previous study[4], split IPv4 addresses into 4 numeric features perform better than one-hot encoding. And unlike this study which has a small amount of different IP addresses, dataset used in this project has too many different IP addresses for one-hot encoding.

Dealing with conn\_state and history can be tricky and requires some deeper knowledge on networking and cybersecurity field. For conn\_state, it is possible to directly one-hot encode the labels because there are limit number of labels. However, for history, the value can be a random string with random uppercase and lowercase letters from the list in image1. And even conn\_state has a limit number of labels, adding more than 10 dimensions to the dataset is still not ideal, and a few labels only has single digit appearances compared to the most appeared label which has more than 25 million appearances. So, for conn\_state, I have combined labels contains RST to a new label RST because they all represent an unusual reset request for the TCP connection. I have also combined S0 and SF into one label because they basically represent the same thing with byte transformed difference.

For connection history, a deeper analysis is required. For example, letter s, h, a, d, f usually represents benign traffic and letter r, c, g, t, i, q could indicate malware traffic. And I basically combine values containing those letters into a new label, for example, if the value contains letter h, or H, I will combine these values into a new label h. However, by doing this the order of combining matters because a value could contain multiple letters. I would not say this is the best way to deal with this feature, but a better method requires more professional knowledge in this field. My order of searching the letters is “^, i|I, r|R, d|D, c|C, q|Q, t|T, h|H, f|F, and a|A”. Since not all searches have a result, only values contain r|R, d|D, and h|H are combined. After the combination I noticed that 3 labels h, A, and F only have single digit appearances and they represent benign normally, so I combine these 3 labels into one new label B.

At this point, I have 5 labels in both conn\_state and history, 3 labels in proto, one-hot encoding can now be performed on them. With a further look at the numeric features, duration, orig\_bytes, and resp\_bytes, all contain over 99% empty values, so they will also be removed. Finally, I have a dataset with 33 features including the target as table 1 shows. At the end, 2 types of malwares and the benign samples are split into 3 dataset which are benign.csv, marai.csv, and okiru.csv.

|  |  |  |  |
| --- | --- | --- | --- |
| Features  category | Number of Features | Features  category | Number of Features |
| target | 1 | proto\_icmp | 1 |
| id.orig\_h | 4 | proto\_tcp | 1 |
| id.resp\_h | 4 | proto\_udp | 1 |
| missed\_bytes | 1 | conn\_state | 5 |
| orig\_pkts | 1 | history | 5 |
| orig\_ip\_bytes | 1 | id.orig\_p | 3 |
| resp\_pkts | 1 | id.resp\_p | 3 |
| resp\_ip\_bytes | 1 |  |  |

Table 1.

A screenshot of a computer

Description automatically generated

Image 1. [3]

**4 IoT Malware Detection**

Most previous studies have traditionally relied on supervised learning methods, necessitating well-labeled datasets for training to build hybrid models for static or dynamic analysis. This conventional approach, while beneficial in structured environments, encounters significant challenges in the IoT context, where labeling vast amounts of data is impractical due to their sheer volume and heterogeneity. Only a limited number of studies have ventured into exploring the use of unlabeled data which is project aims to address because there remains a significant research void in comprehensively employing these unsupervised methods for IoT malware detection that becomes more important given the practical difficulties in obtaining labeled data in real-world settings.

Detecting malware is essentially an anomaly detection problem, however most previous studies have been using both malicious and benign data for training which is logical because the results will be more accurate. However, in real world situations, malicious data are not always accessible especially for a real time malware detection system, malicious data are most likely not available, the training must be done only based on the normal benign data. This project delves into an underexplored territory in IoT cybersecurity research — the exclusive use of unlabeled positive data for training detection models and aim to build on the limited existing research and advance the field by proposing methodologies that are specifically tailored to exploit unlabeled data, ensuring both the practicality and effectiveness of malware detection in diverse IoT settings. This focus on unlabeled data represents a significant departure from traditional methods and positions our study as a crucial contribution to the evolving landscape of IoT cybersecurity.

**4.1 Traditional anomaly detection**

One-class SVM and isolation forest are the focus in this section, both algorithms are proven to be effective for anomaly detection problems. One-Class SVM is adapted from the traditional SVM, specifically designed for anomaly detection. It works by finding a decision boundary that separates most of the data points from the outliers. Isolation Forest is another unsupervised learning method for anomaly detection. It isolates anomalies instead of profiling normal data points. It works by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. In most cases, one-class SVM proven to have more accuracy than isolation forest, however, in this case where the dimensions are high, SVM could encounter significant performance decrease, especially for the non-linear kernels. Isolation forest on the other hand has negligible performance impact with higher dimensions.

Because the dataset is extremely unbalanced, only a subset of malicious samples will be selected. Two types of training have been done on both algorithms, the first type is using both benign and malicious samples for training. 80% of the benign samples are selected for training, and 10% of the number of benign samples selected for training will be selected from the malicious samples to from a 10:1 benign and malicious ratio in the training set. The rest of the benign samples are being used for testing, and the same number of malicious samples are being selected from a 1:1 ratio in the testing set. The second type is one class training, which only uses benign samples in the training data. In this case, only 80% benign samples will be used for training, and the rest 20% combined with the same number of malicious samples will form the same 1:1 testing set. Grid search is also used to find the best parameters for the model.

Testing the model using a different dataset is also crucial to evaluating the robustness of the model. In this project, because the benign samples are the same, two training sets and test set will be created based on the source of malicious samples. The model trained using Mirai will also be tested using the Okiru test set.

**4.2 Deep learning auto encoder**

Auto encoder is a structure of deep learning anomaly detection. It typically consists of two main parts, an encoder, and a decoder. The encoder compresses the input into a latent-space representation, and the decoder reconstructs the input data from this representation. The aim is to learn a representation that captures the most important features of the data and detect anomalies by the reconstruct error.

**5 Results and Comparison**

Because of the nature of malware detection problems or most anomaly detection problems, high recall is the focus of the result because we need to detect as much as malwares possible, and a reasonably loss of the precision score can be accepted.

**5.1 Multi-class training**

Table 3 shows all the results of using mixed training data. The default isolation forest has a good precision but low recall, after tunning the parameters contamination and n\_estimators, the recall increased to 98% with precision drop to 72%. On the other hand, the default one-class SVM has 98% precision and a 71% recall, however after tunning, the recall drops to 34% which is unexpected. The reason caused this issue is most likely that the grid search is based on cross validation of the training set instead of the testing set. This is improved in the positive training section, a customized grid search based on testing set is used for positive learning.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Training  Set:  Mirai | Isolation Forest  Default | | One-class SVM  Default | | Isolation Forest  Tunned | | One-class SVM  Tunned | |
| Precision | Recall | Precision | Recall | Precision | Recall | Precision | Recall |
| Mirai test set | .97 | .35 | .98 | .71 | .72 | .98 | .93 | .34 |
| Okiru test set | / | / | .72 | .94 | .98 | .76 | / | / |

Table 3.

**5.2 One-class training**

Table 4 includes all the results in this section. The default isolation forest and one-class SVM both have very high precision and extremely low recall. However, after using the customized grid search on the Miri test set searching for the best parameters, both models reached a 100% recall which is very ideal, and more surprisingly, the precision for the isolation forest using 0.1 contamination and 100 estimators has 100% recall while maintaining 99% precision. Image 2 shows the confusion matrix for test sets using the tunned isolation forest model. One-class SVM, however, with the RBF kernel, the recall can reach 100% but the precision drops to 50% which is not ideal. And the performance of using the RBF kernel is significantly slower than isolation forest and linear SVM, each iteration takes more than an hour to run compared to an average 6 mins of linear SVM, and less than 10 secs of isolation forest.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Training  Set:  Benign | Isolation Forest  Default | | One-class SVM  Default | | Isolation Forest  Tunned | | One-class SVM  Tunned | |
| Precision | Recall | Precision | Recall | Precision | Recall | Precision | Recall |
| Mirai test set | .99 | .01 | .98 | .12 | .99 | 1.0 | 0.5 | 1.0 |
| Okiru test set | .60 | .00 | / | / | .99 | .99 | / | / |

Table 4.

A black and white squares with numbers

Description automatically generatedA black and white squares with white numbers

Description automatically generated

Mirai Okiru

Image 2.

**5.3 Auto encoder**

Because I used different datasets for the auto encoder, I have also included results of isolation forest using the same dataset. One-class SVM fails to complete because this dataset has 45 features with over 10 million samples for both benign and malicious. Both algorithms achieve decent results, because the dataset is huge, achieving 100% recall is almost impossible, however the auto encoder still has so much room to be optimized. For example, ideally the layers for the auto encoder should be CNN or RNN, the reason I used DNN is because it does not require data transmission from 1D to 2D or 3D.

|  |  |  |
| --- | --- | --- |
| Algorithms | Precision | Recall |
| Isolation Forest | .95 | .94 |
| One-Class SVM | / | / |
| Autoencoder | .97 | .96 |

Table 5.

**8 Conclusion and Future Works**

In conclusion, for both mixed training and positive training tunned isolation forest has better performance than one-class SVM. For positive training, the isolation forest with 0.1 contamination and 100 estimators, even reach 100% recall and 99% precision value. Which proves that positive training for malware detection is feasible. A basic DNN auto encoder with large dataset also achieves a decent result.

However, there is still a lot that can be improved. First, in this project, only one test set for each case is being used and cross validation are not very straight forward to use for unsupervised learning, to further evaluate the robustness of the models, different subset of malicious samples should be selected for the test set to create more test sets and evaluate the overall scores. Second, dimension reduction could be tried especially for SVM, lower dimension could dramatically increase the performance of non-linear SVM and might also yield better results. Third, deep features might be a better way to do feature engineering in this case without requiring any professional knowledge in the networking and cyber security filed. Last but not least, there is still a lot that can be improve for the auto encoder, I believe, at the end, especially with a large amount of training data, it could perform better than traditional unsupervised approaches.

**References**

[1] Sebastian Garcia, Agustin Parmisano, & Maria Jose Erquiaga. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set]. Zenodo. http://doi.org/10.5281/zenodo.4743746

[2] https://en.wikipedia.org/wiki/NetFlow

[3] https://corelight.com/products/zeek-data/

[4] Shao, Enchun (2019). Encoding IP Address as a Feature for Network Intrusion Detection. Purdue University Graduate School. Thesis. https://doi.org/10.25394/PGS.11307287.v1

[5] J. Jeon, J. H. Park and Y. -S. Jeong, "Dynamic Analysis for IoT Malware Detection with Convolution Neural Network Model," in IEEE Access, vol. 8, pp. 96899-96911, 2020, doi: 10.1109/ACCESS.2020.2995887.

[6] S. Nõmm and H. Bahşi, "Unsupervised Anomaly Based Botnet Detection in IoT Networks," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 2018, pp. 1048-1053, doi: 10.1109/ICMLA.2018.00171.