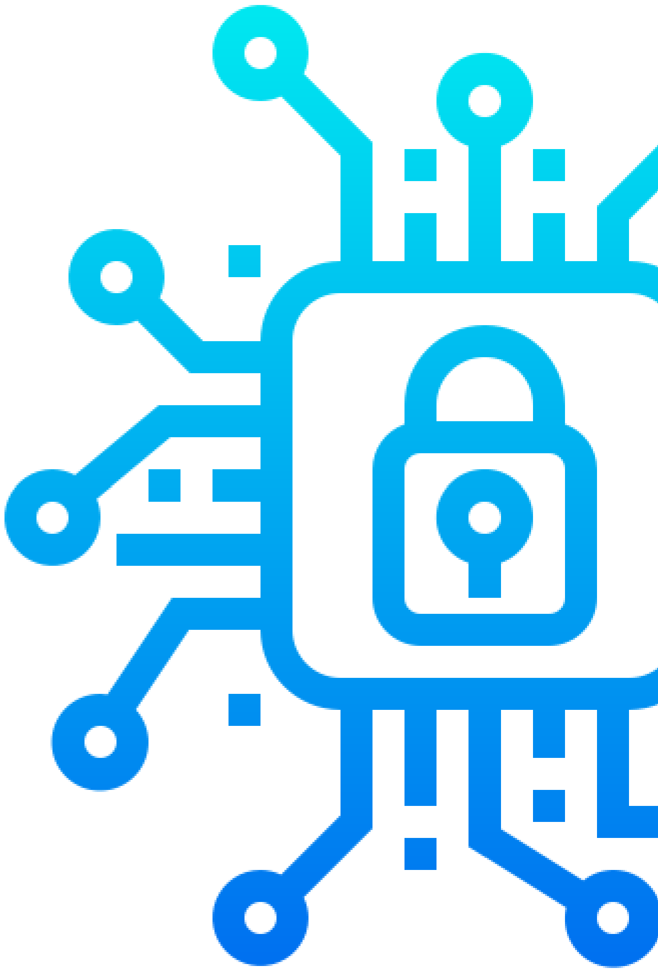
IoT Smart Home Network Malware Detection Using Deep Neural Networks and Transfer Learning



Matthew Ocando

2022-12-05

Table of Contents

Introduction 2
Problem Statement 2
Background 2
Proposed Solution 3
Methodology 3
Results 5
Discussion 9
Conclusion 10
References 11

Introduction

In the growing age of technology, IoT is pushing to become a commodity that makes its way into the homes and daily routines of the average person. With the growing prevalence of these smart devices in everyday households, there is an equally increasing threat of malware taking advantage of them to serve a criminal purpose. IoT cyber security has made great strides in recent years to increase the security of these IoT devices and make them less prone to exploitation, but just as security has progressed, so has malware development. Numerous IoT devices contain security features built within their software that are provided by manufacturers as a guarantee of the safety of the device.

In the ongoing battle between cyber criminals and security, it has long been accepted that prevention of all forms of attack is a relatively impossible task, as there will always exist some vulnerability that a malicious user could take advantage of. As such, it is understandable that prevention is only a first line of defence when it comes to avoiding cyber attacks in the IoT space - the second being attack detection. This area is one of greatly unexplored potential, but unfortunately has many caveats that result in complexity of a solution.

Due to the nature of IoT, there exist many devices in a single network (or home, for example) that come from different manufacturers, use different software to run, and use unique instructions to operate. This diversity introduces a great deal of difficulty when attempting to find a malware detection solution that unifies the IoT smart home device space. However, it is notable that IoT devices all share one common protocol that is necessary for their proper operation - the network protocol stack [1] . All internet connected IoT smart home devices must relay their information in a unified manner that can be relayed by wifi routers. It is therefore intuitive to take advantage of this common ground, and attempt to implement malware detection at the network layer.

Problem Statement

The process of malware detection in the IoT smart home space constitutes the focal point of this project. The problem being addressed in this report is the need for an easily scalable and unifying malware detection solution for IoT smart home network devices. Ideally, this solution should be easy to deploy and keep up-to-date as new information about IoT malware becomes readily available.

Background

Numerous studies have proposed using Deep Neural Networks (DNN) for the classification of malware data in past studies ([2], [3], and many others). The nature of DNNs allows them to essentially serve as function estimators for extremely complex problems [4] where it would otherwise be impossible to determine an exact model for the underlying relationships in a system. As such, complex tasks like detecting malware are ideally suited for this type of approach where the features and parameters of the problem at hand are too vast or too vague to explicitly define.

Moreover, machine learning approaches in general are becoming an extremely popular solution due to the surge of Big Data in recent years [5]. With exponential advancement in computing, as well as steady growth in the field of Data Science, machine learning models are quickly replacing traditional hard-coding for tasks that were previously deemed unfeasible. It is therefore by no surprise that these approaches are also overtaking the area of malware detection.

When it comes specifically to IoT, machine learning and IoT go hand-in-hand. The vast data that passes through any IoT system is ideal for the deciphering capabilities of a machine learning model [6]. Additionally, with ever increasing production of data, the models that support these systems will only get better at their jobs. With reference to the project at hand, IoT smart home networks exchange thousands of data packets every second that contain vital information regarding the workings of each device - a perfect application for a machine learning approach like DNNs.

Proposed Solution

The work in this project proposes to make the following contributions:

1. To implement a Deep Neural Network (DNN) malware classifier capable of classifying IoT smart home device network traffic as either malware or benign.
2. To devise a methodology for updating the trained DNN with new malware data, without the need for retraining the entire model on all the data.

For this project, we will be using part of the full IoT-23 dataset [7]. This dataset contains Zeek network logs from three IoT smart home devices - an Amazon Echo home intelligent personal assistant, a Philips HUE smart LED lamp, and a Somfy smart doorlock. Additionally, this dataset contains various network captures from common IoT malware deployed onto a Raspberry Pi 2, acting as a simulated IoT device.

Methodology

The process of developing the DNN classifier is separated into four major phases. The first phase involved modelling the smart home network space being studied in our reference dataset [7]. Due to the nature of computer networks, graph modelling was found to be the technique most suitable for the IoT space in question. The second phase involved preprocessing of the Zeek network captures used to train the DNN malware classifier. This involved formatting the log files into a manner that would be conducive for training, as well as managing certain features of the dataset either through data encoding or omitting from the final training data. Certain features were not used during training after using insight into the dataset to understand the relevance of some features with regard to malware classification. The third phase was the training step of the DNN, during which certain network parameters were adjusted to produce an optimal performance. The final phase involved adding new network capture data to our original training dataset, and adjusting the existing model to include the new data using transfer learning.

A. IoT Smart Home Network Modelling

As described in the introduction to the methodology, graph modelling was used for the IoT network in question. Due to the nature of the data capture for the dataset, this graph model was separated into two variations. The first illustrated the network arrangement for capturing the networking logs of the three smart home devices. The second illustrated the arrangement for capturing the malware samples on the Raspberry Pi 2.

Additionally, the adjacency matrices for each model were calculated.

B. Zeek Network Capture Preprocessing

The method used for preparing the network capture data for the learning phase can be described as follows:

1. The "ParseZeekLogs" library available through the python PIP package manager was used to compile the Zeek network logs from each device/scenario into a single pandas dataframes for easy data manipulation.
2. Any rows containing improperly formatted data after extraction from the log were dropped.
3. Empty features (columns where every entry was either the same value or no value whatsoever) were removed from the dataframe as they had no influence on training.
4. GIven that the target classifier is only responsible for identifying benign versus malicious traffic, the details regarding the type of malware process that each malicious capture was associated with were removed.
5. Features containing strictly metadata for Zeek were removed (capture identifiers, timestamps etc.).
6. The feature responsible for listing flags attached to package exchanges was removed. This feature was categorical but with extremely high cardinality and thus posed a problem for proper numerical encoding. Additionally, network packet flags have little correlation to the detection of malware as they are a standard part of any network exchange that can be randomly assigned to packets.
7. Any features containing strictly IP address information were removed. Malware attacks can originate from any randomly assigned IP address and thus this feature gave little to no useful information to our DNN.
8. Any remaining categorical features were one-hot encoded.
9. The remaining numerical columns were formatted from strings into numerical data types.
10. Finally, 10% of the training data was set aside for the upcoming transfer learning phase.

C. Training the DNN Malware Classifier

For the training, the fastai python library [8] (built on PyTorch) was used. The methodology proceeded as follows:

1. Our target classification label was defined as either benign or malware.

2. A fastai DataLoader used for managing the input dataframe was prepared. The default 80%-20% split for training versus validation data was used. Additionally, a normalisation was applied on the input dataframe.
3. The DNN learner was prepared. The layers were fully connected (dense), linear, and separated by ReLU activation, with batch normalisation and dropout layers enabled.
4. Training was attempted on a variety of hidden layer dimensions, and accuracy scores were observed. The dimensions that yielded the highest accuracy were chosen.
5. The chosen DNN dimensions were re-trained over 10 epochs, after which the changes in loss and accuracy were observed. The DNN was retrained over the minimum number of epochs that produced the highest accuracy while ensuring that the validation loss did not increase significantly as to indicate overfitting.
6. The resulting metrics (accuracy, training loss, and validation loss) were recorded. A confusion matrix was also generated and analysed.
7. Further metrics were produced from the confusion matrix.
8. The trained model was saved and exported for future use.

D. Tuning the DNN for New Data Using Transfer Learning

The final phase involved using the concept of transfer learning to adjust the existing parameters of the trained DNN classifier to accommodate for new data. The procedure was as follows:

1. The 10% of data set aside during the preprocessing phase was concatenated to the rest of the training data.
2. The model was reloaded from the exported file, and a new DataLoader containing the 10% additional data was created to replace the older one. This DataLoader split the new concatenated data into 80% training and 20% validation data.
3. All model layers aside from the final layer were frozen - these frozen layers could not have their parameters altered.
4. The model was trained on the now extended training data for a single epoch.
5. The remaining layers were unfrozen, and the model was trained for another 10 epochs with a small learning rate (~0.001).
6. The resulting metrics on the validation set were recorded, and a confusion matrix was plotted.
7. Further metrics were calculated from the confusion matrix.
8. The updated model was saved and exported.

Results

A. Graph Model of IoT Smart Home Network

The first graph model illustrates the relationship between the various nodes existing in the network used to capture benign network packets for the dataset relevant to this project. The benign scenario refers to the capture of packets from the three IoT smart home devices shown in the following diagram. The Zeek node represents the computing node used to capture and record the network packets for each device.

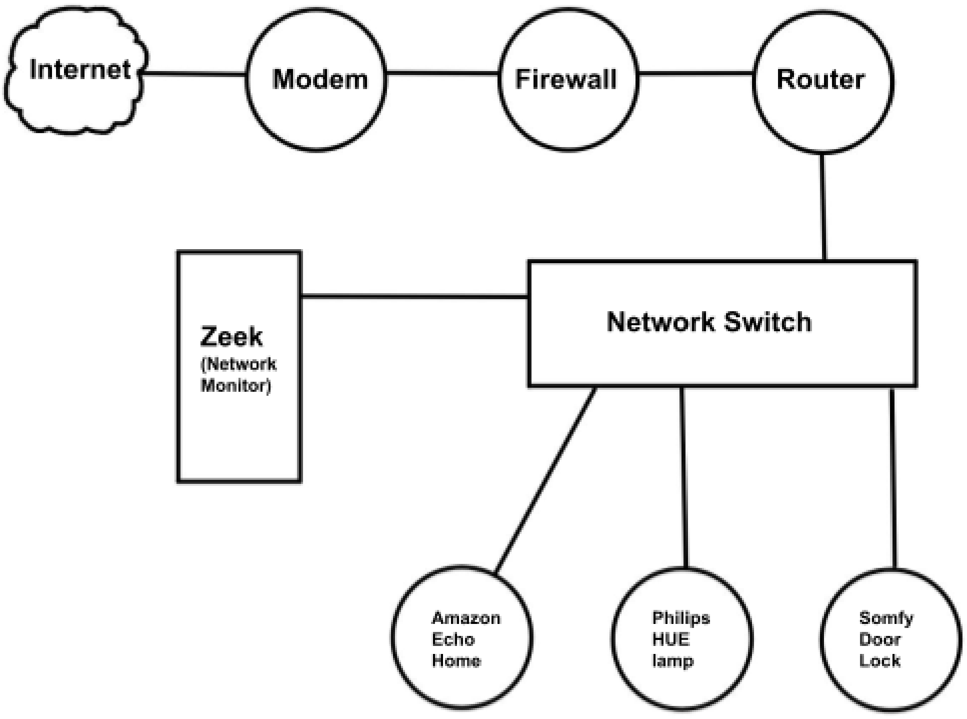


Figure 1.0 - Graphical Model for IoT Smart Home Network Illustrating Packet Capture Setup (Benign Scenario)

Additionally, the following illustrates the adjacency matrix for the graph shown in figure 1.0.

	Internet	Modem	Firewall	Router	Network Switch	Zeek	Amazon Echo Home	Philips HUE Lamp	Somfy Door Lock
Internet	0	1	0	0	0	0	0	0	0
Modem	1	0	1	0	0	0	0	0	0
Firewall	0	1	0	1	0	0	0	0	0
Router	0	0	1	0	1	0	0	0	0
Network Switch	0	0	0	1	0	1	1	1	1
Zeek	0	0	0	0	1	0	0	0	0
Amazon Echo Home	0	0	0	0	1	0	0	0	0
Philips HUE Lamp	0	0	0	0	1	0	0	0	0
Somfy Door Lock	0	0	0	0	1	0	0	0	0

Figure 2.0 - Adjacency Matrix for IoT Smart Home Network (Benign Scenario)

Similarly, the following graph model illustrates a similar relationship, but for the test scenario in which a malware sample is introduced onto a Raspberry Pi 2 acting as an IoT node.

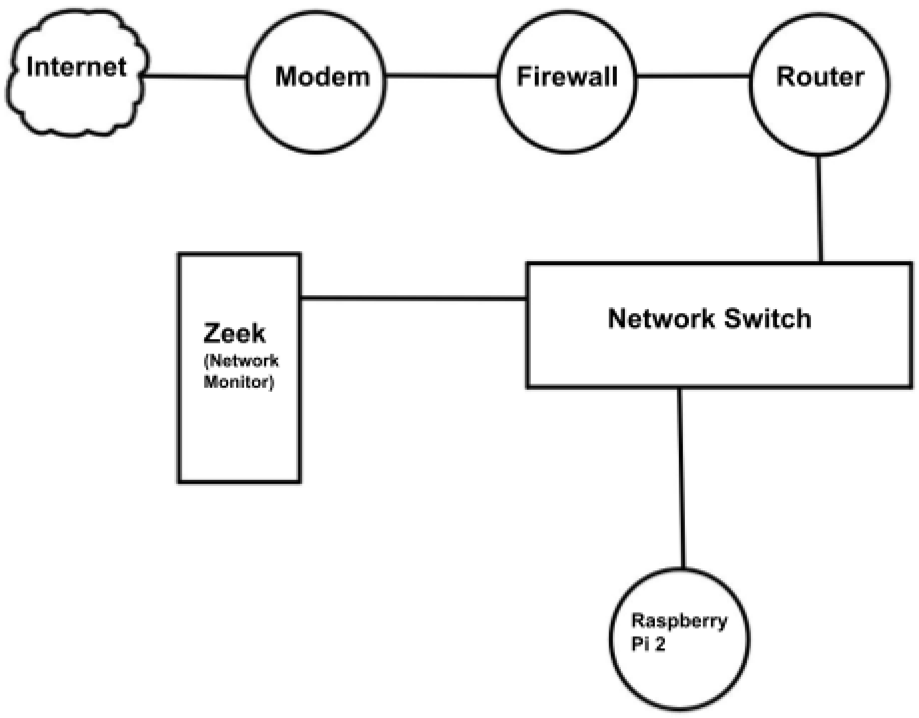


Figure 3.0 - Graphical Model for IoT Smart Home Network Illustrating Packet Capture Setup (Malware Scenario)

Additionally, the following illustrates the adjacency matrix for the graph shown in figure 3.0.

	Internet	**Modem**	**Firewall**	**Router**	**Network Switch**	**Zeek**	**Raspberry Pi 2**
Internet	0	1	0	0	0	0	0
Modem	1	0	1	0	0	0	0
Firewall	0	1	0	1	0	0	0
Router	0	0	1	0	1	0	0
Network Switch	0	0	0	1	0	1	1
Zeek	0	0	0	0	1	0	0
Raspberry Pi 2	0	0	0	0	1	0	0

Figure 4.0 - Adjacency Matrix for IoT Smart Home Network (Malware Scenario)

B. DNN Malware Classifier Performance

The following table summarises the performance metrics obtained from our fully trained DNN classifier:

Table 1.0 - Fully Trained DNN Metrics

Performance Metric	Value
Training Loss	0.003050
Validation Loss	0.016102
Accuracy	99.9204%
Precision	99.795%
Recall	100%
F1-Score	99.897%

As a visual aid to the above data, the following diagram illustrates the confusion matrix for the trained DNN:

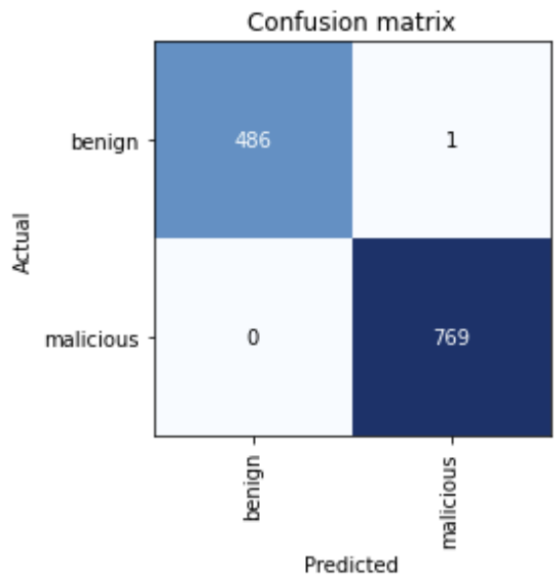


Figure 5.0 - Trained DNN Confusion Matrix

<u>C. Transfer Learning DNN Performance</u>

The following table summarises the performance metrics obtained after using additional malware data to tune our DNN classifier:

Table 2.0 - Transfer Learning DNN Metrics

Performance Metric	Value
Training Loss	0.002588
Validation Loss	0.014166
Accuracy	99.8566%
Precision	99.589%
Recall	100%
F1-Score	99.794%

As a visual aid to the above data, the following diagram illustrates the confusion matrix for the tuned DNN:

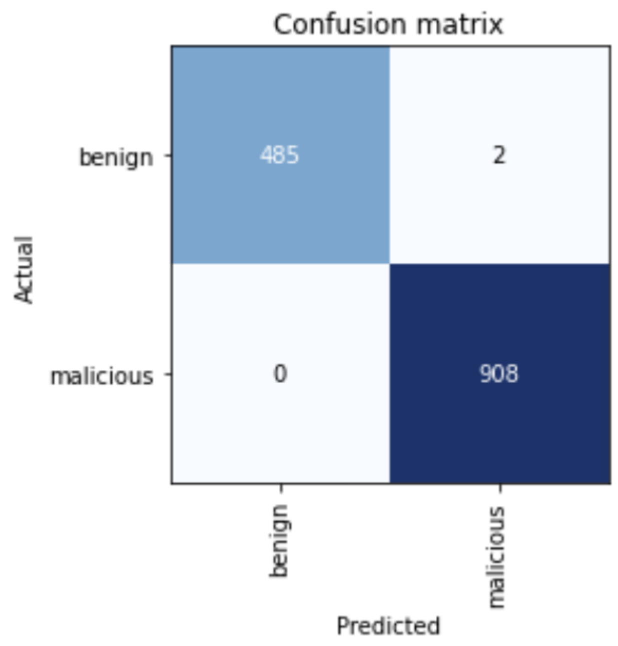


Figure 6.0 - Tuned DNN Confusion Matrix

Discussion

A. Detection

Based on the accuracy metrics for the fully trained DNN malware classifier, there appears to be exceptional performance on our test data as we sit at 99.9204%. Additionally, given the low training and validation loss metrics, there is very little evidence to show that the model may be overfitting or underfitting to the given network packet data.

When comparing the precision and recall values for the classification, it is interesting to note that the recall is at 100%. This is an indication that there were no false negatives when testing on the validation set. For a malware classifier this is an important metric, as we would rather falsely classify benign software as malware, than allow malware to pass under the radar as benign. Accordingly, the 99% precision accounts for the 2 benign packet captures that were misclassified as malicious.

The F1-Score gives a good indication of balance between the precision and recall metrics, but due to the fact that they are already very high, it is a given that the f1-score would also be above 99%.

All-in-all, there is no question regarding the satisfactory performance of the DNN in identifying malware packets.

One interesting thing to note regarding the nature of the dataset used to train the classifier is that the malware packet captures came exclusively from a Raspberry Pi 2 infected with a malicious program that exhibited typical traits of a Mirai botnet. Essentially, this malware seeks to turn an internet connected node - such as an IoT smart home device - into a remotely controlled bot as part of a larger network of malicious bots (bot-net) [9]. This dataset was chosen due to the nature of the malware as its infection and execution process involve a large use of network resources that would present themselves in packet captures. Thus, a great limitation of the DNN classifier is that it can only detect malware that performs visible actions across a network. In other words, a malware that infects an IoT device and acts strictly locally can likely never be detected.

B. Tuning

Based on the results of the transfer learning performed on the pre-trained DNN classifier, it is relatively safe to say that tuning the weights of the pre-trained model rather than re-training a new model from scratch proved to be a viable method for updating the classifier with new malware data. The accuracy of the DNN dropped by only 0.0638% after introducing more than 10% new packet captures to the dataset, equating to a single additionally misclassified packet. Once again, we see our recall maintained at 100%, suggesting that our only classification errors were 2 benign packets mislabelled as malware.

Conclusion

With reference to our problem statement, we can safely conclude the following:

1. A DNN malware classifier for IoT smart home devices was successfully trained to detect network-based malware indicators, and produced a final classification accuracy of 99.897% on the tested dataset.
2. A transfer learning method for updating the trained DNN classifier with new malware data was shown to work as intended with minimal performance impact of only 0.0638% accuracy reduction when tested on the extended dataset.

The objectives outlined in the introduction to this project have therefore all been successfully achieved and appropriately documented.

References

[1]	The internet protocol stack. [Online]. Available: https://www.w3.org/People/Frystyk/thesis/TcpIp.html. [Accessed: 05-Dec-2022].
[2]	A. A. Yilmaz and O. Aslan, “A New Malware Classification Framework Based on Deep Learning Algorithms,” IEEE Xplore, 15-Jun-2021. [Online]. Available: https://ieeexplore.ieee.org/document/9455368. [Accessed: 05-Dec-2022].
[3]	S. E. Coull and C. Gardner, “Activation analysis of a byte-based deep neural network for malware classification,” 2019 IEEE Security and Privacy Workshops (SPW), 2019.
[4]	J. Brownlee, “Neural networks are function approximation algorithms,” MachineLearningMastery.com, 26-Aug-2020. [Online]. Available: https://machinelearningmastery.com/neural-networks-are-function-approximators/. [Accessed: 05-Dec-2022].
[5]	“What is Big Data?,” What Is Big Data? | Oracle Canada. [Online]. Available: https://www.oracle.com/ca-en/big-data/what-is-big-data/. [Accessed: 05-Dec-2022].
[6]	“Machine learning and the internet of things,” ZDNET. [Online]. Available: https://www.zdnet.com/paid-content/article/machine-learning-and-the-internet-of-things/. [Accessed: 05-Dec-2022].
[7]	“IOT-23 dataset: A labeled dataset of malware and benign IOT traffic.,” Stratosphere IPS. [Online]. Available: https://www.stratosphereips.org/datasets-iot23. [Accessed: 05-Dec-2022].
[8]	“Fast.ai-making neural nets uncool again,” fast.ai-Making neural nets uncool again. [Online]. Available: https://www.fast.ai/. [Accessed: 05-Dec-2022].
[9]	“What is the Mirai botnet?” [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/. [Accessed: 05-Dec-2022].