SCTE•ISBE CABLE-TEC EXPO® VIRTUAL EXPERIENCE » OCTOBER 12-15 2020

2020 Fall Technical Forum SCTE·ISBE • NCTA • CABLELABS®

Smart Gateways

Active A.I. in Subscriber Networks

A Technical Paper prepared for SCTE•ISBE by

Kyle Haefner
Senior Security Engineer
CableLabs
858 Coal Creek Cir, Louisville, CO 80027
303-661-3803
k.haefner@cablelabs.com

Table of Contents

List of Figures

List of Tables

1. Introduction

In the last several years progress toward securing Internet of Things (IoT) devices has been made on several fronts. There are now mature specifications for IoT devices that require with encryption, authentication and authorization for every device (1). Governments and industry have released baselines (2), (3), (4) that provide guidance on what should constitute a secure device. There is even recent legislation at the state level aimed at enforcing security in IoT (5).

None of this will guarantee secure devices. There will *always* be devices that are exposed, unpatched and vulnerable. Even companies and manufacturers that prioritize security will inevitably find themselves with vulnerabilities inherited in the supply chain from decades old code like Ripple20 (6). Combine this with malware like Mirai that is constantly being updated to take advantage of these newly discovered vulnerabilities (7) and it becomes clear that building strong security into individual devices is simply not enough. The question that now needs to be answered is, can secure systems be built from networks of potentially insecure devices?

The question posed above is not a mere hypothetical one. Today's subscriber networks consist of not just a heterogenous mix of devices, but also the implicit mix of vulnerabilities and attack surfaces inherent in today's complex home networks. To address this problem in a comprehensive and systematic way, intelligence must be added to the network so as to give the network the ability to know the devices running on it, learn how those devices behave and be capable of actively and surgically blocking traffic that is outside the bounds of what is deemed normal.

This research presents a method whereby a centralized router/gateway can learn a device's behavior on the network and based on that behavior, determine normal and abnormal behavior from that device. The model presented in this paper takes advantage of the predictability of an IoT device's network footprint by developing a formalized measurement of complexity for each device. Low complex and simple devices are more accurately modeled and thus can be more confidently managed autonomously by the network.

After describing the framework necessary to measure the complexity of network devices, this work then uses this complexity measure to inform and tune an anomaly detection algorithm to construct a behavioral model for each device. This tuned model represents the behavior footprint of each device learned from its network traffic and forms the basis for differentiating normal traffic from abnormal.

To demonstrate the efficacy of this model, this work analyzes boundary of each device's learned behavior against seven common types of malware traffic from infected IoT devices. Finally, to illustrate that the model can be effectively applied to a broad spectrum of devices, four different IoT datasets were analyzed: one residential dataset, two lab datasets, and a dataset based on commercial IoT devices. The results show that this model can be an effective way to actively block Distributed Denial of Service (DDoS) attacks and malware traffic especially on low complex devices.

2. Background

There has been a great deal of research in the past few years to grant the network with the capability to learn what devices are running on it, (8) and how to automatically block devices from contributing in DDoS attacks. The majority of recent academic research in IoT behavior and fingerprinting relies on various machine learning (ML) techniques. The ML techniques can be broadly categorized into two main groups, supervised and unsupervised.

2.1. Supervised

Supervised learning requires a large corpus of labeled data. The work that takes advantage of supervised learning typically tries to classify a device on the network based on previous traffic that has been labeled as that same device.

Loepz-Martin et al. (9) build a network traffic classifier (NTC) using a recurrent neural network (RNN) and apply it to labeled IoT traffic. The goal of this is to identify the types of traffic and services exhibited by an IoT device as a step toward identifying the device.

Miettinen et al. (10) have developed a method, called IoT Sentinel, that uses machine learning to designate a device type on the network, referred to by the authors as a device fingerprint. Using the random forest algorithm and 23 network features they were able to identify device types on the network based on the device's traffic. The 23 features are based on layer two, three and four of the OSI networking stack. Expecting that the body of the packet will be encrypted, all the features the authors employed are based on unencrypted parts of the traffic like IP headers information.

Bezawada et al. (8) build on the work done in Miettinin using a machine learning approach to broadly identify the device and place it in a predefined category, such as a light bulb. According to the authors, even devices from different manufacturers can be placed into general categories such as two separate light bulbs can be identified and placed into a lighting category and addressed by security policies based on this category.

Supervised methods are generally highly accurate but require large examples of labeled traffic to adequately learn. These methods generally cannot classify things that were not present in the learning dataset and are unable to classify new or unknown devices.

2.2. Unsupervised

Unsupervised learning does not need data with labels and instead tries to learn underlying patterns in the data itself. It has various advantages in the context of IoT security as there will often not be labeled data available and there will always be new devices for which there exists no labeled data.

AuDI (11) implemented an autonomous device-type identification that uses the periodicity of device communications resulting in abstract device categories that could be used to enforce access control policies. DioT (12) extends the AuDI classification model to create a federated approach by aggregating device anomaly detection profiles.

Ortiz et al. (13) set up a probabilistic framework to monitor device behavior using a Long-Term Short-Term Memory (LSTM) neural network, to learn from inherent sequencing of TCP flows to automatically learn features from device traffic with the intent of categorizing devices and distinguishing between IoT devices and Non-IoT devices. The authors are able to identify previously known devices after only 18 TCP-flow samples and categorize devices into two classes IoT and Non-IoT

3. Methodology

3.1. Data-Sets and Data Collection

3.1.1. Datasets

This work analyzed four different datasets of IoT traffic:

- **home**: sourced from a residence and represents normal usage traffic contains 37 days of traffic from over 25 devices that are a mix of typical IoT devices and more general devices such as laptops and smartphones,
- **lab**: traffic is from several devices in a lab at Colorado State University, it contains approximately 22 devices and several month's worth of traffic,
- **unsw**: IoT traffic from a lab at University of New South Wales (UNSW) contains nearly three weeks of traffic from 29 devices that are mixed IoT and general-purpose devices,
- **scada**: traffic from a Supervisory Control and Data Acquisition (SCADA) test network at Colorado State University's Methane Emissions Test and Evaluation Center. It has nearly 40 devices mostly made up of SCADA devices called Lab jacks.

3.1.2. Attack Data

The malware data comes from Stratosphere Laboratory (14) and contains 20 captures where malware was executed on IoT devices, and 3 captures for benign IoT devices traffic. This dataset was first published in January 2020, with captures ranging from 2018 to 2019.

Table 1: Malware Data

Attack Name	Description	Unique Flows
C&C	This is traffic where a device is connecting to a remote command and control server.	30
C&C Heartbeat	This is traffic that is meant to monitor the status of an infected host.	3
C&C Torri	This is command and control traffic specifically from the Torri botnet	1
C&C FileDownload	This is traffic from an infected device downloading a file or malicious payload	7
DDoS	This is traffic where a device is participating in a distributed denial of service attack	1
Part of Horizontal Scan	This is traffic where a device is scanning locally on the network	49959
Okiru	This is traffic specifically from the Okiru botnet.	99888

3.2. Data collection

For the home, and lab datasets data was collected in the form of IPFIX/Netflow flows from a centralized router to a flow collector as shown in Figure 1: Data Collection Architecture. Flows were saved in a SQL database. For the UNSW data set and the SCADA data set, the pcap files were analyzed by an open source tool called Joy (15). Joy transforms pcap files into json data which was then loaded into a SQL database.

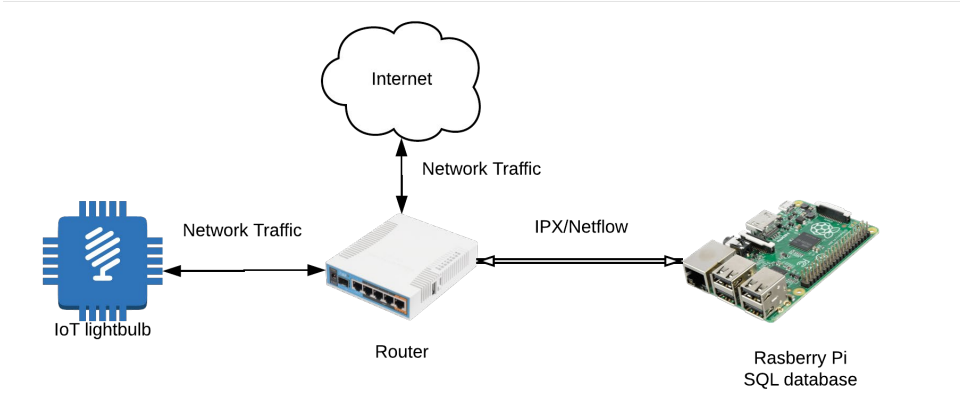

Figure 1: Data Collection Architecture

Table 2: Features from network flows

Feature	Description
IPv4 Source Address	IP Address of the device
IPv4 Destination Address	IP Address the device is connecting with on the network
In Packets	Number of packets received by the device
Out Packets	Number of packets sent by device
L4 Source Port	Source port on device
L4 Destination Part	Remote port that the device is connecting with on the network
Protocol	IANA protocol, eg UDP,TCP

3.3. Device Complexity

The complexity of a device on the network is based on its network traffic. Sensors that talk to a relatively few network endpoints will be measured as low complexity devices. This is compared devices such as laptops and smartphones that make highly varied requests to many network endpoints. These devices will be measured as high complexity devices. To illustrate this, Figure 2 shows where devices fall in a range from low complexity to high complexity.

Figure 2: Spectrum of complexity

General Purpose Device

A device that is capable of running multiple user-space applications. Some examples are smart-phones, tablets, laptops, some streaming devices, and smart TVs. These devices will have higher network complexity.

Single Purpose Device

A device that generally runs a single application. They often are capable of only one or two threads. These devices will have lower network complexity.

For a network flow there are several features that can be examined as shown in Table 2. This research focused on destination-based features. The complexity analysis is based only on destination IP address and destination port, though other features could be added.

To measure a device's complexity on the network this work uses a concept similar to one used in communications, called the Signal to Noise Ratio (SNR) which compares the level of desired signal to the background noise. Within the context of Internet traffic from a device, the signal is defined as traffic data points that can be clustered, and the noise is defined as the datapoints that cannot be clustered. Some of the devices measured had no noise components making the SNR ratio undefined. To account for this this research uses the reciprocal if the SNR, referred here as the noise to signal ratio (NSR) shown in Equation 1. In the case where there are zero points of noise the NSR=0.

This measure of complexity uses the DBSCAN (16) clustering algorithm to compute the number of clusters (signals) and the non-clusters (noise). This algorithm is good at finding areas of high density that are separated by areas of low density. The DBSCAN algorithm has several advantages in that it can find clusters of arbitrary shape and size, and can include clusters that are non-convex, unlike other clustering algorithms such as k-means.

The DBSCAN algorithm takes in two primary parameters, a distance parameter ε and a number of points that are within that distance called, *min_samples,* to form a cluster. To automatically find the distance parameter this work uses the device's *IP_Spread* as shown in Equation 1Equation 1: Distance Calculation. The IP_Spread is the set of IP addresses where the first order octet is unique and represents the total number of unique network that the device connects to.

Equation 1: Distance Calcualtion

$$\epsilon = 128 * IP\_Spread$$

128 is the midpoint of the address space of a class C network. Experimentally setting min_samples = 10 was a good starting value for the total number of neighborhood points necessary for a point to be calculated as a core point.

The number of clusters found by the DBSCAN algorithm and the number of non-clusters is used to calculate the NSR for the device using Equation 2.

Equation 2: Noise to Signal Ratio

$$NSR = \frac{n\_noise}{n\_clusters}$$

3.4. Results Methodology

To evaluate the results of the model a dataset consisting of traffic from several different malware types was used to see how well the model was able to predict normal traffic from abnormal attack traffic. Evaluating the model produces four metrics on how the model is performing. These four metrics are:

- true positives (TP): malware traffic correctly identified as malware traffic,
- true negatives (TN): normal traffic correctly identified as normal traffic,
- false positives (FP): normal traffic incorrectly identified as malware traffic,
- false negatives (FN): malware traffic incorrectly identified as normal traffic.

Table 3 shows the confusion matrix for this analysis.

Table 3: Confusion Matrix for IoT Traffic

	Predicted: Normal Traffic	**Predicted: Malware Traffic**
Actual: Normal Traffic	TN	FP
Actual: Malware Traffic	FN	TP

From these four metrics there are two important factors that are often used, precision, the ratio of correct positive predictions to the total predicted positives and recall, the ratio of correct positive predictions to the total positive examples.

Equation 3: Precision

$$P = \frac{TP}{TP + FP}$$

Equation 4: Recall

$$R = \frac{TP}{TP + FN}$$

And finally, the balanced method for measuring the efficacy of a prediction model is called the F1 score. The F1 score is the harmonic mean of the precision and recall.

Equation 5: F1 score

$$F_1 = 2 * \frac{P * R}{P + R}$$

In Section 4 the results are shown how each device model performs against the malware traffic using the F1 score. An F1 score of 1 is a perfect score, it means that the device model correctly identified all of the device traffic as normal and all of the malware traffic as abnormal.

4. Results

The results across all four of the datasets support the hypothesis that devices can be measured for complexity based on their network flows, and that this measurement can be used to categorize devices into two separate groups, single purpose devices and general-purpose devices. Further, the results support that measuring the complexity of a device is relevant to improve modeling of devices. For brevity only three devices from the Home dataset are shown, one of high complexity, one of median complexity and one of low complexity. Figure 8, Figure 9, Figure 10, and Figure 11 show the results of all devices for each data set analyzed and averaged against the seven malware traffic types.

4.1. Complexity Results

Figure 3 shows the Roku Express device in the home dataset with the largest NSR value of 52.25. Only four clusters were found with 209 points of noise. This is compared to the J Android device which falls in the middle of all devices with an NSR of 10.2 as calculated from 19 clusters and 194 points of noise shown in Figure 4. Finally, in Figure 5 we see the lowest complex device on the home network which is the Eufy light bulb. This device has only 2 clusters and zero points of noise leading to an NSR value of 0.

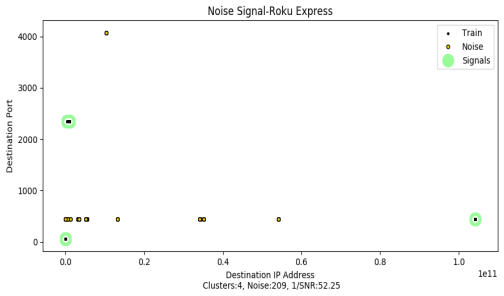

Figure 3: Highest NSR Roku Express

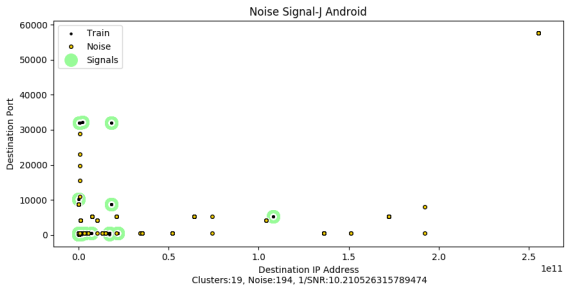

Figure 4: Median NSR Android

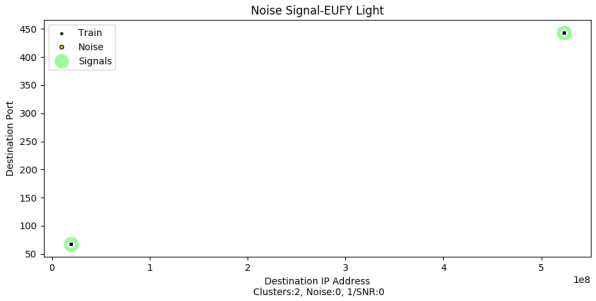

Figure 5: Lowest NSR Eufy Light

Figure 6 shows the NSR complexity of all the devices on the home network. A line is drawn at the average NSR value of 8.5. Almost all the devices that have an NSR above this value are general purpose devices and conversely almost all below 8.5 are single purpose devices. Streaming devices such as Roku and Smart TVs were consistently measured across the four datasets as higher complexity devices. This is likely because these devices are often Linux OS based devices capable of running several user space applications.

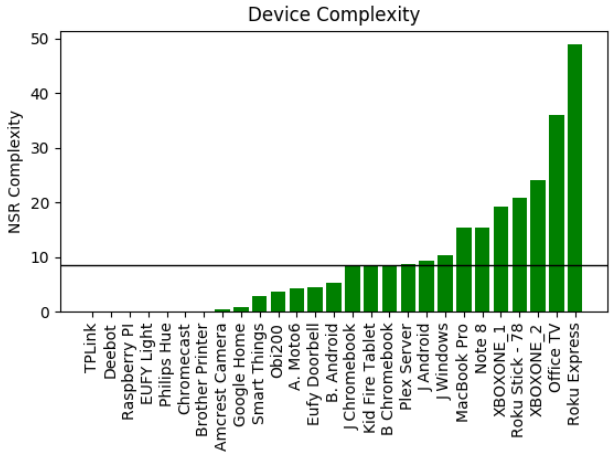

Figure 6: Home Device NSR Complexity

4.2. Behavior Results

Figure 7 shows the three devices, the Roku Express, J Android, and the Eufy light bulb. Each figure shows the normal traffic as black dots, the learned behavior boundary as blue ellipsoids generated from the Gaussians, and the attack traffic represented as red dots. In each figure the port and IP address have been normalized. Normalizing both improves classification and makes it possible to show the traffic and boundaries in non-logarithmic space.

The Roku Express has the highest NSR value of 52 in the home data set with only four clusters and 208 points of noise. This leads to a relatively poor model for the device with a 0.5% false positive rate against the C&C malware traffic. Visually, the figure shows large ellipses that fail to contain several of the normal traffic within their boundaries leading to 17 false negatives. Additionally, some of the ellipses overlap with the malware traffic leading to false positive identification. The number of true positive outliers detected is also poor with the model only detecting 43% of the malware traffic in the C&C traffic. This gives the model an F1 score of 0.59.

For the J. Chromebook device with an NSR value of 8.6 which is just slightly higher than the data set average of 8.548. This NSR value comes from a total of 15 clusters and 129 points of noise and leads the GMM to find a total of 15 ellipsoids. As compared to the model of the Roku, the J. Chromebook shows smaller margins on the ellipsoids that are centered around the normal traffic. The false positive rate of the J. Chromebook model is double that of the Roku model at 1%, however, there is less overlap in these ellipsoidal boundaries with the attack traffic than in the Roku model leading to a true positive accuracy of 100% for the malware traffic. This in turn leads to a much higher F1 score of 0.968 against the C&C malware.

Finally, in the figure for the Eufy light bulb there are only two ellipsoids generated by the model. The Eufy device is a light bulb and has the lowest measured NSR complexity in the home dataset with an NSR of 0 that is composed of just 2 clusters and 0 points of noise. This leads to a behavioral boundary tightly coupled around the normal traffic, with a 0.0% false positivity rate and a 100% accuracy in identifying the malware traffic. This leads to a perfect F1 score of 1.00.

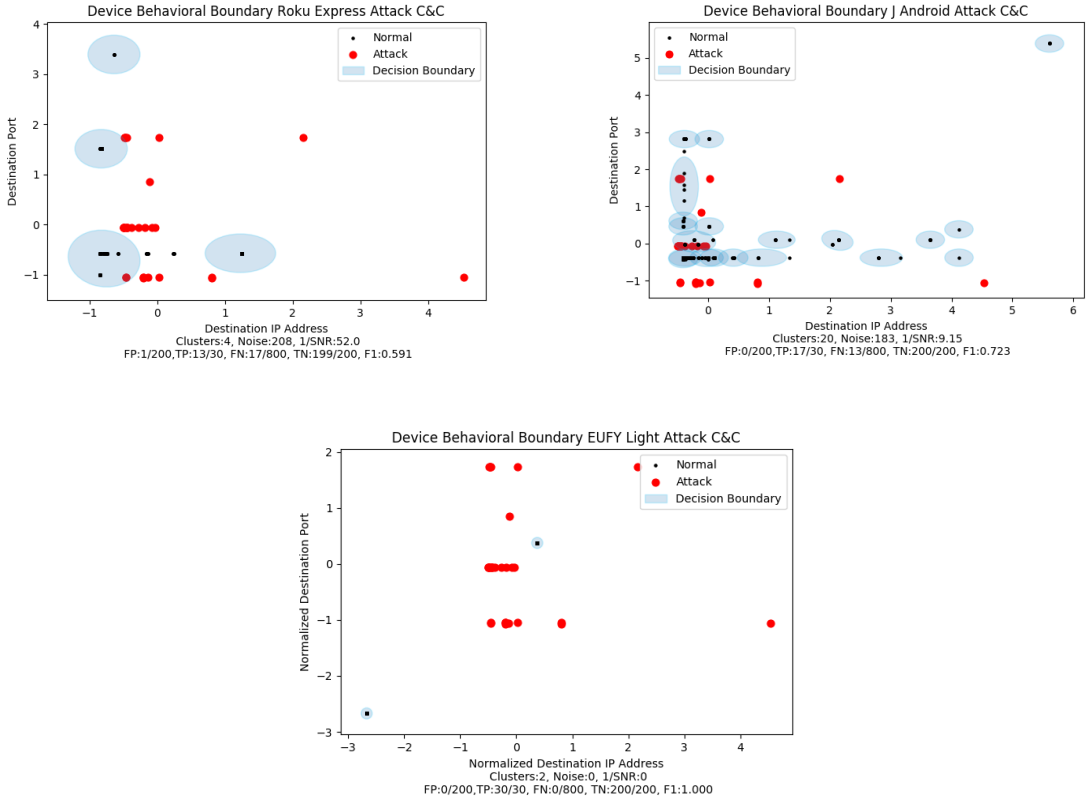

Figure 7: Device Behavior Boundaries

4.3. Overall Results

Every device in each data set was modeled and evaluated against each of the seven malware traffic types. The average NSR complexity score, and the average F1 score for each of the four datasets is shown in Table 4. The table shows the inverse relationship between average NSR complexity and average F1 score, highlighting the notion that simple devices can be more accurately modeled.

Table 4: NSR and F1 Scores

Data Set	Average NSR Complexity	Average F1 Score
Home	8.548	0.901
Lab	9.596	0.879
UNSW	7.170	0.942
SCADA	2.791	0.975

Figure 8 shows the average F1 score for each averaged across the malware traffic. The upper left of Figure 8 shows the model using the non-normalized data. It is shown to illustrate the negative correlation between a device's NSR complexity and the F1 score. This supports the idea that simple devices can be more accurately modeled. In the upper right of the figure we see how the model is improved by normalizing the data. Normalizing generally improves the efficiency and accuracy of clustering algorithms, this is especially true when the clustering algorithm uses the squared Euclidean distance metric as is done in the DBSCAN algorithm.

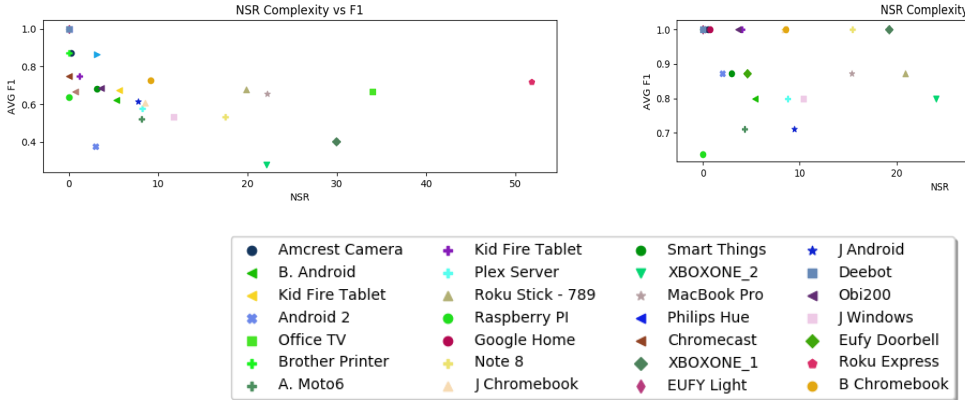

Figure 8: F1 Score vs NSR Complexity [Home]

Figure 9 shows the average F1 score of each device against the seven types of malware traffic. Again, the upper left shows the non-normalized scores and the upper right scores based on normalized data.

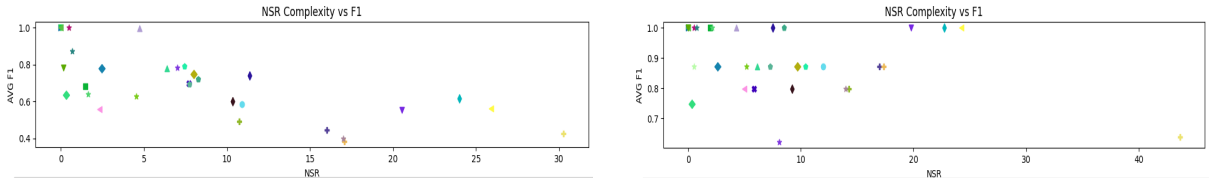

Figure 9: F1 Score vs NSR Complexity [Lab]

Figure 10 and Figure 11 show the results for devices on the UNSW data set and SCADA dataset respectively. The UNSW results are consistent with the Home and Lab data sets in showing a negative correlation of F1 score and NSR score for non-normalized data.

The SCADA data set is distinctly different from the other three datasets, in that there does not appear to be an obvious correlation with complexity and modelability. The SCADA dataset had the lowest average NSR and the highest average F1 score. This is consistent with the notion that SCADA devices are very simple in terms of their network footprint and the results support this.

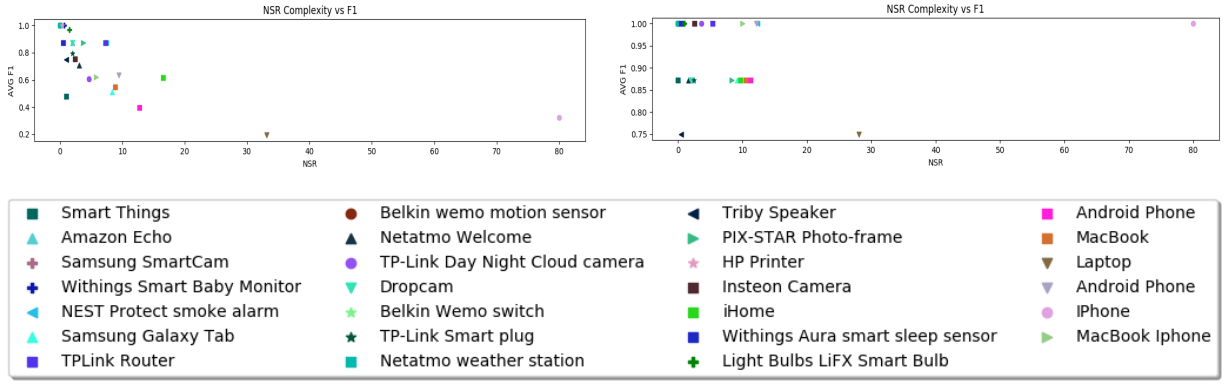

Figure 10: F1 Score vs NSR Complexity [UNSW]

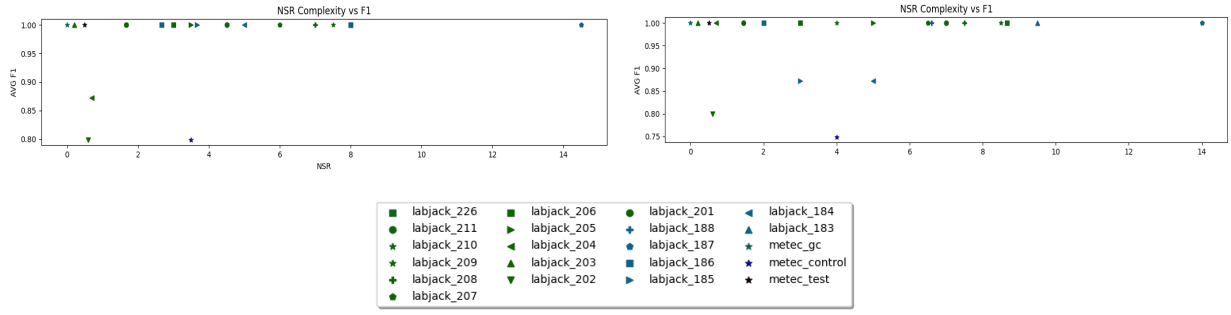

Figure 11: F1 Score vs NSR Complexity [SCADA]

5. Conclusion

This research developed a method for measuring the complexity of IoT devices based on their network traffic. This method called the Noise to Signal Ratio (NSR) uses a clustering algorithm to determine how much of the traffic from a device can be classified as a signal and how much as noise. The number of clusters from this algorithm feeds a Gaussian mixture model that is used to construct a behavioral model for each device and classify normal versus abnormal traffic.

This model was then run against seven very different types of actual malware traffic to determine the efficacy of the model in classifying a device's normal traffic from malware traffic. The results show that the model is effective, with many devices having perfect F1 scores. The results also show that F1 scores are generally higher for less complex devices, supporting the claim that simple devices can be more accurately modeled than complex devices.

This suggests that automatic blocking of malware traffic could be done, especially on devices that are simple, i.e. have low NSR scores.

Future work will examine how the NSR of devices can be applied to other anomaly detection methods such as isolation forest and single class support vector machines.

A.I.	Artificial intelligence
C&C	command and control
DDOS	distributed denial of service
GMM	Gaussian mixture method
IoT	Internet of things
LSTM	long term- short term
M.L.	machine learning
NSR	noise to signal ratio
NTC	network traffic classifier
RNN	recurrent neural network
SCADA	Supervisory Control and Data Acquisition

Bibliography & References

1. **OpenConnectivity.** OCF Solving the IoT Standards Gap. [Online] 2020. https://openconnectivity.org/.

2. **Fagan, M, et al.** *Foundational Cybersecurity Activities for IoT Device Manufacturers.* s.l. : National Intiture of Standards and Technology, 2020.

3. **Various.** *The C2 Consensus on IoT Device Security Baseline Capabilities.* s.l. : The Consumer Technology Association, 2019.

4. —. *CYBER; Cyber Security for Consumer Internet of Things* . s.l. : European Telecommunications Standards Institute, 2019.

5. **SB-327.** SB-327 Information privacy: connected devices. [Online] September 28, 2018. https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201720180SB327.

6. **Ripple20.** Overview- Ripple20. [Online] 2020. https://www.jsof-tech.com/ripple20/.

7. **Micro, New Mirai Variant - Trend.** New Mirai Variant Expands Arsenal, Exploits CVE-2020-10173. [Online] July 10, 2020. https://blog.trendmicro.com/trendlabs-security-intelligence/new-mirai-variant-expands-arsenal-exploits-cve-2020-10173/.

8. **Bezawada, Bruhadeshwar and Bachani, Maalvika and Peterson, Jordan and Shirazi, Hossein and Ray, Indrakshi and Ray, Indrajit.** Behavioral Fingerprinting of IoT Devices. *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security.* 2018, pp. 41-50.

9. **Lopez-Martin, Manuel, et al.** Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access.* 2017, pp. 18042-18050.

10. **Miettinen, Markus, et al.** IoT Sentinel: Automated device-type identification for security enforcement in IoT. *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS).* 2017, pp. 2177-2184.

11. **Marchal, Samuel, et al.** AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication. *IEEE Journal on Selected Areas in Communications.* 2019, pp. 1402-1412.

12. **Nguyen, Thien Duc, et al.** DIoT: A Federated Self-Learning Anomaly Detection System for IoT. *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS).* 2019, pp. 756-767.

13. **Ortiz, Jorge , Crawford, Catherine , Le, Franck.** DeviceMien: Network Device Behavior Modeling for Identifying Unknown IoT Devices. *Proceedings of the International Conference on Internet of Things Design and Implementation.* 2019, pp. 106-117.

14. **Parmisano, Agustin, Garcia, Sebastian and Erquiaga, Maria Jose.** Stratosphere Laboratory, A labeled dataset with malicious and benign IoT network traffic. [Online] April 20, 2020. https://www.stratosphereips.org/datasets-iot23.

15. **Various.** JOY: Network Capture and Analysis Tool. 2020.

16. **Ling, Robert F.** On the theory and construction of k-clusters. *The computer journal.* 1972, pp. 326-332.

17. **Akita.** SOHO & Home Cyber Security as a Service. *akita.cloud.* [Online] 2020. https://shop.akita.cloud/ .

18. **Sentry, Cujo.** Network Security and Device Protection. [Online] 2020. https://cujo.com/sentry/ .

19. **Bitdefender.** Smart Home Cyber Security for Your Business. [Online] 2020. https://www.bitdefender.com/iot/ .

