Attack Surface Monitoring and Analysis of IoT Devices using Open Source Intelligence

Rohan Konda

University of Arizona

CYBV 498: Senior Capstone – Engineering

Professor Dalal Alharthi

Table of Contents

Abstract

In this paper, we combine feature extraction and machine learning techniques to identify IoT devices in a network and detect their behavior based on network scanning and network traffic analysis. The network traffic data is used to train models and eliminate non-IoT devices and classify the IoT devices. Our GradientBooster Classifier algorithm detects and classifies IoT device types with 85% accuracy. Based on normal and malicious IoT traffic behavior of various IoT devices, we use machine learning techniques to identify abnormal behavior of an IoT device based on device type and associated network traffic to create IOT alert notifications.

Open Source Intelligence (OSINT) tools can automate threat intelligence, asset discovery, attack surface monitoring or security assessments. Spiderfoot is one such modular based OSINT platform. In this work, we combine IoT device type classification and identify abnormal traffic of IoT device based on its type generating alert notifications via the Spiderfoot integration module to the Spiderfoot dashboard.

CCS Concepts

Security and privacy/ Mobile and wireless security; Machine learning; Integration

Keywords

Internet of Things (IoT); Cyber Security; Machine Learning;

Network Traffic Analysis

OSINT

Spiderfoot

Malware detection

1. **Introduction**

1.1. Problem Statement

Many homes, businesses, and other organizations suffer from network security issues from deprecated software and malware/malicious programs. The broadening of the attack surface of a network in the digital age has made it more difficult to properly secure a network. The use of Open Source Intelligence tools (OSINT) to monitor the attack surface, are greatly beneficial in securing these networks. This paper aims to provide increased security by investigating techniques to identify vulnerable IoT devices in the network, detect their abnormal behavior and automating these techniques using extension modules provided by OSINT frameworks to monitor IoT devices and provide alerts using OSINT alerting mechanisms.

1.2. Current Research

Several research papers have been authored on the identification, monitoring, and management of IoT devices on a network. This section will cover information and techniques referenced in those papers that are relevant to the research conducted in this paper.

In the 2019 paper, Network-Protocol-Based IoT Device Identification, the authors underline a method for identifying IoT devices on a network [2]. The method in this paper is a modular setup consisting of feature extraction, feature vector generation, and device identification. The features that are extracted and used to identify devices in this paper include the manufacturer name, device name, model, device OS, user agent, and service names. These are derived from network protocols and accessible device information like the MAC address, DHCP information, XML description in UPnP messages, mDNS header, and HTTP header. Using a bag of words model a vector is generated with unique words from the feature extraction and used to build a database of IoT devices. This database can be used later to identify newly connected devices by comparing similar vectors within the database to the vector generated for the new IoT device.

The paper “IoTSense: Behavioral Fingerprinting of IoT Devices” identifies methods for fingerprinting and identifying IoT devices based on their behavior rather than solely static features [5]. This approach of fingerprinting allows for the identification of a wider range of IoT devices and provides additional security as compromised IoT devices may exhibit altered behavior. The methods outlined in the paper combine a static behavioral model which is based on the analysis of the protocols used (ARP, SSL, MDNS, etc.) derived from packet headers, as well as dynamic behavioral modeling that analyzes command and response sequences derived from payload entropy, TCP payload length and window size. After generating feature vectors the authors use a machine learning classifier trained using gradient boosting to determine an IoT devices’ category and identity. The paper reports a 93-99% of mean accuracy in properly identifying the category of an IoT device and a 99.7-100% recognition rate of matching devices to stored profiles.

The paper “Detection of Unauthorized IoT Devices Using Machine Learning Techniques” covers the usage of multi-class machine learning classifiers to detect malicious or unknown classes of IoT devices connected to a network [6].   The proposed method in this paper involved identifying white listed authorized device types and collecting TCP/IP traffic data in pcap files of those devices over several months. A collection of 334 different features were extracted from the data and used to create feature vectors which were used to train a device type classifier. The authors used a Random Forest training algorithm due to its resistance to overfitting, fewer input parameters, and lack of prior feature selection. Due to a lower rate of accuracy on specific IoT device types such as smartwatches [6] they also incorporate a second stage of classification which is based on majority voting of different sessions of the IP stream. The analysis of the data researched in this paper determined that the most important features of eight of the nine device types were all tll related including ttl_min, ttl_firstQ, and ttl_avg. The classifier described in this paper achieved a 96% accuracy of detecting unauthorized IoT devices and demonstrated good portability between different environments.

In ProfilIoT [7], the authors describe machine learning algorithms on network traffic data for accurate identification of IoT devices connected to a network. The paper describes network traffic data collected and labeled from nine distinct IoT devices, and PCs and smartphones. Using supervised learning, they trained a multi-stage meta classifier to identify between IoT and non-IoT devices. In the second stage, each IoT device is associated with a specific IoT device class. The overall accuracy of the model is 99.281%.

The "Anomaly detection in Network Traffic" paper [13] describes a method of detecting malware behavior of infected IoT devices with anomaly detection using unsupervised learning techniques. The data set is highly imbalanced and contains various attacks such as DOS, Probe, U2R, R2L. The KDD data set was used to train the unsupervised machine learning algorithm called Isolation Forest to detect outliers and probable attacks and the results were evaluated using the anomaly score.

In the paper "A Two-Layer Dimension Reduction and Two-Tier Classification Model for Anomaly" [14], the authors present an intrusion detection based on two-layer dimension reduction and two-tier classification module, designed to detect malicious activities such as

User to Root (U2R) and Remote to Local (R2L) attacks. The research applies a two-tier classification module utilizing Naïve Bayes and a Certainty Factor version of K-Nearest Neighbor to identify suspicious behaviors.

In order to create ML models, training datasets are needed for normal and malware traffic that includes different environments such as homes and labs with IoT devices. The "Stratosphere Lab" [12] provides IoT traffic datasets. The IoT-23 dataset consists of twenty three captures (called *scenarios*) of different IoT network traffic. These scenarios are divided into twenty network captures (pcap files) from infected IoT devices (which will have the name of the malware sample executed on each scenario) and three network captures of real IoT devices network traffic (that have the name of the devices where the traffic was captured).

1.3. Theoretical/Contextual Framework

Internet-of-Things (IoT) devices are smart devices to enhance end-users' standard of living. However, there can be many vulnerabilities in these devices for exploitation by hackers. Many security problems can be mitigated by identifying and authenticating these devices and patching the software on these devices. For this, we have to identify the type of these devices and establish a behavioral baseline. The vast variety of these devices and protocols make the task of fingerprinting IoT devices very difficult. In our thesis, we incorporate some of the research methods covered in the Literature review section and integrate them the with Open Source Intelligence tool Spiderfoot to provide added security to the networks.

Attack surface monitoring in the digital context has in the past consisted mostly of monitoring open ports, host names, and IP addresses. However in the modern hyperconnected digital age the attack surface has greatly expanded. Much of people's finances, personal conversations, and record keeping is now done on the internet. Home appliances including thermostats, fridges, smoke alarms, and others now exist as digital IoT (Internet of things) devices. Additionally the attack surface now exists past any individual organizations network as things are done through cloud services or using software built on top of many libraries, frameworks, and programs that can't all be traditionally monitored and scanned.

Open Source Intelligence refers to all the data available in the public domain which can reveal information about a target. A prominent example is the site HaveIBeenPwned which keeps a publicly accessible database of email addresses that are associated with password and account breaches. The use of tools utilizing open source intelligence is an important aspect of modern digital security in order to combat the issue of an ever increasing attack surface.

SpiderFoot [1] is one of these reconnaissance tools that queries hundreds of public sources to gather intelligence on a target. It is an event based modularized system where a module will act on gathered intelligence data and send it to other relevant modules which in turn will generate events for other intelligence gathering modules. SpiderFoot also provides extension modules which allows programmers to develop their own modules and configure these modules to send alerts to the monitoring a specific vulnerability or possible compromise.

1.4. Thesis statement

This paper will detail the use of Open Source Intelligence tools like SpiderFoot to secure an organization's network by identifying and developing patterns of network traffic on a network. It also covers the techniques to identify IoT devices and extracting relevant information to identify the software patches used by IoT.

The thesis then proposes how to combine the above information with OSINT extension modules to provide alerts to the users regarding IoT devices running insecure software or leaks of financial and personal information.

2. Body

2.1. Identifying IoT Devices on the network

2.1.1. Engagement/activities

In a given home or small business office, there are several IoT devices connected to the WiFi network. The first goal is to find all the devices on the network and eliminate all non-IoT devices such as computers, phones, routers, switches using names, using descriptions, message patterns or MAC addresses. There are several tools/utilities such as ARP utilities used to identify all the devices on the network. NMap is a powerful network scanning tool that identifies device details in local networks. Wireshark is another tool that provides feature extraction capability. Significant research has been done on identifying IoT devices based on network traffic and some of these techniques are discussed in [2][3][4][7].

The below block diagram depicts the solution to the problem.

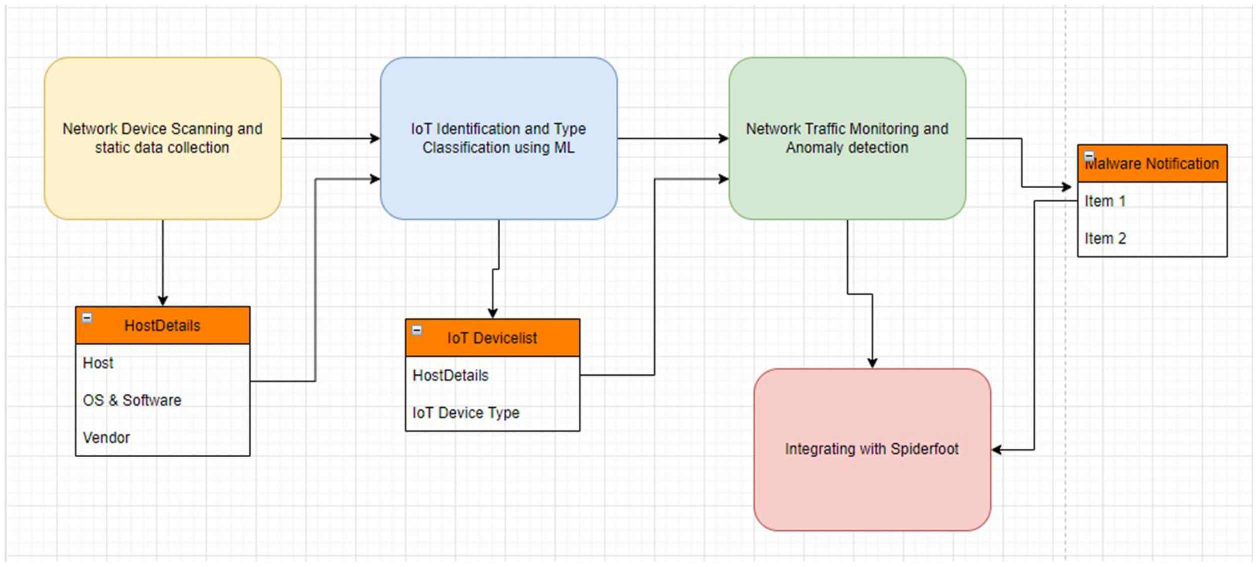
Network Device Scanning and static data collection
IoT Identification and Type Classification using ML
Network Traffic Monitoring and Anomaly detection
Malware Notification
Item 1
Item 2
HostDetails
Host
OS & Software
Vendor
IoT Devicelist
HostDetails
IoT Device Type
Integrating with Spiderfoot

Fig 1. Block Diagram

The system in this paper has 4 major modules:

- Network Device Scanning static data collection
    - This module identifies all the IP addresses (hosts) on the network and produces lists of hosts with OS, Open Ports, Software and Vendor information (HostDetails as shown in Fig 1).
- IoT Identification and Type Classification using ML
    - The module takes the list of hosts (IP addresses) from Network Device Scanning module and filters non-IoT devices and provides Host List with the Device Type (IoT DeviceList as shown in Fig 1) in addition to the details obtained in Network Device Scanning module using ML algorithms based on network traffic.
- Network Traffic Monitoring and Anomaly Detection

    - The module uses the device type information obtained in the IoT Identification module along with network traffic to detect anomalous network behavior by an IoT device along with device software obtained in the first step to generate alerts (Malware Notification as shown in Fig 1).
- Integrating with Spiderfoot
    - This module takes the alerts from the Network Traffic Monitoring module to publish the alerts to Spiderfoot OSINT platform.

2.1.2. Network Device Scanning static data collection

In this module, we focus on identifying all devices on the network with help of static network scanning with the following tools and techniques:

- ARP – Address Resolution Protocol
- NMap

ARP Python libraries provide the following data on wireless devices connected to the router.

- IP Address
- MAC Address
- Device Vendor

NMap is another powerful utility that provides that was used to identify other details:

- OS version
- Port list
    - Protocol
    - Software & Version

However the information from ARP, NMAP does not identify IoT device or its device type.

2.1.2.1. Data from ARP

ARP can be used to extract IP, MAC address and Vendor information. An example of ARP tool to extract home network provides the following information:

IP	MAC	Vendor
192.168.1.1	28:80:88:e7:cf:9e	NETGEAR
192.168.1.15	B0:72:bf:43:1e:11	Murata Manufacturing
192.168.1.5	00:1d:c0:63:e8:ad	Enphase Energy
192.168.1.2	00:0b:82:cb:ea:80	Grandstream Networks, Inc.

2.1.2.2. Data from NMAP

NMap can be used to identify the Operating system version of an IoT Device, port, protocol, software, vendor and version. NMap provides the following device information.

IP	OS	Port	Protocol	Software	Vendor	Version
192.168.1.1	Linux 3.14.77	80	http	uhttpd		1.0
192.168.1.1	Linux 3.14.77	21	ftp			
192.168.1.1	Linux 3.14.77	53		dnsmasq		2.80
192.168.1.2		80	http	httpd	BusyBox	1.113

This module provides HostDetails as shown in Fig 1 with a list of IP Addresses, MAC Address, Vendor, Open Ports, Protocols and software version.

2.1.2.3. Network Device Scanning - Objections

As seen in examples above, the identification and elimination of IoT devices is not very precise because names, descriptions can be changed which might eliminate legitimate devices from the list. We can gather only IP address, MAC address, Vendor, OS, software associated with the devices and open ports but all this information is static and can't determine the type of the IoT device.

2.1.2.4. Network Device Scanning - Refute objections

Even though the techniques to identify and filter out non-IoT devices is not precise, this method can be improved based on continuous monitoring and learning techniques in a given network. The network message based classification also provides a more refined technique. For current research, this is a reasonable compromise.

2.1.3. IoT Identification and Type Classification using ML

Once IoT devices are identified on the network, IoT devices have to be fingerprinted to identify its category such as Thermostat or a specific instance such as Nest Thermostat. The fingerprinting of an IoT device type is a technique of identifying the device type from a sample network activity of the device. The collection of all possible network activities of a specific device constitutes the behavioral profile of the device. The behavioral profile can be used to identify the IoT device. This information can be combined with OS and software information obtained from ARP and NMAP as described earlier to identify vulnerabilities or even software versions being used by the IoT devices.

ProfilioT [7] describes Machine Learning techniques to identify IoT devices based on some parameters. We use Machine Learning techniques such as Logical Regression to broadly classify and remove the devices such as PCs, Smart phones from the list of the devices and only include the IOT devices. The training data uses a lot of data captured from network traffic.

The network traffic data (TCP and UP packets) from local devices (IoT, smart phones and non-IoT devices such as PCs/laptops) connected to the WiFi access point or Router can be captured using the Wireshark tool in the form of pcap files.

There are several sources of reference training data of pcap files for training models. The stratosphere lab provides a lot of malware and normal traffic datasets that includes IoT traffic [11]. The data is labeled with the appropriate device category such as TV, Baby Monitor, Lights, Motion Sensor, Camera, Thermostat, Watch etc.

The TCP and UDP packets from sample and test files are analyzed to extract several other parameters using scapy python tool. Scapy is a powerful python library to extract information from these files. These parameters are extracted using scapy python tool from pcap files.

For example we can extract a lot of factors such as ack,ack_A,ack_B,bytes,bytes_A,bytes_A_B_ratio,bytes_B,ds_field_A,ds_field_B,duration,http_GET,http_POST,http_bytes_avg,http_bytes_entropy,http_bytes_firstQ,http_bytes_max,http_bytes_median,http_bytes_min,http_bytes_stdev,http_bytes_sum,http_bytes_thirdQ using scapy.

The full list of features that are extracted is provided in Appendix A. Some of the features are available in TCP packets and others in UDP packets. The features vary depending on the underlying software protocol e.g. HTTP or DNS that is being used in the message. If some of the features are empty in the message, we will remove such features and clean the data.

2.1.3.1. ML Models for classification

The supervised machine learning algorithms in this paper are based on a single predictive model, e.g. ordinary linear regression, single decision trees, and SVM. Ensemble Learning involves training multiple models and aggregating the predictions of these models. Ensemble also

involves using these results for predicting and either voting or feeding the prediction result to a different machine learning model. ML researchers also build an ensemble of ensembles.
Real-world data is often not as clean as required by models and the data don't follow one single distribution. Depending on the data, some models perform better on some distributions and some other models on certain other distributions. Ensemble helps to combine these ML models to get a better result and has been one of the top models in solving complex problems.
We have used an Ensemble of ML algorithms for classifying IoT devices:

- AdaBoostClassifier
- SVC
- DecisionTree
- KNeighbors
- LogisticRegression
- XGBRF
- RandomForest
- XGB
- LGBM
- GradientBoosting

2.1.3.2. Findings from activities – IoT Identification and Type Classification

When these models are run and compared with training and test data, GradientBoosting at 0.85 is the best prediction model. The model is tested with a different parameter set and the prediction reaches .85. Then the model is used to predict the test data to predict the type of IoT sensors.

Gradient boosting used in regression and classification tasks is one such technique. GBMs build an ensemble of shallow trees in sequence with each tree learning and improving on the previous one. Although shallow trees by themselves are rather weak predictive models, they can be “boosted” to produce a powerful “committee” that, when appropriately tuned, is often hard to beat with other algorithms. [12]

Device	**Accuracy**
TV	0.9
Baby Monitor	0.9
Lights	.47
Camera	1
Motion sensor	.97
Thermostat	.98
Watch	.97

The model can predict some devices more accurately than others. But overall accuracy is 0,85. With this classification, IOT DeviceList provides a list of IoT Devices along with their device types as shown in Fig 1.

2.1.3.3. IoT Identification and Type Classification using ML - Objections

IoT devices are very diverse. The devices include cameras, thermostats, smoker detectors, sensors, video devices, alarming devices, refrigerators. There are hundreds and thousands of manufacturers with no standards. The sheer diversity of these device types and software to be considered is so huge that fingerprinting and classification can be very difficult. The task of identifying the software versions being used by these devices can be even more difficult and not a standard software stack. The research of fingerprinting such IoT devices is very nascent.

2.1.3.4. IoT Identification and Type Classification using ML - refute objections

Even though fingerprinting IoT devices is a complex task, there are some Network traffic based Machine Learning based approaches that can be used to automate this. As shown in the current research, Ensemble learning is a powerful technique.

Once we identify the type of IoT device such as Thermostat or Light bulb or Security Camera, the network behavior of each type of device can be examined to identify any abnormal behavior.

2.1.4. Network Traffic Monitoring and Anomaly Detection

IoT devices on the network are usually prone to network attacks or malware infection because most of the time, their software is not updated. The software vulnerabilities are used by hackers to infect and target these devices for DDoS attacks or use them for launching DDoS attacks or monitor the home or business networks for sensitive information.

When the devices on the network are identified and classified of their device type, the next step in the research is to identify how these IoT devices behave. The key to understanding their normal network behavior and abnormal behavior is critical to identify attacks or malware infection on IoT devices.

Anomaly detection algorithms can learn the normal behavior of systems based on normal network traffic and alert for abnormalities even without knowledge on the characteristics of the attacks.

The training data in this uses the same network traffic with all features as selected in the previous IoT Classification module in addition to device_type as another feature.

The Anomaly detection algorithm identifies outliers based on each device type e.g. number of bytes being passed or communication with IP addresses that are not normal. In case of outliers, this module generates alerts file "MalwareNotification.csv" which contains Host, Malware notification and other details as shown in Fig 1.

There are many different Anomaly detection algorithms:

- Standard unsupervised anomaly detectors (Isolation Forest, LODA, One-class SVM, LOF)
- Clustering and density-based
- Density estimation based
- PCA Reconstruction-based
- Autoencoder Reconstruction-based
- Classifier and pseudo-anomaly based
- Ensemble/Projection-based

Time-series based Anomaly detectors also are available for network traffic based analysis.

- Forecasting-based
- Exploratory Analysis
- ARIMA
- Regression (SVM, Random Forest, Neural Network)
- Recurrent Neural Networks (RNN/LSTM)

There is considerable research on the selection of algorithms for Network traffic analysis and time series based analysis can be appropriate for detecting network traffic based anomalies.

In this paper we have not yet implemented such anomaly detection since our focus is IoT classification and integration with SpiderFoot.

2.1.5. Integrating with SpiderFoot

SpiderFoot [1] is a robust OSINT system and provides an extensible modular framework.

SpiderFoot has all data collection modularised. When a module discovers a piece of data, that data is transmitted to all modules that are 'interested' in that data type for processing. Those modules will then act on that piece of data to identify new data, and in turn generate new events for other modules which may be interested, and so on. Meanwhile, as each event is generated to a module, it is also recorded in the SpiderFoot database for reporting and viewing in the UI.

SpiderFoot streamlines both the processes of data collection and continuous monitoring, automatically triggering notifications via email, Slack and others when an organization's OSINT footprint is detected. Taking the automation a step further, this can help security operations run more efficiently through integration with their SIEM, vulnerability scanners and risk dashboards to support incident response and overall information / security risk management.

2.1.5.1. SpiderFoot Integration Activities

The IoT extension module allows developers to create a module to fingerprint and monitor IoT devices on the network. The extension module is developed from a template provided by SpiderFoot and configured in SpiderFoot as another module. The IoT extension module takes input of a database of fingerprints for the IoT devices, anomalies in network traffic and raises alerts based on abnormal behavior of any of the devices on the network.

In order to test the framework, we generate abnormal messages using network tools mimicking an existing IoT device.

The Network Traffic Monitoring and Anomaly Detection module creates a "MalwareNotification.csv" file with all details of any IOT anomalous behavior that includes the IP address of the device, MAC address, Vendor, OS, Software and versions if available and possible abnormal behavior details and sends a "IOT_ANAMOLY" event to the SpiderFoot instance. The event is processed by our “sp_iotmonitor” extension plugin module. The event handler opens up” MalwareNotification.csv” and reads the details and posts it from the “IOT_MALWARE” event to the SPiderfoot “WEBSERVER_BANNER” to display on the dashboard for review and actions.

2.1.5.2. Findings from SpiderFoot Integration Activities

The events generated from the IoT extension module can be viewed from SpiderFoot UI.

2.1.5.3. SpiderFoot IoT Extension Approach – Objections

- Difficult for residential users to setup a service such as SpiderFoot
- Configuring an extension may need skills to troubleshoot which is difficult for novice users

2.1.5.4. SpiderFoot IoT Extension - refute objections

- Preconfigured SpiderFoot installations on a Raspberry Pi / pre configured device can be made available
- SpiderFoot can be used as a service rather than setting up your own instance for subscription fees. The service provider can configure IoT extension module

3. Conclusion

In this paper, we demonstrated that IoT device information such as Vendor, OS, Software information can be obtained from network scanning tools. The IoT device type was classified based on its network traffic. Our Gradient Boosting classifier method can classify an IoT device, with 85% accuracy. Based on Device type, we also demonstrated we can further identify abnormal behavior of an IoT device using device type and its network traffic. The anomalies in IoT device behavior can then be integrated to provide real-time alerts to homes and organizations about IoT devices that are compromised. We believe that our novel approach can be used in association with the vendor and software for updating the IoT devices with appropriate patches. In future research, we can look to explore how manufacturers can automatically receive these alerts to coordinate releasing of security patches.

3.1. Present/Analyze Findings

- Nmap/ARP tools provide comprehensive static data related to IoT devices but won't be able to recognize specific devices. However they can extract information such as OS, Ports open, Software/Protocols associated with each port which will be very valuable to provide alerts in case of IoT anomalous behavior.
- Gradient Boosting classification algorithm improves accuracy of detection of IoT device type up to 85%.
- Combining Device Type information with OS , Vendor and Software information of IoT Devices is critical to alert manufacturers to provide security patches.

- Anomaly detection based on Network traffic based on IoT device type enhances the accuracy of our IoT abnormal behavior prediction. We recommend time-series based (real-time streaming) anomaly detection algorithms for this purpose.
- Integration with Spiderfoot automates the process of security alerts both for the customer of the IoT devices and manufacturers of the devices.

3.2. Recommendations for future study

An extensive IoT fingerprint database can be integrated into SpiderFoot framework using extension module providing a ready made usable list of commonly used IoT devices such as sensors, thermostats etc. We can use protocols such as MQTT to send/receive messages to compatible IoT devices to check their software versions for enhanced monitoring. We also integrating evaluating real-time streaming based time-series anomaly detection algorithms for network traffic analysis.

References

1. SM7 Software. (n.d.). Documentation - Spiderfoot. Retrieved January 28, 2022, from https://www.spiderfoot.net/documentation/
2. N. Ammar, L. Noirie and S. Tixeuil, "Network-Protocol-Based IoT Device Identification," 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC), 2019, pp. 204-209, doi: 10.1109/FMEC.2019.8795318.
3. M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. -R. Sadeghi and S. Tarkoma, "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, doi: 10.1109/ICDCS.2017.283.
4. Kostas, K., Just, M., & Lones, M. A. (2021). Iotdevid: A behavior-based device identification method for the iot. *ArXiv:2102.08866 [Cs]*. http://arxiv.org/abs/2102.08866
5. Bezawada, B., Bachani, M., Peterson, J., Shirazi, H., Ray, I., & Ray, I. (2018). Iotsense: Behavioral fingerprinting of iot devices. *CoRR*, *abs/1804.03852*. http://arxiv.org/abs/1804.03852
6. Meidan, Y., Bohadana, M., Shabtai, A., Ochoa, M., Tippenhauer, N. O., Guarnizo, J. D., & Elovici, Y. (2017). Detection of unauthorized iot devices using machine learning techniques. *CoRR*, *abs/1709.04647*. http://arxiv.org/abs/1709.04647
7. Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J. D., Ochoa, M., Tippenhauer, N. O., & Elovici, Y. (2017). ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis. *Proceedings of the Symposium on Applied Computing*, 506–509. https://doi.org/10.1145/3019612.3019878

8. Studytonight Technologies Pvt. Ltd. (n.d.). Network programming with python. Studytonight.com. Retrieved February 14, 2022, from https://www.studytonight.com/network-programming-in-python/
9. Bekerman, D., Shapira, B., Rokach, L., & Bar, A. (2015). Unknown malware detection using network traffic classification. *2015 IEEE Conference on Communications and Network Security (CNS)*. https://doi.org/10.1109/CNS.2015.7346821
10. Raff, S. (2022, February 13). *Awesome MQTT*. GitHub. https://github.com/hobbyquaker/awesome-mqtt
11. 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"
12. Boehmke, B., & Greenwell, B. (2020). Hands-on machine learning with R. CRC Press.
13. A. Vikram and Mohana, "Anomaly detection in Network Traffic Using Unsupervised Machine learning Approach," 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 476-479, doi: 10.1109/ICCES48766.2020.9137987.
14. H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha and K. R. Choo, "A Two-Layer Dimension Reduction and Two-Tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks," in IEEE Transactions on Emerging Topics in Computing, vol. 7, no. 2, pp. 314-323, 1 April-June 2019, doi: 10.1109/TETC.2016.2633228.

Appendix A – Feature List

ack,ack_A,ack_B,bytes,bytes_A,bytes_A_B_ratio,bytes_B,ds_field_A,ds_field_B,duration,http_GET,http_POST,http_bytes_avg,http_bytes_entropy,http_bytes_firstQ,http_bytes_max,http_bytes_median,http_bytes_min,http_bytes_stdev,http_bytes_sum,http_bytes_thirdQ,http_bytes_var,http_cookie_count,http_cookie_values_avg,http_cookie_values_entropy,http_cookie_values_firstQ,http_cookie_values_max,http_cookie_values_median,http_cookie_values_min,http_cookie_values_stdev,http_cookie_values_sum,http_cookie_values_thirdQ,http_cookie_values_var,http_count_host,http_count_req_content_type,http_count_resp_code,http_count_resp_content_type,http_count_transactions,http_count_user_agents,http_dom_host_alexaRank,http_dom_resp_code,http_has_location,http_has_referrer,http_has_req_content_type,http_has_resp_content_type,http_has_user_agent,http_inter_arrivel_avg,http_inter_arrivel_entropy,http_inter_arrivel_firstQ,http_inter_arrivel_max,http_inter_arrivel_median,http_inter_arrivel_min,http_inter_arrivel_stdev,http_inter_arrivel_sum,http_inter_arrivel_thirdQ,http_inter_arrivel_var,http_req_bytes_avg,http_req_bytes_entropy,http_req_bytes_firstQ,http_req_bytes_max,http_req_bytes_median,http_req_bytes_min,http_req_bytes_stdev,http_req_bytes_sum,http_req_bytes_thirdQ,http_req_bytes_var,http_resp_bytes_avg,http_resp_bytes_entropy,http_resp_bytes_firstQ,http_resp_bytes_max,http_resp_bytes_median,http_resp_bytes_min,http_resp_bytes_stdev,http_resp_bytes_sum,http_resp_bytes_thirdQ,http_resp_bytes_var,http_time_avg,http_time_entropy,http_time_firstQ,http_time_max,http_time_median,http_time_min,http_time_stdev,http_time_sum,http_time_thirdQ,http_time_var,packet_inter_arrivel_A_avg,packet_inter_arrivel_A_entropy,packet_inter_arrivel_A_firstQ,packet_inter_arrivel_A_max,packet_inter_arrivel_A_median,packet_inter_arrivel_A_min,packet_inter_arrivel_A_stdev,packet_inter_arrivel_A_sum,packet_inter_arrivel_A_thirdQ,packet_inter_arrivel_A_var,packet_inter_arrivel_B_avg,packet_inter_arrivel_B_entropy,packet_inter_arrivel_B_firstQ,packet_inter_arrivel_B_max,packet_inter_arrivel_B_median,packet_inter_arrivel_B_min,packet_inter_arrivel_B_stdev,packet_inter_arrivel_B_sum,packet_inter_arrivel_B_thirdQ,packet_inter_arrivel_B_var,packet_inter_arrivel_avg,packet_inter_arrivel_entropy,packet_inter_arrivel_firstQ,packet_inter_arrivel_max,packet_inter_arrivel_median,packet_inter_arrivel_min,packet_inter_arrivel_stdev,packet_inter_arrivel_sum,packet_inter_arrivel_thirdQ,packet_inter_arrivel_var,packet_size_A_avg,packet_size_A_entropy,packet_size_A_firstQ,packet_size_A_max,packet_size_A_median,packet_size_A_min,packet_size_A_stdev,packet_size_A_sum,packet_size_A_thirdQ,packet_size_A_var,packet_size_B_avg,packet_size_B_entropy,packet_size_B_firstQ,packet_size_B_max,packet_size_B_median,packet_size_B_min,packet_size_B_stdev,packet_size_B_sum,packet_size_B_thirdQ,packet_size_B_var,packet_size_avg,packet_size_entropy,packet_size_firstQ,packet_size_max,packet_size_median,packet_size_min,packet_size_stdev,packet_size_sum,packet_size_thirdQ,packet_size_var,packets,packets_A,packets_A_B_ratio,packets_B,push,push_A,push_B,reset,reset_A,reset_B,ssl_count_certificates,ssl_count_client_cipher_algs,ssl_count_client_ciphersuites,ssl_count_client_compressions,ssl_count_client_elliptic_curves,ssl_count_client_key_exchange_algs,ssl_count_client_mac_algs,ssl_count_server_ciphersuite,ssl_count_server_compression,ssl_count_server_elliptic_curve,ssl_count_server_name.ssl_count_transactio ns,ssl_count_version,ssl_dom_server_ciphersuite,ssl_dom_server_name_alexaRank,ssl_dom_version,ssl_handshake_duration_avg,ssl_handshake_duration_entropy,ssl_handshake_duration_firstQ,ssl_handshake_duration_max,ssl_handshake_duration_median,ssl_handshake_duration_min,ssl_handshake_duration_stdev,ssl_handshake_duration_sum,ssl_handshake_duration_thirdQ,ssl_handshake_duration_var,ssl_ratio_client_elliptic_curves,ssl_ratio_server_name,ssl_req_bytes_avg,ssl_req_bytes_entropy,ssl_req_bytes_firstQ,ssl_req_bytes_max,ssl_req_bytes_median,ssl_req_bytes_min,ssl_req_bytes_stdev,ssl_req_bytes_sum,ssl_req_bytes_thirdQ,ssl_req_bytes_var,ssl_resp_bytes_avg,ssl_resp_bytes_entropy,ssl_resp_bytes_firstQ,ssl_resp_bytes_max,ssl_resp_bytes_median,ssl_resp_bytes_min,ssl_resp_bytes_stdev,ssl_resp_bytes_sum,ssl_resp_bytes_thirdQ,ssl_resp_bytes_var,tcp_analysis_duplicate_ack,tcp_analysis_keep_alive,tcp_analysis_lost_segment,tcp_analysis_out_of_order,tcp_analysis_retransmission,tcp_analysis_reused_ports,ttl_A_avg,ttl_A_entropy,ttl_A_firstQ,ttl_A_max,ttl_A_median,ttl_A_min,ttl_A_stdev,ttl_A_sum,ttl_A_thirdQ,ttl_A_var,ttl_B_avg,ttl_B_entropy,ttl_B_firstQ,ttl_B_max,ttl_B_median,ttl_B_min,ttl_B_stdev,ttl_B_sum,ttl_B_thirdQ,ttl_B_var,ttl_avg,ttl_entropy,ttl_firstQ,ttl_maxs

Appendix B – IoT Datasets

1. https://www.stratosphereips.org/datasets-iot
2. https://www.stratosphereips.org/datasets-iot23

Malware datasets

- Posible Mirai: CTU-IoT-Malware-Capture-34-1
- Posible Mirai: CTU-IoT-Malware-Capture-34-2
- Posible XBash: CTU-IoT-Malware-Capture-41-1
- Possible Mirai: CTU-IoT-Malware-Capture-49-1
- Posible Mirai: CTU-IoT-Malware-Capture-52-1

Honeypots datasets

- Edimax IC-7113W CTU-Honeypot-Capture-2-154
- Edimax IC-7113W CTU-Honeypot-Capture-2-155

Full IoT Dataset:

- https://mcfp.felk.cvut.cz/publicDatasets/IoT-23-Dataset/iot_23_datasets_full.tar.gz

Download a lighter version containing only the labeled flows without the pcaps files (8.8 GB) here:

- https://mcfp.felk.cvut.cz/publicDatasets/IoT-23-Dataset/iot_23_datasets_small.tar.gz