Concordia Institute for Information System Engineering (CIISE)
Concordia University

ENGR 6991 Implementing IoT Sentinel: Automated Device-Type Identification for Security Enforcement in IoT

Submitted to:

Professor Makan Pourzandi

By:

Zahra Zohoorsaadat

December 2018

Table of Contents

List of Tables

1. Introduction

Recently there is a rapid growth of Internet of Things (IoT), and security concerns of using IoT devices becomes remarkable. IoT has so many applications in our life such as Home automation, Healthcare, Smart grid, Smart car, etc. While these applications communicate together and generate data to share, lead to many security concerns such as unauthorized access. On the other hand, applying wireless technology for communication causes many vulnerabilities that can be exploited by attackers. It means that without providing enough security the IoT benefits would be misused.

First time IoT was introduced in 1999 [1], following explosion of the wireless devices market, presentation of the Radio Frequency Identification (RFID) and the Wireless Sensor Networks (WSN) technologies [2]. The IoT concept purposes to connect anything with anyone, anytime, and anywhere, and this goal makes it vulnerable against unauthorized access by adversaries. One of the ideal ways to deal with the security issues would be to patch devices in order to remove weakness. This solution is impractical as most device vendors won't have intention to reduce the device vulnerabilities by providing such patches. On the other hand, most IoT users don't have enough information to perform such tasks or they even disregard to unplug IoT devices that have installed in their network and leave them with their old version of software [3].

M. Miettinen et al in [3] presented IoT SENTINEL. This system identifies the device types connected to a network and applies some security policies to mitigate the effect of vulnerable devices over the network. IoT SENTINEL controls the traffic flows of vulnerable devices in order to protect other devices in the network from threats and prevent data leakage.

This report explains how we implemented the distance discrimination section addressed in [3], and applied a Multi-Layer Perceptron classifier to compare our result with Random Forest Classifier defined in [4].

2. Previous works

A.Pitcher et al in [4] implemented the automated device-type identifier of IoT SENTINENTAL. They generated the network captured datasets by listening the traffic from the devices to the gateway during their setup phase. Each dataset correlates with a device type which is identified each time a new MAC address is detected and received by the gateway. Later, packet captured files (pcap) are filled with n packets { p_1, p_2, p_3, …, p_n} inside a directory with the name of device until the setup phase is done. For instant, WeMolnsightSwitch2 device directory contains MAC address of WeMolnsightSwitch2 iotdevice-mac.txt and Setup-C-*-STA.pcap files [4]:

WeMoInsightSwitch2/_iotdevice-mac.txt ---> 94:10:3e:42:80:69

WeMoInsightSwitch2/Setup-C-1-STA.pcap WeMoInsightSwitch2/Setup-C-2-STA.pcap

WeMoInsightSwitch2/Setup-C-3-STA.pcap WeMoInsightSwitch2/Setup-C-4-STA.pcap

WeMoInsightSwitch2/Setup-C-5-STA.pcap

The network mainly contains smart home devices including home routers and IP cameras. Table 1 shows the device type list [3].

Table 1: Device types

Identifier	Device Model	WiFi	ZigBee	Ethernet	Z-Wave	Other
Aria	Fitbit Aria WiFi-enabled scale	*				
D-LinkCam	D-Link HD IP Camera DCH-935L	*				
D-LinkDayCam	D-Link WiFi Day Camera DCS-930L	*		*		
D-LinkDoorSensor	D-Link Door & Window sensor				*	
D-LinkHomeHub	D-Link Connected Home Hub DCH-G020	*		*	*	
D-LinkSensor	D-Link WiFi Motion sensor DCH-S150	*				
D-LinkSiren	D-Link Siren DCH-S220	*				
D-LinkSwitch	D-Link Smart plug DSP-W215	*				
D-LinkWaterSensor	D-Link Water sensor DCH-S160	*				
EdimaxCam	Edimax IC-3115W Smart HD WiFi Network Camera	*		*		
EdimaxPlug1101W	Edimax SP-1101W Smart Plug Switch	*				
EdimaxPlug2101W	Edimax SP-2101W Smart Plug Switch	*				
EdnetCam	Ednet Wireless indoor IP camera Cube	*		*		
EdnetGateway	Ednet.living Starter kit power Gateway	*				*
HomeMaticPlug	Homematic pluggable switch HMIP-PS					*
HueBridge	Philips Hue Bridge model 3241312018		*	*		
HueSwitch	Philips Hue Light Switch PTM 215Z		*			
Lightify	Osram Lightify Gateway	*	*			
MAXGateway	MAX! Cube LAN Gateway for MAX! Home automation sensors			*		*
SmarterCoffee	Smarter SmarterCoffee coffee machine SMC10-EU	*				
TP-LinkPlugHS100	TP-Link WiFi Smart plug HS100	*				
TP-LinkPlugHS110	TP-Link WiFi Smart plug HS110	*				
WeMoInsightSwitch	WeMo Insight Switch model F7C029de	*				
WeMoLink	WeMo Link Lighting Bridge model F7C031vf	*	*			
WeMoSwitch	WeMo Switch model F7C027de	*				
Withings	Withings Wireless Scale WS-30	*				
iKettle2	Smarter iKettle 2.0 water kettle SMK20-EU	*				

Also, [4] implemented device fingerprinting and device identification phases which we explain them briefly in next sections.

2.1 Device Fingerprinting

This phase is capturing fingerprints from a distinguishable sequence of communications initiated by an inducted device. As mentioned in section 2, after observing a new MAC address, n received packets are recorded during the setup phase. Later, 23 features are extracted and generated a 23 × n matrix F which columns represent packets; pi = {f1,i, f2,i, f3,i, . . . , f23,i} that i ∈ {1, ..., n}, and rows represent fingerprints. Table 2 shows the extracted features from the packets. Moreover, they considered n = 12 for simplifying the classification step, and generated fingerprints with 276 features (12 packets contain 23 features each one). Also, table 3 indicates the number of traces per device.

Table 2: Content of 23 Packet features that are binary except those marked with "(int)"

Type	Features
Link layer protocol (2)	ARP/LLC
Network layer protocol (4)	IP/ICMP/ICMPv6/EAPoL
Transport layer protocol (2)	TCP/UDP
Aplication layer protocol (8)	HTTP / HTTPS / DHCP / BOOTP / SSDP / DNS / MDNS / NTP
IP options (2)	Padding / Router Alert
Packet content (2)	Size (int) / Raw data
IP address (1)	Destination IP counter (int)
Port class (2)	Source (int) / Destination (int)

Table 3: number of traces per device

No.	Device type	Fingerprint
1	Aria	84
2	D-LinkCam	631
3	D-LinkDayCam	114
4	D-LinkDoorSensor	324
5	D-LinkHomeHub	1331
6	D-LinkSensor	1068
7	D-LinkSiren	993
8	D-LinkSwitch	1090
9	D-LinkWaterSensor	1019
10	EdimaxCam	85
11	EdimaxPlug1101W	156
12	EdimaxPlug2101W	145
13	EdnetCam	45
14	EdnetGateway	128
15	HomeMaticPlug	101
16	HueBridge	2256
17	HueSwitch	3260
18	Lightify	628
19	MAXGateway	107
20	SmarterCoffee	26
21	TP-LinkPlugHS100	123
22	TP-LinkPlugHS110	112
23	WeMoInsightSwitch	825
24	WeMoLink	926
25	WeMoSwitch	635
26	Withings	122
27	iKettle2	21

2.2 Device Identification

In this phase a single classifier is created for each device type in table 1. [4] used Random Forest classification algorithm to build these models. A binary decision is provided by each classifier that means the input fingerprint matches the device-type or not. We implemented a binary generator code [5] to create the outputs of each model, and provided results in Appendix A.

3. Distance Discrimination

This section explains how we implemented Distance Discrimination Function [5] considering provided algorithm in [1]. After classification phase, an unknown fingerprint F' may match several device types. We compared F' to fingerprints from each device type it got a match for. The fingerprint comparison was based on computing distance between two fingerprints feature by feature, and it was done by two approaches.

1. Choosing a subset of fingerprint dataset: In this approach based on [1], we chose a subset of five fingerprints randomly, and compute distance between F' and each of five fingerprints.
2. Choosing the whole fingerprint datasets: In this approach, we considered the whole dataset, and compute the distance between F' and each fingerprint of dataset.

Equality of distance between two fingerprints was considered if all features F' from a packet p_i were equal to those of another packet p_j. The obtained absolute distance between two fingerprints, was divided by the length of the longest one to provide a normalized distance value bounded on [0, 1]. Then distances were summed up per device-type and divided to the number of samples which were chosen from dataset to get a global dissimilarity score $s_i \in [0, 1]$ of F' with the type D_i. The lowest dissimilarity score s_i gave the final predicted device-type for F'. After several runs with different F', approach 2 provided more acceptable result compare to approach 1.

Although the accuracy was different from one dataset to another, the result of approach 2 was better while it did not ignore any fingerprint information in a dataset. Table 4 shows the results of 15 runs with choosing F' from Aria randomly.

Table 4: Comparing F' to dataset of matched Device type.

a) Approach 1

Test No.	Fingerprint No.	Matched Model	Score
1	66	Aria	0.332826
2	81	Aria	0.295324
3	60	Aria	0.295324
4	33	Aria	0.333817
5	62	Aria	0.332826
6	66	Aria	0.332826
7	53	HomeMaticPlug	0.199896
8	63	iKettle2	0.172431
9	6	HomeMaticPlug	0.199896
10	29	Aria	0.333817
11	6	HomeMaticPlug	0.199896
12	6	HomeMaticPlug	0.199896
13	66	Aria	0.332826
14	27	Aria	0.295324
15	41	Aria	0.332826

b) Approach 2

Test No.	Fingerprint No.	Matched Model	Score
1	64	Aria	0.306836
2	12	TP-LinkPlugHS110	0.321843
3	42	SmarterCoffee	0.296458
4	52	TP-LinkPlugHS110	0.371051
5	39	TP-LinkPlugHS100	0.370042
6	8	EdnetCam	0.289826
7	79	Aria	0.410893
8	64	Aria	0.415247
9	18	TP-LinkPlugHS100	0.345295
10	42	Aria	0.295
11	45	EdimaxPlug2101W	0.393789
12	10	iKettle2	0.209604
13	41	D-LinkDayCam	0.280664
14	4	WeMoSwitch	0.400051
15	69	EdimaxPlug1101W	0.351271

4. Multi-Layer Perceptron Classifier

We applied Multi-Layer Perceptron (MLP) classifier for our datasets. We also used sklearn MLP classifier [6][7] which implements MLP algorithm that trains using Backpropagation. MLP trains on two arrays: array X of size (n_samples, n_features), which holds the training samples represented as floating point feature vectors; and array y of size (n_samples,), which holds the target values (class labels) for the training samples [6]. In python code implementation, we considered defaults of Random Forest classifier [3][4].

4.1 MLP Classifier Setup

Random forest classifier was trained by 70% of training data, and 30% of testing data, and we applied it for our MLP classifier. We created our training set consists of targeted device fingerprints, and a random selection of all other device fingerprints as two different classes. Then we concatenated these two classes to generate the MLP training set. The python code was provided in [5].

We considered some important parameters for training our MLP classifier such as hidden layer size, solver, learning rate, etc. and change them to get better result. Following description is the brief definition for these parameters [6].

- Hidden_layer_sizes: defines tuple, length = n_layers - 2, default (100,), and determines the number of hidden layers and neurons in the classifier. Default (100,) means if no value is provided for hidden_layer_sizes then default architecture will have one input layer, one hidden layer with 100 units, and one output layer.
- Solver: is the argument to set the optimization algorithm. There are different solver options such as lbfgs, adam and sgd which adam is default. In general setting sgd (stochastic gradient descent) works best, and achieves faster convergence. But we got better result with adam.
- Learning rate: schedules for weight updates. There are different options such as constant, invscaling, and adaptive which constant is default.

- Activation: There are different options for activation such as identity, logistic, relu, and tanh which relu is default. Also, relu is the simplest and most useful activation function.

4.2 MLP Classifier accuracy

Table 5 shows the accuracy of our MLP classifier with different parameters. Appendix B shows the accuracy result compare to RF classifier.

Table 5: accuracy of MLP Classifier with different parameters

No.	Device type	MLP(lbfgs,50/20)	MLP(adam,50/20)	MLP(sgd,50/20)	MLP(sgd,50/20,relu)	MLP(sgd,50/20,logistic)	MLP(adam,50/20,logistic)	MLP(adam,50/20,logistic,invscaling)
1	Aria	0.775510204	0.857142857	0.775510204	0.816326531	0.448979592	0.87755102	0.816326531
2	D-LinkCam	0.886178862	0.926829268	0.799457995	0.829268293	0.688346883	0.918699187	0.945799458
3	D-LinkDayCam	0.818181818	0.742424242	0.742424242	0.787878788	0.560606061	0.742424242	0.818181818
4	D-LinkDoorSensor	0.837696335	0.858638743	0.811518325	0.769633508	0.664921466	0.952879581	0.97382199
5	D-LinkHomeHub	0.935400517	0.913436693	0.474160207	0.674418605	0.634366925	0.940568475	0.967700258
6	D-LinkSensor	0.875202593	0.915721232	0.74554295	0.510534846	0.635332253	0.952998379	0.943273906
7	D-LinkSiren	0.882661996	0.922942207	0.740805604	0.683012259	0.604203152	0.935201401	0.952714536
8	D-LinkSwitch	0.911392405	0.916139241	0.563291139	0.560126582	0.607594937	0.952531646	0.963607595
9	D-LinkWaterSensor	0.877342419	0.933560477	0.722316865	0.741056218	0.608177172	0.977853492	0.960817717
10	EdimaxCam	0.918367347	0.857142857	0.836734694	0.816326531	0.448979592	0.734693878	0.836734694
11	EdimaxPlug1101W	0.787234043	0.659574468	0.723404255	0.659574468	0.64893617	0.808510638	0.691489362
12	EdimaxPlug2101W	0.662650602	0.734939759	0.722891566	0.746987952	0.481927711	0.746987952	0.771084337
13	EdnetCam	0.533333333	0.766666667	0.7	0.7	0.533333333	0.7	0.733333333
14	EdnetGateway	0.705128205	0.871794872	0.743589744	0.794871795	0.538461538	0.897435897	0.91025641
15	HomeMaticPlug	0.887096774	0.822580645	0.790322581	0.903225806	0.725806452	0.967741935	0.85483871
16	HueBridge	0.96867838	0.98013751	0.954927426	0.905271199	0.888464477	0.993888464	0.992360581
17	HueSwitch	0.984599044	0.984599044	0.983005842	0.983005842	0.944237918	0.996282528	0.99787573
18	Lightify	0.927777778	0.963888889	0.927777778	0.894444444	0.780555556	0.955555556	0.961111111
19	MAXGateway	0.921875	0.9375	0.9375	0.890625	0.5	0.953125	0.9375
20	SmarterCoffee	0.8125	0.875	0.875	0.875	0.625	0.75	0.875
21	TP-LinkPlugHS100	0.68115942	0.797101449	0.68115942	0.724637681	0.463768116	0.724637681	0.579710145
22	TP-LinkPlugHS110	0.738461538	0.8	0.8	0.830769231	0.523076923	0.769230769	0.676923077
23	WeMoInsightSwitch	0.845991561	0.892405063	0.605485232	0.715189873	0.61814346	0.951476793	0.947257384
24	WeMoLink	0.889925373	0.958955224	0.858208955	0.84141791	0.63619403	0.996268657	0.986940299
25	WeMoSwitch	0.843243243	0.913513514	0.772972973	0.675675676	0.559459459	0.935135135	0.945945946
26	Withings	0.705882353	0.75	0.75	0.661764706	0.470588235	0.75	0.720588235
27	iKettle2	0.933333333	0.933333333	1	1	1	0.933333333	1

5. Result Improvement

The first phase of every machine learning process is providing the proper data for ML methods. The data preparation phase refers to transforming the raw data to the understandable format for the ML method which includes data preprocessing and feature engineering that are explained in this section. After, we explain normalization method which we applied to improve the accuracy of Random Forest classifier.

5.1 Data preprocessing

The goal of data preprocessing is preparing the raw data for feature processing. We can apply below strategies to find the best one for our datasets.

1. Imbalanced Data: The problem of imbalanced data has a big impact on the results of supervised learning. The most popular strategies are under-sampling, over-sampling or mix of both. The right strategy mainly depends on data. [8] provides the python package that contains different sampling algorithms.
2. More preprocessing methods: Depends on ML method, we can apply more preprocessing methods on data. [9] provides sklearn package for preprocessing.

5.2 Feature Engineering [10]

Feature engineering consists two steps:

1. Feature creation: creating additional features out of existing dataset
    - manual: it can be done by a data scientist; relying on domain knowledge, intuition, and data manipulation.
    - automated: python libraries [11]
2. Feature selection: select the most important features for training. [12] presents more explanations and methods for feature selection. Moreover, the features might be selected by eliminating features [13], calculating its importance [14], and visualization of feature importance [15].

5.3 Normalization

We found out the accuracy of our classifiers, RF and MLP, is very sensitive to the feature scaling, so the raw data should be preprocessed to obtain a better result. Considering table 2 and dataset content, we found out packet content feature which is integer and stores raw data has a significant effect on classifier confusion. So, with preprocessing this feature we could obtain more

accurate classifier output. Based on our input, we implemented normalization function [5] for packet content feature, and provided results in Appendix A.

6. Future Work

Although we obtained an acceptable result after normalization dataset, we still have confusion in our classifiers. In order to segregate and remove the confusion on confused classifiers which are being the ones selected after RF, we suggest using PCA to find the most representative features for the confused classifiers. Moreover, we recommend removing the most representative features, retraining a new set of classifiers, and reapplying the tests only to those newly trained classifiers to distinguish between them. This is similar to multi-layer or multi-view classification. For example, Test1 fingerprint is sent to RF; three classifiers are confused C1, C2, C3. After applying PCA on C1, C2, C3, we determine features 2, 13, 24 being the most representative. So, we remove those features in the data sets for C1, C2 and C3. Now we retrain classifiers, and we obtain C'1, C'2, C'3. Finally, we resubmit Test1 to C'1, C'2, C'3 which could now better distinguish between them.

7. Conclusion

In this report we provided an overview on presented work in [4]. We proposed a distance discrimination approach in which an unknown fingerprint is compared to the whole dataset of device type it got a match for. By comparison with presented approach in [4], our result was more acceptable. Moreover, we implemented a Multi-Layer Perceptron (MLP) classifier and compared its accuracy with Random Forest (RF) presented in [4]. The result showed that MLP classifier is more sensitive to its input features, so it is highly recommended to find and apply an appropriate approach for data preprocessing. Finally, we implemented a normalization method for raw data preprocessing that improved significantly our RF classifiers' accuracy.

Acknowledgement

We would like to thank Professor Makan Pourzandi for his supervision as well as his support and help to drive us to work on IOT Sentinel project. Also, we would like to thank Dr. Amine Boukhtouta for sharing his data with us at the beginning, and his valuable comments during the project.

References

[1] K. Ashton, "That ‘internet of things’ thing," *RFID Journal,* vol. 22, *(7),* pp. 97-114, 2009.

[2] J. Gubbi *et al*, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Comput. Syst.,* vol. 29, *(7),* pp. 1645-1660, 2013.

[3] M. Miettinen *et al*, "IoT sentinel: Automated device-type identification for security enforcement in IoT," in Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on, 2017.

[4] A. Pitcher *et al*, "Implementing IoT SECTINEL: Automated Device-Type Identification for Security Enforcement in IoT," ENGR Project Report, 2018

[5] https://github.com/zohoorsaadat/IoT_Sentinel

[6] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[7] https://scikit-learn.org/stable/modules/neural_networks_supervised.html#tips-on-practical-use

[8] https://imbalanced-learn.readthedocs.io/en/stable/introduction.html

[9] https://scikit-learn.org/stable/modules/preprocessing.html

[10] https://towardsdatascience.com/automated-feature-engineering-in-python-99baf11cc219

[11] https://docs.featuretools.com

[12] https://scikit-learn.org/stable/modules/feature_selection.html

[13] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE

[14] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html#sklearn.ensemble.ExtraTreesClassifier

[15] https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html#sphx-glr-auto-examples-ensemble-plot-forest-importances-py

Appendix A

We provided all model outputs in [5]. Table 6 shows Random Forest outputs for models Aria and D-LinkCam.

Table 6: The output file of models Aria and D-LinkCam

No.	Aria	Matched Device	No.	D-LinkCam	Matched Device
1	Aria	1	1	Aria	0
2	D-LinkCam	0	2	D-LinkCam	1
3	D-LinkDayCam	1	3	D-LinkDayCam	1
4	D-LinkDoorSensor	0	4	D-LinkDoorSensor	0
5	D-LinkHomeHub	0	5	D-LinkHomeHub	1
6	D-LinkSensor	0	6	D-LinkSensor	1
7	D-LinkSiren	0	7	D-LinkSiren	1
8	D-LinkSwitch	0	8	D-LinkSwitch	1
9	D-LinkWaterSensor	0	9	D-LinkWaterSensor	1
10	EdimaxCam	0	10	EdimaxCam	1
11	EdimaxPlug1101W	1	11	EdimaxPlug1101W	1
12	EdimaxPlug2101W	1	12	EdimaxPlug2101W	1
13	EdnetCam	1	13	EdnetCam	1
14	EdnetGateway	0	14	EdnetGateway	1
15	HomeMaticPlug	1	15	HomeMaticPlug	1
16	HueBridge	0	16	HueBridge	1
17	HueSwitch	0	17	HueSwitch	1
18	Lightify	0	18	Lightify	1
19	MAXGateway	1	19	MAXGateway	0
20	SmarterCoffee	1	20	SmarterCoffee	1
21	TP-LinkPlugHS100	1	21	TP-LinkPlugHS100	1
22	TP-LinkPlugHS110	1	22	TP-LinkPlugHS110	1
23	WeMoInsightSwitch	0	23	WeMoInsightSwitch	1
24	WeMoLink	0	24	WeMoLink	1
25	WeMoSwitch	1	25	WeMoSwitch	1
26	Withings	1	26	Withings	0
27	iKettle2	1	27	iKettle2	1
	Noise Ratio	**0.461538462**		**Noise Ratio**	**0.846153846**
	Data Set Size	84		Data Set Size	631

Appendix B

We provided all result in [5]. Table 7 shows the accuracy results of MLP and RF classifiers.

Table 7: Results of MLP and RF classifiers

No.	Device type	RF	MLP(lbfgs,50/20)	MLP(adam,50/20)	MLP(sgd,50/20)	MLP(sgd,50/20,relu)
1	Aria	0.959183673	0.775510204	0.857142857	0.775510204	0.816326531
2	D-LinkCam	0.994579946	0.886178862	0.926829268	0.799457995	0.829268293
3	D-LinkDayCam	0.96969697	0.818181818	0.742424242	0.742424242	0.787878788
4	D-LinkDoorSensor	1	0.837696335	0.858638743	0.811518325	0.769633508
5	D-LinkHomeHub	0.988372093	0.935400517	0.913436693	0.474160207	0.674418605
6	D-LinkSensor	0.97082658	0.875202593	0.915721232	0.74554295	0.510534846
7	D-LinkSiren	0.96497373	0.882661996	0.922942207	0.740805604	0.683012259
8	D-LinkSwitch	0.97943038	0.911392405	0.916139241	0.563291139	0.560126582
9	D-LinkWaterSensor	0.967632027	0.877342419	0.933560477	0.722316865	0.741056218
10	EdimaxCam	0.959183673	0.918367347	0.857142857	0.836734694	0.816326531
11	EdimaxPlug1101W	0.925531915	0.787234043	0.659574468	0.723404255	0.659574468
12	EdimaxPlug2101W	0.987951807	0.662650602	0.734939759	0.722891566	0.746987952
13	EdnetCam	0.9	0.533333333	0.766666667	0.7	0.7
14	EdnetGateway	0.948717949	0.705128205	0.871794872	0.743589744	0.794871795
15	HomeMaticPlug	1	0.887096774	0.822580645	0.790322581	0.903225806
16	HueBridge	0.993124523	0.96867838	0.98013751	0.954927426	0.905271199
17	HueSwitch	0.998937865	0.984599044	0.984599044	0.983005842	0.983005842
18	Lightify	0.997222222	0.927777778	0.963888889	0.927777778	0.894444444
19	MAXGateway	0.984375	0.921875	0.9375	0.9375	0.890625
20	SmarterCoffee	1	0.8125	0.875	0.875	0.875
21	TP-LinkPlugHS100	0.956521739	0.68115942	0.797101449	0.68115942	0.724637681
22	TP-LinkPlugHS110	0.923076923	0.738461538	0.8	0.8	0.830769231
23	WeMoInsightSwitch	0.96835443	0.845991561	0.892405063	0.605485232	0.715189873
24	WeMoLink	0.998134328	0.889925373	0.958955224	0.858208955	0.84141791
25	WeMoSwitch	0.972972973	0.843243243	0.913513514	0.772972973	0.675675676
26	Withings	0.941176471	0.705882353	0.75	0.75	0.661764706
27	iKettle2	0.866666667	0.933333333	0.933333333	1	1

Appendix C

We provided all model outputs in [5]. Table 8 shows Random Forest outputs for models Aria and D-LinkCam.

Table 8: The output file of models Aria and D-LinkCam

No.	Aria	Matched Device	No.	D-LinkCam	Matched Device
1	Aria	1	1	Aria	0
2	D-LinkCam	0	2	D-LinkCam	1
3	D-LinkDayCan	1	3	D-LinkDayCan	1
4	D-LinkDoorSe	0	4	D-LinkDoorSe	0
5	D-LinkHomeH	0	5	D-LinkHomeH	0
6	D-LinkSensor	0	6	D-LinkSensor	0
7	D-LinkSiren	0	7	D-LinkSiren	0
8	D-LinkSwitch	0	8	D-LinkSwitch	0
9	D-LinkWaterS	0	9	D-LinkWaterS	0
10	EdimaxCam	0	10	EdimaxCam	0
11	EdimaxPlug11	0	11	EdimaxPlug11	0
12	EdimaxPlug21	0	12	EdimaxPlug21	0
13	EdnetCam	1	13	EdnetCam	1
14	EdnetGateway	0	14	EdnetGateway	0
15	HomeMaticPl	1	15	HomeMaticPl	0
16	HueBridge	0	16	HueBridge	0
17	HueSwitch	0	17	HueSwitch	0
18	Lightify	0	18	Lightify	0
19	MAXGateway	0	19	MAXGateway	0
20	SmarterCoffee	0	20	SmarterCoffee	0
21	TP-LinkPlugHS	0	21	TP-LinkPlugHS	0
22	TP-LinkPlugHS	0	22	TP-LinkPlugHS	0
23	WeMoInsight	0	23	WeMoInsight	0
24	WeMoLink	0	24	WeMoLink	0
25	WeMoSwitch	0	25	WeMoSwitch	0
26	Withings	0	26	Withings	0
27	iKettle2	1	27	iKettle2	1
	Noise Ratio	**0.153846154**		**Noise Ratio**	**0.115384615**
	Data Set Size	84		Data Set Size	631