A Comprehensive Study of Supervised Machine Learning Assisted Approaches for IoT Device Identification

Yong Wang*, Bhaskar P. Rimal†, Carson Koball*, Michael Fahnlander*, Julia Scheaffer*
Joshua Hammrich* Paolo Gentry* Dylan Westlund* Tyler Salmen* Connor Ford*
* PATRIOT Lab, Dakota State University, Madison, SD, USA
†Department of Computer Science, University of Idaho, Moscow, ID, USA
yong.wang@dsu.edu, bhaskar.rimal@ieee.org, {carson.koball, michael.fahnlander,
julia.scheaffer, joshua.hammrich, paolo.gentry, dylan.westlund, tyler.salmen}
@trojans.dsu.edu, connor.ford@dsu.edu

***Abstract*—Device identification is a fundamental issue in the Internet of Things. A few studies in modern literature indicate that machine learning approaches could be used for device identification. However, the hypothesis that device behavior could be characterized by machine learning techniques for device identification has not been thoroughly investigated. Therefore, we conduct a comprehensive study to examine this hypothesis. We create both a trusted and untrusted environment for experimental testing scenarios. That contains four testing cases, including intra-network, network perimeter, cryptojacking, and DOS. Six supervised machine learning classifiers are selected and evaluated. Among the six classifiers, the AdaBoost classifier with 200 features achieves testing accuracies of 88.23% and is chosen for the testing cases. Our evaluation results show that the AdaBoost classifier is promising for a trusted environment. However, the accuracies of the AdaBoost classifier drop dramatically to less than 20% in both cryptojacking and DOS cases. While the results do not support the hypothesis, the challenges faced by machine learning-assisted approaches in device identification could be complemented by other safeguards such as whitelists and intrusion detection and prevention systems. This paper further discusses future work, including using features from physical layers to examine the hypothesis.**

***Index Terms*—Device Identification, Internet of Things, Machine Learning, CICFlowMeter++, Cryptojacking, DOS**

I. Introduction

The Internet of Things (IoT) is the interconnection of computing devices embedded in everyday objects to the Internet through a home, business, or institutional network [1]. Device identification is a fundamental issue in the IoT. Many critical services such as access control and intrusion prevention are built on correctly identifying each device. The challenges for identifying IoT devices include, but are not limited to, 1) scalability: IoT may include billions of connected devices, and the solution must be scalable; 2) adaptability: IoT devices come in different form factors, operating systems (OSs), manufacturers, and protocols to communicate to other devices. The solution must be able to adapt to a variety of IoT devices; and 3) resistance: IoT devices might be compromised, and the solution must be effective even when they are compromised. Such limitations have been found in existing studies, e.g., [2], [3]. Developing a new approach for device identification to address these issues comprehensively is vital.

We utilize machine learning (ML) analytical approaches for IoT device identification, as features from network traffic may be used to characterize IoT devices without interrupting the examined networks. Few studies, e.g., [2], [3], indicated that ML-based approaches could be used for device identification. However, these studies were all conducted in trusted environments. Importantly, we have noted two questions that have not yet been addressed in the literature. First, it is uncertain if the device classifiers developed from ML are consistent within the network and on the network perimeter. Second, further evaluation is required to determine if the device classifier is effective for device identification when a device is compromised.

In this paper, we conduct a study to test the hypothesis that device behavior can be characterized by ML for device identification. To overcome the limitations of existing studies, we create a comprehensive testing environment, including both trusted and untrusted scenarios. We design two testing cases for the trusted environment, including devices located within a network and devices located on the network perimeter, to test ML for device identification. For the untrusted environment, we design two additional testing cases, including cryptojacking and `DOS` attacks, to evaluate if a device classifier is effective for device identification when a device is compromised. Six supervised ML classifiers, including AdaBoost, Decision Tree, K-Nearest Neighbor, Logistic Regression, Random Forest, and LinearSVC, are selected and experimented with for device identification. Our detailed evaluation results reveal new features as well as new challenges when utilizing ML-assisted approaches for device identification.

The main contributions of this paper are summarized as follows. 1) We create four testing cases to demonstrate the performance of supervised ML within a network, on the network perimeter, and when a device is compromised, 2) We expand CICFlowMeter (formerly ISCXFlowMeter) [4] and create a new feature extraction tool known as CICFlowMeter++,

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

adding several new features in the process. 3) The existing research on ML for device identification was conducted in a trusted environment. While such efforts are important, to the best of our knowledge, ours is the first effort to study ML for device identification in a compromised network environment, and 4) Our experimental evaluation uses a dataset comprised of 16 typical commercial IoT devices that show promising results. We will make our datasets and the CICFlowMeter++ implementation available for research use.

The remainder of this paper is structured as follows. Section II presents related work. Section III describes the proposed ML-assisted approach for IoT device identification, followed by the introduction of implementation in Section IV. The testing, evaluation results, and discussions are presented in Section V. Finally, Section VI concludes the paper with future work.

II. Related Work

Many features from network flows have been used for characterizing device behavior. In [3], authors demonstrated supervised ML-assisted approaches using 80 features in CICFlowMeter [4]. In [5], 972 behavioral features across different protocols and network layers are extracted from network traffic for detecting malware.

In [6], supervised ML techniques were adopted for device identification by utilizing unique flow-based features to create a fingerprint for each device. In [2], authors apply supervised ML on network traffic for device identification. Their ML model includes two stages: 1) a classifier distinguishes between IoT and non-IoT devices, and 2) a specific IoT device classifier is used to identify each IoT device.

In [7], the authors propose a classification approach utilizing the TCP variant as an input feature to identify the OS on a device by leveraging ML and deep learning techniques. Supervised ML is used in [8] and [6], where the approaches use flow statistics to identify IoT devices in real time. [8] filters out noisy features with a genetic algorithm, and [6] utilizes 67 relevant features. In [9], the authors propose a method for traffic classification and application identification using an unsupervised ML technique. Features such as $Forward-Pkt-Len-Var$, $Backward-Pkt-Len-Var$, and $Backward-Bytes$ are extracted to identify applications including FTP, Telnet, SMTP, DNS, HTTP, AOL, Messenger, Napster, and Half-Life. In [10], authors use unsupervised clustering to identify IoT device types in network flow traffic. In [11], the authors use unsupervised ML on data captured directly from the devices to identify cycles in the flow data relating to how often and how predictable the transmission of data is. Unsupervised deep learning is used in [12] and [13], where ML autoencoders combined with clustering algorithms are used to identify arbitrary device types.

Results from [2] and [3] show that supervised ML is a promising method for device identification. Approaches such as [2] and [3] assume device behavior could be characterized through ML. The assumption appears to be true based on observation. However, the following important limitations have been found in the existing ML-assisted approaches used for device identification and are addressed in this paper.

1) All evaluation was conducted in the intra-network without regard to the perimeter of the network.
2) All evaluation was conducted in a trusted environment without regard to a compromised environment.

III. Supervised ML Assisted Approach for Device Identification

A. Device Identification

An IoT device is a physical object that provides one or more functions within a computer system. Device attributes in a device profile include OS, version, MAC address, IP address, model, applications, and other identifying criteria. Device attributes can be a single element like a MAC address or a vector of elements, such as running applications.

Device identification is a systematic process that aims to map attributes in a device profile and uniquely identify each device in a network. A unique device identifier (UDI) is one or more device attributes that can be used to uniquely identify a device. A MAC address can be used as a unique device identifier in a trusted environment. However, a MAC address can be easily spoofed if the device is compromised. This raises security questions regarding the use of such attributes for UDI development in an untrusted environment. Device identification is a fundamental issue in identity and access management. Note that due to the variety of IoT devices that exist, it is very challenging to profile every attribute of a device. This paper utilizes Definition 2 for device identification.

Definition 1. A device is identified when every attribute in the device profile is known and a unique device identifier is found.

Definition 2. A device is identified when a unique device identifier is found.

B. Supervised ML-Based Approach

Fig. 1 shows the approach used to examine the hypothesis in this research, including three phases, i.e., data preparation, ML qualification, and ML validation.

1) Data Preparation: Supervised ML requires a large amount of data to optimize the applied models. Fig. 3 shows our testing network for data collection.

IoT devices are connected to the testing network and the network traffic is collected using `tcpdump` and RaspAP [14]. To test supervised ML on the network perimeter, a network tapping device, `nTAP`, is placed before the traffic reaches the firewall. `nTAP` is a passive, full-duplex monitoring device that provides visibility into the network regardless of traffic. The collected data in `pcap` files proceed through a data preparation process before being used for ML.

2) ML Qualification: Fig. 2 shows the procedure for ML qualification, which includes three steps. Step #1 Feature extraction: Network traffic includes device behavior that cannot be used for ML directly. Instead, features are extracted from these network traffic files for ML purposes. Step #2 Data

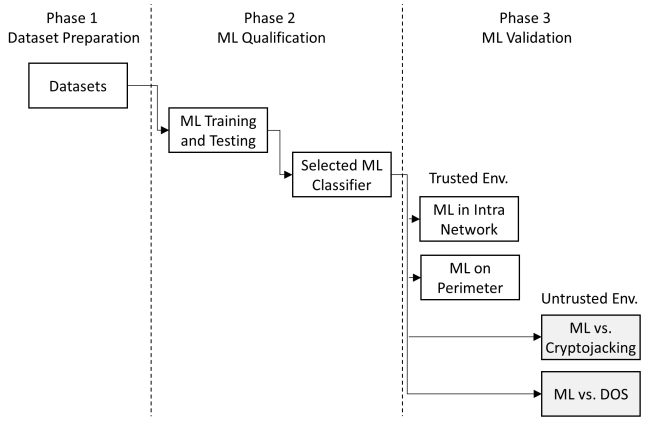

Fig. 1. Approach adopted for examining the hypothesis.

labeling: A label is created for each `tcp` flow based on source and destination addresses. Step #3 ML training and selection: A group of supervised ML classifiers is selected for training, and the best-performing classifier is selected for validating ML on intra-network and network perimeter traffic.

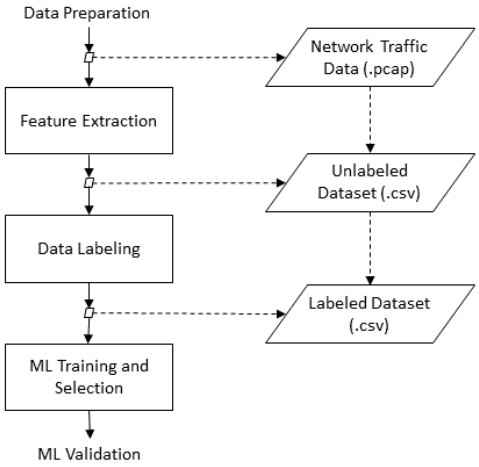

Fig. 2. ML-assisted approach for IoT device identification.

We selected six supervised ML classifiers for IoT device identification, including AdaBoost [15], Decision Tree [16], K-Nearest Neighbor [17], [18], Logistic Regression [19], Random Forest [20], and Linear Support Vector Classifier (LinearSVC) [21]. These six classifiers are selected based on the literature and their performance when processing large datasets. The six classifiers are tested and evaluated for ML validation.

3) ML Validation: When a device is adopted in a network, we consider a two-step process: device provisioning and operation. Device provisioning refers to the device onboarding process where a device is configured, calibrated, and tested for use in a production environment, and is often conducted in a controlled environment. Data preparation and ML qualification take place in this process due to its secure nature. Device operation refers to the process where a device operates without human intervention, often lacking any form of constant monitoring. Since devices generally operate in an untrusted environment and can be compromised, ML validation occurs in this process.

Two testing scenarios and four testing cases are created. **Testing Scenario #1** intends to test ML for device identification in a trusted environment. Testing Scenario 1 includes two testing cases: *Testing Case #1.1* ML within a network: The selected ML classifiers are tested using traffic collected from the intranet. *Testing Case #1.2* ML on the perimeter: The selected ML classifiers are tested using network border traffic.

Testing Scenario #2 intends to test ML for device identification in an untrusted environment. Testing Scenario 2 includes two testing cases: *Testing Case #2.1* ML for cryptojacking: Cryptojacking is the unauthorized use of computing resources to mine cryptocurrency. This testing case validates if a trained ML classifier can identify a device when it is compromised with cryptomining malware. *Testing Case #2.2* ML for `DOS` attacks: A denial-of-service (DoS) attack is a malicious attempt to disrupt the normal traffic of a targeted device with a flood of Internet traffic. This testing case validates if a trained ML classifier can identify devices in `DOS` attacks.

IV. Implementation

A. Experimental Setup

The testing network was set up as shown in Fig. 3. Table I shows a list of the networking devices used.

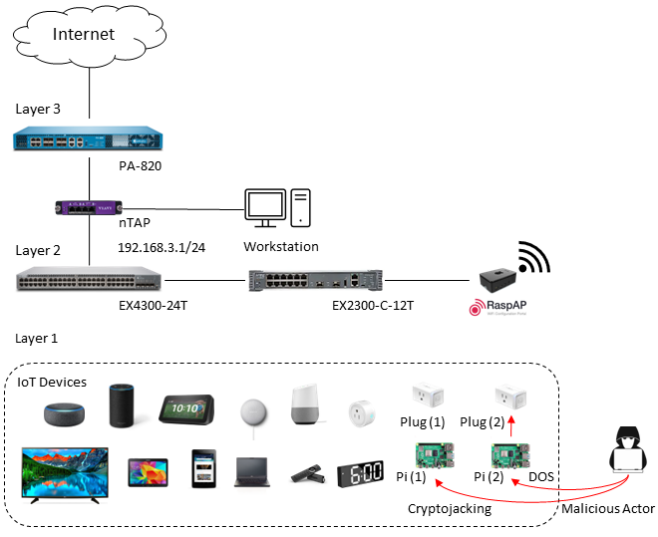

Fig. 3. Testing network setup.

TABLE I
Networking devices used in the testing network.

Device	Model	Manufacture
Firewall	PA-820	Palo Alto Networks
Switch	EX4300-24T	Juniper
nTAP	nTAP	Viavi Solutions
Wi-Fi access point	Raspberry Pi 4 Model B	RaspAP

It is important to differentiate traffic from each device. Capturing traffic through a regular Wi-Fi access point is ineffective due to network address translation. Two approaches, ARP poisoning or a Wi-Fi Pineapple device, could be utilized to collect network traffic and facilitate the data labeling process. ARP poisoning is intrusive and introduces extra traffic into the network capture. A Wi-Fi Pineapple is suitable for a small number of devices but encounters reliability issues when more testing devices are added to the network. Therefore, `RaspAP` is selected for this work because of its reliability and flexibility [14]. Table II shows a list of IoT devices used in the network. Two Raspberry Pis are also included in the testing network, where Raspberry Pi (1) and Raspberry Pi (2) are used to set up the cryptojacking testing and `DOS` testing, respectively.

B. Dataset Preparation

The study in [2] used a proprietary tool that was not publicly available to process data. We have selected an open-source tool - CICFlowMeter for our work. The study in [3] demonstrated a supervised ML-assisted approach using the original features in CICFlowMeter. Further, studies in [2] and [3] show differences between top features selected by ML for device identification. In this work, we have developed a tool, *CICFlowMeter++* by enhancing the original CICFlowMeter. CICFlowMeter++ can extract 233 features from a TCP flow and includes many new features from [2] for comparison.

Our major improvements in CICFlowMeter++ include 1) an improved and simplified feature generation process, 2) the resolution of multiple errors in CICFlowMeter-V4 that resulted in the inaccurate calculation of various packet data points, 3) the addition of 153 new features to assist ML algorithms in identifying IoT devices based on traffic characteristics.

Since we focus on characterizing device behavior on the Internet, only TCP flows between the devices and the Internet are considered in ML. TCP flows between IoT devices are not considered. TCP flows are labeled by the source and destination devices' IP addresses and MAC addresses. Then, a flow label is added to each TCP flow in the CSV file based on the single known IoT device in each flow.

C. ML Qualification

There are 8 features, including `Origin, Src IP, Src Port, Src MAC Addr, Dst IP, Dst Port, Dst MAC Addr`, and `Timestamp`, which are not used to fit the ML models as they can generally be used as device identifiers in a trusted environment. The original protocol feature from the CICFlowMeter++ is further broken into three new features. A total of 227 features are used for ML. The ML classifiers are implemented using python 3.8.8 [22] and Scikit-learn 0.24.1 [21]. The evaluation is conducted in Jupyter Notebook 6.3.0 [23].

Table III shows the hyperparameters used in the ML classifiers based on a semi-exhaustive search process. Tuning occurred in a 5-fold validation schema over the training set optimizing for (balanced) accuracy.

D. ML Validation

Test Case #1.2 validates the performance of ML classifiers on the perimeter of the network. The traffic captured from `nTAP` could not be used for testing directly since the `pcap` file cannot differentiate traffic from multiple devices. `nTAP pcap` data needs to be further processed for validation. Fig. 4 shows the process.

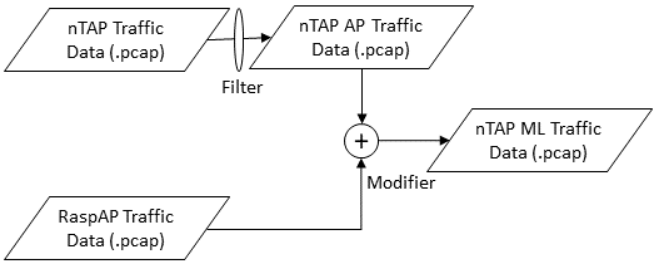

Fig. 4. nTAP data process flow.

First, a filter is applied to the `nTAP pcap` file to remove all non-RaspAP traffic. Next, the RaspAP traffic, which includes individual device IP addresses and MAC addresses, is used as a dictionary to reverse the RaspAP IP address and MAC address back to the device IP addresses and MAC addresses based on the fields in the TCP segment header including `TSval`, `Src Port`, and `Dst Port`. A new `pcap` is generated for the data labeling process, and then for the validation in Test Case #1.2.

In Test Case #2.1, an open source multi-threaded CPU miner which is used to mine Monero via the MinerGate pool is used for cryptojacking. Raspberry Pi (1) is compromised with the miner after the machine learning training and selection process. Network traffic is captured when the miner is mining.

In Test Case #2.2, `hping3` is used to create and send custom packets toward an arbitrary victim device with the intention of blocking network resources. Raspberry Pi (2) is compromised by a malicious actor. The malicious actor uses Raspberry Pi (2) to launch `DOS` attacks on K Smart Plug (2).

V. Testing and Evaluation Results

A. Data Collection

Multiple datasets were collected between March 31, 2022, and May 9, 2022. `tcpdump` was used on both the `RaspAP` and the `nTAP` data to collect network traffic. Data collected through `RaspAP` is used for ML training and data collected through `nTAP` is used to validate the ML classifier on the network perimeter. Overall, approximately 150GB and 200GB of data were collected from `RaspAP` and `nTAP`, respectively.

B. Sample Dataset

Preliminary testing and results indicated that ML was not effective in identifying a device if the device did not generate sufficient data for training. If a device did not generate a sufficient number of TCP flows, e.g., 2,000 flows, it was not considered for evaluation. After removing three unmentioned devices that did not meet this criteria, 16 devices remained in the dataset. Table IV shows the number of TCP flows for each

TABLE II
LIST OF IOT DEVICES USED IN OUR EXPERIMENTS.

Device ID	Device Name	Product Type	Device ID	Device Name	Product Type
1	Amazon Echo	Smart Speaker	9	Google Home Mini	Smart Speaker
2	Amazon Echo Dot	Smart Speaker	10	K Smart Plug (1)	Smart Plug
3	Amazon Echo Show	Smart Show	11	K Smart Plug (2)	Smart Plug
4	Amazon Fire TV Stick	Smart TV	12	Raspberry Pi (1)	Raspberry Pi
5	Lenovo Chromebook	Laptop	13	Raspberry Pi (2)	Raspberry Pi
6	Samsung Galaxy Tablet	Smart Tablet	14	ZMI Smart Clock	Smart Clock
7	Google Nexus Tablet	Smart Tablet	15	Amazon Smart Plug	Smart Plug
8	Google Home	Smart Speaker	16	Samsung Smart TV	Smart TV

TABLE III
CLASSIFIER HYPERPARAMETERS USED IN THIS WORK.

Model	Hyperparameters
AdaBoost	n_estimators = 100 base_estimator = DecisionTreeClassifier(class_weight = 'balanced')
Decision Tree	class_weight = 'balanced'
K-Nearest Neighbor	n_neighbors=5
Logistic Regression	penalty = 'l2' C = 1.0 max_iter = 1000
Random Forest	class_weight = 'balanced' n_estimators = 100
LinearSVC	penalty = 'l2' C = 1.0 max_iter = 1000

device in the dataset. The dataset is divided into two datasets, i.e., the training dataset and the testing dataset. The number of TCP flows in each dataset is shown in Table IV.

TABLE IV
DATASETS GENERATED BY THE DEVICES IN THE EXPERIMENTS.

Device	Full Dataset	Training Dataset	Testing Dataset
Amazon Echo	24,492	19,530	4,962
Amazon Echo Dot	21,002	16,748	4,254
Amazon Echo Show	47,804	38,246	9,558
Amazon Fire TV Stick	44,629	35,662	8,967
Lenovo Chromebook	9,307	7,485	1,822
Samsung Galaxy Tablet	10,218	8,221	1,997
Google Nexus Tablet	9,388	7,530	1,858
Google Home	8,722	6,931	1,791
Google Home Mini	20,358	16,333	4,025
K Smart Plug (1)	51,774	41,422	10,352
K Smart Plug (2)	27,386	21,911	5,475
Raspberry Pi (1)	4,964	3,945	1,019
Raspberry Pi (2)	14,517	11,709	2,808
ZMI Smart Clock	7,185	5,712	1,473
Amazon Smart Plug	5,227	4,157	1,070
Samsung Smart TV	94,976	76,017	18,959

C. Features

Not all features are important to characterize device behavior. The Random Forest classifier comprised of 100 estimators was chosen to rank the features due to its ability to generalize over the data. Each of these estimators, or in this case, Decision Trees, are given a random subset of the data to train. As a result, these trees hold information about different portions of the data set and can prevent overfitting. Additionally, due to the tree-like structures that comprise this model, important features can be trivially chosen by the averaged weighted decrease in node impurity from splitting. Table V shows the top 10 features from this Random Forest classifier.

TABLE V
TOP 10 FEATURES.

No.	Feature and Brief Description	Avg. Importance
1	$FwdInitWinBytes$: Number of Bytes Sent in the Initial Window in the Forward Direction	0.021
2	IAT_{thirdQ}: Inter-Packet Arrival Time Quartile 3	0.020
3	$PacketLength_{mean}$: Packet Length Average	0.017
4	IAT_{max}: Inter-Packet Arrival Time Maximum	0.016
5	$Duration$: Flow Duration	0.015
6	$FlowBytes/s$: Bytes Per Second in the Flow	0.015
7	TTL_{mean}: Time To Live Value Average	0.015
8	$IAT_{secondQ}$: Inter-Packet Arrival Time Quartile 2	0.015
9	$PacketLength_{max}$: Packet Length Maximum	0.014
10	IAT_{mean}: Inter-Packet Arrival Time Average	0.014

D. ML Classifier Training and Selection

Table VI shows the testing and training accuracies for each classifier's top n features. Based on feature ranking in Table V, we gradually increase the number of features used for ML training and testing. Due to limited space, Table VI only includes partial results. As shown in Table VI, the difference in accuracies is not substantial when comparing the usage of the top 20 features to the usage of all features. The LinearSVC classifier is an exception to this observation. The tree-based models, namely, AdaBoost, Decision Tree, and Random Forest performed the best among the six classifiers tested.

Fig. 5 shows the confusion matrix for the AdaBoost classifier. As shown in Fig. 5, the AdaBoost classifier shows 90% or higher accuracies in identifying 10 out of the 16 devices in our experiments.

We also evaluate each classifier on training balanced accuracy, testing balanced accuracy, training F1 weighted score, testing F1 weighted score, fit time, training predict time, and testing predict time in addition to training accuracy and testing accuracy. Of the six classifiers, the AdaBoost classifier was chosen for static analysis due to its high testing balanced accuracy of 89.02% while utilizing the top 200 features.

TABLE VI
TRAINING ACCURACY AND TESTING ACCURACY.

	AdaBoost		Decision Tree		KNN		Logistic Regression		Random Forest		LinearSVC	
Top n Features	Training ACC	Testing ACC	Training ACC	Testing ACC	Training ACC	Testing ACC	Training ACC	Testing ACC	Training ACC	Testing ACC	Training ACC	Testing ACC
5	95.93%	83.23%	95.93%	83.23%	87.72%	82.53%	36.43%	36.52%	95.71%	83.14%	34.15%	34.09%
10	97.18%	85.31%	97.19%	83.48%	88.96%	84.15%	51.35%	51.61%	97.01%	85.35%	48.33%	48.50%
15	97.22%	85.60%	97.19%	83.67%	89.01%	84.26%	55.30%	55.31%	96.77%	85.50%	53.03%	53.06%
20	97.22%	86.85%	97.20%	84.91%	89.13%	84.42%	57.48%	57.57%	96.88%	86.76%	55.22%	55.23%
25	97.22%	86.95%	97.20%	84.96%	89.13%	84.42%	59.83%	59.90%	96.89%	86.78%	57.75%	57.89%
30	97.23%	86.93%	97.20%	85.08%	89.15%	84.35%	61.37%	61.58%	96.82%	86.82%	59.79%	59.87%
200	97.14%	88.23%	97.22%	86.21%	89.39%	84.63%	70.59%	70.61%	96.65%	87.68%	72.24%	72.28%
220	97.15%	88.19%	97.22%	86.30%	89.25%	84.45%	70.42%	70.50%	96.62%	87.54%	72.24%	72.28%
227	97.24%	88.14%	97.22%	86.22%	89.42%	84.65%	70.63%	70.64%	96.64%	87.68%	71.32%	71.43%

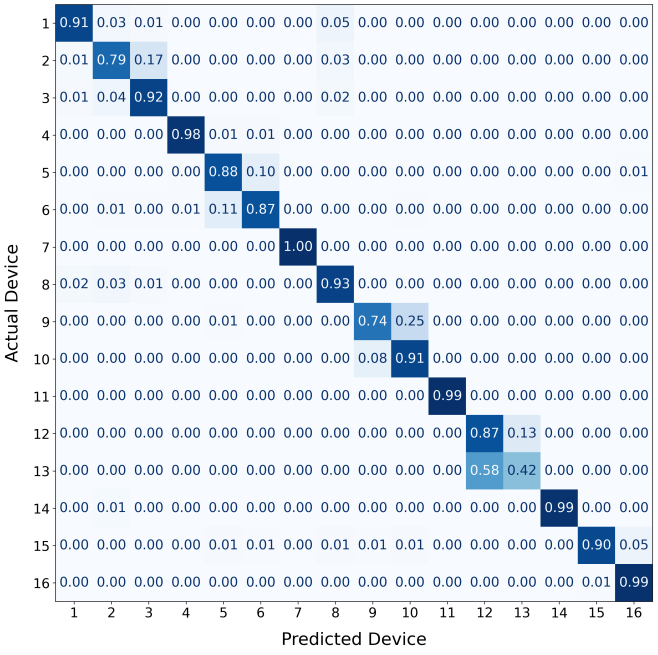

Fig. 5. Device identification confusion matrix normalized across the actual device axis: 1. Amazon Echo 2. Amazon Echo Dot 3. Amazon Echo Show 4. Amazon Fire TV Stick 5. Lenovo Chromebook 6. Samsung Galaxy Tablet 7. Google Nexus Tablet 8. Google Home 9. Google Home Mini 10. K Smart Plug (1) 11. K Smart Plug (2) 12. Raspberry Pi (1) 13. Raspberry Pi (2) 14. ZMI Smart Clock 15. Amazon Smart Plug 16. Samsung Smart TV.

E. ML for Device Identification Validation

The trained AdaBoost classifier utilizing the top 200 features was used to validate ML for device identification in both trusted and untrusted environments.

1) Testing Case #1.1: ML within the Network: We collected testing datasets from intra-network through RaspAPs. We then split them into training and testing datasets as shown in Table IV. The trained AdaBoost classifier shows 88.23% testing accuracy on the testing dataset.

2) Testing Case #1.2: ML on the Network Perimeter: Using the process shown in Fig. 4, a testing dataset (`pcap`) was derived from the `pcap` files collected from the `nTAP`. The trained AdaBoost classifier has an accuracy of 97.22% identifying devices in the derived `pcap` files.

3) Testing Case #2.1: cryptojacking: The trained AdaBoost classifier shows 90% testing accuracy in identifying Raspberry Pi (1), as shown in Fig. 5. The same classifier is used to predict the network traffic of Raspberry Pi (1) in the cryptojacking test case. Our evaluation shows 16.24% accuracy in identifying Raspberry Pi (1) after it was compromised.

4) Testing Case #2.2: DOS Attacks: The trained AdaBoost classifier shows 42% accuracy in identifying Raspberry Pi (2) and 99% in identifying K smart Plug (2), as shown in Fig. 5. In the `DOS` attack test case, Raspberry Pi (2) was used to launch the `DOS` attack, and K smart Plug (2) was the target of this `DOS` attack. Using the same AdaBoost classifier, the testing results show 16.24% accuracy in identifying Raspberry Pi (2) and 19.04% accuracy in identifying K smart Plug (2).

F. Discussions

CICFlowMeter++ includes the 14 most important features from [2]. We successfully conducted a comparison of the top features between [2] and our work. A few features including TTL_{firstQ}, TTL_{thirdQ}, TTL_{avg}, TTL_{max} are among the top 20 features in both [2] and our work. However, our study includes unique, relevant features such as $FwdInitWinBytes$, IAT_{thirdQ}, and $PacketLength_{mean}$ as shown in Table V, which have not been reported before. Additionally, the work in [2] focuses on identifying device types, whereas our work focuses on device identification. We found that certain features are important to characterize device behavior based on our results. Fig. 6 shows the ratio of `Fwd Packet Count` to `Bwd Packet Count` for each IoT device in the dataset. 83 outlier samples were removed from Fig. 6, which had a `Bwd Packet Count` or `Fwd Packet Count` of over 10,000 for viewing purposes. Fig. 6 shows an interesting pattern that might be used to distinguish IoT devices. Further study also reveals correlations among features used for device identification. For example, $IAT_{secondQ}$ and IAT_{mean} have high positive correlations with IAT_{thirdQ}, 0.85 and 0.92 correspondingly. More study is desired to examine how feature correlations affect accuracies.

VI. CONCLUSIONS AND FUTURE WORK

The ML-assisted approaches for device identification assume device behavior does not change and can be characterized through ML. However, the hypothesis that device

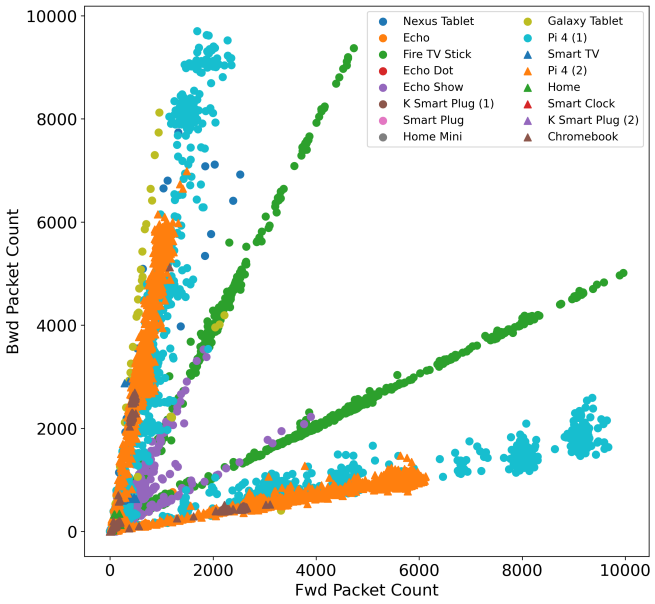

Fig. 6. The ratio of `Fwd packet count` to `Bwd packet count`.

behavior does not change and can be characterized through ML has not been thoroughly investigated. In this paper, we conducted a comprehensive study on supervised ML-assisted approaches for device identification. We tested the hypothesis in two scenarios using four testing cases. Our testing and results show that supervised ML is promising for device identification in a trusted environment. However, our results reveal that supervised ML has difficulty identifying a device correctly when it is compromised. In combination with other safeguards such as white lists and intrusion detection and prevention systems in a network, supervised ML-assisted approaches could still be practicable. However, in consideration of scalability, adaptability, and resistance, supervised ML-assisted approaches for device identification may have challenges in resolving resistance issues.

In this research, features used for supervised ML are extracted from TCP flows. However, features from other layers, e.g., the physical layer, could also be used for supervised ML [24], [25], [26]. Further, supervised ML requires a fully labeled dataset to train the models, which may be expensive to acquire. Additionally, as shown in [10], [11], [12], [13], and [27], unsupervised ML can reach accuracies as good as or better than those in supervised ML approaches while having higher accuracies in both unseen and compromised devices. We plan to expand our features to include physical layer features and integrate supervised and unsupervised ML. Even though we have focused on Wi-Fi-enabled devices IoT devices, we also plan to work with multiple protocols, including ZigBee, Z-Wave, or Bluetooth.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, *et al.*, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surv. Tut.*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] Y. Meidan, M. Bohadana, A. Shabtai, *et al.*, "ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis," in *Proc. ACM SAC*, p. 506–509, 2017.

[3] Y. Wang, B. P. Rimal, M. Elder, *et al.*, "IoT device identification using supervised machine learning," in *Proc. IEEE ICCE*, pp. 1–6, 2022.

[4] A. H. Lashkari, G. Draper-Gil, et al., "Characterization of Tor traffic using time based features," in *ICISSp*, pp. 253–262, 2017.

[5] D. Bekerman, B. Shapira, L. Rokach, *et al.*, "Unknown malware detection using network traffic classification," in *Proc. IEEE CNS*, pp. 134–142, 2015.

[6] S. A. Hamad, W. E. Zhang, Q. Z. Sheng, *et al.*, "IoT device identification via network-flow based fingerprinting and learning," in *Proc. IEEE TrustCom/BigDataSE*, pp. 103–111, 2019.

[7] D. H. Hagos, A. Yazidi, O. Kure, *et al.*, "A machine-learning-based tool for passive os fingerprinting with tcp variant as a novel feature," *IEEE Internet Things J.*, vol. 8, 2021.

[8] A. Aksoy and M. H. Gunes, "Automated IoT device identification using network traffic," in *Proc. IEEE ICC*, pp. 1–7, 2019.

[9] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *Proc. IEEE LCN*, pp. 250–257, 2005.

[10] A. Sivanathan, H. H. Gharakheili, and V. Sivaraman, "Inferring IoT device types from network behavior using unsupervised clustering," in *Proc. IEEE LCN*, pp. 230–233, 2019.

[11] S. Marchal, M. Miettinen, T. D. Nguyen, *et al.*, "AuDI: Toward autonomous IoT device-type identification using periodic communication," *IEEE J. Sel. Areas Commu.*, vol. 37, no. 6, pp. 1402–1412, 2019.

[12] R. Bhatia, S. Benno, J. Esteban, *et al.*, "Unsupervised machine learning for network-centric anomaly detection in IoT," in *Proc. ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, p. 42–48, 2019.

[13] S. Zhang, Z. Wang, J. Yang, *et al.*, "Unsupervised IoT fingerprinting method via variational auto-encoder and k-means," in *Proc. IEEE ICC*, pp. 1–6, 2021.

[14] RaspAP, "RaspAP: Simple wireless AP setup & management for Debian-based devices," *GitHub*, 2022.

[15] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[16] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, *et al.*, "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.

[17] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *Int. Statistical Review/Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[18] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[19] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, *et al.*, *Logistic regression*. Springer, 2002.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[23] T. Kluyver, B. Ragan-Kelley, et al., "Jupyter notebooks - a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (F. Loizides and B. Scmidt, eds.), pp. 87–90, IOS Press, 2016.

[24] H. Jafari, O. Omotere, D. Adesina, *et al.*, "IoT devices fingerprinting using deep learning," in *Proc. IEEE MILCOM*, pp. 1–9, 2018.

[25] D. Nouichi, M. Abdelsalam, Q. Nasir, *et al.*, "IoT devices security using rf fingerprinting," in *Advances in Science and Engg. Tech. Int. Conf. (ASET)*, pp. 1–7, 2019.

[26] Y. Tu, Z. Zhang, Y. Li, *et al.*, "Research on the Internet of Things device recognition based on RF-fingerprinting," *IEEE Access*, vol. 7, pp. 37426–37431, 2019.

[27] X. Liu, M. Abdelhakim, P. Krishnamurthy, *et al.*, "Identifying malicious nodes in multihop IoT networks using diversity and unsupervised learning," in *Proc. IEEE ICC*, pp. 1–6, 2018.