IoTScent: Enhancing Forensic Capabilities in Internet of Things Gateways

Antonio Boiano
DEIB, Politecnico di Milano
Milan, Italy
antonio.boiano@polimi.it

Alessandro Enrico Cesare Redondi
DEIB, Politecnico di Milano
Milan, Italy
alessandroenrico.redondi@polimi.it

Matteo Cesana
DEIB, Politecnico di Milano
Milan, Italy
matteo.cesana@polimi.it

Abstract—The widespread deployment of Consumer Internet of Things devices in proximity to human activities makes them digital observers of our daily actions. This has led to a new field of digital forensics, known as IoT Forensics, where digital traces generated by IoT devices can serve as key evidence for forensic investigations. Thus, there is a need to develop tools that can efficiently acquire and store network traces from IoT ecosystems. This paper presents IoTScent, an open-source IoT forensic tool that enables IoT gateways and Home Automation platforms to perform IoT traffic capture and analysis. Unlike other works focusing on IP-based protocols, IoTScent is specifically designed to operate over IEEE 802.15.4-based traffic, which is the basis for many IoT-specific protocols such as Zigbee, 6LoWPAN and Thread. IoTScent offers live traffic capture and feature extraction capabilities, providing a framework for forensic data collection that simplifies the task of setting up a data collection pipeline, automating the data collection process, and providing ready-made features that can be used for forensic evidence extraction. This work provides a comprehensive description of the IoTScent tool, including a practical use case that demonstrates the use of the tool to perform device identification from Zigbee traffic. The study presented here significantly contributes to the ongoing research in IoT Forensics by addressing the challenges faced in the field and publicly releasing the IoTScent tool.

Index Terms—IoT Forensics, IEEE 802.15.4, LR-WPANs, Home Assistant, Smart Home

I. INTRODUCTION

Considering the Internet of Things (IoT) ecosystem, Consumer IoT (CIoT) devices play a significant role, as they are becoming increasingly prevalent in our society. According to [1], it is projected that CIoT devices will have the largest share, accounting for approximately 50% of the entire IoT device market. This is also demonstrated by the investment made by the Connectivity Standard Alliance, composed of major CIoT hardware and software manufacturers in the smart home field, e.g., Apple, Amazon, and Google, into developing new protocols such as Thread and Matter. The widespread deployment of CIoT devices in homes, workplaces, and buildings where people spend most of their time makes them natural witnesses of daily human activities. Indeed, CIoT devices are designed to interact and support human actions (e.g., controlling lights/doors and electronic equipment); it follows that the behaviour of CIoT devices is strictly connected with human daily lives and can be monitored and analyzed to perform tasks such as human activity recognition. Recently, IoT forensics has emerged as a new research area focusing on identifying and extracting digital information from IoT devices. In particular, the network traffic exchanged by such devices may be conveniently captured, stored and analyzed to provide evidence that can be used in a court of justice [2]. Moreover, monitoring and analysing the network traffic produced by CIoT devices is key for protecting user's security and privacy: the generally poor security design of CIoT devices, due to manufacturers' priorities on cost reduction and time to market, makes them prone to a vast range of cyberattacks [3]. In general, enabling a-posteriori network traffic analysis in CIoT scenario is a challenging task due to the fragmentation inherent in IoT communication technologies (edge/cloud), the transient and ephemeral nature of the traffic generated by IoT devices, and the limited memory size of these devices.

All these observations call for tools able to identify, collect and preserve IoT network traffic to support IoT forensic analyses. Most works dealing with IoT traffic collection and analyses focus on traffic traces at the IP layer or leverage Wi-Fi traffic [4] [5]. Although many currently available CIoT devices are Wi-Fi-based, an increasing number of devices use other communication technologies based on low-power, short-range standards such as IEEE 802.15.1 (Bluetooth) or IEEE 802.15.4 (Zigbee/Thread). It is crucial to develop tools and methodologies to capture and analyse such traffic efficiently. Moreover, networks based on such standards generally rely on the presence of specific hub devices which aggregate traffic and bridge it to the Internet using WiFi/Ethernet connectivity. Examples include popular proprietary solutions such as Philips Hue Bridge and open-source platforms such as Home Assistant[1]. These intermediate points may shadow important details in the network traffic exchanged locally, calling for solutions which can monitor traffic closer to the generating devices.

This paper presents *IoTScent* (IoTS), an IoT forensic tool which enables forensic analysis on IEEE 802.15.4 networks controlled by the open-source Home Assistant platform. IoTS enables Home Assistant to perform several tasks related to IoT forensics, such as traffic capture and feature extraction, providing the basis for network monitoring and analysis tasks. The architecture and implementation of IoTS are presented, highlighting its features through a practical use case relative to IoT device identification.

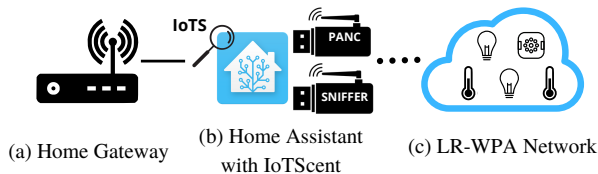[1]https://github.com/home-assistant

Fig. 1: IoTScent Application Setup

The subsequent sections of this paper are structured as follows: Section II offers a presentation of the primarily related studies in IoT forensics. Section III presents an accurate description of the IoTScent tool, describing its architecture, capabilities, and performance. Section IV presents a practical use-case where Device Identification is performed over features extracted by IoTScent. The study concludes in Section V, which provides final reflections and highlights directions for future research.

II. RELATED WORK

Over the past few years, there has been a surge of interest in CIoT, leading to the emergence of a subfield within Digital Forensics, namely IoT Forensics. Numerous studies have been conducted in this field with the aim of defining IoT Forensics enabler frameworks. Cicirelli et al., [6] proposed a framework for developing a human activity recognition system in a smart home environment. Zawoad et al., [7] instead propose a theoretical framework for IoT Device Forensics, IoT Network Forensics, and IoT Cloud Forensics. However, there has been limited research into developing and defining practical frameworks for IoT forensic data collection similar to the one presented in this paper. It is noteworthy that Meffert et al., [8] proposed a centralized Forensic State Acquisition Controller framework, which simultaneously performs state collection and control of IoT devices. Unlike the solution presented in this article, the framework does not directly monitor encrypted traffic for state collection but rather records logs received directly from IoT devices and end-user interaction. On the other hand, Palmese et al., [4] proposed and developed Feature-Sniffer, a user-friendly tool that computes network traffic features on OpenWrt-based Wi-Fi access points. The work shares similarities in technique and collected features compared with the tool proposed in this paper. However, while Wi-Fi-based traffic can provide valuable insights into network usage and behaviour, Feature-Sniffer does not include monitoring and analysing traffic from 802.15.4 networks which may limit its ability to detect certain types of devices or behaviours.

III. TOOL OVERVIEW

IoTScent[2] is an open-source tool, publicly available on GitHub, designed for forensic analysis in networks operated by the IEEE 802.15.4 standard. The tool is implemented as a custom integration to the Home Assistant OS, which can be operated on low-cost general-purpose hardware such as the Raspberry PI. IoTScent eases the tedious task of setting

[2]https://github.com/antonio-boiano/IoTScent/

up and automating the general IoT forensic data collection pipeline and provides ready-to-use functionalities for capturing network traffic and extracting its characteristics in a few steps. Traffic capture is performed through the availability of a dedicated radio interface compatible with the Killerbee Software suite,which was integrated into the Home Assistant framework. Compatible solutions include, e.g., Crossbow Telosb, Texas Instruments CC2530/1, and Silicon Labs EFR32 family SoC. A lightweight dissector is implemented to convert raw data streams into formatted MAC and Zigbee Network packet headers.

In a nutshell, IoTScent provides the following functionalities:

- *Packet capture:* IoTS allows for TCPdump-like traffic capture over the 802.15.4 physical layer, saving raw packet data in a PCAP file format. Traditional packet filters can be used to filter incoming packets (e.g., source/destination addresses, packet type, PAN address, etc.), enabling processing only a subset of sniffed packets.
- *Time-windowed feature extraction:* Most IoT forensic analysis tasks are based on statistical features extracted from network traffic. IoTS allows easily extracting network features in a time-windowed fashion from 802.15.4 traffic. In detail, IoTS organizes the received traffic in time windows of user-defined duration and computes a set of traffic features for each time window and for each device, identified by its source address present in the 802.15.4 MAC header. The output of the feature extraction processes is saved in a CSV file and stored inside the device for easy management and post-processing of the retrieved information. To provide the network administrator with informative data that can be used for IoT forensics analysis, the IoTS feature extraction capabilities have been refined through a systematic review of the most commonly used network features for device identification, classification, and human activity recognition in low-rate wireless personal area (LR-WPA) networks [9], [10], [11]. IoTS employs three primary feature components: Packet Size, Payload Length, and Inter-arrival Time (IAT) to conduct statistical analysis over user-defined time windows. This analysis involves extracting the Mean and Standard Deviation values from the aforementioned features. To improve the computational performance, the mean and the standard deviation are computed online using Welford's formulation [12]. Furthermore, IoTS allows users to differentiate incoming and outgoing packets and perform separate or combined statistics for each direction. Finally, IoTS also permits extracting features from an existing PCAP file containing 802.15.4 raw packets, allowing offline traffic analysis.

A. IoTScent Overview

1) Hardware setup: IoTS can be seamlessly deployed on both IoT gateways or other devices running the Home Automation platform. Such devices generally have no energy or performance constraints and are well-suited for executing
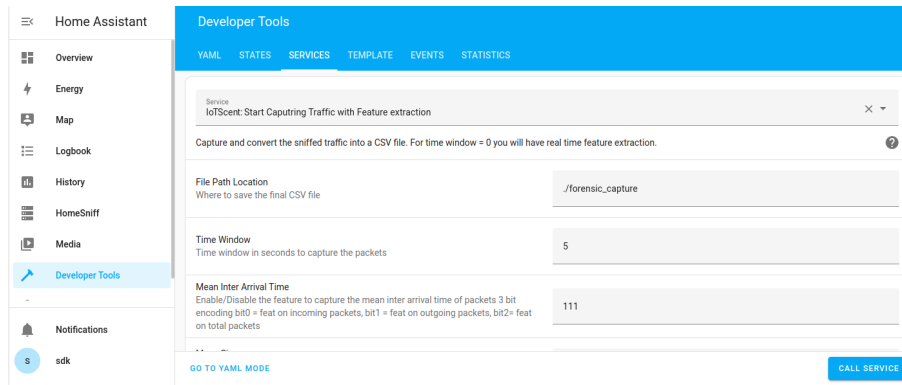
Fig. 2: IoTScent feature extraction as a service call, Home Assistant User Interface

network traffic analysis and feature extraction operations. A standard implementation setup for IoTS is shown in Figure 1. In this configuration, a Raspberry Pi is utilized as an IoT gateway, running the Home Assistant platform, and an IEEE 802.15.4 antenna (SONOFF USB Dongle based on EFR32MG21 SoC) is used as the main Personal Area Network (PAN) Coordinator (PANC) hardware. IoTS is installed as an integration on the Home Assistant platform and requires little configuration (PANC channel) and a compatible IEEE 802.15.4 antenna. The tool was tested with a USB dongle based on the CC2531 System on Chip (SoC), used to perform packet sniffing in promiscuous mode.

2) Software setup: The IoTS controlling logic comprises two primary components: the first is responsible for accessing hardware resources, performing traffic capture and distributing the captured raw packets among asynchronous queues, whereas the second component is dedicated to performing analysis and feature extraction from the packet queues. By separating these tasks, IoTS can avoid resource conflict while efficiently handling the data produced by IoT networks. The two components are detailed in the following:

i) *Packet capture:* Upon IoTS startup, a single task is initiated to manage the 802.15.4 SoC hardware component. This design choice enables the necessary hardware configurations and capability checks to be performed and prevents concurrent task operations, which could result in deadlocks. Multiple asynchronous queues with limited buffers can be defined, allowing multiple capture tasks to run in parallel and limiting IoTS's impact on computational resources. This prevents packet loss in packet burst scenarios and enables each feature extraction task to be handled independently. Limiting the queue size is a prevention mechanism in case the hardware on which IoTS is installed cannot handle the packet rate produced by the observed network. To limit the impacts of IoTS on normal IoT gateway execution, different mechanics are implemented. IoTS can be configured to stop the feature extraction process or limit the packet rate (with packet loss) when the queue reaches its maximum capacity. Additionally, prior to the packet distribution task, this layer is also responsible for packet filtering based on the rules defined by each acquisition process. The syntax used by IoTS for specifying packet filters is based on the one used by Wireshark for Zigbee and 802.15.4 packets.

ii) *Traffic Feature extraction:* The second IoTS component is dedicated to processing the raw packet data received from the first component. Upon launching a new acquisition task, an asynchronous queue is created, which is then passed to the first component to populate it with raw packet data. In addition, it extracts only the user-selected relevant features from the raw packet data to optimise CPU and RAM consumption. These features are then used to calculate the user-requested features and are saved in CSV for the live feature extraction process. Alternatively, the extracted features are saved to a second queue, which will be processed depending on the time windows to extract the requested features per device.

3) User Interface: The IoTS capabilities have been exposed as the so-called Home Assistant Services, which allow the network administrator to interact with IoTS in a wide range of scenarios easily; indeed, service calls can be directly called from Home Assistant's user interface (UI) under the Developer Tools menu, or triggered by Home Assistant Automation script, which allows scheduling or triggering the feature extraction processes. With this approach, network administrators can easily interact with IoTS functionality. System administrators opting for the IoTScent services will encounter five distinct service implementations designed to handle the IoTScent execution tasks: (i) start and configure the Features extraction task, (ii) TCPdump-like traffic capture, (iii) termination of an acquisition task, (iv) retrieval of its execution status, and (v) removal of an acquisition task from memory. Figure 2 illustrates a UI section related to the IoTS feature extraction service call. On this page, users can specify the output file's destination path, time window duration, and the set of features they want IoTS to compute.

B. Performance Evaluation

Tests were conducted on real hardware devices that can be used in the field to validate the tool's performance. The test setup involved installing the IoTS tool as an integration on the

TABLE I: CIoT devices utilized as a testbed in the study

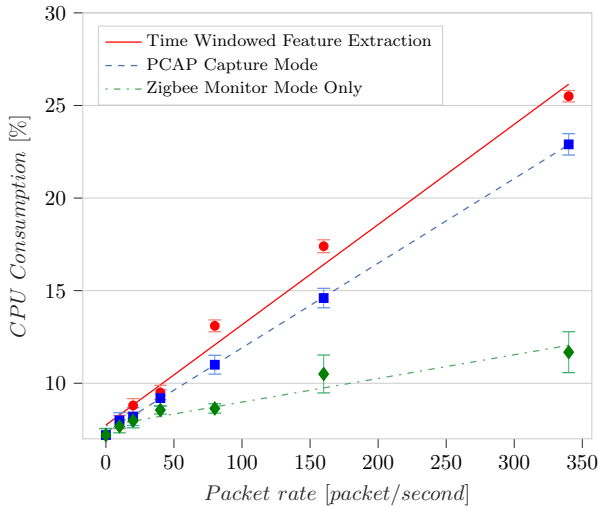| ID	Device Model	Brand	Actions
(a)	Hue Motion Sensor	Philips	Motion Light Intensity
(b)	TS011F	Tuya	On & Off Power Consumption
(c)	Plug Z3	Ledvance	On & Off Power Consumption
(d)	Hue White Lamp	Philips	On & Off Luminosity
(e)	Door Window Sensor	Aqara	Open & Closed
(f)	ZBSA-Motion Sensor	Woolley	Motion
(g)	TS0043 Switch	Tuya	Short press Long press Double press

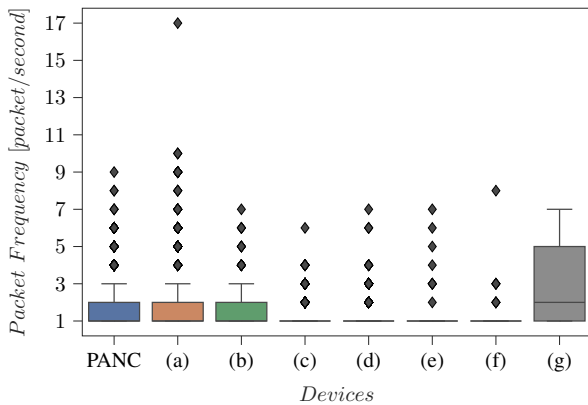Fig. 3: CPU usage of HomeAssistant running *IoTScent* under different packet rates and execution behaviors

Fig. 4: Average Packet Distribution per device (Table I) in the unit of time, considering only active 1-second windows.

Home Assistant OS, which was installed on a Raspberry PI 3 Model B with a 1.2 GHz CPU and 1 GB of RAM. A dongle based on the CC2531, a USB-enabled SoC solution for 2.4 GHz IEEE 802.15.4, was used as a radio interface. The dongle was installed with a firmware version developed by Texas Instruments, which enables the Radio Frequency Monitor mode of the CC2531 over serial communication. A Zigbee network was established using an EFR32MG21-based USB dongle as the PANC and several commercially available CIoT devices listed in Table I. Different experiments were performed by rerunning traffic capture and feature extraction to evaluate the tool's performance under different stress conditions. Figure 3 illustrates the CPU consumption as a function of the packet rate when performing: (i) 5-second time windowed feature extraction with all features selected, (ii) Standard PCAP capture, (iii) Traffic Capture without further processing the packets, to test the acquisition library performance. The system performance was acquired using a bash script, which stores the output of the Linux TOP command as a CSV file. A Crossbow TelosB (TPR2400) mote was used with custom firmware designed to generate 20-byte length IEEE 802.15.4 packets with different destination addresses at a controlled rate to generate additional traffic for IoTS to process, allowing for IoTS tool's performance evaluation under stress condition. To establish a correlation between the performance of the IoTS tool and a real-life scenario, an initial examination of CIoT's traffic consumption was conducted. Figure 4 illustrates the packet distribution per device when in an active state. Specifically, the analysis focuses on devices that actively transmit packets over the network at each time interval. The plot shows that each device has a median of 1 packet per second and many expected outliers, representing the packets exchanged during human-device interaction. The packet rate produced in an LR-WPAN is a function of the number of devices and the number of human-device interactions in the unit of time, which can be considered a sparse event. Experimentally, averaging the number of packets acquired over the entire acquisition time, a packet transmission rate of 4 packets per second was observed. Figure 3 shows a linear dependency between CPU consumption and Packet rate. With a Packet rate of 40 packets per second (one order of magnitude bigger than the average packet rate generated by eight devices), the CPU consumption presents only an increase of 2.3 % for the Time Windowed Feature Extraction process. No significant changes were noticed concerning RAM consumption. When comparing the performance metrics of our tool against those of FeatureSniffer [4], our tool's performance exhibits a performance deficit of fivefold. However, some considerations are needed: Firstly, the two tools are designed to analyze different types of network traffic, leading to different features and packet rates. Secondly, IoTS uses different network hardware interfaces that are unoptimized on both the library and hardware side for network monitoring. Thirdly, IoTS is written in Python, which is slower than unoptimized C by a factor of 45; the choice of programming language was motivated by the integration requirements with the HomeAssistantOS platform, which is

TABLE II: Features importance ranking from forest of trees classification algorithm

Features	Importance Score
Mean Inter-arrival Time	0.14
Mean Outgoing Packet Length	0.10
Mean Incoming Packet Length	0.09
Mean Incoming Inter-arrival Time	0.08
Mean Packet Length	0.07
Mean Outgoing Packet Payload Length	0.06

fully written in Python.

IV. USE CASE

To demonstrate the insights introduced by IoTS in IoT forensics analyses, this chapter showcases the ability to perform device identification based on IoT traffic features extracted by IoTS. Device identification refers to the ability to recognize a device (brand and device type) by analyzing only some features produced by network traffic. Device identification is a primary and fundamental task in many forensic analyses, as it allows investigators to differentiate between traces left by different devices, which is essential to attribute a particular action or activity to a specific device or user since CIoT devices have a limited range of possible actions. Although previous works have already showcased the Device Identification tasks on Zigbee Network [9], [11], the aims of this work are: (i) Showcase the IoTS tool to extract features from a Zigbee Network and use them directly for the Device Identification task, (ii) Identifying and minimizing the number of features essential for the Device Identification task, (iii) Compare multiple Machine Learning (ML) classification algorithms' performance.

A. Methods

The data collection phase involved the Zigbee network previously described in Section III-B and the devices in Table I. The acquisition process was performed by IoTS using a 5-second time window, selecting all the features described in Section III. The traffic was recorded for the duration of five hours under conditions that simulate the typical daily usage of CIoT devices. This was accomplished by simulating the user's interaction with the various IoT devices, including turning the smart bulb on and off, changing its brightness, activating the motion and door sensors, turning the smart outlets on and off, interacting with the wall switch, where for each button pressure was assigned a set of device commands (i.e. toggle on and off the light and the smart outlet) as it would be in a smart home environment. Before starting to analyze the produced dataset[4], data cleaning and feature selection were performed. This step is fundamental since it improves model performance, provides better model explainability, and reduces the dataset dimensionality, translating into lighter and less complex models. As a result, a dataset containing 24,000 entries was obtained after data cleaning. The Feature Importance was computed for the entire acquired dataset using

[4]https://github.com/antonio-boiano/IoTScent/tree/main/Dataset
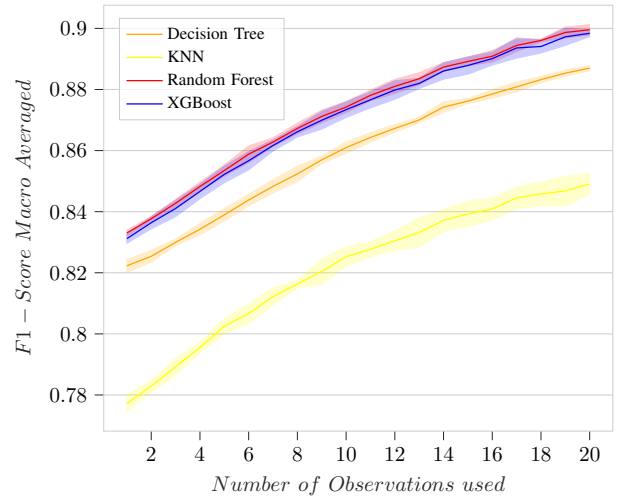

Fig. 5: F1-score evaluation with varying number of observations

a Random Forest (RF) Classifier. Forest of trees algorithm works by constructing an ensemble of decision trees, where each tree is trained on a different subset of data. Therefore, it is possible to compute the feature importance based on how much each feature contributes to the overall performance of the ensemble. In our analysis, only features with an importance score greater than 0.06 were deemed suitable and selected for the analysis step. The complete list of features used for the classification algorithms analysis is reported in Table II.

The performance of popular ML models widely used in literature to solve device identification was evaluated. Among all the models used for the device identification task, we considered mainly decision tree-based ML algorithms since the dataset presents an unbalanced data distribution, with some devices performing periodic transmissions while others producing traffic only when triggered. We evaluated: Decision Tree, XGBoost, Random Forest classifiers, and K-nearest neighbours. The ML algorithms were trained on the features extracted by IoTS described in Table II. To obtain more reliable and unbiased estimates of model performance, 10-fold cross-validation was employed, which involves dividing the data into 10 non-overlapping folds, using 1 fold for testing and the remaining 9 for training. For evaluating the classification model's performance, the F1 score was used to address unbalanced data distribution. To synthesize the result among the different classes, the F1 Score macro average is computed, where the Per-Device F1 Score is averaged by performing the arithmetic mean. By doing so, all classes are treated equally regardless of their support values. Moreover, to improve model performance, multiple time window samples can be grouped together based on the device address and the probability of the estimations, averaged for each class label.

B. Results

The feature importance described in Table II identifies the *mean IAT* feature as the one with the highest normalized

feature importance score (0.14). This is expected since, as described by [11], the features extracted from IAT alone can successfully be used for accurate device identification. This is mainly due to the fact that each device presents a unique IAT fingerprint. Figure 5 depicts the macro averaged F1 Score of different ML Classification models in relation to the number of time windows considered. Combining the observations shows an increase in model performance for all the classification algorithms under analysis. Among the analysed models, the ones reporting the highest F1 score are RF, with an F1 score of 0.899 and XGBoost, with an F1 score of 0.898. The RF model performance per device are shown in Table III The model

TABLE III: Precision, Recall, and F1-Score for each Device from the RF model with 20 observations combined

Device	Precision	Recall	F1-Score
PANC	0.96	0.91	0.93
(a)	0.99	0.87	0.93
(b)	0.92	0.87	0.90
(c)	0.97	0.75	0.85
(d)	0.90	0.76	0.83
(e)	0.96	0.94	0.95
(f)	0.96	1.00	0.98
(g)	0.66	0.87	0.75
Unlabelled	0.99	1.00	1.00
Accuracy			0.90
Macro Avg.	0.92	0.89	0.90
Weighted Avg.	0.91	0.90	0.90

performance achieved with our testbed coincides with the ones achieved in literature [11], [9]. However, different from these works, the analysis focused on identifying a small subset of features that can be computed online with high efficiency without analysing the encrypted traffic. This could potentially yield advantages in ML model size and complexity, data storage requirements, and CPU computational consumption. Moreover, by combining multiple observations, the model performance has been increased by 8%. Nevertheless, the proposed analysis presents some limitations and concerns:

1) It is worth mentioning that IoTS alone has no PANC capabilities and can perform packet sniffing in monitor mode (capture all wireless packets within its range, regardless of the target device's address and network). This means that anyone in reach of the PAN can perform the same analysis as described in this chapter. A solution to this privacy leak was proposed by [9], where spoofed traffic was generated to degrade the ML model's performance. This mechanism can be integrated into IoTS for enhanced user privacy at the IoT gateway.

2) The device identification task has undergone training and testing using the same PAN (same set of devices and PANC). Hence, it is crucial to confirm that the trained models can generalize under different PANs with similar device sets but managed by different PANCs.

V. CONCLUSIONS

This paper proposed IoTScent, an IoT forensic tool for capturing and analysing traffic from IEEE802.15.4-based networks. The integration of IoTS into the popular Home As-

sistant platform was demonstrated, presenting the tool's performance, and providing a real use-case scenario targeting device identification. Future research directions will target the improvement of IoTS computational performance and architecture generalization of the tool to extend the support to the wider range of IoT communication protocols. Data compression algorithms can also be introduced to reduce the space needed to store network traffic features for long-term periods. Furthermore, federated learning-based solutions could be implemented on IoTS to train machine learning models in a distributed manner. This approach could lead to more accurate classification models for forensic analyses resolving the privacy concerns related to centralized training.

REFERENCES

[1] U. Cisco, "Cisco annual internet report (2018–2023) white paper," *Cisco: San Jose, CA, USA*, vol. 10, no. 1, pp. 1–35, 2020.

[2] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A Survey on the Internet of Things (IoT) Forensics: Challenges, Approaches, and Open Issues," *IEEE Comm. Surveys & Tutorials*, vol. 22, no. 2, pp. 1191–1221, 2020.

[3] T. Alladi, V. Chamola, B. Sikdar, and K.-K. R. Choo, "Consumer IoT: Security Vulnerability Case Studies and Solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 2, pp. 17–25, 2020.

[4] F. Palmese, A. E. C. Redondi, and M. Cesana, "Feature-sniffer: Enabling iot forensics in openwrt based wi-fi access points," in *2022 IEEE 8th World Forum on Internet of Things (WF-IoT)*, 2022, pp. 1–6.

[5] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach," in *Proc. of the Internet Measurement Conf.*, ser. IMC '19. New York, NY, USA: ACM, 2019, p. 267–279.

[6] F. Cicirelli, G. Fortino, A. Giordano, A. Guerrieri, G. Spezzano, and A. Vinci, "On the Design of Smart Homes: A Framework for Activity Recognition in Home Environment," *J. Med. Syst.*, vol. 40, no. 9, p. 1–17, sep 2016.

[7] S. Zawoad and R. Hasan, "FAIoT: Towards Building a Forensics Aware Eco System for the Internet of Things," in *2015 IEEE Intl. Conf. on Services Computing*, 2015, pp. 279–284.

[8] C. Meffert, D. Clark, I. Baggili, and F. Breitinger, "Forensic State Acquisition from Internet of Things (FSAIoT): A General Framework and Practical Approach for IoT Forensics through IoT Device State Acquisition," in *Proc. of the 12th Intl. Conf. on Availability, Reliability and Security*, ser. ARES '17. New York, NY, USA: ACM, 2017.

[9] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-Boo: I See Your Smart Home Activities, Even Encrypted!" in *Proc. of the 13th ACM Conf. on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '20. New York, NY, USA: ACM, 2020, p. 207–218.

[10] Y. Wan, K. Xu, F. Wang, and G. Xue, "Iotmosaic: Inferring user activities from iot network traffic in smart homes," in *IEEE INFOCOM 2022 - IEEE Conf. on Computer Comm.*, 2022, pp. 370–379.

[11] L. Babun, H. Aksu, L. Ryan, K. Akkaya, E. S. Bentley, and A. S. Uluagac, "Z-iot: Passive device-class fingerprinting of zigbee and z-wave iot devices," in *ICC 2020 - 2020 IEEE Intl. Conf. on Comm. (ICC)*, 2020, pp. 1–7.

[12] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.