IoTBeholder: A Privacy Snooping Attack on User Habitual Behaviors from Smart Home Wi-Fi Traffic

QINGSONG ZOU, Tsinghua Shenzhen International Graduate School, China and Peng Cheng Laboratory, China
QING LI*, Peng Cheng Laboratory, China
RUOYU LI, Tsinghua University, China and Peng Cheng Laboratory, China
YUCHENG HUANG, Tsinghua University, China and Peng Cheng Laboratory, China
GARETH TYSON, Hong Kong University of Science and Technology (GZ), China
JINGYU XIAO, Tsinghua Shenzhen International Graduate School, China and Peng Cheng Laboratory, China
YONG JIANG, Tsinghua Shenzhen International Graduate School, China and Peng Cheng Laboratory, China

With the deployment of a growing number of smart home IoT devices, privacy leakage has become a growing concern. Prior work on privacy-invasive device localization, classification, and activity identification have proven the existence of various privacy leakage risks in smart home environments. However, they only demonstrate limited threats in real world due to many impractical assumptions, such as having privileged access to the user's home network. In this paper, we identify a new end-to-end attack surface using IoTBeholder, a system that performs device localization, classification, and user activity identification. IoTBeholder can be easily run and replicated on commercial off-the-shelf (COTS) devices such as mobile phones or personal computers, enabling attackers to infer user's habitual behaviors from smart home Wi-Fi traffic alone. We set up a testbed with 23 IoT devices for evaluation in the real world. The result shows that IoTBeholder has good device classification and device activity identification performance. In addition, IoTBeholder can infer the users' habitual behaviors and automation rules with high accuracy and interpretability. It can even accurately predict the users' future actions, highlighting a significant threat to user privacy that IoT vendors and users should highly concern.

CCS Concepts: • **Security and privacy → Human and societal aspects of security and privacy**.

Additional Key Words and Phrases: Smart Home; Internet-of-Things; Privacy

ACM Reference Format:
Qingsong Zou, Qing Li, Ruoyu Li, Yucheng Huang, Gareth Tyson, Jingyu Xiao, and Yong Jiang. 2023. IoTBeholder: A Privacy Snooping Attack on User Habitual Behaviors from Smart Home Wi-Fi Traffic. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 1, Article 43 (March 2023), 26 pages. https://doi.org/10.1145/3580890

*Corresponding Author: Qing Li.

Authors' addresses: Qingsong Zou, Tsinghua Shenzhen International Graduate School, Shenzhen, China and Peng Cheng Laboratory, Shenzhen, China, zouqs21@mails.tsinghua.edu.cn; Qing Li, Peng Cheng Laboratory, Shenzhen, China, andyliqing@gmail.com; Ruoyu Li, Tsinghua University, Shenzhen, China and Peng Cheng Laboratory, Shenzhen, China, liry19@mails.tsinghua.edu.cn; Yucheng Huang, Tsinghua University, Shenzhen, China and Peng Cheng Laboratory, Shenzhen, China, huangyc20@mails.tsinghua.edu.cn; Gareth Tyson, Hong Kong University of Science and Technology (GZ), Guangzhou, China, gareth.tyson@qmul.ac.uk; Jingyu Xiao, Tsinghua Shenzhen International Graduate School, Shenzhen, China and Peng Cheng Laboratory, Shenzhen, China, jy-xiao21@mails.tsinghua.edu.cn; Yong Jiang, Tsinghua Shenzhen International Graduate School, Shenzhen, China and Peng Cheng Laboratory, Shenzhen, China, jiangy@sz.tsinghua.edu.cn.

2474-9567/2023/3-ART43 $15.00
https://doi.org/10.1145/3580890

1 INTRODUCTION

With the development of the Internet of Things (IoT), an increasing number of smart devices have been deployed in homes (expected to reach 5 billion devices by 2025 [39]). Among them, WiFi-connected devices represent a significant fraction of the market [13]. These devices act as sensors and actuators for various household activities [28, 41], and can usually be operated via a mobile app provided by the vendor. The emergence of automation platforms like Mijia [1], SmartThings [2] and HomeKit [3] provides further convenience for users to integrate and control their smart home devices. Specifically, users can customize the interaction rules on the platform according to their own habits.

Unfortunately, this flexibility comes with privacy risks. Most notably, prior research has revealed the ability to identify the type of devices (and their activities) by passively sniffing network traffic between devices and the cloud [40, 42, 58, 64]. This means that an attacker can infer the type and activity of the device (and by proxy the user's behavior at home) without requiring user account permissions. Due to the regularity of users' daily life at home [25, 45], the leakage of these behaviors can provide attackers with a large amount of information, causing serious privacy leakage. For example, once the user-defined rules on the automation platform are exposed to the attacker, it will be possible for event spoofing attacks [15, 18, 26]. The leakage of users' behavioral habits also raises the risk of physical attacks such as burglary, e.g., if an attacker can predict when a user will be outside of the house. Despite this, most existing passive snooping attacks are still far from practical. For example, [24, 40] requires information at the IP layer and above. However, 97% of users already use encryption methods such as WPA/WPA2 [61]. Therefore, it is difficult for an attacker to access such information. Other works analyze common events in smart homes [5, 57], but require prior knowledge of devices and user habits. An alternative strategy is to pre-deploy snooping devices in the user's home [16, 33], yet this is difficult without physical access to the user's residence.

To overcome these challenges, several other works have (theoretically) analyzed the possibility of leaking user behavior habits from encrypted traffic [23, 52, 58]. These attempt to mine patterns from flows, without needing to monitor payloads. Unfortunately, however, these works only focus on each single user action, thus ignoring the correlation between user behaviors over time (referred to as habitual behavior). For example, a user may adjust the temperature of the air conditioner and turn off the lights before going to bed, an attacker can infer when a user is ready to go to sleep from when a bedside light is turned off, but may not be able to obtain a user's habitual behavior such as adjusting air conditioner before going to bed. It should be pointed out that the user's habitual behavior (i.e., a stable pattern of a device activity) may not be obvious. For example, suppose a user usually turns on their bed light after turning off the desk light when about to sleep. Initially, one might assume these could be used to identify the user's habitual behavior. However, these two actions may not always be adjacent because the user may sometimes perform other actions between them, e.g., going to the toilet, turning on the heater, etc. Thus, the sequence and timing of actions may vary on a daily basis. Besides, there may be multiple users operating the devices at the same time. This means that the attacker must spend a great deal of effort before it is possible to infer user behavior for their IoT traffic.

Whereas the above focuses on attackers with special permissions and financial resources, we investigate the case of an attacker with limited technical capabilities and cost. For example, the attacker may not have any professional equipment, network credentials or physical access to the user's home. This paper presents IoTBeholder, a practical and low-cost snooping attack that can infer user habitual behavior from smart device traffic. As a tool, it can easily be used and spread by an attacker. It only requires the attacker to walk around the target's home with a commercial off the shelf (COTS) device, and then deploy the device in a hidden location, after which IoTBeholder can undermine the privacy of smart home users' daily behavior habits. Compared with (pre-)deploying snooping devices (e.g., cameras), IoTBeholder attack has stronger concealment. Audio or video capable devices need to be deployed near windows or even in the users' homes where they can be easily

discovered. In contrast, attackers only need to deploy IoTBeholder near the home. For example, the attacker could easily deploy IoTBeholder on a Raspberry Pi and place it in any place suitable for hiding near the user's home. In addition, as we later show in 9, IoTBeholder can more effectively leak the privacy of users' behavior habits without a lot of labor cost. Our ultimate goal is to highlight the privacy threats introduced by smart home devices. Specifically, IoTBeholder can:

- Obtain the approximate location, type, and ongoing activity of IoT devices deployed in the user's home *without physical and network access.*
- Perceive the user's potential habitual behaviors in daily life, such as when they finish showering, alongside the automation rules configured by the user *without any prior knowledge of the user.*
- Predict the user's indoor actions (e.g., turning on a light) *with high accuracy* in advance.

To confirm the efficacy of the IoTBeholder attack, we set up a real-world testbed including 23 popular devices from 7 different vendors. Our experimental results show that IoTBeholder can identify a wide variety of devices from different vendors with higher accuracy than prior techniques. To verify the effectiveness of our work on the perception of user's personalized habitual behavior, we invite volunteers to live in an apartment with these smart devices to generate a real-world smart home environment. The result shows that, even in the presence of multiple users and more than 10 automation rules, IoTBeholder is still able to identify the user's personalized habitual behaviors, and deployed automation rules with 100% accuracy. For user indoor action prediction, IoTBeholder attains over 92% accuracy. Its efficacy reveals the weakness of today's smart devices with respect to security and privacy. Our contributions include:

- We propose IoTBeholder, a novel attack that combines various information about smart home devices to infer users' habitual behaviors, thereby highlighting the privacy threat introduced by these smart devices (Section 4).
- We propose a method to identify different IoT devices with high accuracy, while only observing 802.11 headers with coarse attributes. Thus, IoTBeholder does not require access to encrypted data (Section 5).
- We propose a method for efficiently identifying device activity in conjunction with logical corrections to improve the accuracy of the attack (Section 6).
- We propose a novel user habitual behavior inference scheme based on the use of smart home devices (Section 7 and 8). The attack requires no prior knowledge about user behavior. To the best of our knowledge, this is the first attack that combines conventional sequential pattern mining technique and state-of-the-art deep learning models for user behavior analysis in smart home scenarios.
- We build a smart home testbed with 23 different devices from 7 different vendors and 10 automation rules from the real world. We evaluate the efficacy of IoTBeholder with the assistance of volunteers (Section 9).

2 RELATED WORK

Prior works have explored the risk of privacy leakage in smart homes from different perspectives. Table 1 summarizes these prior works, alongside a comparison with IoTBeholder.

2.1 RF-based Localization and Tracking

Prior work has confirmed the possibility of coarse-grained localization of signal sources using Radio Frequency (RF) receiving devices in a Wi-Fi environment. Many works propose practical hidden device recognition technologies to enable efficient screenings for threats of hidden electronic devices in daily life [4, 36, 54]. Another series of works propose methods for locating IoT devices in a user's residence given the floor plan [14, 20, 21, 46]. These works can leak the users' privacy to a certain extent, but are only limited to imprecise location of devices in a user's home, while the attacker cannot learn further information about the devices.

Another use of RF signals is for indoor human tracking. They use signal transmitters to actively sense the location of the entity in the area [7, 8, 63, 66], or passively sniff RF signals to infer the location of the object [35, 59, 69]. This type of works can be used for user tracking and even for assisting physical crimes like burglary. However, these methods expose limited user privacy, and cannot be used to predict user actions.

2.2 Household IoT Device Classification

Works on home IoT device classification have been extensively discussed in the networking context [10, 11, 24, 40, 42, 50, 55]. A series of papers propose device fingerprinting systems deployed on gateways [31, 42, 50]. Ma et al. identify the type of devices using the temporal pattern recognition of traffic [40].

Nevertheless, these approaches make strong assumptions that an attacker is capable of sniffing the TCP/IP traffic inside the home network or from the home gateway to the wide area network. This is infeasible for most attackers, considering that most real-world home gateways are secured by Wi-Fi Protected Access (WPA). Accordingly, this paper only relies on encrypted Wi-Fi packets to reflect a more realistic scenario.

Table 1. Comparing existing works vs. IoTBeholder

	Compatible with				
Approach	Limited Access to Network	Limited Access to Physical Environment	High-accuracy Device Classification and Activity Identification	Perception of User Behavior without Prior Knowledge	Action Prediction of User
Pre-Deployed Devices [16, 33]	✔	✗	✗	✔	✗
Human Indoor Tracking [7, 8, 59, 60, 63, 69]	✔	✔	✗	✗	✗
Wireless Encrypted Packets over Air [4, 5, 23, 37, 52]	✔	✔	✗	✗	✔
Network Traffic at Router [24, 40, 58]	✗	✔	✔	✗	✔
Prior Work on User Behavior Analyze in Smart Home [12, 34, 57]	✗	✔	✗	✗	✔
IoTBeholder	✔	✔	✔	✔	✔

2.3 IoT Device Activity Identification

Building on device classification research, a large body of work discusses how to use network traffic signatures to identify device activities [5, 9, 22, 29, 49, 58]. OConnor et al. use the statistics of some attributes in the traffic between IoT devices and the cloud to identify activities [49]. Trimananda et al. observe the request-and-reply pairs generated when the device interacts with the cloud, and propose a method named *pingpong* that uses these pairs for activity identification. This type of work is relevant to ours because they demonstrate the possibility of exposing ongoing device activities in smart homes to a third party.

Yet these methods face the same dilemma as previous work, namely the need to obtain information at the IP layer and above. [58] uses only packet length and direction to identify device activities, but performs poorly

in unreliable wireless network transmission environments (which we discuss in Section 9). More importantly, they ignore the correlation between user behaviors over time. Specifically, they do not exploit repeated patterns to predict what users may do. This limits such works to relatively shallow leakage of users' habits. Acar et al. correlate the sequences of device activity with user behaviors [5]. However, they focus on known and simple user behaviors, and thus lose most of the meanings of privacy leakage.

2.4 User Behavior Analysis in Smart Homes

Research has investigated human behavior in smart homes from various aspects, including movement patterns, app usage history [34], and event fingerprints [12, 57] for the benefit of elderly care and abnormal behavior recognition [27, 53]. However, they are not able to work without access to sensor data inside the smart home.

Some works use encrypted traffic to analyze user behavior in smart home scenarios [52, 58]. Sanchez et al. identify newspaper websites accessed by the smart device over the encrypted network. However, they also ignore the correlation between user behaviors over time. Due to the unpredictability of certain user behaviors and the emergence of multi-user scenarios, it may be difficult to obtain these habits through manual analysis (not to mention the extensive labor required). This also makes it difficult for attackers to use existing technologies to accurately predict user behavior for physical attacks.

3 PROBLEM SETTING AND THREAT MODEL

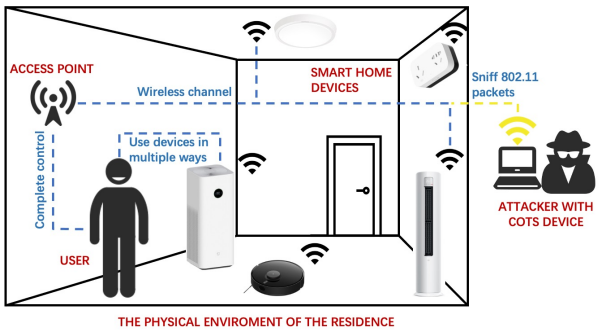

Fig. 1. Threat model with the user, the attacker, and the IoT devices in the smart home

In this section, we first describe our threat model. Our setup contains two actors: the users and the attacker. The users are one or more beneficiaries of smart homes, who have deployed a number of IoT devices. The attacker can be anyone interested in the living habits of the target user. Attackers want to achieve this without attracting user attention or paying extra, e.g., getting additional permissions or buying specialized equipment.

Figure 1 shows an overview of key actors and resources. The physical environment refers to the users' residence, which can be a room in a hotel, a complex multi-room apartment, etc. The IoT devices in the smart home are connected to the Internet through a user-controlled 802.11 wireless network. IoT devices could be of various types, such as cameras, lights, etc. Next, we formulate the capabilities and constraints of the user and attacker, respectively.

User Capabilities and Constraints: Users live normally in this environment and use IoT devices according to their own preferences. We make the following assumptions:

- **Physical Environment:** The users can deploy IoT devices anywhere in the home. Although the users can remotely operate the IoT devices through the app developed by the vendor, it does not prevent us from assuming that most of the time the user is in the room where the device being operated is located (based

on daily experience, people usually focus on changing the environment that they are in), except for device activities triggered by automation rules.
- **IoT Devices:** The users can buy and install any type of IoT device. They can operate these devices physically or through automated platforms. Note, here we assume users cannot alter the fundamental functions of these devices, such as modifying the firmware or changing wireless transmission. This is similar to previous work [19, 38, 42, 55]. In addition to controlling the devices via the app, users can physically operate the device, or deploy rules on the automation platform.
- **Wireless Network:** The users have an 802.11 wireless network and access point. Users can secure the Wi-Fi LAN in any way, such as setting access keys, encrypting traffic (e.g., WPA2/WPA3), and deploying other security tools (e.g., IDS, firewall) to ensure that devices within the LAN are not compromised.

Attacker Capabilities and Constraints: We make the following assumptions about attackers:

- **Physical Environment:** The attacker has limited access to the physical environment in which the user lives. We assume an attacker can walk around the user's residence (e.g., their garden) or deploy COTS equipment for packet sniffing without being discovered by the users, and can also obtain the floor plan of the user's home.
- **IoT Devices:** The attacker has no prior knowledge of the devices in the user's home, including where, how many, and what types of devices are deployed. Infecting any device in a user's home is considered impossible due to the limited capabilities of the attacker. This means the attacker cannot interfere with the normal actions of other devices, nor can they obtain device usage logs.
- **Wireless Network:** The attacker does not have access to the local area network in the user's home. An attacker cannot join the network without the access key. However, an attacker can still put their phone or laptop in promiscuous mode to sniff encrypted broadcast Wi-Fi 802.11 packets over the air. Given that most people do not choose to disable broadcasting their SSID [30, 47, 48], and attackers can still obtain the SSID after deploying this measure [6], we assume that it is usually trivial to determine the SSID of the LAN in the user's home.

Based on the above practical assumptions, it is challenging to implement IoTBeholder. First, without physical access to the user's home, the attacker cannot deploy any device for snooping in the user's home (e.g., [16, 33]), nor know where the IoT devices are deployed. Second, the attacker only has commercial-off-the-shelf (COTS) equipment (e.g., phone, laptop) and cannot access the wireless network in the user's home. In this case, information at the IP layer and above is invisible to the attacker. Third, we assume no prior knowledge about the user. This means that we cannot get the user's habitual behaviors in advance (as assumed in [5, 57]). Instead, we must extract regular user behaviors from the device activity sequences that represent user habitual behaviors. Fourth, due to the unpredictability of certain user behaviors and the emergence of multi-user scenarios, there is a lot of *noise* in the user behavior records obtained by the attacker, which makes it difficult to simply use the deep learning model to accurately predict the user's subsequent behavior . Given these constraints, existing methods are not sufficient in our scenario.

In response to the above observations, IoTBeholder relies on COTS devices, allowing an attacker to obtain the personalized habitual behavior of smart home users. By sniffing Wi-Fi connected IoT devices, it infers the user's personalized habitual behavior and automation rules defined in the platform and predicts the target user's subsequent actions. With the aforementioned constraints, the IoTBeholder attack is deployable, practical, low-cost and highly stealthy.

Definitions: Before proceeding, we highlight the following terms, which are used throughout the paper:

- *Device activity* (or activity) refers to a specific function of an IoT device, e.g., turning on/off a light, changing the temperature of an AC, etc. As most smart devices rely on data synchronization with an IoT hub or

the cloud, we assume that device activities (triggered by any methods, such as apps, physical buttons or automation rules) will all generate network traffic.

- *User action* (or action) refers to an operation performed by a user on a device, which subsequently leads a *device activity*. *User action* and *device activity* are equivalent in this paper. The former is from the perspective of human, the latter is from the perspective of devices.
- *User event* (or event) refers to a user's daily routine events, such as returning home, taking a shower, or going to bed. In a smart home, a series of smart devices can be utilized to help the user fulfill the event (e.g. automatically turning on the kettle). Thus, a sequence of device activities may expose that a specific event is taking place.
- *Habitual behavior* refers to a user's (unique) personal habits, made-up of a combination of specific actions for a given event, such as turning off the desk light and turning on the bedlight before going to bed. Here, going to bed is the event, whereas the combination of actions represent a habitual behavior. Habitual behaviors are likely to be different among users, and therefore the leakage of such information may lead to the exposure of identity or even the future prediction of the user behaviors.

4 SYSTEM OVERVIEW

IoTBeholder is an attack that relies on attacker's phone or laptop. It first requires the attacker to surround the target users' residence for basic device information collection and localization. The attacker can then deploy the device in the surrounding and continue to sniff encrypted 802.11 packets. The attacker can query the current list of identified devices and the current state of the devices or ongoing activities (if any). After collecting a sufficient amount of data, IoTBeholder is able to identify the users' personalized habitual behaviors and automation rules that the user has deployed. At any point after this, the attacker can also request IoTBeholder to make predictions about the user's indoor actions. Figure 2 shows a high-level architecture overview of IoTBeholder.

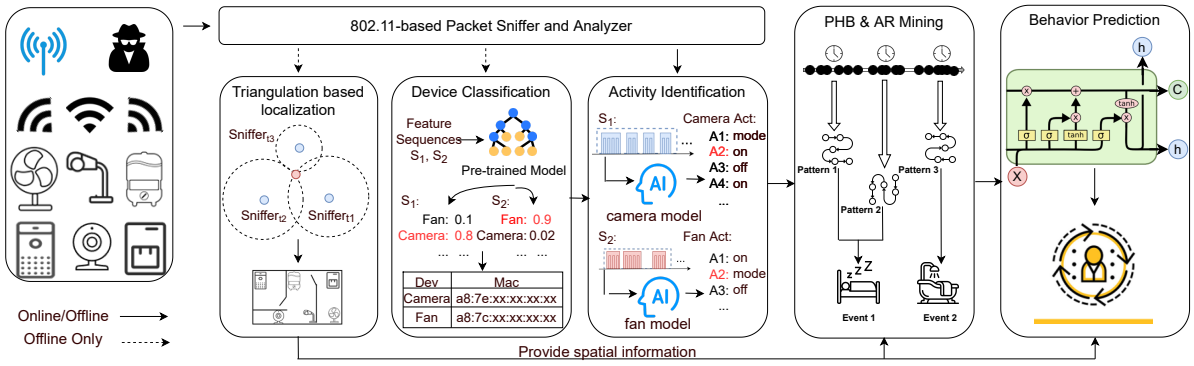

Fig. 2. High-level architecture overview of IoTBeholder

IoTBeholder consists of five main modules:

- **Wi-Fi Based Device Localization Module.** This module is designed to provide the attacker with the coarse-grained location of devices in the target user's home. We use traditional wireless localization methods [14] to map each device to its room. During this process, IoTBeholder also extracts the MAC address of the Wi-Fi packet header as the unique identifier of each device. We will not elaborate on this module in the following, as we use a proven approach.
- **Device Classification Module.** To address the problem of device classification in feature-poor situations and the unreliable nature of wireless network transmission, we build a machine learning model which

processes the raw Wi-Fi packets collected and estimates the most likely type of the device. We discuss the details of this module in Section 5.

- **Device Activity Identification Module.** Prior works discuss device activity identification in the IP layer and above, yet they do not work at the MAC layer [29, 49, 58]. To address this challenge, we design a method that combines machine learning methods, and a logical correction function, which estimates the most likely device activity derived via the machine learning model. The details of this are elaborated in Section 6.
- **Habitual Behavior and Automaton Rules Mining Module.** This module is based on the device activity identification module. As an input, it takes the activity sequences of the devices; it then mines the potential habitual behaviors of users, as well as the automation rules that the users deploy on the automation platform. We combine a sequential pattern mining algorithm and a hierarchical clustering algorithm based on edit-distance ratio to address this task. The details are elaborated in Section 6.
- **User Action Prediction Module.** This module completes the work of predicting the indoor action of the user. It combines a sequential pattern discovery algorithm with state-of-the-art deep learning methods to model user behaviors in smart home scenarios. This is elaborated in Section 8.

5 DEVICE CLASSIFICATION MODULE

The purpose of the device localization module is to obtain the coarse location of smart home devices without requiring physical entry to the user's home. During the device localization process, we let the attacker move around the user's residence with a Wi-Fi signal sniffer (e.g., MacBook), to obtain information about the signals emitted by the device. We then apply a triangulation algorithm [14] to estimate the approximate location of the device. IoTBeholder also extracts the MAC address of the Wi-Fi packet header as the unique identifier of each device during this process. Note, this does not require privileged access to the building and, instead, can easily be performed from outside.

After that, IoTBeholder's device classification module works by taking the encrypted 802.11 packets transmitted by the IoT devices as input. It then groups the collected packets based on MAC addresses and generates fingerprints through feature engineering. The module classifies the input fingerprints and matches each MAC address with a device type.

Identifying device types by network traffic has been discussed in both unencrypted and encrypted scenarios [4, 5]. However, they lack consideration of the difficulty of data collection in practical scenarios. Unlike scenarios where traffic information can be obtained precisely (for example, port mirroring, sniffing on the gateway, etc), there is no connection between the man-in-the-middle sniffer and the IoT devices. Considering the unreliable nature of wireless network transmission, these methods result in low accuracy. To address the issue, this section presents a machine learning-based framework that combines device traffic characteristics from multiple windows to accurately classify devices.

In this section, we analyze the properties of encrypted 802.11 packets that are helpful for device classification as well as the effects of unstable network transmission. Finally, we introduce the classifier used within IoTBeholder.

5.1 Feature Engineering

Encrypted 802.11 packets provide many attributes, even if the sniffer has limited access, such as packet size, packet type, subtype, QoS control, etc. Figure 3 shows scatter plots of packet lengths obtained by a man-in-the-middle wireless Wi-Fi sniffer for four IoT devices in the idle state. Here, a device in the idle state is considered to have not been operated by a user or triggered by an automated rule during this period. Note that the packet length has positive and negative values. A positive value indicates that the source MAC address of the packet is the device MAC address, and a negative value is the opposite.

We make three key observations in Figure 3. The first observation is that the sequence of packet lengths is still a useful feature to distinguish various devices. Idle traffic bursts from the same device have similar patterns. Yet, compared with the Xiaodu-audio and Mecury-camera, there are obvious differences in packet lengths. The second observation is that the traffic generated by devices in the idle state is periodic and varies among devices. For example, a Mijia-bedlight interacts with the cloud more frequently than an Mijia-AC_plug. The third and most important observation is that, because the man-in-the-middle sniffer does not establish a reliable connection with the device, the traffic pattern expected by the device is disrupted sometimes. To make matters worse, it is not occasional but consistent across the entire data collection. This is a key challenge for model referring because it means that (in a real wireless environment) the traffic gathered is likely to be incomplete and thus will bring about loss of information. This phenomenon does not appear to have been addressed in previous device classification efforts. In Section 9, we analyze the impact of this factor on the classifiers.

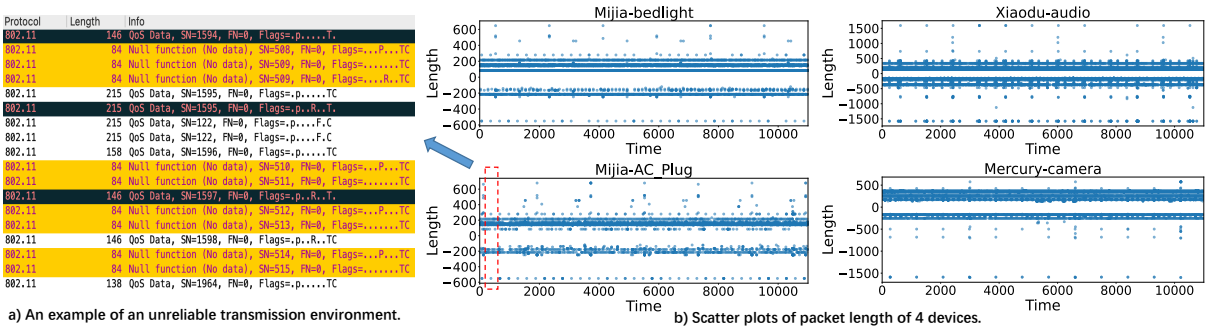

Fig. 3. Illustration of device traffic patterns and unreliable transmission environments.

Next, we introduce the idea of device classification module. We use the definition of traffic burst in [40] to split traffic and empirically set the minimum interval between two bursts to 0.1s. Based on the first observation, we keep the length and direction of a burst as part of our selection of features. To embody the periodicity identified in the second observation, we consider the time interval between each burst of the device as a feature. For up to ten packets within a burst, we extract their length, direction, subtype, and the interval between the burst to which it belongs and the end time of the previous burst. For cases where the number of packets is less than 10, we use 0 to fill in the missing part. However, the issue in the third observation remains unresolved. Common practice is to use each classifier to infer the feature vector, and the classification of the device is then finalized by majority voting. However, due to unreliable wireless packet transmission, the feature vectors obtained by the man-in-the-middle sniffer are distorted. This has a severe impact on the classification performance of the model (see Section 9). To address this problem, we do not use a single feature vector as the basis for judgment. Instead, We use the classifiers' predictions for all bursts over a period of time to get the final classification results. Doing so combines the inferred knowledge of the classifier over a period, thereby reducing the impact of bias caused by data corruption in each single burst.

5.2 Model Training and Inference

5.2.1 Model Training. We pick XGBoost as our machine learning model, and train a binary model for each of the devices shown in Appendix A, except for the lock.

5.2.2 Inference. When the classifier makes inferences, the sniffer first obtains encrypted 802.11 wireless data packets and aggregates them according to the MAC address in the header. Set $f_{d,i}$ represents the i^{th} feature vector from device d, and M_d represents the binary classification model whose main device class is d. A time

window, denoted as *win*, is set to combine the inferred knowledge of the classifier over a period, and finally get the classification result of the device according to formula 1.

IoTBeholder performs this step for each device represented by a MAC address until each MAC address matches a device type.

$$\text{Pred} = \text{argmax}_d \sum_{win} M_d(f_{d,i}) \tag{1}$$

6 DEVICE ACTIVITY IDENTIFICATION MODULE

Device activity identification is a further extension of the device classification. This module infers the ongoing activities based on the packets sent by the smart home devices. We consider it as a binary output problem, i.e., the appearance of an activity like turning on a light or changing the temperature on a thermostat. We do not attempt to infer the parameters of these activities (e.g., the exact temperature). Note, prior works discuss device activity inference in both in-network and wireless scenarios. However, due to the complexity of many IoT devices, it is impractical to consider only a few activities of a device. For example, most Mijia devices provide users with over 3 modes of operation, with at least 4 different activities.

All **79** device activities operated by volunteers during our experiments are presented in Table 7. In addition, we select some high function activities of some speaker devices in [51] (⑧⓪ to ⑨④) as a supplement to verify this module. The table highlights that devices can be operated physically by the user, through an app on the phone, or by the cloud due to predefined automated rules. We generate activity fingerprints using a similar fingerprinting method to the device classification module (minus the *interval*). Then, a binary model is trained for each individual device activity, and a majority vote is used to determine the device activity corresponding to the input. Unfortunately, we find that many device activities are indistinguishable because the traffic corresponding to them is exactly the same. This is also observed in [58]. For example, manipulating the AC_plug to adjust the wind speed and adjust the temperature produces both (-583, 88, 311, -88, 263, -88) packet length sequences.

To address this problem, we introduce a logical correction function to assist the classifier's inference. We first divide device activities into conditioning activities and switching activities. Switching activities include the turning on/off activity. Conditioning activities include the device activities caused by some functional operations performed on the device (except for turning on and off). For example, a bed light has two conditioning activities: change mode and change light. We then make three assumptions:

(1) The conditioning activity of the device always occurs after the turn-on activity of the device.
(2) The turn-on and turn-off activity of the device is alternately performed.
(3) The device will not be turned off immediately after being turned on, nor will it be turned on immediately after being turned off.

Based on these assumptions, we present a formalized representation of our real-time device activity inference. Assume $Pred = \text{argmax}_d M_d(f_{d,i})$ represents the voting result of the classifiers for a certain feature vector, and Δt represents the time threshold elapsed from the last switching operation. When the classifiers give the judgment $Pred$, we first determine the type of the activity. If it is a switching operation, and it has been more than Δt seconds since the last switch operation, the specific type of the operation is determined according to the state of the device at the current moment. Otherwise, we judge it as a possible conditioning activity. If it is a conditioning activity, the activity type is determined according to the judgment result of the classifier, and the device state is updated to be on at the same time.

7 HABITUAL BEHAVIOR AND AUTOMATION RULES MINING MODULE

In the previous sections, we have described how to infer the approximate location, type, and ongoing activities of IoT devices in a smart home from encrypted wireless 802.11 packets. We next present the possibility of mining

a user's personalized habitual (regular) behaviors based on this inference. Habitual behavior refers to a user's (often unique) personal habits, made-up of a combination of specific device activities for a given event, such as returning home or showering. Many prior studies [5, 37] ignore this step as they assume the user's behavior on a given event is common and easy to know, while we find such information can also leak valuable privacy, such as identity and preference, considering the distinguishability of personalized behaviors between different users. Therefore, we build this module to perform the mapping from a sequence of device activities obtained by the previous modules to an event, which constitutes a profile of a user's habitual behaviors on this event. The input to this module are the raw device activity sequences output by the previous modules. The task of this module is to mine the habitual behaviors of users from the device activity sequences (which are likely to consist of several days of data).

Two challenges must be overcome to achieve this goal. First, extracting habitual behaviors of a user (i.e., a stable pattern of a device activity sequence for an event) is not simple. For example, suppose a user usually turns on their bed light after turning off the desk light when about to sleep. Initially, one might assume these could be used to identify the user's habitual behavior. However, these two actions may not always be adjacent because the user may sometimes perform other actions between them, e.g., going to the toilet. Second, the extracted habitual behaviors, such as returning home and getting ready to sleep, should be distinguishable. Besides, there may be multiple users operating the devices at the same time. Thus, the habitual behaviors of the different users must be distinguished.

Accordingly, we combine a sequential pattern mining algorithm and edit-distance ratio based hierarchical clustering to address these issues. We formally define the problem and elaborate our solution in the following two steps.

7.1 User Personalized Habitual Behavior and Automated Rule Mining

Let $A = \{a_1, a_2, \ldots, a_i, \ldots, a_n\}$ represents the set of all possible device activities. $\mathbf{S} = \{S_1, S_2, \ldots, S_i, \ldots, S_m\}$ represents a dataset consisting of non-empty chronologically ordered activity sequences, where $S_i = \{(o_1, t_1), (o_2, t_2), \ldots, (o_i, t_i), \ldots, (o_T, t_T)\}, o_i \in A, t_1 < t_2 < \ldots < t_i < \ldots < t_T$. We use the sequence identifier (s_{id}) and timestamp (t_{id}) to uniquely identify an atom named X. We define a sequence α with k items (k = $\sum_j |\alpha_j|$) as a *k-sequence*. The set of frequent *k-sequences* is denoted as $\mathcal{F}_k$. The support of a sequence, denoted as $\sigma(\alpha, \mathbf{S})$, is the total number of input sequences in the database $\mathbf{S}$ that contain α. The *importance* of a sequence, defined as $\delta(\alpha, \mathbf{S})$, is a variable related to sequence length and support shown in 2. We introduce it to address the fairness-related issues when comparing a small number of long sequences with a large number of short sequences. Given a user-specified threshold called the minimum support (denoted as *min_sup*), we say that a sequence is frequent if it occurs more than *min_sup* times. More details of these definitions are elaborated in [56, 65].

$$\delta(\alpha, \mathbf{S}) = len(\alpha) \times \sqrt{\sigma(\alpha, \mathbf{S})} \tag{2}$$

Let us associate each atom X in the sequence to its list of identifiers, denoted as $\mathcal{L}$, which is a list containing all input sequence (s_{id}) and time identifier (t_{id}) pairs for that atom. We define a function $p : (S, N) \rightarrow S$, where S is a sequence and N is the set of non-negative integers, $p(X, k) = X[1 : k]$. In other words, $p(X, k)$ returns the k-length prefix of X. We define an equivalence relation θ_k on lattice $\mathbf{L}$ as follows: $\forall X, Y \in \mathbf{L}$, where we say that X is related to Y under θ_k, denoted as $X \equiv_{\theta_k} Y$ if and only if $p(X, k) = p(Y, k)$. That is, two sequences belong to the same class if they share a common k-length prefix. $[X]_\theta$ is a sequence that collapses all sequences with a common item prefix into an equivalence class.

Given $\mathbf{S}$ as input-sequences and parameters *min_sup*, *min_imp*, the problem of mining sequential patterns is to find all frequent sequences in $\mathbf{S}$. First, we traverse all device activities a_i with support greater than *min_sup* in $\mathbf{S}$ to form $\mathcal{F}_1$. Then, according to the element in $\mathcal{F}_1$, the frequent 2-sequences candidates with support greater than *min_sup* are calculated by the temporal join method (see details in [65]). Next, we traverse all $[X]_\theta$ and calculate

the support of candidate sequences composed of atoms between them. For a k-sequence N which $\sigma(N, \mathbf{S})$ is greater than *min_sup* and $\delta(N, \mathbf{S})$ is greater than *min_imp*, if the start timestamp and end timestamp are within 1s, we add it to $\mathcal{R}_k$, otherwise we add it to $\mathcal{F}_k$. For each newly obtained N, we repeat the above steps until no new frequent sequences are generated. Finally, we get $\mathcal{F}$, containing all the frequent sequences, and $\mathcal{R}$, containing all the automated rules. The pseudocode for this process is shown in Algorithm 1.

7.2 Edit-distance Ratio Based Hierarchical Clustering

After obtaining frequency sequences $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n\}$, an immediate task is to distinguish sequences representing different events. We address this problem using hierarchical clustering based on the edit-distance ratio. The edit-distance ratio between the two sequences is calculated by Formulas 3 and 4. Compared to the general edit-distance, these formulas consider that when calculating the distance of two sequences composed of the same activity, their distances are smaller than others. This is due to the potential presence of consecutive device activities, such as the adjustment of an AC one degree upon a time. Naturally, this should not be separated into different user events.

Algorithm 1 User Personalized Habitual Behavior Mining

Input: *min_sup*, *min_imp*, S
Output: $\mathcal{F}, \mathcal{R}$

$\mathcal{F}_1 \leftarrow Get\_Frequent\_Itemset(min\_sup, S)$;
$\mathcal{F}_2 \leftarrow Temporal\_List\_Join(min\_sup, S, \mathcal{F}_1)$;
$\xi \leftarrow Get\_Equivalence1\_Classes(\mathcal{F}_1, S)$;
for all $[X] \in \xi$ **do**
    $\mathcal{F} \leftarrow Enumerate\_Frequent\_Seq([X])$;
end for

function ENUMERATE_FREQUENT_SEQ(S)
    for all atoms $M_i \in S$ **do**
        $T_i \leftarrow \emptyset$;
        for all atoms $M_j \in S$, with $j \geq i$ **do**
            $N = M_i \vee M_j$;
            $\mathcal{L}(N) = \mathcal{L}(M_i) \cap \mathcal{L}(M_j)$;
            if $\sigma(N) \geq min\_sup$ and $\delta(N) \geq min\_imp$ **then**
                $T_i \leftarrow T_i \cup \{N\}$;
                if $N_{last} \leq 1$ **then**
                    $\mathcal{R}_{|N|} \leftarrow \mathcal{R}_{|N|} \cup \{N\}$;
                else
                    $\mathcal{F}_{|N|} \leftarrow \mathcal{F}_{|N|} \cup \{N\}$;
                end if
            end if
            $\mathcal{F}, \mathcal{R} \leftarrow Enumerate\_Frequent\_Seq(T_i)$
        end for
    end for
end function

An important parameter for the above clustering is *dis_thre*. Let $E_i, i = 1, 2, \ldots, n$ denote the clusters (hereinafter called events) formed by the clustering algorithm. For each $\mathcal{F}_i$, if its edit-distance ratio to any sequence in E_i

is less than *dis_thre*, we add it to E_i. We first set *dis_thre* to a large value to separate sequences representing different activities. For the same events, multiple users may perform similarly. Therefore, for the formed clusters, E_i, we use a smaller *dis_thre* to distinguish different users. In the end, we get $E = E_i, i = 1, 2, \ldots, n$. According to our observation, each E_i represents the habitual behaviors of an event for users, such as the behavior just after returning home, or after taking a shower. We discuss this interesting observation in detail in Section 9.5.

$$\mathrm{dis}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} \mathrm{dis}_{a,b}(i-1,j) + 1 & \text{if } a_i = a_{i-1} \\ \mathrm{dis}_{a,b}(i,j-1) + 1 & \text{if } b_j = b_{j-1} \\ \mathrm{dis}_{a,b}(i-1,j) + 2 & otherwise \\ \mathrm{dis}_{a,b}(i,j-1) + 2 & otherwise \\ \mathrm{dis}_{a,b}(i-1,j-1) + 4_{(a_i \neq b_j)} \end{cases} \end{cases} \tag{3}$$

$$dis\_ratio_{a,b} = \frac{len(a) + len(b) - dis_{a,b}}{len(a) + len(b)} \tag{4}$$

8 USER ACTION PREDICTION MODULE

After extracting the personalized habitual behaviors of users, an attacker may wish to *predict* a user's actions. Here, the action refers to what the user will operate on the devices at a later time. The user action prediction module predicts the user's actions by combining the knowledge of user behavior habits obtained by the previous module. Note, our attack not only predicts the user's *next* action but also the user's subsequent action sequence as much as possible. This is important as it may allow the attacker to carry out more serious attacks, e.g., burglary.

There are three main challenges that need to be addressed to achieve accurate user action prediction. First, we need to build a model to learn the temporal correlations of user behavior. For example, a user may always turn on a certain light before going to bed at the same time each day. However, given that the duration of the user's behavior sequence is not fixed, this model also needs to perform well in dealing with long-term temporal relationships. Second, there might be significant differences in the association between different activities. For instance, there might be a high correlation between turning off the desk light and adjusting the air conditioner, because these two behaviors usually occur together before bedtime. Despite this, turning off the desk light has nothing to do with entering the door. Using simple encoding mechanisms (such as one-hoting encoding or that listed in Table 7) fails to reflect these complex semantic relationships. Third, recalling what we described in Section 1, there can be unpredictable events in user action sequences. Further, an action sequence may contain actions from multiple users. This means that actions that are not strongly correlated may appear in the same sequence. This will seriously affect the embedding layer in learning the correlation between different user actions.

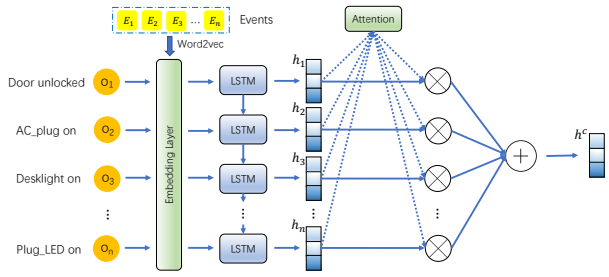

Fig. 4. The architecture of IoTBeholder's user action prediction model.

We choose an attention-based LSTM [67] to address the first problem due to its ability to model long-term historical information within temporal sequences. By computing the attention scores of latent representations at different moments, the attention mechanism can not only capture important information of temporal sequences but also offer interpretability. For the second problem, we adopt skip-gram-based Word2Vec [43, 44] to learn the representations of device activities while considering contextual semantic information. For the third problem, the user habitual behavior and automation rules mining module plays another important role at this time. Since the user actions in $\mathcal{F}_i$ that can be obtained by Algorithm 1 are naturally strongly correlated, we can use $\mathcal{F}$ to train the embedding layer. The user habitual behavior and automation rules mining module thus plays a dual role: it can not only extract the user behavior habits and automation rules explicitly, but also provide reliable prior knowledge for the user behavior prediction model. We conduct ablation experiments in Section 9.6 to demonstrate the necessity of user habitual behavior mining.

Figure 4 shows the architecture of IoTBeholder's user action prediction model. While training the model, we first input the raw device activity sequences, into the LSTM. Note that the labels for training come from the sequence representing the user's habitual behaviors contained in the corresponding raw sequence, that is, the output of the previous module.

9 EVALUATION

We proceed to evaluate the efficacy of IoTBeholder. We present our experimental setup, before showing our results.

9.1 Experimental Setup

Although some existing datasets have been collected on IoT device activities [51, 55], most of them are either collected at the network layer or do not take user usage of the device into account. This makes them unsuitable for evaluating IoTBeholder. Next, we describe our experimental setup.

9.1.1 Testbed and Participants. To consider a realistic and functional smart home, we deployed our experimental platform in an apartment. We invited three volunteers to live in the experimental environment successively. According to the China Population Census Yearbook 2020, in most cities, the proportion of one-person and two-person households accounts for more than 60%, and this proportion is increasing [62]. Therefore, we invited three participants to simulate the life of smart home users and collect their device usage data to verify IoTBeholder in both single-person and multi-person scenarios. The participants involved in the experiment included two men and one woman, none of whom are co-authors. In what follows, we refer to the participant as users.

The testbed consists of 23 recent IoT devices covering 7 vendors. It is located in an apartment with a living room, a bedroom, a kitchen, a bathroom and a hallway. The floor-plan and the device deployment is depicted in Figure 5. Among them, an LED light is connected to a smart plug, and a smart door lock is connected to a smart gateway via Bluetooth. The rest of the devices are connected to a wireless access point inside the apartment. A full list of devices is presented in Appendix A. More than half of the devices are from Xiaomi, as Xiaomi occupies the largest share of China's smart home market [68] and it is convenient for users to use the same app to operate the device and configure automation rules. These devices are configured according to the official instructions and have not been modified by us. Users can download the official app to operate the device, or physically operate the device. The automation rules are deployed by the users without any intervention from us.

9.1.2 Data Collection. We deploy a MacBook outside the apartment to sniff the encrypted Wi-Fi packets generated by the IoT devices through the wall. The collected data is filtered and aggregated based on MAC address and we keep only the data frame. For the device classification module, we collect data for 7 days when the devices are in

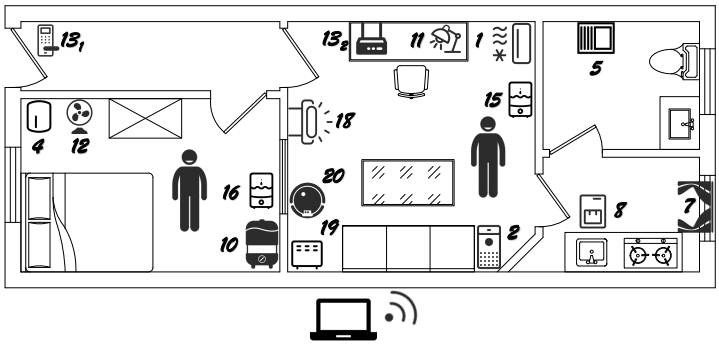

Fig. 5. Overview of testbed setup.

an idle state. The total size of the idle data is **12.5 GB**. For the device activity identification module, we trigger each activity in Table 7 150 times and obtain the Wi-Fi packets.

Finally, we collect 10 days of real-life data for user C, whereas 20 days of data is collected for user A and user B, during which they lived together in the apartment. During this time, the participants undertook their own habits undisturbed and used the equipment in the testbed. We validate the last two modules of IoTBeholder using the traffic generated by the device when used by the users. To avoid bias caused by not being used to the new residence, we let each user live in the experimental environment for at least two weeks before formally collecting data.

9.1.3 Ethical Concerns. All users have full knowledge of the IoT devices and apps used. After check-in, the control of all devices is handed over to the users. Participants were informed about the use of laptops to sniff encrypted Wi-Fi packets. We did not interfere in participants' lives during the experiment. At the participants' request, we disabled any devices that might save audio data, such as cameras and speakers, during the experiment.

The collected data is only visible to the authors of this paper. In addition, the privacy of user habitual behaviors extracted by IoTBeholder, are presented in this paper with the consent of the participants. The entire experiment is consistent with their local governing laws and regulations.

9.2 Performance of Device Classification

We first evaluate the classifier performance of IoTBeholder. We implement some prior work and compare them with our method under an unreliable wirlesss network environment.

9.2.1 Evaluation Metrics. We use *precision*, *recall*, and *F1 score* to evaluate the performance of the device classification. In our experiments, the <*Type, Vendor*> pair was used to uniquely identify a device. *Precision* is defined as the number of correctly classified samples divided by the total number of samples classified as the device, and *recall* is defined as the number of correctly classified samples divided by the total number of samples for the device. The *F1 score* is the harmonic mean of the two, to measure the performance of the classifier on a macro scale.

9.2.2 Evaluation Design and Dataset Division. We filter and group the data using MAC addresses, and then segment it into bursts as mentioned in 5.1. Each burst generates a fingerprint for training or testing. During the training phase, we manually label each burst. In the testing phase, we mark a label for all bursts in a window.

We choose the Multi-Layer Perceptron (MLP), K-Nearest Neighbor (KNN) algorithms and Fisher's Least Square Linear Discriminant (LDA) as our baselines. MLP is a classical deep learning method. KNN is a standard machine learning method, which is reported to achieve high accuracy [5]. LDA is another machine learning method

which is reported to achieve high accuracy on the task of classifying encrypted traffic generated when users visit different websites [52].

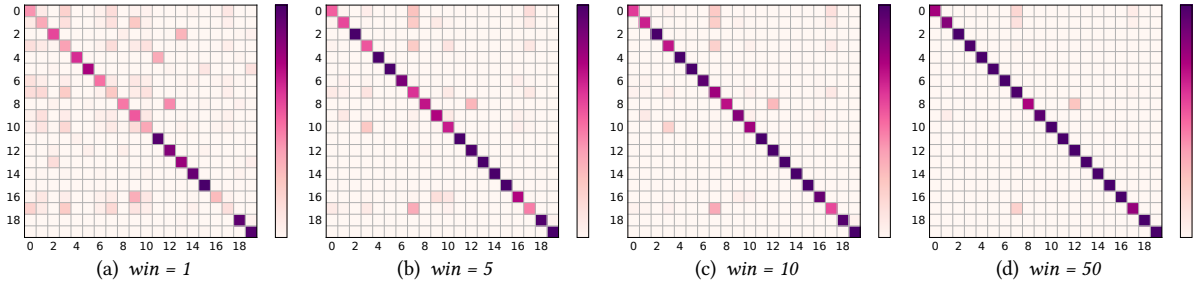

Fig. 6. Device classification confusion matrix using different *win* parameters (window size).

9.2.3 Evaluation Result. Figure 6 shows the confusion matrix for devices with different values of *win* (see Section 5.2). An obvious observation is that, as *win* increases, the performance of the classifier increases accordingly. This is intuitive because when the classifier only makes a judgment based on one burst, there is a high probability of an irregular deviation due to the unreliable network environment. The information loss caused by this factor makes the classifier unable to correctly use the learned knowledge for classification. With the increase of *win*, the confidence of the classifier corresponding to the labeling device will increase, because the occurrence of normal bursts will increase over time. When *win* is 50, we see that IoTBeholder achieves a high accuracy.

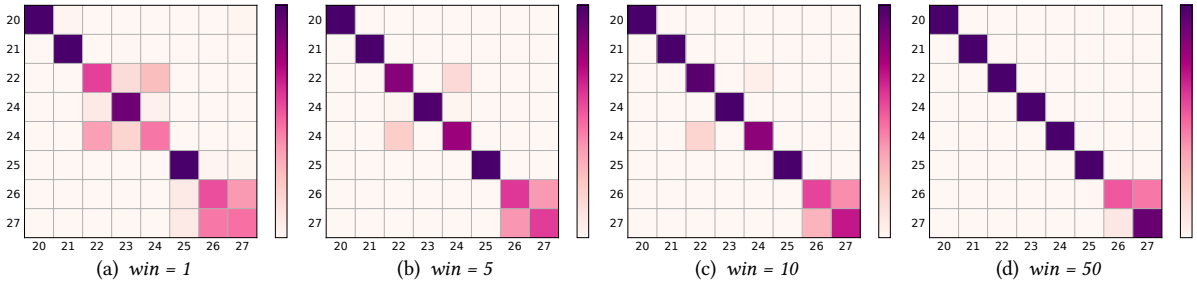

Fig. 7. Device classification confusion matrix using different *win* parameters (window size) of dataset US.

Another important observation is that devices belonging to the same vendor are more likely to be misclassified. For example, device 1 (mijia-AC_Plug) and device 2 (mijia-aircleaner) are easily misclassified as device 7 (mijia-curtains) when *win* takes a small value. To validate this observation, we repeat the experiment with other devices from the same vendor in the US dataset published in [51]. In order to simulate the impact of an unreliable transmission environment, we randomly delete a portion of packets, and then adapt it to the specific Wi-Fi encryption (WPA2) by simply compensating according to the method in [52]. The result in Figure 7 again confirms our observations. Among these 8 devices, two devices (20-21) from TP-LINK are easily distinguishable. The three devices (22-24) from Amazon are easily confused with each other. Yet this problem is gradually eased as *win* increases. The two devices (26-27) from Xiaomi are almost indistinguishable from each other. It shows that, for some vendors, their devices are more difficult to distinguish from each other by traffic analysis. This phenomenon can also be observed in the confusion matrix obtained by the authors' method [32, 42]. This confusion may be due to devices from the same vendor sharing some of the same interfaces, or developers sharing the same Software Development Kit (SDK). We believe that this deserves attention in the follow-up work of device classification.

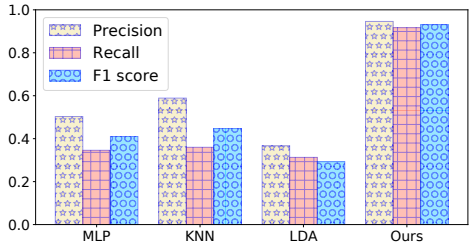

Fig. 8. Comparison of our device classification method with baselines.

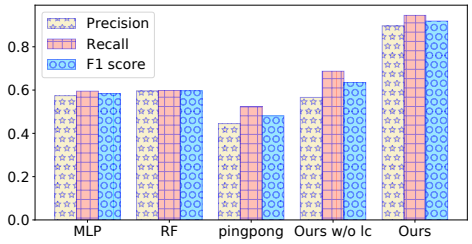

Fig. 9. Comparison of our activity identification method with baselines.

Finally, Figure 8 shows the comparison of our method with other device classification schemes. We see that IoTBeholder has the best performance on *precision*, *recall*, and *F1 score*. Among them, LDA is the worst. The method adopted in [5] (KNN) achieves better performance, but it is still not enough to solve the challenges brought about by the unreliable network environment.

9.3 Performance of Device Activity Identification

This section evaluates the device activity identification module. We compare state-of-the-art device activity identification methods working in 802.11 scenarios and evaluate our need for logical correction.

Table 2. Average *precision*, *recall*, and *F1 score* of the activities from different devices.

Device	Precision	Recall	F1	F1 (ours w/o lc)	Device	Precision	Recall	F1	F1 (ours w/o lc)
1	0.758	0.879	0.814	0.554	15	0.809	0.913	0.858	0.570
2	0.976	0.994	0.985	0.638	16	0.990	0.999	0.994	0.545
4	0.802	0.903	0.849	0.611	18	0.803	0.998	0.890	0.612
5	1	1	1	0.478	19	0.693	0.883	0.776	0.484
7	0.968	0.999	0.982	0.473	20	0.627	0.623	0.625	0.625
8	1	1	1	0.575	22	0.970	0.970	0.968	0.968
10	0.883	0.864	0.874	0.578	23	0.836	0.838	0.833	0.792
11	0.770	0.947	0.849	0.525	24	0.896	0.887	0.890	0.890
12	0.990	0.999	0.994	0.545					
13	0.999	0.928	0.962	0.962	Avg.	0.897	0.945	0.919	0.635

9.3.1 Evaluation Design and Dataset Division. We again use *precision*, *recall* and *F1 score* to evaluate the performance of device activity identification. Bursts are manually labeled with the corresponding activity. Data collected by the device classification module is labeled with *others*. We split the train and test set in an 8:2 manner. Note, thanks to the device classification module, we only need to distinguish the different activities of a device.

We choose MLP, Random Forest (RF), and the method in [58] as our baselines. MLP and RF are common deep learning and machine learning based methods [5, 42], respectively. [58] presents a rule-based method that requires only packet length and direction as features for device activity identification.

9.3.2 Evaluation Result. Table 2 shows the average *precision*, *recall*, and *F1 score* of the activities of different devices, as well as the *F1 score* without logic correction. We also conduct experiments using dataset US published in [51] containing device activities. The data processing procedure is the same as in the device identification part. IoTBeholder identifies device activities with an average of 89.7% *precision* and 94.5% *recall*. This confirms the worrying efficacy of the attack. Our logic correction plays a significant role in this process. Without logic

correction, the average *F1-score* across all devices drops by 28.4%. This is largely due to assumption 2 (see Section 6), because almost all switching operations belonging to the same operating method of a device are indistinguishable by the classifier alone, which is also observed in [58]. Assumptions 1 and 3 contribute to devices in which switching and conditioning operations are indistinguishable.

Figure 9 shows the comparison of our method with other device activity identification schemes. MLP and RF perform similarly in this case, while [58] attains poor performance. We believe this is another effect of an unreliable network environment. Although [58] claims to be able to work in a wireless network environment, its method is based on packet pairs. But in the scenario we set, packet pairs are likely to be corrupted. This again illustrates the worrying robustness of IoTBeholder.

9.4 Performance of Habitual Behavior and Automaton Rules Mining

Table 3. Habitual Behaviors Obtained and Confidence Score

User	Event	Activity sequence of event	Score	Event	Activity sequence of event	Score
A & B	1	⑳→㉘→⑫→⑬→㉖→㉗→⑰(→①)	✓	2	㊸/㊺→⑦→㊾	✓
	3	⑫→⑬→㉖→㉗(→④)(→④)	✓	4	㊷→㊿→⑧	✓
	5	⑰/㉑→㉙→④→㊲→㊶	✓	6	㊻→㊿→⑳(→㉑)	✓
	7	⑬→⑮→㉘→㉙→㉜	✓	8	㊳→㊺→㊹	✘
	9	㊶→⑲→㉛→㊼→㊽→㊳	✓	10	㉙→⑫→⑭	✘
	11	㊼→⑥→㊽→㉛	✓	12	㊾→㉝→⑯	✓
	13	㊼→㊼→㉚→㊾	✓	14	㊶→⑲→㉛→㊼→㊽	✓
	15	⑳/⑯→㊺→㊽→⑰→①	✓	16	⑫→⑬→㊹→㊹	✓
	17	㊶→㊺→㊷(→③)→⑫→⑬	✓	18	㊻→⑯→㊿→⑰	✓
	19	㉜→②→㊾→㊺→⑥→⑫→⑭	✓	20	⑬→⑮→⑳→㊸→㊻	✓
	21	㊼→⑥→㊽→㉛	✘	22	㊼→㊼→㉚→㊾	✓
C	23	⑬→⑮(→㉖→㉗)→㊾→㉞→⑯	✓	24	⑫→⑬→㊹→㊹	✓
	25	㊶→㊺→㊷→③→⑫→⑬	✓	26	㊻→⑯→㊿→⑰	✓
	27	㉜→②→㊾→㊺→⑥→⑫→⑭	✓	28	㉖→㉗→㊹→①	✓
	29	⑬→⑮→⑳→㊸→㊻	✓	30	⑫→⑬→④	✘
	31	⑳→㊺→⑫→⑬→⑰	✓	32	㊹→①→㉙	✘
	33	⑫→⑬→㉘→㉖→㉗→①	✓	34	㊺→⑯→㊿→㊱	✓
	35	⑰→㊸→㊼→㊽→⑲→㊼→⑥	✘	36	⑫→⑭→②	✓
	37	㊼→㊻→㊼→⑤→㊷→④(→④)	✓	38	㊿→①→㊱	✓
	39	㊼→㊻→㊼→⑤→㊸(→⑦)(→㊽)	✓	40	⑫→⑬(→②)→⑰	✓
	41	⑫→⑬→㊷→④(→④)→㊾→⑧	✓	42	⑳→㊺→㊽→⑰→①	✓
	43	⑫→⑬→㉙→㊸→㊻→㊶/㊸	✓			

9.4.1 Evaluation Design. For the collected data, we use the device activity identification module to process bursts one-by-one to determine which type of device activity they represent (or nothing at all). If there is no device activity within 10 minutes after a burst, we intercept and save the currently recorded device activities as a raw sequence. Each occurrence of a burst (representing a device activity) resets the waiting time. The raw sequences obtained in this step are used as the input of Algorithm 1, and then the edit-distance ratio based hierarchical clustering. The data in the single-user scenario and the multi-user scenario are verified separately.

9.4.2 Evaluation Metrics. An important evaluation criterion we set is the user's subjective evaluation, since the users have the clearest understanding of their habitual behaviors. In addition, we also objectively test the effectiveness of the algorithm based on the ground truth records of device activity on the automation platform. We combine the subjective and objective evaluations to rate a confidence score of each user's personalized habitual behaviors extracted by IoTBeholder. If both users and records indicate that this activity sequence is frequent, we rate it as high confidence (✓). If only one of the evaluations considers this activity sequence to be frequent, we rate it as medium confidence (✓✗). Otherwise, we rate it as low confidence (✗). For user-defined automation rules, we check the records on the automation platform to determine whether the automation rules obtained by IoTBeholder are indeed deployed.

9.4.3 Evaluation Results. Table 3 shows all the device activity sequences obtained by IoTBeholder and each of them implies a potential user event. For the single-user scenario, we have mined a total of 21 events of user C. Items with "()" represent the device activities that can either be present or not present in the sequence. Items like "a/b" mean either the activity *a* or activity *b* is present in the sequence.

We see from the confidence scores that most of the obtained device activity sequences are sufficiently representative for a user event. Further, for the multi-user scenario, we find a total of 22 habitual behaviors from user A and user B. This confirms the ability of IoTBeholder to profile a user's habitual behaviors. We make an illustration of these sequences in Section 9.5. We therefore confirm that Algorithm 1 is able to mitigate the issue of multiple users. Though the activity sequence of a single user may be included as a subsequence in this case, IoTBeholder can still mine the high-frequency subsequences hidden in long sequences.

Note that all automation rules set by the users are correctly identified according to the device usage records on the automation platform, as shown in Table 4. Although some automated rules often appear in the device activity sequence, these rules are properly separated by IoTBeholder via Algorithm 1.

Table 4. Automation rules used by the users.

User	Index	Smart app rules
A&B	**R1**	If locker is unlocked outside, turn on LED, open curtains and sweeper returns to charge.
A&B	**R2**	If locker is unlocked inside, turn off LED and AC, close curtains and begin sweeping.
A&B	**R3**	If fan is turned off, turn off bedlight, curtains, LED and desklight.
A&B	**R4**	If fan is turned on, turn on AC and deerma humidifier.
A&B	**R5**	If mijia humidifier is turned on, turn on deerma humidifier and close dehumidifier.
A&B	**R6**	If standheater is turned on, turn off fan and AC.
A&B	**R7**	If mijia humidifier is turned off, turn off deerma humidifier.
C	**R8**	If locker is unlocked outside, turn on desklight, AC and LED.
C	**R9**	If locker is unlocked inside, turn off AC, bedlight and desklight.
C	**R10**	If standheater is turned on, turn on dehumidifier.

9.5 Privacy Analysis

In this section, we illustrate the user privacy that IoTBeholder can reveal. IoTBeholder's leakage of user privacy is divided into multiple stages. The device location module and device classification module can reveal the approximate location and type of IoT devices deployed in the user's home. The device activity identification module provides real-time detection of the status and ongoing activity of IoT devices. However, we emphasize that IoTBeholder can extract the potential behavior patterns of users from the sequences of device activities and predict user actions. Importantly, this is done without any prior knowledge. Next, we focus on analyzing the specific meaning of the events obtained by the IoTBeholder attack and the risk of privacy leakage.

Figure 10 shows four of the events obtained by IoTBeholder representing user habitual behaviors. Among them, event 31 and event 39 come from the data collected from user C. Event 31 probably corresponds to the behavior of user C when getting up. User C first turns on the bedlight, then the LED in the living room through the app and enters the living room. Later, user C turns on the light of the bath heater and starts to wash. After that, user C turns off the light of the bath heater and walks out. An interesting observation is that user C tends to turn off the bed light via the app after washing up. Event 39 probably represents a sequence of actions just after returning to the residence. At the beginning of the event, user C unlocks the door from the outside. The desk light, LED, and AC_plug are turned on due to automation rules. Subsequently, the user enters the living room and turns off the desk light. At the end of the event, user C turns on the air cleaner and humidifier to adjust the air quality in the living room.

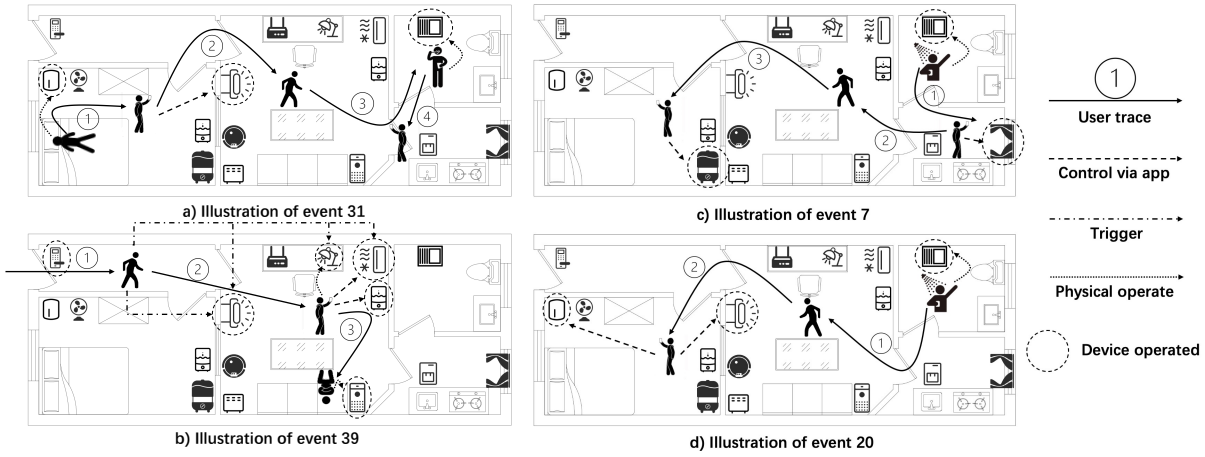

Fig. 10. Illustration of obtained events.

Event 7 and event 20 are classified as an event in the first step of hierarchical clustering, but are distinguished as generated by different users in the second step. In this case, the meaning of the two events is that both users will produce a series of actions after the bath. After one of the users finishes bathing, he first turns off the lighting and ventilation functions of the bath heater and then opens the curtains (user A explained that he was drying clothes afterward). Thereafter, he closes the curtains and turns on the dehumidifier. Another user, who was later confirmed to be user B, goes straight back to the bedroom after walking out of the bathroom and operates different from user A. Attackers can analyze the differences in the habitual behaviors of each user in the smart home based on these subtle differences.

Note, we may not be able to assign a clear meaning to all events, such as event 24 and event 41. However, these *unnamed* events often represent the user's personalized habitual behaviors. This further proves the seriousness of the privacy leak caused by IoTBeholder. IoTBeholder combines information from multiple dimensions to provide attackers with a comprehensive description of user behaviors and establish user portraits. The attacker can restore the details of every event based on the information provided by IoTBeholder.

9.6 Performance of User Action Prediction

This section evaluates IoTBeholder's user action prediction module and verifies the effectiveness of the personalized habitual behavior mining algorithm.

9.6.1 Evaluation Design. We train the LSTM model as defined in Section 8. The purpose of the model is to predict as many of the user's next actions as possible. Note that we do not require the model to strictly predict the next action in the sequence. When the prediction given by the model is a sub-sequence of the sequence that appears later, we consider the prediction to be correct. This also holds in the ablation experiment. We divide the train and test set in a ratio of 8:2. Single-user scenarios and multi-user scenarios are verified separately. For the ablation experiment, we train the LSTM without using the embedding layer, using the raw sequence to train the embedding layer, and using $\mathcal{F}$ to train the embedding layer. It should be noted that for the second case, the labels used come from raw sequences because $\mathcal{F}$ is unknown to it.

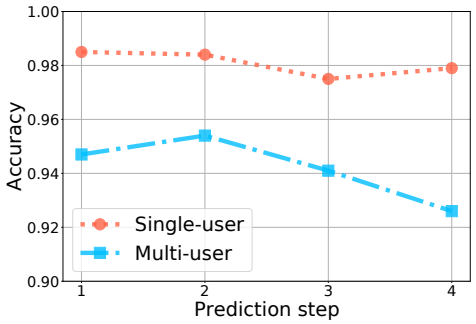

Fig. 11. Prediction performance of IoTBeholder on user actions.

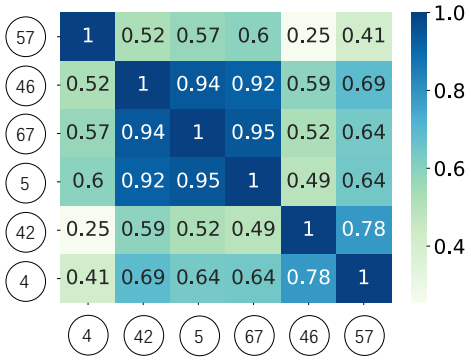

Fig. 12. Visualization of embedding layer.

9.6.2 Performance. Figure 11 shows the prediction performance of IoTBeholder on user actions. We see that the model maintains high accuracy of action prediction in both single-user and multi-user scenarios. For multi-user scenarios, the model performs slightly worse. This may be due to more similar activity sequences generated by users. Naturally, as the number of predicted steps increases, the accuracy of the model prediction will gradually decrease, but it remains above 92%. This shows the effectiveness of IoTBeholder in user action prediction.

9.6.3 Visualization of Embedding. We visualize the embedding layer of the model in Figure 12. The values in the embedding matrix represent the degree of association between two activities. It can be found that activity pairs with higher correlations appear frequently in the results of the user's personalized habitual behavior mining algorithm. Take activities in event 37 as an example: activity 46, 67 and 5 are highly ed because of the automation rules **R8**. Besides, there is also a high ion index between activity 42 and 4 because turning off the desk light is often followed by adjusting the air conditioner in some events. Although the embedding cannot represent the sequential relation, the association between activities reflects confirms the effective mining of users' personalized behaviors of IoTBeholder.

9.6.4 Ablation Experiment. We conclude by inspecting the three variations of our model. Table 5 presents the performance of the model predictions in the three cases. We see that the model performs the worst using the raw sequence to train the embedding layer. This shows that the embedding layer trained using $\mathcal{F}$ better learns the relations between device activities and provide better information for LSTM. However, the embedding obtained by training with raw sequences is disturbed by irregular data, which affects the prediction accuracy of the model. The standard attention-based LSTM achieves 0.509 and 0.693 accuracy in single-user and multi-user scenarios, respectively, but there is a clear gap with our method. This illustrates the effectiveness of frequent pattern mining algorithms combined with attention-based LSTM and the necessity of the habitual behavior and automaton rule mining module.

Table 5. Ablation Experiment.

		Without embedding	Raw sequences for embedding	Ours
Accuracy	Single-user	0.509	0.255	0.981
	Multi-user	0.693	0.286	0.951

10 CONCLUSION AND COUNTERMEASURES

This paper has delineated IoTBeholder, an attack that can identify the personalized habitual behaviors of users in smart homes without requiring special permissions or prior knowledge. We have implemented the various modules of IoTBeholder on a COTS laptop and built a testbed to evaluate it in the real world. The results demonstrate worryingly high accuracy on user habitual behaviors mining and action prediction. Our work confirms that, even in the real world, where attackers are severely restricted, smart homes still face considerable privacy risks.

Our ultimate goal is to improve the security and privacy of smart home users. The IoTBeholder attack relies on being able to sniff wireless Wi-Fi packets. Thus, it cannot be mitigated by wireless access point filtering or obfuscating egress traffic. One solution would be for devices to customize installed applications using the open source environment [17] to introduce fake packets (note, traffic shaping is now enabled for users of the Samsung SmartThings platform). Another possible countermeasure is for access points to inject miscellaneous traffic using the spoofed MAC addresses of device as the source. Although effective, these countermeasures introduce notable overhead. Our future work will therefore involve further refinement of these solutions.

ACKNOWLEDGMENTS

We would like to thank our AE and other anonymous reviewers for their constructive comments. This work is supported by the National Key Research and Development Program of China under grant No. 2020YFB1804704, the National Natural Science Foundation of China under grant No. 61972189, the Major Key Project of PCL under grant No. PCL2021A03-1, Shenzhen Science and Technology Innovation Commission: Research Center for Computer Network (Shenzhen) Ministry of Education, and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989.

REFERENCES

[1] 2021. MIJIA. http://home.mi.com/index.html.
[2] 2021. SmartThings. https://www.smartthings.com/.
[3] 2022. Apple Homekit. https://www.apple.com/ios/home/.
[4] 2022. Lumos: Identifying and Localizing Diverse Hidden IoT Devices in an Unfamiliar Environment. In *Proceedings of 31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA.
[5] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and A. Selcuk Uluagac. 2020. Peek-a-boo: i see your smart home activities, even encrypted!. In *Proceedings of WiSec '20: 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Linz, Austria*.
[6] ACRYLIC. 2022. Hidden wifi ssid: How to know the name of a wireless network with no SSID. https://www.acrylicwifi.com/en/blog/hidden-ssid-wifi-how-to-know-name-of-network-without-ssid/.
[7] Fadel Adib, Zachary Kabelac, and Dina Katabi. 2015. Multi-Person Localization via RF Body Reflections. In *Proceedings of 12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 15, Oakland, CA, USA*.
[8] Fadel Adib and Dina Katabi. 2013. See through walls with WiFi!. In *Proceedings of ACM SIGCOMM 2013 Conference, SIGCOMM 2013, Hong Kong, August 12-16, 2013*, Dah Ming Chiu, Jia Wang, Paul Barford, and Srinivasan Seshan (Eds.).
[9] Noah J. Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping. *Proc. Priv. Enhancing Technol.* 2019, 3 (2019), 128–148.
[10] Noah J. Apthorpe, Dillon Reisman, and Nick Feamster. 2017. A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic. *CoRR* abs/1705.06805 (2017). arXiv:1705.06805

[11] Noah J. Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic. *CoRR* abs/1708.05044 (2017). arXiv:1708.05044

[12] Oya Aran, Dairazalia Sanchez-Cortes, Minh-Tri Do, and Daniel Gatica-Perez. 2016. Anomaly Detection in Elderly Daily Behavior in Ambient Sensing Environments. In *Proceedings of Human Behavior Understanding - 7th International Workshop, HBU 2016 (Lecture Notes in Computer Science)*.

[13] Jacob Arellano. 2019. Very. Bluetooth vs. wi-fi for iot: Which is better? https://www.verypossible.com/insights/bluetooth-vs.-wi-fi-for-iot-which-is-better.

[14] Paramvir Bahl and Venkata N. Padmanabhan. 2000. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proceedings of IEEE INFOCOM 2000, The Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications, Tel Aviv, Israel*.

[15] Simon Birnbach, Simon Eberz, and Ivan Martinovic. 2019. Peeves: Physical Event Verification in Smart Homes. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.).

[16] et al. C. Southworth, J. Finn (Ed.). 2007. *Intimate partner violence, technology, and stalking*. Violence against women.

[17] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick D. McDaniel, and A. Selcuk Uluagac. 2018. Sensitive Information Tracking in Commodity IoT. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA*, William Enck and Adrienne Porter Felt (Eds.).

[18] Z. Berkay Celik, Gang Tan, and Patrick D. McDaniel. 2019. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In *Proceedings of 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA*.

[19] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2018. DeWiCam: Detecting Hidden Wireless Cameras via Smartphones. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea*.

[20] Kevin Chetty, Graeme E. Smith, and Karl Woodbridge. 2012. Through-the-Wall Sensing of Personnel Using Passive Bistatic WiFi Radar at Standoff Distances. *IEEE Trans. Geosci. Remote. Sens.* 50, 4 (2012), 1218–1226.

[21] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. 2010. Indoor localization without the pain. In *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking, MOBICOM 2010, Chicago, Illinois, USA*.

[22] Bogdan Copos, Karl N. Levitt, Matt Bishop, and Jeff Rowe. 2016. Is Anybody Home? Inferring Activity From Smart Home Network Traffic. In *Proceedings of 2016 IEEE Security and Privacy Workshops, SP Workshops 2016, San Jose, CA, USA*.

[23] Scott E. Coull and Kevin P. Dyer. 2014. Traffic Analysis of Encrypted Messaging Services: Apple iMessage and Beyond. *Comput. Commun. Rev.* 44, 5 (2014), 5–11.

[24] Shuaike Dong, Zhou Li, Di Tang, Jiongyi Chen, Menghan Sun, and Kehuan Zhang. 2020. Your Smart Home Can't Keep a Secret: Towards Automated Fingerprinting of IoT Traffic. In *Proceedings of ASIA CCS '20: The 15th ACM Asia Conference on Computer and Communications Security, Taipei, China*.

[25] L. Aneesh Euprazia and K. K. Thyagharajan. 2020. A novel action recognition system for smart monitoring of elderly people using Action Pattern Image and Series CNN with transfer learning. *CoRR* abs/2009.03285 (2020). arXiv:2009.03285

[26] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security Analysis of Emerging Smart Home Applications. In *Proceedings of IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA*.

[27] Chenglong Fu, Qiang Zeng, and Xiaojiang Du. 2021. HAWatcher: Semantics-Aware Anomaly Detection for Appified Smart Homes. In *Proceedings of 30th USENIX Security Symposium, USENIX Security 2021*.

[28] Carles Gomez, Stefano Chessa, Anthony Fleury, George Roussos, and Davy Preuveneers. 2019. Internet of Things for enabling smart environments: A technology-centric perspective. *J. Ambient Intell. Smart Environ.* 11, 1 (2019), 23–43.

[29] Tianbo Gu, Zheng Fang, Allaukik Abhishek, Hao Fu, Pengfei Hu, and Prasant Mohapatra. 2020. IoTGaze: IoT Security Enforcement via Wireless Context Analysis. In *Proceedings of 39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada*.

[30] Amit Gupta, Victor Ojong Etta, Arif Sari, Agbotiname Lucky Imoize, Piyush Kumar Shukla, and Musah Alhassan. 2022. Assessment and Test-case Study of Wi-Fi Security through the Wardriving Technique. *Mobile Information Systems* 2022 (2022), 7936236. https://doi.org/10.1155/2022/7936236

[31] Danny Yuxing Huang, Noah J. Apthorpe, Frank Li, Gunes Acar, and Nick Feamster. 2020. IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 2 (2020), 46:1–46:21.

[32] Ahmed Mohamed Hussain, Gabriele Oligeri, and Thiemo Voigt. 2020. The Dark (and Bright) Side of IoT: Attacks and Countermeasures for Identifying Smart Home Devices and Services. In *Security, Privacy, and Anonymity in Computation, Communication, and Storage - SpaCCS 2020 International Workshops, Nanjing, China, December 18-20, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12383)*, Guojun Wang, Bing Chen, Wei Li, Roberto Di Pietro, Xuefeng Yan, and Hao Han (Eds.). Springer, 122–136.

[33] S. Jeong and J. Griffiths. 2019. Hundreds of south korean motel guests were secretly filmed and live-streamed online. https://www.cnn.com/2019/03/20/asia/south-korea-hotel-spy-cam-intl/index.html.

[34] Jinlei Li, Yan Meng, Lu Zhou, and Haojin Zhu. 2020. Securing App Behaviors in Smart Home: A Human-App Interaction Perspective. In *Proceedings of 26th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2020,*.

[35] Xiang Li, Daqing Zhang, Qin Lv, Jie Xiong, Shengjie Li, Yue Zhang, and Hong Mei. 2017. IndoTrack: Device-Free Indoor Human Tracking with Commodity Wi-Fi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3 (2017), 72:1–72:22.

[36] Zhengxiong Li, Zhuolin Yang, Chen Song, Changzhi Li, Zhengyu Peng, and Wenyao Xu. 2018. E-Eye: Hidden Electronics Recognition through mmWave Nonlinear Effects. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems, SenSys 2018, Shenzhen, China.*

[37] Miao Lin, Vincent W. Zheng, and Shili Xiang. 2018. Sequential context modeling for smart devices by Collaborative Hidden Markov Model. In *Proceedings of 4th IEEE World Forum on Internet of Things, WF-IoT 2018, Singapore.*

[38] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. 2018. Detecting Wireless Spy Cameras Via Stimulating and Probing. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2018, Munich, Germany.*

[39] Knud Lasse Lueth. 2018. State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[40] Xiaobo Ma, Jian Qu, Jianfeng Li, John C. S. Lui, Zhenhua Li, and Xiaohong Guan. 2020. Pinpointing Hidden IoT Devices via Spatial-temporal Traffic Fingerprinting. In *Proceedings of 39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada.*

[41] Gonçalo Marques and Rui Pitarma. 2020. Enabling Smart Homes Through Health Informatics and Internet of Things for Enhanced Living Environments. In *Proceedings of Trends and Innovations in Information Systems and Technologies - Volume 3, WorldCIST 2020, Budva, Montenegro, 7-10 April 2020 (Advances in Intelligent Systems and Computing).*

[42] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *Proceedings of 37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA,*, Kisung Lee and Ling Liu (Eds.).

[43] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[44] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. 2006. Inferring Internet denial-of-service activity. *ACM Trans. Comput. Syst.* 24, 2 (2006), 115–139.

[45] Mustafa A. Mustafa, Alexandros Konios, and Matias Garcia-Constantino. 2021. IoT-Based Activities of Daily Living for Abnormal Behavior Detection: Privacy Issues and Potential Countermeasures. *IEEE Internet Things Mag.* 4, 3 (2021), 90–95.

[46] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. 2004. LANDMARC: Indoor Location Sensing Using Active RFID. *Wirel. Networks* 10, 6 (2004), 701–710.

[47] Haitham Noman, Sinan Noman, and Qusay Al-Maatouk. 2020. WIRELESS SECURITY IN MALAYSIA: A SURVEY PAPER. 7 (02 2020), 310–312. https://doi.org/10.31838/jcr.07.04.57

[48] Haitham Ameen Noman, Sinan Ameen Noman, and Qusay Al-Maatouk. 2019. Wireless Security In Malaysia: A Survey Paper. *Journal Of Critical Reviews* 7, 4 (2019), 2020.

[49] T. J. OConnor, Reham Mohamed, Markus Miettinen, William Enck, Bradley Reaves, and Ahmad-Reza Sadeghi. 2019. HomeSnitch: behavior transparency and control for smart home IoT devices. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2019, Miami, Florida, USA.*

[50] Jorge Ortiz, Catherine H. Crawford, and Franck Le. 2019. DeviceMien: network device behavior modeling for identifying unknown IoT devices. In *Proceedings of Proceedings of the International Conference on Internet of Things Design and Implementation, IoTDI 2019.*

[51] Jingjing Ren, Daniel J. Dubois, David R. Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. 2019. Information Exposure From Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In *Proceedings of the Internet Measurement Conference, IMC 2019, Amsterdam, The Netherlands.*

[52] Ignacio Sanchez, Riccardo Satta, Igor Nai Fovino, Gianmarco Baldini, Gary Steri, David Shaw, and Andrea Ciardulli. 2014. Privacy leakages in Smart Home wireless technologies. In *International Carnahan Conference on Security Technology, ICCST 2014, Rome, Italy, October 13-16, 2014.* 1–6.

[53] Amit Kumar Sikder, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2019. Aegis: a context-aware security framework for smart home systems. In *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA.*

[54] Akash Deep Singh, Luis Garcia, Joseph Noor, and Mani B. Srivastava. 2021. I Always Feel Like Somebody's Sensing Me! A Framework to Detect, Identify, and Localize Clandestine Wireless Sensors. In *Proceedings of 30th USENIX Security Symposium, USENIX Security 2021.*

[55] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2019. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Trans. Mob. Comput.* 18, 8 (2019), 1745–1759.

[56] Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings (Lecture Notes in Computer Science, Vol. 1057)*, Peter M. G. Apers, Mokrane Bouzeghoub, and Georges Gardarin (Eds.).

[57] Vijay Srinivasan, John A. Stankovic, and Kamin Whitehouse. 2008. Protecting your daily in-home activity information from a wireless snooping attack. In *Proceedings of UbiComp 2008: Ubiquitous Computing, 10th International Conference, UbiComp 2008, Seoul, Korea (ACM International Conference Proceeding Series)*.

[58] Rahmadi Trimananda, Janus Varmarken, Athina Markopoulou, and Brian Demsky. 2020. Packet-Level Signatures for Smart Home Devices. In *Proceedings of 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*.

[59] Deepak Vasisht, Anubhav Jain, Chen-Yu Hsu, Zachary Kabelac, and Dina Katabi. 2018. Duet: Estimating User Position and Identity in Smart Homes Using Intermittent and Incomplete RF-Data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2 (2018), 84:1–84:21.

[60] Ju Wang, Hongbo Jiang, Jie Xiong, Kyle Jamieson, Xiaojiang Chen, Dingyi Fang, and Binbin Xie. 2016. LiFS: low human-effort, device-free localization with fine-grained subcarrier information. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, MobiCom 2016, New York City, NY, USA*.

[61] wigle. 2022. Statistics of Wireless Encryption. https://wigle.net/stats#mainstats.

[62] et al. Y. Kang, X. Li (Ed.). 2020. *China Population Census Yearbook 2020*. China Statistics Press.

[63] Lei Yang, Yekui Chen, Xiang-Yang Li, Chaowei Xiao, Mo Li, and Yunhao Liu. 2014. Tagoram: real-time tracking of mobile RFID tags to high precision using COTS devices. In *Proceedings of The 20th Annual International Conference on Mobile Computing and Networking, MobiCom'14, Maui, HI, USA*.

[64] Lingjing Yu, Bo Luo, Jun Ma, Zhaoyu Zhou, and Qingyun Liu. 2020. You Are What You Broadcast: Identification of Mobile and IoT Devices from (Public) WiFi. In *Proceedings of 29th USENIX Security Symposium, USENIX Security 2020*, Srdjan Capkun and Franziska Roesner (Eds.).

[65] Mohammed Javeed Zaki. 2001. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Mach. Learn.* 42, 1/2 (2001), 31–60.

[66] Gergely V. Záruba, Manfred Huber, Farhad Kamangar, and Imrich Chlamtac. 2007. Indoor location tracking using RSSI readings from a single Wi-Fi access point. *Wirel. Networks* 13, 2 (2007), 221–235.

[67] Jiangfeng Zeng, Xiao Ma, and Ke Zhou. 2019. Enhancing Attention-Based LSTM With Position Context for Aspect-Level Sentiment Classification. *IEEE Access* 7 (2019), 20462–20471.

[68] Yifan Zhang. 2022. Top-selling Products: China's Smart Home Industry in 2022. https://equalocean.com/analysis/2022062318323.

[69] Yanzi Zhu, Zhujun Xiao, Yuxin Chen, Zhijing Li, Max Liu, Ben Y. Zhao, and Heather Zheng. 2020. Et Tu Alexa? When Commodity WiFi Devices Turn into Adversarial Motion Sensors. In *Proceedings of 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA*.

A DEVICE INFORMATION

Table 6. IoT device information

No.	Device name	Category	Vendor	No.	Device name	Category	Vendor
1	AC_Plug	Plug	Xiaomi	16	deerma-humidifier	Appliance	Deerma
2	aircleaner	Appliance	Xiaomi	17	honyar-outlet	Appliance	Honyar
3	audio	Speaker	Baidu	18	plug-LED	Appliance	Xiaomi
4	bedlight	Appliance	Xiaomi	19	standheater	Appliance	Xiaomi
5	bathheater	Appliance	Xiaomi	20	sweeper	Appliance	Xiaomi
6	camera-battery	Camera	Hichip Vision	21	mercury-camera	Camera	Mercury
7	curtains	Appliance	Xiaomi	22	echodot	Speaker	Echo
8	water-cooler	Appliance	Xiaomi	23	echoplus	Speaker	Echo
9	skyworth-camera	Camera	Skyworth	24	echospot	Speaker	Echo
10	deerma-dehumidifier	Appliance	Deerma	25	bulb	Appliance	TP-LINK
11	desklight	Appliance	Xiaomi	26	tp-link_plug	Plug	TP-LINK
12	fan	Appliance	Xiaomi	27	xiaomi-hub	Hub	Xiaomi
13	gateway-locker	Gateway	Xiaomi	28	ricecooker	Appliance	Xiaomi
14	aqara-gateway	Gateway	Aqara	29	strip	Appliance	Xiaomi
15	mijia-humidifier	Appliance	Xiaomi				

B DEVICE ACTIVITY INFORMATION

Table 7. Common activities of devices from the mijia platform

Device	Via app	Physically	Via automation rules
AC_Plug	①/② turn on/off, ③ change temperature, ④ change speed.	\	⑤/⑥ turn on/off.
fan	⑦/⑧ turn on/off, ⑨ change mode.	⑩/⑪ turn on/off.	⑫/⑬ turn on/off.
water-cooler	\	⑭/⑮ turn on/off.	\
mijia-gateway-locker	\	⑯ open.	\
bedlight	⑰/⑱ turn on/off , ⑲/⑳ change light/mode,	㉑/㉒ turn on/off, ㉓/㉔ change light/mode.	㉕/㉖ turn on/off.
mijia-humidifier	㉗/㉘ turn on/off, ㉙ adjust gear.	㉚/㉛ turn on/off.	㉜/㉝ turn on/off.
bathheater	㉞/㉟ turn light on/off, ㊱/㊲ turn ventilation on/off.	\	\
deerma-humidifier	㊳/㊴ turn on/off, ㊵ change mode.	\	㊶/㊷ turn on/off.
curtains	㊸/㊹ open/close.	\	㊺/㊻ open/close.
plug-LED	㊼/㊽ turn on/off.	\	㊾/㊿ turn on/off.
aircleaner	(51)/(52) turn on/off , (53) change mode.	\	(54)/(55) turn on/off.
standheater	(56)/(57) turn on/off, (58) change mode.	(59)/(60) turn on/off.	(61)/(62) turn on/off.
dehumidifier	(63)/(64) turn on/off, (65) change mode.	\	(66)/(67) turn on/off.
sweeper	(68)/(69) sweep/return.	\	(70)/(71) sweep/return.
desklight	(72)/(73) turn on/off , (74) change mode.	(75)/(76) turn on/off, (77) change light.	(78)/(79) turn on/off.
echodot	(80) audio, (81) voice	(82)/(83) power.	\
echoplus	(84)/(85) turn on/off, (86) audio, (87) dim.	(88) voice, (89) volume, (90) power.	\
echospot	(91) audio.	(92) voice, (93) volume, (94) power.	\