Hindawi
Security and Communication Networks
Volume 2022, Article ID 5655314, 10 pages
https://doi.org/10.1155/2022/5655314

Research Article

Detecting Privacy Leakage of Smart Home Devices through Traffic Analysis

Ting Yang,[1,2] **Guanghua Zhang**,[3] **Yin Li**,[2] **Yiyu Yang**,[2] **He Wang**,[1] **and Yuqing Zhang**[1,2,4,5]

[1]*School of Cyber Engineering, Xidian University, Xian 710000, China*
[2]*National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing 101408, China*
[3]*School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang 050091, China*
[4]*School of Computer Science and Cyberspace Security, Hainan University, Haikou 570228, China*
[5]*School of Information Science and Engineering, Yanshan University, Qinghuangdao 066000, China*

Correspondence should be addressed to He Wang; hewang@xidian.edu.cn and Yuqing Zhang; zhangyq@ucas.ac.cn

Received 10 May 2022; Accepted 14 June 2022; Published 15 July 2022

Academic Editor: Shui Yu

Under the management of the Internet of Things platform, smart home devices can be operated remotely by users and greatly facilitate people's life. Currently, smart home devices have been widely accepted by consumers, and the number of smart home devices is rising rapidly. The increase of smart home devices introduces various security hazards to users. Smart home devices are vulnerable to side-channel attacks based on network traffic. The event of smart home devices can be identified by network surveillants. Given this situation, we designed a set of standardized workflows for traffic capturing, fingerprint feature extraction, and fingerprint event detection. Based on such workflow, we present IoTEvent, a semiautomatic tool to detect vulnerable smart home devices, which is not limited to specific types of communication protocols. IoTEvent first collects device traffic by simulating touch events for App. Then, it pairs the packet sequences with events and generates a signature file. We also test the usability and performance of IoTEvent on five cloud platforms of smart home devices. Finally, we discuss the reasons for privacy leakage of smart home devices and security countermeasures.

1. Introduction

With the progress of communication technology and network technology, the smart home market has developed rapidly. According to ABI research [1], almost 79 million homes will have a smart home device by 2024. The manufacturers of smart home devices are trying to make them "smart" by connecting them to the cloud platform, and then users can control them using mobile Apps or voice assistants. For example, if we want to listen music and say "Hey Google, play music," a piece of wonderful music will be played by Google Home. Nowadays, smart home platforms are widely used in the process of device development, including Xiaomi [2] and Huawei [3].

However, the rapid growth of the market economy promotes the development of the manufacturing industry toward product practicality and then ignores the safety of products, leading to a number of security vulnerabilities for smart home devices [4]. According to reports, Amazon's Alexa and Google's smart speakers can eavesdrop on users' information and even cheat them by voice. According to researchers, few consumers are aware that smart home devices collect and share private data as part of their normal operations [5]. Smart home cloud platform not only brings convenience to people but also has the risk of privacy leakage [6]. Large amounts of data from devices are collected by cloud platforms, which are transmitted through network traffic. For example, users can control the Mijia App to turn on a smart light bulb. In this process, the smart home cloud platform judges and recognizes the commands sent by the App and then forwards them to the smart light bulb. Although the control commands are encrypted, the commands

can be identified by encrypting traffic analysis. Furthermore, detecting the detailed activities of smart home devices is the premise of implementing attacks against smart home scenarios.

Many researchers [7–10] have noted that there is a link between encryption traffic and device state transitions, but their proposed method cannot accurately detect triggered events. A part of studies [11, 12] analyzed network traffic for specific types of devices. The author of [11] analyzed network traffic of the camera. The author of [12] detected users' presence through traffic analysis of smart speakers. The author of [9] inferred device state transitions through Zigbee [13]/Z-wave [14] network traffic. The author of [15] proposed the network traffic packet signature of TCP-based devices. However, their tool cannot apply to UDP-based devices that follow a connectionless pattern.

In this article, we present IoTEvent to identify trigger events of smart home devices by fingerprinting encrypted traffic. Note that the user's behavior privacy can be inferred from the detection result. The main contributions of this study are summarized as follows:

(i) Design a trigger event detection tool IoTEvent. First, collect encrypted traffic of smart home devices. Then, train the signature event and generate classification model. Finally, based on the model detect the privacy events.

(ii) Test our tool with nine popular devices from five popular cloud platforms. We observe a high accuracy of 99%.

(iii) Analyze the reasons for privacy leakage of encrypted traffic, and then present some suggestions on how to deal with this problem.

This article is organized as follows. The threat model is described in Section 2. Then, we detail the design and implementation in Section 3 and present the evaluation results of devices in Section 4. In addition, we discuss the reason of the privacy leakage of smart home devices and how to avoid it in Section 5. In Section 6, we introduce related works. Finally, we conclude in Section 7.

2. Threat Model

There are four key entities in the process of communication: the smart home devices, the mobile App, the cloud, and the gateway, as shown in Figure 1. The privacy information can be leaked in the communication process by analyzing the timestamp, length, and direction of the data packet. We assume the adversary has two attack methods: WAN sniffing and WiFi sniffing, and knows the brand and model of the device that he wishes to passively monitor (the adversary may be the neighbor or the repairman that has been to the victim's house). Through WiFi sniffing, the adversary can know the MAC addresses to identify which device has sent the traffic. Normally, home routers use NAT [16]: remapping all traffic to the router's IP address. Through WAN sniffing, the adversary can know IP headers of all packets and then find the domain name that communicates with the device by IP address.

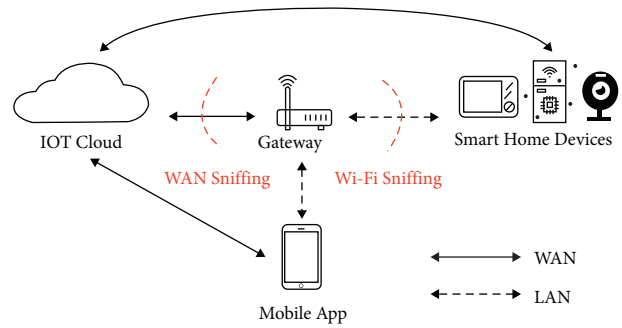

Figure 1: The communication modes of the smart home system. The LAN is the main communication mode between smart home devices and the cloud, and the WAN is rarely used. Mobile App control devices have two paths: when the user is at home, control commands are uploaded to the cloud via the LAN; when the user is not at home, control commands are uploaded to the cloud over the WAN.

To sum up, the adversary should meet the following conditions:

(i) The WiFi sniffer should be within the transmission distance of the wireless router. The WAN sniffer can capture all packets between the router and the cloud.

(ii) The adversary can obtain the same smart home device and extract event signatures of the device

Smart home devices transfer packets to the router through the WiFi, and then the router transfers them to the cloud through the WAN. Before sniffing, the adversary needs to get the event's signature of the same device. During sniffing, the adversary cannot obtain the plaintext data because the data are encrypted and then transmitted. After capturing the packets, the adversary first preprocesses the packets and then performs event detection based on the event's signature of the same device.

3. Detailed Design

In this section, we present each step of IoTEvent and explain how to detect the triggered events of the device in detail.

3.1. Overview. For identifying the privacy events, we present IoTEvent to handle the challenges. Our tool can identify event containing out-of-order packets, which improve the accuracy of event detection. Figure 2 shows the workflow of IoTEvent. In the first step of network traffic collection, IoTEvent can simulate trigger events through scripts, which also record the name and timestamp of the triggered event. After receiving the control command, the device generates packet sequences in a short time. We use tcpdump to capture device traffic on the router. After this step, we collect traffic for different events to prepare for event signatures and data training. Next, in the step of data training, due to device events generating data packets in a short time, we propose a method to divide device traffic into packet sequences. Then,

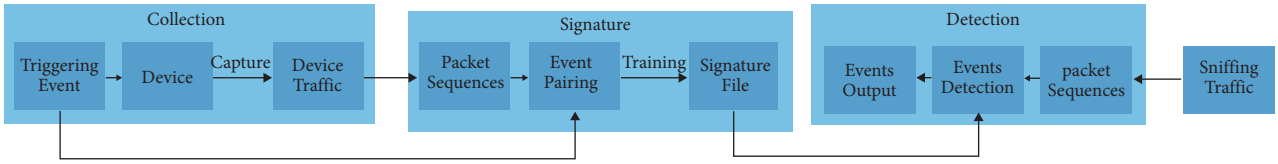

Figure 2: System architecture of IoTEvent.

IoTEvent pairs events and packet sequences through timestamp. Next, IoTEvent extracts the length and direction of the data packet to form an event signature. Through training, a signature file is generated to store event's signature. In the last step of event detection, the sniffed traffic of the device is divided into packet sequences. The main task of event detection is to classify the packet sequences according to the signature file. Finally, the event's name is output, which is also the result of classification.

3.2. Traffic Capturing. The main task of data collection is to capture the network traffic of the device and obtain the timestamp of the triggered events.

According to the survey, we find that the traffic between the device and the cloud includes the following three types: device heartbeat package, device report information, and device operation command. The device heartbeat packet is used to detect whether the device is disconnected. The cloud platform uses a simple communication packet to judge whether the device is running normally. If no response is received from the device within a specified period of time, the device will be judged to have dropped the line. The device report information includes device firmware information, device log information, and device physical state change information. The device operation command refers to the control command sent by the user to the device through the cloud.

The triggered events refer to the state change of the device by the user clicking or sliding the corresponding mobile App. For example, turn on the light by clicking button of the mobile App, as shown in Figure 3. The mobile App sends the control command of light turn on to the cloud. After the cloud receives the control command, it queries the status of the device. The cloud compares it with the status of the device in the command. If they are different, the cloud sends command of light turn on. The device executes the command and returns success message. If they are the same, the cloud does not send the command. Different device events may have different response processes, so the number of data packets generated during the event will also be different.

IoTEvent automatically generates and captures device traffic for the training set. We use the script of Auto.js [17] to trigger all events on the smartphone's screen in turn, which runs on smartphone. All events refer to operations related to the device on the mobile App [18]. For example, in addition to basic control events (light on/off), Mi Control Gateway also includes setting events (volume settings). What we want is to trigger all device-related events on the App. The script is customized according to the device due to the different

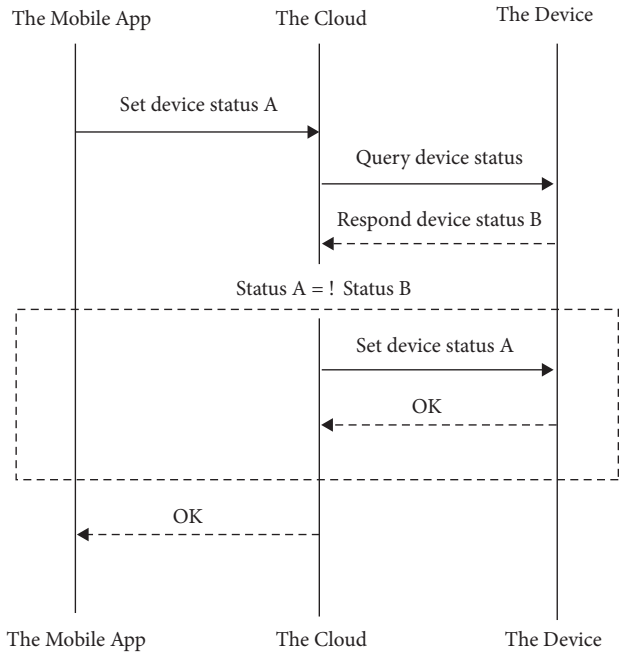

Figure 3: The device event and respond. The query (solid line) and the reply (solid line) of the event are shown in the figure.

functions of the device. Each event is triggered 100 times, and the interval between two clicks is 60 seconds. IoTEvent starts tcpdump to capture the device traffic before triggering the events. The network traffic includes all data packets between the device and the cloud platform, which are the result of event triggering, device logs, and so on. IoTEvent records the network traffic in a PCAP file and writes the name and timestamp of the trigger event in a text file.

3.3. Fingerprint Feature Extraction. The purpose of fingerprint feature is to generate a signature file of the device. IoTEvent divides the captured traffic into packet sequence and then matches the packet sequence and event according to the algorithm. The length and direction information of the data packet are extracted to generate feature vectors, which are trained to generate a classification model.

Due to the difference of SDK between cloud platforms or the difference of the communication protocol of the same cloud platform, the encrypted traffic generated by smart home devices will be different. At present, many scholars have studied the identification of devices on different cloud platforms and made great progress, which we will introduce in related work. Smart home devices will send heartbeat packets to the cloud platform to keep connected at a certain interval, packet number, and length. When we use the

mobile App to control the device, the encrypted traffic of different control commands of some smart home devices is different in terms of traffic rate, packet size, and so on. Based on these differences, we design event signatures to distinguish the control commands.

3.3.1. Packet Sequence. IoTEvent extracts the timestamp from the packet p of the device to obtain the time interval Δt between two packets. When the network is under the good condition, the data sending and receiving rate will be accelerated in a triggered event, so the time interval Δt is smaller. Therefore, IoTEvent judges the number of event according to Δt. If $\Delta t > a$, the packet is marked. Otherwise continue. Based on the recorded data, IoTEvent divides device traffic into packet sequences and then filters out the packet sequences generated by the trigger event.

3.3.2. Event Pairing. Event pairing is to match the event name with the packet sequence. Each packet sequence represents a triggered event, $E_i^n = p_1 p_2 p_3 \cdots p_n$, n is the number of packets, and i is the event name, which is an unknown message. IoTEvent compares the timestamp of the first packet in the packet sequences and the triggered event. If the time interval Δt is less than b, the event name and packet sequence are matched. In smart home devices, the events between device and the cloud are limited. Simple devices (such as smart sockets and smart light bulbs) have relatively few events, while complex devices (such as smart gateways) have more events.

IoTEvent extracts information from packet p, and each packet corresponds to a five-tuple, $p_n = (t_n, \Delta t_n, \mathrm{IP}_d, \mathrm{IP}_s, l)$, t_n is timestamp of the packet, Δt_n is the time interval between this packet and the previous packet, IP_d is the destination IP address, IP_s is the source IP address, and l is the length of packet. We define $d = 1$ for the data package from the cloud to the device, and $d = -1$ for the data package from the device to the cloud. The event can be represented as a $1 \times n$ vector, $E_i^n = [d_1 \times l_1, d_2 \times l_2 \cdots d_n \times l_n]i$ is the name of event, which is also the label of the data set, n is the packet number of event, and the n of the same event may be different due to the existence of disordered data packets (e.g., device log and device network information).

3.3.3. Signature File. The signature file is a classification model for event detection. IoTEvent groups events E_i^n according to the number of packets n, as shown in formula (1). In network traffic, query and reply packets exist in pairs, so the number of packets n of events is even. Generally, the query and reply packet pair of the trigger event is less than 15, so $\max(n) \approx 30$ in

$$\sum_{n=2}^{\max(n)} E_i^n = \left\{E_i^2, \ldots, E_i^{\max(n)}\right\}, n = 2a\left(a \in N^+\right). \quad (1)$$

After the events are grouped, each group is a data set. IoTEvent uses the kNN (K-nearest neighbor) algorithm [19] to train a classification model, which is a supervised learning method for modeling or predicting discrete random variables. The goal is to learn a classification function or model from the training sample data set with known labels, which is also known as a classifier. When getting new data, the new data item can be predicted based on the classifier, and the new data item can be mapped to a class in a given category. In terms of classification, the input training data contain the following information: feature, attribute, label, or class, which can be used to represent $(F1, F2, \ldots FN; \text{label})$. The essence of research is to find out the relationship between features and markers (i.e., mapping). The hierarchical prediction model is to map input variables (attributes) and discrete output variables (categories). In this way, if the unknown data have no label, the unknown data can be predicted by the mapping function. kNN is a case-based classification algorithm. By calculating the distance between the different eigenvalues of the test object and the sample, the label of K adjacent samples is selected as the result.

IoTEvent uses 80% of the generated sample set as the training set to train the model, and the remaining 20% data set is used as the test set. The effect of the detection model is directly judged through the test data. In addition, use the test set to improve the model before it is used to detect. According to the result of event pairing, the product of the length and direction of each packet is taken as a value of the feature vector, the packet number of event is the length of the vector, and the event name is the label. IoTEvent generates the training set S and test set T, E_i is an event in the training set, and E_j is an event in the test set. The distance between two events is calculated in

$$d_{ij} = \left|E_i^n - E_j^n\right| = \sqrt{\sum_{i=1}^{n} \left(x_i - y_i\right)^2}. \quad (2)$$

IoTEvent selects K events in S with the smaller distance d_{ij} as the nearer neighbors of E_j. IoTEvent returns the event name of the most frequent occurrence of the K events as the result. IoTEvent compares test set labels and results to calculate the detection accuracy. Through training, the classification model is optimized to improve the accuracy of event detection. The value of K is related to the number of samples in the training set. IoTEvent chooses the best classification accuracy by comparing different K values. After training, all samples are recorded in the signature file as the classification model of detection.

3.4. Fingerprint Event Detection. The process of event detection is to identify device events in the sniffed traffic. We analyze the process of traffic capture from the perspective of the adversary. IoTEvent divides traffic into packet sequences, then recognizes events based on the signature file, and finally generates event output.

3.4.1. Sniffing Traffic. The adversary has two ways to sniff traffic: WiFi sniffing and WAN sniffing. Through WiFi sniffing, the adversary can capture the data link layer traffic and identify the device through the MAC address. The data

transmitted by WiFi are encrypted. The adversary should filter irrelevant data packets (e.g., 802.11 CTS, 802.11 Frag) and only retain 802.11 encrypted data. Through the WAN sniffing, the adversary can capture the network layer traffic and identify the source IP address and destination IP address, one of which is the router IP and the other is the cloud IP. The adversary can find the domain name of the IP address through IP reverse DNS. Based on the domain name, the adversary can identify whether the packet is between the device and the cloud.

3.4.2. Event Detection. Event detection is to identify the event of the packet sequence and determine whether the event is a privacy event. The private event refers to the event that we can obtain the privacy of the user's behavior, so privacy events include device physical state change information and device operation command. For example, the motion sensor reports movement.

In order to identify privacy events, we define the name of all events set on a certain device as ϕ, the name of private events set as α, and the name of nonprivate events set as β, so $\phi = \alpha + \beta$. We can define all the events E_ϕ as shown in formula (3), among which E_α^k represents the privacy event with k packets.

$$E_\phi = \sum_{k=1}^{\max(k)} E_\alpha^k + E_\beta^k. \tag{3}$$

Event detection is a concrete realization of privacy event identification for encrypted traffic based on event signature. IoTEvent uses the trained kNN model to achieve event detection. The names of all privacy events are put into a set α, and when the privacy event name is detected, an alarm will be raised. The implementation of event detection is shown as follows.

(1) IoTEvent extracts the information from the packet, $p_n = (t_n, \Delta t_n, \mathrm{IP}_d, \mathrm{IP}_s, l)$. Each packet sequence is an event to be detected. The packet sequence can be represented by a $1 \times n$ vector, $E_j^n = [d_1 \times l_1, d_2 \times l_2 \cdots d_n \times l_n]$, and the j is event name, which need to be identified.

(2) Measure the distance. According to the number n of event E_j^n, IoTEvent screens out the events $\sum E_i^n$ with the number n in the signature file and then calculates the distance d_i between the events E_j^n and the events $E_{\mathrm{i}}^{\mathrm{n}}$ in signature file, $d_{ij} = |E_i^n - E_j^n|$.

(3) Select K events with the smaller distance d_{ij} as the nearer neighbors of E_j^n. IoTEvent returns the event name of the most points in the K neighborhood as the result. If it belongs to privacy events set α, an alert is issued; otherwise no alert is issued.

4. Evaluation

In this section, we evaluated the effectiveness of IoTEvent. Specifically, we conducted experiments on nine smart home devices which use five different cloud platforms. We described how the experiment was set up in and then presented the experiment result.

Table 1: List of smart home devices.

The cloud	Vendor	Device name	Device model
Xiaomi	Yeelight	Yeelight light strip	YLDD04YL
Xiaomi	Chuangmi	Mi plug mini	ZNCZ02CM
Xiaomi	Lumi	Mi control hub	DGNWG02LM
Tuya	Tuya	WiFi lamp	2AJ3WABEQPZ05
Tuya	Hongshi	WiFi plug	F2s501-GB
Tp-Link	Tp-Link	Smart WiFi plug	HS100
Hicloud	ORVIBO	Smart socket	S30c
Hicloud	Jellyfish Tur	LED light bulb	BRO-16565
JD	BroadLink	Smart plug	SP mini 3

4.1. Experiment Setup. Before the experiment, we need to complete the construction of the experimental environment. We used mobile phones to control devices, which model is SEN-AL00 and the system is Android 10.0. The encrypted traffic capturing and sniffing were conducted by the sniffer we built. In particular, the traffic capture used the wireless network card, of which the interface is USB 3.0, the speed is 300–866 Mbps, and the wireless standard is IEEE 802.11ac/a/b/g/n. The traffic detection was conducted on a Windows 10 equipped with a Intel Core i7-10710U CPU.

Table 1 provides the smart home devices of the experiments. The first column shows the cloud platforms used by the devices. The second column presents the device vendor. The third column is the device name. The fourth is the device model. IoTEvent runs with a local Python 3.6 programming environment in Windows 10, which achieves traffic capture, training, and event detection of the devices.

4.2. Experiment Results

4.2.1. Data Collection. During the experiment, the protocol of devices is given in Table 2, which includes TLS, TCP, UDP, and MQTT. In addition, we recorded the domain name in Table 2 used by the cloud to send control commands. We found that the state of smart home devices is limited. When the smart home device connects to the cloud platform for the first time, it should report the device hardware and network information, which we call device connection events. In normal operation, the limited state of smart home devices is converted to each other. In general, the events of smart gateway mainly include heartbeat package, device log, and subdevice information, while the gateway with additional functions has relatively more events. Moreover, the number of smart gateway events also depends on the number of connected subdevices. The subdevice reports the heartbeat package and the status change information through the smart gateway. Therefore, the more subdevices the smart gateway connects to, the more events there are.

4.2.2. Training. The manual analysis indicates that the packet sequences of different events are different. In addition

Table 2: Summary of experimental results.

The cloud	Device name	Protocol	The domain name of command	Trigger event	WiFi sniffing Accuracy (%)	WAN sniffing Accuracy (%)
Xiaomi	Yeelight light strip plus	TLS	https://ots.io.mi.com	Turn on Turn off Color Color setting Flow Flow setting Schedules Timer Favorites	100	100
	Mi plug mini	TLS	https://ots.io.mi.com	Power on/off Set time on/off Schedules	100	100
	Mi control hub	UDP	https://ots.io.mi.com	Alert enabled/disabled Light turn on/turn off Colored lamp color Colored lamp brightness Scene color of colored lamp Timed alert Alert trigger device Timer colored lamp Add child device Snooze alarm clock Doorbell trigger device Delay effective time Volume settings Language of voice prompt Network radio Hub alert ringtone Alert volume Alert red light blink time Alert time Linkage alert	98.54	98.88
Tuya	WiFi lamp	MQTT	https://mq.gw.tuyancn.com	Light turn on/turn off Turn on/off	100	100
	WiFi plug	TLS	https://m2.tuyacn.com	Schedules Set time on/off Electricity consumption	100	100
Tp-Link	Smart WiFi plug	TLS	https://use1-api.tplinkra.com	Turn on/off Scheduling on/off Set time on/off Away mode on/off	98.75	98.82
Hicloud	Smart socket	TCP	https://iomplatform.hicloud.com	Turn on/off Timer on/off Delay on/off	99.60	99.54
	LED light bulb	TCP	https://iomplatform.hicloud.com	Turn on/off Fast on Slow on Timer Delay	100	100
JD	Smart plug	TCP	https://live.smart.jd.com	Turn on/off Set time on/off	100	100

to network traffic of triggerable events, the device also generates other network traffic (e.g., device heartbeat, device logs).

For example, Mi control hub is a smart home device that connects to Mi Home. We use it as an illustrated example and manually analyze its network traffic. We get all events that can be triggered from Mi Home App, as given in Table 3, which relate to user privacy marked in gray.

We connected the mobile phone and the device to the LAN. Then, we simulated the user clicking the button and

Table 3: All events of Mi Hub.

The smart home devices	The events
Mi control hub	Alert enabled/disabled
	Light turn on/turn off
	Colored lamp color
	Colored lamp brightness
	Scene color of colored lamp
	Timed alert
	Alert trigger device
	Timer colored lamp
	Add child device
	Snooze alarm clock
	Doorbell trigger device
	Delay effective time
	Volume settings
	Language of voice prompt
	Network radio
	Hub alert ringtone
	Alert volume
	Alert red light blink time
	Alert time
	Linkage alert

recorded the timestamp of the click. At the same time, we used tcpdump to capture network traffic of the device. Through manual analysis of network traffic, we found the packet sequences are different for each event, and the packet sequences of three different events are shown in Figure 4. For the events light turn on/turn off, the packet sequences are the same. We observed an exchange of UDP application data packets between the Mi Home Hub and Internet host where the packet lengths were (138, 154), (106, 106). However, for the event alert enabled, the device sends UDP packets of lengths (154, 186) to an Internet host and receives reply packets of lengths (106, 106). Similarly for the event of alert disabled, these packets were of lengths (170, 138, 170), (106, 106, 106).

All trigger events of different devices are given in Table 2. Except for trigger events, other device events (e.g., heartbeat packet and device log) are signed as other events. Not all trigger events have traffic between the device and the cloud platform, and part events are recorded in the cloud and are triggered when the set conditions are reached. IoTEvent only recorded trigger events that generate traffic. In our experiment, the kNN algorithm is used. The average precision for the different k is shown in Figure 5.

4.2.3. Event Detection. Among the devices we tested were smart gateways and cloud-connected devices. Different types of devices contain different types of privacy information, as given in Table 4. The privacy information contained in the smart gateway that we can detect by encrypting the traffic includes the connected subdevices, the subdevice sensor status, the operation commands, and the state change information. The privacy information contained in the cloud-connected devices that we can detect by encrypting the traffic includes the statue change information and the operational commands.

We simulated using the device, and each event was triggered ten times. We captured the traffic using WiFi sniffing and WAN sniffing, and then IoTEvent was used to detect. The definition of the accuracy is shown in formula (4). TP indicates the number of trigger events that are correctly detected, FN indicates the number of trigger events that are erroneously detected, TN indicates the number of nontrigger events that are correctly detected, and FP indicates the number of nontrigger events that are erroneously detected.

$$\text{Accuray} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4}$$

The result of event detection is given in Table 2. Most event signatures of the devices are unique, so the event would be detected. Through the detection results of events, we can infer the privacy of users' behaviors. For example, if the light is turned off at night, we can judge that users will go to sleep. When the mobile sensor detects someone moving, it can judge someone's activity in the house. Such privacy can be obtained only by monitoring network traffic, which greatly increases the risk of people's privacy leakage.

4.3. Comparison with the Existing Work. We selected two representative works for comparison: the [7] and the [15]. The comparison is mainly carried out from the following three aspects: the range of detection, the accuracy, and the computational overhead (Table 5).

IoTEvent is not designed for specific types of protocols. In our experiments, we proved that IoTEvent is suitable for TCP, UDP, TLS, and MQTT protocol. In the [15], their tool is only suitable for TCP protocol. In the [7], the author identified the event based on the traffic rate, thus which applied any protocol. IoTEvent can accurately detect the name of the event, and the average accuracy can reach 99%. In contrast, the [7] cannot accurately detect the occurrence of the event. The [15] cannot determine the event is unique, because it only clusters for a particular event. The event detection accuracy of [15] is 97%. Compared to [15], IoTEvent required more computation time and memory, but which is within a reasonable range. The [7] required more storage overhead and do not have computational overhead.

5. Discussion

According to the results of event detection, we analyzed the reason for privacy leakage, proposed several suggestions, and discussed the limitations.

5.1. The Reasons. Different cloud platforms have their own communication protocols, Huawei, Tuya, Xiaomi, and JD use custom protocols, so the protocol name is also named by themselves. Before the device development, the device developer should first select a cloud platform and develop the device according to the SDK provided by the cloud platform. Currently, the smart home cloud platform provides a variety of communication protocols in the SDK, including platform custom protocols, MQTT, Coap, Soap, Http, and Https. The experiment found that the device using the platform's custom protocol had a high probability of privacy leakage of

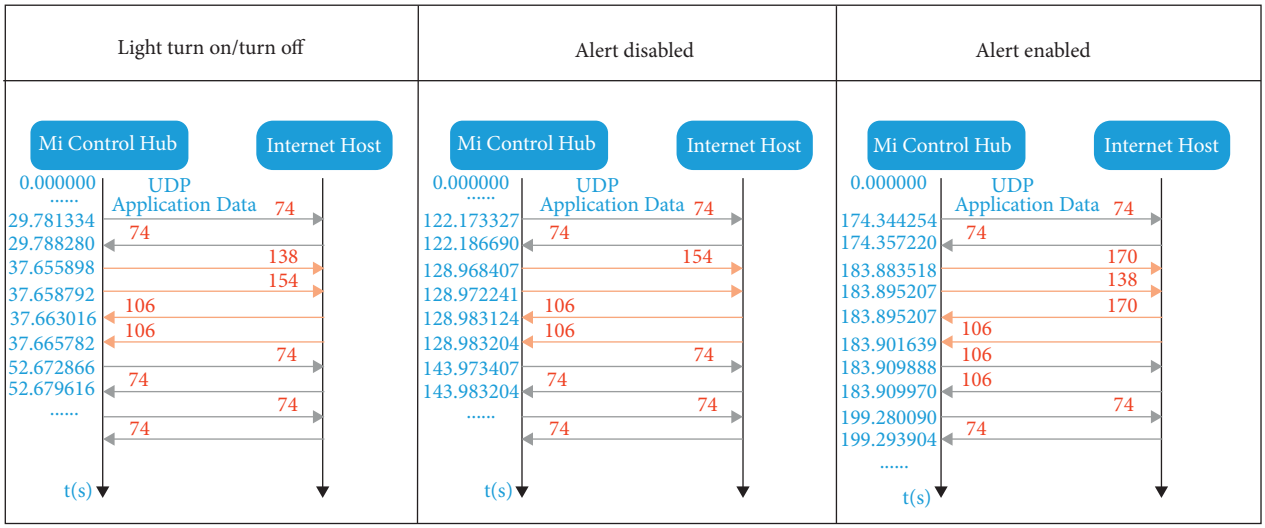

Figure 4: The packet sequences of three different events for Mi control hub: light turn on/turn off, alert enabled, alert disabled, the line of which is marked yellow. The UDP protocol is used to transmit application data in device and cloud communication. The arrow represents packet direction, and the number represents packet length.

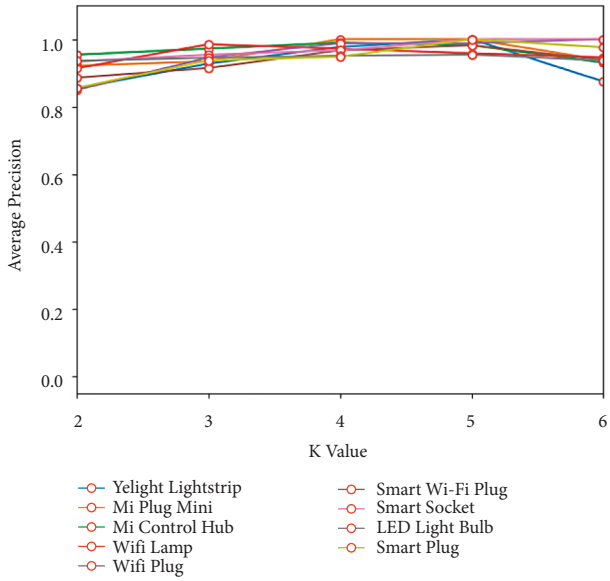

Figure 5: Average precision for the different k values.

Table 4: Type of device privacy.

Device type	Privacy information
The smart gateway	The connected subdevices The subdevice sensor status The operation commands The state change information
The cloud-connected devices	The operation commands The state change information

encrypted traffic. Taking the custom protocol of Xiaomi platform as an example, the Xiaomi cloud platform adopts AES-128-CBC [20] encryption, which requires (key, iv), which are both 16 bytes, and adopts UDP for data transmission. The plaintext must be a multiple of 16, and the ciphertext must be the same length as the completed plaintext. Any change of more than 16 bytes in the transmitted plaintext data will change the length of the transmitted encrypted packet. Moreover, the number of packets is different for different events of the platform custom protocol.

5.2. The Suggestion

5.2.1. Fill in a Random Information. For the custom protocol of smart home cloud platform, random information is added in the design process of the protocol so that the size of the packet is independent of the content. Adding a random information independent of the command to the command data sent can make the information size of the communication packet to change randomly, but it does not change the packet rate.

5.2.2. Change the Encryption Method. For the smart home cloud platform with privacy disclosure of encrypted traffic, the purpose of traffic shaping can be achieved by using encryption. If the number and size of all event packets are unified through encryption, privacy events cannot be detected, so as to alleviate the privacy disclosure of encrypted traffic.

5.2.3. Use VPN. VPN (virtual private network) connects the two LANs together and encrypts the transmission function to make the network more secure and confidential. VPN can also change the user's IP address, making it difficult for the device to be tracked. At the same time, it can also change the rate of communication packets, making it difficult to detect the privacy leakage of encrypted traffic.

Table 5: Comparison with existing works.

Comparison	Existing works		
	Apthorpe's work	Trimananda's work	Our work
The range	No restrictions	Only TCP	No restrictions
The accuracy	—	97%	99%
The computational overhead	High	Reasonable range	Reasonable range

6. Related Work

We review the related work on smart home security from two perspectives: the identification of smart home devices and the privacy leakage of smart home devices.

6.1. Events Identified for IoT Traffic. In recent years, there has been increasing attention paid to the privacy leakage problem of encrypted traffic. In [15], the author presents a tool that can automatically extract packet-level signatures for events from network traffic of TCP-based devices. In [11], an attack tool that infers the house status by inspecting the bit rate variation of the wireless camera traffic was proposed. In [9], the authors design an accurate and efficient smart spying strategy, which can infer a user's activity (such as web browsing, e-mail, and chat) from encrypted wireless traffic. In [10], the authors infer the state transition of smart home devices through Zigbee and Z-Wave encrypted traffic to detect eavesdrop or spoof events of smart home Apps. In [7], the authors examine four smart home devices and find that their network traffic rates can reveal potentially sensitive user interactions even when the traffic is encrypted. In [8], the authors check the action and state of smart home devices through network traffic time characteristics. In [12], the authors showcase risks of machine learning techniques to develop black-box models to automatically classify traffic and implement privacy leaking attacks. In [21], the authors investigate the privacy leakage of encrypted traffic from smart speakers.

6.2. Devices Identified for IoT Traffic. Before our work, we need to identify smart home devices through encrypted traffic. At present, many researchers have studied IoT device identification. The authors of [22] propose an acquisitional rule-based engine (ARE) that can automatically generate rules for discovery and annotation of IoT devices without any training data. The authors of [23] used the network traffic characteristics of IoT devices to train the machine learning model to detect the types of IoT devices. The authors of [24] used the small deviation in the hardware device to realize the device fingerprint so as to achieve the purpose of device identification. The authors of [25] proposed to detect chip-sets, firmware, or drivers by observing the response (or lacking of response) of 802.11 wireless devices to a series of nonstandard 802.11 frames. The authors of [26] proposed two device-type fingerprint identification methods to enhance the existing intrusion detection methods in the integrated circuit environment. The first method measures data response processing time and exploits the static and low-latency nature of the private industrial control system network to develop accurate fingerprints, while the second method uses physical operation time to develop unique signatures for each device type. These methods can accurately identify Internet of Things devices.

7. Conclusion

Nowadays, smart home devices have become an indispensable part of our life, yet deficiency in privacy protection will be an obstacle to its development [27]. In this article, we have proposed a tool to detect the privacy leakage of smart home devices from encrypted traffic. First, we present the threat model. Then, the tool for trigger event detection is designed. Finally, IoTEvent is evaluated with nine smart home devices on five cloud platforms. We are able to detect specific behaviors and actions from encrypted traffic. More specifically, we analyze the reasons for the privacy leakage of these devices on the cloud platform and put forward suggestions to alleviate this problem. Finally, we briefly introduce the related work. In our future work, we would like to extend our tool to more devices on different cloud platforms to make it applicable for a variety of environment.

Data Availability

The data used to support this study are available at https://github.com/yangting111/IoTEvents/tree/main.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (U1836210) and the Key Research and Development Science and Technology of Hainan Province (ZDYF202012).

References

[1] Abiresearch, "Digital Transformation Doesn't Have To Be Disruptive," 2020, https://www.abiresearch.com/.
[2] XiaoMi, "XiaoMi," 2020, https://iot.mi.com/new/index.html.
[3] Huawei, "Huawei," 2020, http://iot.hilink.huawei.com/.
[4] P. Morgner, C. Mai, N. Koschate-Fischer, F. Freiling, and Z. Benenson, "Security update labels: establishing economic incentives for security patching of iot consumer products," 2019, https://arxiv.org/abs/1906.11094.

[5] S. Zheng, N. Apthorpe, M. Chetty, and N. Feamster, "User perceptions of smart home iot privacy," *Proceedings of the ACM on Human-Computer Interaction*, CSCW, vol. 2, pp. 1–20, 2018.

[6] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[7] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: privacy vulnerabilities of encrypted iot traffic," 2017, https://arxiv.org/abs/1705.06805.

[8] A. Acar, H. Fereidooni, T. Abera et al., "Peek-a-boo: I see your smart home activities, even encrypted!," 2018, https://arxiv.org/abs/1808.02741.

[9] T. Hou, T. Wang, Z. Lu, and Y. Liu, "Smart spying via deep learning: inferring your activities from encrypted wireless traffic," in *Proceedings of the IEEE GlobalSIP*, Ottawa, ON, Canada, November 2019.

[10] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "Homonit: monitoring smart home apps from encrypted traffic," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1074–1088, Toronto Canada, October 2018.

[11] Y. Cheng, X. Ji, X. Zhou, and W. Xu, "Homespy: Inferring User Presence via Encrypted Traffic of home Surveillance Camera," in *Proceedings of the ICPADS*, pp. 779–782, Shenzhen, China, December 2017.

[12] D. Caputo, L. Verderame, A. Merlo, A. Ranieri, and L. Caviglione, "Are You (Google) home? Detecting Users' Presence through Traffic Analysis of Smart Speakers," *Computer Science*.

[13] S. Farahani, "ZigBee Wireless Networks and Transceivers," *Newnes*, Elsevier, Amsterdam, Netherlands, 2011.

[14] M. B. Yassein, W. Mardini, and A. Khalil, "Smart homes automation using z-wave protocol," in *Proceedings of the 2016 International Conference on Engineering & MIS (ICEMIS)*, pp. 1–6, IEEE, Agadir, Morocco, September 2016.

[15] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-level signatures for smart home devices," *Signature*, vol. 10, no. 13, p. 54, 2020.

[16] K. Egevang and P. Francis, "The Ip Network Address Translator (Nat)," RFC 1631, May, Tech. Rep, 1994, https://www.rfc-editor.org/rfc/rfc1631.

[17] Auto js, "Auto.js," 2020, http://www.autojs.org/.

[18] H. Liu, J. Li, and D. Gu, "Understanding the security of app-in-the-middle iot," *Computers & Security*, vol. 97, 2020.

[19] P. Soucy and G. W. Mineau, "A simple knn algorithm for text categorization," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, pp. 647-648, IEEE, San Jose, CA, USA, November 2001.

[20] J. Daemen and R. V. Reijndael, "The advanced encryption standard," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 26, no. 3, pp. 137–139, 2001.

[21] C. Wang, S. Kennedy, H. Li et al., "Fingerprinting Encrypted Voice Traffic on Smart Speakers with Deep Learning," 2020, https://arxiv.org/abs/2005.09800.

[22] X. Feng, Q. Li, H. Wang, and L. Sun, "Acquisitional rule-based engine for discovering internet-of-things devices," in *Proceedings of the 27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 327–341, Baltimore, MD, USA, August 2018.

[23] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of iot devices," in *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security*, pp. 41–50, Toronto Canada, October 2018.

[24] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[25] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles, "Active behavioral fingerprinting of wireless devices," in *Proceedings of the first ACM conference on Wireless network security*, pp. 56–61, Alexandria VA USA, April 2008.

[26] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. A. Beyah, "Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems," in *Proceedings of the iNDSS*, San Diego, CA, USA, February 2016.

[27] A. Yang, C. Zhang, Y. Chen, Y. Zhuansun, and H. Liu, "Security and privacy of smart home systems based on the internet of things and stereo matching algorithms," *IEEE Internet of Things Journal*, vol. 7, 2019.