Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Privacy invasion via smart-home hub in personal area networks

Omid Setayeshfar [a,1], Karthika Subramani [a,1], Xingzi Yuan [a], Raunak Dey [a], Dezhi Hong [b], In Kee Kim [a], Kyu Hyung Lee [a,*]

[a] University Of Georgia, Athens, GA, USA
[b] University of California San Diego, Computer Science and Engineering, San Diego, CA, USA

A R T I C L E I N F O

Article history:
Received 29 October 2021
Received in revised form 22 June 2022
Accepted 23 July 2022
Available online 29 July 2022

Keywords:
Smart home security
Smart home hub
Personal area network security

A B S T R A C T

Smart-home devices are being increasingly used in our daily lives. While these devices provide convenient functions to users, such convenience may come at a greater cost, such as the leakage of the user's private information. This paper presents a system ChatterHub to address privacy risks in smart-home devices. Specifically, this work focuses on the devices that use Zigbee or Z-wave and are controlled by a centralized smart-home hub in a personal area network (PAN) for connecting to the Internet. ChatterHub passively eavesdrops on encrypted network traffic from the hub and leverages machine learning techniques to classify events and states of smart-home devices. We deployed ChatterHub on three real-world smart-home settings to evaluate its accuracy and efficiency. The evaluation results show that the attacker can successfully disclose smart-home devices' behaviors with over 89% of recall and F_1-score. We also demonstrate that an attacker can interfere with the smart-home hub's communication and selectively drop packets to disable alerting users of a device's status, such as security sensors and smart-locks. Furthermore, as a mitigation approach, we developed a packet-injection approach to effectively prevent threats from ChatterHub by generating only 9.2 MB of extra network traffic per day.

1. Introduction

According to a recent study [1], on average, 10 smart-home devices were deployed in U.S. houses in 2020. By 2025, nearly 75 billion devices will be installed. The proliferation of smart-home devices is mainly due to the convenience that users can easily access the devices to monitor and control their homes via smartphones and the Internet access [2]. However, this convenience comes at a cost. Specifically, the wide use of smart-home devices risks the breach of user privacy. An adversary with information about the usages/states of smart-home devices could obtain sensitive and private information about the users and their activities. i.e., what sensors are triggered or when the sensors are used. These device states often contain the users' activities in their home which, in turn, the adversary can use to initiate further offenses, e.g., burglary, with the information. In fact, cyber criminals are increasingly targeting smart-home devices [3].

Recent studies [4,5] demonstrated privacy invasion problems present in smart-home devices. Peek-a-Boo [4] and Apthorpe et al. [5] demonstrated how an Internet Service Provider (ISP) or attackers could learn privacy-sensitive

* Corresponding author.
E-mail addresses: omid.s@uga.edu (O. Setayeshfar), ks54471@uga.edu (K. Subramani), x.yuan@uga.edu (X. Yuan), raunak.dey25@uga.edu (R. Dey), dehong@ucsd.edu (D. Hong), inkee.kim@uga.edu (I.K. Kim), kyuhlee@uga.edu (K.H. Lee).
[1] Equal contribution.

https://doi.org/10.1016/j.pmcj.2022.101675
1574-1192/© 2022 Elsevier B.V. All rights reserved.

information from smart-home devices by analyzing wireless network traffic. However, prior works were mainly focused on smart-home devices with WiFi capability and did not conduct a study on the devices connected via personal area network (PAN), which is increasingly used for low-cost smart-home devices. Analyzing network traffic in PANs to identify patterns of user activities and device behaviors is particularly challenging because the network traffic contains events from all the devices connected to a single smart hub.

This work presents a novel method to attack smart homes and user privacy, called ChatterHub, enabling an adversary to infer smart home events and user activities by only sniffing *encrypted* network traffic to/from a target home. More importantly, ChatterHub targets devices in PAN that are equipped with Zigbee or Z-wave and require a smart-hub (e.g., Samsung SmartThings [6]) to connect to the Internet. ChatterHub requires neither *physical proximity* to the target home nor *prior knowledge* of its setup (e.g., list or network topology of devices), making it possible to attack the smart-homes remotely.

Our insight to design ChatterHub is that users' activity can routinely trigger smart devices so that an adversary can identify and learn distinct patterns in the network traffic despite being encrypted. ChatterHub employs a machine learning (ML) model, trained on traffic patterns of popular smart-home devices and hubs, to infer smart-home devices' events. The adversary can further train ChatterHub with their own devices by providing network packet traces and event logs to the training platform. Then, ChatterHub automatically partitions the network traces with our novel segmentation algorithms and feeds the segmented traces (with event labels) into the ML model to detect the events of smart home devices. Finally, the adversary can infer users' activities in the smart home by analyzing the event-triggered times and distinct patterns in the network traces.

We have evaluated the accuracy and effectiveness of ChatterHub on three real-world smart-home environments with Samsung SmartThings hub and 14 smart-home devices. The results show that ChatterHub can correctly identify various capabilities and events of each connected smart devices, i.e., `lock`, `switch`, or `motion` by monitoring the encrypted traffic from the smart-home hub. ChatterHub can also reveal users' daily routines by identifying devices' activity, including changes in the states of smart `lock`, smart LED, and multi-purpose sensors.

Note that this work is an extension of our previous publication [7]. While our previous work introduced the idea of ChatterHub with ML methods to infer smart-home devices and their events, this work takes a step further to use an advanced deep learning (DL) model as a component of ChatterHub and support different attacks for smart-home devices and the network traffic. More specifically, we improved the accuracy of ChatterHub's classifier by incorporating DeepTrafficNet, a mixture of a convolutional network (CNN) [8] and long short-term memory (LSTM) [9], and provide an in-depth analysis of ChatterHub's performance with different classification models, such as random forest, seq2seq, XGBoost, and DeepTrafficNet. In addition, we study selective packet-dropping attacks to understand how they affect smart home devices' states and the user's awareness. We demonstrate that a powerful attacker who can interfere with the network communication (e.g., compromising the router, man-in-the-middle attack) can selectively drop packets to disable security monitors (e.g., motion sensors, door contact sensors) and the user commands (e.g., lock the door). As a result, we show that such attack methods affect the integrity of the devices and the security of the user's home.

In summary, this paper makes the following contributions:

- We developed ChatterHub, a new attack method against smart-home devices connected via personal area network (PAN). ChatterHub can identify user activities and devices' states while the devices are hidden behind a smart-home hub.
- We designed DeepTrafficNet, a new classification model to accurately identify the events and usage patterns of various smart-home devices from encrypted network traffic. With DeepTrafficNet, ChatterHub successfully recognized smart-home devices' events with 89% F_1 score.
- We studied packet-dropping attacks. This attack method allows an attacker to affect the integrity of the devices, hence compromising the security of smart-home.
- We also developed mitigation techniques that could be employed to protect user privacy in smart-home from our proposed attack model. Specifically, using packet padding and random sequence injection can effectively protect user privacy with minimal network overhead (e.g., 9.2MB traffic per day).

We structure the rest of the paper as follows. Section 2 defines the adversary model and specify the adversary's goal. Section 3 describes the design of ChatterHub and how ChatterHub can collect and analyze data from encrypted network traces. Section 4 discusses the evaluation results of ChatterHub, and Section 5 describes packet dropping attacks. Section 6 discusses possible mitigation approaches against adversaries using ChatterHub. Section 7 discusses related work, and we conclude this paper in Section 8.

2. Adversary model and goal

We assume that an attacker only sniffs encrypted network packets from/to the target home. There are three potential weak points where the attacker can eavesdrop on network traffic. First, the attacker can gain access to the traffic from a compromised router (e.g., Mirai attack [10]). Second, the attacker eavesdrops on network traffic from the home router's uplink traffic (e.g., network sniffing device [11]). Third, the attacker can be one who can monitor the network traffic of the target home, such as ISPs. Please note that our adversary model is a passive attacker who collects encrypted network traffic (e.g., TLS/SSL). The attacker only observes limited information in the network traffic, including the size of each incoming and outgoing packet, the source and destination IPs, and timestamps. However, we assume that the attacker

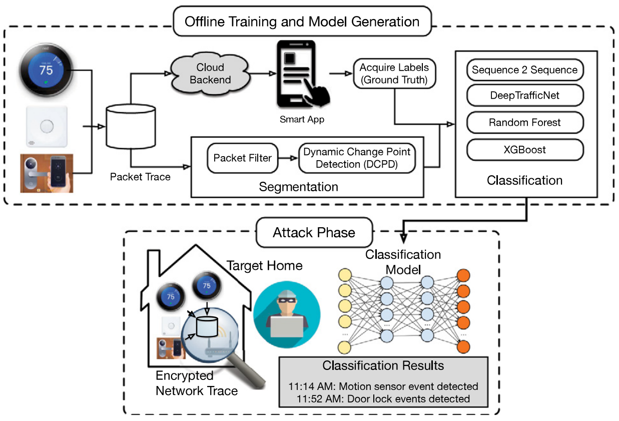

Fig. 1. A high level overview of ChatterHub.

cannot decode or interpret the information inside an encrypted packet. Moreover, we assume that the attacker can access network data generated from the same model/device of a smart-home hub as the target home (e.g., Samsung Smartthings) to train a classification model. The attacker can use a pre-trained classification model on a victim's network to classify the activities of smart-home devices. The attacker does not need prior knowledge of a topology/complete list of smart-home devices deployed in the target home.

The Goal of the Adversary. Once network packet traces from the target home are obtained, the adversary uses a classification model in ChatterHub or trained by the attacker's hub and devices. The adversary can understand the behavior sequences of smart-home devices at the target home by doing so. Therefore, we consider that the attacker can achieve the following goals (but not limited to):

- **Scout Attack.** The attacker targets a range of IP addresses to find vulnerable home routers, similar to Mirai attack [10]. After gaining access to the routers, the attacker analyzes traffic in routers or through a virtual redirection to a sniffer-installed device. Understanding the behaviors of smart-home devices will allow the attacker to find vulnerable targets for a further offensive campaign, such as burglary.
- **Targeted Attack.** The attacker sniffs the outgoing traffic from the target home by gaining access to network sniffing tools [11]. After enough scouting, the attacker can understand the smart-home devices' behaviors, identify patterns of household activities, and use the patterns for physical assault.
- **ISP-level Tracking.** ISPs can learn the patterns of the households' daily life. Such information can be used for targeted advertising or other unwanted activities, potentially violating users' privacy.

Target Devices. Two types of smart-home devices are available on the market; (1) WiFi or Ethernet-enabled devices and (2) devices equipped with home automation network modules, such as Zigbee, Z-wave, or BLE. The category device type can directly connect to the access point. On the other hand, the second category device cannot connect to the Internet directly, and they rely on a smart-home hub responsible for controlling communications among devices. Additionally, the second category devices are hidden behind the hub and considered more secure against remote attackers [12] [5,13–15]. While several existing works [5,13–15] investigated the security and privacy challenges of WiFi-connected devices (the first type of devices),very few studies exist that concern the security and privacy problems in the second category. This work mainly focuses on the privacy and security challenges in the second category of the devices in PAN. While there are several smart-home hubs available on the market, e.g., Apple Home Kit [16], Google Home [17], Amazon Alexa [18], Samsung SmartThings [6], Simplisafe [19], we use Samsung SmartThings as the primary use case of a smart-home setting. This is because Samsung SmartThings has a high market share and supports a wide range of smart-home devices [20,21].

3. System design

This section describes the design of ChatterHub. *Minimally intrusive monitoring* is one of the most important design goals of ChatterHub as the adversary only requires access to the network traffic from/to the home. Moreover, obtaining access at this level is ascertained to be relatively more straightforward compared to using eavesdropping devices that need to be used near the target devices [5,22,23].

Fig. 1 shows an overview of ChatterHub. We use 15 different devices with 12 unique capabilities. Tables 1 and 2 present detailed information about devices, a list of capabilities, events, and the interactions collected.

Table 1
List of devices and capabilities (ZB: Zigbee and ZW: Z-Wave).

Type	Device type	Device name	Comm. Ch.	Capability
Sensors	Multipur. sensors	Centralite micro door sensor	ZB	activity, contact, temperature
		Smartthings multipurpose sensor	ZB	contact, status, temperature
		Samsung multipurpose sensor	ZB	
	Water sensors	Iris smart water sensor	ZB	battery, water, temperature
		Centralite water Sensor	ZB	
	Motion sensors	Centralite motion sensor	ZB	motion, battery, temperature
		Samsung motion sensor	ZB	
Actuators	Smart light	SYLVANIA smart 10Y A19 TW	ZB	switch, switchLevel, colorTemperature
		SYLVANIA smart + Adjust. White RT5/6	ZB	
		Sengled element plus	ZB	
	Smart plug	Centralite smart outlet	ZB	switch
		Sylvania SMART + smart Plug	ZB	
	Smart locks	Kwikset 10-Button Deadbolt	ZW	lock, battery
	Dimming switches	OSRAM LIGHTIFY Dimming switch	ZB	button
Hub	Hub	Samsung smartThings hub	ZB, ZW	ping

Table 2
Event types for capabilities.

Capabilities	# Events	Commands
button (1)	410	`push`, `held`
lock (2)	584	`lock`, `unlock`
motion (3)	406	`inactive`, `active`
switch (4)	1562	`on`, `off`
switchLevel (5)	3181	`change`
temperature (6)	790	`change`
water (7)	117	`dry`, `wet`
colorTemperature (8)	853	`change`
activity (9)	31	`online`, `offline`, `hub disconnection`
status (A)	572	`open`, `close`
contact (B)	708	`open`, `close`
battery (C)	16	`change`

3.1. Training data collection

We collect network traffic to/from our smart-home setup and event logs in the smart-home devices to train the classification model. We monitor the network traffic using `Wireshark` [24] installed on a laptop that is connected to the Samsung SmartThings hub through a bridged network. We obtain the event labels corresponding to the network traffic from the logs delivered through the hub. Samsung SmartThings hub stores event logs, e.g., all commands sent to its smart-home devices along with timestamps. We collect the logs regularly with "Simple Event Logger" [25] provided by the manufacturer. The logs are periodically (e.g., 5 min.) stored in a file as CSV format. Through the above procedure, we can collect over 200k network packets from the smart-hub with over 60k event logs, which will be used to train the classification models in ChatterHub. After that, we label the collected network packets. To gather unique network patterns generated by the devices, we use the following setups.

1. **Single device.** We connect a single device to the hub at a time and monitor the network traffic generated by the device. We repeat this for all the devices to identify the unique traffic patterns generated and consumed by the devices.
2. **Multiple devices.** We connect multiple devices to the hub and monitor traffic concurrently generated by multiple devices. The measured data trains our model on traffic simulated by a more realistic situation. For example, we observed that packets generated by multiple device commands are often overlapped. Therefore, we feed the overlapped data to ChatterHub to train such cases. Furthermore, we connect not only smart-home devices to the hub, but also other home appliances (e.g., laptops, tablets, or smartphones) to the router to represent the real-world workloads.
3. **Only the hub.** We observe network traffic directly from the hub to understand the hub's behaviors. i.e., periodic check of firmware update. We collect traffic generated or consumed by the hub in an isolated environment by disconnecting all devices to correctly understand the hub's behaviors.

Moreover, the smart-home devices can be classified into two categories; *actuators* and *sensors*. We develop an android app that automatically triggers various events for the actuators. However, this approach does not work for the sensor-type

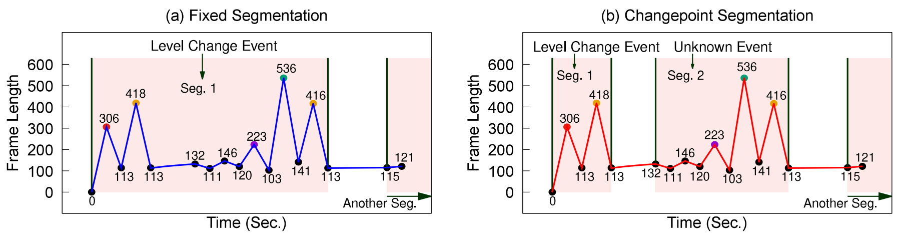

Fig. 2. Fixed segmentation vs. changepoint segmentation. (seg.: segment)

devices, so we manually generate events for such sensors, resulting in fewer data collected from the sensor-type devices than the data collected from actuator devices. We collect the training data from 14 smart-home devices and Samsung SmartThings hub. Table 1 lists the information of each device, and Table 2 shows the number of interactions collected from both device types based on their capability.

3.2. Trace segmentation and labeling

We analyze the collected network traces to extract features for the classifier. First, we design a method to filter out network packets unrelated to the SmartThings hub (e.g., packets generated by other devices connected to the network). When the hub is registered, it connects to an authentication server (`Auth-Server`) in the cloud. The hub exchanges authentication keys with `Auth-Server` and receives an IP address of a communication server (`Comm-Server`) in the cloud. The hub then connects to `Comm-Server` for further communications and operations. The communication channel between the hub and `Comm-Server` always remains established, and network traffic between the hub and `Comm-Server` uses this channel. The traffic in this channel shows an almost identical pattern, indicating that it is not feasible to partition packets based on the network flows. However, we observe that whenever a packet sequence gets exchanged between the hub and `Comm-Server`, it happens over a short time interval (within a few seconds).

Also, the communication interval in packet sequences between two commands is relatively large compared to the interval between packets sent for a single command. We thus apply a segmentation method to the collected network flow to divide the traces into small bursts of packets. For the data preprocessing, we remove all "`TCP-ACK`", certification, and authentication packets before the segmentation process. Only the packets related to application data are retained for further analysis. In order to segment the network flow into separate bursts, we evaluate various approaches proposed from previous studies [22,26] that used a fixed threshold of 4.5 s to segment network packets into multiple bursts. Their results showed that 4.5 s was enough for the communication between a client and server to complete the packets exchange in their evaluation setup. However, we observe cases where the time gap between packet exchange for a single command can last longer than 4.5 s. For example, Fig. 2 shows a case where a fixed threshold approach fails to separate the *level change* event from other events. Also, activities of the hub (e.g., ping, status) can occur along with other device commands within less than 4.5 s. In these cases, the segmentation based on a fixed threshold of 4.5 s often fails to correctly segment the device commands from other packets (e.g., ping and status). Therefore, we develop a dynamic segmentation technique using a *change point detection* method [27] to correctly segment these packets into the bursts.

Dynamic Change Point Detection. Change Point Detection (CPD) is an approach to finding abrupt changes in time-series data [27]. This approach can also be used for estimating the temporal point when the statistical properties of a sequence in observations change [28]. Among several existing implementations of CPD algorithms [29–31], ChatterHub employs PELT (Pruned Exact Linear Time) [28] because it is computationally efficient and outperforms other exact CPD search methods [31]. A *change point* is a temporal point when the statistical properties of its previous and subsequent time points are different. For example, in our smart-home setup, the network packets corresponding to a single command are issued in short intervals compared to intervals between two distinct commands. Therefore, a change point will be the point when a sequence of packets for a single command/event starts and ends. Since our logs are collected over a long period of time, multiple *change points* need to be identified to segment all events. We present detailed evaluation results of Dynamic CPD and fixed threshold segmentation algorithms on our dataset in Section 4.1.

Labeling. After the packet segmentation, we acquire event labels from the logs generated by the hub. We use timestamps for mapping between the label and the segmented trace. However, we observe that slight time differences between *generation of command log* and *packet capture* can occur. Hence, we use a ± 5 s of padding to generate an approximation; then, we map the command to a specific burst of packets. We also observe special cases that *a single command enables multiple capabilities in a device*. For example, a "switch on" command activates the switch, and a "level change" event triggers two capabilities (switch and level). Therefore, a single burst of packets can be mapped to multiple capabilities. We also observe that a number of segments are not associated with any labels (i.e., no logs from the hub and `Comm-Server`), so we label them as *unknown*. To further characterize the unknown packets, we further analyze the source code of device handlers [32] and find that the handler generates the event logs, and some of the handlers do not emit any logs. We observe that most missing events are less important for the user (e.g., device refresh, device ping).

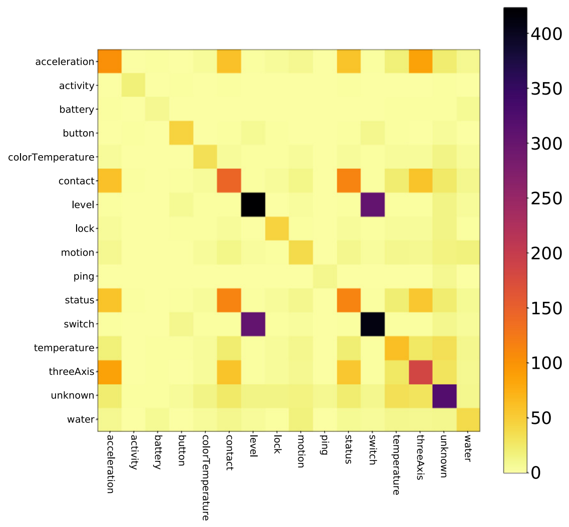

Fig. 3. Label collision matrix for capabilities dataset. This plot shows how unique signatures of the same label collide with other labels.

3.3. Feature extraction

To train the classification models, we extract features from segmented network traffic. From each segment, we form a signature by combining the frame length of packets in that segment. This signature is then used as the feature for training classifiers. We analyze the overlap of signature features in different capabilities and events of the devices. Fig. 3 shows the collision of frame length signatures between multiple capabilities. These collisions are expected because when some devices are activated, they generate multiple capabilities and events simultaneously. For example, a collision between switch and level can be observed as these capabilities concurrently occur whenever a user turns a switch on and off. Similarly, contact and status capabilities can concurrently occur when a user opens a door because they are subsequent events. Please note that there exist a few overlaps of signature between capabilities and *unknown* labels.

3.4. Comparison of classification algorithms

To infer capabilities and events from extracted network features, we evaluate multiple ML methods, including rule-based models, expression-based models, and DL approaches. This section evaluates existing ML models and discusses their limitations. Then we introduce a novel method to improve classification accuracy in ChatterHub.

We started by employing a rule-based model that classifies the traffic based on simple rules. On plotting the changes in the number of unique signatures in our training data, we observed that such a fixed-rule-based approach was insufficient due to an infinite number of signature combinations. Next, we applied regular expression-based models to see if we could find a fixed set of regular expressions that would correctly explain the signatures. However, we observed that the signatures with frame length varied often and were difficult to generalize for a single capability or event. Fig. 4 illustrates the signatures observed for "switch" capability as an example. As shown in Fig. 4, since the signatures are changing dynamically, leveraging the regular expression-based method appeared to be challenging for addressing such dynamics.

We then evaluated following four ML models; (1) Random Forest, (2) DeepTrafficNet, (3) seq2seq, and (4) OneVsRest. Among these models, Random Forest is our *baseline* model [33] because Random Forest, an extension of decision trees, has been widely used in security domains. The other three models are explained below.

Please note that, for DeepTrafficNet and Random Forest, we preprocess the data via the following steps. We begin by extracting the frame length from the segmented traffic and create a vector $X = \langle x_0 \dots x_i f \rangle$ of packet sizes. Then, we turn the collected labels to a vector $Y = \langle y_0, \dots, y_n \rangle$, where n is the number of labels in the dataset. y_i is calculated as 1 if $label_i$ is present in y. Otherwise, y_i will be 0. Finally, to make all Xs the uniform length, we repeatedly pad them until they form the desired length. We choose the length to be the 90%tile of sequence lengths in our training set, and the normal size of the sequence length is 15.

DeepTrafficNet is a custom model for ChatterHub, leverages the combined power of CNN [8] and LSTM [9]. The architecture of DeepTrafficNet is illustrated in Fig. 5. CNN provides effective feature extraction from a large-scale dataset. LSTM is adept at learning sequences from data. An advantage of DL, e.g., deep neural networks (DNN), over traditional ML is the inherent feature extraction, which does not require hand-crafted feature extractions. Moreover, DL shows outstanding performance with large-scale training data. Regarding this, we report the performance of DeepTrafficNet with different sizes of the training dataset in Section 4.2.

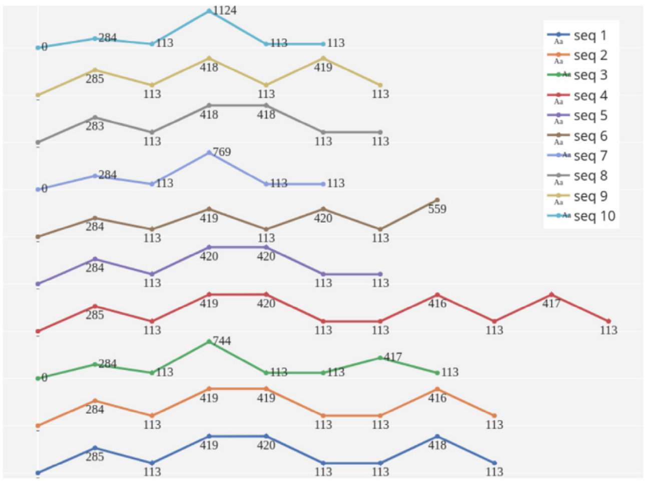

Fig. 4. Sequences of traffic packet sizes observed for multiple instances of "switch" capability.

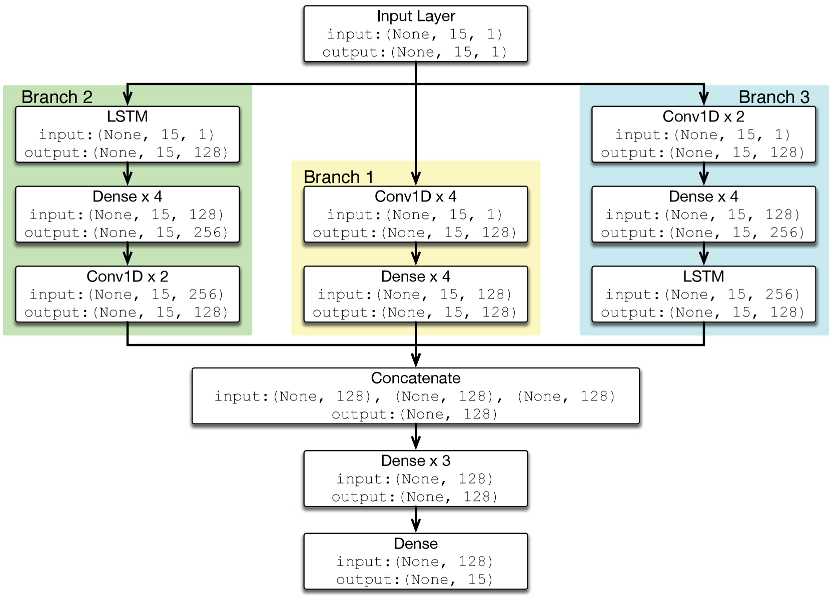

Fig. 5. Architecture of DeepTrafficNet.

DeepTrafficNet is designed to reflect the key goals to solve our problem. In particular, we divide the network into three branches (Br_1, $Br2$, Br_3) operating directly on the input sequence of network traffic, which will be combined to obtain the final classification result. The architecture and parameters of this network are finalized and tuned with thorough experimentation.

- Br_1. The first branch consists of two 1*d*-CNN layers of size 128 with a filter size of 3, followed by four dense layers of size 128 each. Br_1 aims to extract important features from the network traffic irrespective of the ordering of the sequence.
- Br_2. The second branch has a LSTM layer of size 128, followed by a time-distributed dense layer of size 256. These layers analyze the sequence and extract all features related to the packet orders. We then use two 1*d*-CNN layers of size 128 to co-relate the different sequences learned in the LSTM model and try to further extract essential features from the LSTM layers. Br_2 is first to extract meaningful sequences information, followed by searching for patterns in the sequences.
- Br_3. This branch is the inverse of the Br_2 branch with the 1*d*-CNN layers preceding the LSTM layers. Br_3 is designed to perform the first search for meaningful features in the packets, followed by checking the sequence of extracted essential features. The parameters used are the same as Br_2.

After executing each of the previous branches, we flatten each of the layers respectively and then obtain a summary of features using a dense layer of size 128. The last stage of DeepTrafficNet involves five dense layers of size 128, which perform the final processing. With this procedure, we have the output layer, consisting of a dense block of size 16 corresponding to the respective classes (26 for the network with events). We use rectified linear unit (ReLU) as the activation function for all the layers except the last one, in which the activation function is *sigmoid*. We then apply batch normalization [34] and a dropout [35] of 10% after every layer to prevent over-fitting.

We evaluate the outputs from DeepTrafficNet with a loss function, $L(p, q) = 3 \times F1(p, q)^2 + H(p, q)$, and find parameters that minimize the loss function by using $f(x, y) = argmin\ L(\hat{f}(x), y)$. The loss function of DeepTrafficNet is a combination of F_1 and weighted categorical cross-entropy. F_1 is expressed as $2 \times \frac{P(p,q)\times R(p,q)}{P(p,q)+R(p,q)}$, where $P(p, q) = \frac{TP}{TP+FP}$ and $R(p, q) = \frac{TP}{TP+FN}$. In weighted categorical cross-entropy, the weights are defined as an inverse proportion to the frequency of occurrence of the respective classes in the training set. i.e., lowest occurring class has the highest weight to counter the case of disproportionate class distribution in the training set. We use ADAM [36] optimizer with the default parameters. DeepTrafficNet considers 1.8 million parameters and is implemented using `Keras` [37] with `TensorFlow` [38]. We present detailed results of each classification method on our dataset in Section 4.4.

Sequence-to-sequence (seq2seq) model builds a common framework for solving sequential problems. The input of seq2seq is a sequence of certain data units, and the output is also a sequence of data units [39]. seq2seq model is widely used in natural language translation, where the input is usually a sentence, which is a bag of words (tokens) separated by spaces (or separators). The output is also a sentence but in a different language. In our model, we have a sequence of features (e.g., package length), and the desired output is a sequence of capabilities and events. In the preprocessing step, we separate our selected features with a separator ('space' in our model). The seq2seq framework contains two main components: an *encoder* and a *decoder*. The encoder reads the sequence of input data, and the decoder translates encoded data into the final sequence of outputs as specified in the paper [39]. The encoder and decoder are implemented based on LSTM or multi-layered CNN.

OneVsRest follows *one-vs-rest strategy* that uses a classifier for each class fitted against all other classes. Given that our data involves multiple classes, this strategy fits well for our problem. This method ensures that each classifier is independently optimized to identify features for the corresponding class. As this entails a large number of classifiers, we use an efficient classifier, such as XGBoost (Extreme Gradient Boosting), as the base classifier for the OneVsRest classifier. XGBoost is an ensemble method that applies Gradient boosting on decision trees to boost the performance of the various models [40–42]. We use the `XGBClassifier` of `XGBoost` library [43] with its default parameters. After performing the trace segmentation, we obtain the sequence of packet lengths along with the direction, and then we convert this sequence of packet lengths into a text sentence. For example, suppose that one of the sequences for switch capability would be converted to '$-285 + 113 - 1124 + 113 - 113$' where the '$+$ ' and '$-$' represent the direction. Then, we transform the data into a vector matrix by using `CountVectorizer`. For the vectorizer, we set the parameter `ngram` to consider ngrams of range from 1 to 4, such that the relationship between the packets in the sequence are maintained. This transformed count matrix is then given to the OneVsRest classifier with the XGBoost model as the base classifier to predict the labels.

4. Evaluation

4.1. Network trace segmentation

This section reports the evaluation results of various segmentation methods with our collected data and the accuracy results (precision, recall, and F_1) from the segmentation methods. Trace segmentation was performed on the captured traffic to partition the overall traffic flow between a hub and cloud servers (`Auth-Server`, `Comm-Server`) into a set of small bursts that were mapped to specific commands. An accurate segmentation will ensure that each packet burst contains a negligible amount of noisy packets. Note that noisy packets indicate *unknown* packets or packets for a status report of the hub.

As discussed in Section 3.2, we use a PELT-based CPD to segment the network traffic. One reason for choosing CPD over a fixed threshold method is that CDP can produce segments with less noisy signatures over the fixed threshold method, as shown in Table 3. We found that the number of unique sequences for most capabilities is lesser when segmentation is performed using CPD compared to the fixed threshold of 4.5 s We also observe that the range of sequences obtained from segmentation using a fixed threshold is much longer compared to the segmentation with CPD. Such longer sequences often contain more noisy data, hiding the actual signature.

Furthermore, the PELT algorithm can be used with different cost functions, taking the cost function output as a penalty value, which affects the segmentation results. We use two cost functions; least squared deviation (L2) and RBF (Radial Basis Function) kernel. We segmented our traffic with these cost functions and used the segmentation results to train our baseline model (Random Forest). Fig. 6 shows the test results of the trained model with one of our test data. The results report different segmentation methods' classification accuracy (precision, recall, and F_1). The highest precision and F_1 were achieved by segmentation with the PELT algorithm (RBF of the cost function and penalty value of 0.2). Therefore, we used this approach to segment our traffic datasets and train our other models (DeepTrafficNet and seq2seq).

Table 3
Comparison of segmentation methods based on properties of sequences.

Capabilities	Changepoint detection segmentation		Fixed threshold segmentation	
	# Unique sequence	Sequence size	# Unique sequence	Sequence size
colorTemperature	32	2–12	48	4–26
level	439	1–51	526	2–152
switch	439	1–51	525	2–152
contact	166	1–23	269	2–247
status	139	2–23	222	2–247
acceleration	97	2–23	152	2–247
threeAxis	131	2–15	127	9–247
button	92	1–20	74	1–90
lock	35	2–14	68	2–49
motion	63	2–11	99	2–24
water	34	2–8	40	2–16
temperature	78	1–10	116	2–27
battery	12	2–10	12	2–16
activity	17	1–12	18	1–35

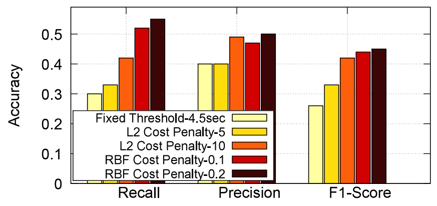

Fig. 6. Performance evaluation of various segmentation methods.

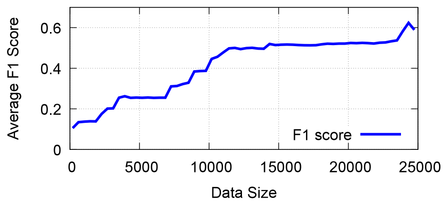

Fig. 7. Impact of training set size to changing to average F1-score.

4.2. *Effect of training data size*

As this attack relies on the efficiency of models to classify the traffic datasets into services and events, training quality and the effort for getting an accurate result can play an essential role for the attacker. We analyze the effect of training data size on the classification accuracy of a fixed set of testing data. Fig. 7 shows the change of accuracy (F_1) with different training data sizes. The results show that the classification accuracy increases as the size of training data increases. Also, the efforts required by the attacker will be minimal, which, in turn, makes the attacks even more eminent.

4.3. *Evaluation on simulated smart-home*

We created a smart-home environment at three authors' homes and recorded the network traffic from the router. The SmartThing Hubs in the three homes were connected to the router via WiFi along with other Internet-connected devices (e.g., laptops, smartphones, tablets). We first trained the classification models of ChatterHub using datasets collected from the lab setting and one of three smart-home setups. To evaluate the accuracy of the classification models, we used datasets obtained from the other two smart-homes, which are not used for training. Then, we conducted two experiments with different attack scenarios; (1) an attacker tried to infer the capabilities of the devices, (2) the attacker tried to detect specific events in those capabilities.

In the first experiment, the attacker would be aware of "switch" being present and used in the target home. In the second experiment, we inferred if the attacker would know if a "switch on" or "switch off" had happened. Moreover, to create more realistic scenarios, we added a water sensor device in the simulated smart-home that was not used when

Table 4
Classification results for capabilities only.

Capabilities	Random forest		DeepTrafficNet		seq2seq		XGBoost	
	Recall	F_1	Recall	F_1	Recall	F_1	Recall	F_1
button	0.000	0.000	0.672	0.075	0.656	0.491	**0.951**	**0.568**
colorTemperature	0.200	0.333	0.125	0.132	0.200	0.333	**0.675**	**0.771**
contact	0.388	0.540	0.422	**0.561**	**0.427**	0.492	0.402	0.559
level	0.850	0.428	0.713	0.174	0.637	0.333	**0.812**	**0.428**
lock	0.196	0.272	**0.521**	**0.276**	0.172	0.269	0.182	0.251
motion	0.160	0.196	0.201	0.144	**0.406**	**0.313**	0.147	0.187
ping	0.994	0.991	**0.995**	0.990	0.963	0.978	0.993	**0.991**
status	0.667	0.764	**0.840**	**0.840**	0.703	0.698	0.667	0.777
switch	0.607	0.655	**0.705**	0.512	0.525	0.376	0.574	**0.673**
temperature	0.075	0.118	0.205	0.169	**0.356**	**0.302**	0.048	0.082
unknown	0.994	0.926	**0.975**	0.934	0.881	0.880	0.942	0.925
Average	0.888	0.896	**0.917**	0.852	0.859	0.848	0.888	**0.897**

Table 5
Classification results for capabilities and events.

Capabilities-events	Random forest		DeepTrafficNet		seq2seq		XGBoost	
	Recall	F_1	Recall	F_1	Recall	F_1	Recall	F_1
button-held	0.000	0.000	0.091	0.087	0.000	0.000	**0.273**	**0.171**
button-pushed	0.000	0.000	0.353	0.383	0.608	**0.569**	**0.863**	0.547
colorTemperature-change	0.200	0.333	0.150	0.126	0.175	0.369	**0.675**	**0.771**
contact-closed	0.194	0.313	**0.387**	**0.428**	0.289	0.296	0.245	0.388
contact-open	0.307	0.419	0.414	0.463	**0.502**	**0.469**	0.341	0.455
level-change	**0.863**	**0.430**	0.738	0.183	0.631	0.330	0.812	0.428
lock-locked	0.049	0.086	0.049	0.090	**0.160**	**0.217**	0.049	0.086
lock-unlocked	0.195	**0.230**	**0.390**	0.126	0.012	0.024	0.171	0.206
motion-active	0.060	0.105	**0.143**	**0.145**	0.101	0.130	0.053	0.096
motion-inactive	0.236	0.230	0.126	0.123	**0.298**	**0.245**	0.238	0.230
ping-ping	0.994	0.991	**0.996**	0.990	0.962	0.977	0.993	**0.991**
status-closed	0.391	0.533	**0.800**	0.584	0.504	0.513	0.470	**0.617**
status-open	0.479	0.596	**0.710**	0.404	0.548	0.515	0.645	**0.755**
switch-off	0.452	0.596	**0.710**	0.434	0.548	0.515	0.645	**0.755**
switch-on	0.226	0.055	**0.581**	0.279	0.290	0.176	0.290	**0.375**
temperature-change	0.076	**0.119**	**0.098**	0.101	0.070	0.102	0.048	0.082
unknown	0.944	0.926	**0.964**	**0.931**	0.939	0.918	0.942	0.925
Average	0.881	0.892	**0.900**	0.864	0.874	0.873	0.882	**0.893**

we trained the classification model. This addition was to consider a situation where the attacker did not have a complete list of installed devices in the target home.

Detecting the Hub. The first step was to identify the IP address of SmartThings server (`Comm-Server`) that communicated with the smart-home hub. To this end, we calculated the statistics about the number of packets communicated to each IP address and the IP addresses that contained the packets similar to events related to "SmartThings hub's ping" and the count of such ping packets. Using these statistics, we could recognize the IP addresses of `Comm-Servers`. Although the hub kept changing the `Comm-Server` from time to time (usually over days), the information collected was enough to identify the servers. Identifying the IP addresses of the hub and `Comm-Server` helped us collect essential network packets by ignoring packets communicating between the hub and `Comm-Server` and the traffic generated by other devices from the smart home.

Classification. We trained our classification models to classify capabilities and events separately. Table 4 reports the classification results for capabilities and events, including recall and F_1-score of the models at the level of device capabilities, such as *switch* and *motion*. Table 5 presents the classification results, considering both capabilities and events from each device, such as switch-on, switch-off, motion-active and motion-inactive. These models were trained to label any sequences they cannot recognize as *unknown*. If there are devices that were not used in the model training, ChatterHub categorized events and capabilities belonging to this device as *unknown*.

We observed that each model performs better for different classes from these results. Overall, the XGBoost model could classify 89% of the capabilities of the smart devices, which implies that the XGBoost model can be used for getting a general idea of the devices in the target home. Then, based on the information from XGBoost, the attacker could use a specific model to obtain more accurate information on a specific device's capability. To demonstrate this use case, we designed two different evaluation cases with our simulated home settings by intentionally deploying different sets of devices. In the first case setup (case #1), we created a smart-home testbed with all devices, including the water and lock devices. In the second case (case #2), we created another smart-home testbed with all devices (including water device)

Table 6
Classification results with respect to specific devices in different home setup.

Case	Case #1 : Home with lock device				Case #2 : Home without lock device			
Capability	OneVsRest − XGBoost							
	Recall	Precision	F_1	Count	Recall	Precision	F_1	Count
contact	0.880	0.349	0.500	50	0.780	0.448	0.569	50
lock	**0.941**	**1.000**	**0.970**	**34**	–	–	–	**0**
switch	0.762	0.941	0.842	21	0.611	0.846	0.710	18
unknown	0.997	0.996	0.997	7105	0.999	0.975	0.987	1910
Average	0.895	0.822	0.827		0.796	0.756	0.750	

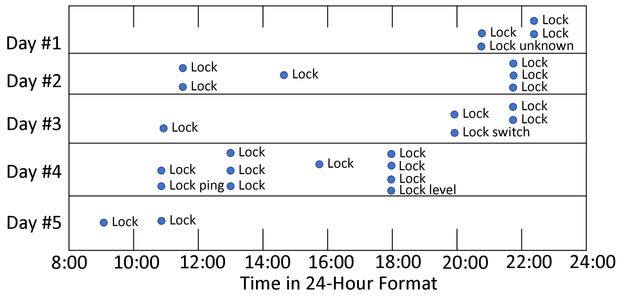

Fig. 8. Detected smart lock behaviors. It shows how the *lock* and *unlock* events (possibly entry and exit behaviors of tenants) can be recognized by the adversary.

except for the *lock* device. As shown in Table 6, our models could detect *lock* with a precision of "1", indicating that the classification results have "0" false positive case. On the other hand, even if our models were trained on signatures of *lock*, in the case #2 (the testbed without *lock* device), no lock capability was detected. Therefore, the results show that our models are accurate enough to detect the presence of devices.

Fig. 8 shows the activities of a smart lock at various times of the day. We measured the devices' activities for five consecutive days. The results show that the lock had the events on multiple days at 11:00 a.m. and 11:00 p.m. (23:00). Based on this observation, the attacker can know the homeowner's daily schedule. Hence, the classification results with further analysis of such patterns reveal potential information about the smart-home devices and the users.

4.4. Analysis of classification results

As discussed in Section 3.4, we decided to use Random Forest as the baseline for evaluating the performance of classifiers, and Random Forest showed the lowest accuracy (recall, precision, F_1) among four classification models.

By comparing with Random Forest, we observed that DeepTrafficNet could learn more relevant features leading to better generalizations. As discussed earlier, DeepTrafficNet could handle unseen signatures better than other models, resulting in less need for large training datasets. However, we also observed that DeepTrafficNet could generate inconsistent classification results for some classes. i.e., *colorTemperature*, *motion*. Our further analysis revealed that such inconsistent results were due to the overlapped packets and the shorter packet sequences for those classes.

While seq2seq model showed comparable classification accuracy with DeepTrafficNet for a few cases (but similar to DeepTrafficNet), we observed that seq2seq failed to handle overlapped packet sequences. seq2seq was designed to process large-scale (language) datasets, and its performance could be affected by the amount of training dataset. However, it was challenging for us to generate enough data to well train seq2seq in our evaluation environments due to the restricted automation and event generation procedures. Instead, we generated synthesized packet sequences by concatenating real sequences. The longer sequence helped seq2seq for better mapping between features and events.

Compared to other models, XGBoost is more resilient to noisy data [44], and it showed more accurate classification results for overlapped packets. However, we also observed that the signature conflict between capabilities from the same devices could lead to misclassifications. Since DeepTrafficNet identifies smart-devices' events with high recall (0.9), an attacker could use this model to filter all the smart-devices related packets in general and then use a model such as XGBoost (with F_1-score of 0.89) to detect the specific devices and events on the filtered traffic for better accuracy

5. Packet dropping attacks

This section discusses packet dropping attack to show its feasibility and effectiveness when an attacker can interfere with the network communication using ChatterHub. For packet dropping attack, we tweak our adversary model discussed

Table 7
Results of packet dropping attack per capabilities.

Capabilities	Event count	Dropped event count
lock	100	100
motion	31	31
switch	100	99

in Section 2 and use an additional assumption that the attacker can selectively drop the network packets (e.g., compromising a router or Man-in-the-middle attack). Please note that we still hold an important restriction on the attacker; that is, the attacker cannot decrypt the packet to understand the content. The following describes two use cases of the packet dropping attack and how they will affect smart home devices and the user's awareness.

Case #1: Dropping Hub Traffic. In this case, the attacker simply drops all network packets generated by and sent to the smart home hub device. By doing so, any events of the smart devices (e.g., motion detection, temperature change) will not be reported to the user, and user commands via the smartphone app (e.g., lock the door, turn on the light) cannot be delivered to target devices. While this attack prevents any command and updates to be relayed from and to the user, we noticed that the user could receive a notification from her smartphone app, indicating that the hub is offline. On analyzing the reason for the offline notification, we identified that if the cloud server fails to receive ping messages from the hub for a certain time interval (e.g., two minutes for Samsung Smartthings Hub), it sends the offline notification to the user app. To avoid the victim user getting alerted about the hub's offline status, the attacker could selectively allow network packets related to hub ping and drop the rest of the traffic. As identified already, the ping from the hub follows a signature of 115 ↑ 121 ↓.[2] We then configured our attack to allow only the hub pings to pass but block all other traffic. Unfortunately, it could not completely prevent the hub's offline status from being notified to the user (but is simply delayed). This is because, in addition to the periodic ping messages, the hub and the cloud server also exchange other types of packets that we do not observe from any logs and possibly classified by our classifiers as "unknown". This demonstrates the need for a more advanced form of packet selection for dropping traffic.

Case #2: Selective Packet Dropping To overcome the defects of previous scenarios, an attacker must use a technique to differentiate the traffic generated by smart devices events' from the hub's event traffic. In this attack, the attacker selectively drops the packets related to specific device activities identified by ChatterHub. This study connects all the devices described earlier, choosing the following security-sensitive devices:"switch", "motion" and "lock". Table 7 shows the result of the selective packet dropping attack. We could successfully drop all lock and motion activity packets and 99 out of 100 switch activities packets. As a result, the user did not receive any notification of the motion detection events, and the user's commands to lock the door and turn on/off the switch were completely ignored without any notifications. In this scenario, we observed that dropping selective packets did not affect the status of other devices connected to the hub or the hub's status.

6. Discussion and ethical consideration

Mitigation Approach. As ChatterHub detects smart devices and their events by identifying patterns in encrypted network traffic, we propose to use "Packet Padding" [45] as a mitigation method against ChatterHub. The idea of packet padding is to generate the identical length of packets by adding bogus bytes at the end of each packet. Network packets with identical lengths can effectively hide unique patterns in the network traffic, significantly decreasing the accuracy of the classifiers. We implement the packet padding technique in our testing router to make each packet of exact 1 KByte. In addition to that, we also developed a random sequence injection method to further mitigate ChatterHub or similar attacks. This method dilutes the effect of sequence timing by irregularly generating 1 KB packets to the network. When we deployed both packet padding and random sequence injection methods together, we observed the network traffic became much more complicated and dynamic. Specifically, when we apply our two mitigation techniques we observe more than 80% collision in the classification process in ChatterHub. It shows that the proposed mitigation techniques can effectively hide the unique network patterns generated by each device. Thus the attacker will have a lower chance of learning the states of smart-home devices.

We evaluate space overhead caused (i.e., additional network traffic generated) by our mitigation techniques from our testing environments. The results show that only negligible size of additional network packets (9.2 MByte per day on average) are required to enable the proposed mitigation methods.

Other Smart-home Hubs. As discussed in Section 3, we identify a hub by observing the network traffic patterns that are regularly and periodically generated by the hub. While we use Samsung Smartthings Hub in this work, we believe the proposed technique to detect a hub can be transferable and applicable to other smart-home hubs that have noticeable network patterns between the hub and the server, e.g., health check packets or periodic checks of firmware updates.

[2] ↑ refers to traffic from hub. ↓ refers to traffic from server.

Overall, our proposed models can also be trained to detect smart home devices connected to those hubs that have not already implemented any counter-fingerprinting mitigation techniques to their generated network traffic.

Ethical Considerations. The collected data from this work does not contain any personally identifiable information. Our measurement and data collection procedures only recorded network traffic traces related to smart-home devices and the hub. We further consult with Human Research Protection department in our institute and confirmed that the data we collected is not applicable to IRB approval.

7. Related work

There exist a number of studies that fingerprint WiFi-connected smart-home devices by analyzing network traffic generated by each device [46–50]. Other works require tapping into the local network to collect network information from individual devices [51,52].

Pingpong [53] proposed packet-level network traffic analysis to identify activities of smart-home devices directly connected to WiFi network. Pingpong creates unique signatures for smart home devices' activities by separately analyzing network traffic generated by each device. On the other hand, our work targets smart-home devices connected to the smart-home hub device via a personal area network (PAN). We only observe the encrypted network traffic from and to the hub device to identify the status and events generated by each smart-home device. Multiple smart-home devices simultaneously communicate with the hub, and identifying the events generated by each device is more challenging.

HoMonit [22] is a smart-home monitoring system that identifies misbehaving smart apps. It analyzed encrypted network traffic between the hub and smart-home devices to fingerprint each device by tapping the PAN network to collect packets transmitted via ZigBee, ZWave, or BLE. Similarly, Peek-a-Boo [4] captured and analyzed the PAN traffic between devices and a hub to identify the activities of smart-home devices. ChatterHub takes a different approach from theirs. We only monitor network traffic between the smart-home hub and the cloud servers. Thus we do not need any tapping device to capture network traffic via the PAN network.

Zhou et al. [54] investigated security flaws in communications between the smart-home devices and the cloud servers, but they did not focus on identifying smart-home devices' activities.

A number of works focus on the security analysis and improvement for smart-home applications [21,55–60] by finding security vulnerabilities. i.e., private data leakage, privilege abuse, and malicious activities. Other studies focus on the analysis of information flow among smart apps, cloud backend, and IoT devices to discover vulnerabilities in the chain of information transfer [13,61–63].

Apthorpe et al. [45] proposed an approach to mitigate network sniffing attacks in smart homes and suggested routing the network traffic through VPNs and injecting fake packets to confuse the attackers. Yoshigoe et al. [64] proposed to generate synthetic packets that prevent adversaries from fingerprinting smart-home devices. A more sophisticated method using differential privacy and adversarial ML was proposed by [65]. Many studies are proposed to extract information from encrypted network traffic, e.g., extracting video content [61], demographic information [66], detecting packets generated from specific application [67], measuring the quality of service [68], analyzing smart-home tenant behavior [69], and extracting the location information [70]. These works are orthogonal to ChatterHub, and we will incorporate these works to improve the performance of ChatterHub.

8. Conclusion

We present ChatterHub, a novel attack method that can correctly identify the capabilities of smart-home devices with passive sniffing of encrypted home network traffic. With ChatterHub, an attacker does not need prior knowledge of the target home. Our evaluation results with three realistic smart-home environments show that the attacker can successfully recognize the capabilities of smart-home devices from the encrypted network traffic. This, in turn, leads the attacker to discover device behaviors, such as door lock's state change. Such information can be used to reveal a households' daily routines. We also demonstrate two mitigation techniques, *packet padding* and *random sequence injection*, which can effectively protect the smart-home from ChatterHub.

Furthermore, we study the effectiveness of packet dropping attacks, and we demonstrate how an attacker can manipulate the actions of the smart home devices by selectively dropping packets that can put a smart home exposed to physical dangers by disabling security monitors such as motion sensors and door contact sensors and skipping door lock events.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank Muhammed AbuOdeh for his contributions to earlier phase of this work. This material is supported, in part, by National Science Foundation under grant No. OAC-1909856. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] D.Y. Huang, N. Apthorpe, F. Li, G. Acar, N. Feamster, Iot inspector: crowdsourcing labeled network traffic from smart home devices at scale, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 4 (2) (2020) http://dx.doi.org/10.1145/3397333, https://doi.org/10.1145/3397333.
[2] Parks Ass., Evolution of smart home and the internet of things, 2021, https://bit.ly/3bzk9sr.
[3] A. Ng, IoT attacks are getting worse – and no one's listening, 2018, https://tinyurl.com/ydf7ma9b.
[4] A. Acar, H. Fereidooni, T. Abera, A.K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, S. Uluagac, Peek-a-boo: i see your smart home activities, even encrypted!, in: Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '20, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450380065, 2020, pp. 207–218, http://dx.doi.org/10.1145/3395351.3399421, https://doi.org/10.1145/3395351.3399421.
[5] N. Apthorpe, et al., Spying on the smart home: Privacy attacks and defenses on encrypted IoT traffic, 2017, CoRR.
[6] Samsung Elec., SmartThings, 2021, https://www.smartthings.com/.
[7] O. Setayeshfar, K. Subramani, X. Yuan, R. Dey, D. Hong, K.H. Lee, I.K. Kim, Chatterhub: privacy invasion via smart home hub, in: 2021 IEEE International Conference on Smart Computing (SMARTCOMP), 2021, pp. 181–188, http://dx.doi.org/10.1109/SMARTCOMP52413.2021.00045.
[8] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, 25, Curran Associates, Inc., 2012, https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
[9] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation (ISSN: 0899-7667) 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735, arXiv:https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.
[10] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, Y. Zhou, Understanding the mirai botnet, in: 26th USENIX Security Symposium (USENIX Security 17), USENIX Association, Vancouver, BC, ISBN: 978-1-931971-40-9, 2017, pp. 1093–1110, https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.
[11] N. Anh, R. Shorey, Network sniffing tools for wlans: merits and limitations, in: 2005 IEEE International Conference on Personal Wireless Communications, 2005. ICPWC 2005., 2005, pp. 389–393, http://dx.doi.org/10.1109/ICPWC.2005.1431372.
[12] Zigbee Alliance, Zigbee: Securing the wireless IoT, 2017.
[13] O. Alrawi, C. Lever, M. Antonakakis, F. Monrose, Sok: security evaluation of home-based iot deployments, in: 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 1362–1380, http://dx.doi.org/10.1109/SP.2019.00013.
[14] C. Lee, L. Zappaterra, K. Choi, H.-A. Choi, Securing smart home: technologies, security challenges, and security requirements, in: 2014 IEEE Conference on Communications and Network Security, 2014, pp. 67–72, http://dx.doi.org/10.1109/CNS.2014.6997467.
[15] Y. Xiao, S. Sethi, H.-H. Chen, B. Sun, Security services and enhancements in the ieee 802.15.4 wireless sensor networks, in: GLOBECOM '05. IEEE Global Telecommunications Conference, 2005., 3, 2005, pp. 5 pp.–, http://dx.doi.org/10.1109/GLOCOM.2005.1577958.
[16] Apple Inc., Apple home kit, 2021, https://www.apple.com/ios/home/.
[17] Google Inc., Google home, 2021, https://store.google.com/home/hub.
[18] Amazon, Inc, Amazon alexa, 2021, https://amzn.to/3wjppZ7.
[19] Simplisafe Inc, Simplisafe home security solution, 2021, https://simplisafe.com/.
[20] M. Ammar, G. Russello, B. Crispo, Internet of things: a survey on the security of iot frameworks, J. Inf. Secur. Appl. 38 (2018) 8–27.
[21] E. Fernandes, J. Jung, A. Prakash, Security analysis of emerging smart home applications, in: 2016 IEEE Symposium on Security and Privacy (SP), 2016, pp. 636–654, http://dx.doi.org/10.1109/SP.2016.44.
[22] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, H. Zhu, Homonit: monitoring smart home apps from encrypted traffic, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450356930, 2018, pp. 1074–1088, http://dx.doi.org/10.1145/3243734.3243820, https://doi.org/10.1145/3243734.3243820.
[23] K. Meng, Y. Xiao, S.V. Vrbsky, Building a wireless capturing tool for wifi, Security and Communication Networks 2 (6) (2009) 654–668, http://dx.doi.org/10.1002/sec.107, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.107.
[24] Wireshark, 2021, https://www.wireshark.org/.
[25] Simple event logger, 2021, https://bit.ly/3v3GdDg.
[26] V.F. Taylor, R. Spolaor, M. Conti, I. Martinovic, Appscanner: automatic fingerprinting of smartphone apps from encrypted network traffic, 2016 IEEE European Symposium on Security and Privacy (EuroS&P) (2016) 439–454.
[27] Y. Kawahara, M. Sugiyama, Sequential change-point detection based on direct density-ratio estimation, Stat. Anal. Data Min. (ISSN: 1932-1864) 5 (2) (2012) 114–127, http://dx.doi.org/10.1002/sam.10124, https://doi.org/10.1002/sam.10124.
[28] A.W. Gachomo Dorcas Wambui, The power of the pruned exact linear time(pelt) test in multiple changepoint detection, in: American Journal of Theoretical and Applied Statistics. Volume 4, Issue 6, November 2015, 2015.
[29] P. Fryzlewicz, Wild binary segmentation for multiple change-point detection, The Annals of Statistics 42 (6) (2014) 2243 – 2281, http://dx.doi.org/10.1214/14-AOS1245, https://doi.org/10.1214/14-AOS1245.
[30] I. Auger, C. Lawrence, Algorithms for the optimal identification of segment neighborhoods, Bulletin of mathematical biology 51 (1989) 39–54, http://dx.doi.org/10.1007/BF02458835.
[31] R. Killick, P. Fearnhead, I. Eckley, Optimal detection of changepoints with a linear computational cost, Journal of the American Statistical Association 107 (2012) 1590–1598, http://dx.doi.org/10.1080/01621459.2012.737745.
[32] Samsung Electronics, SmartThings public GitHub repo, 2021, https://bit.ly/2S62VvN.
[33] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[34] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, JMLR.org, 2015, pp. 448–456.
[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. (ISSN: 1532-4435) 15 (1) (2014) 1929–1958.
[36] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: ICLR (Poster), 2015, http://arxiv.org/abs/1412.6980.
[37] F. Chollet, et al., Keras, 2021, https://keras.io.

[38] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, Tensorflow: a system for large-scale machine learning, in: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, USENIX Association, USA, ISBN: 9781931971331, 2016, pp. 265–283.
[39] D. Britz, A. Goldie, M.-T. Luong, Q. Le, Massive exploration of neural machine translation architectures, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 1442–1451, http://dx.doi.org/10.18653/v1/D17-1151, https://aclanthology.org/D17-1151.
[40] T. Chen, C. Guestrin, Xgboost: a scalable tree boosting system, KDD '16, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450342322, 2016, pp. 785–794, http://dx.doi.org/10.1145/2939672.2939785, https://doi.org/10.1145/2939672.2939785.
[41] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, J. Peng, Xgboost classifier for ddos attack detection and analysis in sdn-based cloud, in: 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), 2018, pp. 251–256, http://dx.doi.org/10.1109/BigComp.2018.00044.
[42] S.S. Dhaliwal, A.-A. Nahid, R. Abbas, Effective intrusion detection system using xgboost, Information (ISSN: 2078-2489) 9 (7) (2018) http://dx.doi.org/10.3390/info9070149, https://www.mdpi.com/2078-2489/9/7/149.
[43] XGBoost Library, https://xgboost.readthedocs.io/.
[44] A. Gómez-Ríos, J. Luengo, F. Herrera, A study on the noise label influence in boosting algorithms: adaboost, gbm and xgboost, in: F.J. Martínez de Pisón, R. Urraca, H. Quintián, E. Corchado (Eds.), Hybrid Artificial Intelligent Systems, Springer International Publishing, Cham, ISBN: 978-3-319-59650-1, 2017, pp. 268–280.
[45] N.J. Apthorpe, D. Reisman, N. Feamster, Closing the blinds: four strategies for protecting smart home privacy from network observers, ArXiv abs/1705.06809 (2017).
[46] N.J. Apthorpe, D. Reisman, N. Feamster, A smart home is no castle: privacy vulnerabilities of encrypted iot traffic, CoRR abs/1705.06805 (2017) arXiv:1705.06805.
[47] D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, R. Gupta, Z. Durumeric, All things considered: an analysis of iot devices on home networks, in: 28th USENIX Security Symposium (USENIX Security 19), USENIX Association, Santa Clara, CA, ISBN: 978-1-939133-06-9, 2019, pp. 1169–1185, https://www.usenix.org/conference/usenixsecurity19/presentation/kumar-deepak.
[48] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, A.-R. Sadeghi, 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 '19, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450367264, 2019, pp. 128–138, http://dx.doi.org/10.1145/3317549.3323409, https://doi.org/10.1145/3317549.3323409.
[49] L. Deng, Y. Feng, D. Chen, N. Rishe, Iotspot: identifying the iot devices using their anonymous network traffic data, in: MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM), 2019, pp. 1–6, http://dx.doi.org/10.1109/MILCOM47813.2019.9020977.
[50] V. Srinivasan, J. Stankovic, K. Whitehouse, Protecting your daily in-home activity information from a wireless snooping attack, in: Proceedings of the 10th International Conference on Ubiquitous Computing, UbiComp '08, Association for Computing Machinery, New York, NY, USA, ISBN: 9781605581361, 2008, pp. 202–211, http://dx.doi.org/10.1145/1409635.1409663, https://doi.org/10.1145/1409635.1409663.
[51] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, I. Ray, Behavioral fingerprinting of iot devices, in: Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security, ASHES '18, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450359962, 2018, pp. 41–50, http://dx.doi.org/10.1145/3266444.3266452, https://doi.org/10.1145/3266444.3266452.
[52] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, S. Tarkoma, Iot sentinel: automated device-type identification for security enforcement in iot, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 2177–2184, http://dx.doi.org/10.1109/ICDCS.2017.283.
[53] R. Trimananda, J. Varmarken, A. Markopoulou, B. Demsky, Packet-level signatures for smart home devices, in: NDSS, 2020.
[54] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, Y. Zhang, Discovering and understanding the security hazards in the interactions between iot devices, mobile apps, and clouds on smart home platforms, in: 28th USENIX Security Symposium (USENIX Security 19), USENIX Association, Santa Clara, CA, ISBN: 978-1-939133-06-9, 2019, pp. 1133–1150, https://www.usenix.org/conference/usenixsecurity19/presentation/zhou.
[55] Y. Jia, Q.A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z.M. Mao, A. Prakash, Contexlot: towards providing contextual integrity to appified iot platforms, in: NDSS, 2017.
[56] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, A. Prakash, Flowfence: practical data protection for emerging iot application frameworks, in: 25th USENIX Security Symposium (USENIX Security 16), USENIX Association, Austin, TX, ISBN: 978-1-931971-32-4, 2016, pp. 531–548, https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/fernandes.
[57] E. Fernandes, A. Rahmati, J. Jung, A. Prakash, Security implications of permission models in smart-home application frameworks, IEEE Security & Privacy 15 (2) (2017) 24–30, http://dx.doi.org/10.1109/MSP.2017.43.
[58] D. Kumar, R. Paccagnella, P. Murley, E. Hennenfent, J. Mason, A. Bates, M. Bailey, Emerging threats in internet of things voice services, IEEE Security & Privacy 17 (4) (2019) 18–24, http://dx.doi.org/10.1109/MSEC.2019.2910013.
[59] D.T. Nguyen, C. Song, Z. Qian, S.V. Krishnamurthy, E.J.M. Colbert, P. McDaniel, Iotsan: fortifying the safety of iot systems, in: Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '18, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450360807, 2018, pp. 191–203, http://dx.doi.org/10.1145/3281411.3281440, https://doi.org/10.1145/3281411.3281440.
[60] Z.B. Celik, E. Fernandes, E. Pauley, G. Tan, P. McDaniel, Program analysis of commodity iot applications for security and privacy: challenges and opportunities, ACM Comput. Surv. (ISSN: 0360-0300) 52 (4) (2019) http://dx.doi.org/10.1145/3333501, https://doi.org/10.1145/3333501.
[61] Y. Li, Y. Huang, R. Xu, S. Seneviratne, K. Thilakarathna, A. Cheng, D. Webb, G. Jourjon, Deep content: unveiling video streaming content from encrypted wifi traffic, in: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), 2018, pp. 1–8, http://dx.doi.org/10.1109/NCA.2018.8548317.
[62] Y. Jia, Y. Xiao, J. Yu, X. Cheng, Z. Liang, Z. Wan, A novel graph-based mechanism for identifying traffic vulnerabilities in smart home iot, IEEE INFOCOM 2018 - IEEE Conference on Computer Communications (2018) 1493–1501.
[63] S. Marchal, M. Miettinen, T.D. Nguyen, A.-R. Sadeghi, N. Asokan, Audi: toward autonomous iot device-type identification using periodic communication, IEEE Journal on Selected Areas in Communications 37 (6) (2019) 1402–1412, http://dx.doi.org/10.1109/JSAC.2019.2904364.
[64] K. Yoshigoe, W. Dai, M. Abramson, A. Jacobs, Overcoming invasion of privacy in smart home environment with synthetic packet injection, in: 2015 TRON Symposium (TRONSHOW), 2015, pp. 1–7, http://dx.doi.org/10.1109/TRONSHOW.2014.7396875.
[65] X. Zhang, J. Hamm, M.K. Reiter, Y. Zhang, Statistical privacy for streaming traffic, in: NDSS, 2019.
[66] H. Li, Z. Xu, H. Zhu, D. Ma, S. Li, K. Xing, Demographics inference through wi-fi network traffic analysis, in: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9, http://dx.doi.org/10.1109/INFOCOM.2016.7524528.
[67] R. Alshammari, A.N. Zincir-Heywood, Machine learning based encrypted traffic classification: identifying ssh and skype, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–8, http://dx.doi.org/10.1109/CISDA.2009.5356534.
[68] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, K. Papagiannaki, Measuring video qoe from encrypted traffic, in: Proceedings of the 2016 Internet Measurement Conference, IMC '16, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450345262, 2016, pp. 513–526, http://dx.doi.org/10.1145/2987443.2987459, https://doi.org/10.1145/2987443.2987459.

[69] B. Copos, K. Levitt, M. Bishop, J. Rowe, Is anybody home? inferring activity from smart home network traffic, in: 2016 IEEE Security and Privacy Workshops (SPW), 2016, pp. 245–251, http://dx.doi.org/10.1109/SPW.2016.48.

[70] G. Ateniese, B. Hitaj, L.V. Mancini, N.V. Verde, A. Villani, No place to hide that bytes won't reveal: sniffing location-based encrypted traffic to track a user's position, in: M. Qiu, S. Xu, M. Yung, H. Zhang (Eds.), Network and System Security, Springer International Publishing, Cham, ISBN: 978-3-319-25645-0, 2015, pp. 46–59.