ChatterHub: Privacy Invasion via Smart Home Hub

Omid Setayeshfar*§, Karthika Subramani*§, Xingzi Yuan*, Raunak Dey*,
Dezhi Hong†, Kyu Hyung Lee*, and In Kee Kim*
*University of Georgia, Computer Science, {omid.s, ks54471, x.yuan, raunak.dey25, kyuhlee, inkee.kim}@uga.edu
†University of California San Diego, Computer Science and Engineering, dehong@ucsd.edu

2021 IEEE International Conference on Smart Computing (SMARTCOMP) | 978-1-6654-1252-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/SMARTCOMP52413.2021.00045

***Abstract*—Smart-home devices promise to make users' lives more convenient. However, at the same time, such devices increase the possibility of breaching users' privacy as they are tightly connected to the users' daily lives and activities. To address privacy invasion through smart-home devices, we present ChatterHub. This novel approach accurately identifies smart-home devices' activities with minimal monitoring of encrypted traffic in the home network. ChatterHub targets devices that can only connect to the Internet through a centralized smart-home hub (e.g., Samsung SmartThings) using *Zigbee* or *Z-wave*. Specifically, ChatterHub passively eavesdrops on encrypted network traffic from the hub and leverages machine learning techniques to classify events and states of smart-home devices. Using ChatterHub, an adversary can identify smart-home devices' specific activities without prior knowledge of the target smart home (e.g., list of deployed devices, types of communication protocols). We evaluated the accuracy and efficiency of ChatterHub in three real-world smart-home environments, and the evaluation results show that an attacker can successfully disclose smart-home devices' behaviors with over 88% F_1 score. We further demonstrate that ChatterHub successfully recognizes privacy-sensitive activities, including open and close of a smart door lock and turn on and off of smart LED. Additionally, to mitigate the threats posed by ChatterHub, we introduce two approaches, packet padding and random sequence injection. These mitigation approaches can effectively prevent threats from ChatterHub with only 9.2MB of additional network traffic per day.**

***Index Terms*—IoT Hub, Smart Home Privacy, Smart Home Devices, Sniffing Encrypted Network**

I. Introduction

The blooming of the Internet of Things (IoT) promotes massive smart-home devices to become connected to the Internet, with an estimate of 10 smart devices per home on average in 2020 [1]. We expect the number of installed smart-home devices to reach 75 billion by 2025. Smart-home devices promise to make the user's daily life more convenient. According to a recent study [2], the main reason for smart device purchase is convenience, as users can easily control and monitor smart-home devices over the Internet. Most smart-home devices can be accessed via smart apps on smartphones or smart-home platforms. e.g., *"Front door unlocked at 13:52 by code A", "Motion detected in living room at 17:03 "*.

However, this convenience comes at a cost. For example, an adversary with access to smart-home devices' state information (such as what is triggered or used and when), could acquire sensitive information about the users and their activities. These device states often contain the users' activities in their living space, and the adversary can exploit it to commit further offenses, such as burglary and aggravated robbery. Indeed, cybercriminals are increasingly targeting smart-home devices [3]. Recent studies [4], [5] demonstrated privacy invasion problems present in smart-home devices. For example, Peek-a-Boo [4] showed that attackers could identify smart-home devices' states and actions by passively listening to the wireless around a smart-home. Apthorpe et al. [5] showed an Internet Service Provider (ISP) could learn privacy-sensitive information from smart-home devices by analyzing traffic.

This work presents a novel method to attack smart-homes, called ChatterHub, enabling an adversary to infer smart home events and user activities by sniffing *encrypted* network traffic to/from a target home, even though devices are hidden behind a smart-hub (e.g., Samsung SmartThings [6]) and do not directly connect to the Internet. ChatterHub requires neither physical proximity to the target home nor prior knowledge of its setup (e.g., list or topology of smart-home devices), making attacks on smart-homes more feasible.

The intuition behind designing ChatterHub is that users' activity routine in a smart home can trigger smart devices, manifesting as distinct patterns in the network traffic, albeit encrypted, and hence the users' activities and smart devices' events are discoverable and learnable. To infer smart-home devices' events, ChatterHub employs a classification model trained with traffic patterns of popular smart-home devices and hubs. The adversary can further train ChatterHub with their own devices by providing network packet traces and event logs to the training platform. ChatterHub automatically partitions the network trace with our novel segmentation algorithms and feeds the segmented traces (with event labels parsed from the event logs) into machine learning models to detect smart home devices' events. This way, the attacker can infer the occupancy pattern of the home by analyzing the event timing and patterns.

We have evaluated the accuracy and effectiveness of ChatterHub on real-world testbed environments with Samsung SmartThings hub and 14 smart-home devices. The results show ChatterHub can successfully discover the capabilities and events of the devices, e.g., *lock*, *switch*, or *motion* based on their encrypted traffic, and reveal users' daily routines by tracking devices' activity, including changes in lock's state, smart LED's state (i.e., on→off, off→on), and multi-purpose sensor's states (i.e., detecting motion on doors or windows).

In summary, this paper makes the following contributions:

1) We explore a new adversarial approach against smart-home devices hidden behind a smart-hub, which could leak critical user's privacy, including households' daily routine.

§Equal contribution

978-1-6654-1252-0/21/$31.00 ©2021 IEEE
DOI 10.1109/SMARTCOMP52413.2021.00045

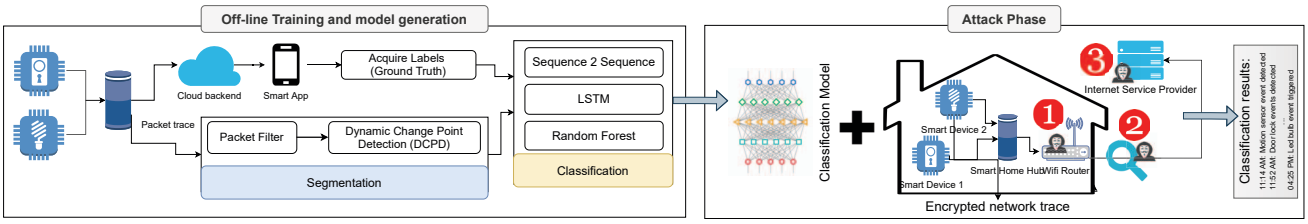

Fig. 1: A high level overview of ChatterHub

2) We design a classification model that can accurately identify the events and usage patterns of various smart-home devices from encrypted network traffic.
3) We evaluate ChatterHub in three real smart-home environments. The evaluation results show that ChatterHub can successfully recognize smart-home devices' events with 88% F_1 score on average.
4) We show that a combination of packet padding and random sequence injection techniques can mitigate threats from ChatterHub at an average cost of 9.2MB traffic per day.
5) All the data sets, source code, and classification models used in this work are publicly available to the community[1].

II. Adversary Model, Assumption, and Goal

We assume that an attacker only passively sniffs encrypted network packets from/to the target home. In this work, we consider three potential points at which the attacker can eavesdrop on network traffic. First, the attacker can gain access to the traffic from a compromised router. Second, the attacker can eavesdrop on network traffic from the home router's uplink traffic. Third, the attacker can be the one who can monitor the network traffic of the target home, Considering these scenarios, encryption remains the only form of protection for users' data. Nonetheless, our adversary model is a passive attacker who collects encrypted network traffic (e.g., TLS/SSL). The attacker can only observe the size of each incoming and outgoing packet, the source and destination IPs, and timestamps. In addition, the attacker does not rely on decoding or interpreting the information inside traffic packets.

We also assume the attacker has access to a trained model or can collect his data from a hub and desired devices to train a model. However, the attacker does not require prior knowledge of a targeted smart-home topology or devices deployed.

The Goal of the Adversary. Once the network packet traces from the target home are obtained, the adversary proceeds to leverage a classification model, provided by ChatterHub or trained on the attacker's own hub and devices. By doing so, the adversary can understand the pattern of network traffic generated by the smart-home devices of interest. 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 [7]. After gaining access to the routers, the attacker analyzes traffic either in the routers or through a virtual redirection to a sniffer installed device. Understanding smart-home devices' behaviors will allow the attacker to find vulnerable targets for a further offensive campaign, such as burglary.
- **Targeted Attack.** The attacker first gains access to network sniffing tools [8] and sniffs the outgoing traffic. After enough scouting, the attacker can understand the smart-home devices' behaviors, identify household activities patterns, and use the patterns for physical assault.
- **ISP-level Tracking.** Internet providers such as ISPs and VPNs who has complete access to users' traffic can learn the patterns of the households' daily life. Such information can be used for targeted advertising based on user behaviors or other activities, potentially violating users' privacy [9].

Target Devices. In general, 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 i.e., *Zigbee*, *Z-wave*, or Bluetooth Low Energy (*BLE*). The first type of device can directly connect to the access point. On the other hand, devices in the second category cannot connect to the Internet directly, so they require a smart-home hub to manage communications among devices. Additionally, since the second type of devices is hidden behind the hub, they are considered to be more secure against remote attackers [10]. A large body of work [5], [11]–[13] studied security and privacy of the first type of devices, while the security of home automation network devices (the second type of devices) has gained little attention. This work focuses on the second type for their high market share and diversity [14], [15].

III. System Design

Minimally intrusive monitoring is the most important goal of ChatterHub as the adversary only requires access to the network traffic from/to home. Obtaining access at this level is ascertained to be relatively simpler compared to using eavesdropping devices that have to be placed near the target devices [5], [16], [17].

Fig. 1 illustrates an overview and the control flow of ChatterHub. In ChatterHub's training, all the communication from the devices are transmitted through the hub. We collect these communication packets through 1) accessing the cloud backend logs and 2) monitoring the network traffic. Network traffic will be passed to a *segmentation* module, which separates network traces into sequences associated with events.

A. Training Data Collection

We first collect network packets to/from our smart-home setup and smart-devices' event logs, then label them for model

[1] https://github.com/karthikaS03/ChatterHub

TABLE I: List of devices and capabilities. Communication is shown by (Zigbee) for Zigbee and (Z-Wave) for Z-Wave. Capability references correspond to Table II

Type	Device	Device Name	Cap.
Sens.	Multi Sensors	Centralite Micro Door Sensor (Zigbee)	9, B, 6
		Smartthings Multipurpose Sensor (Zigbee)	B, A, 6
		Samsung Multipurpose Sensor (Zigbee)	B, A, 6
	7 Sensors	Iris Smart 7 Sensor (Zigbee)	C, 6, 7
		Centralite 7 Sensor (Zigbee)	C, 6, 7
	3 Sensors	Centralite 3 Sensor (Zigbee)	3, 6, C
		Samsung 3 Sensor (Zigbee)	3, 6, C
Act.	Smart Lights	SYLVANIA Smart 10Y A19 TW (Zigbee)	4, 5, 8
		SYLVANIA Smart + Adjust. (Zigbee)	4, 5, 8
		Sengled Element Plus (Zigbee)	4, 5, 8
	Smart Plugs	Centralite Smart Outlet (Zigbee)	4
		Sylvania SMART+ Smart Plug (Zigbee)	4
	2s	Kwikset 10-1 Deadbolt (Z-Wave)	2, C
	Switches	OSRAM LIGHTIFY Dimming 4(Zigbee)	1
Hub	Hub	Samsung SmartThings Hub (Zigbee, Z-Wave)	ping

training. In this work, we used 15 different devices with 12 unique capabilities as described in Table I (list of devices), and Table II (capabilities) shows events associated with each capability. In our dataset, an event is represented as the combination of a capability and its event (e.g., switch-on, lock-unlocked). We used the following setups for model training.

1) **Single device.** We connect a single device to the hub and observe the network traffic generated. This is to understand the unique traffic patterns generated by each device.
2) **Multiple devices.** We connect multiple devices to the hub and monitor traffic concurrently generated by all devices; we use this data to train our model with a more realistic setup. For example, we observed packets (generated by multiple device events) often overlapped each other. We connect not only smart-home devices to the hub, but also other home appliances (e.g., computers, tablets, smartphones) to the router to create more realistic traffic.
3) **Only the hub.** We also observe the network traffic from an isolated hub's (with no other devices attached) operations to understand the hub's behaviors (e.g., firmware update).

We connect `Wireshark` installed on a laptop to Samsung SmartThings hub through a bridged network to monitor the network traffic. We obtain event labels from the logs delivered through the hub. Samsung SmartThings hub stores event logs (e.g., all events and commands sent to/by smart-home devices along with timestamps). We collect the logs regularly by using "Simple Event Logger" [18] provided by the manufacturer. We have collected over 200,000 network packets from the smart-hub with over 60,000 event logs and use them for training the classification model in ChatterHub.

B. Trace Segmentation, Labeling, and Feature Extraction

We first design a method to filter out network packets that are not related to the SmartThings hub (i.e., packets generated by PCs or tablets), and then we perform packet segmentation.

TABLE II: 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`

Packet Segmentation. When the hub is registering, 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 is then connected to *Comm-Server* for further communications and operations. This communication channel between the hub and *Comm-Server* remains established, and hub relays the information through this channel. Suppose the devices communicate with *Comm-Server* directly via Wi-Fi. In that case, traffic is segmented based on each session, and each device's traffic can be separated based on their unique destination and source IP addresses. However, the challenge we encounter is that all communications go through the hub, and there is a lack of discerning parameters. Thus, it is not possible to partition packets based on the network flow information. Also, the communication interval in the sequence of packets between two events is relatively large compared to the interval between packets sent for a single event. Thus, we apply a segmentation method to divide the network traces into small bursts of packets. To segment the network flows into separate bursts, we try to leverage approaches from previous studies [16], [19] that use a fixed threshold of 4.5 seconds to segment network packets into multiple bursts. Previous works show that 4.5 seconds is enough for the communication between a client and server to complete packets exchange. However, we observed that the time gap between packet exchange for a single event could last longer than 4.5 seconds. Fig. 2 shows a case where a fixed-threshold approach fails to separate the *level change* event from other events. Also, events of the hub (e.g., ping, status) can occur along with other device events within an interval of shorter than 4.5 seconds. As such, the segmentation based on a fixed threshold often fails to correctly segment the device events from other packets (e.g., ping and status). Therefore, we develop a dynamic segmentation technique using *change point detection* [20] to segment these packets into bursts correctly.

Dynamic Change Point Detection (CPD). A *change point* is a temporal point when the statistical properties of its previous and subsequent time points are different. In our smart-home

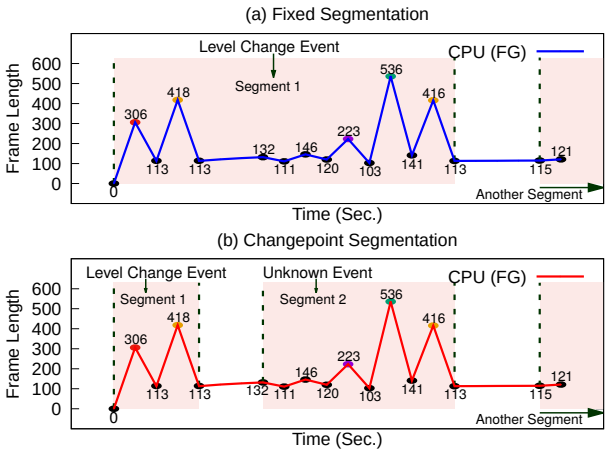

Fig. 2: Fixed Segmentation vs. Changepoint Segmentation

setup, the network packets for a single event are issued in short intervals compared to the intervals between two distinct events. Therefore, a change point will be when a sequence of packets for a single event starts or ends. Since our logs are collected over a long time, multiple change points need to be identified to segment all events. CPD is an approach to find abrupt changes in time-series [20]. CPD can also be used for estimating the temporal point when the statistical properties of a sequence change [21]. ChatterHub employs PELT (Pruned Exact Linear Time) [21] because it is computationally efficient and outperforms other exact CPD search methods [22]. We present detailed evaluation results of Dynamic CPD and fixed threshold segmentation algorithms on our dataset in §IV-A.

Labeling. After we segment the packets from network traces into different bursts, we obtain event labels from the hub's logs. We use timestamps to align the labels and the segmented trace. However, we observe that slight time differences between *generation of event log* and *packet capture* can occur. Hence, we allow for ±5-the second difference between the two; then, we map the event to a specific burst of packets. We also observe special cases where *a single user activity enables multiple events in a device*. For example, a "switch on" user event from the app triggers two events (switch-on and level-change). Therefore, a single burst of packets could be mapped to multiple events. 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 that we label them as *unknown*. To characterize the unknown packets, we further analyzed the source code of device handlers [23] and found that the handler generates the event logs, and some of the handlers do not emit any logs. We found that most of the missing events are less important for the user (e.g., device refresh, device ping).

Feature Extraction. For feature extraction, we begin by forming a signature via fetching the frame length of multiple packets in each segment. We then use this signature as the feature for our classifier.

These signatures show a significant amount of collision across different classes. These collisions are the result of events happening in small intervals or events that some happen together, e.g., when a user opens a door, both contact-open and status-open events occur concurrently.

C. Classification Models

We train classification models using extracted network features to classify smart home network traffic into smart home devices' capabilities and events. Given the dynamic nature of the data, we consider the following machine learning models: 1) Random Forest, 2) OneVsRest classifier, and 3) SEQ2SEQ.

Random Forest (RF) Model. RF constructs an ensemble of decision trees by taking a random subset of the features to decide a node split in building each tree. We only use RF as a baseline to identify better algorithms because RF largely depends on the training data's completeness.

OneVsRest Classifier. A key characteristic of our data is that a single traffic segment may contain the data related to multiple capabilities that usually occur together or were subsequently activated. Therefore, we need a multi-class classifier that can identify all the classes in segmented traffic. As a result, we follow the *one-vs-rest strategy* that uses a classifier for each class fitted against all other classes. 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 XGBoost (Extreme Gradient Boosting) . XGBoost is an ensemble that applies Gradient boosting on decision trees to boost the performance of the various models [24]–[26]. In this project, we use the `XGBClassifier` of `XGBoost` library [27] with its default parameters. We use `CountVectorizer` as vectorizer, with `ngram` range from 1 to 4 so that the relationship between the packets in the sequence is maintained. The output of the vectorizer is directly fed into the XGBoost model.

SEQ2SEQ Model. Sequence-to-sequence (SEQ2SEQ) model solves sequential problems. The input to SEQ2SEQ is a series of data units, and the output is also a sequence of data units [28]. SEQ2SEQ model is applied to address various problems in multiple disciplines. Specifically, SEQ2SEQ caught our attention because of its application in natural language translation, for which the input is usually a sentence and the output is a sentence in a different language. In our model, we have a sequence of package lengths, and the output is a sequence of *events*. We use sequences of capabilities and events as labels. It is worth noting that SEQ2SEQ is a model to translate natural languages, so the order of the sequence will affect the result. We maintain the original order from ground truth, even if one label appears multiple times. The SEQ2SEQ framework contains two main components: an *encoder* and a *decoder*. The encoder reads the input, and the decoder translates the encoder's output to a final sequence of outputs [28].

IV. Evaluation Results

We evaluate ChatterHub with real-world smart-home environments. In the smart-home setup, we deploy a set of smart-home devices and other Internet-connected devices (e.g., laptops, smartphones), and then we connect them to the hub.

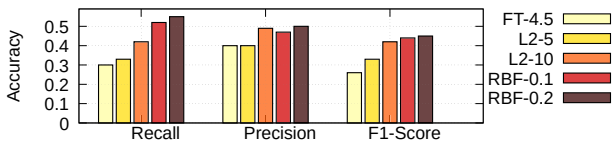

Fig. 3: Performance Evaluation of various Segmentation Methods. (FT-4.5: Fixed Threshold with 4.5 seconds, L2-5: L2 Cost Penalty-5, L2-10: L2 Cost Penalty-10, RBF-0.1: RBF Cost Penalty - 0.1, RBF-0.2: RBF Cost Penalty - 0.2)

A. Network Trace Segmentation

We perform trace segmentation on the captured traffic to partition the overall traffic flow between the hub and cloud servers (e.g., *Comm-Server*) into a set of small bursts, which map to specific commands. Therefore, ChatterHub first needs to identify the IP address of the target hub, and then it performs network trace segmentation, which will generate a set of proper packets related to a specific command/event from the devices at a time. An accurate segmentation will ensure that each packet burst contains a negligible amount of noise packets. Note that noise or noisy packets indicate *unknown* packets or packets for the hub's status report. The hub randomly sends these packets to the cloud servers.

Identifying the IP address of the Hub. We monitor all network traffic from and to the target home router and identify the hub's IP using the pattern signature of "hub's ping" events. While the hub keeps changing the IP address of *Comm-server* from time to time (usually over days), we can successfully identify the IP address of *Comm-Servers*. Then, we can extract necessary traffic between the hub and *Comm-Server* (excluding the traffic from other devices in the home).

Network Trace Segmentation. As we discussed in §III-B, we develop a PELT-based Dynamic Change Point Detection (CPD) algorithm to segment the network traffic.

The PELT algorithm can be used with different cost functions, and it takes the output of the cost function as a penalty value, which affects the segmentation results. We compare the two most dominant cost functions, least squared deviation (L2) and kernalized mean change with radial basis function (RBF) kernel by running their output through our baseline model. Fig. 3 shows the results of classification for different parameters. We use PELT (RBF cost function and penalty value of 0.2), which achieved the best F_1 and precision.

B. Evaluation in Smart-home Environments

To evaluate ChatterHub in the real world, we set up three smart-home environments at three homes along with other devices and record the network traffic from their home router for a total of 10 days.

We train the classification models with data collected from the lab setting (explained in §III-A) plus the data obtained from one of three home configurations. We then test the model on data from two remaining smart homes not used for training.

After the model training, we conduct two experiments; 1) an attacker tries to infer the capabilities of devices, and 2) an attacker tries to detect specific events of those capabilities. For example, the attacker will be made aware of "switch" being present and used in the first experiment's target home. The attacker will then infer if a "switch on" or "switch off" has happened in the second experiment.

TABLE III: Classification results for capabilities and events

Capabilities	Random Forest		SEQ2SEQ		XGBoost	
	R.	F_1	R.	F_1	R.	F_1
button-held	0.00	0.00	0.00	0.00	0.27	0.18
button-pushed	0.00	0.00	0.61	0.57	0.98	0.73
colorTemperature	0.23	0.37	0.17	0.27	1.00	0.96
contact-closed	0.25	0.32	0.29	0.30	0.40	0.44
contact-open	0.37	0.45	0.50	0.47	0.47	0.54
switchLevel	0.87	0.42	0.63	0.33	0.89	0.55
lock-locked	0.71	0.50	0.77	0.46	0.77	0.53
lock-unlocked	0.12	0.17	0.06	0.10	0.77	0.68
motion-active	0.08	0.12	0.10	0.13	0.48	0.17
motion-inactive	0.28	0.23	0.30	0.25	0.62	0.36
ping-ping	0.96	0.98	0.96	0.98	1.00	0.99
status-closed	0.49	0.57	0.50	0.52	0.69	0.80
status-open	0.60	0.63	0.80	0.66	0.77	0.74
switch-off	0.58	0.71	0.55	0.52	0.71	0.80
switch-on	0.13	0.17	0.29	0.18	0.32	0.38
temperature	0.63	0.25	0.07	0.10	0.77	0.37
unknown	0.59	0.73	0.94	0.92	0.95	0.90
F_1 Known Average	0.83	0.72	0.78	0.81	0.92	0.76
F_1 Average	0.69	0.73	0.88	0.88	0.93	0.82

It is worth noting that when we test the classification models, we add more sensors and devices (e.g., water sensor), which do not exist in the training dataset, to two test smart-homes to test the scenario where the attacker does not have a list of installed devices in the target home.

Classification Accuracy: We generate the ground truth for two different sets of labels (capabilities and events) so that we can train our classification models on both data sets to classify capabilities and events separately. Table III reports the classification accuracy (recall, and F_1-score) of events from each device, such as switch-on, switch-off, motion-active and motion-inactive. If some devices in the target home have not been used in the model training, ChatterHub categorizes the events and capabilities belonging to this device as *unknown*. This is also observed from our results in the case of *water* capability, as shown in Table IV; where "0" for water sensor activities means water sensor was not used while training.

Overall, our classifiers generate multi-label outputs, indicating that a single segment of traffic packets can be classified into more than one class. Thus, our models identify multiple activities happening concurrently without an explicit time gap in the transmission of the network packets. The classification results reported in Table III are the accuracy of each class. To decide the models' overall performance, we calculate the micro average score for F_1 and recall (μAvg) [29], which takes into consideration the imbalanced class sizes. μAvg calculates a F_1-score across different classes by adding up their

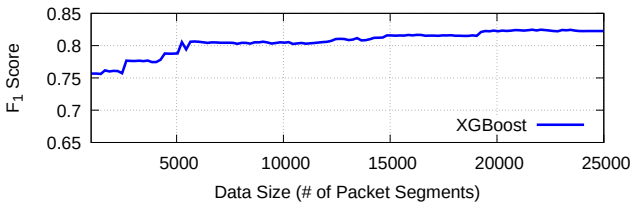

Fig. 4: Effect of training set size to the F_1-score (XGBoost)

respective confusion quadrants. This shows how a multi-class model considers the whole class list to reduce bias towards their underlying class distribution. The average of F_1-score is calculated by $\frac{2\times\sum tp_i}{2\times\sum tp_i+\sum fp_i+\sum fn_i}$, where tp, tn, fp, fn indicate true positive, true negative, false positive, and false negative, respectively. fp_i denotes the number of false positives for the i^{th} class. We report this result as `Average` in Table III. However, since the focus of our system is to detect *known* device activities, we calculate the μAvg for only the *known* classes and exclude *unknown* classes from the computation. The average results are reported as `known-average`.

Among three classification models, RF (the baseline model) shows the lowest recall, and precision III. Overall, SEQ2SEQ gives us the highest F_1-score (0.81) for *known-average*, compared to the XGBoost model's F_1 result of 0.76. Although the XGBoost model has a higher individual F_1-score for some of the capabilities and events, the average F_1-score is lower because of higher false positive cases resulting in lower precision score. On the other hand, SEQ2SEQ shows higher precision results, indicating that SEQ2SEQ's accurate performance for identifying the events of devices. Based on this observation, this limitation of SEQ2SEQ in identifying some activities is in overlapped packet sequences. But XGBoost is more resilient to such noise in the data [30]. Hence, it shows higher accuracy in the presence of overlapped of packets. However, XGBoost's misclassification is a result of signature conflicts between multiple activities from a same device.

Effect of Training Data Size. As this attack relies on the model to accurately classify traffic, we further analyze the impact of training data size on a fixed set of test data classification accuracy. Fig. 4 shows how XGBoost F_1-score changes as the size of training data grows. The results show the accuracy increases with the size of training data. And, decent accuracy is possible with fewer training data.

C. In-Depth Analysis of Smart-Home Results

Our threat model is based on attackers' capabilities to monitor network traffic in a smart home to infer the smart home devices' activities and the user's behavior. Therefore, while our efforts are to create a model that works best in all scenarios, we demonstrate our model is useful for attackers to identify private information about the user and her home correctly. In this section, we explain such cases in detail.

Target Classification Model for Specific Devices. We discuss how an attacker can use a specific model to obtain more accurate information on a targeted device from its capabilities. In this case, we train our XGBoost model only to detect three

TABLE IV: Classification results w/ and w/o a specific (Lock) device in different home setups. (R: Recall, P: Precision)

Capability	Case #1 : Home w/ Lock			Case #2 : Home w/o Lock		
	OneVsRest – XGBoost					
	R	P	F_1	R	P	F_1
Contact	0.88	0.35	0.50	0.78	0.45	0.57
Lock	**0.94**	**1.00**	**0.97**	-	-	-
Switch	0.76	0.94	0.84	0.61	0.85	0.71
Unknown	0.99	0.99	0.99	0.99	0.98	0.99
Average	0.90	0.82	0.83	0.80	0.76	0.75

capabilities (contact, lock, and switch). To this end, we use a single trained XGBoost model and design two different evaluation test sets with intentionally deploying different devices (e.g., lock). In the first case setup (case #1), we create a smart-home testbed with all devices, including lock and water. In the second case (case #2), we create another smart-home testbed with all devices, including water except the *lock*. As shown in Table IV, in case #1, our model was able to detect *lock* with a precision of "1.00" indicating the model has no false-positives. Moreover, even if our model was trained on signatures of *lock*, in case #2 (the testbed without *lock* device), no lock capability was detected. Therefore, our model is highly accurate in detecting the presence of the devices.

Identifying Recurring Patterns. Fig. 6 shows the activities of a smart lock at various times of day. We measured the device's activities for 5 consecutive days. The results show that at 11:00 and at 23:00, the lock had the events on multiple days at the same time. Based on this observation, the attacker can infer the homeowner's daily schedule. Hence, with further analysis of such patterns, the classification results could reveal information on the smart-home devices and the users.

Another example is the *switch-on/off* events reported in Table III. F_1-scores of these events by XGBoost are 0.38 (*switch on*) and 0.80 (*switch-off*). Although F_1-scores are less than 0.8, ChatterHub can still identify user actions with light switches (e.g., user turning lights on/off). Fig. 5 shows ChatterHub correctly identifies 20 out of 25 events. ChatterHub only has three misclassifications (i.e., event *on* recognized as *off*, and vice versa) and two false detections (i.e., non-switch events recognized as switch events but part of the switch device itself). Further, the patterns of *on/off* events provide more confidence in the actual presence of a smart light in the home.

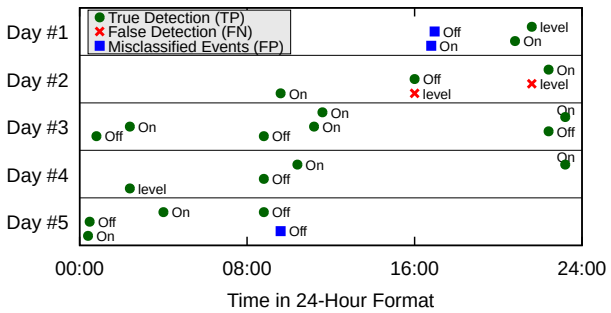

Fig. 5: `Switch` events detected by ChatterHub.

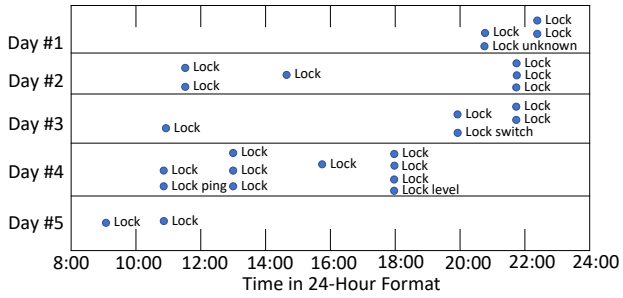

Fig. 6: `Lock` events detected by ChatterHub.

V. Discussions

Mitigation Approach. ChatterHub identifies the events of devices by monitoring encrypted packets, including the size of each packet, the order of packets in a sequence, and the timing of sequences. Packet padding [31] is an intuitive and effective mitigation method against ChatterHub. It generates packets with identical lengths by adding additional bytes at each packet's end, i.e., padding. Packet padding can effectively hinder ChatterHub and other similar attack methods. We implement packet padding in our testing router, and it pads each packet in a sequence to 1KB. We evaluate packet (space) overhead caused by the padding with three traces collected from our testbed. The result shows that only a negligible amount of traffic is generated by this method (on average, 9.2MB per day). Furthermore, we develop a random sequence insertion method for diluting the effect of sequence timing by irregularly generating 1Kb packets to the network. When deploying both packet padding and random sequence insertion methods, an additional 10MB of network traffic is additionally generated per day, and more than 80% collision is observed in the classification process; thus, the attacker will have a very low chance of learning patterns from the network traffic.

Overlapped Packets. As we discussed in §III-A, the overlapping of packet sequences is one of the major challenges to accurate classification. Suppose the target home has a larger number of smart-home devices than our experiment setup. In that case, there will be more chances for overlapping of packet sequences, implying that ChatterHub's classification results can be less accurate. However, our setup conservatively constitutes realistic smart-home setups as we deploy many devices (14+) that repeatedly generate network traffic, so we believe there will be minimal impact on the classification accuracy with more devices. Another potential limitation is when the target home has multiple devices of the same type, ChatterHub cannot tell which one contributes to the detected capability. For example, if the target home has two identical smart lock devices installed on two separate doors, the attacker would be able to recognize all the lock activities but cannot distinguish one lock from the other.

VI. Related Work

Most of the research works that follow fingerprinting smart-home devices from network traffic focus on independent devices that directly connect to WiFi [32]–[37] unlike our setup where devices are connected to a central device (hub). These works also requires tapping to the local network for information on individual devices [38], [39] unlike our threat model where the network tapping can be acquired remotely.

Pingpong [40] proposes packet-level network traffic analysis to identify activities of smart-home devices. Similar to ChatterHub, Pingpong analyzes packet-level traffic to create unique signatures for smart home devices' activities. However, they only study WiFi-connected devices, but our focus is smart-home devices hidden behind the hub, increasing the complexity of network traffic. We observe that many security-critical devices (e.g., smart lock, motion sensor, smart switch) are hidden behind the hub to be more secure.

HoMonit [16] is a smart-home monitoring system that identifies misbehaving smart apps. They have analyzed encrypted network traffic between the hub and smart-home devices to fingerprint each device. However, it requires tapping into the network between the hub and the devices. Similarly, Peek-a-Boo [4] focuses on capturing the traffic between devices and a hub. However, our work focuses on the communication between a hub and the cloud servers (e.g., *Auth-Server* and *Comm-Server*). Due to hub devices' inter-operability, fingerprinting devices by analyzing the hub and server communications becomes complicated and difficult, compared to the encrypted traffic analysis done on HoMonit. Also, Zhou et al. [41] investigated potential security flaws in communications between the smart-home devices and the cloud servers, but this work does not focus on identifying smart-home devices' activities inferred from encrypted traffic.

A number of works focus on the security analysis and improvement for smart-home applications [15], [42]–[47] where they discovered 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 [11], [48]–[50].

While existing solutions mainly focus on preventing the leak of sensitive data from the context of smart apps, cloud backend, and/or the smart-home platform, this work demonstrates that an adversary can still infer activities and states of smart-home devices by eavesdropping on encrypted network traffic.

Apthorpe et al. [31] proposed an approach to mitigate network sniffing attacks in a smart-home environment and suggested routing the network traffic through VPNs and injecting fake packets to confuse the attackers. Yoshigoe et al. [51] proposed to generate synthetic packets that prevent adversaries from fingerprinting smart-home devices. A more sophisticated method using differential privacy and adversarial machine learning has been suggested by [52]. There are many studies to extract information from encrypted network traffic, such as extracting video content [48], demographic information [53], detecting packets generated from specific application [54], measuring the quality of service [55], analyzing smart-home tenant behavior [56], and extracting the location information [57]. Also, there are approaches [58], [59] to build multipurpose tools to facilitate analysis of encrypted network traces. These works are orthogonal to ChatterHub.

VII. Conclusion

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

VIII. Acknowledgments

We thank the anonymous reviewers and Muhammed AbuOdeh for their helpful feedback. This material is supported, in part, by the 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 and et al., "IoT Inspector: Crowdsourcing Labeled Network Traffic from Smart Home Devices at Scale," *IMWUT*, 2020.
[2] Parks Ass., "Evolution of Smart Home and the Internet of Things," https://bit.ly/3bzk9sr, 2020.
[3] A. Ng, "IoT Attacks are Getting Worse – and No One's Listening," https://tinyurl.com/ydf7ma9b, 2018.
[4] A. Acar, H. Fereidooni, and et al., "Peek-a-Boo: I See Your Smart Home Activities, Even Encrypted!" *CoRR*, vol. abs/1808.02741, 2018.
[5] N. Apthorpe and et al, "Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic," *CoRR*, 2017.
[6] Samsung Elec., "SmartThings," https://www.smartthings.com/, 2020.
[7] M. Antonakakis and et al., "Understanding the Mirai Botnet," in *USENIX Security*, 2017.
[8] N. T. Anh and R. Shorey, "Network Sniffing Tools for WLANs: Merits and Limitations," in *IEEE ICPWC*, 2005.
[9] "State of Push Notification Advertising," https://bit.ly/3hFf0CW, 2020.
[10] Zigbee Alliance, "Zigbee: Securing the Wireless IoT," 2017.
[11] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "SoK: Security Evaluation of Home-Based IoT Deployments," in *IEEE S&P*, 2019.
[12] C. Lee and et al., "Securing Smart Home: Technologies, Security Challenges, and Security Requirements," in *IEEE CNS*, 2014.
[13] Y. Xiao and et al., "Security Services and Enhancements in the IEEE 802.15.4 Wireless Sensor Networks," in *IEEE GLOBECOM*, 2005.
[14] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *J. Info. Sec. and App.*, vol. 38, 2018.
[15] E. Fernandes, J. Jung, and A. Prakash, "Security Analysis of Emerging Smart Home Applications," in *IEEE S&P*, 2016.
[16] W. Zhang, Y. Meng, and et al., "HoMonit: Monitoring Smart Home Apps from Encrypted Traffic," in *ACM CCS*, 2018.
[17] K. Meng, Y. Xiao, and S. V. Vrbsky, "Building a Wireless Capturing Tool for WiFi," *Security and Communication Networks*, 2009.
[18] "Simple Event Logger," https://bit.ly/3v3GdDg, 2020.
[19] V. F. Taylor and et al., "AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic," in *Euro S&P*, 2016.
[20] Y. Kawahara and M. Sugiyama, "Sequential Change-Point Detection Based on Direct Density-Ratio Estimation," in *SDM*, 2009.
[21] G. Wambui and et al., "The Power of the Pruned Exact Linear Time (PELT) Test in Multiple Changepoint Detection," in *AJTAS*, 2015.
[22] R. Killick and et al., "Optimal Detection of Changepoints with a Linear Computational Cost," in *J. American Statistical Association*, 2012.
[23] S. Electronics, "SmartThings Public GitHub Repo," https://bit.ly/2S62VvN, 2020.
[24] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *ACM KDD*, 2016.
[25] Z. Chen and et al, "XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud," in *IEEE BigComp*, 2018.
[26] S. S. Dhaliwal, A.-A. Nahid, and R. Abbas, "Effective Intrusion Detection System Using XGBoost," *Information*, vol. 9, no. 7, p. 149, 2018.
[27] "XGBoost Library," https://xgboost.readthedocs.io/.
[28] D. Britz and et al., "Massive Exploration of Neural Machine Translation Architectures," *CoRR*, vol. abs/1703.03906, 2017.
[29] V. Van Asch, "Macro-and Micro-Averaged Evaluation Measures," *Belgium: CLiPS*, vol. 49, 2013.
[30] A. Gómez-Ríos and et al., "A Study on the Noise Label Influence in Boosting Algorithms: AdaBoost, GBM and XGBoost," in *HAIS*, 2017.
[31] N. Apthorpe and et al., "Closing the blinds: Four strategies for protecting smart home privacy from network observers," *CoRR*, 2017.
[32] M. R. Shahid and et al., "IoT Devices Recognition Through Network Traffic Analysis," in *IEEE BigData*, 2018.
[33] N. Apthorpe, D. Reisman, and N. Feamster, "A Smart Home Is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic," *CoRR*, 2017.
[34] D. Kumar and et al., "All Things Considered: An Analysis of IoT Devices on Home Networks," in *USENIX Security*, 2019.
[35] T. J. OConnor and et al., "Homesnitch: behavior transparency and control for smart home iot devices," in *ACM WiSec*, 2019.
[36] L. D. and et al., "IoTSpot: Identifying the IoT Devices Using their Anonymous Network Traffic Data," in *MILCOM*, 2019.
[37] V. Srinivasan and et al, "Protecting your daily in-home activity information from a wireless snooping attack," in *UbiComp*, 2008, pp. 202–211.
[38] B. Bezawada and et al, "IoTSense: Behavioral Fingerprinting of IoT Devices," *CoRR*, 2018.
[39] M. Miettinen and et al, "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT," in *ICDCS*, 2017.
[40] R. Trimananda and et al, "Packet-Level Signatures for Smart Home Devices," in *NDSS*, 2020.
[41] W. Zhou and et al., "Discovering and Understanding the Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms," in *USENIX Security*, 2019.
[42] Y. J. Jia and et al., "ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms." in *NDSS*, 2017.
[43] E. Fernandes and et al., "FlowFence: Practical Data Protection for Emerging IoT Application Frameworks," in *USENIX Security*, 2016.
[44] ——, "Security Implications of Permission Models in Smart-home Application Frameworks," *IEEE Secur. & Priv.*, vol. 15, no. 2, 2017.
[45] D. Kumar and et al., "Emerging Threats in Internet of Things Voice Services," *IEEE Secur. & Priv.*, vol. 17, no. 4, 2019.
[46] D. T. Nguyen and et al., "IotSan: Fortifying the Safety of IoT Systems," in *ACM CoNext*, 2018.
[47] Z. Celik and et al., "Program Analysis of Commodity IoT Applications for Security and Privacy: Challenges and Opport." *ACM CSUR*, 2019.
[48] Y. Li and et al., "Deep Content: Unveiling Video Streaming Content from Encrypted WiFi Traffic," in *IEEE NCA*, 2018.
[49] Y. Jia and et al., "A Novel Graph-based Mechanism for Identifying Traffic Vulnerabilities in Smart Home IoT," in *IEEE INFOCOM*, 2018.
[50] S. Marchal and et al., "AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication," *IEEE JSAC*, 2019.
[51] K. Yoshigoe and et al, "Overcoming Invasion of Privacy in Smart Home Environment with Synthetic Packet Injection," in *TRONSHOW*, 2015.
[52] X. Zhang, J. Hamm, M. K. Reiter, and Y. Zhang, "Statistical Privacy for Streaming Traffic." in *NDSS*, 2019.
[53] H. Li and et al., "Demographics Inference Through Wi-Fi Network Traffic Analysis," in *IEEE INFOCOM*, 2016.
[54] R. Alshammari and et al., "Machine Learning Based Encrypted Traffic Classification: Identifying ssh and skype," in *CISDA*, 2009.
[55] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring Video QoE from Encrypted Traffic," in *ACM IMC*, 2016.
[56] B. Copos and et al., "Is Anybody Home? Inferring Activity From Smart Home Network Traffic," in *IEEE S&P Workshops*, 2016.
[57] G. Ateniese and et al., "No Place to Hide That Bytes Won't Reveal: Sniffing Location-based Encrypted Traffic to Track a User's Position," in *NSS*, 2015.
[58] E. Papadogiannaki and et al., "OTTer: A Scalable High-Resolution Encrypted Traffic Identification Engine," in *RAID*, 2018.
[59] M. Lotfollahi and et al., "Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning," *Soft Comp.*, 2020.