Securing App Behaviors in Smart Home: A Human-App Interaction Perspective

Jinlei Li*, Yan Meng*, Lu Zhou*, Haojin Zhu *†
*Shanghai Jiao Tong University
†Shanghai Institute for Advanced Communication and Data Science
{ricardolee, yan_meng, zl19920928}@sjtu.edu.cn, zhu-hj@cs.sjtu.edu.cn

2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS) | 978-1-7281-9074-7/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICPADS51040.2020.00049

***Abstract*—Smart home has become a mainstream lifestyle due to the maturity of the IoT platform and the popularity of smart devices. While offering great convenience and entertainment, smart home suffers from malicious attacks that inject improper commands and actions to home devices, which may breach the user's safety and privacy. Traditional solutions mainly focus on generating security policies relying on app analysis to constraint apps' behaviors. However, these policies lack flexibility to adapt to the highly dynamic smart home system. We need to consider not only the app behaviors but also the user behaviors for enforcing an appropriate security policy. In this study, we propose WIPOLICY, a cross-layer security enforcement system for smart home by monitoring the behaviors of both apps and users. The key novelty of WIPOLICY is incorporating user activity recognition via the physical-layer wireless signals into the definition and enforcement of security policies to constraint the app behavior. We implement WIPOLICY on the Samsung SmartThings platform with 187 SmartApps, and 24 behavior policies are defined and enforced. The case study demonstrates the effectiveness of WIPOLICY on thwarting app's misbehavior.**

***Keywords*-smart home; wireless sensing; static analysis; human-app interaction; guide policy**

I. INTRODUCTION

With the rapid development of the Internet of Things (IoT) technology, smart home is becoming a popular system that offers great convenience and entertainment to users in their daily lives. In recent years, the smart home consumer market has developed rapidly. According to the prediction of Statista [1], the global market size of the smart home will reach $53.45 billion in 2022. To promote compatibility between different vendors, some major players in the market have developed smart home platforms, including Samsung SmartThings [2], Apple HomeKit [3], and Google Nest [4]. These platforms encourage software developers to develop applications with a unified abstraction of smart devices.

With the popularity of smart home platforms, their security vulnerabilities have recently received increasing attention. For example, a remote attacker may hijack the application *Play The Music* to disturb the user via Internet. Existing research efforts mainly focus on malicious applications [5] and system design flaws [6], [7], or rely on app analysis to constraint apps' behaviors [8], [9]. However, most of the existing security solutions fail to take the human factor into consideration. These methods are not appropriate to cope with the highly dynamic home environment where the important users behaviors are hardly detected and often ignored.

We take user behaviors into consideration of constructing app behaviors policy. Note that, in many real-world smart home applications, human behavior plays an essential role in determining if an event is violating a user's intention. For example, playing music at a speaker appears to be a benign application, but it becomes malicious if the speaker plays music suddenly and loudly while the user is sleeping. It is obvious that user behavior is an important factor, and the security policy defined based on user behavior is highly desired for securing the emerging smart home system.

In this study, we propose WIPOLICY, a novel security system for smart home by considering the behaviors of both users and apps. Specifically, WIPOLICY utilizes the physical-layer wireless signal to recognize the user activity and adopts the user activity information into the application-layer policy definition and enforcement.

First, in the application layer, WIPOLICY performs a static program analysis to extract necessary application information, including devices and actions. WIPOLICY classifies all possible actions and formulates apps' behaviors policy related to user activity. In addition, in the physical layer, WIPOLICY utilizes wireless signals to sense and recognize user activities in real-time and enforces the corresponding policy on apps to enhance the safety and security of the smart home environment.

The contributions of this study are summarized as follows:

- We propose WIPOLICY, a novel cross-layer system to define and enforce policies for app behaviors with the user behavior info as preconditions in the smart home.
- We elaborate on the design of WIPOLICY, and propose a human activity inference method via Wi-Fi signal analysis and a policy generation method via static analysis.
- We perform comprehensive evaluations on the Samsung SmartThings platform. The results show that WIPOLICY can successfully define and enforce 24 app behaviors policies. In the case study, we launch two typical apps in a real-world environment with both normal and attack cases for 30 times, respectively, and WIPOLICY

Haojin Zhu is the corresponding author.

2690-5965/20/$31.00 ©2020 IEEE
DOI 10.1109/ICPADS51040.2020.00049

authorize them with only one failure case.

The remainder of this paper is organized as follows. In Section 2, we introduce the background and related works. Section 3 discusses the research motivation and key insights. We elaborate on the detailed design of WIPOLICY in Section 4, which is followed by evaluation and discussion in Section 5 and 6, respectively. Finally, we conclude this paper in Section 7.

II. BACKGROUND AND RELATED WORK

A. Smart Home System

A smart home system usually contains two components: the physical layer and the application layer, as shown in Fig. 1. In the physical layer, multiple sensors and devices collect physical status and send events to a hub. The hub is usually used as a centralized gateway to connect devices in the physical environment and communicate with the application layer. In the application layer, cloud service is used to synchronize device status and provide interfaces for remote control. The smart home platforms are responsible for providing app-specific services by managing devices and their actions.

These platforms encourage software developers to develop applications with a unified abstraction of smart devices. As one of the most popular smart home platforms, Samsung SmartThings provides many attractive features and abstractions of various smart devices, thereby fostering a vibrant software market. So far, SmartThings supports 133 device types and 187 SmartApps in the official GitHub repository [10]. SmartThings cloud hosts SmartApps[1] and device handlers, which are virtual representations of physical devices and their specific capabilities.

B. Smart Home Security Flaws

Researchers have identified many security flaws in off-the-shelf devices for smart home [11], [12], [13]. Smart devices tend to suffer from emerging unauthenticated control signals. For example, an adversary can control a smart TV with a speaker playing synthetic voice commands [14]. Ho *et al.* [15] described how a Bluetooth smart lock can unlock mistakenly due to improper trust.

Besides, Fernandes *et al.* [5] recently revealed several design flaws in the design of SmartThings that allowed malicious applications to disrupt the platform. The corresponding defense methods include information flow control [6], context-based permission systems [7], source code verification [8], and user-centered authorization and enforcement mechanisms [9]. Celik *et al.* [16] utilized code instrumentor to store the app information in a dynamic model and enforce relevant IoT security policies. Zhang *et al.* [17] leveraged side-channel inference capabilities to monitor SmartApps from encrypted wireless traffic.

[1]Smart home platform has a different name convention for applications. In this study, we use the Samsung SmartThings term SmartApp to describe third-party applications, including Amazon's skills and Google's actions.

C. Channel State Information (CSI)

In this paper, we consider the Wi-Fi wireless communication protocol which is widely applied by many smart devices. Wi-Fi standards like IEEE 802.11n/ac are designed to significantly improve the channel capacity of the wireless system [18]. For a Wi-Fi communication system with single antenna pair, due to Orthogonal Frequency Division Multiplexing (OFDM) mechanism, the Wi-Fi packet is transmitted via N_s subcarriers simultaneously. CSI characterizes Channel Frequency Response (CFR) in different subcarriers. For a given subcarrier, the CSI value H can be defined as:

$$H = |H|\, e^{j\angle H} = \alpha e^{-j2\pi f \tau}, \tag{1}$$

where α is the signal magnitude attenuation, f is the frequency and τ is the time-of-light.

From Equation 1, the human movement will change the propagation paths of Wi-Fi signal, thus cause CSI fluctuation. Recent studies demonstrate it is feasible to leverage CSI to sense human activity. Shi et al. [19] and Meng et. al [20] showed that existing Wi-Fi signals generated by indoor IoT devices can be utilized to achieve user authentication based on body activities. Qian et al. [21] and Wang et al. [22] demonstrated using Wi-Fi signals could achieve human localization and tracking with centimeter-level precision.

III. MOTIVATION

In this section, we elaborate the rationale behind WIPOLICY by showing an example.

Threat Scenarios. The lack of proper policy enforcement in the smart home causes lots of issues. Fig. 2 shows a popular app (*e.g.*, "play the music"), which plays pre-defined music when the user gives a command in a smartphone. However, when the user is sleeping, and her device is hijacked by an attacker, this app may play a pre-collected voice command to unlock the door, or play a strange sound for frightening the user. The traditional solutions mainly focus on app analysis and platform authorization. However, in this case, since the app's working logic (*i.e.*, receiving an event $\rightarrow$ play corresponding sounds) looks normal, and the platform is separated from the physical world, these methods cannot achieve good results.

Insights. However, from the perspective of human-app interaction, this malicious behavior could be easily detected. Human behavior usually plays an essential role in determining if an event is violating a user's intention. For example, playing a sound when the user is sleeping or far away from home is suspicious. In this study, the insight of WIPOLICY is incorporating user activity recognition into apps' behavior policies. Related works show that it is feasible to utilize wireless signals to sense and recognize user activities in real time. In addition, WIPOLICY can extract the events and

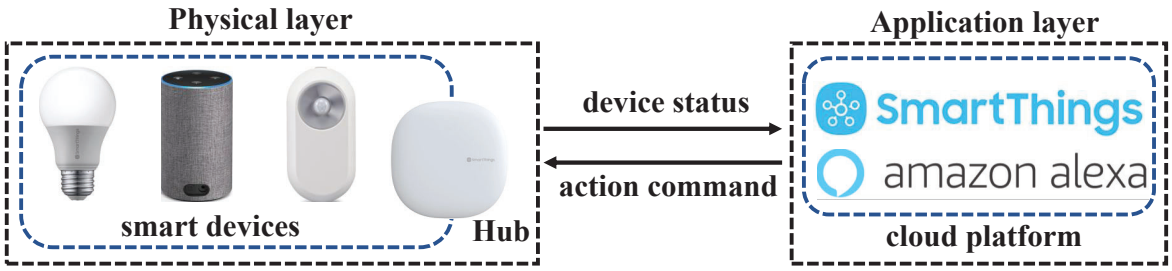

Figure 1. Smart Home Platform.

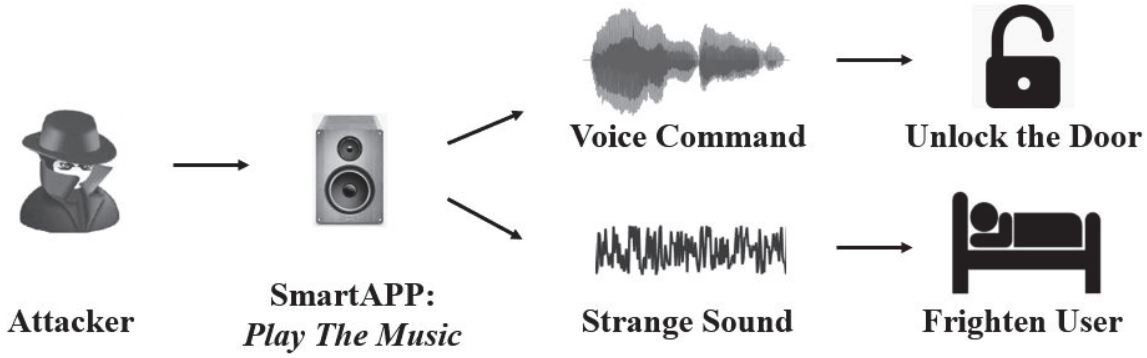

Figure 2. A typical scenario in smart home.

actions of each smart app through static program analysis. For all possible actions of the apps, WIPOLICY will formulate corresponding behavior policies according to different user activities, enhancing the safety and security of multiple smart home scenarios.

It is worth mentioning that the design of WIPOLICY does not need to change the current SmartThings infrastructure or modify the SmartApps. In this study, only open-source SmartApps that are transparent for us are considered. It is important to note that auditing the working logic of SmartApp is an important research topic, which is worthy of separate research [7], [9], [17], and thus is out of the scope of this work.

IV. DESIGN

A. Overview

The basic strategy of WIPOLICY is to incorporate the CSI based user activity recognition into the construction progress of apps' behaviors policy. WIPOLICY will formulate policies through static program analysis. In the smart home environment, since common platforms control smart devices through wireless signals, it is technically feasible to take advantage of these existing wireless infrastructures to collect the CSI data related to user activities.

As shown in Fig. 3, WIPOLICY consists of the following three modules: *APP Behavior Building Module*, *Human Activity Inference Module*, and *Misbehavior Checking Module*.

B. APP Behavior Building

In this module, WIPOLICY characterizes SmartApp logic and generates appropriate policies.

SmartApp analysis. The purpose of this part is to capture event-action control dependency of SmartApps through static program analysis. Events are trigger conditions of applications, such as "when the temperature is greater than a threshold". Actions are commands specified by SmartApps. For example, turning a switch on or off.

Fig. 4 shows a sample source code for SmartApp. Since SmartApps are written in Groovy, AstBuilder[2] is used to extract their logic to perform a static analysis. WIPOLICY converts the source code of the SmartApp into an Abstract Syntax Tree (AST) during the Groovy compilation phase. Then, WIPOLICY extracts the capabilities from the *preferences* section (Fig. 4, line 6), which is designed to let the user set the appropriate devices and thresholds. Specifically, the *input* methods (Figure 4, lines 8 and 11) of the *preferences* section are scanned to extract the capabilities requested by the SmartApp. When the device status changes, SmartApps use *subscribe* methods to request notifications. These notifications will trigger the *handler* methods to react to these status changes. To further determine a particular action, WIPOLICY scans the *subscribe* methods and their

[2]Available on http://docs.groovy-lang.org/next/html/gapi/org/codehaus/groovy/ast/builder/AstBuilder.html

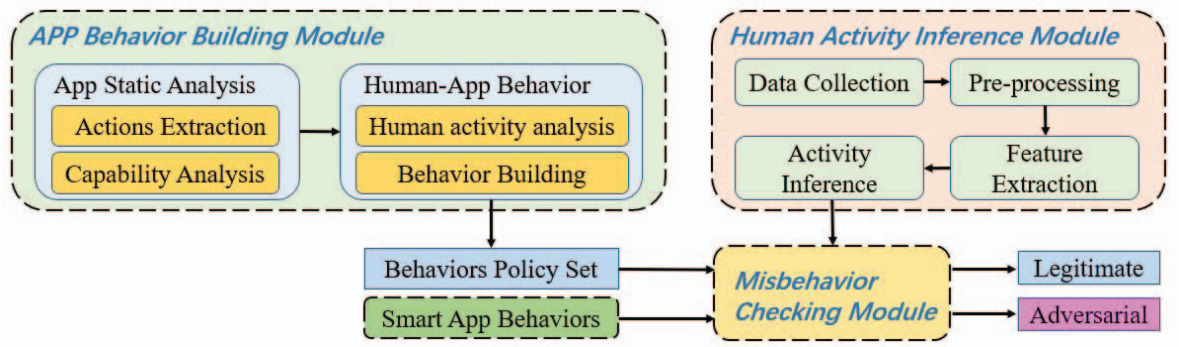

Figure 3. Workflow of WIPOLICY.

```
1  definition(
2      name: "Brighten My Path", namespace: "smartthings", author: "SmartThings",
3      description: "Turn your lights on when motion is detected.", category: "Convenience"
4  )
5
6  preferences {
7      section("When there's movement...") {
8          input "motion1", "capability.motionSensor", title: "Where?", multiple: true
9      }
10     section("Turn on a light...") {
11         input "switch1", "capability.switch", multiple: true
12     }
13 }
14
15 def installed() {
16     subscribe(motion1, "motion.active", motionActiveHandler)
17 }
18
19 def updated() {
20     unsubscribe()
21     subscribe(motion1, "motion.active", motionActiveHandler)
22 }
23
24 def motionActiveHandler(evt) {
25     switch1.on()
26 }
```

Figure 4. A SmartApp code example: *Brighten My Path.*

corresponding functions, *installed* and *updated* (Figure 4, lines 16 and 21).

Behaviors policy. By classifying different capabilities, we can divide actions into 8 categories, respectively. In the process of formulating behavior policies for various SmartApps, we mainly focus on the different permissions of different action categories. Table I shows their typical associated capabilities or devices.

SmartApp behaviors policies represent the physical behavior norms that users expect from smart home security. In practice, the security and safety requirements in the smart home environment are dependent on its unique application background, which needs special tailoring.

Different action categories will apply to different policies, which are summarized in Table II. WIPOLICY divides the user activities into three categories, namely absence, do-not-disturb (DND), and motion. Absence means that no user is in the smart home environment. DND means that the user does not want to be disturbed, such as sleeping or concentrating on work. Motion means that the user is in a non-static state, such as moving from one room to another. In Table II, *Authorized* means that the SmartApp can be used normally, and *Rejected* means that WIPOLICY will reject the execution of actions in the corresponding user activity. It

Table I
THE ACTION CATEGORIES AND TYPICAL ASSOCIATED CAPABILITIES OR DEVICES.

Category	Associated capability or device
Temperature	Thermostat, Heater, Fan, AC
Humidity	Humidifier, Vent_fan, Valve
Light	Light, Dimmer, Color_control
Voice	Music_player
Alarm	Alarm
Lock	Lock, Garage_door
Switch	Switch, Camera
	Coffee_machine, Pet_feeder
Mode	Mode

Table II
POLICIES FOR DIFFERENT ACTION CATEGORIES.

	Absence	DND	Motion
Temperature	Rejected	Authorized	Authorized
Humidity	Specific	Authorized	Authorized
Light	Rejected	Specific	Authorized
Voice	Rejected	Specific	Authorized
Alarm	Authorized	Authorized	Authorized
Lock	Rejected	Rejected	Authorized
Switch	Specific	Specific	Authorized
Mode	Rejected	Authorized	Authorized

should be noted that when the policy is *Specific*, WIPOLICY will make decisions relying on additional information[3].

C. Human Activity Inference

This subsection shows the procedures of human activity inference. As shown in Fig. 3, WIPOLICY detects human activity by the following four steps: *Data Collection, Pre-processing, Feature Extraction and Activity Inference*.

Data collection. In smart home platforms, to collect the CSI data, WIPOLICY leverages smart devices with Wi-Fi capabilities. More specifically, for a given antenna pair, the Wi-Fi packets are sent by transmitter antenna with a fixed rate r, then CSI data $H_{M\times N_s}$ are extracted by receiver antenna from the preamble sequences of $M = \tau \times r$ packets, where τ and N_s are the transmission time and subcarriers number respectively.

Pre-processing. As shown in the 1st column of Fig. 5, there are high noises in each subcarrier. To remove these noises and facilitate the further recognition, we utilize the wavelet-based de-nosing and principal component analysis (PCA) on the original CSI data $H_{M\times N_s}$.

Firstly, for the i-th subcarrier $H(:,i)$, we choose Daubechies D4 wavelet and 2-level decomposition to get the de-noising waveform, and represent the whole processed CSI as $B_{M\times N_s}$. Then, to extract the strongest correlation component with human activity, PCA is applied to the $B_{M\times N_s}$ as below:

[3]For example: when the user is absent, the policy on Humidity is usually rejected, but the vent fan will be authorized when sensing water leak. For Light and Voice, the user can set the time range (such as 0:00-6:00) of rejection in DND state, and allow execution at other times. The user can freely set which device switches can be turned on in Absence and DND.

$$P_{M\times N_s} = B_{M\times N_s} \times E_{N_s\times N_s}(B^T_{M\times N_s} \times B_{M\times N_s}), \quad (2)$$

where E_{N_s,N_s} is the sorted Eigenvector matrix of Covariance Matrix of B_{M,N_s}. We choose the first component $P(:,1)$ as the final processed value.

Feature extracting. As shown in the second column of Fig. 5, it is observed that the waveforms of processed CSI among absence, DND, and motion modes are quite different. To extract useful features, we first split the pre-processed data $P(:,1)$ into multiple chunks C. During splitting, the moving time window method with a window length of 2 seconds and a time step of 0.5 seconds is chosen. For the i-th chunk $C(i)$, we perform fast Fourier transform (FFT) to obtain its frequency domain spectrum F. As shown in the third and fourth columns of Fig. 5, the spectrum during different user activity modes are quite different. In the motion scenario, high power components of the spectrum are located in the lower frequency (0 - 5 Hz). However, in the absence scenario, the spectrum power is decentralized. The spectrum of DND mode is between that of absence and motion. In this study, we use the low-frequency power's proportion s as the feature of the chunk $C(i)$:

$$s = \frac{\sum_{j=1}^{f_c} F(j)}{\sum_{j=1}^{j=f_s/2} F(j)}, \quad (3)$$

where f_c is the threshold of low-frequency, and f_s is the sampling rate of CSI. In this study, f_c and f_s are set to 5 Hz and 1000 Hz respectively. As shown in Fig. 5, the proportion s is quite different among different modes, which demonstrates its effectiveness.

Activity Inference. After we get the power proportion s from all chunks, we chose the average of them as the final score S. Then, the threshold based classification method is deployed in activity recognition as below:

$$Mode = \begin{cases} absence, & S < Thr_1 \\ motion, & S > Thr_2 \\ DND, & others \end{cases} \quad (4)$$

where $Thr1$ and $Thr2$ are two thresholds to determine the activity mode. In this study, the thresholds are set empirically as mentioned in [17], [23], [24].

D. Misbehavior Checking

In this work, WIPOLICY combines the app behaviors policy set with the result of human activity inference. If the action of an app fails to pass a policy, WIPOLICY will reject it. An app is authorized to execute actions if and only if all policies are passed. Another alternative solution is to provide users with an interface to approve each rejected action. However, this approach is less secure for users who tend to ignore warnings. Therefore, WIPOLICY focuses on directly blocking the action that violates policies.

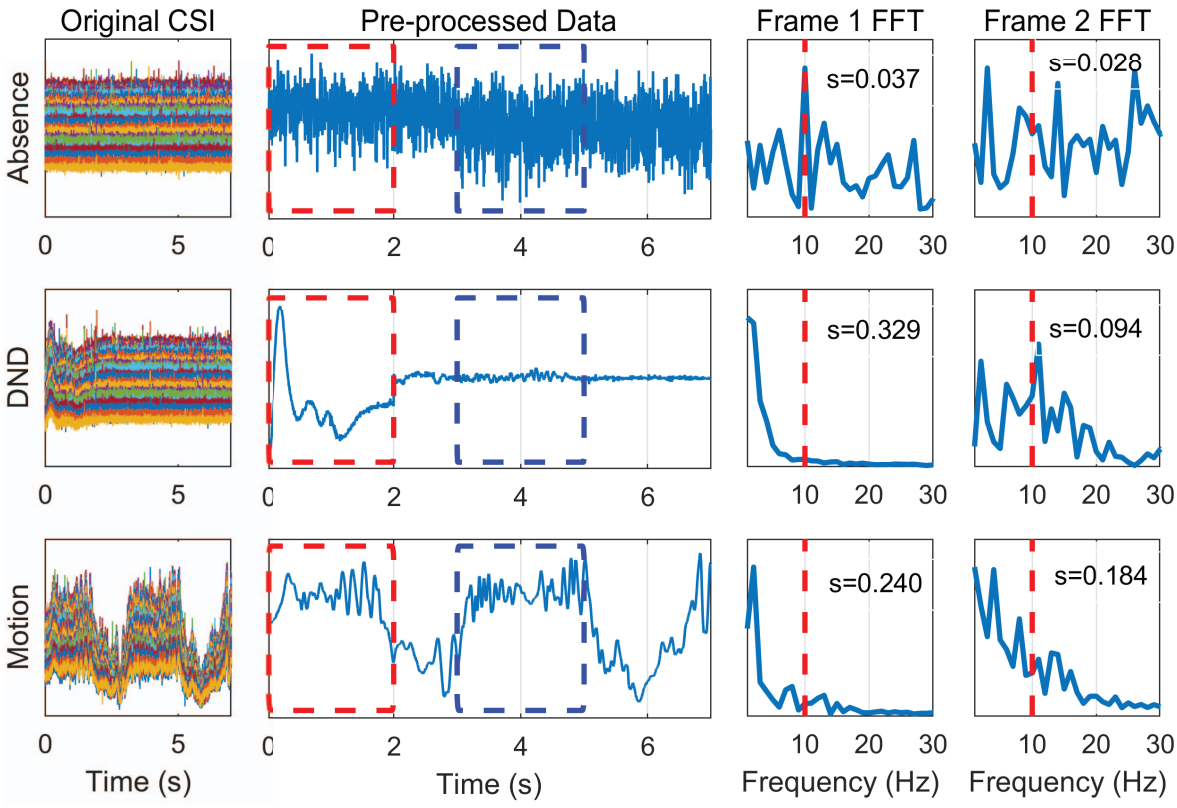

Figure 5. Illustration of CSI based activity inference. The 1st and 2nd columns represent the CSI data pre-processing. The 3rd and 4th columns illustrate the feature extraction from two CSI frames marked by red and blue rectangles.

V. Evaluation

In this section, we conduct an evaluation on the performance of WiPolicy in multiple aspects and implement WiPolicy in a real-world environment.

A. SmartApp analysis

Among the 187 SmartThings applications, we successfully extract event-action pairs from 154 (accounting for 82.4%) applications. The reason that the rest smartapps fail to analyze is that they are mostly used to interact with other platforms or connect to a specific smart device, and their names usually end with *-connect* or *-control*. It is worth mentioning that there are 42 (accounting for 22.5%) apps that only push a notification to the platform or send a text message to the designated mobile phone, which do not involve any actions associated with other smart device. In this step, we extract a total of 256 event-action relationship information.

Fig. 6 shows the amount of extracted actions with respect to different categories. It is obvious that the actions belonging to category Light and Voice are the most, appearing 78 and 57 times respectively. The number of SmartApps containing actions belonging to categories Humidity (5 times) and Alarm (10 times) is relatively small, which is because there are fewer types of corresponding smart devices.

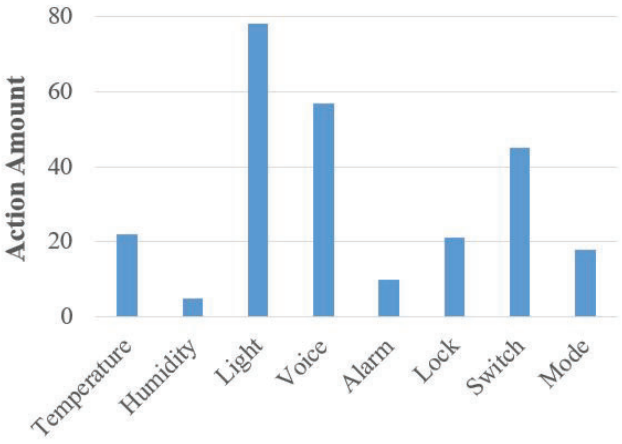

Figure 6. The amount of extracted actions.

B. Activity Inference Performance

To evaluate WiPolicy's performance on activity inference, we require the volunteer to perform each activity for

Table III
ACTIVITY INFERENCE RESULTS.

Threshold (Thr_1, Thr_2)	Accuracy (%)			
	Absence	DND	Motion	Overall
(0.14, 0.04)	72%	94%	96%	87.3%
(0.14, 0.03)	72%	100%	94%	88.7%
(0.13, 0.03)	88%	98%	94%	93.3%
(0.12, 0.03)	**100%**	**98%**	**94%**	**97.3%**
(0.1, 0.03)	100%	94%	94%	96%
(0.1, 0.015)	100%	94%	84%	92.7%

Table IV
CASE STUDY RESULTS.

	Absence	DND	Motion	Reject	Pass
Speaker Weather Forecast	Rejected	Rejected	Authorized	**20/20**	**10/10**
Keep Me Cozy	Rejected	Authorized	Authorized	**9/10**	**20/20**

50 times as described in Section 4.2. The final activity inference accuracy are shown in Table III. When Thr_1 and Thr_2 are set to 0.12 and 0.03 respectively, the accuracy of detecting motion, DND and absences are 100%, 98% and 94% respectively, and the overall accuracy is 97.3%. The CSI data are extracted by Universal Software Radio Peripheral (USRP) and processed by MATLAB R2018b with Intel i7-7700HQ CPU and 8GB RAM. The average time overhead for recognizing one activity is 0.12 second.

C. Real-world Case Study

In this subsection, we explore the implementation of WIPOLICY on the SmartThings platform in real-world environment. WIPOLICY communicates as a virtual device and continuously infers the user's activities in real time by analyzing the collected CSI data. When at a certain set time, the SmartApp *Speaker Weather Forecast* applies to execute and start a smart speaker, and the App *Keep Me Cozy* turns on the thermostat. WIPOLICY checks the corresponding policies with the user activity status. This SmartApp will be authorized to execute if and only if all policies are passed.

For each SmartApp, we recruit volunteers to perform each activity for 10 times while launching this SmartApp. For *Speaker Weather Forecast*, all samples have been processed correctly. For *Keep Me Cozy*, 1 sample in the absence passes unexpectedly. This is because when Thr_1 and Thr_2 are set to 0.12 and 0.03 respectively, one absence activity was wrongly recognized as DND.

VI. DISCUSSION

The evaluation part demonstrates the effectiveness of WIPOLICY. However, there are still some limitations. The distance between the user and the antennas affects the performance. When the distance is too long (depending on the hardware condition), some user activities may be wrongly recognized. In practice, deploying multiple antennas on a smart home and dynamically choosing them can make the performance of WIPOLICY more stable. Besides, user's pets may affect the recognition accuracy. More optimized threshold may help distinguish pets actions from user actions.

In this study, only open-source SmartApps that are transparent for us are considered. This limitation is similar to related work [9], [16], [25]. As one of the most popular smart home platforms, SmartThings encourages software developers to develop open source applications and integrates them into the official GitHub repository, which makes it easy for researchers to obtain source code.

In addition, though this work mainly considers SmartApps in the SmartThings, the presented approach can be potentially applied to other smart home platforms. When we analyze sophisticated apps in other platforms such as IFTTT, the activity categories (*i.e.*, Absence, DND, and Motion) in this work are no longer sufficient. Related works show that it is feasible to utilize wireless signals to sense and recognize more user activity status. Besides, other communication protocols such as ZigBee and Z-Wave can also be considered. We leave detecting more fine-grained activities for future work.

VII. CONCLUSION

In this paper, we propose WIPOLICY, a cross-layer enforcement system for smart home by monitoring the behaviors of both apps and users. WIPOLICY incorporates user activity recognition into apps' behavior policies from the perspective of human-app interaction. WIPOLICY can work directly with existing SmartThings platforms and is extensible to similar platforms. Results show that WIPOLICY thwarts misbehavior of smart home apps with high accuracy.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (No. 61672350, No. 61972453, No. 61722203) and China Scholarship Council (201906230162).

REFERENCES

[1] Statista, "Forecast market size of the global smart home market," 2020. [Online]. Available: https://www.statista.com/statistics/682204/global-smart-home-market-size/

[2] Samsung, "Smartthings," 2019. [Online]. Available: https://www.smartthings.com

[3] Apple, "Homekit," 2020. [Online]. Available: https://www.apple.com/shop/accessories/all-accessories/homekit

[4] Google, "Nest," 2018. [Online]. Available: https://www.nest.com

[5] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 636–654.

[6] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging iot application frameworks," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 531–548.

[7] Y. J. Jia, Q. A. Chen, S. Wang, A. Rahmati, E. Fernandes, Z. M. Mao, A. Prakash, and S. J. Unviersity, "Contexlot: Towards providing contextual integrity to appified iot platforms." in *Network and Distributed System Security Symposium (NDSS)*, 2017, pp. 1–15.

[8] Z. B. Celik, P. McDaniel, and G. Tan, "Soteria: Automated iot safety and security analysis," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018, pp. 147–158.

[9] Y. Tian, N. Zhang, Y.-H. Lin, X. Wang, B. Ur, X. Guo, and P. Tague, "Smartauth: User-centered authorization for the internet of things," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 361–378.

[10] SmartThings, "Smartthings public github repo," 2019. [Online]. Available: https://github.com/SmartThingsCommunity/SmartThingsPublic

[11] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, and 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)*, 2019, pp. 1133–1150.

[12] S. Manandhar, K. Moran, K. Kafle, R. Tang, D. Poshyvanyk, and A. Nadkarni, "Towards a natural perspective of smart homes for practical security and safety analyses," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1166–1183.

[13] Y. Zhu, Z. Xiao, Y. Chen, Z. Li, M. Liu, B. Y. Zhao, and H. Zheng, "Et tu alexa? when commodity wifi devices turn into adversarial motion sensors," in *Network and Distributed System Security Symposium (NDSS)*, 2020, pp. 1–15.

[14] B. Michele and A. Karpow, "Demo: Using malicious media files to compromise the security and privacy of smart tvs," in *IEEE Consumer Communications and Networking Conference (CCNC)*, 2014, pp. 1144–1145.

[15] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity internet of things devices," in *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2016, pp. 461–472.

[16] Z. B. Celik, G. Tan, and P. D. McDaniel, "Iotguard: Dynamic enforcement of security and safety policy in commodity iot." in *Network and Distributed System Security Symposium (NDSS)*, 2019, pp. 1–15.

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

[18] "IEEE Std. 802.11n-2009: Enhancements for higher throughput." http://www.ieee802.org, 2009.

[19] C. Shi, J. Liu, H. Liu, and Y. Chen, "Smart user authentication through actuation of daily activities leveraging wifi-enabled iot," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2017, pp. 5:1–5:10.

[20] Y. Meng, H. Zhu, J. Li, J. Li, and Y. Liu, "Liveness detection for voice user interface via wireless signals in iot environment," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–16, 2020.

[21] K. Qian, C. Wu, Z. Yang, Y. Liu, and K. Jamieson, "Widar: Decimeter-level passive tracking via velocity monitoring with commodity wi-fi," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2017, pp. 6:1–6:10.

[22] J. Wang, H. Jiang, J. Xiong, K. Jamieson, X. Chen, D. Fang, and B. Xie, "Lifs: Low human-effort, device-free localization with fine-grained subcarrier information," in *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2016, pp. 243–256.

[23] Y. Meng, Z. Wang, W. Zhang, P. Wu, H. Zhu, X. Liang, and Y. Liu, "Wivo: Enhancing the security of voice control system via wireless signal in iot environment," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '18, 2018, p. 81–90.

[24] Y. Meng, W. Zhang, H. Zhu, and X. S. Shen, "Securing consumer iot in the smart home: Architecture, challenges, and countermeasures," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 53–59, 2018.

[25] Q. Wang, W. U. Hassan, A. Bates, and C. Gunter, "Fear and logging in the internet of things," in *Network and Distributed System Security Symposium (NDSS)*, 2018, pp. 1–15.