Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8022033, 11 pages
https://doi.org/10.1155/2022/8022033

Research Article

HomeGuardian: Detecting Anomaly Events in Smart Home Systems

Xuan Dai,[1] **Jian Mao**,[1,2] **Jiawei Li**,[1,2] **Qixiao Lin**,[1,2] **and Jianwei Liu**[1]

[1]*School of Cyber Science and Technology, Beihang University, 37 Xueyuan Road, Haidian District, Beijing, China 100191*
[2]*Beihang Hangzhou Innovation Institute Yuhang, Xixi Octagon City, Yuhang District, Hangzhou, China 310023*

Correspondence should be addressed to Jiawei Li; daweix@buaa.edu.cn

Received 29 April 2022; Accepted 23 May 2022; Published 13 June 2022

Academic Editor: Yan Huang

As a typical application of Internet of Things (IoT), home automation systems, namely, smart homes, provide a more convenient and intelligent life experience through event recognition, automation control, and remote device access. However, smart home systems have also given rise to new complications for security issues. As an event-driven IoT system, smart home environments are vulnerable to security attacks, and vulnerable devices are far-spread due to the quick development cycles. Attack vectors to smart homes inevitably manifest in abnormal event contexts. In this paper, we propose HomeGuardian, a context-based approach to identify abnormal events in smart homes. In our approach, we extract temporal context and environmental context from system logs, aggregate (embed) these hybrid contexts, and construct a learning-based classifier to identify the abnormal events. We develop a testbed to implement and evaluate our approach.

1. Introduction

Smart home, as a ubiquitous computing IoT application in a home environment [1–3], provides remote control and automation services for home users. Quick development cycles of smart devices lead to an increasing expansion of the smart home market scale [4–6].

However, smart home industry develops rapidly without neither a unified security standard nor a unified supervision mechanism, and the inconsistency leads to many security problems. Once the devices are accessed illegally by attackers, users' privacy suffers severe leakage [7, 8]. Moreover, if the smart home hub is accessed by attackers, he/she may obtain control privileges (e.g., door locks) and implement data interception or workflow interference. SmartThings exposes over 20 vulnerabilities in its hub [9]. In addition to hardware vulnerabilities, malicious software in smart homes also results in security and privacy issues. Malicious smart home applications can steal private information by obtaining nonnecessary permissions [10]. In addition to controlling devices directly, physical interactions and automation rules also introduce security risks in smart homes [11, 12]. These security issues will inevitably lead, directly or indirectly, to abnormal device state changes (i.e., events).

Since abnormal events are the most intuitive manifestation of abnormalities in the visible aspect, researchers focus on analyzing event sequences (i.e., device state changes) in the smart home. Hidden Markov Model (HMM) is widely used to analyze sequences for abnormal event detection in smart homes [13]. Event correlation analysis is also applied in smart home scenarios [14, 15]. However, these methods only take into account sequence order without considering specific timing information of smart home events. Besides, in IoT systems, smart devices interact with each other through automation rules and physical effects. Therefore, the states of surrounding devices also should be considered to validate the device behaviors.

To solve the above problems, we propose an anomaly detection system based on event context. In our approach, we take two types of context into account, *temporal* context and *environmental* context.

The *temporal* context of a candidate event is based on the event's time intervals and the history device states. We

further predict the successive events as the *temporal* context to feed HomeGuardian. The *environmental* context for an event is the states of the devices related to the event, which indicates the physical context of the candidate event. Specifically, HomeGuardian analyzes correlations among devices and selects the states of devices that are highly correlated to the candidate event as its *environmental* context features. Given the two contexts, HomeGuardian then implements a difference-based anomaly detection, by inspecting whether the input event is expected, considering the two contexts.

To sum up, we make the following contributions:

(i) We propose HomeGuardian, a context-based approach to identify abnormal events in a smart home system. In our approach, we extract temporal context and environmental context from system log, aggregate (embed) these hybrid contexts, and construct a learning-based classifier to identify the abnormal events

(ii) We develop a self-configured testbed based on the Home Assistant platform to implement and evaluate our approach. According to real-world smart home scenarios, we connect virtual devices with the real hardware environment and configure automation rules to simulate/generate smart home event data

(iii) We evaluate the effectiveness of HomeGuardian using the system log captured from our self-developed testbed. The experiment results illustrate that the F1-scores of HomeGuardian to detect abnormal events are above 0.90 for all device types

2. Background and Problem Statement

2.1. Security Problems of Smart Home. Smart device state changes triggered by automation rules or remote control often cause security risks. For instance, user-setup rules can be triggered accidentally. Figure 1 illustrates how an attacker opens a window and breaks in when nobody is home via triggering rule IF *Temperature* > 25 THEN *open* the *window*. Moreover, a smart home may catch fire if heating devices (e.g., ovens and electric heaters) are turned on by abnormally triggered automation rules.

Vulnerable smart devices may be controlled by an attacker remotely to launch abnormal events or distort device states [9]. The related attacks include network penetration, firmware backdoor exploitation [16], replay attacks [17, 18], and man-in-the-middle attacks [19, 20]:

(1) http.//[Router_IP]/…&SystemCmd = [Malicious_ Code]&…

(2) Heater. off ⟶ on

(3) Temperature sensor. 23 ⟶ 27

(4) If (*temperature* > 25) then window. $off \longrightarrow on$

In addition to devices, smart applications (apps for short) also contribute to system vulnerabilities. Without the knowledge of users' environment settings and behavioral patterns [21], it is challenging for smart apps to precisely define a condition, such as "at home," because the hardware/software settings and user behavioral patterns (e.g., wake-up time, bedtime, and time to take a shower) vary in homes.

2.2. Problem Analysis. To detect anomalies in smart home systems, most existing approaches focus on program analysis for platforms and apps while overlooking device interactions, which are leveraged in real-world attacks like the scenario in Figure 1. Hence, additional contexts such as device states are required for robust and noninvasive anomaly detection.

As shown in Table 1, the correlations of state changes (i.e., events) include objective environmental changes, user behavior patterns, influence between devices, and automation rules for system configuration in a smart home system. Automation rules and smart apps manipulate devices to meet user demands. Besides, a device can be affected by a physical channel between another device. For example, an air conditioner has an impact on an adjacent thermometer. User behavior and environmental changes (e.g., day-night cycle or season alternation) can also cause periodic changes in states of thermometers, hygrometers, and other devices.

However, it is challenging to extract environmental correlation in a smart home system. Static-analysis-based methods cannot capture physical environmental interactions among devices. IoTMon [11] analyzes the description of IoT apps to recognize common physical channels between IoT devices and discover potential correlations between devices and the environment. Nevertheless, it is not effective to detect real-world runtime physical environmental interaction influences.

Hence, robust anomaly detection should consider the time at which devices interact through physical channels. Moreover, some interactions between devices and the physical environment may occur either immediately (e.g., turn on a light) or slowly and continuously (e.g., boil water via a heater), temporal context of the smart home matters when detecting anomalies. To sum up, we should implement a systematical analysis when detecting smart home anomalies, taking both environmental context and temporal context into consideration.

3. System Design

3.1. Overview. When analyzing the correlation of smart home devices, we consider the influence between automation rules and devices. We first extract the correlated device states as environmental context. We pinpoint the environmental states when target events are triggered and then build feature vectors. For device behavior regularity caused by user behavior and environmental changes, we extract the state changes of target devices from the logs, model the behavior, and refine the temporal context to forecast the next event. Finally, we construct a classifier and implement the abnormal detection systems in smart homes.

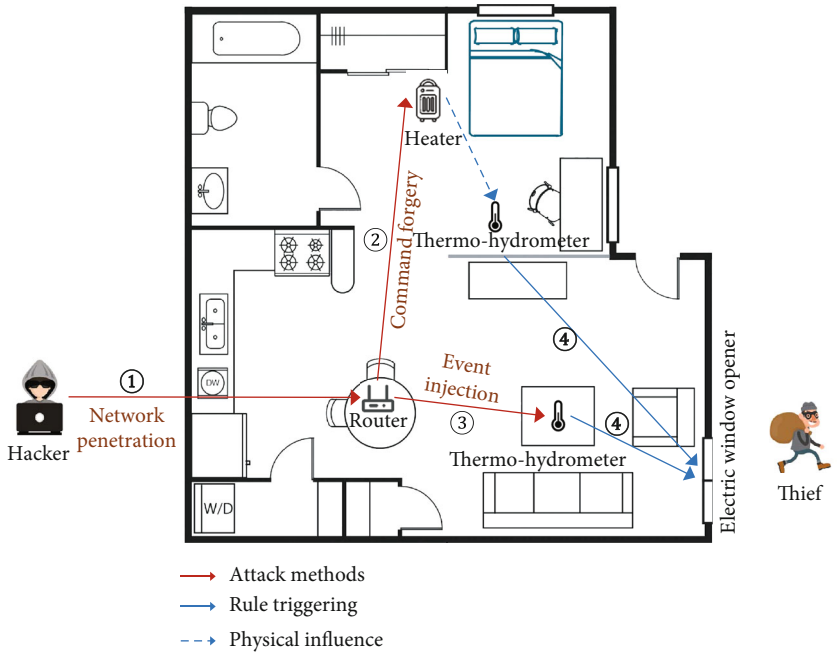


Figure 1: An example scenario where an attacker breaks into smart home to open the window.

Table 1: IoT device influences.

Influence	L	Source	Processing by
Automation rules	A	Automation rules and apps	System analysis
Influences among devices	P	Physical influences	
User behavior	A	User commands	Behavior modeling
	N	Network traffic	
	P	User activities	
Environmental changes	P	Periodic changes	

Note: L: layer; A: app layer; N: network layer; P: physical layer.

The framework of the anomaly detection system HomeGuardian is shown in Figure 2. The purpose of HomeGuardian is to screen out abnormal events from smart home platform logs. It is composed of three modules, namely, *testbed platform*, *context extraction*, and *anomaly detection*. Specifically, the testbed platform is a smart home experimental platform with multiple functions, such as device control and behavior simulation. The context extraction module extracts the environmental context among smart devices by analyzing the device configurations and rule configurations of the smart home platform. Besides, it also uses machine learning algorithms for behavior modeling to predict the following state of the target device. Environmental context and temporal context are the input of the anomaly detection module. The anomaly detection module is composed of a neural network, which filters out abnormal events.

3.2. Temporal Context. The smart home events recorded in platform logs can be represented as (timestamp, device, state), namely, the date and time when the device state changes occur, the device name, and its state value after the event occurs, respectively. Note that apart from successive values (e.g., temperature and humidity), states can also be presented in binary values (e.g., ON/OFF for a switch and OPEN/CLOSE for a door). A log sample reads as $(2021-05-1108:58:31, light.L001, on)$

To obtain the temporal contextual feature of the smart home events, the log entries corresponding to the target device, i.e., the state changes of the target device, are first filtered out from the log. The time interval of event occurrence is calculated based on the event timestamps. The change of the time interval length reflects the frequency of the target device events with the event change pattern. We determine the analysis method by collecting data from the smart home testbed platform. To keep the consistency of time increments, we denote the first presence of a certain event with the timestamp timestamp_0 as an initial event e_0 and then calculate the time interval between e_0 and the subsequent events $e_i|_{i=1,2,\ldots}$ as $t_i = \text{timestamp}_i - \text{timestamp}_0$, where timestamp_i is the timestamp for event e_i. In this way, the temporal context features are monotonically increasing in chronological order.

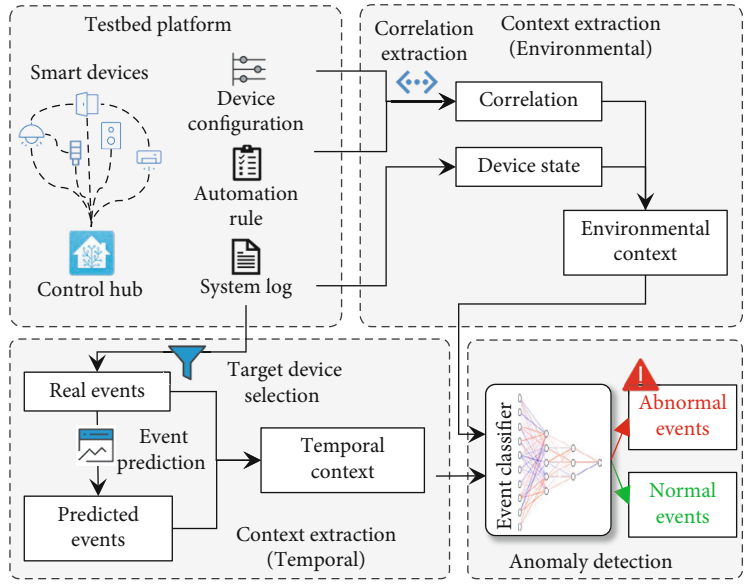


Figure 2: Framework of HomeGuardian.

After that, a learning model is constructed based on the features extracted from the log. It predicts the moment and state value of the next state change of the target device. The inputs of the model include the prior N states of the target device $\mathbf{s} = (s_1 ; s_2 ; \cdots ; s_N)$, and the prior N occurrence times of the target device $\mathbf{t} = (t_1 ; t_2 ; \cdots ; t_N)$. Among them, the parameter $\mathbf{t}$ is the record starting point at the specified time, which can be updated in a period to prevent the data from being too large. The output of the model is denoted as (t^P, s^P). t^P is the time interval of the next event, and s^P is the state of the next event. We apply a regression algorithm to extract the relationships among input variables. The outputs are presented as $t^P = \alpha_1^T \mathbf{t} + \delta_1 + \varepsilon_1$ and $s^P = \alpha_2^T \mathbf{s} + \delta_2 + \varepsilon_2$, where α_1 and α_2 are the weight coefficient vectors of each feature vector, i.e., how closely each data quantity is associated with the device state value. δ_1 and $\delta_2$are constant terms of intercepts. ε_1 and ε_2 are errors obeying a normal distribution with a mean of 0. t^P and s^P are predictions of the time of the next event and the state of the device for the same device. The temporal context $\mathbf{f}_t$ includes the time prediction t^P and state prediction s^P for subsequent events and the actual time t^R and actual state t^R of the event occurrence. It can be presented as $\mathbf{f}_t = (t^P ; s^P ; t^R ; s^R)$.

3.3. Environmental Context. The environmental context of system events refers to the device states correlated to the target device. The correlations come from user-defined automation rules, physical channels, and spatial relationships between devices. An important characteristic of smart homes is that smart devices may cause impacts on the physical environment. Such physical influence brings the correlation of state changes between devices [11]. Further, there are interactions between user-defined automation rules and IoT applications. Physical channels, such as temperature, humidity, and brightness, enable devices with certain attributes to interact with each other. For example, there is an automation rule that controls a radiator to turn on or off according to room temperature. In this case, the temperature channel connects the heater and temperature sensor with correlation.

Based on the aforementioned analysis, relevant device selection is determined according to automation rules. The spatial distribution of the devices and the physical channels shared among the devices impacts their correlation as well. The correlation degree $R_{D,C}$ between a device D and a physical channel C is obtained by Natural Language Processing (NLP), which refers to device correlations. Then, we obtain the semantic similarities $R_{D,C}$ of each device name words and physical channels in smart homes. We use a threshold Θ_R and consider the devices whose $R_{D,C} > \Theta_R$ as containing the *physical property* C.

The devices with the same physical channel and spatially-near locations are considered correlated. Meanwhile, the devices affected by the automation rule are also considered correlated. Then, we get the correlation matrix $\mathbf{G}$.

During the training phase of our model, since the data collected from smart home logs only includes the state change of devices, it is necessary to maintain a cache matrix variable $\mathbf{M}_{L\times N}$ to record the current state of all devices in the environment. Each row of the matrix represents the previous N state values of the target device, and each column represents the state of all L devices in the current time $\mathbf{s}_D = (s^{D1} ; s^{D2} ; \cdots ; s^{DL})$, i.e., the environment state. In this step, we select k other devices with the highest correlation of target devices for the following calculation. First, we obtain the data related to the target event for anomaly detection, including states of all L devices in the environment, and the corresponding correlation vector which is denoted as $\mathbf{g} = \mathbf{G}_{*,j}$. $\mathbf{g}$ is the correlation vector between the target device and other devices, and the values of each item are 0 or 1. (i.e., a column in correlation matrix $\mathbf{G}$). The devices with

correlation are marked as 1, and others are marked as 0. The Hadamard product of $\mathbf{g}$ and $\mathbf{s}_D$ is the device state value vector correlated with the target equipment. Furthermore, we remove the zero values and reduce the dimension of the state value vector. After the above steps, we obtain the current state changes correlated to the target devices, i.e., the environmental context features, which are denoted as $\mathbf{f}_e = (s^{D1}; s^{D2}; s^{D3}; \cdots; s^{Dk})$.

Algorithm 1 summarizes the above environmental context feature extraction process.

3.4. Event Classifier. Given $\mathbf{f}_t$ and $\mathbf{f}_e$, we utilize Neural Networks (NN) to classify normal events and abnormal events. According to the characteristics and experience of the target problem, if the NN has two hidden layers and an appropriate activation function, it can fit any decision boundary or smooth mapping with any accuracy [22]. Since the context feature may differ for different types of candidate events, for example, some events may pay more attention to the historical trend, some pay more attention to the surrounding environment, and some need to be comprehensively consider these two factors. In order to consider the impact of time and environmental context for event classification at the same time, we concatenate the two feature vectors as NN input, and we learn the weight relationship of each feature to the judgment result by training our NN.

The input of our NN is vector $\mathbf{x} \in R^{k+4}$ concatenated from temporal context $\mathbf{f}_t \in R^4$ and environmental context $\mathbf{f}_e \in R^k$, i.e., $\mathbf{x} = \{\mathbf{f}_t, \mathbf{f}_e\}$. The output is the judgment result $\hat{y}$. We use ReLU on the hidden layers and Sigmoid on the output layer to map the result to $[0, 1]$. When training, the normal events are marked as 0, abnormal events are marked as 1 , and we set a threshold on the result to give judgment. The input layer has n nodes, there are $n/2$ nodes in the first hidden layer, and 3 nodes in the second hidden layer. Temporal context $\mathbf{f}_t$ includes predicted time t_i^P, predicted value s_i^P, real time t_i^R, and real value s_i^R, which describes deviations between predicted and true values. The environmental context $\mathbf{f}_e$ includes the states of the selected k devices, which consist of related device states $s_i^{D1}, s_i^{D2} \cdots s_i^{Dk}$. The forward propagation process of our NN is described as $\mathbf{h}_1 = ReLU(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$, $\mathbf{h}_2 = ReLU(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$, and $\hat{y} = \text{Sigmoid}(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$, where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ refer to the weight matrices for each layer, and $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ refer to the layer bias vectors. We use binary crossentropy (BCE) [23] as our loss function, which is denoted as $L(y, \hat{y}) = -y \cdot \log \hat{y} + (1 - y) \cdot \log (1 - \hat{y})$, where $\hat{y}$ refers to the probability to predict samples as positive of our model, i.e., the probability of an event to be predicted as abnormal. y refers to the sample label, which is 1 when the sample is positive and 0 when negative. We adopt a binary classification method to determine whether there are abnormal events in the smart home system. Regular device state changes in a smart home are generally normal events. The training process requires not only the event log under a normal environment but also abnormal events. To achieve this, we obtain the abnormal data from our self-designed testbed by simulation. The features of normal and abnormal events are extracted based on the log generated by the smart home platform.

4. Implementation

4.1. Simulation-Based Data Collection. Our anomaly detection system is deployed on a heterogeneous system with different brands of devices connected to Home Assistant. Normal behavior is obtained directly from the system logs. To simulate abnormal behavior, we reproduce an injection attack for forgery sensor event. In particular, we intercept the token through a man-in-the-middle attack via a ZigBee gateway and forge POST requests from sensors to the Home Assistant server to overwrite the states of real devices. After injecting the forgery event, normal and abnormal events are indistinguishable in platform system logs. Thus, we capture the records of events replied by the Home Assistant to mark the events injected by our attack.

After processing the data collected from the virtual and real smart home platforms, we observe that the time interval between two events varies randomly. Thus, this feature is inappropriate as a criterion for effective detection. At the same time, if there are unexpected situations such as network disconnection and system downtime in the smart home system, the time interval of events will increase and drop sharply, which might seriously affect the accuracy of our behavior model. Therefore, in this work, the event time information is uniformly converted to Unix timestamp [24]. Through actual testing, we found that the frequency of event occurrence is different for different devices. Therefore, an appropriate start time could be selected according to the device types. Take motion sensors for instance, if the event frequency is high, the first state change moment of each week could be selected as the starting time point. Differently, temperature sensors have low-frequency events, and the first state change every three months could be selected as the starting point as seasonal effects need to be considered.

4.2. Environment Association Extraction. In this step, we use word2vec [25] to convert the extracted keywords into two vectors, namely, V_D (device name word vector) and V_C (physical channel semantics vector) and then calculate their cosine similarity, which is given by $R_{D,C} = \mathbf{v}_D \cdot \mathbf{v}_C / \|\mathbf{v}_D\| \cdot \|\mathbf{v}_C\| = \sum_{i=1}^{n} v_{Di} \times v_{Ci} / \sqrt{\sum_{i=1}^{n} v_{Di}^2} \times \sqrt{\sum_{i=1}^{n} v_{Ci}^2}$. We use the word vector model from Google News [26] as a pre-trained corpus. We extract keywords of device entities in Home Assistant (e.g., "light," "sensor," and "door") and use word2vec to embed them. The correlation between devices and physical channels is represented by the cosine similarity between device keywords and physical channel keywords (e.g., "motion," "illuminance," and "sound").

Based on the established rule set and semantic association information, we obtain the association table of IoT devices. As shown in Table 2, the relevance includes deterministic association rules in which the trigger condition involves the correlation between the monitoring device and the target device to carry out the instruction. These associations are determinate because as long as the trigger condition is met, the smart home system will instruct target devices to perform corresponding operations. It is important to note that associations through physical channels need to combine with the spatial location of devices.

Input: n_D, n_C, Θ_R, $\mathbf{s}_D$
Output: $\mathbf{f}_e$
/* compute similarity of word vector, build correlation matrix $\mathbf{G}$ */
for $i = 0 \longrightarrow len(n_C)$ **do**
  for $j = 0 \longrightarrow len(n_D)$ **do**
    $R_{D,C} = similarity(n_C, n_D)$
    if $R_{D,C} > \Theta_R$ **then**
      $\mathbf{G}_{i,j} \longleftarrow 1$
    else
      $\mathbf{G}_{i,j} \longleftarrow 0$
/* build feature vector */
$\mathbf{f}_D \longleftarrow \mathbf{G}_{*,j} \mathbf{es}_D$
/* remove 0 values*/
for *item* **in** $\mathbf{f}_D$ **do**
  if $item \neq 0$ **then**
    $\mathbf{f}_e.append(item)$
return $\mathbf{f}_e$

Algorithm 1: Environmental Context Feature Extraction.

Table 2: Correlation between smart home devices.

Causality	Code	a	b	c	d	e	f	g	h	i
Temperature sensor	a	—	—	√	√	—	—	—	—	—
Humidity sensor	b	—	—	√	—	√	—	—	—	—
Air conditioner	c	○	○	—	√	√	—	—	—	—
Smart socket	d	—	—	√	—	—	—	—	—	—
Heater	e	○	○	√	—	—	—	—	—	—
Humidifier	f	○	○	√	—	—	—	—	—	—
Wireless switch	g	—	—	—	—	—	√	—	—	—
Bedside lamp	h	—	—	—	—	—	—	—	○	—
Night light	i	—	—	—	—	—	—	—	○	—
Motion sensor	j	—	—	—	—	—	—	√	—	—
Door and window sensor	k	—	—	—	—	—	—	—	—	√

√: association rules triggered; ○: environment triggered.

During data processing, we record the latest N states of each device in smart homes (according to device type, N is selected from 2, 10 in our implementation). For each state, we also require the current states of other devices (i.e., the environment state). In the detection phase, HomeGuardian then implements real-time anomaly detection by capturing device states in the current environment from the Home Assistant platform through GET requests.

5. Evaluation

5.1. Experiment Setup. To collect testing event data, we develop a self-configured testbed based on the Home Assistant platform. Our testbed supports virtual device simulation, which is based on HH114 dataset from CASAS [27]. The testbed consists of real smart devices and simulated virtual devices. Event data related to experiments are logged and can be extracted on-demand.

We mainly focus on four types of devices below: motion sensors, temperature sensors, lights, and illumination sensors. The virtual devices can also be bound to devices in the laboratory, which can reflect the physical interactions (e.g., interactions between a virtual light L001 and a real illumination sensor LS001.) Figure 3 shows the layout of virtual devices in a room map in our experiments. Environmental factors influence devices located in the same color area. The devices marked with an asterisk are real devices, and devices with two asterisks are simulated virtual devices.

By sending events to the testbed platform iteratively, we simulate logs containing interactions produced by preset custom automation rules. Since events in the CASAS dataset are captured under a real scenario that can reflect user behaviors, simulated logs are largely consistent with the real logs. The logs can be directly obtained from the Home Assistant platform if connected devices exist. The virtual devices are used as supplementary. Users manually manipulate or use external scripts to control the real devices and record real events.

5.2. Effectiveness on Event Prediction. In this subsection, we first determine the count of historical events N selected

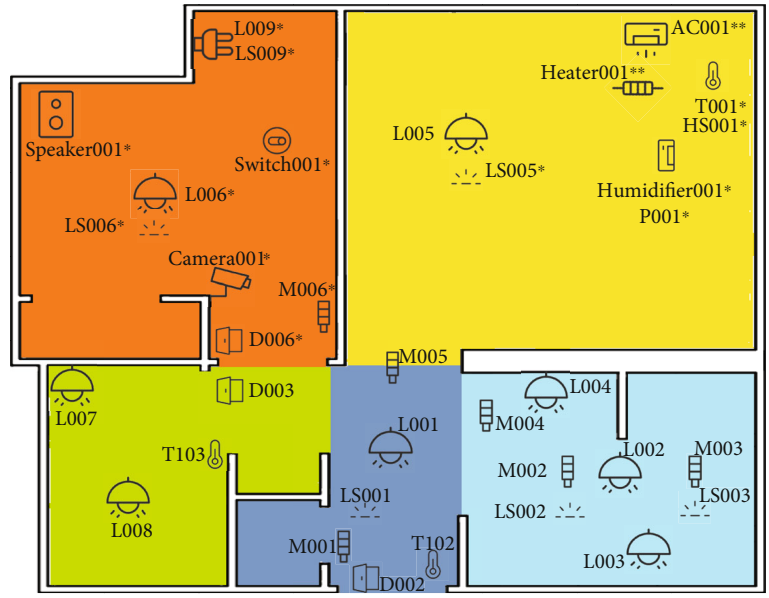


Figure 3: Device deployment layouts of testbed.

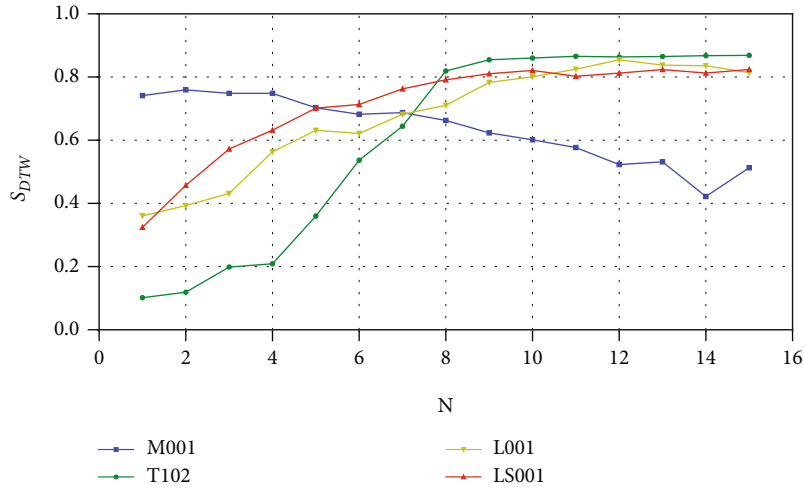


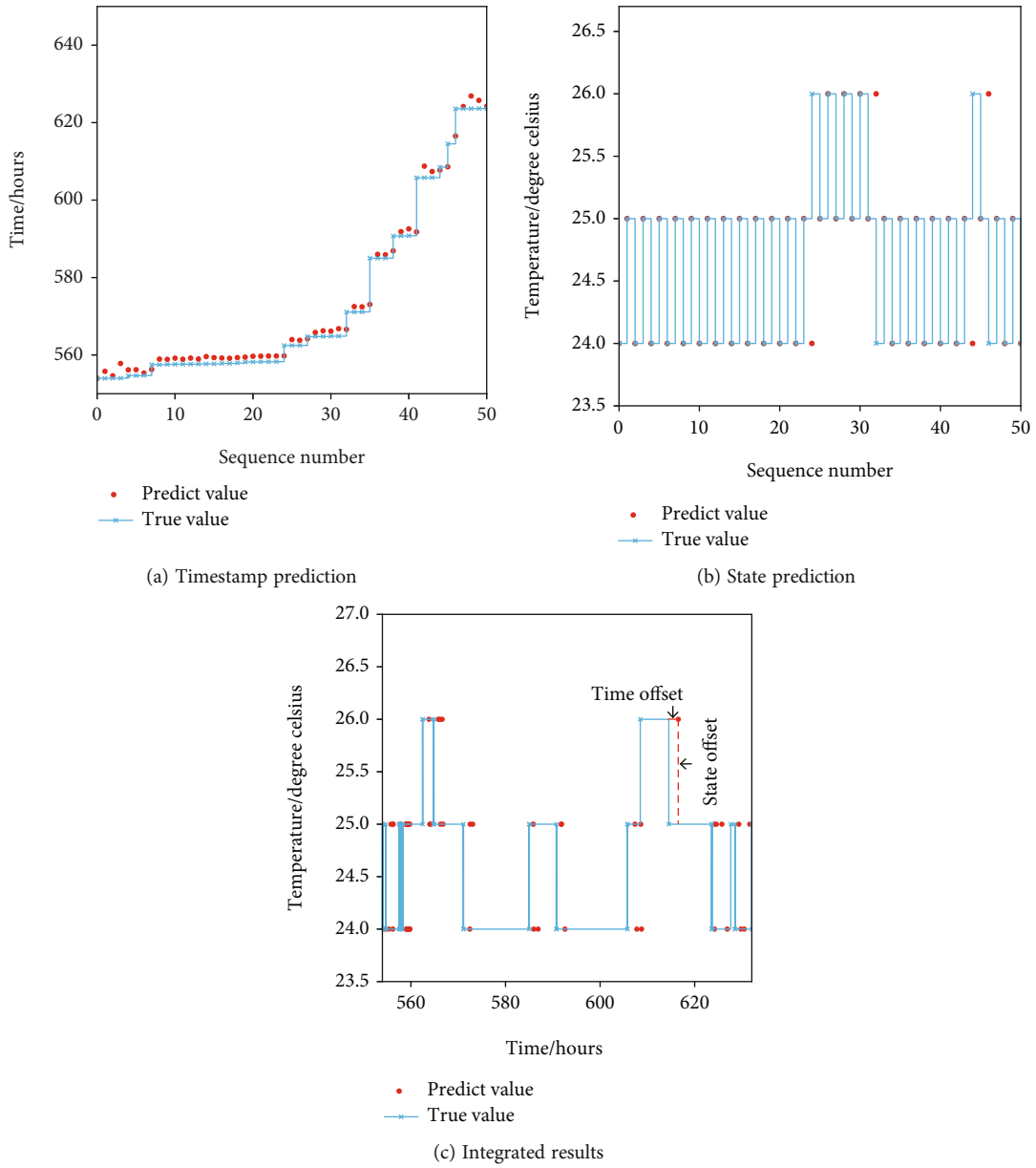
Figure 4: The influence of historical data length N on event prediction results.

when predicting the target one and evaluate the effectiveness of the event prediction mechanism. We select four types of typical devices for the following test.

5.2.1. Parameter Selection. We use dynamic time warping (DTW for short) as an algorithm measuring the distance between a couple of time series and calculate similarity distances L_{DTW} between a predicted sequence A' and a real sequence A. Thus, the similarity of these sequences is calculated as $S_{\mathrm{DTW}} = 1 - 2\mathrm{L}_{\mathrm{DTW}}/\mu_A + \mu_{A'}$. Where μ represents the mean state value of each sequence. The similarity is then used as an evaluation benchmark for prediction effectiveness. As shown in Figure 4, we can observe the impact of N selection on event prediction results by calculating S_{DTW} of event predictions for four different devices with incremental N values selected.

For motion sensors, the prediction effectiveness decreases as the history length N increases as shown by the blue discount, which means these device states are not strongly correlated with historical data. Therefore, we pay more attention to the environmental context when anomaly detection, such as changes under correlation between each motion sensor and its surrounding motion sensors in real scenarios. Consequently, we take $N = 2$ for devices of this kind. For temperature sensors, illumination sensors, and lights, the prediction effect will not achieve the best goal until N is about 9 to 10, and it only fluctuates slightly with N larger than 10. Accordingly, we take $N = 10$ for data processing.

5.2.2. Results. We divide the first 80% of the dataset in the specified time interval as the training set and the rest of the dataset as the testing set. Rest experiments mentioned in this paper all take the same method. Taking the temperature sensor T102 as an example, the prediction results of its state changes are shown in Figures 5(a) and 5(b). The cyan



(a) Timestamp prediction

(b) State prediction

(c) Integrated results

Figure 5: Prediction results of temperature sensor T102.

Table 3: Event prediction NRMSE.

Device	M001	M002	T102	T103	L001	L002	LS001	LS002
$NRMSE_t$	0.011	0.014	0.023	0.025	0.028	0.021	0.021	0.023
$NRMSE_s$	0	0	0.1394	0.1648	0	0	0.051	0.060

Note: $NRMSE_t$: NRMSE of timestamps; $NRMSE_s$: states prediction.

data represents the true value, the red data represents the predicted value, and the fitting effect is as expected.

Exceptionally, the output values of the model need to be adjusted to the specifics of different IoT devices. Since the accuracy of the device T102 thermometer is 1 degree Celsius in the current dataset, the output of the model can be rounded. Binary value devices, such as human motion sensors and light switches, output in the form of ON/OFF. The subsequent output value must be the inverse of the current state. Thus, the binary value device only needs to predict the moment of the next occurrence, without considering device states.

While using event timestamps as the x-coordinate and the state value of the device as the y-coordinate, the predicted events and the real events are put together for comparison. The moment offset and state offset of the

Table 4: Abnormality classifier effectiveness for each device.

Device	Temporal context only			Environmental context only				Single-output NN				Dual-output NN
	Precision	Recall	F1-score	Precision	Precision	Recall	F1-score	Precision	Precision	Recall	F1-score	Precision
M001	0.84	0.92	0.88	0.69	0.84	0.92	0.88	0.69	0.84	0.92	0.88	0.69
M002	0.85	0.88	0.86	0.48	0.85	0.88	0.86	0.48	0.85	0.88	0.86	0.48
T102	0.93	0.89	0.91	0.41	0.93	0.89	0.91	0.41	0.93	0.89	0.91	0.41
L002	0.52	0.63	0.57	0.99	0.52	0.63	0.57	0.99	0.52	0.63	0.57	0.99
LS001	0.57	0.39	0.46	0.97	0.57	0.39	0.46	0.97	0.57	0.39	0.46	0.97
LS002	0.41	0.28	0.33	0.98	0.41	0.28	0.33	0.98	0.41	0.28	0.33	0.98

prediction results can be obtained intuitively, as shown in Figure 5(c). Then, real values and predicted values of time and state will be passed into the classifier of the anomaly detection, respectively. For quantitative evaluation of forecast results, we use the normalized root mean square error (NRMSE) [28] as an indicator. For a given sequence $\mathbf{Y}$ and its estimate $\widehat{\mathbf{Y}}$, there are $\mathrm{MSE} = 1/n\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2$, and $\mathrm{NRMSE} = \mathrm{RMSE}/Y_{\max} - Y_{\min} = \sqrt{\mathrm{MSE}}/Y_{\max} - Y_{\min}$.
NRMSE relates the root mean square error (RMSE) to the observed variable range. Thus, it can be interpreted as a fraction of the overall range that is typically resolved by the model. The NRMSE of prediction results is shown in Table 3. The NRMSE for binary states prediction and timestamps prediction is 0 and 0.02, respectively. The effectiveness of the prediction achieves as expected.

5.3. Effectiveness on Anomaly Detection. First, we take an experiment on the motion sensor *M001*. We extract a total of 24,765 log entries generated by *M001* within a week. Further, we simulate 2,500 injection attacks conducted on our testbed platform. After that, temporal context features and environmental context features at the corresponding moment are extracted for anomaly detection. We select the first 80% of the data as the training set and the last 20% as the test set. We use Youden *J* statistic [29] to obtain the optimal receiver operating characteristic (ROC) threshold $\Theta = \mathrm{argmax}(\mathrm{TPR} - \mathrm{FPR}) = \mathrm{argmax}((\mathrm{TP}/\mathrm{TP} + \mathrm{FN}) + (\mathrm{TN}/\mathrm{TN} + \mathrm{FP}) - 1)$. The F1-score of abnormal events generated by the classifier is calculated as 0.90636, with an accuracy of 0.96.

To achieve better classification while avoiding overcomplicated schemes, we fine-tune the NN structure. Specifically, there are two nodes in the output layer. One acts as the normal label $\widehat{y}_1$, the other acts as the abnormal label $\widehat{y}_2$, and both fall in the range 0 to 1 through the softmax function and add up to 1, i.e., $\widehat{\mathbf{y}} = (\widehat{y}_1; \widehat{y}_2) = \mathrm{Sigmoid}(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$. During training, normal events are labeled as (1,0), and abnormal events are labeled as (0,1). During classification, once the output shows $\widehat{y}_1 > \widehat{y}_2$, the input event is classified as normal, and $\widehat{y}_1 < \widehat{y}_2$ is the criteria to determine abnormal events. We feed the dataset into a new dual-output classification network, and its results compared to the original one are shown in Table 4. F1-score is selected as the effectiveness evaluation criterion. The same methods are applied to motion sensor M002, temperature sensors T102/T103, smart lights L001/L002, and light sensors LS001/LS002. Anomaly detection of different devices has different dependencies on temporal or environmental contexts. Combining the two features can adapt to different types of devices without obvious F1-score degradation. Besides, the classifier achieves a better classification of smart devices of different value types. Since a dual-output neural network structure avoids threshold selection, it has better classification ability than a single-output neural network classifier.

6. Related Work

6.1. Event Sequence-Based Detection. Current studies mainly use network traffic and environmental sensor states at the time of the event as features. Considering network traffic features, Saxena et al. [30] propose a method to detect the identity and behavior of home devices using encrypted network traffic, choosing statistical features of ZigBee network traffic packets as a basis for classification. Zhang et al. [12] present an approach based on physical event fingerprints. They construct automata for IoT application behavior and extract event features from the wireless communication environment as fingerprints.

Since smart home sensors may change correlatively when they are affected by the same physical event, Laput et al. [31] propose a method to obtain event fingerprints based on heterogeneous sensor data. They collect data from all sensors except cameras, manually label event data, and use an SVM model to classify abnormal events. Birnbach et al. [32] also use heterogeneous sensor data to build fingerprints for events and detect spoofed events. They extract the relative mutual information of each sensor and event as fingerprints and select data for SVM classification.

6.2. Application Analysis-based Detection. In addition to extracting the characteristics of events, the correlations of IoT applications are also considered in anomaly detection.

To find risky physical channel associations, Ding et al. [11] provide IoTMon, a solution for identifying and analyzing hidden interaction chains between IoT applications. It analyzes SmartApp interaction using static analysis and SmartApp descriptions via NLP to identify smart home environments. Soteria [33] models IoT applications based on intermediate representation (IR). The state model is automatically extracted from the SmartThings IoT applications to detect whether the program has rule conflicts.

However, approaches based on static analysis cannot solve the runtime policy violation problem in realistic smart home systems. IoTGuard [34] is a dynamic policy

enforcement system that blocks insecure states by monitoring the runtime behavior of IoT applications. HAWatcher [14] is based on semantic analysis of event logs and physical-channel-related descriptions. It generates associations and uses event logs for verification. Based on the analysis of the impact of physical channels between devices, Ozmen et al. [35] use formulas of physical laws for the first time to quantify the specific results of interactions between devices, improving the accuracy of the analysis results.

7. Conclusion

In this paper, we propose HomeGuardian, a context-based approach to detect abnormal events in smart home systems. In our approach, we predict the temporal context and infer environmental context based on system log, device, and rule configurations. By converging these hybrid event contexts, we construct a learning-based classifier to detect abnormal events. We evaluate HomeGuardian based on the event data collected from our self-configured testbed. The experiment results show that the F1-scores are beyond 0.90 for all device types.

Data Availability

The public data used to support this study are available at DOI:10.1109/JBHI.2015.2461659. The prior study (and dataset) is cited at relevant places within the text.

Conflicts of Interest

The authors declare that they have no interest conflicts.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No. 62172027), the Beijing Natural Science Foundation (No. 4202036), the National Key R&D Program of China (No. 2020YFB1005601), Hangzhou Innovation Institute, Beihang University, under Grant 2020-Y5-A-022, and the National Natural Science Foundation of China (No. U1733115, No. 61871023).

References

[1] S. Suresh and P. V. Sruthi, "A review on smart home technology," in *2015 online international conference on green engineering and technologies (IC-GET)*, pp. 1–3, IEEE, 2015.

[2] Z. Tong, F. Ye, M. Yan, H. Liu, and S. Basodi, "A survey on algorithms for intelligent computing and smart city applications," *Big Data Mining and Analytics*, vol. 4, no. 3, pp. 155–172, 2021.

[3] Z. Cai, Z. Xu, J. Wang, and Z. He, "Private data trading towards range counting queries in internet of things," *IEEE Transactions on Mobile Computing*, vol. 1, 2022.

[4] B. Zhang, Y. Song, and Z. Tang, "Data mining-based smart home control platform," *Office Automation*, vol. 423, no. 10, pp. 24–28, 2020.

[5] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2020.

[6] Y. Huo, J. Fan, Y. Wen, and R. Li, "A cross-layer cooperative jamming scheme for social internet of things," *Tsinghua Science and Technology*, vol. 26, no. 4, pp. 523–535, 2021.

[7] A. Acar, H. Fereidooni, T. Abera et al., "Peek-a-boo: I see your smart home activities, even encrypted!," *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 207–218, 2020.

[8] Z. Cai and Z. He, "Trading private range counting over big iot data," *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 144–153, 2019.

[9] NIST, "Cve-2018-3911 detail," https://nvd.nist.gov/vuln/detail/CVE-2018-3911, 2018.

[10] Y. Tian, N. Zhang, Y. H. Lin et al., "Smartauth:user-centered authorization for the internet of things," *26th USENIX Security Symposium (USENIX Security 17)*, pp. 361–378, 2017.

[11] W. Ding and H. Hongxin, "On the safety of iot device physical interaction control," *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 832–846, 2018.

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

[13] Y. Yonghua, C. Li, M. A. Jonas et al., "Detecting abnormal behaviors in smart home," in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, pp. 37–42, IEEE, 2019.

[14] F. Chenglong, Q. Zeng, and X. Du, "Hawatcher:semantics-aware anomaly detection for appified smart homes," *30th USENIX Security Symposium (USENIX Security 21)*, pp. 4223–4240, 2021.

[15] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks: a survey towards private and secure applications," *ACM Computing Surveys*, vol. 1, 2021.

[16] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "Sok: security evaluation of home-based iot deployments," in *2019 IEEE symposium on security and privacy (sp)*, pp. 1362–1380, IEEE, 2019.

[17] U. Satapathy, B. K. Mohanta, D. Jena, and S. Sobhanayak, "An ecc based lightweight authentication protocol for mobile phone in smart home," in *2018 IEEE 13th international conference on industrial and information systems (ICIIS)*, pp. 303–308, IEEE, 2018.

[18] G. Ho, D. Leung, P. Mishra et al., "Smart locks: lessons for securing commodity internet of things devices," *Proceedings of the 11th ACM on Asia conference on computer and communications security*, pp. 461–472, 2016.

[19] T. Melamed, "An active man-in-the-middle attack on bluetooth smart devices," *Safety and Security Studies*, vol. 15, p. 2018, 2018.

[20] A. Ranieri, D. Caputo, L. Verderame, A. Merlo, and L. Caviglione, Eds., "Deep adversarial learning on google home devices," https://arxiv.org/abs/2102.13023, 2021.

[21] Y. J. Jia, Q. A. Chen, S. Wang et al., "Contexlot: Towards providing contextual integrity to appified iot platforms," in *NDSS, volume 2*, p. 2, San Diego, 2017.

[22] X. Hao, G. Zhang, and S. Ma, "Deep learning," *International Journal of Semantic Computing*, vol. 10, no. 3, pp. 417–439, 2016.

[23] PyTorch, "Bceloss," https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html, 2019.

[24] Wikipedia, "Unix time," https://en.wikipedia.org/wiki/Unix_time, 2022.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing*, vol. 26, 2013.

[26] Google, "Googlenews-vectorsnegative300," https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing, 2013.

[27] D. J. Cook, M. Schmitter-Edgecombe, and P. Dawadi, "Analyzing activity behavior and movement in a naturalistic environment using smart home techniques," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 6, pp. 1882–1892, 2015.

[28] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, and V. A. E. Kamaev, "A survey of forecast error measures," *World Applied Sciences Journal*, vol. 24, no. 24, pp. 171–176, 2013.

[29] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, no. 1, pp. 32–35, 1950.

[30] U. Saxena, J. S. Sodhi, and Y. Singh, "Analysis of security attacks in a smart home networks," in *2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence*, pp. 431–436, IEEE, 2017.

[31] G. Laput, Y. Zhang, and C. Harrison, "Synthetic sensors: towards general-purpose sensing," *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 3986–3999, 2017.

[32] S. Birnbach and S. Eberz, *Peeves: Physical Event Verification in Smart Homes*, 2019.

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

[34] Z. Berkay Celik, G. Tan, and P. D. Mc-Daniel, "Iotguard: dynamic enforcement of security and safety policy in commodity iot," *NDSS*, 2019.

[35] M. O. Ozmen, X. Li, A. C. A. Chu, Z. B. Celik, B. Hoxha, and X. Zhang, "Discovering physical interaction vulnerabilities in iot deployments," https://arxiv.org/abs/2102.01812, 2021.