IMEO: Anomaly Detection for IoT Devices using Semantic-based Correlations

Seungmin Oh[1], Jihye Hong[2], Daeho Kim*[3], Eun-Kyu Lee*[4], Junghee Jo[5]
Department of Information and Telecommunication Engineering, Incheon National University, Incheon, Korea[1,2,3,4]
Department of Computer Education, Busan National University of Education, Busan, Korea[5]

***Abstract*—In the Internet of Things (IoT) security, anomalies due to attacks or device malfunctions can have serious consequences in our daily lives. Previous solutions have been struggling with high rates of false alarms and missing many actual anomalies. They also take a long time to detect anomalies even if they successfully detect anomalies. To overcome the limitations, this paper proposes a novel anomaly detection system, named IoT Malfunction Extraction Observer (IMEO), that utilizes semantics and correlation information for smart homes. Given IoT devices installed at home, IMEO creates virtual correlations based on semantic information such as applications, device types, relationships, and installation locations. The generated correlations are validated and improved using event logs extracted from smart home applications. The finally extracted correlations are then used to simulate the normal behaviors of the smart home. Any discrepancy between the actual state of a device and the simulated state is reported as abnormal while comparing correlations and event logs. IMEO also utilizes the observation that malfunctions of IoT devices occur repeatedly. An anomaly database is created and used so that repetitive malfunctions are quickly detected, which eventually reduces processing time. This paper builds a smart home testbed on a real-world residential house and deploys IoT devices. Six different types of anomalies are analyzed, synthesized, and injected to the testbed, with which IMEO's detection performance is evaluated and compared with the state-of-the-art correlation-only detection method. Experimental results demonstrate that the proposed method achieves higher performance of detection accuracy with faster processing time.**

Keywords*—*Security; anomaly detection; semantics; Internet of Things; attack

I. Introduction

The rapid growth of Internet of Things (IoT) has enjoyed a wide variety of applications. According to a market report, the global number of connected IoT devices is expected to grow to up to 16.7 billion endpoints by the end of 2023 [1]. It also reports that this number will grow by 16% annually to reach 29.7 billion by 2027. IoT devices are increasingly being integrated through IoT platforms such as SmartThings [2] and Homekit [3]. This allows users to connect to IoT devices from different vendors using smart applications, thus providing great convenience for IoT heterogeneity.

With the development of IoT applications (this paper is specifically focused on smart homes), there are also growing security and safety concerns [4]. Various causes, including attacks, device errors, malfunctions, and misconfigurations, can cause anomalies in IoT and eventually lead to unexpected (and often unfavorable) outcomes. IoT anomalies have intrinsic properties, and followings show some examples of anomalies and consequences in smart homes. First, IoT devices can extend cyberspace attacks to the physical world. For instance, a "close the water valve" command can be blocked by an attacker, resulting in flooding of the room. Second, it is very often that the malfunction of the device is rarely noticed until a specific result occurs. An electronic heater that has received a "too cold" command from a smart home application can cause a fire due to the relay switch not being able to turn off the heater. Third, when IoT devices are connected to each other via automation, abnormal behaviors of one device can cause unwanted behaviors of other devices, which exacerbates the effects of the anomaly. For instance, a smart door lock that is automatically unlocked only when there are residents, is released due to false events by a human presence sensor.

To address these concerns, many previous research on anomaly detection utilize data mining techniques that profile normal behaviors of a system and report off-profile events as anomalies. These works generally accept event logs as inputs without fully considering semantics of each event that can actually be obtained from smart apps, device types, and device functions. There are three limitations to be considered in this approach. First, the logic in some smart apps is too complex to be accurately extracted, which may cause incorrect normal operations and malfunctions. Say, a smart app logic generates an event pattern, "If two motion sensors in a living room both do not detect movement, turn off the smart plug after 30 minutes". It is difficult to mind this when considering the 30-minute delay and the "AND" logic between two sensors. Therefore, it may not be possilbe to detect the anomaly, "a smart plug fails to turn off". Second, it is often difficult to interpret the learning results, making it difficult to explain them and thus confusing to users. Third, when configurations are changed, learning results are not updated quickly. A long retraining process is required to adapt to the changes, and a lot of false alarms occur before retraining is finished.

An intuitive approach to improve the accuracy of anomaly detection is to incorporate semantic information such as device types, installation locations, relationships, and automation logic. However, there are many technical challenges to be resolved to realize the approach. 1) Typical mining techniques accept event logs as inputs; however, representing a variety of semantic information in the form of event logs has not been studied. 2) Patterns extracted from event logs may conflict with those of system behavior derived from smart home applications; identifying and resolving these conflicts are not easy. 3) It is unknown how to update a system profiling effectively when a smart home application changes.

To overcome the challenges, this paper proposes IoT Mal-

*Corresponding authors

function Extraction Observer (IMEO), a new anomaly detection system for smart home applications. Technically, it constructs correlations by using semantic information, explaining how a device's states and events correlate with those in another device, and verifies them by using event logs as evidence. Given the correlations, IMEO simulates normal behaviors and compares the simulated states to those in a real world via contextual and consequential checkings. It then reports anomalies if finding inconsistencies in comparison. Because the correlations become explainable along with the semantics in IMEO, they can help resolve conflicts with smart home applications. Thanks to the explainability, the correlations can be updated with the changes of the smart home application. This paper implements a prototype of IMEO and builds a real-world smart home testbed consisting of three rooms. Then, experiments are conducted, and the results show that IMEO can reach a high precision of 99.33% and a recall of 95.35%, demostrating better performance than a prior method.

The rest of the paper is organized as follows. Section II summarizes possible anomalies that can occur in IoT environment. Section III reviews research works on anomaly detection for IoT. Section IV introduces a reference architecture for an IoT system and defines a threat model in the architecture. In Section V, three correlation channels and the representation of correlations are described. We present the proposed system in Section VI. Our testbed is implemented and experimented in Section VII, which is followed by evaluation in Section VIII. Finally, Section IX concludes the paper.

II. Anomalies in IoT

Previous works have reported that IoT devices are often unreliable and vulnerable to malicious attacks [5], [6], [7]. This section discusses anomalies in IoT caused by devices' malfunctions and attacks.

A. Malfunctions in IoT Devices

In general, IoT devices communicate with an IoT platform; they report any *event* records and receive *command* messages to/from the platform. This subsection categorizes malfunctions in terms of them.

1) Events: There are two types of causes related to event records. (i) *Faulty event* refers to devices' reporting incorrect values. This is mainly attributed to a device defect or physical inteference. For instance, a door knocking sensor goes active and then inactive without reason [8], and a motion detector sees motions in an empty room and turns on lights [9]. (ii) *Event loss or delay* represents that event records are not reported to the platform (or any server) in a timely manner. For instance, status updates from presence sensors have been reported to suffer from long delays [10]. This may cause significant delays in executing related automation when after a resident leaves home. If the update is lost, an automation rule may fail to lock a door while away.

2) Commands: There are cases inducing malfuntions on devices. (iii) *Bogus command* refers to a phenomenon called "poltergeist" frequently reported on a user forum [11]. For instance, users reported that lights or sensors turned on in the middle of the night. It is said that this phenomenon occurred in an office where no one was present or in the hallway of a house where everyone was sleeping. They also reported that a heater connected to an air conditioner suddenly reacted and turned on. (iv) *Command failure* represents that the IoT platform issues a command that is not executed on an IoT device. A physical problem or cyber problem may cause this. The physical problem is mainly attributed to a malfunction of an IoT device. For instance, if an electrical relay inside a smart plug is broken, it may prevent the plug from cutting off the power supply. The platform recognizes that the plug is turned off, though. The cyber problem includes unstable network connections and system crashes that prevent commands from being executed.

B. Attacks on IoT Devices

This subsection identifies potential vulnerabilities on IoT devices and discovers five different types of IoT attacks.

(i) *Fake event* is an event maliciously generated by an attacker. This can trigger an IoT device to behave unexpectedly, which can lead to unfavorable consequences [12]. For example, when the attacker injects an event indicating the presence of a fake person, a door lock is unlocked. (ii) *Event interception* by an attack can intercept and discard event records. For instance, the attack blocks the wireless connections of window and door sensors so that they stop sending event records, which can lead to a home security system fails to alarm [13]. (iii) *Fake command* is injected by by an attacker into an IoT device [14]. Say, smart speakers and smart switches may accept fake commands from a local network without authenticating sources [15], [16]. (iv) *Command interception*. An attacker can also intercept commands and prevent them from being delivered to IoT devices [17]. (v) *Compromised device*. An attack can gain access to an IoT device and perform the following attacks. In a "stealthy command", the attacker takes control of a device to execute commands and prevents corresponding feedback events from being sent in order to remain covert [18]. This is similar to the fake command, except that no feedback events are sent. "Denial of Execution" means that when a valid command is sent to the device, it does not execute the command and sends back a feedback event reporting that the command was executed.

C. Repeated Malfunctions of IoT Device

A user forum notes that users are frequently having trouble with malfunctions of specific IoT devices; devices that malfunctioned once malfunction repeatedly after that or cause other problems. For instance, a sensor once disconnected from a network once experienced multiple network disconnections over several days. As such, repeated malfunctions of the device can cause difficulties in operating the automation involved or problems that incorrectly trigger other actuators.

III. Related Works

With the advancement of IoT devices and the advent of home automation applications, security and privacy issues are attracting great attention. However, most research has focused on detecting threats, attacks, and malicious codes rather than IoT malfunctions. For example, HomeGuard [19] presents the first systematic classification of threats caused by interference between different automated applications, such as automation collisions, serial execution, and loop triggering. The authors propose a method to detect these threats using SMT (Satisfiability Modulo Theories) Solver, where it performs symbol execution to extract automation rules from an application. PFirewall [20] is a study that recognizes the continuous inflow of excessive IoT device data into an IoT automation platform. It protects users' personal information from the platform by minimizing data without changing the IoT device or platform. HoMonit [21] focuses on detecting smart home applications that are malfunctioning, unlike this work on detecting anomalies of IoT devices. Given a physical event, Orpheus [22] automates system call tracking and then checks for attacks via comparison. However, it is not possible to detect anomalies such as fake events and event intercepts.

Many previous studies allow anomaly detection systems to learn the normal behavior of smart homes from historical data. For example, SMART [23] trains activity classifiers for multiple users based on different subsets of sensor readings, and further trains another classifier that takes a vector of activity classification results as input to detect sensor failures. DICE [24] checks contexts in smart homes to detect anomalies during state transition.

Traditional mining-based solutions are not clear how they can accurately learn complex behaviors in smart homes. The main difference between these conventional anomaly detection systems and this work is that IMEO extracts various semantic information such as device types, device relationships, and smart home applications and injects the information into mining processes. IMEO is not only more accurate in detection, but each detected anomaly can be interpreted as violating the correlation, so it can be explained by itself.

IV. Threat Model

A. IoT System Architecture for Smart Homes

IoT devices in smart homes are increasingly integrated through IoT platforms for seamless automation. IoT integration platforms such as SmartThings, OpenHAB, and Amazon Alexa support automation programs. Although these platforms can handle numerous IoT devices, they are summarized as a small number of abstract devices. For instance, smart lights, regardless of their brands, shapes, sizes, or wireless technologies, are equally abstracted as light. Each abstract device has events and commands associated with it.

This paper considers SmartThings, one of the leading IoT integration platforms, for a system architecture for smart homes as it supports rich automation logic. A typical SmartThings deployment, as shown in Fig. 1, has a cloud-centric architecture consisting of four layers. At the top is SmartThings Cloud. It is a cloud where smart apps run and interact with abstracted functions. The cloud uses various communication techniques such as Wi-Fi, Zigbee, and ZWave to communicate with IoT devices through the network connection layer. IoT devices can be divided into cyber and physical parts. The cyber part manages the interface for humans and connects communication between the cloud and the physical part, while the physical part performs its functions in the physical world. For instance, the Philips' Hue smart bulb consists of a physical part of an LED bulb and a cyber part of a microcontroller with a built-in wireless component.

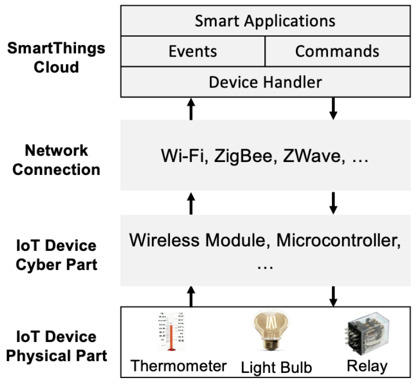

Fig. 1. IoT System architecture for smart homes; SmartThings.

Terms used in the SmartThings are briefly described in the following. An IoT device has one or more instrumental capabilities that are classified as actuators or sensors. Each capability defines one or more attributes. For instance, a smart plug device has a "switch" attribute and optionally a "power" attribute. The state (value) of each attribute is stored in the cloud and updated by events transmitted from the IoT device. For instance, a multi-purpose sensor has a capability of "contact sensor", and the cloud changes the state of its attribute "contact" from "open" to "closed" when it receives an event "contact closed" from the sensor. In the case of an actuator, SmartThings ensures that the state of the its attribute is udpated by a feedback event transmitted by the IoT device after a command is executed by the actuator.

B. Threat Model

This paper focuses on detecting malfunctions and attacks in IoT devices described in the previous section and on reporting repeated malfunctions immediately. It is also noted that IMEO is able to detect attacks that violate correlations. Attackers who know the correlations can avoid our detection by constructing an attack that does not violate the correlations.

Our threat model assumes that the IoT platform is not compromised. As with other anomaly detection tasks, we assume that there is no or few anomalies during training. No malicious or conflicting rules in installed smart home applications are also assumed. It is predicted that the average household could have more than 500 IoT devices in the near future [25]. Therefore, considering dense deployments, we propose to leverage scenarios where IoT devices have one or more other devices nearby to interact with and leverage this to detect abnormal physical behavior of the devices.

V. Correlations

IoT devices instrumented in the same home may be interrelated in the form of simultaneous or temporally related events [26], [27], [28], [29]. These correlations can occur due to the execution of application, physical interactions, or user activities. This section investigates causes of these correlations and classifies them into three channels as shown in Fig. 2. It also defines their representations.

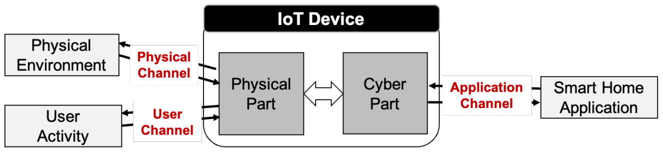

Fig. 2. Three correlation channels for IoT devices.

A. Correlation Channels

1) Application channel: Smart home applications directly cause correlations between triggers and actions as programmed. But, it also induces some additional correlations implicitly. For instance, in the SmartThings application, each automation rule implies a correlation worth verifying.

2) Physical channel: Two devices can be interrelated via certain physical properties. First, when an actuator takes an action it changes a physical property, which can be detected by a nearby sensor. For instance, an illuminance sensor can be affected by a nearby smart light turning on or off. Second, different sensors can be affected by the same physical event, creating temporally correlated IoT events. For instance, a door opening inevitably involves movement of the door, which captures an acceleration sensor and a contact sensor installed on the door, causing corresponding events to occur sequentially. As IoT device types increase, physical channel correlations can be widely observed across a variety of physical characteristics such as lighting, power, sound, and temperature.

3) User activity channel: A user activity gives rise to changes on a device, and the states of devices also reflects user activities. Thus, the user activity channel can lead to correlations between devices. For instance, when a TV is turned on it means that a user is nearby, and the motion sensor must detect it. When a user returns home, there must be continuity in events such as "person presence" showing the user's movements and "contact sensor open" showing a front door opening.

B. Representation of Correlations

This paper represents an *event* and a *state*. Let denote $E_a^{\alpha(A)}$ an event reporting that the attribute α of device A should change to value a and $S_b^{\beta(B)}$ a state indicating that the attribute β of device B has value b. Based on them, it defines two types of correlations as below.

1) Event-to-Event (E2E) correlation: represents that an event should be followed by another. For instance, E2E correlation $\langle E_{active}^{motion(A)} \rightarrow E_{on}^{switch(B)}\rangle$, given motion sensors A and light B, indicates that an active event $E_{active}^{motion(A)}$ should be followed by another event $E_{on}^{switch(B)}$.

2) Event-to-State (E2S) correlation: represents that an event occurring means that a state is true. For instance, E2S correlation $\langle E_{high}^{power(Plug)} \rightarrow S_{on}^{switch(heater)}\rangle$ indicates that the state $S_{on}^{switch(heater)}$ should be true, when an event $E_{high}^{power(Plug)}$ occurs.

3) State-to-Event (S2E) correlation: represents that an event occurs only when a state satisfies a true condition. For instance, S2E correlation $\langle S_{>60}^{illuminance} \rightarrow E_{off}^{smartplug}\rangle$ indicates that the event $E_{off}^{smartplug}$ occurs only when the state $S_{>60}^{illuminance}$ becomes true.

4) Coditional (AND) correlation: represents that events and states are combined with conditions using the $\wedge$ symbol. For instance, a correlatin $\langle E_{active}^{Motion} \wedge S_{present}^{Presence} \rightarrow E_{on}^{Switch(Light)}\rangle$ represents that an event E_{active}^{Motion}, only when a state $S_{present}^{Presence}$ is true, should be followed by another event $E_{on}^{Switch(Light)}$.

VI. Proposed System: IMEO

This section proposes a novel anomaly detection system, named IoT Malfunction Extraction Observer. Fig. 3 demonstrates an overall architecture of the proposed system, and the followings describe operation steps. IMEO first generates correlations regarding three channels hypothetically by using a smart home application and Natural Language Process (NLP) techniques. We note that SmartThing is used for our application. It also receives event logs from the application that are then used to verify the correlations. Only correlations that have passed verification are used to detect anomalies. Upon receiving event logs in real time, it checks them through verified correlations. When an event log does not match correlations, it reports the event as an anomaly. IMEO reuses the detection results for better performance. It saves details of identified anomalies and maintains an anomaly history. Upon reoccurring the same malfunction, it can detect the anomaly directly from the history, instead of performing the correlation matching process.

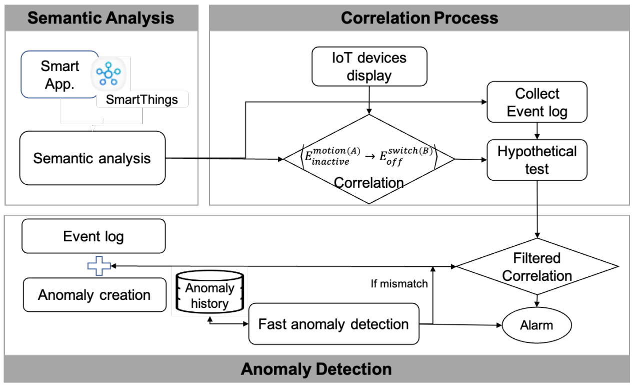

Fig. 3. System architecture of the proposed IMEO system.

A. Analyzing Semantics

The semantic analysis module first extracts semantics from the smart home application and then converts them to correlations (of application channels). To capture semantics, it investigates automation logics and related configurations, such as a temperature threshold in an air conditioner, in the application. For instance, given a semantic telling "Turn off the smart plug when illuminance is greater than 60," an E2E correlation $\langle E_{>60}^{illuminance} \rightarrow E_{off}^{smartplug}\rangle$ is generated. This can also be represented by an S2E correlation $\langle S_{>60}^{illuminance} \rightarrow$

$E_{off}^{smartplug}\rangle$. It is noted, however, that the S2E correlation does not guarantee to be true necessarily and has to be verified in the following step.

B. Mining Correlations

There are many pattern mining methods, but few achieve both good usability and high accuracy in the context of smart home applications. Supervised mining methods are more accurate, but often require well-annotated data sets or user interventions. Unsupervised mining methods can be applied to unannotated data, but they are less accurate.

To overcome limitations of existing methods, this paper proposes a semantic-based mining method. Semantic information includes device type and installation location, which can be obtained from a smart home application. Based on this semantic information, IMEO generates virtual correlations corresponding to physical channels and user activity channels. Each virtual correlation is then independently verified. This paper assumes that there will be no or very few anomalies in the training phase, as with other anomaly detection tasks.

1) Processing event logs: It is necessary to preprocess event logs for two reasons as follows. First, repetitive sensor readings introduce noise into raw event log data. For instance, some power meters periodically report similar but slightly fluctuating measurements. Second, numerical readings of a device cannot be incorporated into logical calculations. Therefore, our preprocessing module eliminates duplicated records and transforms numerical data to binary information. To this end, the module applies the Jenks natural breaks algorithm [30] to the remaining readings and classify them as *low* or *high*. Then, it looks at the events for a given attribute on each device and removes events that do not continuously change state. Finally, two temporally adjacent events for the same attribute of the device have opposite values. However, our measurement observed that there were values whose differences were too small to be binarized using the Jenks natural breaks algorithm. So, these values were all added up to find the average value. Based on this, a value was classified into *low* if it was lower than the average value, and classified into *high* if it was greater than the average value .

2) Generating virtual correlation: In addition to E2S correlations generated from the application channels, correlations can be generated with other semantic information such as device attributes and relationships between attributes in the physical and user activity channels. To this end, semantic information is utilized to construct a table displaying correlated attribute pairs, and then each pair is filled with devices with matching attributes to create a correlation.

For physical channel correlations, IMEO sets up seven physical properties (illuminance, sound, temperature, humidity, vibration, power, and air quality) related to IoT devices in smart home environment. An NLP technique is used to determine whether the attributes of two IoT devices can be associated through physical properties. To obtain IoT abstract attributes, we refer to the description from the SmartThings developer site [31]. In order to objectively evaluate the relevance between attributes and physical properties, Google's word2vec model [32] is used to calculate the semantic similarity score between each word in the list and the physical properties. Then, this score is used as the *correlation score* between the physical property and attribute. The top 10 attributes with the highest scores for each physical property are considered to be interrelated through the corresponding physical properties.

Attributes that represent users' characteristics can be expressed as *presence* and *motion*. Since IoT devices are always influenced by users, it is natually assumed that all attributes related to each physical properties in the physical channel are related to users. User activity channel correlations are formed by considering that each physical property has a correlation with *presence* and *motion*.

Eventually, IMEO can find attribute pairs considered to be correlated with each other, from which it is possible to identify all the attributes of IoT devices installed in a smart home environment. Given a pair of two correlated attributes α and β on device A and B, respectively, IMEO generates four E2E correlations $\langle E_a^{\alpha(A)} \rightarrow E_b^{\beta(B)}\rangle$, $\langle E_{a'}^{\alpha(A)} \rightarrow E_b^{\beta(B)}\rangle$, $\langle E_a^{\alpha(A)} \rightarrow E_{b'}^{\beta(B)}\rangle$, $\langle E_{a'}^{\alpha(A)} \rightarrow E_{b'}^{\beta(B)}\rangle$, and four E2S correlations $\langle E_a^{\alpha(A)} \rightarrow S_b^{\beta(B)}\rangle$, $\langle E_{a'}^{\alpha(A)} \rightarrow S_b^{\beta(B)}\rangle$, $\langle E_a^{\alpha(A)} \rightarrow S_{b'}^{\beta(B)}\rangle$, $\langle E_{a'}^{\alpha(A)} \rightarrow S_{b'}^{\beta(B)}\rangle$. Symmetrically, eight correleations can also be generated with events on $\beta(B)$ leading the correlations.

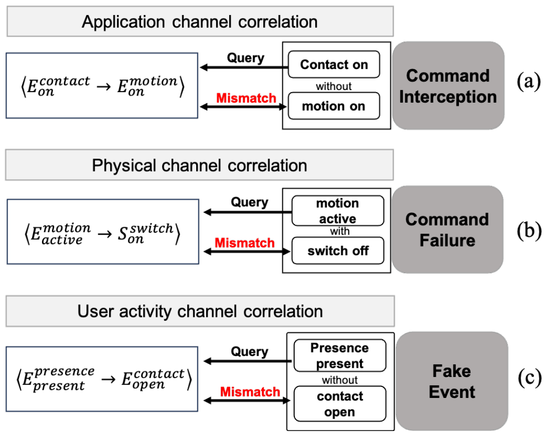
Application channel correlation
$\langle E_{on}^{contact} \rightarrow E_{on}^{motion}\rangle$
Query
Mismatch
Contact on
without
motion on
Command Interception
(a)
Physical channel correlation
$\langle E_{active}^{motion} \rightarrow S_{on}^{switch}\rangle$
Query
Mismatch
motion active
with
switch off
Command Failure
(b)
User activity channel correlation
$\langle E_{present}^{presence} \rightarrow E_{open}^{contact}\rangle$
Query
Mismatch
Presence present
without
contact open
Fake Event
(c)

Fig. 4. Anomaly detection process with three examples.

C. Detecting Anomalies

IMEO detects anomalies via two checking processes; *contextual check* and *consequential check*. The contextual check determines anomalies with E2S correlations. Suppose there is a correlation that a switch is *on* only when a motion sensor is *active* as shown in Fig. 4(b). Upon detecting the switch is *off* when the sensor is *active*, IMEO finds that this violates the correlation and determines to be an anomaly. The consequential check sees whether the next event occurs within 60 seconds when the preceding event of an E2E correlation occurs. If the next event occurs within the time boundary, the correlation is judged to be correct. For instance, Fig. 4(a) shows a correlation telling that an event "a contact sensor is *on*" should be followed by another event "a motion sensor is *on*". IMEO considers it normal if the motion sensor changes to on within 60 seconds after the contact sensor is turned on. It determines it to be abnormal if the motion sensor does not turn on within 60 seconds.

IMEO also performs a fast anomaly detection process. This is based on the feature, which has been frequently reported in user forums, that IoT devices malfunction repeatedly once malfunctioned. IMEO makes use of the feature to speed up the detection process while not sacrificing detection accuracy performance. Upon detecting an anomaly (i.e., a mismatch between an event log and correlation(s)), the anomaly detection module records a pair of event log, correlation(s) in a database of anomaly history. When the same malfunction occurs later, the module searches for the database and immediately determines if the event is abnormal before going through the checking processes. Once identified as anomaly this time, the resulting pair data is also recorded in the database. The number of repeated occurrence for each record is also saved in the database. Once sorted efficiently, this information can help accelerate the search process. Considering the repetition property of IoT devices, the fast anomaly detection process is expected to reduce detection time meaningfully.

VII. Experimental Setup

In order to evaluate the proposed IMEO system, we have built a real-world testbed as shown Fig. 5. Data was collected for three weeks to obtain event logs of IoT devices required for training and for one week for testing. We have applied correlations verified through the collected data to every events, from which the performance of IMEO is evaluated.

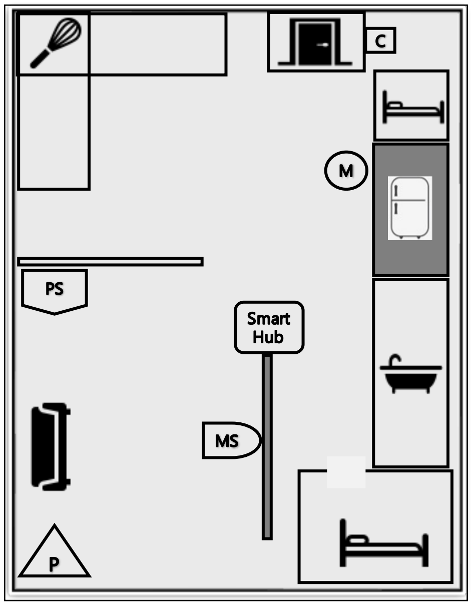

Fig. 5. Deployment layout of IoT devices.

A. Preliminary

1) Testbed: We have deployed IoT devices in a testbed of a private house consisting of three rooms and used Samsung SmartThings as an application. There are four residents in the testbed; one undergraduate student and three ordinary family members (two women and one man) who go to work during the day and returns home at night. None of them had any experience of using a smart home automation system. Residents are allowed to set automation of their interests that is fulfilled by IoT devices compatible with our smart home application. They are also given enough time to get used to the installed automation system before collecting data.

TABLE I. A List of IoT Devices Deployed in the Testbed

Device name	Attributes	Deployment	Abbr.
Motion Sensor (SmartThings)	motion	on wall	M
Multi Sensor (SiHAS)	motion, illuminance, humidity, temperature	on wall	MS
Contact Sensor (SmartThings)	contact, acceleration	on doors	C
Smart Button (SmartThings)	room control unit	on wall	PS
Smart Plug (SmartThings)	switch, power	as light lamp	P

2) IoT devices: Our testbed are deployed with five different types of IoT devices. Fig. 5 illustrates a deployment layout of devices, and Table I describes details of the devices including abbreviations. The motion sensor (denoted as M) is placed in the living room to detect the presence and motion of residents. The multi-function sensor (denoted as MS) is an IoT device that can detect human movements and perform a variety of roles, including detection of illumination, humidity, and temperature. The contact sensor (denoted as C) is able to detect the opening and closing of the front door. It can be used as a factor to determine whether residents are inside the house. The smart button (denoted as PS) is able to control the operation of smart devices in the testbed by unit. It can also check the temperature of the testbed. The smart plug (denoted as P) is a power plug used to control electrical devices.

TABLE II. Automation Rules Extracted from the Application

Index	Automation rules
R1	If (illuminance $<$30), then smart plug (on).
R2	If (illuminance $>$400), then smart plug (off).
R3	If C (opened) and MS (detected) for 15 minutes, then TV (on).
R4	If C (opened) and MS (undetected) for 20 minutes, then TV (off).
R5	If C (opened) and M (undetected) for 15 minutes, then P (off).
R6	If PS (pressed), then toggle TV
R7	If PS (double pressed), then toggle P.
R8	If PS (held), then turn off TV, air conditioner and P.
R9	If (illuminance $<$5), then TV (off).

3) Automation rules: We have extracted nine automation rules from the installed smart home application in the form of "automation operation if conditions arise". The extracted rules of the test bed are listed in Table II.

4) Ethical concerns: All participants are fully aware of installed devices and the application. Experiments did not use sensitive devices such as cameras or microphones. All the data collected was considered as sensitive personal identification information and thus was removed immediately after experiments. For testing purposes, anomalies are generated intentionally and injected into the event log (see Section VII-C). To avoid safety issues, the injected anomalies did not target safety-sensitive devices. We notified participants that there might be some deviation from existing automated rules, but did not disclose the details of the anomalies (e.g. device and time). In addition, participants were advised to maintain a normal lifestyle and not panic if abnormalities were found. The purpose of maintaining their lifestyle habits is to avoid biasing their behavioral during experiments. Details of injected anomalies were presented to participants after testing.

TABLE III. A Partial List of Verified Correlations Obtained from the Testbed

ID	Correlation
$\mathcal{C}1$	$\langle E^{illuminance(\mathsf{MS})}_{<30} \rightarrow E^{power(\mathsf{P})}_{on} \rangle$
$\mathcal{C}2$	$\langle E^{illuminance(\mathsf{MS})}_{>400} \rightarrow E^{power(\mathsf{P})}_{off} \rangle$
$\mathcal{C}3$	$\langle E^{contact(\mathsf{C})}_{open} \wedge E^{motion(\mathsf{M})}_{detect} \rightarrow E^{TV}_{on} \rangle$
$\mathcal{C}4$	$\langle E^{contact(\mathsf{C})}_{open} \wedge E^{motion(\mathsf{M})}_{undetect} \rightarrow E^{TV}_{off} \rangle$
$\mathcal{C}5$	$\langle E^{contact(\mathsf{C})}_{open} \wedge E^{motion(\mathsf{M})}_{detect} \rightarrow E^{switch(P)}_{off} \rangle$
$\mathcal{C}6$	$\langle E^{humidity(\mathsf{MS})}_{<40} \rightarrow S^{contact(\mathsf{C})}_{close} \rangle$
$\mathcal{C}7$	$\langle E^{humidity(\mathsf{MS})}_{>60} \rightarrow S^{contact(\mathsf{C})}_{close} \rangle$
$\mathcal{C}8$	$\langle E^{acceleration(\mathsf{C})}_{on} \rightarrow E^{motion(\mathsf{MS})}_{detect} \rangle$
$\mathcal{C}9$	$\langle E^{contact(\mathsf{C})}_{open} \rightarrow E^{motion(\mathsf{M})}_{detect} \rangle$
$\mathcal{C}10$	$\langle E^{motion(\mathsf{M})}_{detect} \rightarrow E^{contact(\mathsf{C})}_{open} \rangle$
$\mathcal{C}11$	$\langle E^{motion(\mathsf{MS})}_{active} \rightarrow E^{switch(\mathsf{P})}_{on} \rangle$
$\mathcal{C}12$	$\langle E^{acceleration(\mathsf{C})}_{active} \rightarrow E^{contact(\mathsf{C})}_{closed} \rangle$
$\mathcal{C}13$	$\langle E^{illuminance(\mathsf{MS})}_{<5} \rightarrow E^{power(TV)}_{off} \rangle$
$\mathcal{C}14$	$\langle E^{button(\mathsf{B})}_{held} \rightarrow E^{airconditioner}_{off} \rangle$
$\mathcal{C}15$	$\langle E^{button(\mathsf{B})}_{held} \rightarrow E^{power(TV)}_{off} \rangle$
$\mathcal{C}16$	$\langle E^{button(\mathsf{B})}_{held} \rightarrow E^{switch(\mathsf{P})}_{off} \rangle$
$\mathcal{C}17$	$\langle E^{power(\mathsf{P})}_{low} \rightarrow S^{switch(\mathsf{P})}_{off} \rangle$
$\mathcal{C}18$	$\langle E^{power(\mathsf{P})}_{high} \rightarrow S^{switch(\mathsf{P})}_{on} \rangle$
$\mathcal{C}19$	$\langle E^{switch(\mathsf{P})}_{on} \rightarrow E^{power(\mathsf{P})}_{high} \rangle$
$\mathcal{C}20$	$\langle E^{motion(\mathsf{M})}_{detect} \wedge E^{contact(\mathsf{C})}_{open} \rightarrow E^{contact(\mathsf{C})}_{close} \rangle$
$\mathcal{C}21$	$\langle E^{contact(\mathsf{C})}_{open} \wedge E^{motion(\mathsf{M})}_{undetect} \rightarrow E^{switch(\mathsf{P})}_{off} \rangle$
$\mathcal{C}22$	$\langle E^{motion(\mathsf{MS})}_{detect} \rightarrow S^{switch(\mathsf{P})}_{on} \rangle$
$\mathcal{C}23$	$\langle E^{motion(\mathsf{MS})}_{undetect} \rightarrow S^{switch(\mathsf{P})}_{off} \rangle$
$\mathcal{C}24$	$\langle E^{motion(\mathsf{MS})}_{detect} \rightarrow S^{contact(\mathsf{P})}_{closed} \rangle$

B. Training for Testing

1) Training IMEO: In the testbed, 14 E2E correlations are generated from the automation rules. Additionally, it generates 10 E2S correlations in the application channel, 749 correlations in the physical channel, and 417 correlations in the user activity channel, for a total of 1,176 correlations. Then, they are verified using 20,002 event logs collected during the three-week training phase. A total of 96 correlations passed the verfication test. Table III lists portions of the verfied correlations.

2) Findings: Belows share some interesting obervations from the testbed.

O1. While C is a contact sensor, it has an additional correlation, $\mathcal{C}12$= $\langle E^{accelation(\mathsf{C})}_{acitve} \rightarrow E^{contact(C)}_{closed} \rangle$. This implies that an event $E^{accelation(C)}_{acitve}$ is followed by another event $E^{contact(C)}_{closed}$, explaining that a front door (C) usually closes immediately after it opens.

O2. The E2S correlation $\mathcal{C}18$ indicates that the power of P is high only when P is on.

O3. The smart plug P turns a light on and off. Each time P is turned on, the power usage increases (see $\mathcal{C}19$ in the table).

O4. Physical and user activity channel correlations cannot be obtained without mining because they are not included in the application. On the other hand, there are correlations that can be easily extracted from the application but are difficult to mine. For instance, it is difficult to mine correlations that involve delays accurately, but their relations can be derived from rules like R3, R4, and R5.

3) Baseline: This research selects a correlation-only detection method [33], [34], [35] as a baseline approach, because it has been widely used for anomaly detection and is a well-established for mining correlations and rules. It is noted that IMEO is also based on correlation mining. For comparison, the correlation-only method is performed on the same data collected from our testbed.

TABLE IV. A List of Anomalies Selected and Simulated

Index	Anomaly type	Anomaly Creation Method
A1	Faulty/Fake events	To insert events into the data set
A2	Event loss/Interception	To remove events from the data set
A3	Bogus/Fake commands	To toggle from a bogus application
A4	Command failures(cyber)/ Command interception	To cut off devices' power supply
A5	Command failures(physical)	To cover bulbs with a paper
A6	Command failures(physical)	To unplug connected appliances

C. Generating Anomalies

To evaluate IMEO, six anomalies are simulated on the testbed as listed in Table IV. Because the same anomaly often occurs in IoT devices when malfunctions occur, six representative abnormalities are selected. To this end, we refer to the attacks from the literatures investigating IoT attacks and malfunctions frequently discussed in the Samsung SmartThings community. In order to simulate IoT devices as abnormal cases, event logs collected during the three-week training phase are arbitrarily modified. We also disrupt the automation rules, and resulting event logs are used to detect malfunctions. Tests are conducted multiple times for each malfunction. If an IoT attack has the same impact as a malfunction on event logs, we group and simulate it as one case. For instance, since faulty events due to sensor malfunctions and fake events due to attacks have the same impact, a total of 100 MS motion events are grouped and simulated by randomly injecting them into the test event log (see A1 in Table IV).

1) Faulty / Fake events: Events of devices such as motion sensors, presence sensors, and contact sensors are known to be unreliable, so we insert them to simulate the faulty/fake events.

2) Event loss / Interception: To simulate this, we randomly remove events on some devices from the test event log. We primarily select devices for which users have reported discomfort with event loss, such as multi-function sensors, contact sensors, and motion sensors.

3) Bogus / Fake commands: Users have frequently reported inconvenience that both smart lights and plugs have been unexpectedly turned on or off. To simulate these, a bogus application has been developed, that IMEO does not know about, and arbitrary commands to the appplication turn the smart plug on and off randomly.

4) Command failures (cyber) and Command interception: Our experiments simulate command errors and command interceptions in the cyber part of the smart plug. To this end, we make the power of target devices disconnected so that they are unable to respond to any commands. Experiments are conducted several times a day on each target device.

5) Command failures (physical): Command failures in the physical part are simulated in the multi-function sensor (illuminance) and the smart plug. The multi-function sensor is covered by blackout curtains, and appliances are physically unplugged from smart plugs. Two deivces still respond to commands with feedback events, but these commands have no physical effect. For each device, experiments are conducted several times a day.

VIII. Performance Evaluation

This section evaluates the performance of IMEO, the proposed anomaly detection system for IoT devices. First, it shows how correctly the proposed method can detect anomalies. Then, IMEO's performance is compared with the baseline approach (correlation-only detection method, COD) described in Section VII-B, which is followed by demonstration of overall performance in both methods. The last part evaluates the processing time of IMEO.

A. Evaluation Metrics

For evaulation, this paper uses the following metrics. Given the context of anomaly detection, *accuracy* is the ratio of event logs reported correctly (true anomalies and true normal events) to the entire events. *Precision* indicates the ratio of correctly detected anomalies to those reported to be abnormal (i.e., percentage of anomaly detection that is correct), and *recall* indicates the ratio of correctly detected anomalies to all the anomalies (i.e., percentage of anomalies that can be detected). *F1 score* is the harmonic mean of the precision and recall and represents the detection accuracy considering an imbalanced event situation where there are a relatively small number of anomaly events or vice versa.

$$Accuracy = \frac{True\,Positive + True\,Negative}{All\,Events}$$
$$Precision = \frac{True\,Positive}{True\;Positive + False\,Positive}$$
$$Recall = \frac{True\,Positive}{True\,Positive + False\,Negative}$$
$$F1\,Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

B. Results of Anomaly Detection

Table V summarizes detection results of IMEO across six different anomaly types described in Section VII-C. For three out of six types, IMEO successfully detects all events. The following detection examples illustrate how IMEO detects anomalies.

Detection 1. When entering the house, a resident is detected by the contact sensor C attached to the front door, which should be followed by a motion-active event of M in the living room. However, as the motion-active event of M becomes an faulty/fake event, the user activity E2E correlation $\langle E_{open}^{contact(\mathsf{C})} \rightarrow E_{detect}^{motion(\mathsf{M})} \rangle$ is violated and an anomaly is detected.

TABLE V. Detection Results of IMEO Across 6 Different Types of Anomaly Events

Anomaly Index	# of instances	Precision	Recall	Correlation(s) violated
A1	100	98.97%	94.12%	$\mathcal{C}22$
A2	100	100%	100%	$\mathcal{C}12$
A3	100	97.02%	98%	$\mathcal{C}12$, $\mathcal{C}19$
A4	100	100%	100%	$\mathcal{C}11$, $\mathcal{C}12$
A5	30	100%	80%	$\mathcal{C}1$, $\mathcal{C}2$
A6	100	100%	100%	$\mathcal{C}17$, $\mathcal{C}18$

Detection 2. When a resident leaves home, the motion sensor M in the living room is detected, and the contact sensor M attached to the front door must be detected immediately. Then, the front door is expected to be closed. However, as a malfunction occurs in which the event is lost or intercepted, it violates the correlation $\langle E_{detect}^{motion(M)} \wedge E_{open}^{contact(C)} \rightarrow E_{closed}^{contact(C)} \rangle$, and an anomaly is detected.

Detection 3. The smart plug P shall be off when the illumination sensor in MS exceeds a certain threshold by automation rules R1 and R2. Conversely, if the illuminance is lower than a certain criterion, the plug P should be on. However, the smart home application changes its behavior by ghost/fake commands. An anomaly is detected in violation of the E2S correlation $\langle E_{low}^{illuminance(\mathsf{MS})} \rightarrow S_{off}^{switch(\mathsf{P})} \rangle$ generated by the automation rule.

Detection 4. The smart plug P should behave appropriately according to automation rules related to the illuminance sensor MS. Due to command failure (cyber)/command interception, power supply to the smart plug is temporarily cut off. Consequently, the E2S correlation $\langle E_{low}^{illuminance(\mathsf{MS})} \rightarrow S_{off}^{switch(\mathsf{P})} \rangle$ is violated, resulting in the omission of all instances in this case.

Detection 5. Due to the blackout curtain, the illuminance value in the multi-function sensor MS is detected as low or zero. Thus, E2S correlations related to the illuminance events are detected as violation (anomalies A5). But, 6 instances are missing because the living room is brightened by natural light when anomalies occurr. In additon, as appliances connected to the smart plug P are broken (simulated by physically unplugging power cables), there is no operation at all on the plug. Therefore, events associated with A6 are detected as malfunction.

For Detection 1, 3, and 5, some instances are missing, which should be attributed to the incompleteness of the simulations injecting malfunctions. For example, six instances in Detection 1 are missing because fake motion-active events of M are injected during the time when real motion sensors (M) are logged as event logs. Similarly, two missed instances of Detection 3 are resulted from manipulating the smart plug P with a bogus application that is not allowed for the smart home application. Although malfunction is detected, there is a delay time until it is recorded in the event log because they are randomly manipulated operation states. For example, IoT devices turn on or off themselves by random bogus commands. However, there were cases in which the immediately preceding state of off is recorded although a device is on. In Detection 5, six instances are missing because the living room is brightened by natural light when anomalies occur.

TABLE VI. Anomaly Detection Results of Two Methods

Anomaly Index	Correlation-only (COD)		IMEO	
	Precision	Recall	Precision	Recall
A1	98.36 %	95.6 %	98.97 %	94.12 %
A2	91.11 %	82.79 %	100 %	100 %
A3	100 %	100 %	97.02 %	98 %
A4	100 %	100 %	100 %	100 %
A5	100 %	66.67 %	100 %	80 %
A6	100 %	100 %	100 %	100 %

TABLE VII. Comparison of Overall Performance

Method	Precision	Recall	F1 score	Accuracy
IMEO	99.33%	95.35%	0.97	99.98%
COD	97.83%	94.12%	0.95	99.32%

C. Performance Comparison

As summarized in Table VI, IMEO achieves better performance in 5 types. In A2 (faulty and fake events), especially, precision and recall increase from 91.11% and 82.79% to 100% in both. And recall in A5 (physical command failures) increases from 66.67% to 80%. COD can detect E2E, E2S correlation violations in smart home applications, but 17.22% events are lost for anomalies with event interception. In A3 (bogus and fake commands), however, there are small performance degradation.

Table VII compares overall performance of anomaly detection with two methods. IMEO has an average detection precision of 99.33% and a recall of 95.35% across 6 different types of anomaly events, while COD has 97.83% and 94.12%, respectively. It is noted that two methods show similar results in accuracy: 99.98% with IMEO and 99.32% with COD. However, IMEO performs better than COD with respect to the F1 score. This implies that IMEO is able to detect anomalies well even in a real-world smart home environment where malfunctions and/or attacks could occur rarely.

D. Fast Anomaly Detection

We believe that it is critical to reduce anomaly detection time using correlation data for increasing the security of smart home automation. Once detecting an anomaly, in this sense, IMEO memorizes a detection record as described in Section VI-C. When the same malfunction occurs later, it can immediately determine if the event is abnormal by searching for the record history. On the other hand, if the future malfunction is not the same as the history, additional comparisons are made in IMEO, which is likely to take more time. Thus, our experimentation study also measures how long it takes to detect anomalies when using the baseline method and the proposed method. As compared in Table VIII, IMEO takes 1,666 seconds while COD takes 2,647 seconds on average. This shows that the fast anomaly detection process in IMEO can improve performance by 58.9%.

TABLE VIII. Processing Time to Detect Anomalies

	Correlation only (COD)	IMEO
Processing time	44m 07s	27m 46s

IX. Conclusion and Discussion

As IoT devices are integrated and combined with the physical environment, anomalies in them can have serious consequences in our daily lives. Previous solutions that only used a data mining technology to detect anomalies have been struggling with high rates of false alarms and missing many actual anomalies. It also takes a long time to detect anomalies even if they correctly detect anomalies. To overcome the limitations, this paper proposed an anomaly detection system, named IoT Malfunction Extraction Observer, that made use of semantic information at smart homes such as smart home applications, configurations, device types, and installation locations. IMEO used event logs to detect malfunctions to take advantage of semantic information from different channels (smart home applications, physical activities, and user activities). It also provided an easier and faster way to detect frequently repeated malfunctions. These allowed for more reliable malfunction detection. We developed and deloyed the proposed method in a real-world testbed. Experiments were conducted with various abnormal instances, and the results demonstrated that the proposed method achieved higher performance of detection accuracy with faster processing time.

A. Discussion

The evaluation results are very promising, but we consider IMEO as the first step in the anomaly detection using semantic information in smart homes. IMEO has some limitations that we are trying to address in the future.

First, correlations due to user activity channels are useful for detecting anomalies, but false events can occur if there is a deviation in user activity. Such cases rarely occur, but we found them during evaluation. Some events generated events that were irrelevant to the correlation when a user encountered an abnormal situation (e.g., a state with an open front door). One day, for instance, a person wants to read in his or her living room, so he or she turns on the extra light, not the living room light, to increase the illumination. If this rarely happens during the experiments, malfunctions can occur. The key question is how to constantly update correlations to adapt to changes in IoT devices and user activities.

Next, IMEO can only compare and experiment with correlations that result in forward and backward events within a short interval. Correlations that require long intervals, such as the relationship between turning on the air conditioner and temperature events, cannot be detected yet.

Third, the physical constraints of IoT devices are problems to be solved. For example, some IoT devices may be placed relatively far away, and physical channel correlations between them may be very small. One way to solve this problem is to explore the correlations across the entire home rather than separate rooms, which will lead to more correlations between IoT devices. The opposite can also be considered. Recently, the size of houses has decreased, and more and more IoT devices are being installed in the houses. In this case, the redundancy of the correlations can be strong and this information needs to be interpreted differently.

Last, an attacker who knows correlation, that is, semantic-based detection, can construct an attack that does not violate correlation to avoid detection. The study of robustness to this type of attack will be an interesting topic. The key to IMEO execution is that it imposes additional constraints on the attacker. In the correlation channel, each attribute is included in at least four correlations. Attacking the device without violating the correlation is a barrier for the attacker.

ACKNOWLEDGMENT

This work was supported by Incheon National University Research Grant in 2023.

REFERENCES

[1] F. Brügge, M. Hasan, M. Kulezak, K. L. Lueth, E. Pasqua, S. Sinha, P. Wegner, K. Baviskar, and A. Taparia, "State of IoT – Spring 2023," https://iot-analytics.com/product/state-of-iot-spring-2023/, May 2023.

[2] "Samsung SmartThings," https://www.smartthings.com/.

[3] "Apple Homekit," https://www.apple.com/home-app/.

[4] E. Fernandes, J. Jung, and A. Prakash, "Security Analysis of Emerging Smart Home Applications," in *IEEE Symposium on Security and Privacy*, May 2016, p. 636–654.

[5] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "SoK: Security Evaluation of Home-Based IoT Deployments," in *IEEE Symposium on Security and Privacy*, May 2019, pp. 1362–1380.

[6] N. E. ElHady and J. Provost, "A Systematic Survey on Sensor Failure Detection and Fault-Tolerance in Ambient Assisted Living," *MDPI Sensors*, vol. 18, no. 7, June 2018.

[7] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse, "The Hitchhiker's Guide to Successful Residential Sensing Deployments," in *ACM Conference on Embedded Networked Sensor Systems*, November 2011, p. 232–245.

[8] "Door knocker going crazy," https://community.smartthings.com/t/door-knocker-going-crazy/55570.

[9] "Motion Detection False Positive," https://community.smartthings.com/t/motion-detection-false-positive/119816.

[10] "Mobile Device Presence Update Delay," https://community.smartthings.com/t/mobile-device-presence-update-delay/98672.

[11] "Samsung SmartThings User Forum," https://community.smartthings.com /t/october-2017-are-the-poltergeists-back-devices-randomly-turning-on/101402.

[12] 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 *USENIX Security Symposium*, August 2019, pp. 1133–1150.

[13] L. Russell, "Wireless security monitoring versus a cellular jammer," https://www.home-security-systems-answers.com/wireless-security-monitoring.html, 2014.

[14] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, "Smart-phones Attacking Smart-homes," in *ACM Conference on Security and Privacy in Wireless and Mobile Networks*, July 2016, p. 195–200.

[15] H. Liu, T. Spink, and P. Patras, "Uncovering Security Vulnerabilities in the Belkin WeMo Home Automation Ecosystem," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, March 2019, pp. 894–899.

[16] S. Notra, M. Siddiqi, H. H. Gharakheili, V. Sivaraman, and R. Boreli, "An Experimental Study of Security and Privacy Risks with Emerging Household Appliances," in *IEEE Conference on Communications and Network Security)*, October 2014, pp. 79–84.

[17] M. Fránik and M. Čermák, "Serious flaws found in multiple smart home hubs: Is your device among them?" https://www.welivesecurity .com/2020/04/22/seriousflaws-smart-home-hubs-is-your-device-amongthem/, 2020.

[18] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O'Flynn, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," in *IEEE Symposium on Security and Privacy*, May 2017, pp. 2375–1207.

[19] H. Chi, Q. Zeng, X. Du, and J. Yu, "Cross-App Interference Threats in Smart Homes: Categorization, Detection and Handling," in *IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2020, pp. 411–423.

[20] H. Chi, Q. Zeng, X. Du, and L. Luo, "PFirewall: Semantics-Aware Customizable Data Flow Control for Smart Home Privacy Protection," in *Network and Distributed Systems Security (NDSS) Symposium*, February 2021.

[21] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "HoMonit: Monitoring Smart Home Apps from Encrypted Traffic," in *ACM conference on Computer and Communications Security*, October 2018, p. 1074–1088.

[22] L. Cheng, K. Tian, and D. Yao, "Orpheus: Enforcing Cyber-physical Execution Semantics to Defend Against Data-oriented Attacks," in *Annual Computer Security Applications Conference (ACSAC)*, December 2017, p. 315–326.

[23] K. Kapitanova, E. Hoque, J. A. Stankovic, K. Whitehouse, and S. H. Son, "Being Smart about Failures: Assessing Repairs in Smart Homes," in *Annual Computer Security Applications Conference (ACSAC)*, September 2012, p. 51–60.

[24] J. Choi, H. Jeoung, J. Kim, Y. Ko, W. Jung, H. Kim, and J. Kim, "Detecting and Identifying Faulty IoT Devices in Smart Home With Context Extraction," in *IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2018, pp. 610–621.

[25] R. van der Meulen and J. Rivera, "Gartner Says a Typical Family Home Could Contain More Than 500 Smart Devices By 2022," http://www.gartner.com/newsroom/id/2839717, Gartner, Tech. Rep., 2014.

[26] W. Ding and H. Hu, "On the Safety of IoT Device Physical Interaction Control," in *ACM conference on Computer and Communications Security*, October 2018, p. 832–846.

[27] J. Han, A. J. Chung, M. K. Sinha, M. Harishankar, S. Pan, H. Y. Noh, P. Zhang, and P. Tague, "Do You Feel What I Hear? Enabling Autonomous IoT Device Pairing Using Different Sensor Types," in *IEEE Symposium on Security and Privacy*, May 2018, pp. 836–852.

[28] Z. B. Celik, G. Tan, and P. D. McDaniel, "IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT," in *Network and Distributed Systems Security (NDSS) Symposium*, February 2019.

[29] A. K. Sikder, H. Aksu, and A. S. Uluagac, "6thSense: A Context-aware Sensor-based Attack Detector for Smart Devices," in *USENIX Security Symposium*, August 2017, p. 397–414.

[30] G. F. Jenks, "The Data Model Concept in Statistical Mapping," *International Yearbook of Cartography*, vol. 7, pp. 186–190, Oct. 1967.

[31] "SmartThings Developers," https://developer.smartthings.com/docs/.

[32] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, December 2013, pp. 3111–3119.

[33] C. Fu, Q. Zeng, and X. Du, "HAWatcher: Semantics-Aware Anomaly Detection for Appified Smart Homes," in *USENIX Security Symposium*, August 2021, pp. 4223–4240.

[34] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," in *International Conference on Very Large Data Bases*, September 1994, p. 487–499.

[35] S. S. Khan and M. G. Madden, "One-Class Classification: Taxonomy of Study and Review of Techniques," *The Knowledge Engineering Review*, vol. 29, no. 3, p. 345–374, January 2014.