Decision-Tree Based Root Cause Localization for Anomalies in Smart IoT Systems

Michael Wang
Methacton High School
1005 Kriebel Mill Rd
Eagleville, PA 19403, USA
MW668@student.methacton.org

Chenglong Fu, Xiaojiang Du
Dept. of Computer and Information Sciences
Temple University
Philadelphia, PA, 19122, USA
{chenglong.fu, dux}@temple.edu

***Abstract*—With the rapid growth of Internet of Things (IoTs), Internet-connected devices and home appliances gain popularity on the consumer electronic market. New home IoT products with built-in network connections and intelligent functionalities are quickly rolled out to the market. As predicted by Gartner, there will be more than 500 IoT devices deployed in a typical household by 2022. The easy device integration and advanced automation logic also brings new challenges with regard to security and privacy. IoT devices have been reported as unreliable because of the constraints in costs and resources. Anomalies of IoT devices include malfunctions of the physical part or the cyber part of an IoT device, as well as abnormal behaviors due to malicious attacks. Abnormal IoT devices could cause severe consequences, because they reside in the home environment and have critical functions that can change the physical world, such as door (smart lock) opening, smart oven burning (which could cause fire), or smart water valve opening (which could cause flooding). In this paper, we study the important issue of localizing the root cause of anomalies in a smart environment (e.g., smart homes and smart offices). We propose to use decision trees for efficient and effective anomaly root cause localization. We construct decision trees from automation rules that control the operations of smart IoT devices in a smart environment. Our performance evaluation on data collected from real smart homes demonstrate the effectiveness of our proposed approach.**

***Keywords*—IoT, smart home, smart environment, anomaly, root cause localization, decision trees**

I. Introduction

Along with the rapid growth of the Internet of Things (IoTs), Internet-connected devices and home appliances gain popularity in the consumer electronics market. New home IoT products with built-in network connections and intelligent functionalities are quickly rolled out to the market. Traditional home electronic devices can also be easily converted to IoT devices by connecting them to smart plugs. As predicted by Gartner, there will be more than 500 IoT devices in a typical household [2] by 2022. The increasing number of IoT devices facilitate the process of smart homes evolving from isolated devices for remote monitoring and controlling to integrated platforms such as SmartThings, Homekit, and Alexa. These platforms not only provide universal interoperability among heterogeneous IoT devices from different manufacturers, but also allow home IoT devices to work autonomously according to user-specified trigger-action programs.

Despite the development of the smart home, the easy device integration and advanced automation logic also brings new challenges with regard to security and privacy which have been studied by many recent research works [3,11,12]. IoT devices have been reported as unreliable because of the constraints in costs and resources [15,18]. On one hand, IoT devices make it possible for cyber space anomalies to affect the physical world and induce severe consequences to home safety. For example, the loss of network packets that carry the command towards a smart valve may cause the room flood. On the other hand, users can pay less attention to their home electrical device by relying on the home automation system, which makes the device anomalies harder to be noticed until they cause severe damages. For instance, the connection loss of a smart plug could prevent it from cutting off the power to a connected radiation heater. This greatly increase the risk of fire hazard because the user may think the plug should have already been turned off automatically. Moreover, due to the interoperability, IoT devices with different sensing and actuating capabilities are chained together, which further exaggerates the impact of anomalies because abnormal behaviors of one device could trigger inappropriate actions of other chained devices according to various automation logic.

Localizing the root cause of smart home devices is a challenging task due to the complexity of situations of device malfunctions. When a user reports a device anomaly, it could either be caused by the reported device itself or by other devices that are associated with it through automation rules. Some prior works have explored the possible solution of anomaly localization by using Random Forest or Naïve Bayes [1]. However, they all require long periods of training and are not efficient enough to deal with real-world anomalies, which need to be handled quickly to avoid further damage.

In this paper, we propose a root cause localization system that utilizes home automation rules to infer possible root cause devices of each reported anomaly and localize the final root cause device by getting their intersection. More specifically, we build decision trees from automation rules that are associated with each reported anomaly. We go through the tree by checking the history devices event log and get the output of possible root causes. Since a single device malfunction could cause anomalies of multiple other devices, the root cause could be accurately identified by comparing the output of decision trees from these associated devices. The proposed system is tested on a real-world testbed and is proved to be accurate and efficient in identifying the root cause of anomalies.

Contributions of this paper can be summarized as:

1) We propose a novel anomaly root cause localization system that utilizes semantic information of home automation rules.

2) We design a method to automatically build and traverse decision trees according to home automation rules and smart home event logs. The root cause localizing process requires no user intervention.

3) We evaluate our proposed system with data collected from the real-world smart home automation deployment.

The rest of the paper is organized as follows. In Section II, we discuss related works. In Section III, we describe our design of anomaly localization system with details of building and traversing decision trees. In Section IV, we present the procedure and results of evaluation of our proposed system on the real-world testbed. Finally, we conclude in Section V.

II. Related Works

Despite the popularity of smart home automation systems, many vulnerabilities are discovered that can cause critical threats to users' security and safety. In [11], authors find security vulnerabilities of a smart app's over-privileged and weak authorization of third-party cloud interface, which can be exploited to retrieve sensitive user data and inject spoof device events. In [15], Ronen *et al.* shows the vulnerabilities in smart light bulb's communication protocol and firmware and show the possibility of compromising a large number of devices by spreading a worm virus among them. [3] analyzes the large-scale botnet attack targeting consumer IoT devices caused by malware 'Mirai'.

As a result, a reliable anomaly detection and localization mechanism is required to protect home automation systems against possible cyber-attacks and system deficiencies. There are a series of research works published on this topic in recent years. Some works use ensemble methods to detect and localize anomalies by searching for violations of system invariants. [6] proposes methods to detect data injection attacks by comparing the system states' signatures with those generated from pre-created anomalous cases. CLEAN [17] implements a clustering-based outlier detector. It uses the Least Common Subsumer (LCS) to measure the similarity among data points (events) and reports isolated data points as outliers. IDEA [13] studies the impact of sensor failure on recognizing different activities. It leverages the redundancy of sensors on recognizing activities and reports the absence of an expected event as the sensor's failure. While some other works concentrate on incorporating semantic information of automation rules to detect and localize anomalies, HomeGuard [8,9] is the first work to study anomalies caused by Cross-App Interference (CAI) of home automation systems. The authors develop a semantic analysis tool to extract rules from automation apps with symbolic execution and program instrumentation. Celik *et al.* propose the mechanism of tracking sensitive data flow using taint analysis and discovering security violations via static code analysis in [4] and [5], respectively. In [12] and [14], authors utilize the same method to extract automation rules as the context of home automation systems. Based on it, the authors develop a context-based permission system and user-centric authorization system.

Prior to choosing decision trees as the main method to determine the root cause of anomalies, we research several other possible methods for root cause analysis. We find that Random Forest or Naïve Bayes [1] could be used. While both have their advantages and disadvantages, ultimately, we find that they are not as efficient as decision trees, in our case. As the possible methods that have been considered by us, we briefly discuss them below.

- Random Forest is essentially a collective decision based on multiple decision trees. The anomaly root cause localization is found using different decision trees based on the feature vector. (Device, device capability malfunctioning, time of malfunction) Since there are multiple decision trees that are trained using data, random forest would work as it is essentially the average decision of multiple decision trees.

- Naive Bayes is based on the concept of conditional probability. It can be summarized in the statement, "What is the probability of something happening given something." However, Naive Bayes can also have multiple conditions so the statement can also be, "What is the probability of something happening given something, something else, etc." You can characterise the localization of the root cause of an anomaly as "What is the probability that a device is the root cause given this device is malfunctions, etc." Essentially, for example, if trying to find a root cause, the algorithm says, "Because device 1 is not working, and device 2 connected to it is not working, and device 1 can only work if device 2 is on, device 2 is the root cause of the anomaly." The decision tree is more deterministic, but Naive Bayes uses the same basic concept but with probability instead.

III. Decision-Tree Based Root Cause Localization

Home automation rules provide very useful semantic information for identifying the root cause of reported anomalies within the home automation systems. Normally, the malfunction of a smart home device could cause anomalies of multiple other smart home IoT devices via different automation rules. For each reported anomaly, we can infer possible devices that cause anomalies by checking its related automation rules. Then, the real root cause device can be identified from the intersection of suspected devices derived from different anomalies. To automate this checking procedure, we propose a decision tree-based approach that encode a device's related automation rules into decision trees. In this section, we illustrate the procedures to build decision trees from automation rules and utilize those trees to localize the root cause of anomalies.

Table I. Example rules that are related to the living room light

Case1	**R1.1**	<MS=inactive, CS = closed, L=off>
	R1.2	<PS=away, --, L=off>
Case2	**R2.1**	<MS=active, CS=closed, L=on>
	R2.2	<CS-open, IS<30lux, L=on>

A. Decision Tree Construction

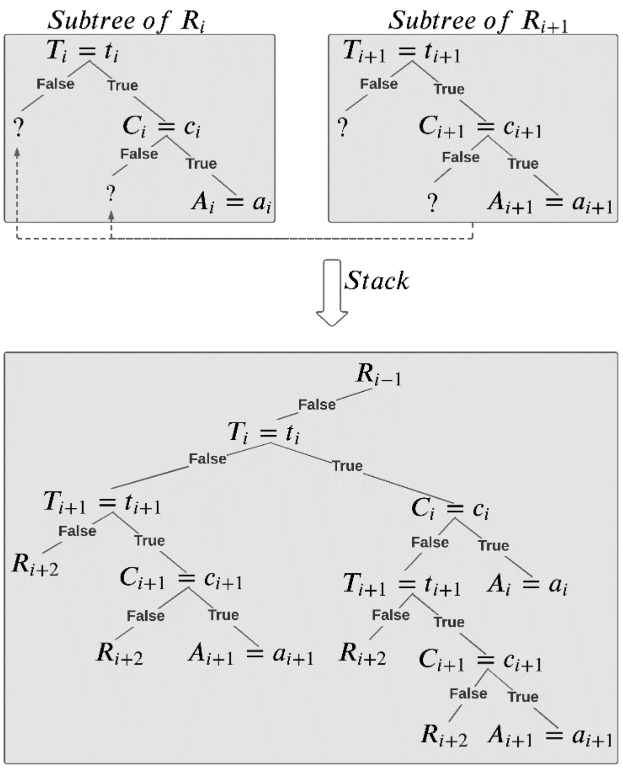

Fig. 1. Steps to build and stack subtrees.

A reported anomaly includes the anomalous actuator device's name and its abnormal status. For example, users could report an anomaly of "the living room light is falsely in the 'on' state". This can be either be caused by the failure of execution of an automation rule that turns the light off or the false action of an automation rule that turns the light on. For each case, we build a decision tree to localize the possible root cause device.

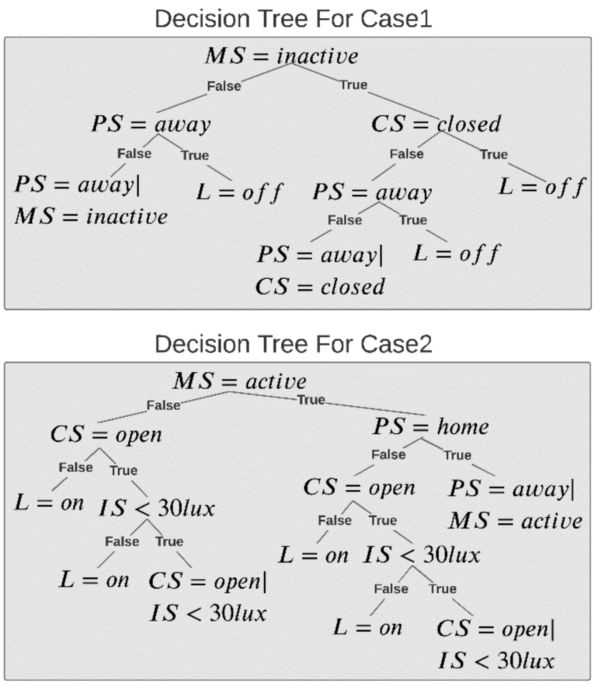

Fig. 2. Decision Trees for two cases

For the first case, we collect all rules $R_1, R_2, R_3, \cdots, R_n$ that have the action of "turning the living room light off", which possibly fails to execute. We consider standard trigger-action rules with optional conditions. We denote the trigger event as $T = t$, which means the trigger device T's states change to t. The ith rule R_i in the collection is represented as $\langle T_i = t_i, C_i = c_i, A_i = a_i \rangle$. As illustrated in Figure 1, the construction of the decision tree can be summarized as the following three steps:

1) For each rule, we build a right-chain subtree with its trigger being the root node, which is followed by the condition and action. The undefined leaf nodes (marked as ?) are left blank.

2) After getting all subtrees, we stack them by placing the subtree of R_{i+1} to the blank nodes of subtree of the previous rule R_i.

3) For each leaf node that remains blank in the last two levels, we find the path from it to the root node and set its value to be the 'OR' combination of nodes in the path that have the decision of 'False'.

For the second case, we collect all rules that have the action of "turning on the living room light" as the suspected triggering rules. Then, the decision tree is built with similar procedures as the first case except for the following differences: 1) when build the subtrees for a rule, the leaf node is set as the "OR" combination of its ancestor nodes instead of the action of the rule; 2) After stacking all subtrees, blank nodes are set as the common action of all rules.

Take the example anomaly, "the living room light is falsely on", as an instance, we assume the living room light can be controlled by the following four automation rules as listed in the Table I. For simplicity, we use the abbreviations of $MS, CS, PS, IS, and\ L$ to represent the motion sensor, the contact sensor, the presence sensor, the illuminance sensor, and the light in the living room. Rule $R_{1.1}$ and $R_{1.2}$ have the action of "turning off L" and are for case1. Rule $R_{2.1}$ and $R_{2.2}$ have the action of "turning on L" and are for case2. Then, we can build a decision tree for each case as shown in Fig. 2 in the aforementioned steps.

B. Root Cause Localization

We generate two decision trees for each reported anomaly. The first tree is used to look for possible failing rules, and the second tree is used to find unexpectedly executing rules. On receiving an anomaly at time t_0, we backtrack the event log of the home automation system to find the latest time (t_1) that the reported device changes to the current state. Then, we collect all events in the period of $[t_1, t_0]$ to form the searching event set. Then, we traverse the decision tree. At each node, a decision is made by checking whether the event of the node is in the searching event set. Finally, the leaf node we reach indicates the possible missing events that can cause the anomaly. The second tree is used for finding faulty events that trigger the last device status change at t_1 unexpectedly.

For this purpose, we collect events within 10 seconds before t_1 to form the search event set. We empirically select 10 seconds as the length of the time window because most home automation system can guarantee the finish of automation rule execution within 10 seconds. Then, we traverse the second decision tree in the same way as the first one. The leaf node we finally reach indicates the unexpected events that possibly triggers the anomaly.

In the example trees as shown in Figure 2, if no event of the motion sensor and the contact sensor happens since the last time the light is turned on, we can get the result of "PS = away or Contact = closed missing" by traversing the decision tree of case1. Then, assuming the last time the light is turned on by the user's manual operation, the decision tree of case2 is traversed with two consecutive false decisions and gets the result of "L = on", meaning the anomaly is possibly caused by the malfunction of the light which turns itself on unexpectedly.

IV. Performance Evaluation

A. Automation Rules

In order to determine the root cause of any anomaly that generates within the system. There needs to be a comprehensive set of rules and interactions that govern each Internet of Things device that we use. The rules are created by the user and form the basis of much of the project. This allows us to essentially find out what is the ground truth of an anomaly as it violates the rules and possible interactions that we have determined. For this project, we have created this list and collectively refer to it as the automation rules.

B. Anomaly Generation

For the decision trees to be of any use, they need the generated anomalies. These anomalies are created when some device causes it. Thus, the anomalies are generated when the user inserts a root cause. However, as mentioned before, the outcomes of only some trees may be deterministic, meaning that the resulting root cause from the decision tree is for sure one device, but there may be a case when a tree's root cause is one device out of several, it is simply unknown to that tree. Solving these cases are crucial and we need specific anomalies in order to bring about such a situation. For example, looking at Table II, we can simulate an anomaly that is caused by the missing event of cs1: off. This is to affect both rules that use cs1: off as an input, rule 1-6 and rule 1-9. Thus, forcing two trees to be used to find the root cause when it is unknown to the trees individually.

Table II. A part of the automation rules that were determined for this project. Each rule is given a designation, an input, and a root cause.

Designation	Condition	Result
1-1	ms2: no motion	bulb2.off
1-2	is1: >= 200 lux	bulb1.off
1-3	ps1: left	o2.off
1-4	is1: <= 20 lux	bulb1.on
1-5	ps1: left	bulb1.off
1-6	ms2: no motion	bulb1.off
1-7	ps1: left	bulb2.off
1-8	ps1: at home	bulb2.on
1-9	cs1: off	bulb2.on
1-10	ps1: at home	o2.on
1-11	ms1: motion detect	o2.on
1-12	ts1: >= 72F	o1.on
1-13	ms1: motion detect	o1.on
1-14	ps1: left	o1.off
1-15	ms1: no motion	o1.off
1-16	cs1: off	o1.off

C. Decision Tree Construction and Root Cause Localization

The creation of the decision trees is rather simple as each rule is essentially the result of a device state. Thus, many decision trees come down to one step trees. If it is not one device, it is the other from the rule. One example, not related to the previous rules, is given in Fig. 3:

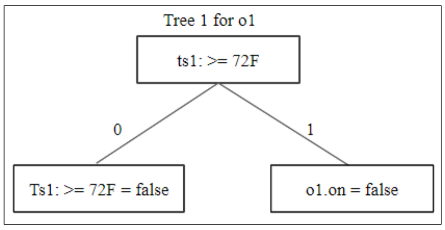

Fig. 3. A basic one-step decision tree that only incorporates one decision and only deterministic outcomes.

However, there are more complex trees that are created when there is ambiguity regarding the true root cause. There is ambiguity due to multiple rules having the same conditions but different results. For example, rules 1-9 and 1-16 have cs1:off as a root cause, but connect to different devices.

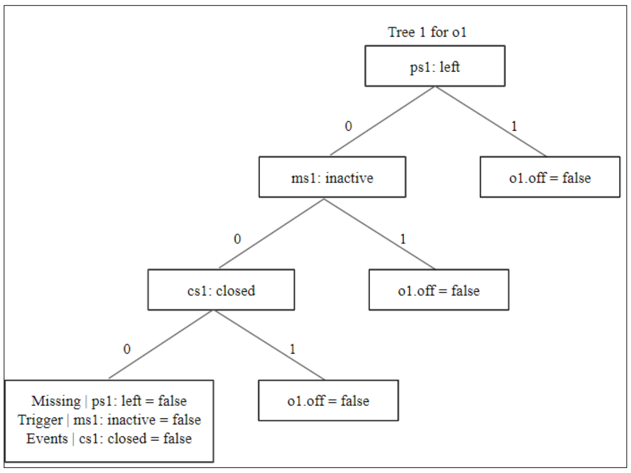

Fig. 4(a)

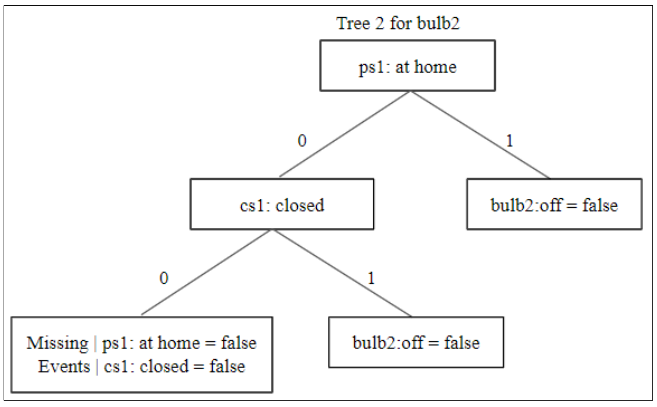

Fig. 4(b)

Fig. 4. A set of two decision trees. They are needed for the aforementioned ambiguous case. As seen for both trees, there is an end result on the tree that can be one of the devices.

As shown in Fig. 4, the two trees are used to check whether the expected automation rule fails to run properly due to a missing event. For the devices o1 and bulb2, we check events from the time of the alarm of an anomaly to the last state change. Once we gather this information, we go through the decision tree by checking whether the event in the node happens. Since, for example, there are no instances of ms1 or ps2 in the two devices' searching periods, both trees output the leaf node at the bottom left. Then, we take the intersection of the two sets, which is cs1: closed.

However, in a different situation, the two anomalies could be because of an unexpected trigger event. One example of this is the anomaly of o1 not turned off. In this case, we used rules 1-12 and 1-13 to find the root cause, and the decision tree is given in Fig. 5(a). For the first tree, we check events that happened within 10 seconds before the last state change of o1. This is because the automation rule is guaranteed to be finished executing in 10 seconds. Since there are no events from ts1 and ms1, going through the decision tree yields o1 as the root cause of the anomaly. As shown in Fig. 5(b), the second tree is built using rules 1-1 and 1-7, meant for the anomaly of bulb2: off. Similarly, we check within 10 seconds of the last state change of the bulb2 device. The result is that the only matched event is ms2: no motion. By getting the intersection of suspected root causes generated by two decision trees, we get an empty set, which means the reported anomalies are not falsely triggered by any unexpected device events. As a result, combining the results from decision trees in Figure 4, we can localize the root cause to be the missing event of the contact sensor turns closed (cs1:closed).

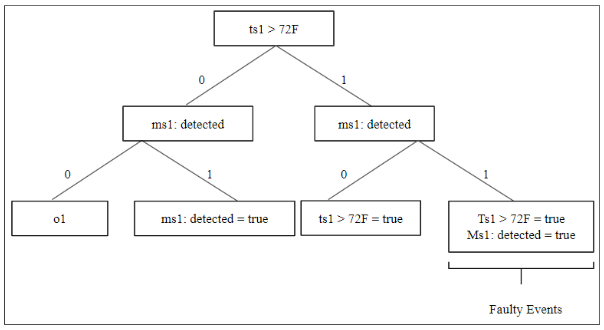

Fig. 5(a)

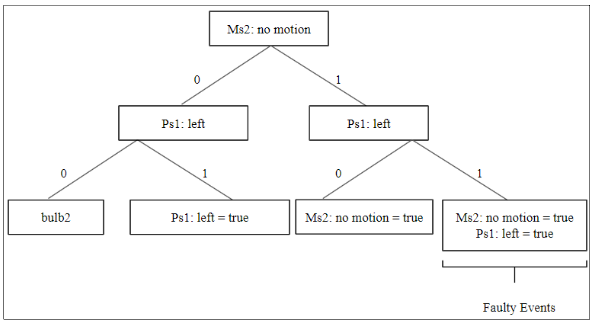

Fig. 5(b)

Fig. 5. The trees based on the automation rules, more specifically, rules 1-12 and 1-13 for the first tree and rules 1-1 and 1-7 for the second tree.

V. CONCLUSION

IoT devices are unreliable due to the constraints in costs and resources. Furthermore, IoT devices are vulnerable to and are targets for malicious attacks because these devices are directly connected to the physical world, such as smart homes and smart offices. Abnormal IoT devices could cause severe consequences, which may affect the safety of home residents or cause damage to property. In this paper, we studied the important issue of localizing the root cause of anomalies in a smart environment. We chose decision trees for efficient and effective anomaly root cause localization. The decision trees are constructed from automation rules of smart IoT devices, hence our approach is explainable, unlike a black-box style solution, such as neural networks or deep neural networks. We evaluated the performance of the decision-tree based approach on data collected from real smart homes, and the results showed that the proposed approach is very effective in localizing the root cause of anomalies.

Acknowledgement: This work was supported in part by the US National Science Foundation (NSF) under grants CNS-1828363, and CNS-2016589.

REFERENCES

[1] Random trees and Naïve Bayes, https://www.edureka.co/blog/artificial-intelligence-algorithms/

[2] Van der Meulen and J. Rivera,2014 "Gartner says a typical family home could contain more than 500 smart devices by 2022"

[3] Antonakakis, Manos, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, et al. 2017. "Understanding the mirai botnet." USENIX Security Symposium (USENIX Security).

[4] Celik, Z. Berkay, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. 2018. "Sensitive information tracking in commodity IoT." 27th USENIX Security Symposium (USENIX Security 18). 1687–1704.

[5] Celik, Z. Berkay, Patrick McDaniel, and Gang Tan. 2018. "Soteria: Automated iot safety and security analysis." 2018 USENIX Annual Technical Conference (USENIX ATC 18). 147–158.

[6] Chen, Yuqi, Christopher M. Poskitt, and Jun Sun. 2018. "Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system." 648–660.

[7] Cheng, Long, Ke Tian, and Danfeng Daphne Yao. 2017. "Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks." Proceedings of the 33rd Annual Computer Security Applications Conference. 315–326.

[8] Chi, Haotian, Qiang Zeng, Xiaojiang Du, and Jiaping Yu. 2020. "Cross-app interference threats in smart homes: Categorization, detection and handling." 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). 411–423.

[9] Chi, Haotian, Qiang Zeng, Xiaojiang Du, and Jiaping Yu. 2018. "Cross-App Interference Threats in Smart Homes: Categorization, Detection and Handling." arXiv arXiv–1808.

[10] Ding, Wenbo, and Hongxin Hu. 2018. "On the Safety of IoT Device Physical Interaction Control." Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 832–846.

[11] Fernandes, Earlence, Jaeyeon Jung, and Atul Prakash. 2016. "Security analysis of emerging smart home applications." 2016 IEEE Symposium on Security and Privacy (SP). 636–654.

[12] Jia, Yunhan Jack, Qi Alfred Chen, et al. "ContexIoT: Towards providing contextual integrity to appified IoT platforms." Proceedings of The Network and Distributed System Security Symposium, 2017.

[13] Kodeswaran, Palanivel A., Ravi Kokku, Sayandeep Sen, and Mudhakar Srivatsa. 2016. "Idea: A system for efficient failure management in smart iot environments." Proceedings of the 14th MobiSys. 43–56.

[14] Preuveneers, Davy, and Wouter Joosen. 2015. "SmartAuth: dynamic context fingerprinting for continuous user authentication." Proc. of the 30th Annual ACM Symposium on Applied Computing. 2185–2191.

[15] Ronen, Eyal, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. 2017. "IoT goes nuclear: Creating a ZigBee chain reaction." Security and Privacy (SP), 2017 IEEE Symposium on. 195–212.

[16] Sikder, Amit Kumar, Hidayet Aksu, and A. Selcuk Uluagac. 2017. "6thsense: A context-aware sensor-based attack detector for smart devices." 26th USENIX Security. 397–414.

[17] Ye, Juan, Graeme Stevenson, and Simon Dobson. 2015. "Fault detection for binary sensors in smart home environments." Pervasive Computing and Communications (PerCom). 20–28.

[18] Pa, Yin Minn Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. "IoTPOT: analysing the rise of IoT compromises." In 9th USENIX Workshop on Offensive Technologies (WOOT 15).