Aegis+: A Context-aware Platform-independent Security Framework for Smart Home Systems

AMIT KUMAR SIKDER, LEONARDO BABUN, and A. SELCUK ULUAGAC, Cyber-physical Systems Security Lab, Florida International University, USA

The introduction of modern Smart Home Systems (SHSs) is redefining the way we perform everyday activities. Today, myriad SHS applications and the devices they control are widely available to users. Specifically, users can easily download and install the apps from vendor-specific app markets, or develop their own, to effectively implement their SHS solutions. However, despite their benefits, app-based SHSs unfold diverse security risks. Several attacks have already been reported to SHSs and current security solutions only consider smart home devices and apps individually to detect malicious actions, rather than the context of the SHS as a whole. Thus, the current security solutions applied to SHSs cannot capture user activities and sensor-device-user interactions in a holistic fashion. To address these limitations, in this article, we introduce Aegis+, a novel context-aware platform-independent security framework to detect malicious behavior in an SHS. Specifically, Aegis+ observes the states of the connected smart home entities (sensors and devices) for different user activities and usage patterns in an SHS and builds a contextual model to differentiate between malicious and benign behavior. We evaluated the efficacy and performance of Aegis+ in multiple smart home settings (i.e., single bedroom, double bedroom, duplex) and platforms (i.e., Samsung SmartThings, Amazon Alexa) where real users perform day-to-day activities using real SHS devices. We also measured the performance of Aegis+ against five different malicious behaviors. Our detailed evaluation shows that Aegis+ can detect malicious behavior in SHS with high accuracy (over 95%) and secure the SHS regardless of the smart home layout and platforms, device configurations, installed apps, controller devices, and enforced user policies. Finally, Aegis+ yields minimum overhead to the SHS, ensuring effective deployability in real-life smart environments.

CCS Concepts: • **Security and privacy** → *Intrusion/anomaly detection and malware mitigation*; *Intrusion detection systems*;

Additional Key Words and Phrases: Smart home system, context-awareness, threat detection, security framework, malicious apps

ACM Reference format:
Amit Kumar Sikder, Leonardo Babun, and A. Selcuk Uluagac. 2021. Aegis+: A Context-aware Platform-independent Security Framework for Smart Home Systems. *Digit. Threat.: Res. Pract.* 2, 1, Article 6 (February 2021), 33 pages.
https://doi.org/10.1145/3428026

6

This work is supported by the US National Science Foundation (Awards: NSF-CAREER-CNS-1453647, NSF-1663051) and Florida Center for Cybersecurity's Capacity Building Program. The views expressed are those of the authors only, not of the funding agencies.
Authors' addresses: A. K. Sikder, L. Babun, and A. S. Uluagac, Cyber-physical Systems Security Lab, Florida International University, 10555 West Flagler St. EC 3900, Miami, Florida, 33174; emails: {asikd003, lbabu002, suluagac}@fiu.edu.

2576-5337/2021/02-ART6 $15.00
https://doi.org/10.1145/3428026

1 INTRODUCTION

The capabilities of the smart home devices have evolved from simply controlling lights and opening garage doors to connecting our living spaces to the cyber world [11, 34, 68]. Such functionality provides more autonomous, efficient, and convenient daily operations [58]. For instance, sensor-activated lights offer improved energy efficiency while smart locks and motion-activated cameras offer a more-secure home environment. Compared to early Smart Home Systems (SHS) with fixed device setup procedures and limited functionalities, modern SHSs have adopted a more user-centric, app-based model. Similar to the smartphone ecosystem, SHS's users can download apps from the vendor's app market (or even develop their own) and easily set up and control the smart devices, which makes SHSs more popular and versatile than ever [67].

The integration of programming platforms with smart home devices surely enhances the functionalities of SHSs, but also exposes the vulnerabilities of the smart devices to attackers [3, 55]. These attackers can release malicious apps in third-party markets and public repositories (e.g., GitHub) easily. Then, careless users may download and utilize them to control their devices. From here, the attackers can exploit smart home devices in several ways. For instance, they can perform denial-of-service attacks to obstruct normal operations of SHSs [69], compromise one device in SHS and get access to other connected devices [17, 23], or even leak personal information such as unlock code of a smart lock and get physical access to the home [10, 32]. Recently, a repository of malicious apps in different smart home platforms has been published exhibiting several vulnerabilities of the current smart home app development ecosystem [76]. Nonetheless, a security solution that comprehensively detects these emerging threats associated with SHSs does not currently exist and is direly needed.

Recent studies have proposed the implementation of enhanced permission models for SHSs, which depend on specific user permissions [32] or the source code analysis of the apps to detect vulnerabilities, which is only effective against specific types of attacks [16, 42]. Users can also install devices from different smart home platforms in the same physical environment, which minimizes the effectiveness of existing vendor-specific security solutions [72, 73]. Moreover, existing solutions focus on the detection of malicious activities that affect smart home devices and apps individually. However, a more holistic approach that also considers user activity contexts and sensor-device-user interactions (e.g., movement directions, sensors activated, rooms involved) is needed. For instance, if a user walks from the bedroom to the hallway, she may activate multiple devices and sensors along her path (i.e., walking context) in a certain sequence: moving towards the bedroom door, opening the door, entering the hallway, closing the door, and reaching the hallway. A user cannot simply skip some of these steps and reach the hallway directly from the bedroom. Again, device actions in an SHS are correlated with each other, which can be observed from a context-aware model. For example, a smart light triggered by the motion sensor can be verified by checking the user's presence in the home using a presence sensor. A contextual awareness of devices and applications that considers these types of sensor-device-user interactions can provide valuable information about malicious activities occurring in the SHSs, something that is missing in current security smart home solutions.

To address these emerging threats and shortcomings of SHSs, we present Aegis+, a novel platform-independent context-aware security framework to detect malicious behavior in an SHS. Aegis+ observes the changing patterns of the states (active/inactive) of smart home entities (sensors and devices) for user activities and builds a contextual model to detect malicious activities. Here, context-awareness refers to the ability of Aegis+ to understand the changes in sensors and devices' states due to ongoing user activities and determine if the behavior in the SHS is benign or not. Smart home devices are normally configured with different sensors to provide autonomous control and uninterrupted operations. Thus, different sensors in an SHS can sense user activities (motion, opening doors, etc.) and trigger associated devices to perform pre-defined tasks. Aegis+ correlates these sensor-device relations with different user activities and builds a context-aware model to define benign user behavior. Aegis+ also uses the app context to understand the trigger-action scenarios between smart home entities (sensors and devices) and automatically upgrades the framework if new devices are added

to the SHS. As a security framework, Aegis+ observes the current states (active or inactive) of sensors and devices and checks with the learned user behavior to detect any malicious behavior. Specifically, Aegis+ utilizes a Markov Chain–based machine learning technique to detect malicious behavior. Additionally, Aegis+ uses an action management system to alert the users in the event of malicious behavior and considers user responses to improve the context-aware model for better accuracy (adaptive training mode). We tested Aegis+ in real SHSs scenarios where 20 different users performed typical daily activities in three different home layouts generating over 85,000 sensor-device correlated events. Furthermore, we considered different device settings (sensor-device relations), apps, smart home platforms (four different platforms including Samsung SmartThings, Philips HUE, LIFX Smart Light, and Amazon Alexa) and user policies to evaluate the performance of Aegis+ against five different threats. Our extensive evaluation demonstrates that Aegis+ can detect different threats to SHS with high accuracy and F-score (over 95%). In addition, Aegis+ yields minimum overhead in terms of latency and computing resource utilization, making this solution suitable for real-life deployment. Note that this work is an extension of our previous work [62]. We significantly improved the framework from our prior work to build a platform-independent security framework and implemented it in different smart home layouts and platforms. We evaluated the performance with new user data and against new threats (e.g., attacks targeting voice-enabled smart devices).

Contributions: Our main contributions are noted as follows:

- **Aegis+.** We present a novel context-aware security framework to detect malicious activity in SHSs. We capture sensor-device co-dependence in smart homes to understand the context of the user activity and detect malicious behavior. Additionally, we implemented an action management system to alert users about Aegis+'s findings.
- **User-specific configurations.** We designed Aegis+ to support different smart home layouts and configurations. Aegis+ allows easy integration of new devices and apps creating app contexts and reconfiguring the training data automatically. We also introduced an adaptive training model to improve the detection mechanism from user responses automatically.
- **High accuracy and minimal overhead.** Through a detailed evaluation, we demonstrated how Aegis+ can detect different malicious activities in an SHS. Our results show that Aegis+ can achieve high accuracy and F-score and impose minimum overhead in the system.

Organization: The rest of the article is organized as follows: In Section 2, we present the background information. Then, we discuss the adversary model in Section 3. Section 4 details Aegis+'s architecture, and Section 5 details the implementation of Aegis+ in the real-life smart home environment. Section 6 evaluates the efficacy of Aegis+ in detecting different malicious behavior in SHS and details the performance of Aegis+ in different smart home layouts, platforms, and configurations. In Section 7 and Section 8, we discuss how different types of users will be benefited by deploying Aegis+ in real-life SHS and the related work, respectively. Section 9 concludes the article.

2 BACKGROUND

In this section, we describe the components of an SHS that we assume for Aegis+. We also detail different features used in Aegis+ to detect different malicious activities in SHSs.

The term *smart home* is commonly used to portray a residence comprising numerous connected entities (sensors and devices) that are capable of communicating with each other and that can be controlled both centrally (via a hub) and remotely (via a smartphone). In Figure 1, a typical architecture of an SHS is shown. Different smart home systems such as Samsung SmartThings, Amazon Alexa, and Google Home use similar architecture. The only difference among these platforms resides in the communication protocols used to connect the different components [33]. An SHS can incorporate a single-platform structure where all the smart devices and sensors

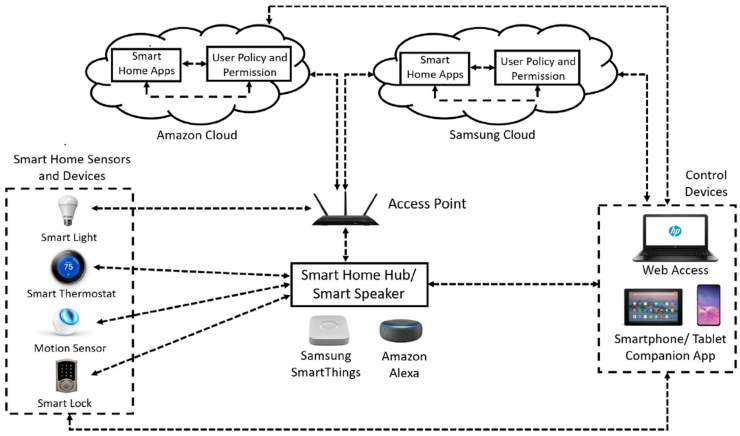


Fig. 1. A smart home environment and its main components.

are connected to a common access point (i.e., a hub). SHSs can also feature a multi-platform architecture where smart devices from different smart home platforms share the same physical environment without any interconnection. Both single and multi-platform SHS have four basic building blocks, as shown in Figure 1. The first block of the SHS comprises sensors and devices in the system. These smart home devices and sensors are connected to each other via a smart hub. As there is no generic interoperability standard among smart home devices, the hub provides a common access point for all the entities in the SHS. In some cases, hubs also act as a controlling device and allow users to control smart devices using voice commands (e.g., Amazon Alexa, Google Home). Modern smart home systems also allow devices to perform autonomous tasks as standalone devices. For example, *LIFX smart light* can directly connect to an access point such as Wi-Fi routers and perform pre-scheduled tasks. The installed smart home devices are connected to the hub, which can be further connected to both a cloud backend service and a smartphone/tablet companion app. Users utilize the smartphone app to control and configure the smart home entities or install different apps from the app stores. Indeed, we can group SHS architectures in two main categories: a cloud-based architecture where the installed apps run in the cloud backend (e.g., SmartThings), and hub-based architecture where the installed apps run the hub locally (e.g., Apple HomeKit). Users may also develop their own apps using the web interface of the cloud backend part of the SHS. For example, Samsung SmartThings allows its users to publish their own apps and share them with other users [53]. Users can develop their own app or simply copy the source code available online to install the app in their SHS.

2.1 New Design Features Considered by Aegis+

Context-awareness. Context-awareness refers to the ability of a system to use situational and environmental information about the user, location, and devices to adapt its operation accordingly [45, 61]. In an SHS, all the sensors and devices follow different trigger-action scenarios to perform tasks. Here, sensors are used to provide input in the devices (trigger) and devices take autonomous decisions (actions) based on these inputs. When a user performs a task in an SHS, several smart home sensors and devices may become active in a sequential pattern. The pattern of active devices and sensors is different but specific for distinct user activities. Existing SHS cannot observe these patterns in sensors' and devices' states over time and can not understand the context of the user activity. For example, while a user moves from one bedroom to a hallway, several devices and sensors become active in a sequential manner (Figure 2): moving towards bedroom door (sub-context 1: BL1, BLi1, BM1 are active), bedroom door opens (sub-context 2: BL1, BLi1, BM1, BD1 are active), entering the hallway (sub-context 3: BL1, BLi1, BD1, HLi2, HL2, HM2 are active), bedroom door and light close and reaches the hallway (sub-context 4: HLi2, HL2, HM2 are active). To complete the activity (moving from the bedroom to the hallway), the

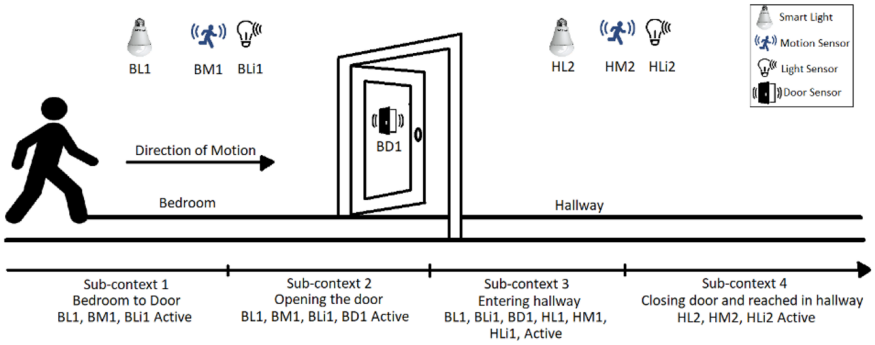


Fig. 2. Context-awareness feature, which is not considered in existing solutions to protect SHSs against cyber attacks.

user must follow the sub-contexts in the same sequential pattern. The user cannot skip one specific sub-context and move to the next one to complete the activity. For instance, the transition from sub-context 1 to sub-context 4 is not possible, as a user cannot go to the hallway from the bedroom without opening the door. Motivated by this, Aegis+ is designed to understand this property of SHS to build a context-aware model for different user activities and usage patterns and differentiates between benign and malicious activities of smart home devices and sensors.

Sensor-device co-dependence. In an SHS, sensors and devices can be configured as independent entities. However, they work in a co-dependent manner to provide autonomous functionalities [57]. For instance, smart lights can be configured with motion sensors to light up when motion is sensed. Here, the smart light depends on the input from the motion sensor while the motion sensor alone cannot provide any significant function in an SHS. The functions of devices and sensors create a co-dependent relationship with each other. In this way, sensors and devices in the SHS can build many-to-many co-dependent relationships. However, existing SHSs do not consider this co-dependent relationship and can not visualize the context of a user activity by observing the usage pattern of smart home entities. In short, sensors and devices in an SHS are configured as independent components, but in reality they are function-wise co-dependent. Aegis+ considers these relations to build the context of the user activities in an SHS.

User activity-device correlation. In an SHS, different users utilize and control smart home devices in multiple ways. For instance, a user can set a security camera to take pictures whenever a motion is detected in the associated motion sensors. However, users control devices in multiple ways. For example, a user can unlock a door by using the smartphone app or entering the code manually. Here, the state of the lock can be determined by user activity on the smartphone or by using a presence sensor to detect the user near the smart lock. In short, by observing the user activities in an SHS, it is possible to determine the normal operation of smart home devices. One can define normal or malicious user behavior with the user activity-device correlation. Current SHS cannot correlate user activity and device actions correctly, which is considered as a feature in Aegis+ to differentiate benign and malicious activities.

Multi-platform correlation. Modern SHSs allow users to install smart devices from different vendors within the same physical environment. These installed smart devices can perform autonomous tasks collectively via a common hub or as standalone devices connected to the Internet via an access point (e.g., Wi-Fi router). However, any on-going task in a smart device can be perceived by other standalone smart devices, even if they are not interconnected via a hub. For instance, if a user gives a voice command to a smart speaker to open the main door, a presence sensor connected to a smart light can verify the presence of the user inside the home environment and confirm the user command as a valid input in the smart home system. Here, the presence sensor can verify a benign user activity even if it is not connected to the smart speaker. Hence, smart devices from

different smart home platforms are correlated and can capture user activity context properly. In short, sensors and devices from multiple smart home platforms that are configured as standalone devices can be correlated via user activity context. Existing SHSs do not consider these correlations between standalone devices from multiple platforms, which is considered as a feature in Aegis+ to build user activity context in a multi-platform smart home environment.

3 PROBLEM SCOPE AND THREAT MODEL

In this section, we introduce the problem scope and articulate the threat model.

3.1 Problem Scope

This work assumes a fully automated smart home, *S*, with several smart home devices and sensors, which is illustrated in Figure 3. The SHS includes smart light, smart smoke detector, smart locks, smart thermostats, smart speakers, motion sensors, smoke sensors, light sensors, presence sensors, and temperature sensors. Here, the following sensor-device triggering rules are configured—the smart lights are configured with motion sensors, the smart smoke detector is configured with a smoke sensor. Additionally, the SHS allows manual device control by the users (e.g., unlocking smart lock with PIN). We also assume that the user utilizes customized third-party apps to control the devices. Furthermore, the SHS has more than one user authorized to control the devices in the system. Authorized users in the smart home systems can control the smart devices either via using controller devices such as smartphones or by using voice commands via the installed smart speaker in the SHS. We assume the following trigger-action incidents are happening throughout the day in the SHS: (1) one user is walking inside the bedroom but the lights are not triggered by the motion sensors, (2) one user is trying to unlock the smart lock using PIN code, (3) a fire alarm installed in the kitchen is being triggered in the system, (4) a smart light in the guest bedroom executes a blinking pattern while no user is present, (5) a user is trying to change the temperature of the smart thermostat but the temperature is not changing, (6) the smart speaker installed in living room executes a voice command to turn on the smart TV even though no user is present in the room.

We propose Aegis+ as a novel security framework that builds a context-aware model based on user activities to determine benign and malicious incidents in the SHS. Aegis+ answers several questions that may arise from the above-mentioned incidents: (1) What is the reason for no activity in the smart light installed in the bedroom?; (2) Is a legitimate user or an attacker is trying to unlock the door using PIN code?; (3) Is the fire alarm being triggered by a malicious app?; (4) What caused the smart light to blink and what is the intent of this activity?; (5) Why is the user not able to change the temperature of the smart thermostat?; (6) Why is the smart speaker executing a voice command without the presence of any user in the room? Aegis+ differentiates between normal and malicious activities happening in an SHS. Furthermore, Aegis+ detects malicious activities occurring in a device by observing the ongoing activities of all the connected devices in the SHS.

3.2 Threat Model

Aegis+ considers *anomalous user behaviors* (e.g, unauthorized users changing the device states) that may disrupt the normal functionalities of the SHS. Also, device vulnerabilities that may cause device malfunction or open doors to threats such as impersonation attacks and false data injection attacks are considered by Aegis+. Additionally, this work assumes carelessly designed and malicious smart home applications that may cause unauthorized or malicious activities in the SHS. These malicious activities may facilitate side channel and denial-of-service (DoS) attacks. Moreover, Aegis+ considers threats arising from malicious user commands on the controller device such as smart speaker. To better capture the threat model, we classify it in the following five categories:

- *Threat-1: Impersonation Attack.* An unauthorized smart home user can try to get access to smart home devices or applications by stealing valid user credentials or recording legitimate voice commands using a malicious app.

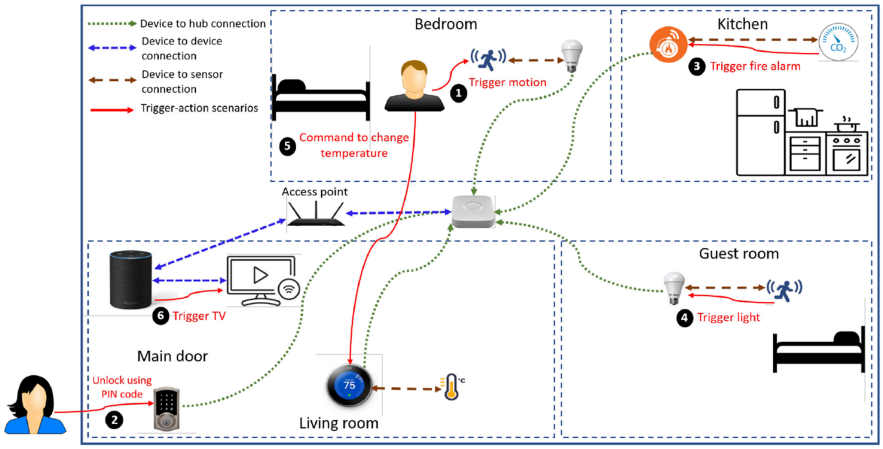


Fig. 3. Sample smart home with multiple users generating multiple trigger-action scenarios.

- *Threat-2: False Data Injection.* A malicious smart home app can exist in the SHS and inject forged data (e.g., sensor data, voice commands) to perform malicious activities. This threat represents false data injection in a smart home device.
- *Threat-3: Side channel attack.* A carelessly developed smart home app with design imperfections installed in the SHS can perform legitimate, yet vulnerable, side-channel activities that can be harnessed by other apps (considered malicious) in the system or the attacker himself. This threat represents a side channel attack on smart home devices.
- *Threat-4: Denial-of-service.* A malicious smart home app installed in the system can impede the normal behavior of other smart home devices and applications. This threat represents a denial-of-service attack in an SHS.
- *Threat-5: Triggering a malicious app.* A malicious smart home app can exist in the system that can be triggered by a specific activity pattern or device action (e.g., switching a smart light in a specific on/off pattern) in a smart home environment.

In Section 5, we present specific examples of attack scenarios that are used later to evaluate the effectiveness of AEGIS+. The information leakage caused by a compromised device or untrusted communication channel in the SHS is considered out of the scope of AEGIS+. We also assume that the data collected from the devices and central management system (e.g., Hub, cloud) is not compromised.

4 AEGIS+ FRAMEWORK

AEGIS+ has four main modules: (1) data collector, (2) context generator, (3) data analysis, and (4) action management, as depicted in Figure 4. First, AEGIS+ collects data from smart home entities (sensors and devices) from day-to-day user activities. The *data collector module* uses smart apps to collect all the device states (active/inactive) from installed smart home devices. Additionally, this module is fed from smart app rules extracted from different smart apps using the *app rule extractor* that are stored in the multi-platform rule repository.

The device state data is used to understand the context of the user activities and feeds the *context generation module*. This module creates context arrays generated from usage patterns and the predetermined user policies in the smart apps. Each context array contains the overall information of the user activities and device states in the SHS.

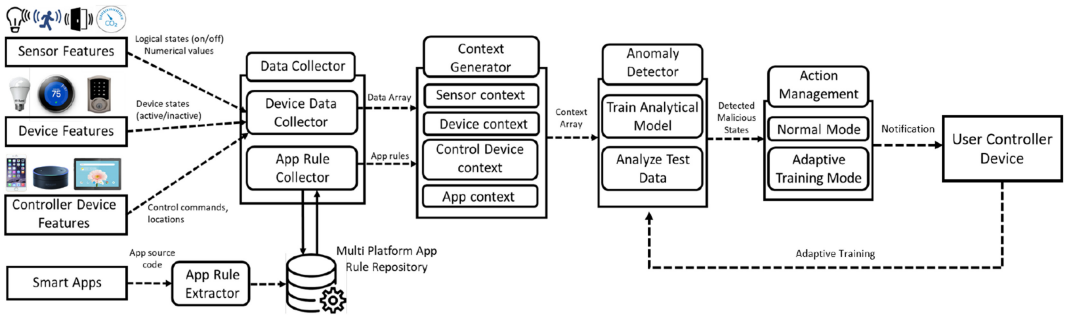


Fig. 4. Aegis+ framework.

The context arrays generated in the context generation module are used by the *anomaly detector module* to implement machine learning–based analysis and build the context-aware model of the SHS. Additionally, *anomaly detector module* decides whether or not malicious activities occur in the SHS.

Finally, the malicious activities detected by the anomaly detector module are forwarded to the *action management module*. This module notifies the users regarding the unauthorized activities. Also, it offers adaptive training mode where users can validate any false positive or false negative occurrence and re-train the detection model to improve the performance of Aegis+.

4.1 Data Collector Module

The data collector module has two sub-modules: *device data collector* and *app data collector*.

4.1.1 Device Data Collector. Aegis+ collects data from smart home devices and sensors using the data collector module. In an SHS, there can be multiple devices and sensors connected through a hub and operating in a co-dependent manner. Additionally, devices and sensors can act as standalone smart devices and perform different automated tasks individually. The data collector module of Aegis+ collects the state (active or inactive) of these devices autonomously for both connected and standalone smart devices and forwards these data to the context generation module. Based on the type of data, the collected data is governed by:

$$Data\ array,\ E = \{S, D, M\}, \tag{1}$$

where S is the set of features extracted from the sensors, D is the set of features extracted from the devices, and M is the features extracted from the associated controller devices (e.g., smartphone, smart tablet, smart speakers) in an SHS. We describe the characteristics of these features below.

- *Features extracted from sensors (S):* An SHS can comprise several sensors such as motion and light sensors. They sense changes in the vicinity of the devices and work as input to multiple devices. Sensor data can be both logical states (e.g., motion sensor) and numerical values (light sensor). For Aegis+, we consider both logical states and numerical values of sensors to create the context of user activities.
- *Features extracted from devices (D):* In an SHS, several devices can be connected with each other and with different sensors or act as standalone devices. These devices remain active based on user activities in a smart home environment. Aegis+ observes the daily activities of users and collects the device state data (active/inactive state) to build the context of the associated activity.
- *Features extracted from controller devices (M):* In an SHS, Smartphone or tablet works as a control device to the SHS and users can control any device using the associated smart app of the smart home. Additionally, smart speakers such Amazon Alexa, and so on, can be used as a controller device by allowing users to control installed smart devices using voice commands. Aegis+ considers any control command given from

the controller device as a feature to understand the context of user activity. The location of the connected controller device can also work as an input to control multiple devices. For example, a thermostat can be configured to the desired temperature whenever the user's smartphone is connected to the smart home network. Aegis+ considers the location of the controller device as a feature to build the context of user activities.

As user activities on an SHS can vary based on the number of users, Aegis+ considers multi-user settings to understand the user activity contexts correctly. Moreover, user activities also change based on the daily routine of users. For this, in the data collection process, Aegis+ also offers time-based activity settings (weekday and weekend settings).

4.1.2 App Rule Extractor. Modern SHSs offer an app-centric model where users install different apps to automate the functionality of smart home devices. These apps mostly define trigger-action scenarios for specific devices. For instance, an app can automate a smart light by configuring it with a motion sensor or a light sensor. Here, the sensors work as a trigger and the state of the smart light (on/off) refers to the action. These trigger-action scenarios can represent the app context, which can be used to validate the user activity context in an SHS. Additionally, the app context is also used to train the analytical model for new devices in SHS. Aegis+ utilizes source code analysis to identify and extract relevant app information related to these trigger-action scenarios. In this work, we assume that the source code of the smart home apps is freely available for analysis. We consider this assumption realistic, as some of the most popular smart home platforms implement their apps as open-source (e.g., Samsung SmartThings, openHAB, Windows IoT, and AWS IoT) [20, 37, 52]. Aegis+ implements a logic extractor to collect the smart app logic and infer the app context. The logic extractor takes the source of the apps and generates its Asymmetric Syntax Tree (AST). With the AST, the app rule extractor implements the Inter-Procedural Control (ICFG). The ICFG contains the nodes that define the different trigger-action events that are of interest of Aegis+ [16]. Further, the rule extractor visits every single node of the ICFG and collects the trigger-action scenarios that are defined in every app. For example, if a smart light is configured with a contact sensor, Aegis+ extracts the following logic from the app. In Section 5, we provide the implementation details of the app rule extractor in Aegis+. Finally, as we found most smart home apps that feature the same type of devices define similar trigger-action scenarios, we define an app rule extractor module that is able to analyze apps in parallel to Aegis+'s analysis. With this, we can collect and analyze apps from different platforms and create a multi-platform app rule repository that further feeds the data collector module previously described. There are several advantages to this design approach. First, it permits building a corpus of trigger-action scenarios that include information from several apps from multiple platforms simultaneously. Second, it prevents the need to update Aegis+'s framework every time new apps and devices are included in the SHS. Finally, it reduces the overhead introduces by Aegis+ as smart apps are mostly analyzed prior to Aegis+'s execution.

```
1 Trigger: Contact1
2 Action: Switch1
3 Logic 1: contact1 = on, light1 = on
4 Logic 2: contact1 = off, light1 = off
```

Listing 1. Trigger-action scenarios extracted from a sample app.

4.2 Context Generation Module

The data collector module forwards the collected data to the context generation module to build the context of different user activities. Then, the context generation module maps and aggregates the data to build context arrays. Each context array consists of information on the usage patterns in the SHS for different activities, which can be used for further analysis and to determine malicious activities in the system. The context array modeling process has the following steps:

- *Context of sensors:* Sensor features collected in the data collector consist of both logic state (on/off) and numerical values. Aegis+ observes the sensor data and generates the conditions of the sensors. These conditions represent the changing pattern of the sensor. If the current sensor value is different than the previous one, Aegis+ considers this as an active condition that is represented as 1. Similarly, conditions labeled as inactive are represented as 0.
- *Context of devices:* Data collector of Aegis+ collects device state (active/inactive) data for every connected device. These device state data are converted to logical states (1 represents active and 0 represents inactive) to build the context of user activities in an SHS.
- *Context of controller devices:* There are two features of the controller device (e.g., smartphone, tablet, speakers) that are collected by Aegis+: Control command for the devices and the location of the controller device. For any command from the smartphone, tablet, or smart speakers, Aegis+ considers the active condition of controller device, which is represented as 1 in context array or 0 otherwise. An SHS allows two different states to represent the location of the controller device—*home* and *away*. The home location indicates that the controller device is connected to the home network and away represents otherwise. Aegis+ represents the "home" location of the smartphone as 1 and the "away" location as 0 in the context array.

The final context array can be represented as follows:

$$Context\ Array,\ C = [\{S_1, S_2, \ldots, S_X\}, \{D_1, D_2, \ldots, D_Y\}, \{M_1, M_2\}], \tag{2}$$

where $S_1, S_2, \ldots, S_X$ captures the conditions of X number of sensors in the SHS, $D_1, D_2, \ldots, D_Y$ the conditions of Y number of sensors in the SHS, and M_1, M_2 the conditions of smartphone/tablet in the SHS.

The context generation module also generates the app's context. As most of the app's logic represents a trigger-action scenario, the context generation module converts the logic in a binary representation. For example, the logic extracted from the app presented in Listing 1 is given below:

```
1 App Context 1: contact1 = 1 , Light1 = 1
2 App context 2: contact1 = 0, Light1 = 0
```

Listing 2. Generated app context of a sample app.

Here, for the contact sensor, 1 and 0 represent the contact state from "open" or "close," respectively. Similarly, for the smart light, 1 and 0 represent the light state from "on" or "off," respectively. These app contexts are used to validate the sensor-device co-dependence captured in the context array. Additionally, these app contexts are used to update the training dataset whenever a new device is added to the SHS.

Finally, as noted before, we found that the app context is highly dependent on the specific type of devices the smart app controls. For instance, an app controlling a smart thermostat is likely to contain trigger-actions related to `Temperature = high, Thermostat = on`. Similarly, if the smart app controls a smart light and a motion sensor, the trigger-action `Motion = on, Light = on`. We use this finding to group together apps with similar expected trigger-action scenarios and improve the practicality of Aegis+. More specifically, we consider the devices that every specific app controls to infer expected trigger-action scenarios. With this, we found that Aegis+ does not need to consider and analyze every new app in the market to maintain the context generation module up-to-date. This finding considerably increases the usability of Aegis+ as incorporating new apps to control the SHS would not limit the effectiveness of the context generation module.

4.3 Anomaly Detector Module

In the third module, Aegis+ takes context arrays generated in the context generation module as input and trains a Markov Chain–based machine learning model that is used to detect malicious activities in SHS.

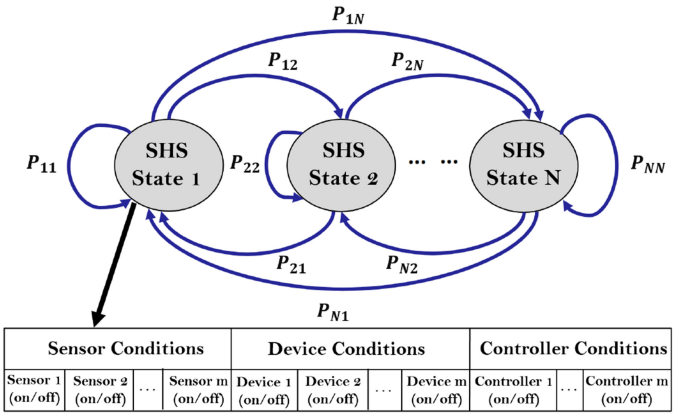


Fig. 5. Markov Chain model for AEGIS+.

The Markov Chain model is based on two main assumptions: (1) the probability of occurring a state at time $t+1$ only depends on the state at time t only and (2) the transition between two consecutive states is independent of time [59]. AEGIS+ uses this Markov Chain model to illustrate a series of events in an SHS. Here, a series of events denotes user activity and usage pattern, and the state represents the context array at a specific time t generated in the context generation module. The probabilistic condition of Markov Chain model is shown in Equation (3), where X_t denotes the state at time t for a user activity in the SHS [60]:

$$P(X_{t+1} = x | X_1 = x_1, X_2 = x_2 \ldots, X_t = x_t) = P(X_{t+1} = x | X_t = x_t), when,\ P(X_1 = x_1, X_2 = x_2 \ldots, X_t = x_t) > 0. \tag{3}$$

AEGIS+ considers the context array given in Equation (2) as an array of variables and observes its changes over time. For every user activity on an SHS, several context arrays are created. These arrays follow a different but specific pattern for different user activities. Each element of the context array represents the condition of a smart home entity (active/inactive states of sensor, device, or smartphone). For a distinct time, t, we consider the combination of all the smart home devices' and sensors' condition as binary output (1 for the active state of an entity and 0 for the inactive state). Thus, the number of total states (A) will be the exponent of 2 and can be represented as an n-bit binary number, where n is the total number of entities in the SHS. Let assume P_{ij} denotes the transition probability of the system from state i at time t to state j at time $t+1$. If the SHS has n number of entities (sensors, devices, and controllers) and $m = 2^n$ states in the system, then the Markov Chain model of AEGIS+ can be illustrated as in Figure 5. Here, each transition probability from one state to another state represents an element of the transition matrix. The transition probability matrix of the Markov Chain model constructed from context arrays can be represented by the following equation:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \ldots & \ldots & p_{1m} \\ p_{21} & p_{22} & p_{23} & \ldots & \ldots & p_{2m} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ p_{m1} & p_{m2} & p_{m3} & \ldots & \ldots & p_{mm} \end{bmatrix}. \tag{4}$$

If the SHS has $X_0, X_1, \ldots, X_T$ states at a given time $t = 0, 1, \ldots, T$, then the elements of the transition matrix can be shown as $P_{ij} = \frac{N_{ij}}{N_i}$, where N_{ij} denotes the number of transition from X_t to X_{t+1}, X_t is the state at time t, and X_{t+1} is the state at time $t+1$ [75]. If we represent Q as the initial state at time t = 0, then the initial probability

distribution of the SHS can be represented by the following equation:

$$Q = \left[q_1\ q_2\ q_3\ \ldots \quad \ldots\ q_m,\right], \tag{5}$$

where q_m is the probability that the model is in state m at time 0. Probability of observing a sequence of states $X_1, X_2, \ldots, X_T$ at a given time $1, \ldots, T$ in the SHS can be computed using the following equation:

$$P(X_1, X_2, \ldots, X_T) = q_{x1} \prod_{2}^{T} P_{X_{t-1}Xt}. \tag{6}$$

As the number of states in SHS depends on the total number of installed devices and sensors, predicting the probability of the next state can introduce overhead in terms of computing time and resource usage. Instead of predicting the next state using the Markov Chain model, Aegis+ determines the probability of transition between two states in the SHS at a given time. We train the Markov Chain model with the generated context arrays from the context generation module and construct the transition matrix. Using this transition matrix, Aegis+ determines the probability of transition from one state (i.e., context array) to another state over time. For example, in Figure 2, the transition between sub-context 1 and sub-context 2 is valid, as the user can perform this activity. However, the transition from sub-context 1 to sub-context 4 is invalid, as the user cannot go from the bedroom to the hallway without going through sub-context 2 and 3. Hence, the transition probability matrix can manifest the state transition based on user activity contexts. Let us assume a and b are SHS's overall state in time t and $t+1$. We determine the probability of transition from state a to b, which can be found by looking up in the transition matrix and calculating $P(a,b)$. As users cannot skip any sub-context of an activity, $P(a,b)$ will result in zero if transition from state a to b is malicious. Thus, Aegis+ defines benign or malicious device behavior based on user activities.

4.4 Action Management Module

Finally, the action management module notifies the users in the event of malicious activity in the SHS. The action management module has two operation modes—detection mode and adaptive training mode.

- *Detection mode:* In the detection mode, Aegis+ pushes a notification in the controller device's user interface (smartphone, smart tablet) to notify the users if malicious activity is detected. Aegis+ provides the device ID and the installed app names to the user for further action.
- *Adaptive training mode:* As Aegis+ builds a contextual model from user activities, it is important to verify the correct context of an ongoing user activity [48]. In an SHS, users can perform different activities in an irregular pattern. For example, a guest may come to the house, which will introduce some new activity patterns in the SHS. These abrupt data patterns may cause a higher false positive rate in the contextual model. To address this issue, Aegis+ offers the adaptive training mode, a user feedback process to improve the performance. In the adaptive training mode, whenever a malicious activity is detected, Aegis+ sends a notification to the controller device's user interface (smartphone, smart tablet) for user confirmation. Users can either confirm the malicious activity from the controller device or mark the activity as benign. If the user confirms the activity as benign, Aegis+ labels that activity context and trains the framework automatically. Hence, Aegis+ can correctly and automatically improve the training dataset by adding irregular or new user activity patterns.

5 AEGIS+ IMPLEMENTATION

We developed Aegis+ as a multi-platform security framework for SHSs. To implement and test the effectiveness of Aegis+ in real-life SHSs, we chose several smart home platforms and devices, including Samsung SmartThings, Amazon Alexa, Google Home, and so on. We provide details of Aegis+'s design considerations and implementation steps in the following subsections.

5.1 Design Features and Goals

Existing SHSs offer diverse sets of devices and sensors to automate day-to-day activities. However, configurations of SHSs depend on users' demands, device choices, home layout, home occupancy, preferences, and social relationships. As AEGIS+ builds the context-aware model based on data collected for user activities, we consider following design features and goals:

Home layouts. SHSs vary based on the home layout, as number of devices, sensors, and apps differ in different layouts. To test the effectiveness of AEGIS+, we selected three different smart home layouts: single bedroom apartment, double bedroom home, and duplex home. We selected these three layouts, as these are the most common rental units in USA [24].

Home occupancy. SHS is typically a multi-user environment where users share the installed smart home devices. As the home environment usually has more than one occupant, we considered several multi-user environments (two, three, and four users in the same layout) to implement AEGIS+ in real-life.

Types of devices and sensors. As SHS configuration depends on users' personal needs and preferences, the types of devices and sensors installed in SHSs vary in different layouts. Since AEGIS+ offers a platform-independent security framework, we considered 14 different types of off-the-shelf devices to build a real-life smart environment.

Date- and Time-dependent Activity. User activity in an SHS depends on the user's daily routine, which may change for different days of the week. For example, a working adult may spend more time at home on the weekends than weekdays, which increases user interaction in SHS. We considered this while designing AEGIS+ and implementing in a real-life environment.

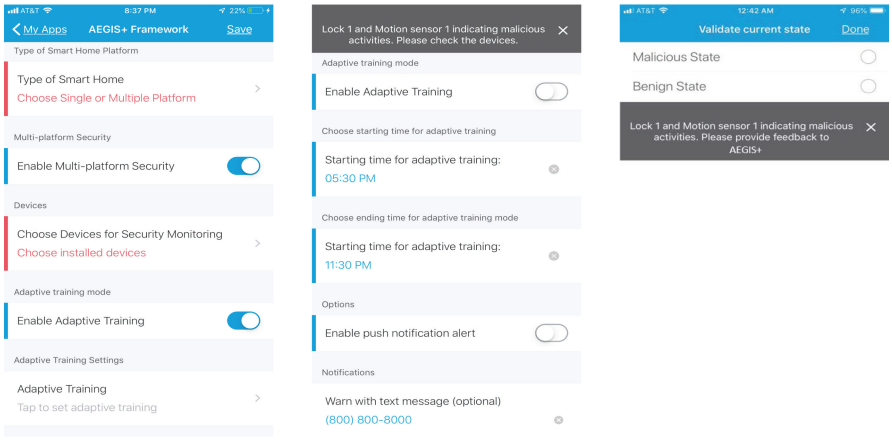
User-specific rules. Current smart home platforms allow users to build and define multiple policies to control smart home devices. The context of user activities may change based on user-defined policies in SHS. For example, a smart light can be controlled via the motion, door, or presence sensor. To understand the event associated with the light sensor and build user activity context, one must understand the user-defined policy enforced in the smart light. We addressed this property in the implementation of AEGIS+ by allowing users to define their own policies in the SHS in the data collection process.

Platform Independence. Modern SHSs allow users to install devices from different smart home platforms in the same physical home environment. These devices can be connected via a hub or perform tasks as standalone devices. AEGIS+ offers platform independence by allowing users to install smart home devices from different platforms in the same home layout. Also, we considered both hub-connected and standalone smart devices in the data collection and contextual model of AEGIS+.

5.2 Implementation

As the majority of the smart home environments support their functionality with services running in the cloud, we implemented AEGIS+ as a cloud-based solution. The integration of AEGIS+ in real-life smart home platforms was divided into the following integration phases:

- **Smart home integration.** In the smart home integration phase, we built a customized smart app (AEGIS+ app) to represent the data collector module described in Section 4. Figure 6 shows the sample user interfaces of the AEGIS+ app implemented in the Samsung SmartThings platform. This app has two modules: data collection and action management. In the data collection module, AEGIS+ app lists the device states as log files in SHS and forwards the logs to AEGIS+'s context generation module. AEGIS+ app also logs the information of the installed apps to generate app contexts in the context generation module. The action management module of AEGIS+ app captures any notification generated in the anomaly detector (implemented in the cloud) in the event of any malicious activity in the SHS and notifies the users. Addi-



(a) Aegis+ interface (b) Normal Mode (c) Adaptive Training Mode

Fig. 6. User interfaces of Aegis+ in different operation modes.

tionally, users can control operational modes (detection and adaptive training) of Aegis+ using the action management module.

- **Cloud integration.** We implemented the analytical model of Aegis+ (context generation and anomaly detector) as a cloud-integrated security framework. The context generation module collects device states forwarded from the Aegis+ app and creates user activity contexts by considering device states, sensor states, and timestamps. As users utilize different devices to implement their smart home environments, we also designed an app context extraction process capable of analyzing apps from multiple smart home platforms. We implemented a tool similar to the available static analysis tool to automatically analyze and extract the trigger-action scenarios that define the app context [12, 13, 16]. For trigger-actions defined for other platforms, we found that the manual analysis and extraction were feasible, as these platforms mostly define specific binary combinations of trigger-action interactions to control the devices. For the SmartThings apps, as they are written in Groovy [52], we used specific compile-time tools offered by the Groovy Metaprogramming capabilities [27]. We first used the `ASTTransformation` class to extract the app's AST and build its ICFG. From there, we take advantage of pre-defined class visitors to explore the different nodes in the ICFG and extract the different trigger-action scenarios defined within the app's event handler methods [16]. In total, we used the app context extraction tool to automatically analyze 485 Smarthings apps, extracting a total of 587 different trigger-action scenarios. These trigger-action scenarios were further stored in the multi-platform app rule repository that supports Aegis+ analysis. The app information forwarded from the Aegis+ app are cross-referenced with App rule repository to add app contexts into the training dataset for data validation purpose. The app contexts are only used in the adaptive training mode when the user installs a new device or new apps (details in Section 4.4). Aegis+ uses the generated app contexts to validate user activity context in a new smart home configuration to minimize the effect of new device and app addition. The device and app contexts are used to build the user activity context in an SHS by the anomaly detection module of Aegis+. The anomaly detection module uses this user activity context to train and detect any malicious events in the SHS. In the event of any malicious activity in SHS, the anomaly detector sends a notification to the Aegis+ app, which uses push notification to alert users in real-time. Figure 7 shows the implementation of Aegis+ in a multi-platform SHS.
- **Notification System.** To understand the malicious events occurred in the smart home environment, we implemented a user-customized notification system in Aegis+. The action management module of Aegis+

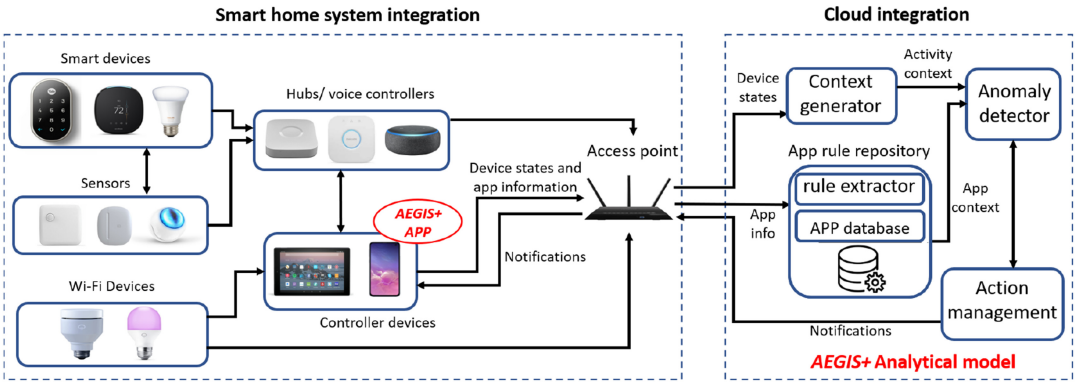
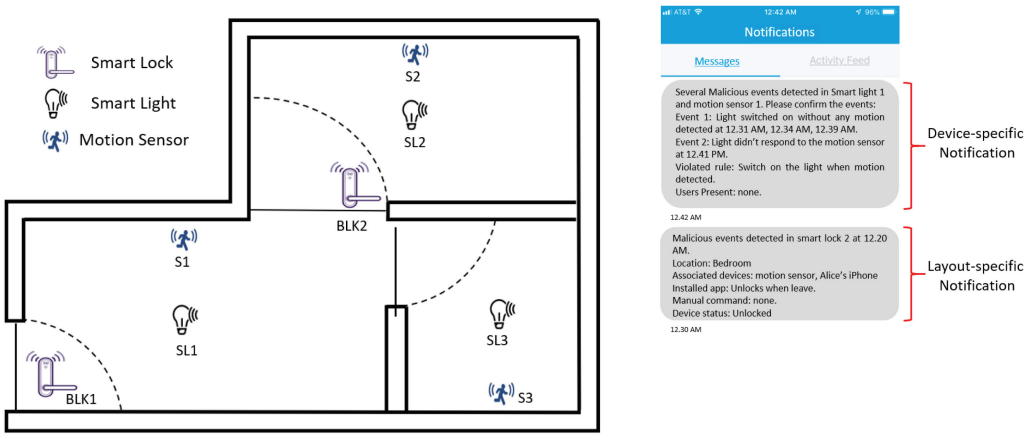


Fig. 7. AEGIS+ implementation in a multi-platform smart home environment.

(Figure 7) collects and sends a push notification to users' controller devices (smartphones or tablets) in case of a malicious event. Users can check and provide feedback to these notifications to improve the performance in the adaptive training mode. AEGIS+ offers two different customized options to provide details of malicious events to the users.

Device-specific notification. In an SHS, multiple users perform different tasks simultaneously. As AEGIS+ considers on-going user activities in an SHS to build a context-aware model, it is possible to generate multiple alert notifications due to diverse and simultaneous user activities. Also, attackers can target several smart devices and perform malicious activities concurrently. To detect malicious events effectively, AEGIS+ provides device-specific notification options to users. Here, AEGIS+ groups the malicious events occurred in a specific device and provides detailed information of the events including the time of occurrence, exploited device policy, app contexts, associated user activity, and so on. For instance, Figure 8(b) shows device-specific notification generated by AEGIS+ in an SHS. Here, a smart light, SL1, installed in the living room generated several malicious events that are notified to the user using the AEGIS+ app. The generated device-specific notification provides the nature of the events (light switching on/off without any motion sensor input) with additional details such as occurrence times, user policies, and so on. Users can also select a device and check all the generated alerts associated with the device at any time using the web interface of the smart home platform. In the adaptive training mode, users can select a particular malicious event and verify whether the malicious event is correctly identified or not to improve the performance of AEGIS+.

Layout-specific notification. Smart home environment differs based on installed devices, user needs, and home layouts. Hence, it is important to provide details of the malicious events depending on the user-customized smart home environment. Several smart home platforms such as WebThings allow users to upload their floorplan to identify the smart devices' locations within the home [40]. AEGIS+ adapts this feature and provides a web-based emulator to build and design a user-specific smart home layout (Figure 8(a)). Users can select the smart devices installed in their homes and provide specific details of the smart home environment such as device location, associated sensors, and so on. AEGIS+ creates a database to identify detailed information based on the device ID assigned by the system. The action management module uses this database to provide detailed information of a malicious event. For instance, if a malicious event occurs in the bedroom lock (BLK 2 in Figure 8), AEGIS+ identifies the location of the device and provides the list of apps and devices associated to the lock in the push notification, which is illustrated in Figure 8(b). This detailed representation helps users to understand the nature of the malicious event and act accordingly in adaptive training mode.



(a) Sample Home Layout in Aegis+ Emulator (b) Aegis+ Notification

Fig. 8. User interfaces of Aegis+ in different operation modes.

5.3 Data Collection and Real-life Testbed

To test the efficacy of Aegis+, we implemented a smart home testbed where the users had the freedom to design their own smart home environment and perform regular activities in a timely order. While collecting the user activity data, we considered five features that satisfy aforementioned design goals: *Anonymous User ID*, *User Role*, *Smart Home Layout*, *Activity Day-time*, and *User Policy*. For each user, we assigned an anonymous ID to ensure privacy. We also assigned a specific user role to each participant to understand the context of the user activities in a multi-user scenario. As mentioned earlier, we considered three different home layouts (single bedroom apartment, two bedroom home, and duplex home) and let the users choose their layout and smart home devices. Moreover, we considered users' daily routine and collected the user activities performed in both weekdays and weekends, separately. Finally, we allowed the users to define their own policies in the SHS during the data collection process.

We obtained the necessary Institutional Review Board (IRB) approval to collect daily usage data of an SHS with multiple users. We invited current smart home users to participate in our study by circulating university-wide open calls and community outreach via emails and flyers. We selected 20 smart home users to collect daily usage data and provided monetary compensation to the users. While selecting participants for our study, we considered several features: (1) owns more than one smart home device, (2) uses customized or vendor-specific apps/automation rules to control devices, (3) diverse user roles (working adults, housewives, young adults, student, etc.), (4) beginner level knowledge on using smart home devices, (5) shares smart home environment with multiple users. To create a smart home environment, we considered the most common off-the-shelf devices. We chose 8 types of devices and 6 types of sensors to build the smart home testbed (total 14 types of devices). The detailed device lists and apps selected in our SHS environment are given in Table 1. We collected data from 20 different individuals with different user roles, user policies, and smart home layouts. Our dataset consisted of over 85,000 events collected in a 15-day period. We considered two implementation scenarios based on smart home platforms while collecting user activity data.

- **Single platform SHS.** As discussed in Section 2, single platform SHS offers a single access point or hub where all the installed devices and sensors in a smart home environment are connected to perform various tasks collectively. We considered single platform SHS in our testbed to collect daily user activity data and implement Aegis+. We chose the Samsung SmartThings platform to create a single platform

Table 1. List of Devices Used in the Data Collection

Device Type	Model	Description	Selected Apps
Smart Home Hub	Samsung SmartThings Hub	• Works as a central access point for smart home entities. • Supports Wi-Fi, ZigBee,and Z-Wave.	• Samsung SmartThings App.
Smart Speaker	Amazon Alexa Google Home	• Works as a central access point for smart home entities. • Supports Wi-Fi, ZigBee,and Z-Wave.	• Amazon Alexa app. • Samsung SmartThings App integration.
Smart Light	Philips Hue Light Bulb	• Uses a separate communication bridge to connect with smart home hub. • Uses ZigBee to communicate with other components in SHS. • Supports up to 12 different sensors.	• Philips Hue app. • Brighten Dark Places. • Brighten my path. • Bright when dark and bright after sunset.
Smart Lock	Yale B1L Lock with Z-Wave Push Button Deadbolt	• Uses Z-Wave to connect with other devices. • Offers different pin code for different users. • Provides both manual and remote access.	• Lock it at a specific time. • Unlock it when I arrive.
Fire Alarm	First Alert 2-in-1 Z-Wave Smoke Detector and Carbon Monoxide Alarm	• Uses Z-Wave to connect with the hub • Provides built-in smoke and CO sensors.	• CO detector. • Smart alarm.
Smart Monitoring System	Arlo by NETGEAR Security System	• Uses Wi-Fi to connect with smart home hub. • Offers both live monitoring and still pictures.	• Cameras on when I'm away.
Smart Thermostat	Ecobee 4 Smart Thermostat	• Uses Wi-Fi to communicate with smart hub. • Can be configured with sensors.	• Ecobee connect. • It's too cold. • It's too hot. • Keep Me Cozy.
Smart TV	Samsung 6 Series UN49MU6290F LED Smart TV	• Connects with smart home hub using Wi-Fi.	• Power allowance. • Make it so.
Motion, Light, & temperature sensor	Fibaro FGMS-001 Motion Sensor	• Uses Z-Wave to connect with the hub. • Can be configured with different devices simultaneously.	• Any apps associated with smart devices.
Door Sensor	Samsung Multipurpose Sensor	• Uses ZigBee protocol to connect with smart home hub.	• Any app associated with smart devices.

smart home environment because of its large app market and compatibility with other smart devices [28]. We implemented the Aegis+ app as a third-party app in the Samsung SmartThings hub that uses the *ListEvent* command from SmartThings API to collect the device logs. These device logs are forwarded to the Aegis+'s context generation module. We also created an app context database that consists of 485 official *Samsung SmartThings* apps to cross-reference and create app contexts. Whenever users install an app in SHS, Aegis+ searches for the existing app context in the database and adds the context into the training dataset for data validation purposes. For any third-party app, users can manually use the source code of the app in Aegis+ and generate the app context, which is later added in the database.

- **Multi-platform SHS.** For multi-platform SHS, we considered four different smart home platforms and devices in a single physical environment. We chose Samsung SmartThings, Philips Hue, LIFX smart bulb, and Amazon Alexa as different smart home platforms. Here, the smart home platforms and devices are installed as independent entities and no common access point is considered in the installation. However, they share the same network access point and perform different tasks independently. We customized and

implemented Aegis+ app in each of these smart home platforms. Aegis+ app collects device logs using following APIs: *ListEvent* command from SmartThings API, *LIFX HTTP API* for LIFX smart lights [35], *HUE API* for Philips Hue [30], and *Smart home skill API* for Amazon Alexa [5]. All the device logs collected by Aegis+ app in different smart home platforms are forwarded to cloud integrated context generation module of Aegis+. We also used existing static analysis tools [12, 13, 16] to create app contexts from official apps and automation rules (485 SmartThings app, 20 Alexa skills, 10 LIFX rules) that are used to add app contexts in training dataset. These device logs are used to create the context-aware model and detect malicious activities in SHS.

For collecting the malicious dataset, we considered five different threat models (Section 3). We built five malicious apps to represent these threats. Our malicious apps cover several known threats presented by other researchers in [32, 76]. Our handcrafted malicious apps also included smart home attacks using smart speakers such as Amazon Alexa, Google Home, and so on [15, 77]. To perform the attack described in Threat 1, we built a battery monitor App for smart locks that leaks the unlock code via SMS to the attacker. We realized the impersonation attack by unlocking the smart lock as an outsider using the leaked unlock code. For Threat 2, we built an app that injects false smoke sensor data to trigger the fire alarm in the SHS. For Threat 3, we created an app that flickered a smart light in a specific pattern while nobody was in the home. To perform the denial-of-service attack described in Threat 4, we developed an app that stopped the smart thermostat for a pre-defined value. For Threat 5, we created an app that could generate morse code using a smart light while no person was in the room and triggered a smart camera to take stealthy pictures. Our malicious apps cover several existing attacks on smart home devices presented by the researchers [32, 76]. In Table 2, we mapped our threat models with existing malicious apps presented by the researchers. Additionally, we added some malfunctioning devices (e.g., a smart lock without power, fused smart light) in the SHS to test Aegis+ in cases that include device malfunction. We collected 24 different datasets (4 datasets for each attack scenario) for a total of over 15,000 events. We used 75% of the benign user data to train the Markov Chain model. Then the remaining 25% of data along with the malicious dataset was used in the testing phase, which is a common practice [9, 26, 60].

6 PERFORMANCE EVALUATION

In this section, we evaluate the effectiveness and feasibility of Aegis+ in detecting malicious activities in an SHS with real user data. We train the anomaly detector module of Aegis+ with data collected from multiple smart home users for benign daily activities. For testing purposes, we use the user data as well as the malicious data collected from the adversary model described in Section 3.

In the evaluation of Aegis+, we consider several research questions:

RQ1 What is the performance of Aegis+ in different smart home layouts and devices? (Section 6.2)
RQ2 What is the impact of different apps, policies, and configurations on the performance of Aegis+? (Section 6.3)
RQ3 What is the impact of different user behavior on the performance of Aegis+? (Section 6.4)
RQ4 What is the performance overhead introduced by Aegis+ in an SHS? (Section 6.5).

6.1 Performance Metrics

In the evaluation of Aegis+, we used six different performance metrics: True Positive rate (TP), False Negative rate (FN), True Negative rate (TN), False Positive rate (FP), Accuracy, and F-score. TP rate indicates the percentage of correctly identified benign activities, while TN rate refers to the percentage of correctly identified malicious activities. However, FP and FN illustrate the number of malicious activities identified as benign and the number of benign activities detected as malicious activities, respectively. F-score is an indicator of the accuracy of a framework, which considers TP and TN as computational vector. The performance metrics are defined by the

Table 2. Malicious Apps Mapping of Aegis+, ContextIoT [32], IoTBench [76], and Existing Voice Command Attacks

Aegis+Threat Model	App Description	ContextIoT [32]	IoTBench [76]	Voice command attacks
Threat-1: Impersonation attack	A battery monitoring app leaks unlock code via SMS and an attacker uses the code to unlock the smart lock by impersonating as valid user	• Backdoor pin code injection. • Lock access revocation. • LockManager. • App Update – PowersOutAlert. • Lock access revocation.	• Permissions-Implicit 2.	
	An app that captures and replays a voice command to impersonate valid users.	—	—	• Stealing voice attack [4]. • Voice spoofing attack [41].
Threat-2: False data injection	An app injecting false sensor data to a device.	• Fake alarm. • Remote control – FireAlarm. • Remote command – SmokeDetector.	—	—
	A voice command that maliciously or mistakenly injects false data to a connected smart device.	—	—	• Hidden voice commands [15]. • Dolphin attack [77].
Threat-3: Side channel attack	An app flickers a smart light in a specific pattern while no user is present.	• Leaking information. • Creating seizures using strobed light. • IPC – MaliciousCameraIPC & PresenceSensor. • MidnightCamera.	• Side Channel - Side Channel 1. • Side Channel - Side Channel 1.	
	A malicious signal (audio or light) causing a smart speaker to execute a legit yet vulnerable command.	—	—	• Light commands [70].
Threat-4: Denial-of-Service	A malicious app that cancels any ongoing tasks in smart devices at a pre-defined value.	• Disabling vacation mode. • Abusing permission.	—	
Threat-5: Triggering a malicious app	An app changing state of devices in a specific pattern to trigger a malicious app in a connected device.	• Surreptitious surveillance. • Undesired unlocking. • IPC – MaliciousCameraIPC & PresenceSensor.	—	

following equations:

$$TP\ rate = \frac{TP}{TP + FN}, \tag{7}$$

$$FN\ rate = \frac{FN}{TP + FN}, \tag{8}$$

$$TN\ rate = \frac{TN}{TN + FP}, \tag{9}$$

$$FP\ rate = \frac{FP}{TN + FP}, \tag{10}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{11}$$

Table 3. Performance Evaluation of Aegis+ in Different Smart Home Layouts

Smart Home Layout	SHS platforms	Normal Training						Adaptive Training					
		Recall	FN	TN	FP	Accuracy	F-score	Recall	FN	TN	FP	Accuracy	F-score
Single Bedroom	Single platform	0.95	0.05	1	0	0.9547	0.9604	0.97	0.03	1	0	0.9712	0.9847
Home	Multi-platform	0.94	0.06	0.99	0.01	0.965	0.9643	0.97	0.03	0.98	0.02	0.975	0.9749
Double Bedroom	Single platform	0.93	0.07	1	0	0.9340	0.9655	0.96	0.04	1	0	0.964	0.9795
Home	Multi-platform	0.92	0.08	0.97	0.03	0.945	0.9443	0.945	0.055	0.99	0.01	0.9675	0.9669
Duplex Home	Single platform	0.91	0.09	1	0	0.9119	0.9529	0.96	0.04	1	0	0.9614	0.9688
	Multi-platform	0.90	0.1	0.96	0.04	0.93	0.9290	0.93	0.07	0.98	0.02	0.955	0.9543

$$F - score = \frac{2 * TP * TN}{TP + TN}. \tag{12}$$

In addition to the aforementioned metrics, we considered several parameters, such as power usage, memory usage, CPU usage, latency, and so on, to measure performance overhead of Aegis+ in real-life SHS.

6.2 Evaluation with Different Home Layouts

To evaluate Aegis+ in different smart home layouts, we consider two important criteria (1) different smart home layouts, (2) multiple numbers of users. An SHS can have different smart home layouts and different numbers of devices. We tested the efficiency of Aegis+ in a multi-user environment and different smart home layouts.

Different smart home layouts: User activities in a smart home can vary depending on the home layout, as different layouts can lead to different usage patterns. To evaluate Aegis+, we considered three different layouts: single bedroom home, double bedroom home, and duplex home. Here, we considered a single authorized smart home user in different layouts. We collected data from 15 different users in these layouts. Our dataset also includes single platform and multi-platform SHSs. Table 3 presents the evaluation results associated with different smart home layouts. We can observe that accuracy and F-score for different layouts in single platform SHS vary from 96%–91% and 97%–95%, respectively. For multi-platform SHS, we can observe that the accuracy and F-score vary from 96%–93% and 96%–92% in different smart home layouts, respectively. Aegis+ also achieves high TP (96%–91%) and TN rate (100%–96%) irrespective of layouts and number of platforms (single and multi-platform). One can safely confirm that variation in different layouts has a minimal impact on the performance of Aegis+. Table 3 also shows how the performance of Aegis+ improves in adaptive training mode. Here, whenever the controller device (e.g., smartphone, tablet) is connected in the smart home network, we infer the user is in home location and using adaptive training mode. One can notice that the accuracy of Aegis+ increases from 95% to 97% in adaptive training mode for single bedroom layout for both single and multi-platform SHSs. For double bedroom and duplex home, Aegis+ achieves 96.4% and 96.1% accuracy in single platform and 96.6% and 95.4% in multi-platform SHS, respectively. As adaptive training mode uses user validation to reduce FP and FN events, F-score increases to approximately 95%–97% for all three layouts. In summary, Aegis+ can achieve accuracy and F-score over 95% for all three smart home layouts.

Different number of authorized users: Smart home platforms allow users to add more than one authorized user for the same SHS. Hence, an SHS can have multi-user scenarios with different user activities happening at the same time. To evaluate this setting of the smart home in Aegis+, we collected data from several multi-user settings with different users performing their daily activities at once. We used different smart home layouts with several multi-user scenarios (two, three, and four authorized controllers/conflicting users) in our data collection process. Additionally, we performed the aforementioned attack scenarios to collect malicious dataset and tested the efficiency of Aegis+ in different multi-user environments. Table 4 illustrates the detailed evaluation of Aegis+

Table 4. Performance Evaluation of AEGIS+ in Different Multi-user Scenarios

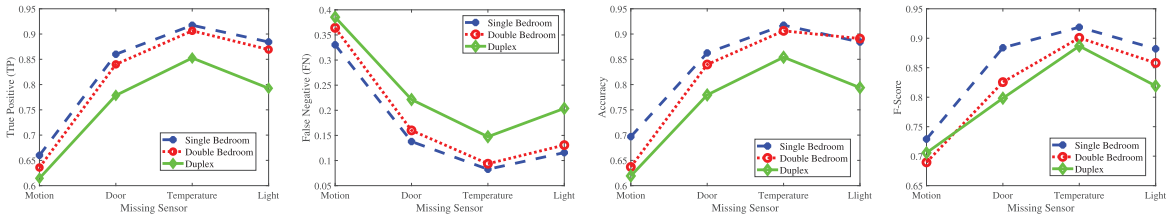
Smart Home Layout	No of Controllers	Normal Training						Adaptive Training					
		Recall	FN	Precision	FP	Accuracy	F-score	Recall	FN	Precision	FP	Accuracy	F-score
Single Bedroom Home	2	0.9472	0.0528	1	0	0.9477	0.9729	0.9685	0.0315	1	0	0.9711	0.9839
	3	0.9399	0.0601	1	0	0.9405	0.9690	0.9564	0.0436	1	0	0.96	0.9777
	4	0.9041	0.0959	0.96	0.04	0.9352	0.9312	0.9482	0.0588	1	0	0.9525	0.9734
Double Bedroom Home	2	0.9222	0.0778	1	0	0.9229	0.9595	0.9654	0.0346	1	0	0.9682	0.9823
	3	0.9058	0.0942	0.9529	0.0471	0.9062	0.9288	0.9523	0.0477	0.9785	0.0215	0.9545	0.9652
	4	0.8806	0.1194	0.8941	0.1059	0.8807	0.8873	0.9476	0.0524	0.96	0.04	0.9486	0.9537
Duplex Home	2	0.9017	0.0983	1	0	0.9038	0.9483	0.958	0.042	1	0	0.9615	0.9785
	3	0.8901	0.1099	0.9238	0.0762	0.8909	0.9067	0.9512	0.0488	0.975	0.025	0.9531	0.9629
	4	0.8694	0.1306	0.8857	0.1143	0.8698	0.8775	0.9388	0.0612	0.953	0.047	0.94	0.9458

in different smart home settings. For single bedroom layout, we can observe that accuracy and F-score reach the peak (0.9477 and 0.9729, respectively) for the two users setup. If we increase the number of authorized users in the SHS, the accuracy gradually decreases with an increasing FN rate. Similarly, for double bedrooms and duplex home layout, AEGIS+ achieves the highest accuracy and F-score for two authorized users' setup. Both accuracy and F-score decreases while the FN rate increases as the number of authorized users increases. The highest accuracy achieved in two bedrooms and duplex home layouts are 92.29% and 90.38%, respectively. As different users interact with smart home devices in varied ways, the FN rate increases with the number of users in the system. To minimize the number of FN events, we implement the adaptive training mode in AEGIS+. In a multi-user scenario, a notification is pushed in all the controller devices if AEGIS+ detects a malicious event in adaptive training mode. All the authorized users can confirm the event based on their activities and AEGIS+ trains the analytical model with validated data. One can notice from Table 4 that AEGIS+ achieves the highest accuracy and F-score (97% and 98%, respectively) for two users setup in single bedroom layout. Adaptive training mode also decreases FN rate approximately by 38.6% and increases the accuracy to 96% and 95.25% for three and four authorized user scenarios, respectively. For two bedroom and duplex home layout, adaptive training mode also increases the efficiency of AEGIS+. Adaptive training mode reduces FN and FP rate approximately by 60% while accuracy and F-score increases to approximately 96% and 98%, respectively, in a double bedroom and duplex home layout. In summary, AEGIS+ can minimize the effect of conflicting user activities in a multi-user scenario in adaptive training mode while increasing efficiency.

6.3 Evaluation with Different Smart Home Configurations

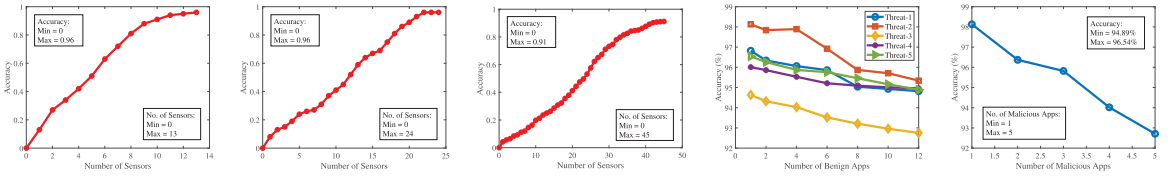
In this subsection, we evaluate AEGIS+ based on different smart home configurations including (1) different sensor configurations, (2) different user policies, (3) different controller device configurations, and (4) number of installed apps.

Different sensor configurations: To evaluate the efficiency of AEGIS+ based on deployed sensors, we use several combinations of sensors to build the context-aware model of user activities and report accuracy in Figure 9. Since AEGIS+ considers different smart home sensors and devices as co-dependent components, we try to understand to what extent changing the combinations of sensors in an SHS affects AEGIS+'s performance. For this, we tested the efficacy of AEGIS+ with four different combinations of sensors: without the motion sensor, without the door sensor, without the temperature sensor, and without the light sensor. As seen in Figures 9(c) and 9(d), decreasing the number of sensors from the context-aware model in AEGIS+ declines the accuracy and F-score of the framework. Removing the motion sensor resulted in the lowest accuracy and F-score (61% and 68% in duplex



(a) True Positive (TP) (b) False Negative (FN) (c) Accuracy (d) F-score

Fig. 9. Performance evaluation of Aegis+ with different sensors.

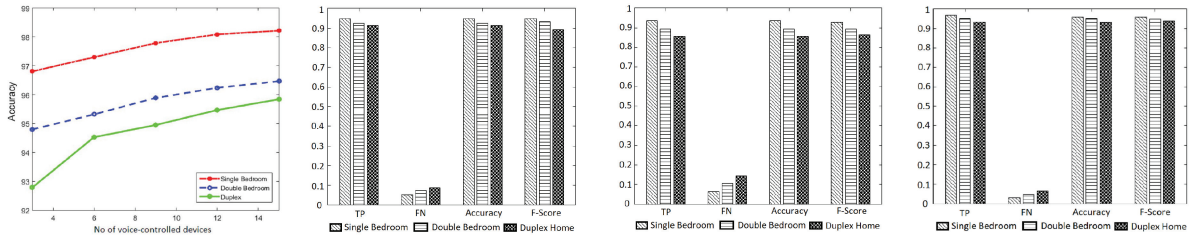


(a) Single Bedroom (b) Double Bedroom (c) Duplex Home (d) Accuracy vs. benign (e) Accuracy vs. malicious

Fig. 10. Accuracy of Aegis+ with different number of sensors (a), (b), (c) and with different number of benign and malicious apps (d), (e).

home layout, respectively). As motion sensors are configured with the majority of the devices (smart light, smart lock, etc.) and used in most of the user activity context, it affects the performance of Aegis+ significantly. We can also observe that removing sensors from the SHS introduces a high FN rate, as our proposed framework cannot build the context of the user activities correctly (Figure 9(b)). Again, Figure 9(c) illustrates that removing the temperature sensor from the SHS does not influence the performance significantly (85%–91% accuracy and 88%–91% F-score in different layouts). The main reason is that the temperature sensor can be configured with a limited number of devices; hence, it is affected by user activities less than other sensors. Without the door sensor and light sensor, Aegis+ can achieve moderate accuracy ranges from 77%–86% and 79%–88%, respectively. Figure 10 illustrates the change in accuracy of Aegis+ for changing the number of sensors in different smart home layouts. For all three smart home layouts (single bedroom, double bedroom, and duplex home), limiting the number of sensors in the system decreases the accuracy of Aegis+. In conclusion, limiting the number of sensors in an SHS can reduce the efficiency of Aegis+ by introducing FN cases in the system.

Evaluation Based on Installed Apps: Smart home users can install multiple smart apps to configure and control the same devices or different devices at the same time. For example, users can install two different apps to control a smart light at a time with motion and door sensors, respectively. To test the effectiveness of Aegis+ based on the installed apps, we installed 12 benign apps in total in the system to build the context-aware model of user activities. Figure 10(d) shows the accuracy of Aegis+ in detecting malicious apps in an SHS based on installed apps. Here, we installed different malicious apps (Section 3) in the system with multiple benign apps to determine the effectiveness of Aegis+. From Figure 10(d), one can notice that Aegis+ achieves the highest accuracy of 98.15% for Threat-2 and the lowest accuracy of 94.34% for Threat-3 for only one benign smart app installed in the system. With the increment of benign apps in the SHS (highest 12 benign apps), accuracy ranges between 98% to 95% and 94% to 92.5% for Threat-2 and Threat-3, respectively. The accuracy of Aegis+ in detecting Threat-1, Threat-2, and Threat-5 varies between 96% to 93%. We also tested different malicious apps installed at once in the SHS with a fixed number of benign apps (12 benign apps) to understand the effectiveness of Aegis+ completely. Figure 10(e) depicts the accuracy of Aegis+ based on the number of malicious apps installed in the system. One can notice that Aegis+ can achieve an accuracy of 98% for one malicious app installed in the



(a) Voice-controlled SHS (b) User Policy 1 (c) User Policy 2 (d) User Policy 3

Fig. 11. Performance evaluation of AEGIS+ in a voice-controlled SHS (a) and performance of AEGIS+ in a policy-enforced SHS (c), (d), (e).

SHS, which decreases very little with a higher number of malicious apps (92.57% with five malicious apps). In conclusion, the performance of AEGIS+ changed very little with the change in the number of benign apps and malicious apps installed in the SHS.

Evaluation Based on Controller Device Configurations: Modern SHS allows users to configure smart home devices with different controller devices. Smart home users control devices usually via vendor-specific controller apps installed in smartphones and tablets. Additionally, smart home devices can also be controlled via voice commands if configured with a smart speaker (e.g., Amazon Alexa, Google Home). As explained in Section 3 and 5, attackers can manipulate the voice-controlled configuration of smart home devices by injecting false commands and impersonation (details in Table 2). In this subsection, we test the efficacy of AEGIS+ in detecting malicious activities targeting voice-controlled smart devices. From Figure 11(a), one can notice that the accuracy of AEGIS+ increases with the number of voice-configured smart home devices, as the increment in the number of voice-controlled devices allows AEGIS+ to understand the user activity contexts properly. For single bedroom layout, the accuracy of AEGIS+ varies from 98%–96% in detecting malicious activities targeting voice-controlled devices. AEGIS+ also achieves high accuracy (96%–94% and 95%–92%, respectively) in double bedroom and duplex home layout. In summary, AEGIS+ achieves high accuracy in detecting malicious activities targeting voice-controlled smart home configurations regardless of home layouts.

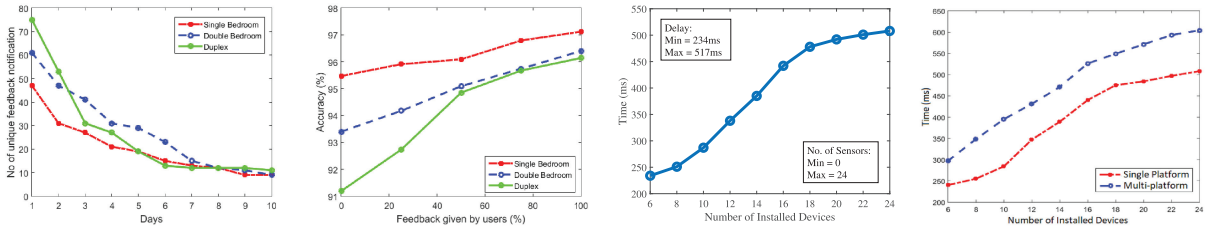
Evaluation Based on User Policies: In modern SHS, users can define customized policies to control smart home devices. For example, users can impose a time window to activate a smart light in an SHS. In this subsection, we test the efficiency of AEGIS+ with different policies enforced in SHS. We consider the following user policies to evaluate AEGIS+:

User Policy 1: Users can apply time-specific operations for different smart home entities. In policy 1, users configure a smart light with the motion sensor that will be enforced only from sunset to sunrise.

User Policy 2: Users can apply sensor-specific operations for different smart home devices. In policy 2, users configure smart lights with light, motion, and door sensors.

User Policy 3: Users can configure smart speakers to control smart home devices using voice commands. In policy 3, users configure smart lights with smart speakers to control via voice commands (e.g., bedroom light on).

Figures 11(b), 11(c), and 11(d) present the performance of AEGIS+ in these policies enforced in SHSs. One can observe that AEGIS+ achieved accuracy as high as 95% while enforcing time-specific operations in SHS (Figure 11(b)). The F-score also ranges from 89% to 94% for different smart home layouts with time-specific operations with low FN rate (5%–8%). For User Policy 2, one can observe a slight fall in the accuracy and F-score as changing sensor-device configuration introduces FN cases in the system. From Figure 11(b), we can see that AEGIS+ can perform with an accuracy ranging from 85% to 93% for different smart home layouts while changing the sensor-device configurations. AEGIS+ also achieves F-score ranging from 86.5%–92% for different configura-



(a) User notification vs.time (b) Accuracy vs. Feedback (c) Training time vs. Device (d) Training time vs. Platforms

Fig. 12. Performance of Aegis+ in terms of user feedback (a), (b) and performance overhead of Aegis+ (c), (d).

tions. For controller-specific operation (voice-controlled operation), the accuracy of detecting malicious events ranges from 96%–93% with low FN rate (3%–6%) in different smart home layouts. Also, Aegis+ achieves high F-scores (96%–94%) in detecting malicious activities in the voice-controlled device configuration. In summary, Aegis+ can detect malicious activities in policy-enforced SHS with high accuracy and F-score.

6.4 Evaluation with Different User Behavior

In this subsection, we test Aegis+ in terms of user interactions and behavior in the SHS. Aegis+ uses an adaptive training method that requires users' feedback to detect FP and FN cases. This adaptive training method may cause user fatigue with excessive feedback notifications [2]. To determine how the user fatigue may affect the performance of Aegis+, we performed accuracy vs. user feedback study, which is shown in Figures 12(a) and 12(b). Figure 12(a) shows the number of notifications generated in adaptive training mode by Aegis+ in different smart home settings over a 10-day period. One can notice that in all three layouts the number of generated notification decreases significantly. For the single home layout, the number of notifications decreases by 59% in 5 days. For double bedroom and duplex layout, the number of notifications also decreases by 52.45% and 74.67%, respectively. This indicates that users only have to deal with higher feedback requests for a short period of time. Note that Aegis+ pushes a notification for both FP and TN events, as our test dataset includes both normal user activity and malicious events. Hence, the number of notifications generated for only FP events is lower than it seems in Figure 12(a). For example, in day 10, the total number of notifications is 10 among which 6 notifications are from FP. Figure 12(b) shows how user feedback affects the accuracy of Aegis+ in detecting different threats. One can notice that the accuracy of Aegis+ increases very little from 50% to 100% user feedback. This indicates that if the users actively train Aegis+ in the initial period (1-5 days) in adaptive training mode, then the performance improves significantly. Again, Aegis+ always provides the option to choose a specific time for adaptive training mode to the users. In conclusion, Aegis+ can negate the consequence of user fatigue by terminating adaptive training mode after an initial period, which are configurable by the users.

6.5 Performance Overhead

We illustrate the performance overhead of Aegis+, including resource overhead and latency. We identify three major features that could introduce a time delay in real-time operation.

Delay in adaptive training model: Aegis+ offers adaptive training mode where any malicious event detected by Aegis+ is forwarded to the user for validation. Aegis+ uses this validated data to retrain the analytical model, which introduces a time delay in the operation. In Figure 12(c), we illustrate the time needed for retraining the framework with respect to the number of devices installed in the device. One can notice that Aegis+ takes approximately 230 ms to train when the system has 6 different installed in the SHS. The training time increases to 519 ms for 24 installed devices in the SHS. In short, Aegis+ introduces negligible overhead in terms of time delay in adaptive training mode.

Delay in multi-platform SHS: AEGIS+ allows multi-platform smart home configuration where smart home devices form different platforms can share the same physical home environment. AEGIS+ uses multiple customized apps to collect device's and sensor's state information to build a context-aware model from user activities. As smart home devices from different platforms vary in resource (memory, CPU, command execution frequency, etc.), collecting data and building the context-aware model from multi-platform SHS introduces a time delay in the operation of AEGIS+. Figure 12(d) shows a comparison between time delay introduced in single and multi-platform (4 different smart home platforms) SHS with respect to the number of installed devices. One can notice that for a minimum of 6 devices AEGIS+ takes approximately 220 ms and 300 ms in single and multi-platform SHS, respectively. The highest time delay introduced in multi-platform SHS is approximately 600 ms for 24 installed device compared to 508 ms in single platform SHS. In summary, AEGIS+ introduces minimal time delay in multi-platform SHS compared to single platform SHS. Hence, AEGIS+ is effective in detecting malicious activities irrespective of the number of smart home devices and platforms installed in the smart environment.

Delay in action management module: Action management module of AEGIS+ alerts users in the event of malicious activity in SHS. The alert message is sent to the controller device (smartphone, tablet, etc.) in the form of notification, which introduces a time delay in the action management module. We use a *SmartThings* app to send notifications to controller devices of authorized users. This app communicates with the cloud server via *http* protocol, which is connected with the action management module (Section 4.4). On average, action management takes 210 ms time to send a notification to the controller device from the moment of malicious activity detection, which is low for real-world deployment. In short, we conclude that AEGIS+ meets the efficient demands in the action management module.

7 USER SCENARIOS AND BENEFITS

In this section, we illustrate how deploying AEGIS+ in a smart home can help different groups of consumers using several use scenarios and discuss different benefits of AEGIS+.

7.1 User Scenarios

We illustrate three different user scenarios to understand the benefits of AEGIS+ among vendors, end-users, and developers.

Vendors- Smart home vendors can use AEGIS+ to detect abnormal behavior in a customer's home. Here, a customer, *Alice*, installs several smart security devices (smart lock, smart camera, smart fire alarm, etc.) and the corresponding smart apps to control them. However, one of the installed apps has malicious code that injects false data when no one is at home to trigger the fire alarm (Table 2- Threat- 2). As Alice does not have any idea of this malicious event, she calls the security service provider/vendor for support. In this situation, the security service provider can identify that the alarm is generated from a false data from the state model generated by AEGIS+ and support the customer with appropriate suggestions such as deleting the malicious app, reinstalling the correct app, and so on.

End-users- End-users constitute the most common victims of malicious events in a smart home. Attackers can perform several malicious activities including gaining physical access to the home. For instance, a smart home user, *Bob*, installs a new smart lock and the corresponding app in the SHS. However, the installed app has a malicious snippet to forward the unlock code to the attacker so he can unlock the smart lock by impersonating Bob (Table 2- Threat 1). AEGIS+ can identify this event and notify the user in real-time. Moreover, Bob can change the state of the lock to unlock and prevent any physical access to the smart home. Smart home users also tend to install devices from different vendors in the same physical home environment. AEGIS+ provides a platform-independent security framework to the end-users that can detect malicious activities irrespective of the platform of the devices. For instance, a malicious app can inject a voice command to the smart speaker to open the front

door (Table 2- Threat 2). The contextual model of Aegis+ can confirm the presence of the user by checking the state of the installed presence sensor even if it is not connected to the smart speaker. Hence, Aegis+ can detect malicious activities in smart home devices using the context-aware model in a multi-platform SHS.

Developers- Developers or tech-enthusiastic users can deploy Aegis+ in their SHS and specify different rules to enhance the security of their homes. For instance, *Kyle*, a smart home user, installs multiple smart lights and motion sensors in his SHS. Kyle also builds a new smart app to control the lights with motion. By using the logic extractor of Aegis+, Kyle can understand whether his app logic is correct or not. Moreover, Kyle can use the adaptive training mode to see how the overall state of the SHS changes with new devices and apps. If the action of the new smart lights contradicts the existing system, or any malicious event occurs (e.g., Table 2- Threat 3), Kyle can understand the cause of the event and take necessary steps. Moreover, Kyle can understand the working conditions of smart home devices and improve his technological knowledge using Aegis+.

7.2 Discussion

Deployability in Real-life System- One of the prime features of Aegis+ is easy deployability in real-life systems. Aegis+ uses simple smart apps to collect device states from multiple smart home platforms in an SHS and build the context-aware model. The detection mechanism runs in the cloud, which does not hamper the normal operation of the SHS. Users can install Aegis+ similarly to any other smart app.

Applicability and Real-life Threats- Security risks may arise from smart home apps performing side-channel attacks. For instance, a smart app can flash the light in a specific pattern to leak information or trigger another connected device, which can be considered as a threat. While most of the existing solutions consider this threat as out of scope [16, 39], Aegis+ successfully detects such malicious behaviors. In addition, Aegis+ can detect device malfunctions inside an SHS. For instance, if a smart light is configured with the motion sensor, one should expect that the light turns on due to the active motion. Other outcomes from this specific context may be categorized by Aegis+ as a malfunction. Hence, Aegis+ can automatically detect any false data injection or adversarial attacks in SHSs [43]

Multi-user activity in SHS- In SHS, more than one user may perform different activities simultaneously [63]. As Aegis+ utilizes user activity contexts to detect malicious actions, correctly distinguishing between different user activities is key. Instead of single-context analysis, Aegis+ uses a pattern of contexts to understand the user activities. Hence, Aegis+ can detect simultaneous activities performed by different users and devices in an SHS. For instance, if two users are walking towards the same point from opposite directions, Aegis+ observes the related contexts to identify two different motion activities.

Trigger-action effect time- Smart home devices use sensors to automate tasks. For instance, a smart light can be triggered by a motion sensor or a door sensor. Each trigger-action scenario has an effect time (time duration of a device being active). This effect time has to be correctly considered to build the context of the user activity. Aegis+ mitigates this time dependency by considering the pattern of device utilization. For instance, the user sets a smart light to remain on for two minutes if a motion is detected. This case is detected by Aegis+ by checking consecutive states of the overall smart home and is used to detect malicious apps or malfunctioning devices (if the motion is sensed by the sensor and it holds the state for 20s, then the smart light should be also on for at least 20s, otherwise it is broken or malicious). Aegis+ uses these trigger-action scenarios to mitigate the effect of the time interval and builds the contextual model from device state patterns.

Detecting rare events- In a smart home, different autonomous events occur based on device configuration and user activities. These events may include rare events such as triggering fire alarms. As Aegis+ uses daily user activities to train its analytical model, these rare events might be unaddressed and flagged as threat. To solve this, we use the app context to verify unrecognized events in Aegis+. Any alert triggered in Aegis+ is verified with the app context generated from the installed app (Section 4.2). If the app context is matched with the rare event,

AEGIS+ considers the event as benign and retrains the model automatically. Users can also check and verify rare natural events through action management module (Section 4.4).

Multi-platform support- AEGIS+ supports multi-platform smart home environment, which allows users to install smart devices from different platforms in the same home environment. For instance, users can install both Samsung SmartThings devices and Wi-Fi devices such as LIFX smart bulbs in the same home environment. While existing solutions cannot ensure security in this multi-platform environment, AEGIS+ observes devices state changes and builds a contextual model that can envision user activity contexts and multi-platform correlation properly in an SHS. Also, AEGIS+ only considers device states to build the contextual model, which does not need any platform-specific and app-specific modification. Hence, AEGIS+ can detect malicious activities in a multi-platform SHS.

Customized notification system- AEGIS+ offers user-customized notification system to alert the users regarding malicious events in the smart environment. For instance, users can choose device-specific notification to check all the malicious events reported by AEGIS+ for a specific device. Also, users can choose layout-specific notification to understand the nature of a reported event in user-customized SHS. Additionally, AEGIS+ allows users to set customized notifications for specific devices at installation time. This feature of AEGIS+ allows users with different technical knowledge (beginner or expert smart home users) to understand the nature of the malicious events and act accordingly.

8 RELATED WORK

Smart Home Systems (SHSs) have become very popular with their user-centric customization options and third-party app development. Developers have offered different apps to increase the functionalities of smart home devices. Nonetheless, the nature of the app-based models introduces several malicious threats to SHSs.

Security Vulnerabilities. In recent years, several works have outlined security threats to SHSs [21, 47, 50, 54, 64]. These threats mainly focus on three SHSs components: communication protocols, devices, and apps. As the concept of the smart home is still evolving, there are several implementation flaws in the communication protocols for SHSs. Attackers may exploit these flaws to leak sensitive user information from smart home devices. Several prior works have reported multiple implementation flaws of SHS's communication protocol that can be abused to leak sensitive user information from smart home devices [25, 38, 44, 56]. For instance, an attacker can capture the network packets covertly using simple sniffing devices and extract shared information, even from the encrypted traffic [1, 14, 29]. Fernandes et al. reported several design flaws in popular smart home platforms (e.g., Samsung SmartThings) that includes system and app-level vulnerabilities [23]. Chi et al. showed that it is possible to exploit the smart home platform by triggering malicious activities from legitimate user action [19]. As smart home devices use multiple apps to automate different tasks, researchers showed that it is possible to execute a malicious task while performing legitimate user actions in connected devices [46, 49]. Moreover, current smart home platforms use smartphones as user interface and control devices, which can also be used to launch attacks to smart home devices [23, 65]. Jia et al. reported the existence of several malicious apps that can be migrated from smartphone and IoT platforms to SHSs [32]. Recently, a group of researchers published an online repository, *IoTBench* [16, 31], which revealed several malicious apps for existing smart home platforms, including Samsung SmartThings (19 malicious apps) and OpenHab (33 third-party rules).

Existing Security Solutions. While researchers and developers reported these various threats to SHSs in recent years, there is no comprehensive security solution that addresses these threats and secures the system. Developers have introduced several policy-based security measures to limit unauthorized access to SHSs, which depends completely on user decisions [66, 71]. However, researchers proposed several countermeasures to bolster the security of SHSs by implementing encrypted data traffic in smart home communication protocols (e.g., ZigBee, Z-Wave, Wi-Fi) [8, 22, 36, 51]. Additionally, researchers proposed several static and forensic analysis tools to

detect threats in application level of SHSs [13, 16]. In the following, we discuss existing security mechanisms proposed by the researchers and their shortcomings:

- *Permission-based approach.* Most of the SHS platforms use permission-based app management where users are asked to configure and allow permissions (device access, sensor access, etc.) at installation time. However, once the user approves the permissions, SHS does not provide any explicit information about how the app is using the granted accesses. This existing coarse-grained permission model could lead to unauthorized device access and sensitive information leakage in SHS [23]. Jia et al. introduced *ContextIoT*, a context-aware permission model to restrict unauthorized device access and detect malicious activities in SHSs [32]. Their proposed model creates a runtime context of each app event and asks for user permission before executing any unknown activity in the SHS.
- *Policy and configuration analysis.* Several policy-based security measures were proposed to limit unauthorized access to SHSs [18, 66, 71]. Similar to permission-based approaches, these solutions depend on user decisions. Mohsin et al. presented *IoTSAT*, a framework to analyze threats on SHSs using device configurations and enforced user policies [39]. IoTSAT creates a behavioral model based on device configurations and network policies and compares it with enforced policies to identify unusual activities in the SHSs [39]. Wang et al. introduced *iRuler*, an automation rule analysis framework to detect inter-rule vulnerabilities in smart home apps [72]. *iRuler* uses natural language processing techniques to detect trigger-action information flow and detect security risks in implemented applets in IFTTT platform. Similar to this work, Babun et al. proposed *IoTwatch*, a runtime analysis tool to identify privacy violations in smart home apps [12]. *IoTwatch* collects the privacy settings of the users at installation time and collects runtime privacy-sensitive data sharing in smart home devices and apps to detect privacy violations using natural language processing.
- *Network analysis.* Researchers have proposed several network analysis techniques to detect malicious activities in SHSs. Yamauchi et al. presented a network-based intrusion detection system (IDS) that uses benign network packets generated from user activities to detect malicious events in SHSs [74]. Researchers implemented a network observer in the home gateway that learns user behaviors from the network traffic and detects malicious user commands from learned behavior. Anthi et al. proposed a supervised intrusion detection system to detect malicious and known network attacks in smart home devices [6]. In a recent work, Apthorpe et al. showed that it is possible to mitigate privacy leakage from smart home traffic by implementing traffic shaping [7].
- *Static analysis.* Recently, static analysis of smart home apps have been proposed to detect information leakage and cross-app interference. Berkay and Babun et al. introduced a static analysis tool, *SaINT*, to track sensitive information in smart home apps [16]. SaINT performs source code analysis to trace all the sensitive information from sources to sinks and identify potential information leakage. Chi et al. proposed a static analysis tool to extract app context from smart home apps to detect cross-app interference [19]. Their proposed model considers data sources and sinks in an app to track the information flow and extracts the rules presented in an app. The extracted rules are saved and compared when a new app is installed to find possible conflicts of operations.
- *Forensic analysis.* Forensic analysis of smart home data has been proposed to identify malicious events in an SHS. Wang et al. proposed a security tool, *ProvThings*, which logs runtime data from smart home apps and performs provenance tracking to detect malicious activities [73]. ProvThings implements data collector in the source code of an app to identify data flows and create provenance graphs. These graphs are used to infer app functionalities and detect malicious activities in the SHS. Babun et al. proposed *IoTDots*, a forensic analysis tool that can detect user behavior from logged data in an SHS [13]. IoTDots collects data from smart home apps and performs a context analysis to detect violation of security policies in a smart environment.

Table 5. Comparison between AEGIS+ and Other Existing Security Framework for SHS

Prior Work	Context Aware	Platform independent	Diverse generalized threats	No source code modification	User permission independent	Active notification	Run-time detection	Real-life implementation
ContextIoT [32]	●	○	●	○	○	●	●	●
IoTSAT [39]	○	●	○	●	●	○	○	○
iRuler [72]	○	○	○	○	●	●	●	●
IoTWatch [12]	○	○	●	○	●	●	●	●
Network IDS [74]	○	●	○	●	○	○	●	○
Supervised IDS [6]	○	●	○	●	●	○	●	○
SaINT [16]	○	●	●	●	●	○	○	●
HomeGuard [19]	○	●	○	○	●	○	●	●
ProveThings [73]	○	○	●	○	●	○	○	●
IoTDots [16]	○	○	●	○	●	○	○	●
AEGIS+	●	●	●	●	●	●	●	●

Differences from the existing solutions. The main differences between AEGIS+ and other existing solutions (although they are useful) can be articulated as follows: (1) While other solutions focus on securing shared data and improving the current user permission system [32], AEGIS+ detects malicious behaviors by considering user and device activity contexts in an SHS. (2) AEGIS+ considers both smart home configurations and installed apps to build a context-aware model and detect threats at runtime that outdo user-dependent solutions [32]. (3) Additionally, no source code modification [13] is needed for AEGIS+ to collect data from smart home devices and detect malicious activities in an SHS. (4) Unlike threat-specific existing solutions [16, 73], AEGIS+ can detect five different types of threats in an SHS, which makes it a more robust solution. (5) AEGIS+ provides a platform-independent solution, as it only considers user activity to build the context-aware model irrespective of smart home systems, specific devices, and development platforms. (6) Finally, AEGIS+ collects data from a common access point and performs behavior analysis at runtime, which reduces cost in terms of processing and overhead from other prior works [13, 16]. In addition, AEGIS+ does not store usage data from smart home devices, which reduces the privacy risks and concerns from prior solutions [13]. Table 5 summarizes the differences of AEGIS+ and other existing solutions.

In summary, AEGIS+ offers a context-aware security framework that uses behavior analysis, usage patterns, and app context to detect malicious activities at runtime and ensures security against five different threats to SHS with high accuracy and minimal overhead.

9 CONCLUSION

Modern app-based smart home systems (SHSs) expose the smart home ecosystem to novel threats. Attackers can easily manipulate the SHSs to perform different attacks or deceive users into installing malicious apps. However, current detection solutions only consider threats to devices and apps in the SHS, skipping the very complex and rich context-based relationships among smart home platforms, devices, and apps. In this article, we presented AEGIS+, a novel context-aware platform-independent security framework for smart homes that detects malicious activities by (1) observing the change in device behavior based on user activities in multi-platform SHS, (2) correlating sensor-device trigger-action scenarios with user activities in multi-platform environment, and (2) building a contextual model to differentiate benign and malicious behavior. We evaluated AEGIS+ in multiple smart home settings, with real-life users, with real SHS devices (i.e., Samsung SmartThings, LIFX smart bulbs, Amazon Alexa, Google Home platform) and with different day-to-day activities. Our detailed evaluation

shows that Aegis+ can achieve over 95% of accuracy and F-score in different smart home settings. We also tested Aegis+ against several malicious behaviors. Aegis+ is highly effective in detecting threats to smart home systems regardless of the smart home layouts, smart home platforms, the number of users, and enforced user policies. Finally, Aegis+ can detect different malicious behavior and threats in SHS with minimum overhead.

APPENDIX

A SAMPLE SMART APP FOR APP CONTEXT

One of the features of Aegis+ is using app context to verify the device states in the SHS. To build the app context, we used similar static analysis approaches used in prior works [16, 19]. We performed a source-to-sink taint analysis similar to Reference [16] to extract the app context. Additionally, we consider the sources for smart apps proposed in References [19, 32]. We then built the abstract syntax tree (AST) and modeled a trigger-action scenario of an app. We tracked the *Subscribe* method to represent the trigger and followed the conditional statement (e.g., if and switch) to reach the sink. This flow from entry point (source) to a sink is used to construct the condition of an app, which is then represented into app context. We collected 150 official *Samsung SmartThings* apps (available in their website) and created the app context database using this method.

```
/* This is a sample smart light app for Samsung SmartThings */
definition(
    name: "Smart Light App",
    namespace: "smartthings",
    author: "anonymous",
    description: "Turn lights on when door is open.",
    category: "Convenience",
)
preferences {
        section("When the door opens/closes...") {
                input "contact1", "capability.contactSensor", title: "Where?"
        }
        section("Turn on/off a light...") {
                input "light1", "capability.light"
        }
}
def installed() {
        subscribe(contact1, "contact", contactHandler)
}
def updated() {
        unsubscribe()
        subscribe(contact1, "contact", contactHandler)
}
def contactHandler(event) {
        if (event.value == "open") {
                light1.on()
        } else if (event.value == "closed") {
                light1.off()
        }
```

Listing 3. A code snippet of a sample smart app.

Listing 4. trigger-action Scenario of a sample app

```
Trigger: Contact1
Action: Switch1
Logic 1: contact1 = on, light1 = on
Logic 2: contact1 = off, light1 = off
```

Listing 4. trigger-action Scenario of a sample app.

```
App Context 1: contact1 = 1 , Light1 = 1
App context 2: contact1 = 0, Light1 = 0
```

Listing 5. Generated app context of a sample app.

REFERENCES

[1] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. 2020. Peek-a-Boo: I see your smart home activities, even encrypted! In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 207–218.

[2] Devdatta Akhawe and Adrienne Porter Felt. 2013. Alice in Warningland: A large-scale field study of browser security warning effectiveness. In *Proceedings of the 22nd USENIX Security Symposium*. 257–272.
[3] H. Aksu, L. Babun, M. Conti, G. Tolomei, and A. S. Uluagac. 2018. Advertising in the IoT era: Vision and challenges. *IEEE Commun. Mag.* 56, 11 (2018), 138–144. DOI: https://doi.org/10.1109/MCOM.2017.1700871
[4] Federico Alegre, Ravichander Vipperla, Nicholas Evans, and Benoît Fauve. 2012. On the vulnerability of automatic speaker recognition to spoofing attacks with artificial signals. In *Proceedings of the 20th European Signal Processing Conference (EUSIPCO'12)*. IEEE, 36–40.
[5] Amazon Alexa. 2018. Understand the Smart Home Skill API. Retrieved from https://developer.amazon.com/en-US/docs/alexa/smarthome/understand-the-smart-home-skill-api.html.
[6] Eirini Anthi, Lowri Williams, Małgorzata Słowińska, George Theodorakopoulos, and Pete Burnap. 2019. A supervised intrusion detection system for smart home IoT devices. *IEEE Internet Things J.* 6, 5 (2019), 9042–9053.
[7] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2019. Keeping the smart home private with smart (er) IoT traffic shaping. *Proc. Priv. Enhanc. Technol.* 2019, 3 (2019), 128–148.
[8] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. *arXiv preprint arXiv:1708.05044* (2017).
[9] Amazon AWS. 2019. Splitting the Data into Training and Evaluation Data. Retrieved from https://docs.aws.amazon.com/machine-learning/latest/dg/splitting-the-data-into-training-and-evaluation-data.html.
[10] Leonardo Babun, Hidayet Aksu, Lucas Ryan, Kemal Akkaya, Elizabeth S. Bentley, and A. Selcuk Uluagac. 2020. Z-IoT: Passive device-class fingerprinting of ZigBee and Z-wave IoT devices. In *Proceedings of the IEEE International Conference on Communications (ICC'20)*. IEEE, 1–7.
[11] Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2019. A system-level behavioral detection framework for compromised CPS devices: Smart-grid case. *ACM Trans. Cyber-phys. Syst.* 4, 2 (Nov. 2019).
[12] Leonardo Babun, Z. Berkay Celik, Patrick McDaniel, and A. Selcuk Uluagac. 2019. Real-time analysis of privacy-(un) aware IoT applications. *arXiv preprint arXiv:1911.10461* (2019).
[13] Leonardo Babun, Amit Kumar Sikder, Abbas Acar, and A. Selcuk Uluagac. 2018. IoTDots: A digital forensics framework for smart environments. *arXiv preprint arXiv:1809.00745* (2018).
[14] Joseph Bugeja, Andreas Jacobsson, and Paul Davidsson. 2016. On privacy and security challenges in smart connected homes. In *Proceedings of the European Intelligence and Security Informatics Conference (EISIC'16)*. IEEE, 172–175.
[15] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *Proceedings of the 25th USENIX Security Symposium*. 513–530.
[16] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. 2018. Sensitive information tracking in commodity IoT. In *Proceedings of the 27th USENIX Security Symposium*. 1687–1704.
[17] Z. B. Celik, P. McDaniel, G. Tan, L. Babun, and A. S. Uluagac. 2019. Verifying Internet of Things safety and security in physical spaces. *IEEE Secur. Priv.* 17, 5 (2019), 30–37.
[18] Antorweep Chakravorty, Tomasz Wlodarczyk, and Chunming Rong. 2013. Privacy preserving data analytics for smart homes. In *Proceedings of the IEEE Security and Privacy Workshops (SPW'13)*. IEEE, 23–27.
[19] Haotian Chi, Qiang Zeng, Xiaojiang Du, and Jiaping Yu. 2018. Cross-app threats in smart homes: Categorization, detection and handling. *arXiv preprint arXiv:1808.02125* (2018).
[20] OpenHAB Community. 2017. Openhab documentation. Retrieved from http://docs.openhab.org/index.html.
[21] Tamara Denning, Tadayoshi Kohno, and Henry M. Levy. 2013. Computer security and the modern home. *Commun. ACM* (Jan. 2013), 94–103.
[22] Ali Dorri, Salil S. Kanhere, Raja Jurdak, and Praveen Gauravaram. 2017. Blockchain for IoT security and privacy: The case study of a smart home. In *IEEE International Conference on Pervasive Computing and communications Workshops (PerCom workshops)*. IEEE, 618–623.
[23] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'16)*. IEEE, 636–654.
[24] Joint Center for Housing Studies of Harvard University. 2020. America's rental housing, 2020. Retrieved from https://www.jchs.harvard.edu/americas-rental-housing-2020.
[25] Behrang Fouladi and Sahand Ghanoun. 2013. Honey, I'm home!!, Hacking ZWave home automation systems. Black Hat USA.
[26] Google. 2019. Cloud AutoML. Retrieved from https://cloud.google.com/automl/.
[27] Groovy. 2020. Groovy Metaprogramming. Retrieved from http://docs.groovy-lang.org/docs/next/html/documentation/core-metaprogramming.html.
[28] Rachel Gunter. 2017. Making Sense of Samsung's SmartThings Initiative. Retrieved from https://marketrealist.com/2017/12/making-sense-samsungs-smartthings-initiative.
[29] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. 2016. Smart locks: Lessons for securing commodity internet of things devices. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ACM, 461–472.

[30] Philips Hue. 2018. How to develop for Hue: Hue API. Retrieved from https://developers.meethue.com/develop/hue-api/.
[31] IoTBench. 2017. Retrieved from https://github.com/IoTBench.
[32] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Z. Morley Mao, Atul Prakash, and Shanghai JiaoTong University. 2017. ContexIoT: Towards providing contextual integrity to appified IoT platforms. In *Proceedings of the Network and Distributed System Security Symposium*.
[33] Ah-Lian Kor, Colin Pattinson, Max Yanovsky, and Vyacheslav Kharchenko. 2018. IoT-enabled smart living. In *Technology for Smart Futures*. Springer, 3–28.
[34] Changmin Lee, Luca Zappaterra, Kwanghee Choi, and Hyeong-Ah Choi. 2014. Securing smart home: Technologies, security challenges, and security requirements. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS'14)*. IEEE, 67–72.
[35] LIFX. 2018. LIFX http API. Retrieved from https://api.developer.lifx.com/.
[36] Teddy Mantoro, Media A. Ayu, and Siti Munawwarah binti Mahmod. 2014. Securing the authentication and message integrity for Smart Home using smart phone. In *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS'14)*. IEEE, 985–989.
[37] Microsoft. 2020. Windows IoT core documentation. Retrieved from https://developer.microsoft.com/en-us/windows/IoT/explore/IoTcore.
[38] Byungho Min and Vijay Varadharajan. 2015. Design and evaluation of feature distributed malware attacks against the Internet of Things (IoT). In *Proceedings of the 20th International Conference on Engineering of Complex Computer Systems (ICECCS'15)*. IEEE, 80–89.
[39] Mujahid Mohsin, Zahid Anwar, Ghaith Husari, Ehab Al-Shaer, and Mohammad Ashiqur Rahman. 2016. IoTSAT: A formal framework for security analysis of the internet of things (IoT). In *Proceedings of the IEEE Conference on Communications and Network Security (CNS'16)*. IEEE, 180–188.
[40] Mozilla-IoT. 2020. Mozilla WebThings Documentation. Retrieved from https://iot.mozilla.org/docs/gateway-user-guide.html.
[41] Dibya Mukhopadhyay, Maliheh Shirvanian, and Nitesh Saxena. 2015. All your voices are belong to us: Stealing voices to fool humans and machines. In *Proceedings of the European Symposium on Research in Computer Security*. Springer, 599–621.
[42] J. Myers, L. Babun, E. Yao, S. Helble, and P. Allen. 2019. MAD-IoT: Memory anomaly detection for the Internet of Things. In *Proceedings of the IEEE Globecom Workshops (GC Wkshps'19)*. 1–6.
[43] A. K. M. Newaz, Nur Imtiazul Haque, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A. Selcuk Uluagac. 2020. Adversarial attacks to machine learning-based smart healthcare systems. *arXiv preprint arXiv:2010.03671* (2020).
[44] A. K. M. Iqtidar Newaz, Amit Kumar Sikder, Leonardo Babun, and A. Selcuk Uluagac. 2020. HEKA: A novel intrusion detection system for attacks to personal medical devices. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS'20)*. IEEE, 1–9.
[45] A. K. M. Iqtidar Newaz, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A. Selcuk Uluagac. 2019. Healthguard: A machine learning-based security framework for smart healthcare systems. In *Proceedings of the 6th International Conference on Social Networks Analysis, Management and Security (SNAMS'19)*. IEEE, 389–396.
[46] A. K. M. Iqtidar Newaz, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A. Selcuk Uluagac. 2020. A survey on security and privacy issues in modern healthcare systems: Attacks and defenses. *arXiv preprint arXiv:2005.07359* (2020).
[47] Sukhvir Notra, Muhammad Siddiqi, Hassan Habibi Gharakheili, Vijay Sivaraman, and Roksana Boreli. 2014. An experimental study of security and privacy risks with emerging household appliances. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS'14)*. IEEE, 79–84.
[48] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2014. Context aware computing for the internet of things: A survey. *IEEE Commun. Surv. Tutor.* 16, 1 (2014), 414–454.
[49] Mohammad Ashiqur Rahman, Md Hasan Shahriar, Mohamadsaleh Jafari, and Rahat Masum. 2019. Novel attacks against contingency analysis in power grids. *arXiv preprint arXiv:1911.00928* (2019).
[50] Luis Puche Rondon, Leonardo Babun, Kemal Akkaya, and A. Selcuk Uluagac. 2019. HDMI-walk: Attacking HDMI distribution networks via consumer electronic control protocol. In *Proceedings of the 35th Computer Security Applications Conference*. 650–659.
[51] Shammya Shananda Saha, Christopher Gorog, Adam Moser, Anna Scaglione, and Nathan G. Johnson. 2020. Integrating hardware security into a blockchain-based transactive energy platform. *arXiv preprint arXiv:2008.10705* (2020).
[52] Samsung. 2018. Marketplace in the SmartThings Classic app. Retrieved from https://support.smartthings.com/hc/en-us/articles/205379924-Marketplace-in-the-SmartThings-Classic-app.
[53] Samsung. 2018. Samsung SmartThings Development Guide. Retrieved from https://developers.smartthings.com/.
[54] Michael Schiefer. 2015. Smart home definition and security threats. In *Proceedings of the 9th International Conference on IT Security Incident Management & IT Forensics (IMF'15)*. IEEE, 114–118.
[55] Tara Seals. 2015. BlackHat: Critical ZigBee Flaw Compromises Smart Homes. Retrieved from https://www.infosecurity-magazine.com/news/blackhatcritical-zigbee-flaw-smart/.
[56] Md Hasan Shahriar, Nur Imtiazul Haque, Mohammad Ashiqur Rahman, and Miguel Alonso Jr. 2020. G-IDS: Generative adversarial networks assisted intrusion detection system. *arXiv preprint arXiv:2006.00676* (2020).

[57] Masudur R. Siddiquee, Tao Xue, J. Sebastian Marquez, Roozbeh Atri, Rodrigo Ramon, Robin Perry Mayrand, Connie Leung, and Ou Bai. 2019. Sensor fusion in human cyber sensor system for motion artifact removal from NIRS signal. In *Proceedings of the 12th International Conference on Human System Interaction (HSI'19)*. IEEE, 192–196.

[58] Amit Kumar Sikder, Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, Kemal Akkaya, and Mauro Conti. 2018. IoT-enabled smart lighting systems for smart cities. In *Proceedings of the IEEE 8th Computing and Communication Workshop and Conference (CCWC'18)*. IEEE, 639–645.

[59] Amit Kumar Sikder, Hidayet Aksu, and A. Selcuk Uluagac. 2017. 6thSense: A context-aware sensor-based attack detector for smart devices. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security'17)*. 397–414.

[60] Amit Kumar Sikder, Hidayet Aksu, and A. Selcuk Uluagac. 2019. A context-aware framework for detecting sensor-based threats on smart devices. *IEEE Trans. Mob. Comput.* 19, 2 (2019), 245–261.

[61] Amit Kumar Sikder, Hidayet Aksu, and A. Selcuk Uluagac. 2019. Context-aware intrusion detection method for smart devices with sensors. US Patent 10,417,413.

[62] Amit Kumar Sikder, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2019. Aegis: A context-aware security framework for smart home systems. In *Proceedings of the 35th Computer Security Applications Conference*. 28–41.

[63] Amit Kumar Sikder, Leonardo Babun, Z. Berkay Celik, Abbas Acar, Hidayet Aksu, Patrick McDaniel, Engin Kirda, and A. Selcuk Uluagac. 2020. Kratos: Multi-user multi-device-aware access control system for the smart home. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 1–12.

[64] Amit Kumar Sikder, Giuseppe Petracca, Hidayet Aksu, Trent Jaeger, and A. Selcuk Uluagac. 2018. A survey on sensor-based threats to internet-of-things (IoT) devices and applications. *arXiv preprint arXiv:1802.02041* (2018).

[65] Vijay Sivaraman, Dominic Chan, Dylan Earl, and Roksana Boreli. 2016. Smart-phones attacking smart-homes. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 195–200.

[66] Vijay Sivaraman, Hassan Habibi Gharakheili, Arun Vishwanath, Roksana Boreli, and Olivier Mehani. 2015. Network-level security and privacy control for smart-home IoT devices. In *Proceedings of the IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'15)*. IEEE, 163–167.

[67] Statista. 2018. Installed base of home automation/smart home systems in the United States from 2012 to 2017 (in millions). Retrieved from https://www.statista.com/statistics/286813/installed-base-of-smart-home-systems-us/.

[68] Biljana L. Risteska Stojkoska and Kire V. Trivodaliev. 2017. A review of Internet of Things for smart home: Challenges and solutions. *J. Clean. Prod.* 140 (2017), 1454–1464.

[69] Darlene Storm. 2016. Hackers demonstrated first ransomware for IoT thermostats at DEF CON. Retrieved from https://www.computerworld.com/article/3105001/security/hackers-demonstrated-first-ransomware-for-iot-thermostats-at-def-con.html.

[70] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. 2020. Light commands: Laser-based audio injection attacks on voice-controllable systems. In *Proceedings of the 29th USENIX Security Symposium*. 2631–2648.

[71] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, Xianzheng Guo, and Patrick Tague. 2017. Smartauth: User-centered authorization for the internet of things. In *Proceedings of the 26th USENIX Security Symposium*. 361–378.

[72] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A. Gunter. 2019. Charting the attack surface of trigger-action IoT platforms. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 1439–1453.

[73] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. 2018. Fear and logging in the internet of things. In *Proceedings of the Network and Distributed Systems Symposium*.

[74] Masaaki Yamauchi, Yuichi Ohsita, Masayuki Murata, Kensuke Ueda, and Yoshiaki Kato. 2019. Anomaly detection for smart home based on user behavior. In *Proceedings of the IEEE International Conference on Consumer Electronics (ICCE'19)*. IEEE, 1–6.

[75] Nong Ye et al. 2000. A Markov chain model of temporal behavior for anomaly detection. In *Proceedings of the IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, Vol. 166. 169.

[76] Amit Kumar Sikder, Z. Berkay Celik, and Leonardo Babun. 2018. A micro-benchmark suite to assess the effectiveness of tools designed for IoT apps. Retrieved from https://github.com/IoTBench/.

[77] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 103–117.

Received May 2020; revised September 2020; accepted October 2020