

IoT-AD: A Framework To Detect Anomalies Among Interconnected IoT Devices

Hasniuj Zahan, Md Washik Al Azad, Ihsan Ali, and Spyridon Mastorakis

Abstract—In an Internet of Things (IoT) environment (e.g., smart home), several IoT devices may be available that are interconnected with each other. In such interconnected environments, a faulty or compromised IoT device could impact the operation of other IoT devices. In other words, anomalous behavior exhibited by an IoT device could propagate to other devices in an IoT environment. In this paper, we argue that mitigating the propagation of the anomalous behavior exhibited by a device to other devices is equally important to detecting this behavior in the first place. In line with this observation, we present a framework, called IoT Anomaly Detector (*IoT-AD*), that can not only detect the anomalous behavior of IoT devices, but also limit and recover from anomalous behavior that might have affected other devices. We implemented a prototype of *IoT-AD*, which we evaluated based on open-source IoT device datasets as well as through real-world deployment on a small-scale IoT testbed we have built. We have further evaluated *IoT-AD* in comparison to prior relevant approaches. Our evaluation results show that *IoT-AD* can identify anomalous behavior of IoT devices in less than 2.12 milliseconds and with up to 98% of accuracy.

Index Terms—Internet of Things (IoT), anomaly detection, network traffic anomalies, device interaction anomalies

I. INTRODUCTION

With the proliferation of Internet of Things (IoT), different IoT environments, such as smart homes, have become a reality. Such environments may consist of a number of IoT devices from different vendors. These devices may be interconnected to each other and adhere to conditions defined by users (e.g., users select a temperature of their preference for their smart thermostat). In this context, there are several IoT device controller platforms, such as IFTTT, Samsung SmartThings, and the Apple Homekit¹, which integrate the services offered by IoT devices of different vendors. However, when these devices exhibit behavior that does not follow the conditions defined by users or report inaccurate readings, this behavior is defined as an anomaly. Given the interconnected nature of IoT devices, an anomaly occurred/triggered by an IoT device can propagate and affect other IoT devices.

We present an anomaly propagation scenario in a smart home (Figure 1). In this scenario, a faulty or compromised motion sensor interacts with and turns on a smoke sensor even if no smoke is actually detected in the house. As a

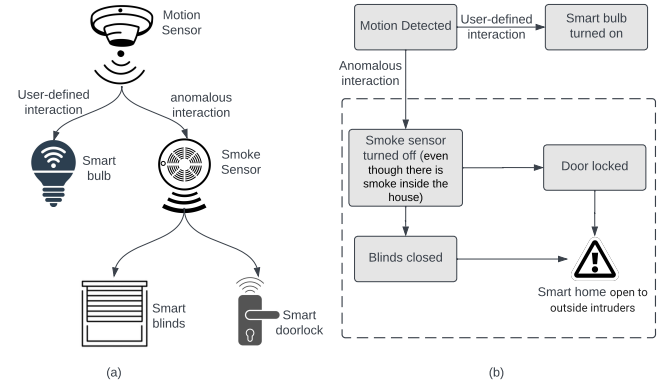


Fig. 1: (a) Interactions among IoT devices; (b) Propagation of an anomaly across IoT devices.

result, the smoke detector interacts with sensors that open the windows and unlock the door of the house, so that residents can escape the smoke and exit the house safely. In this context, the anomalous behavior of a faulty or compromised motion sensor has propagated and affected a smoke sensor, smart window sensors, and a smart lock, resulting in a situation that the residents' house is open to outside intruders.

Prior research has identified different types of anomalies and has proposed methods to identify these anomalies [1]–[4]. The majority of prior research has considered: (i) packet-level anomalies related to the network traffic generated by IoT devices (e.g., while communicating with controller platforms, cloud services, or other IoT devices); and (ii) state/behavior anomalies based on the analysis of the IoT device behavior over time. However, prior research has focused on anomaly detection methods, but did not investigate mechanisms to: (i) mitigate the effects of anomalies that propagate from one IoT device to another, thus affecting the state/behavior of multiple devices in an IoT environment; and (ii) enabling affected IoT devices to recover from such propagated anomalies.

Motivation and research questions: Solely detecting IoT device anomalies and isolating devices that experienced anomalies may not be sufficient in real-world scenarios, such as smart homes or buildings. Given the interconnected nature of real-world IoT environments, anomalies can propagate among IoT devices before they can be identified. In other words, an anomaly caused by an IoT device can affect other IoT devices. To this end, mechanisms are necessary to mitigate the effects of anomalies that can propagate among IoT devices. To address this challenge, in this paper, we investigate the following research questions:

Hasniuj Zahan (email: hzahan@unomaha.edu) and Ihsan Ali (email: ihsanali@ieee.org) are with the Department of Computer Science, University of Nebraska at Omaha, Nebraska, USA. Md Washik Al Azad (email: malazad@nd.edu) and Spyridon Mastorakis (corresponding author, email: mastorakis@nd.edu) are with the Department of Computer Science and Engineering, University of Notre Dame, Indiana, USA.

¹IFTTT: <https://www.ifttt.com/>, Samsung SmartThings: <https://www.smartthings.com/>, Apple Homekit: <https://developer.apple.com/apple-home/>.

- How can we realize a framework that detects packet-level anomalies and anomalies related to the interactions among IoT devices?
- How can such a framework enable affected IoT devices to recover from propagated anomalies and revert to their last known stable state as if the detected anomalies had never occurred?

Our contributions: In this paper, we present IoT Anomaly Detector (*IoT-AD*), a framework that: (i) detects packet-level anomalies and interaction anomalies, which can propagate across IoT devices; and (ii) mitigates the effects of propagated anomalies, enabling IoT devices to recover from propagated anomalies. *IoT-AD* essentially acts as an extended IoT device controller framework, which is able to detect anomalies in IoT environments and help associated IoT devices recover from these anomalies. The contribution of our work is two-fold:

- We present the *IoT-AD* design, which features mechanisms to detect packet-level anomalies and anomalies related to the interactions among IoT devices, maintain the state of IoT devices over time, and ultimately enable IoT devices to recover from anomalies that may have propagated among them.
- We implement an *IoT-AD* prototype, which we evaluate based on open-source IoT device datasets as well as through real-world deployment on a small-scale IoT testbed we have built. We further evaluate this prototype in comparison to prior relevant approaches for anomaly detection in IoT environments. Our evaluation results demonstrate that *IoT-AD* is a light-weight framework that can identify IoT device anomalies in less than 2.12 milliseconds (ms) and with up to 98% of accuracy.

The rest of our paper is organized as follows. In Section II, we discuss a brief background on IoT anomaly detection and mitigation, and present prior related work. In Section III, we present our threat model and a design overview of *IoT-AD*. In Section IV, we present the design of *IoT-AD*, in Section V, we present the evaluation of *IoT-AD*, and, in Section VI, we discuss extensions and limitations of the *IoT-AD* design. Finally, in Section VII, we conclude our paper.

II. BACKGROUND AND PRIOR RELATED WORK

In this section, we present a brief background on IoT anomaly detection and mitigation, and discuss related work.

A. Anomaly Detection of IoT Devices

The purpose of anomaly detection is to identify patterns whose behavior is considered atypical compared to typical ones. In prior research, different approaches have been used to detect anomalies in an IoT environment, such as traffic analysis, packet-level signatures, and semantics-based models. Homonit [5] is a framework, mainly for the Samsung SmartThings platform, which can detect anomalies by monitoring encrypted traffic patterns of smart devices' activities, and comparing these patterns with expected behaviors inferred from the source code. HADES-IoT [6] is a light-weight host-based IoT anomaly detection framework that can proactively

detect and prevent the execution of unauthorized functions on IoT devices. IoT-Praetor [7] leverages a device usage description model to profile different communication and interaction behaviors among IoT devices to automatically detect anomalies in real time. A semantics-based approach, called HAWatcher [8], was proposed by Wang *et al.*, where semantic information of IoT devices, such as applications, device types, installation locations, and event logs, is used to generate correlations. Subsequently, a shadow execution simulation is executed based on these correlations to simulate a smart home environment in parallel with the actual smart home environment. The behavior of the real and the simulated environment is compared to detect behavioral anomalies.

HomeSnitch [9] uses the perspective of software-defined networking to track communication between devices and servers, classify device actions, and find anomalous behavior. PingPong [10] is a tool that extracts packet-level signatures for device events (*e.g.*, a smart plug turning ON/OFF) from network traffic to identify anomalies in smart home networks. Orpheus [11] is an anomaly detection tool that uses event logs and system logs to detect data-oriented exploits and different runtime attacks. Xu *et al.* [12] proposed a system that analyzes home network traffic using a bloom filter to detect anomalous traffic behavior. Li *et al.* proposed a light-weight statistical learning approach for IoT devices where different system information, such as CPU and memory usage, and network throughput, can be used to detect anomalies [13]. The same authors also proposed a technique to detect attacks against IoT devices based on the energy consumption of different system components (*e.g.*, CPU, network) [14]. These systems mostly used either event logs to infer device activity or packet-level signatures to find out deviations from the usual traffic patterns.

Furthermore, anomaly detection frameworks have been proposed based on features of the physical layer, such as the Received Signal Strength Indicator (RSSI), the Power Spectral Density (PSD), and the Signal-to-Noise Ratio (SNR). Martins *et al.* used raw RSSI data to extract the silence and activity periods of IoT devices and detect anomalies based on this information [15]. Tang *et al.* proposed an anomaly detection framework for wireless sensor networks, which is also based on RSSI data [16]. Rajendran *et al.* proposed SAIFE, which analyzes the PSD of the wireless spectrum within an IoT environment to detect anomalies [17]. The challenges of detecting anomalies based on physical layer information are the following: (i) the collected signals may be susceptible to noise; and (ii) the features used to detect anomalies may depend on the distance between the sender and the receiver.

B. Machine Learning for IoT Device Anomaly Detection

Due to recent technological advancements, machine learning has become a powerful technique for the detection of anomalies by identifying irregular data patterns. Yin *et al.* [18] and Jakkula *et al.* [19] used a one-class support vector machine to identify activity patterns in a smart home using a sensor-based dataset. Ramapatrani *et al.* [20] used a hidden Markov model to train network-level data of a smart home in order to detect anomalies. Fahad *et al.* [21] used the density

TABLE I: Comparison of design properties of *IoT-AD* and other prior related work (Y: Yes, N: No, L: Limited).

	Homonit [5]	IoT-Praetor [7]	HAWatcher [8]	PingPong [10]	Yamauchi <i>et al.</i> [27]	IoT-AD
Traffic anomaly detection	Y	Y	N	L	N	Y
Interaction anomaly detection	L	Y	Y	N	Y	Y
Stable state recovery	N	N	N	N	L	Y

based spatial clustering algorithm for user activity recognition, while Trimananda *et al.* [10] used the same algorithm to cluster packet pairs and detect anomalies in network traffic. With the combination of two different techniques, the principal component analysis and a sliding window algorithm, Wu *et al.* [22] can identify user activities by analyzing WiFi signals. Branch *et al.* [23] used a K-NN machine learning method for outlier detection, while Narudin *et al.* [24] used both naive bayes and random forest machine learning algorithms to detect anomaly-based malware.

C. Anomaly Mitigation in Smart Homes

To reduce the effects caused by an anomaly, it is necessary to create prevention and recovery methods. Noah *et al.* [25] introduced a stochastic traffic padding algorithm that shapes the traffic to hide the metadata and keep user activities private. Gaurav *et al.* [26] proposed a framework to enhance the security of a smart home that uses attribute based access control. The advantage of using attribute based access control is the ability to specify fine-grained security policies and consider environmental conditions to make access decisions. Yamauchi *et al.* [27] proposed a system that detects anomalous events by analyzing event sequences and dropping packets associated with anomalous events.

How is *IoT-AD* different from prior related work: In Table I, we present a comparison of the *IoT-AD*'s design properties to prior work. The majority of prior work focused on: (i) anomalous behavior detection of IoT devices using traffic analysis; and/or (ii) analyzing the interaction patterns among the interconnected IoT devices in an IoT environment. For example, PingPong [10] provides a way to generate signatures of IoT events by analyzing the packet sequences, which can be used to identify anomalous events, while HAWatcher [8] generates hypothetical correlations among events of different IoT devices (*i.e.*, interactions among devices) to detect anomalies. IoT-Praetor [7] and Homonit [5] are specific to the Samsung SmartThings platform. A drawback of prior work is that these frameworks did not provide any mechanisms to help IoT devices recover from anomalies and revert to their last known stable states. To this end, *IoT-AD* not only utilizes traffic and interaction analysis to detect anomalies that can propagate among interconnected IoT devices but also provides mechanisms that enable affected IoT devices to recover from propagated anomalies and revert to their last known stable state as if the detected anomalies had not happened.

III. THREAT MODEL AND DESIGN OVERVIEW

In this section, we first describe the threat model that we consider in the context of *IoT-AD* and we then present an overview of the *IoT-AD* design.

A. Threat Model

As IoT devices are resource-constrained and manufactured for minimal and specific functions to reduce cost and complexity, these devices suffer from malfunctions and security vulnerabilities [28]–[31]. As a result, IoT devices may introduce anomalies into an IoT environment. In this paper, we have considered anomalies that are caused by: (i) device malfunctions; and (ii) devices compromised due to malicious attacks. We further assume that an IoT device controller platform that may be available will operate legitimately.

Anomalies due to IoT device malfunction: As IoT devices have limited power and computing resources, they are prone to malfunction. In most cases, there are no built-in malfunction detection mechanisms. As a result, it is challenging to identify a faulty or malfunctioning device in a timely manner and repair or replace the device. Device malfunctions can be either software-based or hardware-based.

Ghost commands, command failures, delays in status updates, and event losses occur as a result of software-based device malfunctions. Such malfunctions can happen due to a system crash, network connectivity issues, or bugs in application or operating system code. Hardware-based malfunctions can create false events and command failures. For example, a defective motion sensor can erroneously detect human presence when there are no humans around. Another example of hardware-based malfunctions is command failures due to melted capacitors and faulty circuit boards. Both types of malfunctions (software-based and hardware-based) can lead to unexpected communication patterns (*e.g.*, sending updates irregularly instead of periodically) or unexpected interactions with other IoT devices (*e.g.*, a thermostat turns on the fan even though the temperature sensor indicates that the room temperature has not changed). Both types of unexpected patterns can be used to identify anomalies generated by an IoT device.

Anomalies due to compromised IoT devices: Another reason for IoT device anomalies may be the fact that an IoT device has been compromised. Given that IoT devices are resource-constrained, malicious actors can compromise and gain access to these devices. For example, attackers can exploit weak usernames and passwords chosen by users, network vulnerabilities, or create malicious applications that users can install on their IoT devices. If attackers gain access to an IoT device, they can compromise the behavior of the device (*e.g.*, produce bogus readings), so that the device interacts with other devices and the anomaly propagates. In addition, attackers can use a compromised IoT device to access other IoT devices, which happened during the Mirai botnet attack [32]. Since the behavior/patterns of a compromised device will typically

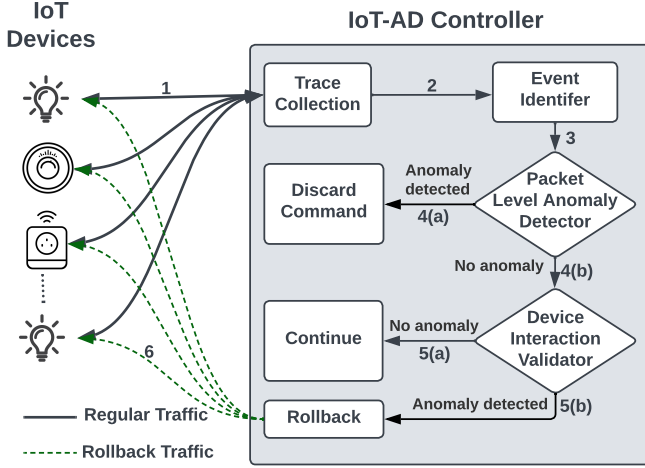


Fig. 2: Operation workflow of *IoT-AD*.

deviate from its regular operation patterns, such changes of device operation patterns can be used to identify anomalies.

B. IoT-AD Design Overview

In the context of *IoT-AD*, we consider an IoT environment (e.g., a smart home), where IoT devices are interconnected through a controller. The controller realizes the intelligence of the IoT environment, communicates with IoT devices, analyzes network traffic and interactions among IoT devices, and instructs devices to change their current state as necessary. *IoT-AD* operates in five steps (Figure 2). The controller is responsible for executing these steps, which we present below:

Device monitoring: The controller monitors all IoT devices and receives measurements/readings from them over time. These measurements/readings are transmitted to the controller over the local network through a series of packet exchanges.

Event identifier and device status update: Each IoT device communicates with the controller through a series of network packet exchanges (application layer protocols). Given the heterogeneous nature of IoT devices (e.g., different vendors, different software stacks), each device may implement different application layer protocols to communicate with the controller. For each device, the controller identifies events (as communicated by devices) that affect the status of each device and maintains a log where it records the status of each device over time. For instance, when a smart light bulb turns on (this is the event in our example), the bulb will communicate with the controller and the controller will update the status of this device to “on”.

Packet-level anomaly detection: As IoT devices communicate through the controller, the controller monitors the generated network traffic and produces packet-level signatures for different events. These signatures are used by the controller over time to identify packet-level (network traffic) anomalies. If a packet-level anomaly is detected, the controller will discard the event.

Device interaction validation: IoT devices interact with each other via the controller. We use the term “device interaction” to refer to events triggered by a measurement/reading of an

IoT device that affects one or more other devices. For example, if a motion sensor detects human presence in a room, based on the time of the day, it may trigger a smart light to turn on. The controller will validate the interactions among devices and identify anomalies related to these interactions. Such an anomaly may occur, for example, when a motion sensor attempts to trigger a smoke detector when it detects motion in a room of the house, since this interaction is not meaningful.

Rollback: Once an anomaly is detected, the controller identifies how much this anomaly has “propagated” among IoT devices. For example, a motion detector may attempt to trigger a smoke detector (meaningless interaction), which can subsequently unlock the smart lock of a house’s front door and windows (meaningful action, since residents will need to exit the house in the case of a fire). In other words, the controller will identify which devices have been affected by an interaction anomaly and revert the state of the affected devices back to the most recent stable state.

IV. *IoT-AD* DESIGN

In this section, we present the components of the *IoT-AD* design in detail. A list of symbols and abbreviations used in the design of *IoT-AD* is provided in Table II.

A. Device Monitoring

In *IoT-AD*, the controller is the entity that conducts a number of operations (e.g., packet-level anomaly detection, device interaction validation). To this end, existing IoT devices with adequate resources (e.g., smart TVs, smart refrigerators), resourceful WiFi routers, or smart hubs² can act as controllers. All devices in an IoT environment are connected via wireless to a controller, which is responsible for monitoring all devices within this environment and for collecting measurements/readings from these devices over time. Depending on the nature of an IoT device, the state information of devices can be updated to the controller either periodically or whenever a state change occurs. For instance, the temperature readings

TABLE II: List of symbols and abbreviations used in the design of *IoT-AD*.

Symbol/Abbreviation	Description
L2 headers	Link layer header fields
L4 headers	Transport layer header fields
L3 headers	Network layer header fields
SYN	TCP synchronization flag
ACK	TCP acknowledgement flag
FIN	TCP finish flag
TCP cmd	TCP command
TCP rsp	TCP response
Event e	An event detected by the controller

²Examples of a smart TV, a refrigerator, a WiFi router, and a smart hub:

- Smart TV: <https://developer.samsung.com/smarttv/development/specifications/general-specifications.html>
- Smart refrigerator: <https://www.samsung.com/us/home-appliances/refrigerators/all-refrigerators/>
- WiFi router: <https://www.asus.com/us/Networking-IoT-Servers/WiFi-Routers/ASUS-Gaming-Routers/RT-AX88U/>
- Smart hub: https://store.google.com/us/product/nest_wifi_specs

of a thermostat can be updated to the controller after a certain time interval. Another example may be that when an event occurs, such as a smart light bulb turning on or off, the state of the light bulb will be updated to the controller.

Apart from monitoring the states of IoT devices, the controller can also analyze packets that are exchanged through the controller among the devices in chronological order. When a user, for example, tries to turn on a smart device (e.g., a smart bulb) through an application, multiple packets are exchanged between the smart device and the application. The controller monitors various L2-L4 packet headers (e.g., IP addresses, TCP/UDP ports, packet lengths) and records the timestamp of each packet. The headers of exchanged packets are used to identify the communication patterns among devices and keep an event log for each device.

B. Event Identifier and Device Status Updates

An event is a series of packet exchanges between two or more IoT devices within the local network that have interdependencies (e.g., a motion sensor can turn on a light bulb or even multiple light bulbs and plugs). When a device communicates with a cloud server for a particular service, such as searching for a firmware update, sending device information, receiving control commands from a user application, this communication can be also identified as an event.

Figure 3 illustrates an example of packet exchanges between two IoT devices through the controller in *IoT-AD*. First, a TCP connection is established between device 1 and the controller. The device then sends a command related to an event (e.g., a motion sensor detecting motion in a room) to the controller. After the command is successfully received by the controller, the TCP connection with device 1 is closed, and the controller follows the same process to communicate a command with device 2. This command is triggered by the event of device 1 (e.g., a motion sensor that detected motion in a room, which in turn activated the light in this room). Although the lengths of the exchanged packets can be different, features, such as TCP flags, destination ports and source ports can be common depending on the device and event type. The sequence of packets and their features remain the same each time a particular event occurs. As a result, such packet bursts can be used to detect that an event has occurred at a particular time.

The *IoT-AD* controller keeps a log file for each device based on a chronological occurrence of events as shown in Figure 4(c). When an event is detected, the controller updates the log file of the device that generated this event with information related to the event. The controller stores logs related to an event e , at least, until e and all events triggered due to e have been validated. After that, logs of validated events can be archived on a remote cloud and be deleted by the controller if it does not have adequate resources.

Prior research [10], [33], [34] has shown that each event has a unique pattern of exchanged packets depending on the device and its manufacturer, which can be used as a signature. This is due to the fact that IoT devices typically conduct a limited number of operations (i.e., space of potential events).

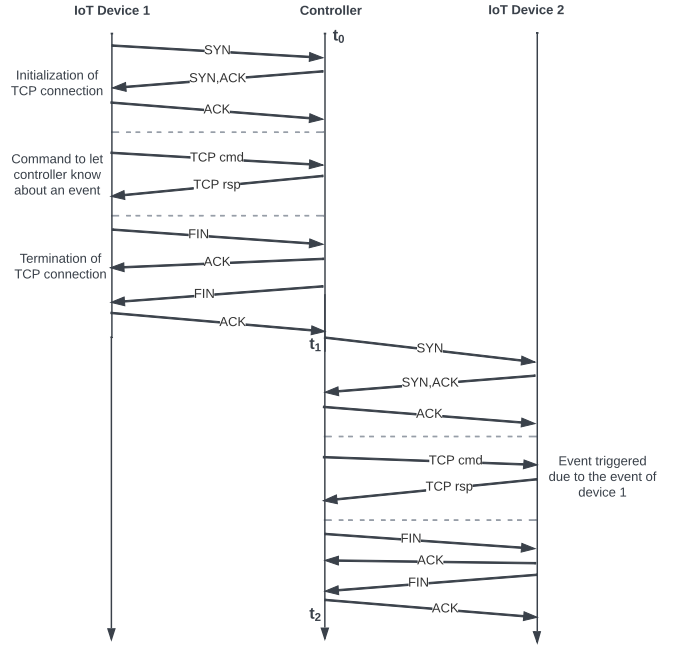


Fig. 3: An example of the network traffic generated for communication between two IoT devices through the *IoT-AD* controller.

In *IoT-AD*, we create packet signatures based on tuples of different header fields. Specifically, we have taken into account the following features: source and destination port numbers, the exchanged packet lengths, TCP flag types, the time span of each event, and the direction of the packets within that time span. Based on these features, *IoT-AD* generates packet signatures for various events.

C. Packet Level Anomaly Detection

An event will be considered as a packet-level anomaly if the set of exchanged packets for a particular event does not match (or does not have a similar pattern) to any prior generated event signatures. An anomaly can occur for various reasons, such as faulty devices and/or sensors, and device(s) compromised by an attacker.

IoT-AD uses a light-weight machine learning model running on the controller to detect packet-level anomalies. The model is trained based on previously extracted event signatures. Since each IoT device typically performs a specific set of operations over time, the features of potential events for available IoT devices can be collected as part of the training dataset. Once the model is trained, the controller uses this model to identify whether incoming events are valid. If an incoming event is valid based on the prediction of the model, the controller allows the event to continue. Otherwise, the event is deemed an anomaly and the controller terminates it.

D. Device Interaction Validation

If no packet-level anomalies are identified for an event, the controller will let the event be executed. The controller will also log event-related information, such as which device

triggered the event and which device's state has been changed because of that particular event. Logs of event-related information will be used to validate events that occur over time based on conditions defined by users and/or the vendor of each device. We provide examples of such conditions below.

We define a device interaction as an event on a device triggered by another device in the IoT environment. Similarly, the same event can be triggered by events of different devices. For example, an event that turns on a smart bulb can be triggered by a motion detector. The same bulb can also be turned on by a light sensor. An interaction among devices will be considered as a valid one if it follows user- or vendor-defined conditions, so that the IoT environment as a whole operates as expected. For instance, a user can set a condition that if the room temperature as detected by a smart temperature monitor exceeds a certain threshold value, then the monitor will send a command via the controller to turn on the AC and the fan. In this case, turning on the AC and the fan by the temperature monitor will be considered as valid interactions. However, if the temperature monitor tries to turn on a light, this will be considered as an invalid interaction (anomaly).

The controller creates and maintains data structures over time, called interaction trees. These trees represent sequences of interconnected events triggered by a measurement or a reading of a device. The device that starts an interaction is the root device of a tree. For example, in Figure 4(a), device A is the root device of the interaction tree. When the root device generates a new measurement/reading, this triggers the creation of a new interaction tree. Figure 4(a) shows how interaction trees are formed in *IoT-AD*. For interaction tree 1 of device A, two events are triggered on two different devices (device B and device C). Furthermore, the event of device B further triggers events on device D and device E.

To record the device interactions and the corresponding measurements of root devices in a consistent manner, we introduce a key-value pair-based logging format, which is used to map the information of each event to a unique key. A unique key is generated based on two different IDs as shown in Figure 5. The first part of the key "X" is the sequence number of the measurement/reading of a root device (for every new interaction tree, the sequence number is incremented) and "Y" is the sequence number of a specific event within the "X" interaction tree. We elected to not use timestamps as a part of the unique key generation, since time synchronization among IoT devices is a challenge on its own.

Within an IoT environment, each IoT device may act as the root device of different interaction trees. Each tree is generated dynamically when the root device generates a new measurement/reading (e.g., a temperature sensor may capture a new reading periodically every few seconds). *IoT-AD* makes use of automation rules and conditions (e.g., defined by users or device manufacturers) to verify interactions among devices. For example, as shown in Figure 4(b), a measurement/reading captured by device A triggers multiple events on other IoT devices. In this example, an anomalous event has occurred on device F, and *IoT-AD* detects the anomaly based on the automation rules and conditions. Subsequently, device F interacted with several other devices, as illustrated in Figure 4(b).

As a result, due to the anomalous event on device F, the state of device G, device H, and device I will also be affected.

E. Rollback

To mitigate the effects of anomalies that propagate among devices and help affected devices revert their state to the last known stable state, *IoT-AD* offers an automated recovery mechanism, called "rollback". The rollback process begins as soon as an interaction anomaly is detected. At this point, the *IoT-AD* controller will use the corresponding interaction tree to identify the devices that got affected due to this interaction anomaly. For instance, in Figure 4(b), an anomalous event on device F causes anomalous events on devices H, I, and G. *IoT-AD* creates a list of those devices and analyzes the logs of these devices to find out the last known stable state for each device. For instance, device F was triggered on due to the anomalous interaction with device A. Therefore, the last stable state of device F was "triggered off". Hence, *IoT-AD* sends commands to affected devices, so that they roll back their state to the last known stable state (in this example, "triggered off" for device F). Finally, *IoT-AD* isolates the device that initiated the anomaly until the device owner or administrator can troubleshoot.

The above mechanism essentially assumes that all interactions are legitimate until an anomaly is found. In other words, *IoT-AD* allows interactions to take place among devices (and potential anomalies to propagate among devices), while verifying such interactions asynchronously. If an anomaly is detected, the state of all affected devices will be rolled back. An alternative to this mechanism would be to first verify each interaction that a device would like to perform and, only if this interaction is valid, let the device interact with another device. In this case, anomalies would not be able to propagate, since devices would interact with each other only after each interaction has been verified. However, verifying automation rules, especially when the number of IoT devices grows and/or the rules are complex, may not happen instantly. To this end, we would need to essentially stop the operation of the IoT environment until an interaction has been verified.

V. EVALUATION

In this section, we first present our evaluation setup and we then discuss our evaluation results.

A. Evaluation Setup

Our evaluation process involves three parts: (i) we first use open-source IoT device datasets; (ii) we then build a small-scale testbed of IoT devices, which we use to deploy *IoT-AD* in the real world; and (iii) we finally compare *IoT-AD* to relevant approaches that have been previously proposed.

1) *Open-Source IoT Datasets*: In this part of our evaluation, we made use of three publicly available datasets of IoT device data. These datasets were collected through controller platforms, such as IFTTT and SmartThings. Based on this data, we generated packet-level signatures and then replayed this data in the *IoT-AD* controller for packet-level anomaly

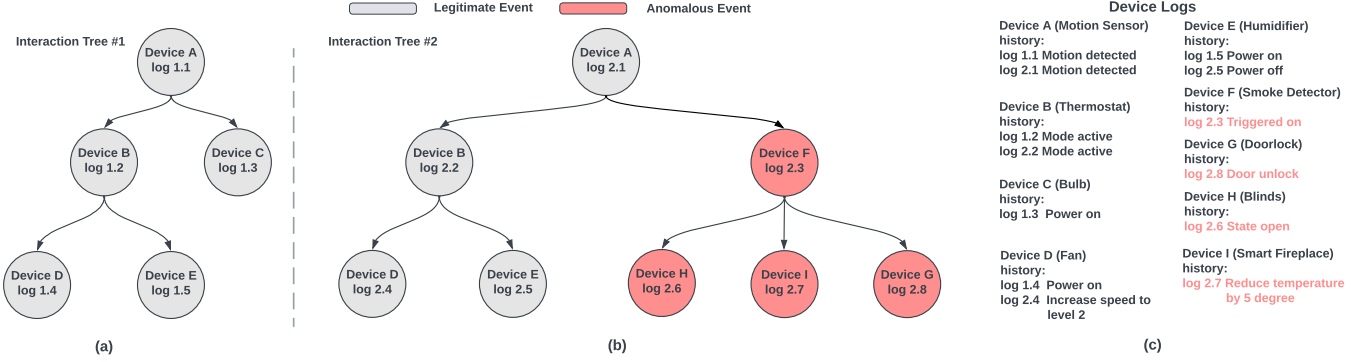


Fig. 4: Illustration of interactions among IoT devices.

TABLE III: Datasets used for the evaluation of *IoT-AD*.

Dataset	Setup	No. of devices	Duration	Content Type	Data Type
MUD [35]	Smart Home	10	21 days	Packet level Data + Device Interaction Data	pcap
Mon(IoT)r [36]	Smart Home	45	3 days per device	Packet level Data + Device Interaction Data	pcap
PingPong [20]	Smart Home	16	15 days	Packet level Data + Device Interaction Data	pcap
Testbed data	Smart Home	12	10 days	Packet level Data + Device Interaction Data	pcap

Fig. 5: Device log generation in *IoT-AD*. “X” is the sequence number of a measurement/reading of a root device, while “Y” is the sequence number of a specific event within the “X” interaction tree.

detection and device interaction validation. We summarize the characteristics of these datasets in Table III and we further discuss them below.

Manufacturer Usage Description (MUD) dataset [35]: This dataset includes real-world traffic traces of 10 different types of smart devices collected over a period of 21 days. It contains two types of traces: volumetric attack traces (e.g., ARP spoofing, TCP/UDP flooding) and benign traces. This dataset provides information about the number of MUD flows per minute, the start and the end time of an attack, and the MUD flows impacted due to an attack.

Mon(IoT)r dataset [36]: The Mon(IoT)r dataset was created by collecting traces from a number of devices over 85 days. The data was collected in two different countries (in the US and the UK). Each PCAP file of the dataset represents the traces related to an event of a specific device. The dataset has multiple instances of the same event for a device, which helps identify the patterns (generate signatures) for each event. This dataset also contains the timestamps of each event.

PingPong dataset [20]: This dataset contains PCAP files with network traffic traces of 22 popular commercially available IoT devices. The creators of the dataset identified different functions or events of these devices and captured them in these PCAP files. The dataset also includes the timestamps of the captured functions for each device.

2) *Small-Scale Smart Home Testbed:* We retrofitted a research space to an one-bedroom apartment to create a small-scale smart home testbed, where we evaluated the *IoT-AD* design. Figure 6 shows the layout of the testbed and the locations of the used IoT devices. Table IV lists the details for the devices we used. Our testbed contains a total of 12

TABLE IV: List of IoT devices used in our testbed and their abbreviated labels.

Abbr	Device Name
C	Intel NUC mini kit as controller
L1	Govee Smart Light
L2	Sengled Smart Bulb
L3	Tp-link Smart Bulb
L4	Kasa Smart Light
S1	TP-link Smart Plug
S2	Teckin Smart Plug
S3	WeMo Smart plug
S4	KMC 4 Outlet WiFi Smart Plug
S5	Amazon Smart Plug
T1	Ecobee Thermostat
E1	4th Generation Echo Dot with Clock
K1	Korex Smart Kettle

IoT devices. These devices can interact with each other via the controller of *IoT-AD* and trigger different events based on predefined conditions. All devices are wirelessly connected to the controller over a local network.

We used an Intel Next Unit Computing (NUC) mini kit as an *IoT-AD* controller in our testbed. The controller runs a daemon program that can monitor all packets exchanged among the IoT devices. We used PyShark, a popular Python library to capture network traffic and interaction traces from the devices over a period of 10 days. We also used different APIs that come from device vendors to interact with the devices and collect the timestamps of various device functions. More information about our testbed dataset is provided in Table III. We make the dataset collected from our testbed and the *IoT-AD* implementation code publicly available for the community to use³.

3) *Comparison to previously proposed approaches:* We compared the *IoT-AD* design to an Artificial Neural Network (ANN) based framework for IoT intrusion and anomaly detec-

³Our testbed dataset and *IoT-AD* implementation code are available at <https://github.com/Hasnuij-Zahan/IoT-anomaly-detector>.

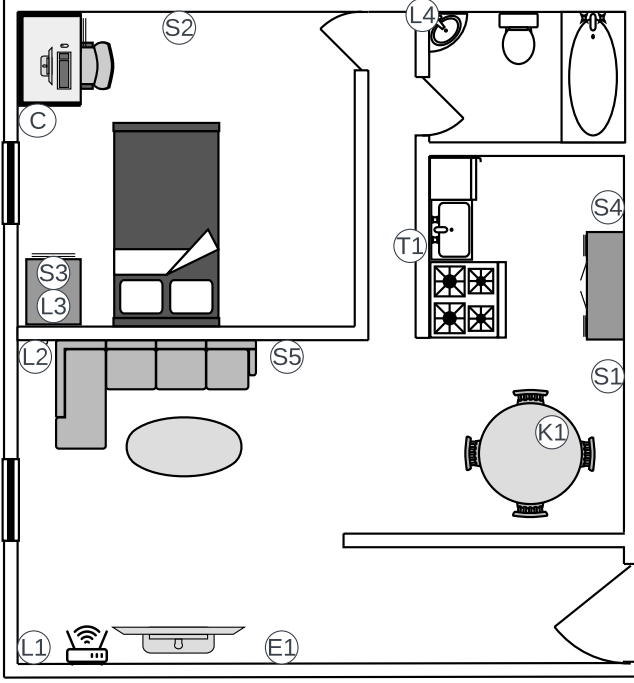


Fig. 6: Floor plan of our IoT testbed and device deployment layout.

tion [37] and a Convolutional Neural Network (CNN) based framework for IoT anomaly and intrusion detection [38]. For this comparison, we used the four datasets mentioned above (MUD, Mon(IoT)r, PingPong, and our testbed dataset).

4) *Evaluation Metrics*: For the evaluation of the detection of packet-level anomalies, we used four machine learning algorithms: a random forest algorithm [39], a k-Nearest Neighbors (kNN) algorithm [40], a decision tree [41], and an autoencoder [42]. We used these algorithms instead of more complex models (e.g., a neural network [43]) due to their lightweight nature. For the validation of device interactions, we implemented the *IoT-AD* controller in software (based on the design described in Section IV). We consider the following metrics for our evaluation:

- 1) *Inference time*: The time needed for the *IoT-AD* controller to identify whether: (i) a specific event contains packet-level anomalies; or (ii) a device interaction is illegitimate (anomalous).
- 2) *Memory usage*: The amount of memory required by the *IoT-AD* controller to detect packet-level anomalies and validate interactions.
- 3) *CPU usage*: The CPU usage of the *IoT-AD* controller while validating the interactions among devices and identifying packet-level anomalies.
- 4) *Accuracy*: The accuracy of different models in terms of identifying packet-level anomalies.
- 5) *Precision*: The precision of a model refers to the ratio of the number of true positive anomaly predictions to the total number of positive anomaly predictions.
- 6) *F1 score*: The F1 score of a model indicates the balance between the precision and the recall of the model for the

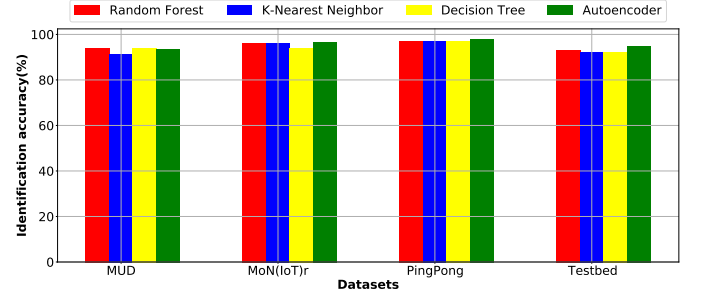


Fig. 7: Packet-level anomaly detection accuracy using *IoT-AD* for different datasets.

identification of anomalies.

B. Evaluation Results

1) *Evaluation of packet-level anomaly detection*: **Packet-level anomaly detection accuracy**: In Figure 7, we present our results on the identification accuracy of packet-level anomalies for different datasets and learning algorithms. Our results demonstrate that through the use of packet-level signatures that are created for IoT devices, *IoT-AD* is able to achieve over 90% and, in some instances, up to 98% of accuracy.

Precision of packet-level anomaly detection: Figure 8 shows the precision results of packet-level anomaly identification for different models and datasets. Similar to the accuracy results, *IoT-AD* achieves precision scores over 90% in all cases.

F1 score of packet-level anomaly detection: In Figure 9, we present the F1 scores of different anomaly identification models and datasets. The results of Figure 9 indicate that all models can identify packet-level anomalies with low false positive and false negative rates. Specifically, the models achieve F1 scores over 90% (and up to 98% in some instances).

Inference time: In Figure 10, we present the inference time per packet-level anomaly. Our results indicate that *IoT-AD* requires less than 2.12 milliseconds on average to identify whether an event contains a packet-level anomaly. The inference time depends on the nature of each device and dataset. For example, in the MoN(IoT)r dataset, the average inference time is higher as compared to other datasets. This is due to the fact that the IoT devices in this dataset are more complex in nature (e.g., fireTV, smart fridges), thus their events are represented through more complex communication protocols that involve more packet exchanges than rudimentary devices (e.g., smart bulb, smart switch). Further analysis of our results shows that another factor, which can impact inference time, is how “noisy” communication between devices is. In this context, noise typically includes traffic that is exchanged between devices, but is not related to actual events (e.g., traffic related to miscellaneous network protocols or traffic related to firmware updates).

Memory usage: In Figure 11, we present the results of the controller’s memory usage while detecting packet-level anomalies. Our results indicate that the controller requires a few hundred MB (up to 227MB) for the identification of packet-level anomalies. To this end, *IoT-AD* can detect packet-

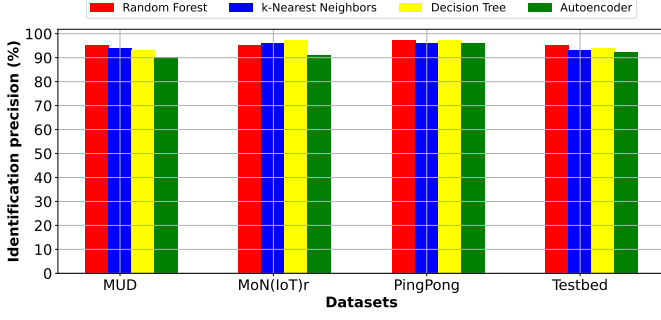


Fig. 8: Packet-level anomaly detection precision score using *IoT-AD* for different datasets.

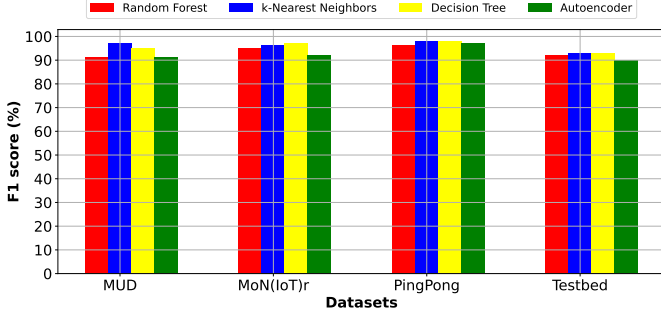


Fig. 9: Packet-level anomaly detection F1 score using *IoT-AD* for different datasets.

level anomalies even when deployed on controller devices with limited memory resources.

CPU usage: In Figure 12, we present the CPU load of the *IoT-AD* controller during the detection of packet-level anomalies. The controller uses less than 10% of its available CPU in order to identify packet-level anomalies in the traffic exchanged between IoT devices. Our results demonstrate that *IoT-AD* can detect packet-level anomalies even when the deployed controller has limited CPU resources.

2) Evaluation of interaction validation and rollback:

Inference time: In Figure 13, we present the inference time per device interaction based on different datasets. Our evaluation results show that the *IoT-AD* controller requires 1.74ms-1.83ms to validate an interaction and roll back to the last

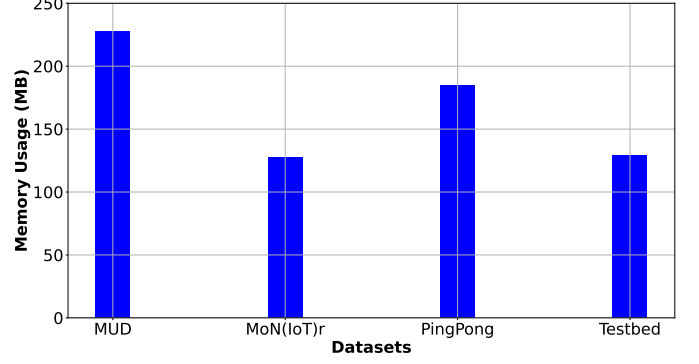


Fig. 11: Memory usage during packet-level anomaly detection.

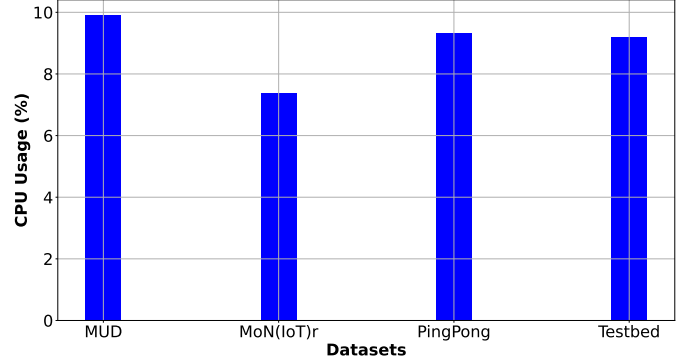


Fig. 12: CPU usage during packet level anomaly detection.

known stable state, if there is an anomaly.

Memory usage: Figure 14 shows the required memory to validate device interactions for different datasets. To run the device interaction validation module of the *IoT-AD* controller, it takes around 54 MB of memory on average to validate an interaction among devices. This module not only detects interactions among devices, but also detects device interaction anomalies and performs rollback, if necessary.

CPU usage: Figure 15 shows the results of CPU usage during device interaction validation and rollback when an anomaly is detected. The CPU usage during the device interaction validation and rollback process is 16.8%-23.1%. The results show that these processes are lightweight, so that they can

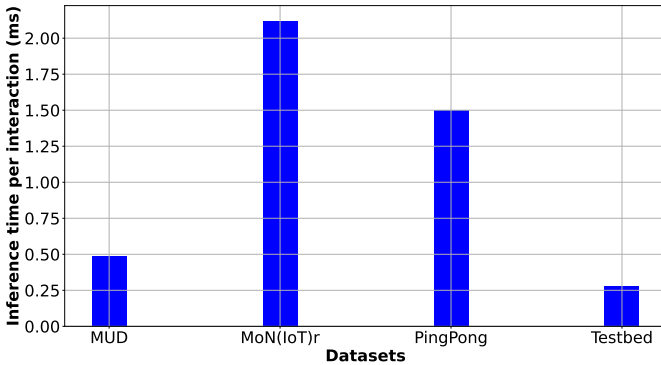


Fig. 10: Inference time for packet-level anomalies.

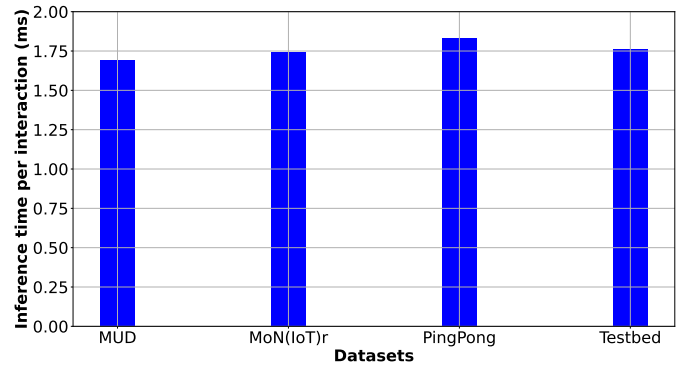


Fig. 13: Time for inference and rollback per device interaction.

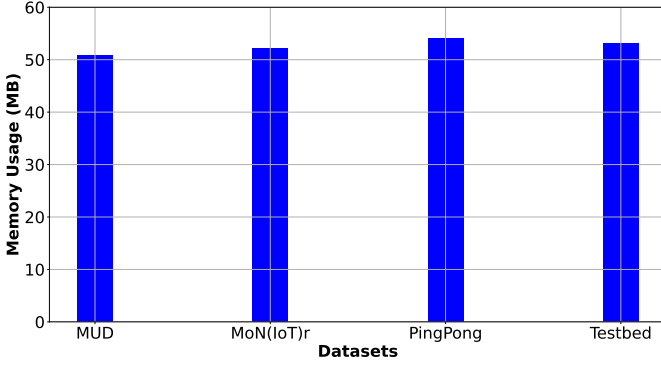


Fig. 14: Memory usage during device interaction validation and rollback.

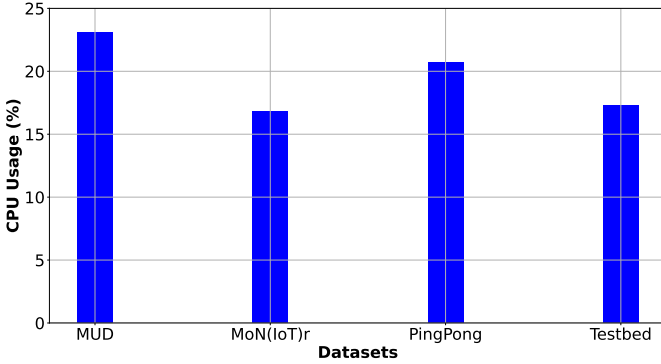


Fig. 15: CPU usage during device interaction validation and rollback.

run on IoT controllers with limited computing resources.

3) *Comparison to prior related frameworks*: Although *IoT-AD* achieves 1%-3% lower anomaly detection accuracy as compared to ANN and CNN based frameworks (because of their complex architectures) as shown in Figure 16, Figures 17 and 18 indicate that *IoT-AD* results in 22%-63% lower memory usage and 39%-62% lower CPU usage. These results indeed verify the lightweight nature of *IoT-AD*, which makes it suitable for deployment on resource-constrained IoT device controllers. In addition, Figure 19 shows that the inference time of *IoT-AD* is up to 48% lower than other frameworks, which is desirable for the detection of anomalies in real-time. Finally, unlike other frameworks, *IoT-AD* provides mechanisms to detect packet-level anomalies, validate interactions among interconnected and interdependent IoT devices, and rollback to the most recent known stable state(s) in case that an anomaly is detected.

VI. DISCUSSION

In this section, we discuss the process of selecting a device that can act as the *IoT-AD* controller and the placement of this controller. We also provide a security analysis of *IoT-AD* and discuss the limitations of the *IoT-AD* design as a whole.

A. IoT-AD Controller Selection and Placement

During the evaluation of the *IoT-AD* design, we chose a smart hub to act as the controller in the IoT environment.

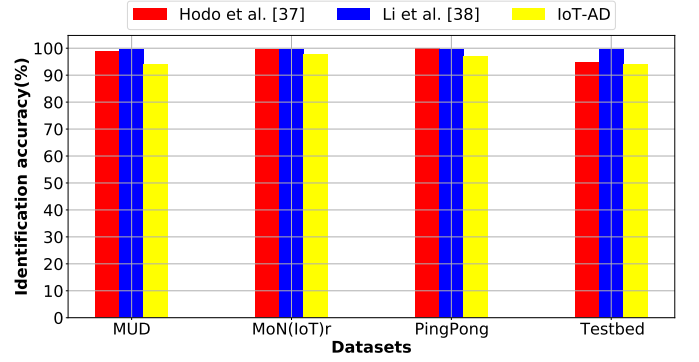


Fig. 16: Comparison of packet-level anomaly detection accuracy between *IoT-AD* and other frameworks.

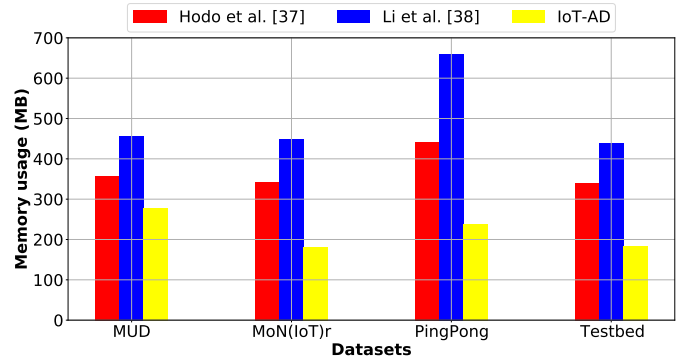


Fig. 17: Comparison of memory usage between *IoT-AD* and other frameworks.

Alternatively, we could have chosen an IoT device with adequate resources to act as the controller (e.g., a smart TV, a smart refrigerator). As a rule of thumb, any device with adequate resources to identify packet-level and interaction anomalies (such resource requirements have been quantified through our evaluation process in Section V) should be able to act as the controller.

The *IoT-AD* controller could be deployed within a local IoT environment or it could be remotely deployed on a cloud. A local controller could reduce the overall latency when it comes to detecting anomalies and helping IoT devices interact with

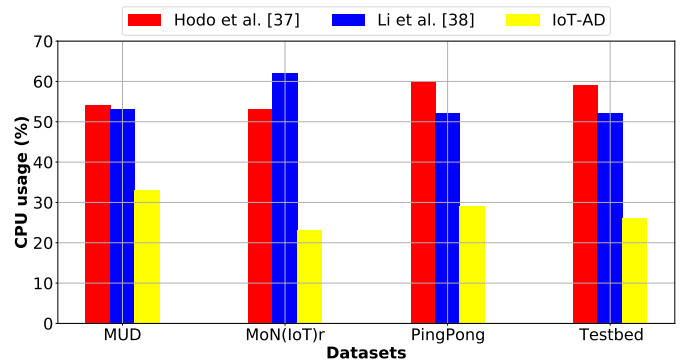


Fig. 18: Comparison of CPU usage between *IoT-AD* and other frameworks.

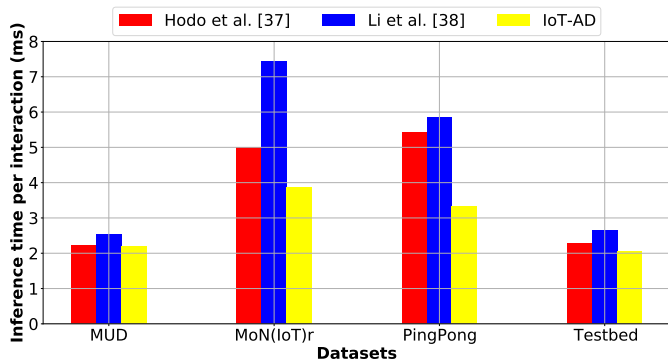


Fig. 19: Comparison of inference time between *IoT-AD* and other frameworks.

each other. Another benefit of having a local controller is that an external network observer (e.g., an Internet Service Provider (ISP) or a cloud service provider) cannot analyze the traffic to identify what is happening inside an IoT environment [44], [45]. On the other hand, a controller deployed on a remote cloud could make the maintenance of the controller transparent to users and help with scalability and reliability concerns.

B. Security Analysis of the IoT-AD Design

We consider different potential attack scenarios that can take place in an IoT environment and we analyze the ability of *IoT-AD* to mitigate these attack scenarios.

Distributed Denial-of-Service (DDoS) attacks: Due to vulnerabilities (such as weak authentication or encryption), IoT devices can be compromised by an attacker and used as a part of a botnet (e.g., Mirai [32]) to carry out DDoS attacks. In such scenarios, the *IoT-AD* controller monitors traffic from and to the IoT devices. If unknown traffic patterns are detected, which do not match legitimate functions and communication among IoT devices, the controller will identify these patterns as anomalies and isolate the IoT devices causing them. As a result, *IoT-AD* will be able to prevent the orchestration of DDoS attacks.

Spoofing attacks: If an IoT device gets compromised, an attacker can spoof its IP address and/or its MAC address. As a result, the attacker can send malicious information to another IoT device within the IoT environment to manipulate its state or to a network outside the IoT environment. Since the *IoT-AD* controller has a global view of the IoT environment (knowledge of MAC addresses and IP addresses of all devices), it can detect spoofed packets originating within the IoT environment and discard them.

Impersonation attacks: In this case, an attacker communicates with other devices as a legitimate device within the IoT environment and sends instructions to change the states of other devices. Because of the interaction trees created by *IoT-AD*, impersonating devices will be identified and eventually isolated from the network.

Passive attacks: Passive attacks (such as traffic analysis, sniffing, eavesdropping) can be used to gather valuable information about activities (events) within an IoT environment without altering packets [44]–[46]. *IoT-AD* can defend against this

type of attack by incorporating noise into the communication between the controller and IoT devices. The noise patterns will be agreed upon in advance between the controller and IoT devices. Since the controller will be aware of the noise patterns, it will be able to separate them from legitimate traffic.

C. Limitations of the IoT-AD Design

The *IoT-AD* design has the following limitations, which we plan to address in our future work.

Single point of failure: The *IoT-AD* controller introduces a single point of failure into an IoT environment. To mitigate this concern, two approaches can be used. The first approach involves maintaining multiple local IoT controllers in an IoT environment, since any IoT device with adequate resources can act as a controller. In this case, mechanisms to synchronize the state of local controllers will be needed. The second approach involves running a remote controller on a cloud. In this case, it would be the responsibility of the cloud provider to ensure that controller failure instances are effectively mitigated.

Anomalies out of *IoT-AD*'s scope: *IoT-AD* is designed to detect and recover from anomalies which are identified by monitoring network traffic and IoT device interactions. However, there are additional causes of anomalies, such as hardware or power failures of IoT devices, which *IoT-AD* does not currently consider. For example, the design of *IoT-AD* can be extended to detect anomalies of IoT devices using features of the physical layer. To achieve that, the controller will need to be equipped with additional hardware (e.g., software-defined radio, spectrum analyzer) to collect physical layer data and our models will need to be trained accordingly in order to identify anomalies based on physical layer data.

Scalability of the *IoT-AD* design: In an IoT environment with hundreds of IoT devices, such as a large building, a large number of events and interactions may take place. In such cases, the main challenge for *IoT-AD* would be to identify devices with adequate resources, which can act as controllers. A solution could be to maintain multiple controllers and assign groups of IoT devices to different controllers, so that the load is distributed among the available controllers.

VII. CONCLUSION AND FUTURE WORK

In this paper, we presented *IoT-AD*, a framework that detects and mitigates the impact of traffic and interaction anomalies among IoT devices. Our evaluation results demonstrate that *IoT-AD* is a lightweight framework that can identify IoT device anomalies in less than 2.12 ms and with up to 98% of accuracy. In our future work, we plan to extend the *IoT-AD* design in order to: (i) mitigate single point of failure concerns; (ii) identify additional categories of anomalies; (iii) support large-scale IoT environments with hundreds of IoT devices; and (iv) conduct additional experiments to further investigate the trade-offs of the *IoT-AD* design.

ACKNOWLEDGEMENTS

This work is partially supported by the National Science Foundation (awards CNS-2104700, CNS-2306685, CNS-2016714, and CBET-2124918) and ACM SIGMOBILE.

REFERENCES

- [1] A. K. Sikder, L. Babun, H. Aksu, and A. S. Uluagac, "Aegis: A context-aware security framework for smart home systems," *CoRR*, vol. abs/1910.03750, 2019. [Online]. Available: <http://arxiv.org/abs/1910.03750>
- [2] J. Choi, H. Jeoung, J. Kim, Y. Ko, W. Jung, H. Kim, and J. Kim, "Detecting and identifying faulty iot devices in smart home with context extraction," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2018, pp. 610–621.
- [3] U. A. B. U. A. Bakar, H. Ghayvat, S. F. Hasanm, and S. C. Mukhopadhyay, "Activity and anomaly detection in smart home: A survey," 2016.
- [4] F. Cauteruccio, L. Cinelli, E. Corradini, G. Terracina, D. Ursino, L. Virgili, C. Savaglio, A. Liotta, and G. Fortino, "A framework for anomaly detection and classification in multiple iot scenarios," *Future Generation Computer Systems*, vol. 114, pp. 322–335, 2021.
- [5] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "Homonit: Monitoring smart home apps from encrypted traffic," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1074–1088. [Online]. Available: <https://doi.org/10.1145/3243734.3243820>
- [6] D. Breitenbacher, I. Homoliak, Y. L. Aung, Y. Elovici, and N. O. Tippenhauer, "Hades-iot: A practical and effective host-based anomaly detection system for iot devices (extended version)," *IEEE Internet of Things Journal*, 2021.
- [7] J. Wang, S. Hao, R. Wen, B. Zhang, L. Zhang, H. Hu, and R. Lu, "Iot-praetor: Undesired behaviors detection for iot devices," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 927–940, 2021.
- [8] C. Fu, Q. Zeng, and X. Du, "HAWatcher: Semantics-Aware anomaly detection for appified smart homes," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 4223–4240. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity21/presentation/fu-chenglong>
- [9] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi, "Homesnitch: Behavior transparency and control for smart home iot devices," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 128–138. [Online]. Available: <https://doi.org/10.1145/3317549.3323409>
- [10] R. Trimnanda, J. Varmarken, A. Markopoulou, and B. Demsky, "Pingpong: Packet-level signatures for smart home device events," 2019. [Online]. Available: <https://arxiv.org/abs/1907.11797>
- [11] L. Cheng, K. Tian, and D. D. Yao, "Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks," ser. ACSAC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 315–326. [Online]. Available: <https://doi.org/10.1145/3134600.3134640>
- [12] K. Xu, F. Wang, and X. Jia, "Secure the internet, one home at a time," vol. 9, no. 16, p. 3821–3832, nov 2016. [Online]. Available: <https://doi.org/10.1002/sec.1569>
- [13] F. Li, A. Shinde, Y. Shi, J. Ye, X.-Y. Li, and W. Song, "System statistics learning-based iot security: Feasibility and suitability," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6396–6403, 2019.
- [14] F. Li, Y. Shi, A. Shinde, J. Ye, and W. Song, "Enhanced cyber-physical security in internet of things through energy auditing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5224–5231, 2019.
- [15] P. Martins, A. B. Reis, P. Salvador, and S. Sargento, "Physical layer anomaly detection mechanisms in iot networks," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.
- [16] J. Tang, P. Fan, and X. Tang, "A rssi-based cooperative anomaly detection scheme for wireless sensor networks," in *2007 International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE, 2007, pp. 2783–2786.
- [17] S. Rajendran, W. Meert, V. Lenders, and S. Pollin, "Saife: Unsupervised wireless spectrum anomaly detection with interpretable features," in *2018 IEEE international symposium on dynamic spectrum access networks (DySPAN)*. IEEE, 2018, pp. 1–9.
- [18] J. Yin, Q. Yang, and J. J. Pan, "Sensor-based abnormal human-activity detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, pp. 1082–1090, 2008.
- [19] V. R. Jakkula and D. J. Cook, "Detecting anomalous sensor events in smart home data for enhancing the living experience," in *Artificial Intelligence and Smarter Living*, 2011.
- [20] S. Ramapatrani, S. N. Narayanan, S. Mittal, A. Joshi, and K. Joshi, "Anomaly detection models for smart home security," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, 2019, pp. 19–24.
- [21] L. G. Fahad and M. Rajarajan, "Anomalies detection in smart-home activities," *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 419–422, 2015.
- [22] X. Wu, Z. Chu, P. Yang, C. Xiang, X. Zheng, and W. Huang, "Tw-see: Human activity recognition through the wall with commodity wi-fi devices," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 306–319, 2019.
- [23] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta, "In-network outlier detection in wireless sensor networks," *Knowledge and Information Systems*, vol. 34, no. 1, p. 23–54, Jan 2013.
- [24] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Comput.*, vol. 20, no. 1, p. 343–357, jan 2016. [Online]. Available: <https://doi.org/10.1007/s00500-014-1511-6>
- [25] N. J. Aphorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the smart home private with smart(er) iot traffic shaping," *CoRR*, vol. abs/1812.00955, 2018. [Online]. Available: <http://arxiv.org/abs/1812.00955>
- [26] G. Goyal, P. Liu, and S. Sural, "Securing smart home iot systems with attribute-based access control," in *Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*, ser. Sat-CPS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 37–46. [Online]. Available: <https://doi-org.leo.lib.unomaha.edu/10.1145/3510547.3517920>
- [27] M. Yamauchi, Y. Ohsita, M. Murata, K. Ueda, and Y. Kato, "Anomaly detection in smart home operation from user behaviors and home conditions," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 2, pp. 183–192, 2020.
- [28] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.
- [29] I. Andrea, C. Chrysostomou, and G. C. Hadjichristofi, "Internet of things: Security vulnerabilities and challenges," *2015 IEEE Symposium on Computers and Communication (ISCC)*, pp. 180–187, 2015.
- [30] H. Chi, Q. Zeng, X. Du, and J. Yu, "Cross-app interference threats in smart homes: Categorization, detection and handling," *CoRR*, vol. abs/1808.02125, 2018. [Online]. Available: <http://arxiv.org/abs/1808.02125>
- [31] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An in-depth analysis of iot security requirements, challenges, and their countermeasures via software-defined security," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10250–10276, 2020.
- [32] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis et al., "Understanding the mirai botnet," in *26th {USENIX} security symposium ({USENIX} Security 17)*, 2017, pp. 1093–1110.
- [33] M. F. B. Abbas and T. Srikanthan, "Low-complexity signature-based malware detection for iot devices," in *Applications and Techniques in Information Security*, L. Batten, D. S. Kim, X. Zhang, and G. Li, Eds. Singapore: Springer Singapore, 2017, pp. 181–189.
- [34] W. Yassin, N. I. Udzir, A. Abdullah, M. T. Abdullah, H. Zulzalil, and Z. Muda, "Signature-based anomaly intrusion detection using integrated data mining classifiers," in *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, 2014, pp. 232–237.
- [35] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on iot devices via sdn-based monitoring of mud activity," in *Proceedings of the 2019 ACM Symposium on SDN Research*, ser. SOSR '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 36–48. [Online]. Available: <https://doi.org/10.1145/3314148.3314352>
- [36] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 267–279. [Online]. Available: <https://doi.org/10.1145/3355369.3355577>
- [37] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of iot networks using artificial neural network intrusion detection system," in *2016 International*

- Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2016, pp. 1–6.
- [38] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, and L. Cui, “Robust detection for network intrusion of industrial iot based on multi-cnn fusion,” *Measurement*, vol. 154, p. 107450, 2020.
 - [39] G. Biau and E. Scornet, “A random forest guided tour,” *TEST*, vol. 25, no. 2, p. 197–227, Jun 2016.
 - [40] N. S. Altman, “An introduction to kernel and nearest-neighbor non-parametric regression,” *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.
 - [41] Y.-Y. Song and L. Ying, “Decision tree methods: applications for classification and prediction,” *Shanghai archives of psychiatry*, vol. 27, no. 2, p. 130, 2015.
 - [42] G. E. Hinton and R. Zemel, “Autoencoders, minimum description length and helmholtz free energy,” *Advances in neural information processing systems*, vol. 6, 1993.
 - [43] J. E. Dayhoff, *Neural network architectures: an introduction*. Van Nostrand Reinhold Co., 1990.
 - [44] N. J. Apthorpe, D. Reisman, and N. Feamster, “A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic,” *CoRR*, vol. abs/1705.06805, 2017. [Online]. Available: <http://arxiv.org/abs/1705.06805>
 - [45] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, “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*, ser. WiSec ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 207–218. [Online]. Available: <https://doi.org/10.1145/3395351.3399421>
 - [46] N. Apthorpe, D. Reisman, and N. Feamster, “Closing the blinds: Four strategies for protecting smart home privacy from network observers,” *arXiv preprint arXiv:1705.06809*, 2017.