Seeing Is Believing: Extracting Semantic Information from Video for Verifying IoT Events

Chenglong Fu
chenglong.fu@charlotte.edu
University of North Carolina at Charlotte
Charlotte, North Carolina, USA

Xiaojiang Du
xdu16@stevens.edu
Stevens Institute of Technology
Hoboken, New Jersey, USA

Qiang Zeng
Zeng@gmu.edu
George Mason University
Fairfax, Virginia, USA

Zhenyu Zhao
z.zhao@temple.edu
Temple University
Philadelphia, Pennsylvania, USA

Fei Zuo
fzuo@uco.edu
University of Central Oklahoma
Edmond, Oklahoma, USA

Jia Di
jdi@uark.edu
University of Arkansas
Fayetteville, Arkansas, USA

ABSTRACT

Along with the increasing popularity of smart home IoT devices, more users are turning to smart home automation platforms to control and automate their IoT devices. However, IoT automation is vulnerable to spoofed event attacks. Given that IoT devices are intricately linked with the physical environment and operate autonomously, event-based attacks can pose serious safety and security challenges. Our observations show that many IoT events are accompanied by visual modifications in objects such as shape alterations (for example, contact sensor events correspond with door movement) or changes in color/brightness (for example, a functioning microwave oven with the internal light switched on). These alterations can be detected by the commonly deployed smart cameras, providing a visually rich but challenging to manipulate channel for verifying IoT events. We introduce IoTSentry, the first system of its kind to extract high-level semantic information from streaming video data and pixels for IoT event verification. We have designed a Siamese deep neural network to identify variations in the appearance of IoT devices and interior objects. These are used as the yardstick for verifying IoT events received at IoT automation platforms. Upon assessing IoTSentry with 21 IoT devices (8 types), the results demonstrate that IoTSentry can be trained within 120 seconds, yielding an accuracy rate of over 96.7% in recognizing device states. We have deployed the 21 IoT devices and IoTSentry on two real-world smart home test sites. Over the course of our one-week evaluation, IoTSentry consistently achieved an average detection rate of 99.24% in identifying attack instances. Moreover, it triggered no more than 2 false alarms per day on each test site.

CCS CONCEPTS

• **Security and privacy** → *Artificial immune systems*; • **Computer systems organization** → *Sensors and actuators.*

WiSec '24, May 27–30, 2024, Seoul, Republic of Korea

ACM ISBN 979-8-4007-0582-3/24/05…$15.00
https://doi.org/10.1145/3643833.3656124

KEYWORDS

Internet of Things, Security, Smart Home

ACM Reference Format:
Chenglong Fu, Xiaojiang Du, Qiang Zeng, Zhenyu Zhao, Fei Zuo, and Jia Di. 2024. Seeing Is Believing: Extracting Semantic Information from Video for Verifying IoT Events. In *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '24), May 27–30, 2024, Seoul, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3643833.3656124

1 INTRODUCTION

In recent years, the global IoT market has experienced a boom in development, projected to reach $1,8 trillion in 2028 [11]. Among these, the smart home IoT market possesses the largest share, at 79.13 billion USD in 2020, and boasts a compound annual growth rate of 25.3% [34]. Concurrently, IoT devices have increasingly become the targets of attacks [6, 13, 19, 20, 30, 42, 45, 50, 56]. These attacks pose serious safety and security concerns due to the close integration of IoT devices with the physical environment.

The interoperability of smart home automation platforms allows IoT event attacks to pose risks not only to compromised devices but also to others within the network. Attackers can exploit vulnerabilities in these devices to trigger malicious activities. For instance, by spoofing the "user present" event, coupled with the automation rule "unlock the door when the user is back," they create an opportunity for burglaries. An attacker gaining full control over a compromised device complicates the task of verifying the device's integrity.

There are approaches aimed at detecting spoofed events using side-channel information, notable among them [16, 31]. However, these existing methods have two major limitations. Firstly, verification of each type of IoT event typically requires a specific corresponding sensor deployed in close proximity. The diversity of IoT events implies a need for various types of IoT devices to be densely deployed. Secondly, event verification robustness is strongly influenced by the devices' relative distances.

To mitigate these limitations, we suggest leveraging real-time video footage from home security cameras to verify IoT events. Home security cameras, being one of the most common smart home IoT devices, are installed in approximately 14.6% of American households [2]. This number increases to 42% in residences

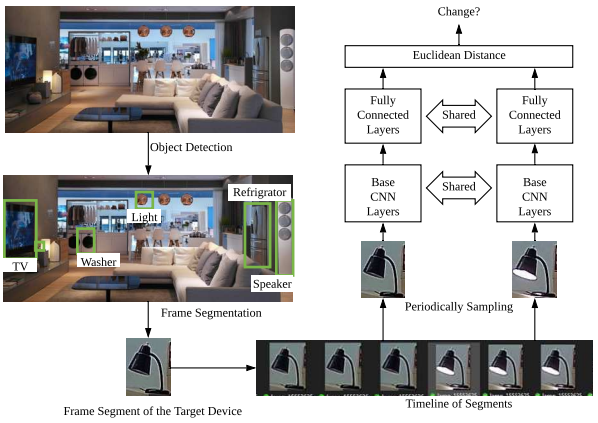

Figure 1: Workflow of IoTSentry.

equipped with smart home IoT systems [1], indicating a significant inclination towards incorporating security cameras as a part of smart home setups. Furthermore, the indoor cameras takes the largest share of 38.9% [3], which implies that home security cameras are highly likely to be installed together with other smart home IoT devices. In the future, home security cameras that are carried by self-navigating drones (e.g., Ring Always Home Cam [4]) can further expand the coverage of video surveillance. Many IoT events produce visible evidence such as light flickers or door movements, which can be captured by cameras. By monitoring discrepancies in reported states, cameras can detect faulty or malicious events. Besides, cameras can oversee a larger area and achieve higher sampling rates, significantly surpassing other sensors.

While using cameras to detect motion and recognize faces is commonplace, extracting semantic information from video data for IoT event verification is a novel tactic. In this study, we demonstrate that cameras can be used as "broad-spectrum" sensors, leveraging the rich semantics in videos for verifying various IoT events.

Our approach utilises recent advances in deep neural networks (DNNs) to identify state changes of IoT devices from videos. However, creating a universal DNN model poses three prominent challenges. First, the features and appearance of IoT devices are highly diverse, making it difficult, if not impossible, to build a model that is universally applicable to all IoT devices. Second, recognition accuracy can be greatly influenced by variations in viewing angle, distance, and lighting conditions. Finally, in specific IoT deployments, most devices exhibit infrequent state changes resulting in a limited data pool for training a model from scratch.

We propose IoTSentry, a video-based IoT event verification system that aims to tackle the aforementioned challenges. We design a semantics-assisted transfer learning approach to overcome the training overhead and the model generalization issue. Specifically, instead of seeking a universally applicable model, a semi-finished Deep Neural Network (DNN) model for devices sharing similar appearance characteristics is prepared by IoT researchers, device vendors, or platform operators. These semi-finished models can be fine-tuned and adapted for IoT devices deployed in different households. That is, models of different devices share the same base model, with top layers fine-tuned for each device.

We apply the Siamese Convolutional Neural Network (CNN) to detect changes between two images of the same device, rather than detecting its state directly. This method ensures more accurate detection results as the two frames are captured in quick succession, maintaining the same viewing angle, distance, and lighting conditions. Furthermore, the process can generate a large training dataset of image pairs derived from a small number of images. The key idea is that the state changes recognized by the AI system should align with the IoT events reported to an IoT platform.

The workflow of the proposed detection system is illustrated in Figure 1. The home automation platform selects a pre-trained base CNN model to be shared by all devices. Each image from the video is segmented using object detection models such as YOLO [48] and SSD [43]. Then, we add fully connected layers on top of the base CNN model to form the Siamese network and fine-tune the model using pairs of image segments for each device. Sample frames are then periodically selected from the video footage for comparison in the Siamese network. A low similarity score produced for two consecutive sample images indicates a change in the device's state during the interval when the two images were taken.

To evaluate our proposed system, we first conduct a validation to test the performance of the Siamese DNN model on detecting the state of 21 IoT devices in an offline setting. The results show that our fine-tuned DNN model achieves an accuracy of 98.77% in recognizing devices' state changes, and 96.67% in recognizing specific states by comparing the image with reference samples. We also assess the reduction of training overhead due to the semi-finished model by comparing the training of the fine-tuning model with direct training. Results indicate that our methods reduce training overhead by 45% on average. Finally, we deploy 21 IoT devices and the prototype of our IoTSentry on two smart home test beds, conducting a week-long test with injected attack cases. IoTSentry successfully detects 99.24% and 97.23% of event spoofing attack attempts. Moreover, IoTSentry yields no more than two false alarms per day on each test bed, demonstrating its practical usability.

Our contributions are as follows:

- This is the first IoT event verification solution by utilizing inexpensive cameras that are already prevalent in many homes. It offers a broader spectrum of verification, as it accommodates many more types of IoT devices and promises more reliable performance.
- We propose the image comparison method to achive more robust detection accuracy and two-stage transfer learning method to alleviate the workload of model training.
- We evaluate IoTSentry on two real-world testbeds. The results show that IoTSentry achieves device state recognition accuracy of over 96% on 21 devices across 8 types. During a one-week deployment, the IoTSentry prototype successfully detected 98.94% of generated attack cases while yielding fewer than 2 false alarms per day.

The remainder of this paper is structured as follows. In Section 2, we introduce the fundamental knowledge of smart home IoT systems. We then present our threat model along with our assumptions of potential attackers in Section 3. The design and implementation of IoTSentry is explained in Section 4, and the evaluation details are outlined in Section 5. In Section 6, we enumerate related works

and compare IoTSentry with other event verification solutions. Finally, in Section 7, we assess the limitations and future work, concluding in Section 8.

2 BACKGROUND

2.1 Smart Home Automation Platforms

With the number and variety of IoT devices increasing rapidly, smart home IoT platforms such as Samsung SmartThings, Google Home, and Amazon Alexa are gaining significant popularity in the IoT market. These IoT platforms provide uniform messaging interfaces, enabling easy management and interoperability among IoT devices of different types and from different vendors.

More importantly, these platforms introduce the new feature of trigger-action programming, which allows the implementation of user-customized automation rules. These automation rules enable IoT actuators to work autonomously when triggered by the specified trigger. On these automation platforms, IoT devices are abstracted into various template abstract devices that define the uniform messaging format of *events* and *commands*. The states of each device are stored on the platform's cloud server and updated based on received events.

However beneficial they may be, automation rules also expose new attack vectors, enabling attackers to trick automation platforms by manipulating events. By creating fake events, attackers can exploit automation rules to wrongly trigger or disable the execution of the associated automation rules, which can eventually cause hazardous situations. For example, in [18], the authors demonstrate how a door can be maliciously unlocked with a 'light turned on' event, and in [29], the authors illustrate how to trigger fake alarms with spoofed smoke events. Subsequent works [23, 55] further explore more security issues across multiple rules.

2.2 Vision-based Smart Home Safety

There has been an increasing number of home security camera vendors adapting deep learning to their products. This technique enables cameras to provide more accurate and intelligent alerts regarding potential safety risks [52]. For example, Arlo smart cameras can recognize and capture human faces as well as common objects such as vehicles, package deliveries, and animals from surveillance video streams [54]. Some works further explore applications such as fall [25] and inactivity [27] detection for elderly people.

Besides the home camera itself, cloud computing infrastructures have also been established to support these advanced AI applications which are scalable and cost-effective. For example, the Amazon Web Service (AWS)'s Rekognition service costs as low as 0.4 USD for processing 1000 images using newest computer vision models. These modules can seamlessly work with smart cameras that are integrated through the AWS IoT platform. Moreover, edge computing devices such as Deeplens and Panorama also empower AI-based applications to be hosted locally with acceptable cost.

3 THREAT MODEL

In this work, we examine potential attackers who can remotely control IoT devices to initiate event spoofing attacks. Equipped with this ability, attackers can malevolently prompt actions of other devices associated with the manipulated event, using home automation rules. The event spoofing attack might occur due to compromised devices [50] or vulnerable cloud APIs of the device vendor [35], which assist attackers in injecting spoofed events directly. These spoofed events can be utilized by attackers to trigger false security alarms [15] or instigate dangerous actions such as unlocking smart locks or activating heaters [18, 36].

Attackers may also possess the potential to maliciously intercept safety-critical commands, thereby inducing dangerous situations. For instance, attackers could intercept a command intended to turn off an electric heater when the user departs from home, while fabricating the anticipated feedback event of "plug turned off" to maintain their stealth. Different methods to verify the appropriateness of these actions have been extensively examined in other research works [18, 36, 44] and are beyond our scope. In this work, our objective is to detect failed command executions.

In this study, we presuppose that home cameras used for collecting training and testing samples are not compromised by attackers. We assume these cameras can provide genuine video footage or image snapshots whenever requested by our IoTSentry.

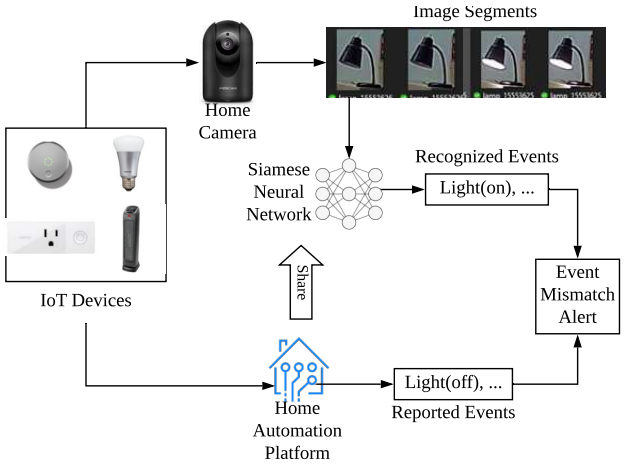

Figure 2: Overview of IoTSentry.

4 IOTSENTRY SYSTEM DESIGN

In this section, we present the design and implementation of IoTSentry. We initially describe the constituent components of IoTSentry, followed by a discussion on their usage to address the challenges presented in Section 1.

As shown in Figure 2, IoTSentry operates by comparing events recognized by the DNN model with those reported by smart home IoT devices. IoTSentry is comprised of four modules: 1) The Image Segment Extractor, which utilizes existing object detection models to divide an image frame into device and object segments and match them with identities on the home automation platform. 2) The Device State Recognizer, tasked with recognizing state changes of devices using a DNN model. 3) The Event Exporter, designed to export real-time events from various smart home automation platforms. 4) The Attack Detector acts as the decision module by invoking the Device State Recognizer and the Event Exporter, comparing their results and reporting any mismatches as alarms.

4.1 The Image Segment Extractor

Commonly used smart home security cameras have wide view angles up to 110 degrees [7] to cover a large portion of a room. Although the expansive image frame allows coverage of more devices, irrelevant objects could potentially be included, causing interference with device state recognition. Therefore, as a preliminary step, we first need to segment the entire image frame into smaller sections, each of which represents a device or object. We achieve this goal by using existing object detection solutions such as Fast R-CNN [32], Single Shot MultiBox Detector [43], and Yolo [48] to highlight objects in the image frame with bounding boxes. We examine the performance of different versions of these three object detectors that are pre-trained on the COCO dataset [41], finding that Yolo v4 offers the best detection accuracy. Once a device or object has been detected, we record the coordinates of its bounding box and use this to directly extract the segment in future frames. With visualized bounding boxes and labels, users can conveniently map the segments to the corresponding IoT devices' identities on the automation platform.

4.2 The Device State Recognizer

The Device State Recognizer is the core component of IoTSentry, tasked with recognizing device states using the vision side channel. This system accepts a pair of image segments of the same device from the Image Segment Extractor as input and then predicts whether the state of the device in the two image segments is identical. Despite the latest advancements in deep neural network (DNN) and computer vision [59] achieving high precision in object classification, the development of a universally applicable DNN for all devices is currently impractical given the variety of IoT devices and the persistent introduction of new models. Alternatively, training dedicated models for specific devices could provide a solution, but this approach presents two significant challenges. First, it requires a substantial number of image samples to create the training dataset, which might take an extended period to collect from a typical smart home deployment, considering many IoT devices' states (e.g., a door) only change a few times each day. Second, training DNN models is computationally intensive, taking into account the scale of installed IoT devices. Simplifying by fine-tuning models pretrained on public datasets would invariably result in high training overhead and potentially compromise system scalability.

To address the initial challenge posed by the training dataset, we employ the Siamese neural network model [17]. Instead of training models to directly recognize the state of a given device, we train a model capable of determining whether the devices in two segments are in the same state. Using image input, we explore the combinations of image segments and assemble them into pairs labeled either 'same' or 'different'. This method allows for the expansion of even a small number of image segments into a substantial dataset. For instance, as depicted in Figure 3, we can generate 5,000 training image pairs from 100 images of a door (50 open and 50 closed) for the purpose of training the Siamese network. This set includes 2,500 same-state pairs and 2,500 different-state pairs.

Although fine-tuning pre-trained CNN models has proven to be effective in reducing training overhead, fine-tuning DNN models for a tremendous number of IoT devices still faces scalability issues.

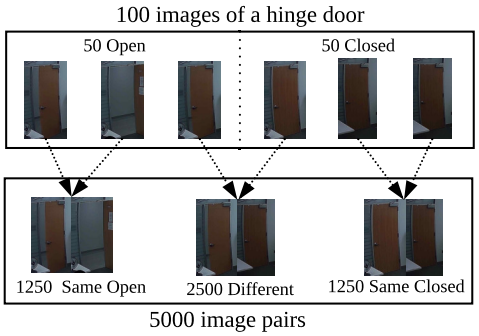

Figure 3: The example of the hinge door is used to illustrate how the training dataset can be expanded using image pairs.

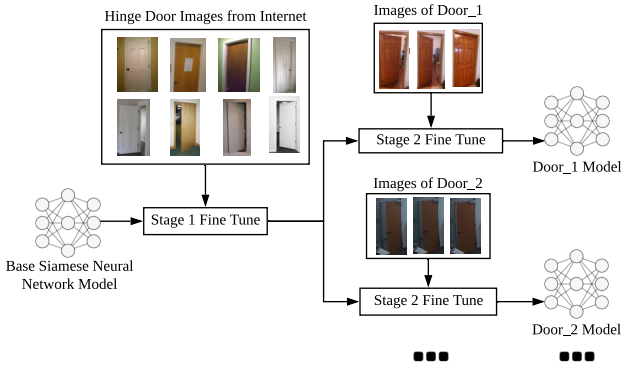

Figure 4: Workflow of the two stage fine tune.

To further reduce the training overhead, we propose a two-stage fine-tuning scheme. As illustrated in Figure 4, it begins with fine-tuning a Siamese network model that integrates a pre-trained base CNN model with images of a type of device. Then, we further fine-tune the model with images of a specific IoT device to allow it to accurately recognize the device's state changes. This scheme assists in reducing training overhead because certain common features learned in the first stage can be shared among different devices of the same type.

In the first stage, we train the model with images of device types collected from the Internet, allowing us to include a greater variety of images. Owing to the variance in images, the training and validation accuracy of the first stage fine-tuning may show only a slight improvement, thereafter stagnating as semi-finished. Although the resultant semi-finished model cannot achieve satisfactory accuracy, it has already learned some common patterns of that type of device. In the second stage, we further fine-tune the semi-finished model with images captured for specific devices. Leveraging the embedded patterns, the training time and resource overhead can be significantly reduced. Taking the hinged door as an example, we first train the Siamese network model with images of hinged doors gathered from the Internet, which include images of doors of different colors, textures, and sizes. We manually label the downloaded images as either 'open' or 'closed,' deriving a pairwise training dataset to train the semi-finished model. In the second stage, we continue to train the semi-finished model with images of a specific hinged door. The

accuracy can be improved to a satisfactory extent with considerable reduction in effort.

4.3 The Event Exporter

The event exporter is deployed on a Virtual Private Server (VPS) to collect real-time events from IoT devices across various automation platforms. In our experiment, the event exporter gathers events from multiple sources, given that devices may be integrated into different smart home automation platforms. For platforms allowing user-customized automation programs like SmartThings [8], we install a smart app to automatically report device events to our VPS via HTTPS requests. Contrarily, for platforms that lack this support (for instance, Alexa [9] and Tuya [12]), we link them to the IFTTT platform [10] and utilize an applet to synchronize the states of associated devices with a virtual service hosted on our VPS. Within our VPS, we establish virtual device instances for each device subjected to monitoring and keep an account of their states. The state of a virtual instance changes every time it receives an event from the automation platform.

4.4 The Attack Detector

As the decision module of IoTSentry, the Attack Detector scrutinizes a device's state as determined by the DNN model and events reported by the Event Exporter. Instead of being a passive observer, the Attack Detector actively retrieves devices' states from two distinct channels both periodically and when new events occur. With a predetermined sampling period set at 60 seconds, sufficient time is afforded for the manifestation of the event's visual effect. The DNN model within the Device State Recognizer is only equipped to verify whether the state of a device visible in two different images is identical. Consequently, we must monitor a device's state by comparing the most recent image of a device to a reference image. Upon receiving an event, an immediate image of the related device is taken and paired with the most recent image taken during the periodic sampling.

Confirmation of the event by the Device State Recognizer, signified by a 'different' output, renders the event harmless. The virtual instance's state, as maintained by the Event Exporter, is then changed accordingly. While in the absence of an event, two images from consecutive periods are paired, and the Device State Recognizer is expected to produce a 'same' output. A conflict between the results (R1) from the Device State Recognizer and those from the Event Exporter initiates a further validation process.

The Attack Detector randomly selects an image from the device's training dataset to serve as the reference image. The state of the device in the reference image is identifiable through its label. This reference image is then paired with the disputed image, and the device state recognition procedure is repeated to yield a second result (R2). Inconsistency between R1 and R2 results in the former being disregarded as inaccurate, indicating an issue with the two consecutive images. On the other hand, if R2 aligns with R1, the Event Exporter's result is deemed problematic. In such cases, the Attack Detector sends out an alert, such as an app notification on the user's phone, to notify the user of a possible attack.

5 EXPERIMENT & EVALUATION

In this section, we present our experiment in a real-world setting with commercial IoT devices. We first describe the experiment setup and then evaluate the accuracy and overhead of the Device State Recognizer. Afterwards, we describe the results on two real-world testbeds where we generate event spoofing attacks for testing.

5.1 Experiment Setup

Table 1: List of Tested Devices. The devices tested are classified into eight types. In the case of the door and refrigerator, the term 'device' refers to the contact sensors affixed to them.

Label	Name	Events	Amount
D	Room Door	open/close	3
R	Refrigerator Door	open/close	2
L	Light Bulb	on/off	3
M	Monitor	on/off	2
H	Electric Heater	on/off	2
O	Microwave oven	on/off	2
K	Electric Kettle	on/off	2
P	Smart Plug	on/off	5

There are 21 IoT devices of 8 types (listed in Table 1) within the two testbeds, which consist of a studio apartment and a two-bedroom apartment. In each testbed, we have installed a single Foscam smart home camera [14] costing less than $50 from Amazon.com. The camera is strategically positioned at the corner of the kitchen/living room area to capture all devices within a single frame, with a resolution set at 1920*1080. For the devices lacking built-in smart control features, we utilize smart plugs to control them and to monitor their operational status. For instance, devices such as an electric heater, microwave oven, and monitor, are connected to smart plugs. The power output measurements obtained serve as an indicator of their functional states. A power threshold is set at 10 watts: events reporting power measures exceeding this threshold are labeled as "turned on" and those reporting less as "turned off". For the room door and refrigerator door, we have affixed contact sensors. For both the training and testing, we are using a workstation with an Intel i9-9820X CPU and two Nvidia 2080Ti GPUs.

Ethical Concerns. Our experiment received approval following review by the IRB committee at our university. In a bid to circumvent privacy issues stemming from the usage of the camera, we blocked the camera's internet access through the addition of firewall rules at the home router, thereby forcing the camera to function without a memory card. Every image was collated by triggering the camera's snapPicture CGI [5] from participant-owned PCs via the participant's local area network. Image segmentation was also carried out through the users' PC to enable users to exclude redundant parts of images potentially containing sensitive information. Furthermore, we enabled participants to view their accumulated image segments first, subsequently removing any containing sensitive details prior to further processing in the workstation. The data drive within the workstation is encrypted; access is strictly limited to the authors of this paper.

Figure 5: Samples from our image dataset are taken for each device under four distinct lighting conditions.

5.2 Image Data Collection

Our Siamese network model was trained using image samples collected from the internet or segmented from image frames captured by installed cameras. For the stage 1 fine-tuning, we retrieved images of each type of device from the internet, based on a criterion requiring the device and its operational state in the image to be easily identifiable by humans. We procured 100 images representing both the 'on/open' and 'off/closed' states for every type of device, such as an electric oscillating radiant heater. Irrelevant sections of the downloaded images were manually cropped to focus on the target device only.

For stage 2 fine-tuning, we captured images of devices using cameras placed in two testbeds. The location of these cameras was selected based on the same criterion used in stage 1 data collection, guaranteeing that a human viewer can easily determine all testing devices' state through the image. As is common with most smart home cameras, we allowed the cameras to remain stationary, maintaining a steady position and angle during the image collection period. As depicted in Figure 5, we recorded images of each device under differing lighting conditions at four separate times: 1) in the morning with abundant natural light; 2) at dusk, where the natural light dims; 3) at night under artificial light; 4) at night with the room in darkness, switching the camera to its IR night vision mode. We gathered 50 images of each state of a device for training and validation, along with another 10 images aimed for evaluation.

Following the image collection, we combined the image samples into pairs of two different states and sorted through their combinations. This method allowed us to acquire 2,500 same-state pair samples and an equivalent number of different-state pair samples for the training of each device. As for pairs featuring images of the same state, we labeled them as '1'. Meanwhile, pairs with images of different states were labeled as '0'.

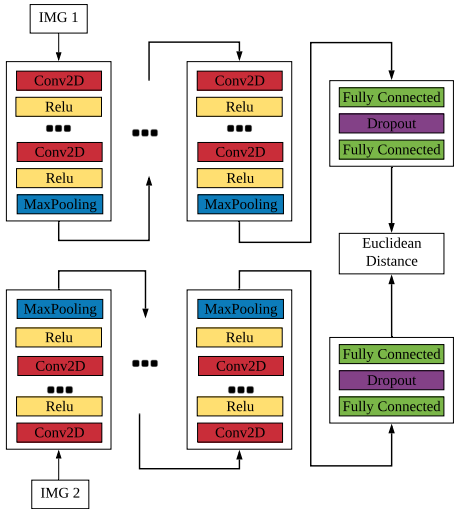

Figure 6: The architecture of our Siamese network model.

Table 2: Hyperparameters of Our Siamese Network Model.

Parameters	Value Space	Optimal
FC1 Nodes	{128,196,256,512,1024}	512
FC2 Nodes	{128,196,256,512}	512
Dropout Rate	[0.05, 0.5]	0.3
Learning Rate	[0.0001, 0.5]	0.02
Batch Size	{8,16,32}	16
Base CNN	{VGG16, Resnet50, InceptionV3, Plain CNN}	Resnet50

5.3 Device State Recognition

In this subsection, we carry out a validation experiment to assess the performance of the Device State Recognizer. We conduct the evaluation using an offline approach using our collected dataset.

5.3.1 Siamese Network Model. We take the standard approach to construct our Siamese model, the structure of which is shown in Figure 6. In this model, our foundation is formed by layers of convolutional blocks using CNN models that are pre-trained on the ImageNet [26] dataset. We discard the top classification layers of the CNN model, utilizing the remainder as the feature extractor. We then freeze the weights of the convolutional blocks. On top of these foundational convolutional layers, we incorporate two fully connected layers utilizing the ReLU activation function and interspersed these with a single dropout layer. The final stage is calculating the Euclidean distance between the vector outputs of the two input images to produce a result. If the distance is less than 0.5, we deduce that the devices shown in the two input images are in the same state. Conversely, if the distance is equal to or exceeds 0.5, we infer that they are in different states.

5.3.2 Hyper-parameters. For training our siamese model, certain hyperparameters (as outlined in Table 2) need to be determined. In addition to the usual hyperparameters like batch size and the number of nodes in the two fully-connected layers, we also need

to compare the performance of varied pre-trained CNN models used as feature extractors, and select the one that yields the highest accuracy. The ideal set of hyperparameters is determined by applying the NNI (Neural Network Intelligence) [38] tool to scour the parameter value space automatically. Remember that we test eight types of devices (Table 1). One representative device is selected from each of these eight types. We bypass the fine-tuning stage 1 and directly train the Siamese model with stage 2 images.

The Siamese network model incorporates seven hyperparameters for each device type (enumerated in Table 2). The findings demonstrate that all eight types of devices perform optimally with a dropout rate of 0.3, and a batch size of 16. Regarding other parameters, we observe no significant improvement, exceeding 1%, in state recognition accuracy with a count of nodes in fully-connected layers exceeding 512 and a learning rate below 0.02. Consequently, we adopt 512 and 0.02 to minimize training overhead. As it relates to the choice of a base CNN feature extractor, the plain CNN is discarded as it displays an accuracy that is at least 25% lower than alternative choices. Among the four pre-trained CNN models, Resnet50 records the highest average accuracy.

Table 3: Accuracy of device state recognition.

Device	Stage 1 Accuracy			Stage 2 Accuracy		
	Pairwise	On/ Open	Off/ Closed	Pairwise	On/ Open	Off/ Closed
D1	90.45%	70.00%	80.00%	98.73%	90.00%	100.00%
D2	84.52%	80.00%	90.00%	100.00%	100.00%	100.00%
D3	88.95%	70.00%	90.00%	99.67%	90.00	100.00%
R1	87.33%	70.00%	90.00%	100.00%	100.00%	100.00%
R2	91.17%	80.00%	90.00%	100.00%	100.00%	100.00%
L1	92.16%	100.00%	80.00%	100.00%	100.00%	100.00%
L2	91.43%	100.00%	80.00%	100.00%	100.00%	100.00%
L3	80.43%	90.00%	70.00%	100.00%	100.00%	100.00%
M1	97.45%	100.00%	90.00%	100.00%	100.00%	100.00%
M2	95.31%	90.00%	90.00%	100.00%	100.00%	100.00%
H1	95.22%	70.00%	60.00%	96.15%	90.00%	90.00%
H2	58.74%	40.00%	70.00%	93.54%	90.00%	80.00%
O1	86.72%	50.00%	60.00%	100.00%	100.00%	100.00%
O2	56.16%	40.00%	60.00%	96.32%	100.00%	80.00%
K1	66.34%	50.00%	40.00%	93.39%	100.00%	90.00%
K2	69.01%	60.00%	70.00%	98.21%	100.00%	90.00%
P1	76.31%	80.00%	70.00%	100.00%	100.00%	100.00%
P2	85.50%	70.00%	90.00%	100.00%	100.00%	100.00%
P3	52.99%	40.00%	50.00%	100.00%	100.00%	100.00%
P4	88.34%	80.00%	90.00%	98.16%	90.00%	100.00%
P5	89.92%	70.00%	80.00%	100.00%	100.00%	100.00%
Ave.	**82.12%**	**71.43%**	**75.71%**	**98.77%**	**97.62%**	**96.67%**

5.3.3 Accuracy of State Recognition. With the optimal hyperparameters, we conduct a two-stage training and evaluate the accuracy of the Siamese network model in recognizing the states of specific devices. In the first stage of training, we train the newly constructed model with image samples downloaded from the internet. This step yields eight semi-finished models for the eight types of devices/objects. In the second stage of training, for each device, we use the corresponding semi-finished model of its type and continue to train the model with image samples captured by our cameras.

We measure the accuracy of the models using three metrics: 1) pairwise accuracy is the accuracy of predicting whether the devices in a pair of images are in the same state; 2) accuracy in recognizing a device's 'on' state; 3) accuracy in recognizing a device's 'off' state. The final two accuracy measures are determined by comparing the testing samples with reference image samples.

As the results presented in Table 3 indicate, we evaluate the accuracy of the semi-finished models from stage one using an evaluation dataset. These models achieve an average pairwise accuracy of 0.82. In terms of state recognition accuracy, they achieve a score of 0.71 and 0.75 for recognizing the 'on' and 'off' states, respectively. For some devices showing more pronounced appearance changes when changing states, such as lamps and monitors, the semi-finished models achieve better accuracies, nearing 90%. This is because these devices share more common features typical of their device type, which can be discerned from the downloaded images. Meanwhile, other devices with a more varied appearance and fewer common features (e.g., different smart plugs display different colors and shapes of indicator lights), the recognition accuracy is only slightly better than an untrained model (i.e., 50% pairwise accuracy).

Following this, we conduct the second stage of training by fine-tuning the eight models for their corresponding devices and then evaluate them using the same evaluation dataset. As demonstrated in Table 3, after fine-tuning, accuracies for both pairwise and specific states significantly improve, exceeding 0.96. Out of the 21 tested devices, models trained via stage two fine-tuning achieve 100% accuracy on 13 devices. We subsequently examine the cases of H2 and O2, which have comparatively lower accuracies. The heater, H2, is a radiator heater that only exhibits changes in appearance through the brightness of its indicator light. In the two instances where the heater was misclassified as being in the 'on' state, we noticed that its indicator light was illuminating due to nearby light sources, preventing even humans from accurately determining its working state. Unlike H2, H1 does not face this issue because it is an infrared radiant heater that shows a color change in a large area when turned on. Misclassification of the microwave oven, O2, was caused by a similar issue: sometimes the front panel reflects natural light, making it appear as though the interior light is turned on. After augmenting our training dataset with image samples of these particular cases, our models achieve 100% state recognition accuracy for both devices.

5.3.4 Reduction of Training Overhead. To highlight the influence of stage 1 fine-tuning on curbing the total training overhead, a single device from each type is selected to illustrate the contrast in its processes of training based on stage 1 versus training without stage 1. As depicted in Figure 7, when the finish threshold for training is defined as an accuracy level of 98%, it requires an average of 3.375 epochs to train a model without stage 1. In contrast, when utilizing the semi-finished model from stage 1 training, this number dwindles to 1.625, representing a decrease in computation by 51.86%. For certain device types with more shared features, such as doors, lamps, refrigerators, and monitors, the reduction in training overhead exceeds 75%. Moreover, for items like electric kettles and smart plugs, which have either minor visual alterations or highly varied

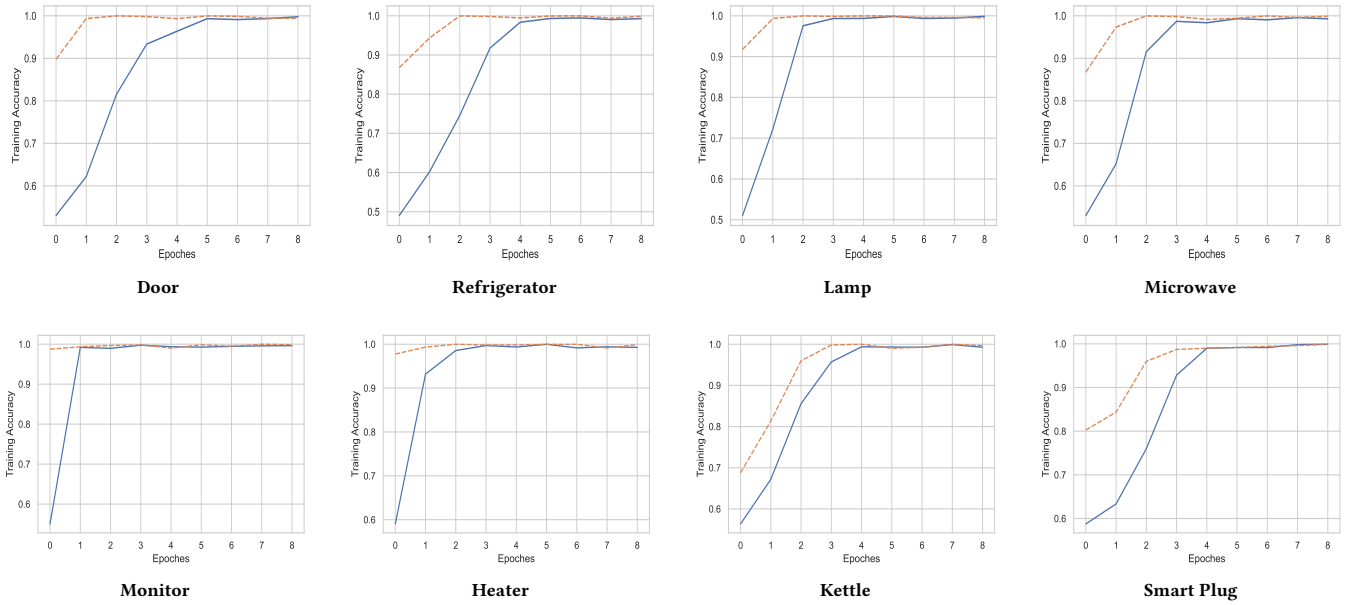

Figure 7: Comparison of Stage 2 Training Overhead With and Without Stage 1 Fine-Tuning: Each chart depicts a dashed line representing the improvement in training accuracy attributed to Stage 1 fine-tuning. Conversely, the solid line represents the scenario without the fine-tuning of Stage 1.

appearances, stage 1 training still proves effective and diminishes training overhead by 25%.

In order to fully comprehend this enhancement, we analyzed the duration of both stage 1 and stage 2 training. Following the standard resizing of input image samples to 224*224, a single iteration of a 16-image pair batch is completed in 88 milliseconds on our workstation. Given that the Siamese network structure is consistent in both stages, the training expense for one iteration with the same batch size remains equal. For the stage 1 training with a set of 20,000 training image pair samples, each epoch necessitates 1,250 iterations, totaling approximately 110 seconds. In contrast, stage 2 training, with fewer input image samples, requires merely 312 iterations and culminates in under 30 seconds. This signifies that training a Siamese network model specifically for any device would require no longer than 120 seconds (4 epochs) on our dual-GPU workstation. This overhead can be further mitigated by training a more extensive semi-finished model in stage 1 using a larger sample size of images.

5.4 Detection of Event Attacks

Aside from the validation test for the Device State Recognizer, we have also implemented other parts of IoTSentry on two testbeds. We conducted end-to-end tests to evaluate its performance in detecting event spoofing attacks.

5.4.1 Attack Case Generation. During the week-long testing period, the residents in two testbeds keep their normal living patterns and generate 1,559 events that are relevant to our testing devices/objects from the two testbeds. Then, for each device, we randomly inject 100 events into the collected event log for simulating event spoofing attacks targeting that device. Please note that, rather than triggering the devices' actions, spoofed events are only inserted into the events log, which aligns with recently disclosed smart home IoT attacks via the automation platforms and communication links, and integration channels. [18, 29, 35].

For the spoofing attack event, 200 spoofed events were randomly injected during the week-long testing period.

5.4.2 Performance of Detection. The test cases are evaluated autonomously. For each test case, we construct a time window beginning 60 seconds before the event occurrence and concluding 60 seconds post-event. Subsequently, we gather all image samples from the specific device, captured within the defined time window. To illustrate, in a scenario of a spoofed 'open' event of a door at instance t_e, we collect images of the door captured within the time boundaries $[t_e - 60, t_e + 60]$. We then monitor the device's state within our defined time window to verify event occurrence. In the event of spoofing attack cases, we deem the detection process successful if the device state recognizer reports no change in state within the image samples in the observed time window.

The results presented in Table 4 demonstrate that IoTSentry achieves an average detection rate of 99.24% in detecting the event spoofing attack. Moreover, IoTSentry maintains a 100% detection rate in 10 out of the 21 testing cases. Upon examining detection accuracies across different devices, we observed a similar trend as with the validation test of the device state recognizer. That is, the state changes of devices with implicit features, such as H2, K2, and smart plugs, result in a higher number of failed testing cases, proving difficult to distinguish even for human observers.

Table 4: Performance of IoTSentry in detecting event spoofing attack cases.

Device	Testing Cases	Detected Cases	Detection Rate	False Alarms
D1	100	99	99.00%	2
D2	100	99	99.00%	1
D3	100	100	100.00%	0
R1	100	100	100.00%	0
R2	100	99	99.00%	2
L1	100	100	100.00%	0
L2	100	100	100.00%	1
L3	100	100	100.00%	0
M1	100	100	100.00%	0
M2	100	100	100.00%	0
H1	100	99	99.00%	1
H2	100	98	98.00%	3
O1	100	100	100.00%	1
O2	100	97	97.00%	1
K1	100	100	100.00%	1
K2	100	96	96.00%	4
P1	100	99	99.00%	0
P2	100	100	100.00%	1
P3	100	99	99.00%	1
P4	100	98	98.00%	1
P5	100	100	100%	0
Ave.			**99.24%**	**1.78/day**

5.5 False Alarm Test

The false alarm rate is a critical aspect of a detection system; frequent false alarms can pose nuisances to users, possibly compelling them not to use the system. A dedicated false alarm test was conducted, wherein all images procurement during the seven-day testing period were analyzed with the device state recognizer. The recognition results were then compared with events exported from the home automation server. As no assault cases were injected into this test, all alarms triggered by the attack detector were subsequently classified as false alarms. As depicted in Table 4, a total of 20 false alarms manifested over the course of the week-long testing on two testbeds. This resulted in an average of 1.42 false alarms per day for each testbed. Out of these 20 false alarms, half were triggered by the electric heater H2 and the kettle K2.

6 RELATED WORKS

Security of Smart Home IoT.. Recent research extensively studies the potential vulnerabilities and corresponding defenses linked with smart home Internet of Things (IoT) systems [21, 22, 29, 35, 37, 55, 57, 58, 60, 61]. The rapid growth of the IoT industry has led to an influx of low-cost IoT devices riddled with various vulnerabilities [13, 39], which result in significant attack surfaces. Owing to the integration of smart home IoT devices within physical environments, these vulnerabilities allow attackers to extend the damage from the cyberspace to the physical environment. Ronen et al. demonstrated a method to compromise Philips Hue light bulbs in [50]. They exploited the proximity-based authentication scheme of the bulb and created a worm, which propagated through previously infected devices. Using this approach, thousands of light bulbs could be compromised within hours. In [35], Yan et al. detailed and analyzed the vulnerability of the MQTT protocol adopted frequently by smart home IoT devices. Through exploiting these vulnerabilities, attackers could execute large-scale attacks such as IoT remote control, data spoofing, and denial-of-service. Zhou et al. exposed implementation flaws in the IoT device pairing and binding protocol in [61]. The flaws enabled attackers to sever the connection between an IoT device and its associated cloud server and replace it with malicious phantom devices. This breach provided attackers an opportunity to perform denial-of-service attacks or forge counterfeit events targeting the automation cloud server.

Event Verification. Considering that IoT platforms welcome devices from various vendors and that any device, regardless of its relevance to sensitive attributes, can potentially impact other benign devices with events, developing robust IoT event verification systems becomes critical to prevent security risks instigated by event-based attacks. Current research primarily uses anomaly/outlier detection techniques, which identify unusual events from event logs exported from smart home automation platforms. HAWatcher [31] introduced the concept of inter-device correlation, combining IoT semantic information with conventional data mining procedures. Collecting correlations from the set training period allows IoT devices to verify each other and send alerts for any violated correlations. Peeves [16] presented an approach that uses readings from multiple sensors as evidence to confirm real-world event occurrences. The authors proposed a measure, termed relative mutual information (RMI), to calculate the relationship between an event and sensor reading fluctuations. The RMI from different sensors gets used collectively in a machine learning model to verify a sequence of smart home events. Ozmen et al. [46] further enhance these two EVS solutions by software patching and sensor placement inferencing techniques and prevent them from being circumvented by intelligent adversaries. DICE [24] and Aegis [53] implemented similar methods, involving a state transition graph or Markov chain model, to record transitions in states of smart home IoT devices during the training phase. Afterward, during deployment, any events triggering unseen transitions get reported as faulty or malicious. Compared to our IoTSentry, these methods face usability issues as they all require the assumption that IoT devices with various sensing capacities can be deployed densely enough so that inter-device correlation can be established. Furthermore, the accuracy of these 'indirect' event verification methods may drop due to interference from users' activities and physical environments. IoTSentry only assumes a security camera being installed and offers better detection accuracy.

HoMonit [60], another relevant piece of work, suggests using wireless traffic from IoT devices as side-channel information to verify IoT events. The authors develop a Deterministic Finite Automaton (DFA) to monitor the states of the smart home system, updated through events gathered from wireless traffic. Afterward, the states predicted by the DFA are compared to the outcome of smart apps (automation rules) to detect misbehaving smart apps. Although the authors claim they can detect 'spoofed events', they assume such events are produced by malicious smart apps, not real-world devices. In our threat model, compromised devices can generate spoofed events, which come with actual wireless traffic.

This method allows the attack to bypass HoMonit's check since it does not disrupt the DFA transition. Contrarily, our IoTSentry can effectively identify this attack by directly checking the events against their associated physical ground truth.

Side-channel information for IoT security. Many other works explore various ways to utilize different kinds of side-channel information in the IoT security domain. IOTCUPID [28] proposes to utilize the inter-event correlations to securely pair IoT devices with heterogeneous sensing modalities. Perceptio [33] was the first to introduce the concept of a physical security boundary, within which devices are considered to be co-located to capture sensory data from the same event. The fused data from these devices are utilized as a secure source from which to derive and deliver secure keys. T2Pair [40] suggests the use of universal operation sensing, which exploits the correlated vibration/accelerometer readings to capture the user's physical operations on two devices to be paired.

Aside from vibration and movement, other forms of side-channel information are used for IoT applications. The authors of [47] explore the potential to correlate the inherent acoustic noises of home appliances with their operational states. They investigate different classification methods capable of accurately classifying multiple appliances operating simultaneously. The resulting recognition aids the smart home system in identifying abnormal energy consumption and preventing hazardous status of devices. In concurrence with our work, IDIoT[51] also applies the vision side channel, with the aim to detect and identify wearable IoT devices for the elderly. The authors of this paper employ a 2D camera to capture the user's pose and match it with readings from the 3D inertial sensors embedded in the IoT device. Through the integration of visualized information with user motion measurements, the study claims to be capable of matching multiple IoT devices with their respective identities, even when they are covered by clothing or stored in a user's pocket.

7 DISCUSSION & FUTURE WORKS

Unapplicable situations. IoTSentry is not designed as a universal solution to verify events of all devices. There may be situations where events do not display explicit appearance changes or occur out of the camera's view. These situations may pose difficulties for humans in recognizing device states using only visual information. For instance, the August Smart Lock [6] displays minimal appearance changes during the rotation of the rounded handle. Although not effective in all scenarios, the vision channel utilized by IoTSentry remains the most comprehensive among all side channels, implying IoTSentry can cover most IoT devices. The acoustic channel could serve as a significant supplement to the vision channel to detect exceptions (e.g., smart lock rotation) with audio streams, which can also be picked up by home cameras. We aim to explore this possibility in future work.

Non-binary device states. This work primarily focuses on detecting spoofed events in devices with binary states. While quite a few IoT devices generate numeric values (e.g., illuminance, dimmer level, and air quality index), devices crucial for security and safety typically produce binary events (e.g., motion, lock, switch, and contact). This selection is consistent with other state-of-the-art event verification solutions [16, 31, 46, 49]. Moreover, IoTSentry, is also capable of detecting non-binary spoofed events, particularly when they markedly deviate from a device's actual states. In these instances, IoTSentry discerns state changes between "high" and "low" instead of "on" and "off", which is also adopted in [31].

Changes of camera's viewing angle and distance. The accuracy of device state recognition might be influenced if the test images are captured from varying angles and distances. Currently, we have not integrated these variations into our evaluation as most smart home cameras tend to be stationary. However, if movable cameras become commonplace in the future, we can account for this by incorporating images of devices taken from different viewing angles and distances into our training dataset. The current version of IoTSentry has demonstrated this ability by training DNN models with images under varied lighting conditions, yielding models that can adapt to a range of light conditions.

Situation of camera compromise. If the camera used as IoTSentry's input source is compromised, attackers could potentially circumvent the detection of event-based attacks by replaying historic images. This stratagem, however, greatly increases the complexity and cost of the attack as attackers would have to determine the location of the target device within the image and select images that consistently result in the same recognition states. This selection process could be time-consuming. In situations where multiple cameras are installed within a household, attackers would need to identify the specific camera used to verify events for their target device or compromise all cameras. Other state-of-the-art IoT event verification systems [16, 31, 46, 49] also need the similar assumption that the verifying device cannot be compromised at the same time. In contrast, IoTSentry incurs much lower cost than existing EVS solutions as it does not require IoT devices to be densely installed for forming relative mutual information associations or correlations and can achieve decent performance when very few IoT devices are installed.

8 CONCLUSION

This work explores the feasibility of verifying IoT device events through the visual channel. Utilizing cameras that are already extensively deployed, IoTSentry extracts rich semantic information, which is not commonly utilized in most existing works, and compares it with the received IoT events to detect event spoofing attacks. We designed a Siamese neural network and implemented a two-stage fine-tuning process to address the scarcity of training samples and significantly reduce the overhead of the model training. We constructed and implemented a prototype of IoTSentry, and evaluated it on two real-world smart home testbeds with 21 devices. Our experimental results reveal the high accuracy and low false alarm rate of IoTSentry. Our work implements the widely accepted conception of "seeing is believing" through AI, which markedly raises the bar for IoT event-based attacks.

ACKNOWLEDGEMENT

This work was supported in part by the United States National Science Foundation (NSF) under grants CNS-2204785, CNS-2205868, CNS-2309477, CNS-2310322, and CNS-2309550.

REFERENCES

[1] [n. d.]. *2023 Home Security Market Report.* (Accessed on 01/05/2024).
[2] [n. d.]. 64 Smart Home Statistics & Facts 2023. https://smarthomelady.com/smart-home-statistics/#google_vignette. (Accessed on 01/05/2024).
[3] [n. d.]. *North America Smart Home Security Cameras Market Report.* (Accessed on 01/05/2024).
[4] [n. d.]. Ring Always Home Cam | Indoor Flying Camera, Home Security Camera. https://ring.com/always-home-cam-flying-camera. (Accessed on 01/05/2024).
[5] 2015. Foscam IPCamera CGI User Guide. https://www.foscam.es/descarga/Foscam-IPCamera-CGI-User-Guide-AllPlatforms-2015.11.06.pdf.
[6] 2019. August lock no longer respond consistently to smartthings. https://community.smartthings.com/t/august-pro-no-longer-responding-consistently-to-smartthings/147965.
[7] 2019. Field of View Explained: Getting the Right Angle. https://www.brickhousesecurity.com/field-of-view-explained/.
[8] 2020. Samsung SmartThings, Add a little smartness to your things. https://www.smartthings.com/
[9] 2021. Alexa. https://developer.amazon.com/en-US/alexa.
[10] 2021. IFTTT Platform. https://ifttt.com/.
[11] 2021. Market Research Report: IoT. Fortune Business Insights. https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307
[12] 2021. Tuya Smart. https://www.tuya.com/.
[13] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. Sok: Security evaluation of home-based iot deployments. In *2019 IEEE symposium on security and privacy (sp)*. IEEE, 1362–1380.
[14] Amazon.com. 2021. Foscam Security Camera. https://www.amazon.com/Foscam-Storage-Dialogue-Connection-Security/dp/B07DJ5RSTM/ref=sr_1_3?dchild=1&keywords=foscam&qid=1624455167&sr=8-3.
[15] Akshaya Asokan. 2020. FBI Warns Of Swatting Attacks Targeting Smart Home Devices. https://www.bankinfosecurity.com/fbi-warns-swatting-attacks-targeting-smart-home-devices-a-15685/.
[16] Simon Birnbach, Simon Eberz, and Ivan Martinovic. 2019. Peeves: Physical Event Verification in Smart Homes. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1455–1467.
[17] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems* 6 (1993), 737–744.
[18] Z Berkay Celik, Gang Tan, and Patrick D McDaniel. 2019. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT.. In *NDSS*.
[19] Jonathan Chadwick. 2020. Smart plugs sold on Amazon, eBay and other popular online retailers are vulnerable to hackers and could start FIRES, consumer watchdog warns. https://www.dailymail.co.uk/sciencetech/article-8793311/Smart-plugs-open-hackers-start-fires-says.html.
[20] Haotian Chi, Chenglong Fu, Qiang Zeng, and Xiaojiang Du. 2022. Delay wreaks havoc on your smart home: delay-based automation interference attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 285–302.
[21] Haotian Chi, Qiang Zeng, and Xiaojiang Du. 2023. Detecting and Handling {IoT} Interaction Threats in {Multi-Platform}{Multi-Control-Channel} Smart Homes. In *32nd USENIX Security Symposium (USENIX Security 23)*. 1559–1576.
[22] Haotian Chi, Qiang Zeng, Xiaojiang Du, and Lannan Luo. 2021. PFirewall: Semantics-Aware Customizable Data Flow Control for Smart Home Privacy Protection. In *Network and Distributed Systems Security (NDSS) Symposium 2021*.
[23] Haotian Chi, Qiang Zeng, Xiaojiang Du, and Jiaping Yu. 2020. Cross-app interference threats in smart homes: Categorization, detection and handling. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 411–423.
[24] Jiwon Choi, Hayoung Jeoung, Jihun Kim, Youngjoo Ko, Wonup Jung, Hanjun Kim, and Jong Kim. 2018. 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)*. IEEE, 610–621.
[25] Koldo De Miguel, Alberto Brunete, Miguel Hernando, and Ernesto Gambao. 2017. Home camera-based fall detection system for the elderly. *Sensors* 17, 12 (2017), 2864.
[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
[27] Qifan Dong, Yang Yang, Wang Hongjun, and Xu Jian-Hua. 2015. Fall alarm and inactivity detection system design and implementation on raspberry pi. In *2015 17th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 382–386.
[28] Habiba Farrukh, Muslum Ozgur Ozmen, Faik Kerem Ors, and Z Berkay Celik. 2023. One key to rule them all: Secure group pairing for heterogeneous iot devices. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3026–3042.
[29] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *2016 IEEE symposium on security and privacy (SP)*. IEEE, 636–654.
[30] Chenglong Fu, Qiang Zeng, Haotian Chi, Xiaojiang Du, and Siva Likitha Valluru. 2022. Iot phantom-delay attacks: Demystifying and exploiting iot timeout behaviors. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 428–440.
[31] Chenglong Fu, Qiang Zeng, and Xiaojiang Du. 2021. HAWatcher: Semantics-Aware Anomaly Detection for Appified Smart Homes. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association.
[32] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.
[33] Jun Han, Albert Jin Chung, Manal Kumar Sinha, Madhumitha Harishankar, Shijia Pan, Hae Young Noh, Pei Zhang, and Patrick Tague. 2018. Do you feel what I hear? Enabling autonomous IoT device pairing using different sensor types. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 836–852.
[34] Mordor Intelligent. 2020. Smart Homes Market - Growth, Trends, and Forecasts. url = https://www.mordorintelligence.com/industry-reports/global-smart-homes-market-industry.
[35] Yan Jia, Luyi Xing, Yuhang Mao, Dongfang Zhao, XiaoFeng Wang, Shangru Zhao, and Yuqing Zhang. 2020. Burglars' IoT Paradise: Understanding and Mitigating Security Risks of General Messaging Protocols on IoT Clouds. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 465–481.
[36] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Zhuoqing Morley Mao, Atul Prakash, and SJ Unviersity. 2017. ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms.. In *NDSS*, Vol. 2. 2–2.
[37] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Z Morley Mao, Atul Prakash, and Shanghai JiaoTong Unviersity. 2017. ContexIoT: Towards providing contextual integrity to appified IoT platforms. In *Proceedings of The Network and Distributed System Security Symposium (NDSS)*.
[38] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-Keras: An Efficient Neural Architecture Search System. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1946–1956.
[39] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. 2019. All things considered: an analysis of IoT devices on home networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1169–1185.
[40] Xiaopeng Li, Qiang Zeng, Lannan Luo, and Tongbo Luo. 2020. T2Pair: Secure and Usable Pairing for Heterogeneous IoT Devices. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 309–323.
[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.
[42] Zhen Ling, Junzhou Luo, Yiling Xu, Chao Gao, Kui Wu, and Xinwen Fu. 2017. Security vulnerabilities of internet of things: A case study of the smart plug system. *IEEE Internet of Things Journal* 4, 6 (2017), 1899–1909.
[43] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.
[44] Sunil Manandhar, Kevin Moran, Kaushal Kafle, Ruhao Tang, Denys Poshyvanyk, and Adwait Nadkarni. 2020. Towards a natural perspective of smart homes for practical security and safety analyses. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 482–499.
[45] TJ OConnor, William Enck, and Bradley Reaves. 2019. Blinded and confused: uncovering systemic flaws in device telemetry for smart-home internet of things. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. 140–150.
[46] Muslum Ozgur Ozmen, Ruoyu Song, Habiba Farrukh, and Z Berkay Celik. 2023. Evasion attacks and defenses on smart home physical event verification. NDSS.
[47] Nilavra Pathak, Md Abdullah Al Hafiz Khan, and Nirmalya Roy. 2015. Acoustic based appliance state identifications for fine-grained energy analytics. In *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 63–70.
[48] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).
[49] Phillip Rieger, Marco Chilese, Reham Mohamed, Markus Miettinen, Hossein Fereidooni, and Ahmad-Reza Sadeghi. 2023. ARGUS: Context-Based Detection of Stealthy IoT Infiltration Attacks. *arXiv preprint arXiv:2302.07589* (2023).
[50] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. 2017. IoT goes nuclear: Creating a ZigBee chain reaction. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 195–212.
[51] Carlos Ruiz, Shijia Pan, Adeola Bannis, Ming-Po Chang, Hae Young Noh, and Pei Zhang. 2020. IDIoT: Towards ubiquitous identification of iot devices through visual and inertial orientation matching during human activity. In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 40–52.
[52] Brianne Sandorf. 2020. 5 Best Motion-Detector Cameras of 2021. https://www.reviews.org/home-security/best-motion-detector-cameras/.
[53] 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 Annual Computer Security Applications Conference*. 28–41.

[54] Arlo Smart. 2019. Arlo Smart Animal and Vehicle Detection. https://kb.arlo.com/000062115/Arlo-Smart-Animal-and-Vehicle-Detection.
[55] 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 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1439–1453.
[56] Daniel Wroclawski. 2019. Can a Burglar Really Jam Your Wireless Security System? https://www.consumerreports.org/diy-home-security-systems/can-burglar-jam-your-wireless-security-system/.
[57] Xuening Xu, Chenglong Fu, and Xiaojiang Du. 2023. MP-Mediator: Detecting and Handling the New Stealthy Delay Attacks on IoT Events and Commands. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. 46–62.
[58] Xuening Xu, Chenglong Fu, Xiaojiang Du, and E Paul Ratazzi. 2023. VoiceGuard: An Effective and Practical Approach for Detecting and Blocking Unauthorized Voice Commands to Smart Speakers. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 582–596.
[59] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2021. Scaling Vision Transformers. *arXiv preprint arXiv:2106.04560* (2021).
[60] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. 2018. HoMonit: Monitoring Smart Home Apps from Encrypted Traffic. In *ACM SIGSAC Conference on Computer & Communications Security (CCS)*. 1074–1088.
[61] Wei Zhou, Yan Jia, Yao Yao, Lipeng Zhu, Le Guan, Yuhang Mao, Peng Liu, and Yuqing Zhang. 2019. Discovering and understanding the security hazards in the interactions between IoT devices, mobile apps, and clouds on smart home platforms. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1133–1150.