Blinded and Confused: Uncovering Systemic Flaws in Device Telemetry for Smart-Home Internet of Things

TJ OConnor, William Enck, Bradley Reaves
{tjoconno,whenck,bgreaves}@ncsu.edu
North Carolina State University

ABSTRACT

The always-on, always-connected nature of smart home devices complicates Internet-of-Things (IoT) security and privacy. Unlike traditional hosts, IoT devices constantly send sensor, state, and heartbeat data to cloud-based servers. These data channels require reliable, routine communication, which is often at odds with an IoT device's storage and power constraints. Although recent efforts such as pervasive encryption have addressed protecting data in-transit, there remains little insight into designing mechanisms for protecting integrity and availability for always-connected devices. This paper seeks to better understand smart home device security by studying the vendor design decisions surrounding IoT telemetry messaging protocols, specifically, the behaviors taken when an IoT device loses connectivity. To understand this, we hypothesize and evaluate sensor blinding and state confusion attacks, measuring their effectiveness against an array of smart home IoT device types. Our analysis uncovers pervasive failure in designing telemetry that reports data to the cloud, and buffering that fails to properly cache undelivered data. We uncover that 22 of 24 studied devices suffer from critical design flaws that (1) enable attacks to transparently disrupt the reporting of device status alerts or (2) prevent the uploading of content integral to the device's core functionality. We conclude by considering the implications of these findings and offer directions for future defense. While the state of the art is rife with implementation flaws, there are several countermeasures IoT vendors could take to reduce their exposure to attacks of this nature.

CCS CONCEPTS

• **Security and privacy → Access control**; **Mobile and wireless security**; **Usability in security and privacy**.

ACM Reference Format:
TJ OConnor, William Enck, Bradley Reaves. 2019. Blinded and Confused: Uncovering Systemic Flaws in Device Telemetry for Smart-Home Internet of Things. In *12th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '19), May 15–17, 2019, Miami, FL, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3317549.3319724

WiSec '19, May 15–17, 2019, Miami, FL, USA

ACM ISBN 978-1-4503-6726-4/19/05…$15.00
https://doi.org/10.1145/3317549.3319724

1 INTRODUCTION

The rapid growth of smart-home IoT devices offers convenience, connecting us to a broad-array of sensors and actuators in our homes. The always-responsive nature of IoT provides on-demand access to seamlessly monitor and control every aspect of our homes. For example, smart-locks allow us to remotely schedule and control access to our homes from a smart phone, and connected doorbells can detect motion and send video push-notifications to our smart phones. The always-on, always-connected nature of smart home IoT devices also offers extensive forensic evidence for criminal investigations and legal proceedings. For example, data from Fitbit, Google Nest, Amazon Echo, and Ring Doorbell devices have aided law enforcement in solving crimes [16, 26]. However, the swift adoption of the IoT smart home market makes these devices, and the information they provide, vulnerable to attacks that compromise privacy and security [5, 30, 38, 45].

Most smart home IoT research focuses on protecting the confidentiality of the privacy-sensitive information they generate [6, 20, 31, 36]. While such efforts have improved the confidentiality of IoT devices, there remains a gap in availability and integrity of the information they provide. Recent high-profile examples have illustrated what happens when IoT availability fails, blinding or separating the user from the device. A flawed software update for NEST Thermostats caused a battery drain that deactivated devices, shutting off heat and leaving remote users disconnected [9, 29]. While the NEST failure unintentionally disrupted availability, an adversary can craft attacks to degrade availability of smart-home devices. A recent attack against the Amazon Key service demonstrated a crafted 802.11 de-authentication frame could knock a smart-lock offline, forcing the lock to remain unlocked and denying remote access to the user [17].

We hypothesize that the NEST and Amazon Key incidents are not isolated occurrences, but rather indicative of a larger systemic design flaw in many IoT devices. Specifically, designers falsely assume devices are always connected. By violating this assumption, attackers may blind devices and confuse their state by selectively suppressing device telemetry (i.e., data collected and transmitted to the cloud). Telemetry may be classified into channels for each source of data [13]. These channels carry data from two IoT subsystems: the *always-responsive* and the *on-demand* subsystems [28]. To enable an always-connected environment, devices rely on the always-responsive subsystem to present a continually connected view of devices. In contrast, the on-demand subsystem delivers sensor measurements to remote servers and implements actuator functionality. By selectively suppressing one or both of these channels, IoT device functionality can be completely disrupted.

Coarse-grained and fine-grained approaches differ in their ability to suppress device telemetry, and therefore, it is important to

characterize the practical implications. *Transparent* attacks provide no indication to the end user. *Permanent* attacks eliminate uploading buffered content after the attack. For example, coarse-grained suppression (i.e., jamming) can blind devices and confuse a device's state; however, it is largely ineffective at controlling user perception as it may trigger device alerts and in-app status changes. In contrast, selectively dropping on-demand packets can blind devices without an impact on user perception.

In this paper, we broadly characterize how smart home IoT devices address the availability and integrity of telemetry reporting. We analyze vulnerabilities in 24 popular consumer smart home devices that span a breadth of device types, including connected motion sensors, security cameras, doorbells, garage door openers, and locks. Our analysis identifies a systemic design flaw that enables sensor blinding and state confusion attacks. In each device we studied, we find instances of device implementation immaturity that allows an IoT device to be trivially disrupted.

This paper makes the following contributions:

- *We propose an attack methodology against telemetry protocols and buffering for smart home IoT devices.* Our attack is capable of blinding sensors and confusing the state of actuators.
- *We evaluate the susceptibility 24 popular consumer smart home devices.* We find that 22 suffer from design flaws that enable attacks including: (1) transparently disrupting the reporting of device status alerts, and (2) preventing the uploading of content integral to the device's core functionality.

Findings: We uncover critical design flaws that inform several broad findings. First, IoT sensor devices commonly isolate the telemetry for on-demand and always-responsive subsystems without co-mingling state and sensor knowledge, allowing an attacker to independently blind or suppress either subsystem. Second, battery constraints often cause vendors to eliminate the always-responsive subsystem by increasing or eliminating timeouts, allowing an attacker to blind the on-demand subsystem. Third, smart home actuator devices commonly fail to repudiate the delivery of state change messages, exposing the devices to state confusion attacks. Fourth, immature IoT implementations fail to buffer sensor measurements and state changes despite their computational capacity to do so.

Organization: Section 2 provides background, motivates our work, and details the adversary threat model. In Section 3, we expand the details of sensor blinding and state confusion attacks. In Section 4, we uncover the design flaws from a broad array of devices and summarize our findings in a list of recommendation. Section 5 offers insight for defense. Section 6 discusses related work. Section 7 summarizes our conclusions.

2 BACKGROUND & MOTIVATION

The offline actions and message telemetry protocols for smart home devices are fundamentally different. Vendors distinguish themselves by implementing a variety of strategies for telemetry protocols and offline content buffering and state management. For example, the Amazon Echo is designed to exist always-connected to the cloud. The automated speech recognition and natural language processing algorithms for the device are exclusively processed server-side [2]. In contrast, Samsung's SmartThings framework provides a combination of cloud and local storage and processing with the introduction of a local controller [15]. The SmartThings controller (i.e., hub) offers optional processing, memory, and storage not available at the limited device level (e.g., a water leak sensor).

Buffering strategies on devices also vary. The CleverLoop Security Camera uses a machine learning algorithm to filter out unimportant movements and record only essential videos, locally storing seven rolling days of video alerts [18]. Once reconnected, the CleverLoop camera uploads up to 8GB of data to cloud-based severs. In contrast, the WyzeCam Camera provides optional SD-Card storage for the device, storing offline video and motion events only locally on the card. Poor design decisions for offline buffering and telemetry introduce two unique attack vectors: *sensor blinding* and *state confusion*. The following section reviews the telemetry of IoT devices to provide necessary context for these attack vectors.

2.1 Overview of IoT Telemetry

IoT devices consist of two subsystems: *always-responsive* and *on-demand* [28]. The always-responsive subsystem maintains a perpetual connection to remote servers to report the availability of the device and listen for server-side instructions. In turn, the servers use low-bandwidth messages to monitor connectivity health. We label this message exchange *heartbeats*, since they periodically indicate the connectivity health of a device. When a timeout expires without receiving any heartbeats, servers mark the device as offline and present the user with a smart phone alert. In our experiments, we measured the timeout period as brief as forty seconds and as long as thirty minutes. Further, some battery constrained devices entirely eliminate the always-responsive subsystem due to the power constraints of periodic messaging.

Conversely, the on-demand subsystem delivers environmental sensor measurements and actuator state changes to remote servers. As an example, a security camera may send an initial motion sensor notification before uploading a video recording. We describe the initial low-bandwidth messages as on-demand *event notifications*, since they are prioritized notifications of a triggered on-demand event. We label the video recording as on-demand *content messages*, since they are often high-bandwidth messages carrying the content of a triggered event. Event notifications and content messages are the primary channels for sensor reporting. Actuators commonly use only event notifications to report state changes.

As depicted in Figure 1, devices may establish independent channels for heartbeats, event notifications, and content messages. Channel independence often occurs by network flow division, separating channels among different protocols, transport layer ports, and servers. In contrast, some devices create channel separation through time division by sharing time slots on the same network flow. In this case, a device may interleave event notification and content messages between periodic heartbeats. Attacking *flow division* and *time division* channels require unique approaches, described in Section 3.

Sensor Blinding: Sensor blinding attacks the integrity and availability of sensor devices by preventing the delivery of sensory measurements to always-connected IoT servers. Sensor-based IoT

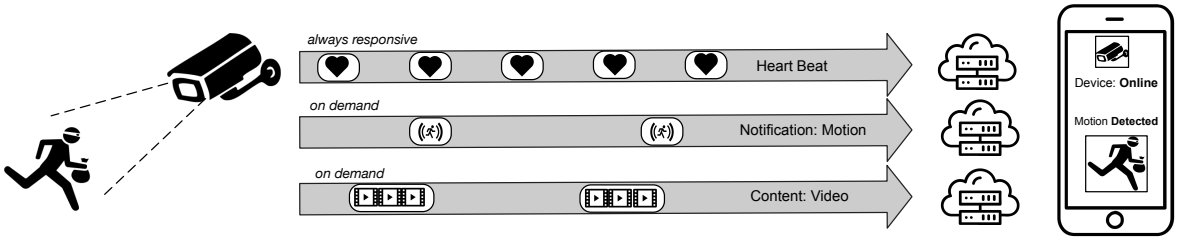


Figure 1: The systemic isolation of connectivity (e.g., heart beat), event notification, and content (e.g., video recording) channels for IoT devices present a blinding vulnerability when a channel is independently suppressed.

devices summarize raw sensory measurements (e.g., motion detection, video recording, auditory, water detection, or other environmental sensors) and report this information. These devices rely on uninterrupted and untampered delivery of messages. However, we discovered in our experiments that devices commonly fail to ensure server-side delivery of sensory measurements. As an example, our evaluation identified the Ring Video Doorbell did not buffer any of the historical events that occurred while the device lacked connectivity. An attacker can exploit this design flaw to blind the device's notification mechanism, which reports motion detection. While our work focuses on smart home devices, contemporary research has shown weaknesses in protection mechanisms for connected sensors by maliciously spoofing sensor reporting, leading to isolated attacks against the US Department of Vehicles [12] and agricultural cyber-physical systems [39].

State Confusion: State confusion attacks the integrity of actuator devices' state reported to always-connected IoT servers. State confusion is a special subset of sensor blinding attacks, where the sensor blinding impacts actuators. Actuators implement mechanical movement and control, changing the physical state of a device. Disrupting connectivity for actuator devices can pin the device in a fixed state, which can have disastrous consequences for actuator devices (e.g., smart-locks, garage doors, sprinkler systems, or smart-outlets). For example, the Amazon Key service attack demonstrated that a simple 802.11 de-authentication frame could knock a connected smart-lock offline, pinning the device in the current unlocked state [17]. Once disconnected from the wireless network, the device remained indefinitely in the unlock state. Related research has shown the ability to influence a drone's state by GPS spoofing, followed by jamming the control channel, forcing the drone to crash while in the spoofed state [21].

2.2 Threat Model

There are many reasons why an attacker may wish to blind sensors or confuse states of smart home IoT devices. Many smart home devices are used to provide physical security. Therefore, criminals may use sensor blinding and state confusion of actuators to gain physical access to a home without creating digital forensic evidence. Criminals may attack homes opportunistically by themselves, or they may be part of organized crime, which may obtain access to tens of thousands of wireless routers. Outside of this traditional threat model, the attacker may also be the perpetrator of domestic abuse. Recent popular media articles [10] have reported instances of domestic abuse that involves stalking using smart home devices. In addition to these attacks, a domestic partner may blind sensors to eliminate forensic evidence or confuse states of actuators to retain physical access to areas.

2.2.1 Attacker Goals. We consider an attacker whose goal is to disable the core functionality of a device by blinding a sensor or confusing the state of an actuator. As an illustration, we imagine an attacker that has the goal of circumventing a security camera without triggering an alert or being recorded, or preventing a smart-lock from securing a residence. Ultimately the success of the attacker relies on two variables: the ability to avoid producing alerts (a transparent attack) and the ability to avoid leaving a record (a permanent attack).

2.2.2 Attacker Capabilities. An attacker may execute a coarse-grained approach of physical layer suppression (e.g, jamming). Alternatively, the attacker may have or can gain access to the wireless router to implement a fine-grained network layer suppression. We consider the following two attacker capabilities:

Physical Layer Suppression: A physically located attacker can broadly jamming the wireless signal, or exploit flaws in the 802.11 networking scheme including forging un-encrypted de-authentication frames or flooding carrier-sense management frames. Such an adversary can jam traffic at the physical layer but will not achieve their objective of a stealthy attack. Jamming suppresses all device traffic and is a coarse-grained approach that will provide the user with an indication of the attack.

Local Network Layer Suppression: An attacker with control over the wireless router has fine-grained ability to selectively suppress packets. To compromise a wireless router, an attacker may gain access through several different attack vectors [40–44]. The attack may be mounted remotely from the Internet, or locally from a compromised or malicious device on the LAN. Alternatively, the attack may have a pre-existing relationship with the victim and been given administrative access to the router. This fine-grained access permits selectively suppressing on-demand traffic without adversely affecting user perception of device availability. Once controlling the router, the attacker can control traffic through a variety of techniques: 1) remotely proxying device traffic, 2) local establishment of firewall policies, or 3) connecting the router to a criminally-run remote security orchestrator that selectively suppresses on-demand security camera traffic.

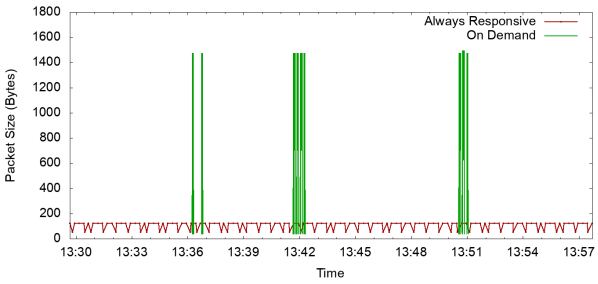


Figure 2: The simplistic nature of IoT devices facilitates subsystem classification at the packet layer, enabling an attacker to isolate on-demand events.

2.2.3 Attacker Assumptions. Our approach relies on two key assumptions: 1) the attacker can fingerprint a device and 2) the attacker understands the telemetry model for the target device.

Device Fingerprinting: We assume the attacker can fingerprint a device. This is a reasonable assumption as several recent works have demonstrated highly accurate device fingerprinting with nominal traffic [1, 5, 8, 27]. Further, devices can be identified using IoT reconnaissance tools such as IoTScanner [35] or intercepting medium access control layer headers.

Telemetry Models: We assume the attacker can learn the telemetry model for a specific device. The attack model is generic as it applies across device types, vendors and models. However, the attacker requires knowledge of specific device telemetry and messaging protocols to succeed. As smart home devices are commercial in nature, an attacker can procure and study the telemetry protocol for any device in preparation for an attack.

3 METHODOLOGY

In this section, we discuss the different design features that enable attacks against IoT telemetry by considering telemetry division, message timeouts, and buffering decisions. Further, we offer insight for understanding attack severity by discussing attack transparency and permanence.

3.1 Attacking Flow Division Telemetry

Flow division telemetry, depicted in Figure 1, separates heartbeats, notifications, and content across distinct network flows. To illustrate this isolation, consider the *Tend Secure Lynx Indoor Security Camera*, a connected surveillance camera which supports motion detection and high definition video. In our observations, we determined the camera sent heartbeats to an AWS server using the TinyMessage protocol (TCP port 5104) and separately delivered notification and content messages to AWS servers over SSL. Isolating the always-responsive and on-demand subsection facilitates ideal conditions for sensor blinding and state confusion attacks by enabling the attacker to easily identify and blind the on-demand subsystem. An attack can benefit from the simplistic nature of IoT to classify the data intensive on-demand subsystem and the periodic always-responsive subsystem. In the case of the Lynx Camera, Figure 2 depicts three separate on-demand video recordings that peak from the routine low-bandwidth heartbeat messages.

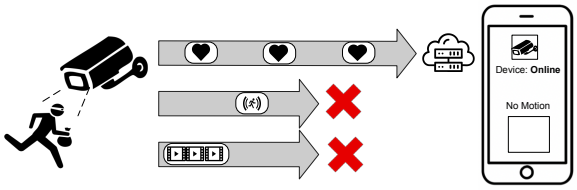


Figure 3: On storage constrained IoT devices, the lack of buffering event notifications and content in embedded channels presents a blinding vulnerability when individual packets bearing notification or content are suppressed.

Transparent Blinding: The design flaw of incorrectly separating the privilege of each subsystem enables our attack. In the case of flow division telemetry, devices rarely co-mingle the functionality of the always-responsive and on-demand subsystems. Specifically, a heartbeat is solely responsible for reporting connectivity health and does not advertise on-demand events. Conversely, on-demand messages (notifications, and content) have no impact on connectivity health. By their nature, on-demand subsystem messages are often delivered without prior notification. Thus, an attacker can transparently suppress the on-demand flows, creating the conditions for sensor blinding and state confusion attacks. In the naive case of the Lynx Camera, a firewall rule blocking outbound SSL traffic will blind the camera's on-demand subsystem. With the firewall rule applied, an owner's companion app will report the camera as online; however, the device will silently fail to report any motion detection or video recordings. We label this attack as *transparent blinding*, as the user is unaware the camera system is failing to report on-demand events. Figure 3 depicts this transparent attack.

Semi-Transparent Blinding: A partial disruption of the always-responsive subsystem often results in only *semi-transparent blinding*. In this case, the companion app may display intermittent connectivity but avoids overwhelming the user with alerts. In select cases, an adversary is unable to suppress the on-demand subsystem without at least partially disrupting the always-responsive subsystem. This is common in the case of flow-division telemetry, as described in the following subsection. A full disruption of the always-responsive subsystem commonly results in a smart phone alert that informs the user of the disruption. However, a partial disruption often results in only an in-app status message. Thus, the user is only aware of the blinding if they have the companion app open during the attack. In our experiments, we found that several devices were vulnerable to semi-transparent blinding for sufficiently long periods of time. As an example, we found the Iris Hub displayed an intermittent connectivity message inside the companion app after a minute of sensor blinding but waited thirty minutes to deliver a smart phone alert. In our experiments, we also discovered that devices did not aggregate blinded time. To illustrate this flaw, we could blind the Iris hub for 29 minutes, permit 1 minute and then repeat without ever triggering an alert.

Permanent Event Blinding: Event notification and content messages are frequently discarded with on-demand connection failures. The unique storage and processing constraints of IoT enables this problem. In our experiments, we observed all device types rarely

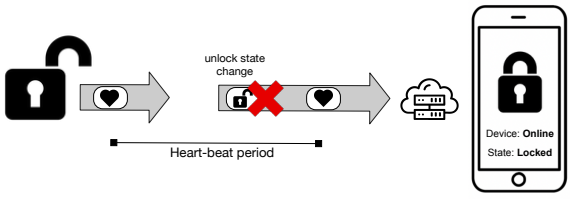


Figure 4: On battery constrained IoT devices, lengthy timeouts between heartbeats present an opportunity to suppress unbuffered state changes of actuators.

buffer on-demand events (even at hubs, whose purpose is to provide auxiliary storage). As a running example, the Lynx Camera discarded video recordings while our firewall rule was established. After removing the firewall rule, the camera only uploaded newly detected on-demand events. A combination of factors including the processing power required for data transformation, limited local buffer, and low-power mode/sleep requirements may contribute to this design decision [13]. However, the device is *permanently blinded* when the device fails to buffer and deliver suppressed on-demand events. The design decision of not buffering failed events enables an adversary to selectively suppress the on-demand system for a brief period, resulting in the device permanently failing to ever report the suppressed events. As we discuss in Section 4, an attacker can also leverage wireless denial-of-service attacks to permanently blind wireless devices by forging de-authentication frames. While several systems support auxiliary storage (e.g., secure digital memory cards) to buffer events, an attacker can remove or physically alter this storage while the device is blinded.

3.2 Attacking Time Division Telemetry

Attacking time division telemetry requires a more fine-grained approach that selectively suppresses individual packets instead of network flows. An attacker must permit always-responsive packets while suppressing on-demand packets to achieve the goal of transparently and permanently blinding devices. This is trivial for connection-less transport layer protocols which fail to maintain state or guarantee delivery. However, most devices rely on SSL, which uses connection-oriented TCP, guaranteeing in-order delivery. To address this challenge, an adversary can benefit from two key design failures: the lack of on-demand event buffering (as previously introduced) and lengthy timeout periods. In the following paragraphs, we address how an attacker can leverage these design failures to attack time division telemetry.

Packet Signatures: Time division telemetry attacks require identifying packets from the on-demand subsystem. The attacker must be able to distinguish between always-responsive packets and on-demand packets. An attacker can benefit from the simplistic nature of IoT to strengthen the classification accuracy. Apthorpe et al. [5] previously demonstrated the ability to infer privacy-sensitive, on-demand activities from analyzing traffic rates of IP traffic. Further, Celosia et al. [11] observed that artifacts of Bluetooth L2CAP layer could be used to determine a device state change. In our experiments, we discovered this classification holds true for smart home devices at the packet level.

We observed that we could identify packets based on the consistent nature of the always-responsive subsystem and urgent nature of the on-demand subsystem. To illustrate this, consider the case of the Momentum Axel security camera which supports motion detection, audio detection and video recording over time divided telemetry. The camera's always-on-subsystem sends fixed size 52, 60, or 108-byte heartbeats while the on-demand subsystem sends 617 or 761 byte notification packets (depending on if audio or motion triggered the event), and 1500-byte content packets. Transport layer options can strengthen classification of on-demand activities. We discovered several on-demand packets from several devices (including from the Momentum Camera) set the TCP PUSH flag in order to promptly forward and deliver data. The transport layer uses the TCP PSH Flag to push data out of a TCP socket immediately, rather than waiting for additional data to fill the buffer [19]. Finally, we discovered the always-responsive subsystem sends packets on a consistent and predictive interval. In contrast, the on-demand subsystem interleaves packets between periodic heartbeats.

Buffer Failures: Storage and processing constrained IoT devices often fail to buffer suppressed on-demand content and notifications. In our observations, the lack of acknowledgments for suppressed packets terminates connection-oriented protocols. However, the always-connected nature of IoT immediately establishes a new connection. In design, this new connection should carry the suppressed on-demand events. However, the new connection often began with always-responsive heartbeats instead of the suppressed on-demand content. We observed several cases of motion detectors, security cameras, and even smart hubs simply discarding the suppressed on-demand content and notifications. To amplify the severity of the attack, we often observed a lack of user alerts since the always-responsive subsystem succeeded in delivering a heartbeat within timeout windows. Figure 4 illustrates this attack scenario. When the adversary suppresses an on-demand state change, the IoT device simply discards the change and establishes a new connection. The next section expands upon our empirical evaluation and the findings and results for 24 popular smart-home devices.

4 RESULTS

In this section, we describe the experimental setup for evaluating the vulnerability of 24 popular smart-home devices to sensor blinding and state confusion attacks. We intentionally chose a simple setup to ease reproducibility and provide all flow modifications to repeat the experiments used in our evaluation. We conclude this section by summarizing a list of our findings that highlight key design failures that contribute to sensor blinding and state confusion attacks against our data-set of devices.

4.1 Experiment Setup

We set up a laboratory smart home environment to examine the scope and severity of attacks against a broad array of devices. In our threat model, the adversary's objective is to blind sensors and confuse the state of actuators. The severity of the attack relies on the adversary's transparency and permanence. For each device in our environment, we measured the impact of network layer and physical layer suppression of traffic for a thirty-minute window.

Table 1: To demonstrate the broad scope of the problem, we have profiled 24* popular smart home IoT devices and measured the effectiveness of sensor blinding and state confusion attacks by analyzing their transparency and permanence for these devices. Our results indicate systemic design flaws contributed to attacks against all devices classes.

Device	App Downloads	Firmware Version	Attack Type	Network Layer Suppression		Physical Layer Suppression	
				Transparent Attack	Permanent Attack	Transparent Attack	Permanent Attack
D-Link DCH-S150 Motion Sensor	100,000+	1.23	Sensor Blinding	●	○	◑[2]	○
Belkin F7C028 Motion Sensor	500,000+	2.00.11057	Sensor Blinding	◑[2]	●	◑[2]	●
Wasserstein Motion Sensor	5,000+	1.10	Sensor Blinding	●	●	●	●
iView S200 Motion Sensor	5,000+	1.10	Sensor Blinding	●	●	●	●
TuyaSmart M01 Motion Sensor	5,000+	1.10	Sensor Blinding	●	●	●	●
D-Link DCS-8010LH Camera	100,000+	1.02.02	Sensor Blinding	○	○	○	●
Momentum MOCAM-720-01 Camera	100,000+	5.1.8	Sensor Blinding	◑[3]	○	◑[3]	◑[4]
Merkury MI-CW007-199W Camera	10,000+	2.0.9	Sensor Blinding	●	●	◑[2]	●
Geenie CN-CW003 Camera	100,000+	1.10.16	Sensor Blinding	●	●	◑[2]	●
Tend Secure Lynx Indoor 2 Camera	50,000+	00.15.003	Sensor Blinding	●	●	◑	●
Wyze V1 Camera	100,000+	3.9.3.72	Sensor Blinding	●	◑[4]	◑[8]	◑[4]
Wyze V2 Camera	100,000+	4.9.3.64	Sensor Blinding	●	◑[4]	◑[8]	◑[4]
Canary 1 Security Camera	100,000+	4.0.0	Sensor Blinding	◑[5]	○	◑[5]	●
Iris Door Contact Sensor	100,000+	2.2.0.009	Sensor Blinding	◑[2]	●	–	–
Iris Motion Sensor	100,000+	2.2.0.009	Sensor Blinding	◑[2]	●	–	–
Iris Water Sensor	100,000+	2.2.0.009	Sensor Blinding	○[6]	○	–	–
SmartThings Contact Sensor	100,000,000+	000.024.00022	Sensor Blinding	●	●	–	–
SmartThings Motion Sensor	100,000,000+	000.024.00022	Sensor Blinding	●	●	–	–
SmartThings Water Sensor	100,000,000+	000.024.00022	Sensor Blinding	●	●	–	–
SmartThings Button	100,000,000+	000.024.00022	Sensor Blinding	●	●	–	–
Ring Pro Doorbell (Wired)	1,000,000+	Up to Date[1]	Sensor Blinding	◑[2]	●	◑[2]	●
Ring Doorbell (Battery)	1,000,000+	Up to Date[1]	Sensor Blinding	●	●	●	●
MyQ Garage Door Opener	500,000+	Up to Date[1]	State Confusion	◑[2]	◑[7]	◑[2]	◑[7]
Schlage Deadbolt (w/ Iris Hub)	100,000+	2.2.0.009	State Confusion	◑[2]	●	–	–
Schlage Deadbolt (w/ SmartThings Hub)	100,000,000+	000.024.00022	State Confusion	●	○	–	–
Total				15/25	16/25	4/15	11/15

● Attack succeeded. ◑Attack partially succeeded (i.e., semi-transparent blinding and/or partial event buffering). ○Attack failed (i.e., alerted smart phone and/or buffered all offline events).

* Our data-set consists of 24 total devices and 25 configurations since the Schlage Deadbolt is configured with both the Iris and SmartThings Hubs.

[1] Device only reports *firmware up to date*; does not report firmware version number.
[2] Smart phone app displayed offline status; smart phone did not alert user.
[3] Smart phone app displayed online; smart phone alerted user.
[4] Device buffered a single motion detection event.
[5] Smart phone only alerts offline when app opened.
[6] Hub alerts water leak via audible alarm; smart phone did not alert user.
[7] Smart phone app is only confused for 5 minutes.
[8] Clicking a device in the smart phone app results in an error message.

Tested Devices: Our evaluation consisted of a representative set of 24 smart home IoT devices for consumers, available during 2018 from well-known US retailers, including Walmart, Lowe's, Target and Amazon. These devices covered IoT classes related to security cameras, motion sensors, smart home environmental monitoring, connected doorbells, garage door openers and smart-locks. Most devices connected directly through Wi-Fi to our network. In the case of smart-hubs, they were connected to the network via Ethernet and connected to their sensors and actuators via ZigBee or Z-Wave. We did not consider suppressing ZigBee or Z-Wave traffic but instead focused on selectively suppressing traffic that the smart-hub generated to cloud servers. For each device, we downloaded the iPhone companion app to manage the device. As an indication of the popularity of each device, we recorded the number of application downloads for the Android app version on the Google Play Store. (The Apple App Store does not release app download metrics). Note, the Geenie and Merkury devices use the same companion app. Further, the Iris and SmartThings sensors are managed by their respective singular apps. All other apps are single purpose apps. An overview of the tested devices is shown in Table 1.

Labeling Results: Our evaluation considered the ability to blind a device's sensors and confuse the state (if the device maintained state). We used full (●), half-filled (◑) and empty (○) circles to label the severity of the transparency or permanence of the attack. Under the *transparent* column, a full circle represents the attack succeeded without an in-app status change or smart phone alert. A half-filled circle indicates semi-transparent blinding (e.g., the companion app displayed an offline status message but failed to provide a smart phone alert). Under the *permanent* column, a full circle indicates that the sensor behavior or state change is never reported after the attack. A half-filled circle indicates that the attack buffered a period but not all of the attack (e.g., a camera system buffered the most recent event) or that the state confusion lasted only a brief period (e.g., the companion app for a lock remained confused for five minutes after the attack).

Table 2: Packet Signatures for Eliminating the On-Demand Subsystem of Devices

Device	On-Demand Packet Signature
D-Link Motion Sensor	PROTOCOL == SSL AND PAYLOAD LENGTH > 475 BYTES
Belkin Motion Sensor	PROTOCOL == TCP AND (DST PORT == 8443 OR DST PORT == 3478)
Wasserstein Motion Sensor	PROTOCOL == MQTT AND PAYLOAD CONTAINS "out"
iView Motion Sensor	PROTOCOL == MQTT AND PAYLOAD CONTAINS "out"
TuyaSmart Motion Sensor	PROTOCOL == MQTT AND PAYLOAD CONTAINS "out"
Canary Security Camera	PROTOCOL == SSL AND PACKET LENGTH == 1500 BYTES
D-Link Camera	PROTOCOL == SSL AND PSH/ACK SET and PAYLOAD LENGTH > 600 BYTES
Momentum Camera	PROTOCOL == SSL AND PSH/ACK SET and PAYLOAD LENGTH > 600 BYTES
Merkury Camera	PROTOCOL == MQTT AND PAYLOAD CONTAINS "smart/device/out"
Wyze Camera	PROTOCOL == TCP AND DPORT == 8443
Tend Secure Camera	PROTOCOL == SSL
Geenie Camera	PROTOCOL == MQTT AND PAYLOAD CONTAINS "smart/device/out"
Iris Hub	PROTOCOL == SSL AND PSH/ACK FLAGS AND PAYLOAD LENGTH > 250 BYTES
SmartThings Hub	PROTOCOL == SSL AND PAYLOAD LENGTH > 359 BYTES
Ring Pro DoorBell	PROTOCOL == SSL OR TCP DST PORT == 15063 OR 9999
Ring Doorbell	PROTOCOL == TCP AND DPORT == 80
MyQ Garage Door Opener	PROTOCOL == TCP AND PAYLOAD LENGTH == 125 BYTES

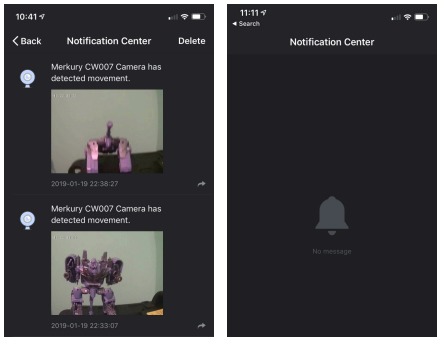
Network Layer Suppression: To suppress traffic with the fine-grained approach described in Section 3, we configured a consumer grade wireless router as a software defined switch and connected it to a local software defined controller. Our controller logic implemented OpenFlow flow modifications that matched the appropriate header fields that correlated to on-demand events for each device. For reproducibility, we have included the matching fields in Table 2. During the thirty-minute attack window, we stimulated on-demand activities for each device.

Physical Layer Suppression: To determine the effect of physical layer suppressed, we implemented a coarse grained denial-of-service attack by exploiting a vulnerability of the data-link layer. This approach suppressed both the always-responsive and on-demand subsystems. This attack replicated a local criminal from our threat model in Section 2. For each device, we forged 802.11 de-authentication frames using the aireplay-ng packet injection tool [14]. These forged frames disrupted the wireless connectivity of the devices, eliminating both subsystems. We recognize there are more subtle approaches to reactive jamming, that could selectively suppress the on-demand subsystem, but leave these approaches for future work [46].

4.2 Evaluation Results

Table 1 summarizes the results of our evaluation. Our results demonstrate that 22 of 24 devices suffer from critical design flaws that enable attacks to transparently disrupt the reporting of device status alerts or prevent the uploading of content integral to the device's core functionality. These results confirm our hypothesis that the attacks against NEST and Amazon Key are not isolated instances. IoT developers falsely assume devices are always connected and are unprepared when that assumption is violated. In our data-set, 15 devices fail to provide the user any indication they are under an attack and 16 devices fail to buffer any content for the duration of the attack. An additional two devices fail to buffer any content when the physical channel is eliminated by jamming.

We find that our proposed attack vector, selectively suppressing network layer traffic, offers stealth over physical layer suppression. To this end, we demonstrate a reliable methodology for blinding security cameras and motion sensors that relies on attacking flow division telemetry. Our attack methodology, purposely simple and easily replicated, has significant ramifications about the forensic value of the data these always-connected devices produce. Ultimately, we uncover immature IoT implementations that fail to implement telemetry protocols and buffer sensor measurements and state changes despite their computational capacity to do so.



(A) Device functioning correctly (B) Device sensor blinded

Figure 5: The design decision to isolate telemetry channels into separate flows that do not co-mingle state and sensor knowledge leads to blinding and confusion vulnerabilities. In the figure above, the Merkury camera is functioning properly in (a) and blinded in (b).

In the following subsection, we summarize these findings by examining the problematic designs of telemetry, battery conservation, controller storage, controller prioritization, and buffering during traffic suppression.

4.3 Evaluation Findings

Finding 1: ***Flow division telemetry commonly isolates the on-demand and always-responsive subsystems.*** The design decision to isolate telemetry channels into separate flows, without co-mingling state and sensor knowledge, facilitates sensor blinding vulnerabilities. We successfully blinded the availability of the on-demand subsystem in several devices in our evaluation data-set: including the Ring Doorbell, the Merkury, Wyze, Lynx, and Geenie security cameras and several motion sensor devices.

Figure 5 illustrates the results of sensor blinding the Merkury security camera. In the case of this example, we stimulated the on-demand motion sensor by placing an object in front of the camera system. We passively observed that the camera system sent the motion notification over a plain-text MQTT connection and sent heartbeats and video content over SSL. We identified the traffic from each subsystems by correlating the packet timings to the history in the companion app. We then repeated the experiment suppressing only the on-demand MQTT flow. When triggered, the camera attempted to initiate the MQTT connection but our flow modifications suppressed the corresponding packets. Despite the device uploading a video over the SSL connection, the companion app displayed a blank event history. Although the attack eliminated the on-demand channel from delivering motion notifications, the companion app failed to alert the user. After eliminating the flow modification, the camera system returned to full functionality but failed to subsequently upload the buffered motion notifications.

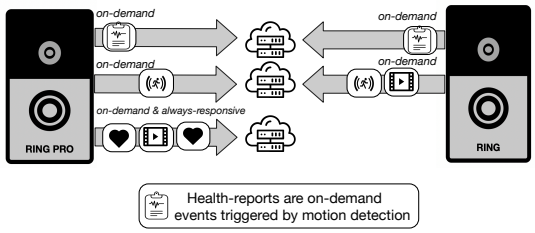


Figure 6: The standard Ring uses on-demand *health reports*, triggered via motion, to report the health of the device. The design decision to conserve battery by eliminating the always-responsive subsystem facilitates transparently blinding the standard Ring over the Ring Pro.

Finding 2: *Battery constraints often eliminate the always responsive subsystem*. Battery constraints present a key design decision in the integrity of the always-responsive subsystem. In our experiments, we determined vendors often eliminate the always-responsive subsystem by increasing or eliminating timeouts. This trade-off conserves battery power at the cost of device responsiveness. Table 3 enumerates the heartbeat intervals and timeouts for all devices to provide an understanding the trade-off made by different devices. To specifically illustrate the battery paradigm, we compared the evaluation of the professional and standard version of the Ring Doorbell. The professional version is connected to low voltage power while the standard version is battery-powered. The doorbells offer similar functionality. However, the differing implementations of the always-responsive and on-demand subsystems facilitate transparent blinding against the standard version.

Figure 6 depicts the differing channels for the flow-division telemetry. In our evaluation, both devices transmitted a *health report* (carrying metrics about device power and signal strength) over SSL. Also, both delivered on-demand content and notifications to remote servers. The standard version used HTTP-over-TLS for content and notification, the professional version delivered notifications over TCP Port 15063 and content separately. The key distinction is the always responsive subsystem, which enables streaming video. The professional version, which supports streaming video by default, transmitted periodic heartbeats over SSL. In contrast, the standard version did not send any periodic heartbeats and lacked streaming video in the default configuration. Instead of the heart-beats, the standard version uses the health reports to determine device health and connectivity. This problematic decision is amplified by the fact that the health reports are only delivered on-demand with motion sense notifications. Thus, the server does not anticipate them on any interval and they cannot provide ground truth about the device.

To blind the motion sensors of both devices, we enabled flow modifications that dropped notification and content packets. During the period of our flow modifications, our attack blinded both devices by preventing the delivery of on-demand notifications and content for motion-triggered events. Further, nether device buffered the events. However, we ran into a challenge transparently blinding the professional version. On the professional version, we could not distinguish between packets originating from the always responsive subsystem (heartbeats) and the on demand subsystem (content) since they were encapsulated in the same network flow. However, this presented no issue for the standard version since it lacked an always responsive subsystem. Thus, our flow transparently blinded the standard version. During the period of the attack, the companion app reported the standard Ring doorbell as online and failed to sense or report any motion detection events. We also found similar vulnerabilities in the Wasserstein, iView, and Tuya battery-powered systems that eliminated the always-responsive subsystem.

Table 3: Lengthy heart-beat time periods facilitate traffic suppression and blinding attacks without alerts.

Device	Telemetry	Protocols	HeartBeat (s)
D-Link Motion Sensor	Time Division	SSL	5
Belkin Motion Sensor	Flow Division	STUN, SSL, MQTT	270
Wasserstein Motion Sensor	Flow Division	MQTT, HTTP	—
iView Motion Sensor	Flow Division	MQTT, HTTP	—
TuyaSmart Motion Sensor	Flow Division	MQTT, HTTP	—
Canary Security Camera	Time Division	SSL	30
DLink Camera	Time Division	SSL	55
Momentum Camera	Time Division	SSL	30
Merkury Camera	Flow Division	SSL, MQTT, UDP	120
Wyze V1 Camera	Flow Division	SSL, MQTT-S, UDP(1001)	5
Wyze V2 Camera	Flow Division	SSL, MQTT-S, UDP(1001)	5
Tend Secure Camera	Flow Division	SSL,TinyMsg	15
Geenie Camera	Flow Division	MQTT, UDP	100
Iris Hub	Time Division	SSL	5
SmartThings Hub	Time Division	SSL	30
Ring Pro DoorBell	Flow Division	H264, SSL, RTP, TCP(9999)	30
Ring Doorbell	Flow Division	H264, SSL, RTP	—
MyQ Garage Door Opener	Time Division	MQTT-S	10

Finding 3: *Flawed controller designs introduce device layer vulnerabilities.* To illustrate state confusion attacks, we analyzed the different methods the Iris and SmartThings controllers handled suppressed telemetry for the Schlage Deadbolt. We included the Schlage touchscreen deadbolt in our data-set as it offers Z-Wave connectivity and supports different controllers including SmartThings, Iris, Alexa, and Wink. By pairing the deadbolt with a controller, a user can remotely control the state of the deadbolt using the controller's companion app. In our analysis, we discovered the design of the Iris controller permitted suppressing state reporting to the companion app.

Further, we identified the two controllers distinctly handled suppressed telemetry, leading to a significant vulnerability in the state reporting of the Iris controller's companion app. To understand the impact of controller design decisions, we suppressed the on-demand telemetry that reported the unlock action. Both controllers reported on-demand and always-responsive telemetry over SSL, complicating the attack. However, we determined that packets carrying the on-demand telemetry were quite larger than the always responsive heartbeat messages. Thus, our flow modifications dropped packets larger than 250 and 359 bytes for the Iris and SmartThings controllers, respectively.

While suppressing the on-demand channel, we physically unlocked the deadbolt. We verified the deadbolt did correctly report the state change over Z-wave to the controller. Subsequently, our flow modifications dropped the on-demand state change messages from both controllers to their remote servers. After the flow modifications expired, we examined the status of the companion app. Initially both controllers reported the deadbolt as locked as we had suppressed the initial state change. However, after a maximum period of 100 seconds, the SmartThings controller updated the state as unlocked. In contrast, the Iris controller failed to subsequently

update the state of the lock in the companion app. Figure 7 depicts the companion app reporting the incorrect state of the lock thirty minutes after the flow modifications expired.

As we analyzed the telemetry of both devices, we discovered the systemic design flaw that lead to the attack. The SmartThings controller sent a periodic state update every 100 seconds. We passively observed this by locking and unlocking the deadbolt and then correlating the subsequent network traffic and smart phone app status. Even without a change to the state of the device, the SmartThings controller periodically transmitted the state of the device every 100 seconds. However, the Iris controller treated a state change as a single fixed event and only reported the state change when the action physically occurred. Thus, suppressing the single state change permanently confused the Iris smart phone app.

Finding 4: ***Controllers can prioritize buffering event notification based on severity.*** Comparing the distinct responses for a *water leak* between the Iris and SmartThings controller offers insight about buffering priorities. We evaluated this distinction by suppressing both controller's on-demand channels for thirty minutes. During the attack, we stimulated an on-demand event by placing the water leak sensors for both controllers under a running water faucet. During traffic suppression, the smart phone application for the Iris controller displayed an in-app offline status but failed to provide a smart phone alert.

However, the Iris controller distinguished itself by uniquely responding with an audible water leak alarm, lasting until the end of the attack. After ending the telemetry suppression, the Iris controller immediately uploaded the buffered water leak. In the default configuration, the back-end Iris services sent an email and phone call about the water leak. In contrast, the SmartThings app displayed no history of a previous water leak. We further examined the SmartThings device history using the SmartThings *Groovy IDE* and learned the hub never buffered or uploaded suppressed telemetry (i.e., the water leak notification). This behavior correlated to the sensor behavior we saw for SmartThings contact, motion, and button sensors that contained blank event histories after telemetry suppression. The SmartThings hub has 512MB DDR3 RAM, and 4GB of Flash Memory. After operating system demands, memory and storage exists to buffer priority events. In our observations, the SmartThings hub encapsulated a water leak detection message in a single packet of 418 bytes that consisted of 20 fields of a record. For high priority events (e.g., water leaks, carbon monoxide or fire alarms), the hub should buffer the events locally and deliver when the telemetry suppression ends.

Finding 5: ***Devices are less likely to buffer when both the on-demand and always-responsive subsystems are suppressed.*** Attacking the physical channel is a coarse-grained approach that reduces the transparency of the attack as it suppresses both the on-demand and always-responsive subsystems. In our experiments, physical layer suppression frequently resulted in smart phone alerts and in-app status changes, indicating connectivity losses at a higher degree than fine-grained network layer suppression. While physical layer suppression lacks transparency, we discovered that devices were less likely to buffer content when the physical channel was suppressed. In particular, the Canary and D-Link cameras both buffered content under network layer suppression but failed to buffer the same content under physical layer suppression. Without a connection to the wireless network, the devices failed to buffer any triggered motion alerts until they regained connectivity. Thus an attacker who valued attack permanence over transparency would benefit from physical layer suppression.

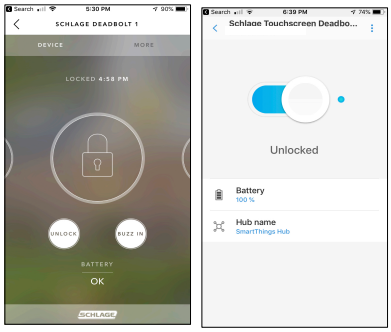


Figure 7: The Iris controller fails to buffer and re-transmit suppressed telemetry of the Schlage Deadbolt, facilitating the problematic case where the Iris app falsely reports the deadbolt as locked when an unlock action occurs during telemetry suppression.

5 POTENTIAL COUNTERMEASURES

This section discusses defenses against sensor blinding and state confusing attacks introduced in the previous sections and reserves future work. Several countermeasures exist to defend against our proposed attack vector. However, a full evaluation of countermeasures deserves a separate work with an emphasis on user studies to understand the transparency of the attack vector and the impact on the usability of devices.

Traffic Shaping: Traffic shaping solutions offer the ability to obscure on-demand activities from an attacker by manipulating traffic. Without the ability to isolate and identify the on-demand system, the adversary's attack is marginalized if not entirely defeated. Traffic shaping has been widely studied as a countermeasure to securing HTTPS from leaking privacy-sensitive information [23, 48]. Traffic shaping presents a solution as encryption is not enough to protect user privacy for IoT devices. The packet size and frequency of encrypted network traffic between an IoT device and the cloud is enough to reveal device-level activity [1, 5]. Our proposed attack vector relies on the ability to infer packets containing on-demand notifications and content. Contemporary works have proposed methods for padding IoT device packets. Apthorpe et al. [4] developed stochastic traffic padding to obfuscate user level activities. Further, Malekzadeh et al. [24] proposed Replacement AutoEncoder, a novel algorithm that transforms and masks the features of sensitive data. However, these methods do not address the discriminative nature of packet timings that reveal the always-responsive subsystem. Previous traffic shaping works have proposed random timing delays to protect privacy-sensitive disclosure of a channel [34]. However, this solution requires further study, as the always-responsive subsystem relies on the predictable arrival of periodic heartbeat messages.

Per-IoT Virtual Private Networks: Virtual Private Networks (VPNs) offer the ability to defend against an attacker that manipulates traffic at the border of our residential network. However, they do not protect against the model of a locally compromised wireless router. In this case, the IoT device itself would need to establish the VPN to benefit from the security and privacy it provides. This approach is similar to the per-app VPN offered for iPhones since iOS 9 [3]. Per-IoT virtual private networks would provide partial protection from an attacker inferring traffic activities and selectively suppressing the on-demand subsystem of IoT. However, VPN encapsulation would still need to incorporate traffic shaping to prevent discovery on subsystems based on size or timing. While applying per-IoT VPNs can guarantee security by placing a layer of abstraction over the on-demand and always-responsive subsystems, that guarantee comes at a performance cost. Further study can examine the impact on the quality of the on-demand system content and the user perception of device responsiveness.

Secure IoT Design: An IoT device's battery constraints, limited storage, and small processing capacity present challenges for secure software design development. However, as we discussed in Finding 4, they are often falsely used to motivate the challenges of IoT. As we analyzed in our findings, design flaws contributed to the severity of the attack by enabling transparency and permanence of our attacks. IoT firmware often isolates the on-demand and always responsive subsystems, segregating their functionality on different applications that establish distinct connections. Applications that implement network heartbeats must unify this segregation and gain awareness of the on-demand subsystem state. Further, the on-demand subsystem must enforce repudiation ensuring that content and notifications are acknowledged by the remote backend application. When feasible, the on-demand system must buffer notifications and content. As identified in Finding 4, IoT devices require priority buffer scheme to preserve preferential content and notifications. Periodic re-synchronization of sensor state is a low bandwidth solution that can mitigate the effects of sensor blinding and state confusion. However, periodic re-synchronization does not address the case of battery-constrained devices that are unable to benefit this approach. Finally, devices must select short-term timeouts that balance overwhelming the user while providing a transparent view of connectivity. All of these decisions impact the usability of the device and require further user study.

WSN and Embedded Security Approaches: The field of Wireless Sensor Networks (WSNs) offers insight for securing on-demand sensor reporting [7]. As WSN nodes have little provisions for security, they are vulnerable to attacks that compromise the integrity and availability of their reports. Roy et al. [32] proposed an attack-resilient computation algorithm that minimizes communication while verifying node integrity. Based on this approach, we consider introducing message counters into the always-responsive subsystem that maintain the on-demand subsystem state. Sciancalepore et al. [33] proposed a distributed protocol suitable for memory constrained devices, that guaranteed message delivery under jamming by using decoy messages. Based on their work, we hypothesize that on-demand decoy messages can determine the presence of blinding or confusing attacks. Further, the field of *trusted scheduling* for embedded systems offers another approach. Masti et al. [25] proposed trusted scheduling to ensure execution of safety-critical applications for embedded devices. From this insight, we consider introducing a subsystem-dependent telemetry model. This approach would delay delivery of always-responsive messages until on-demand messages were acknowledged and repudiated. All of these proposed approaches require further user study.

6 RELATED WORK

Protecting IoT devices and the privacy-sensitive information they produce is an emerging field of computer security. To this end, there have been several surveys that holistically examined the security and privacy of IoT [37, 50, 51]. In the closest related work, Obermaier and Hutle [31] examined the authentication and encryption schemes of cloud-based video system surveillance. However, their work narrowly focused on the implementation for four specific camera models. Wu and Lagasse [49] demonstrated an isolated case of the predictive nature of IoT by training a neural network to detect a single hidden wireless camera and classify video recording traffic. Wood et al. [47] narrowly focused on the implementation immaturity of four medical devices, including one device that leaked sensitive health information in clear-text traffic. While these works demonstrate the implementation immaturity and predictive nature of IoT, they offer little insight against a broader nature as we have demonstrated in our work.

Concurrent work has provided threat modeling for IoT device sensors. Chen et al. [12] examined spoofing attacks that confused sensors and caused traffic congestion to the U.S. Department of Transportation traffic control system. Uluagac et al. [39] analyzed sensory channel threats for cyber physical systems. Their work focuses on protecting the integrity of the sensory channel (e.g., light, temperature, infrared) to prevent adversarial use as a component of an attack. Lakshminarayana et al. [22] studied the impact of signal jamming against communication based train control systems and developed counter measures to limit the attacks' impact. These works provide insight into specific IoT deployments. However, we investigate problems in device telemetry, battery conservation, and implementation maturity that span several smart home IoT classes.

7 CONCLUSION

In this work, we hypothesized that the NEST and Amazon Key incidents are not isolated occurrences, but rather indicatives of a larger systemic design flaw in many IoT devices. This paper explored the design flaws in smart home IoT devices that facilitate sensor blinding and state confusion attacks. We have shown the telemetry and messaging protocols of smart home IoT devices commonly isolate the always-responsive and on-demand subsystems, enabling these attacks. To demonstrate the broad scope of the problem, we have profiled 24 popular smart home IoT devices and measured the severity of these attacks by analyzing their transparency and permanence. We uncover that 22 of 24 studied devices suffer from critical design flaws that (1) enable attacks to transparently disrupt the reporting of device status alerts or (2) prevent the uploading of content integral to the device's core functionality. Further, we examined the impact and feasibility of several countermeasures including traffic shaping, per-IoT VPNs, and secure design approaches.

REFERENCES

[1] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and A Selcuk Uluagac. 2018. Peek-a-Boo: I see your smart home activities, even encrypted! https://arxiv.org/pdf/1808.02741.pdf

[2] Vikram Sathyanarayana Anbazhagan, Rama Krishna Sandeep Pokkunuri, Swaminathan Sivasubramanian, Stefano Stefani, and Vladimir Zhukov. 2018. Service for developing dialog-driven applications. US Patent App. 15/360,814.

[3] Apple Computers, Inc. 2018. iOS 12 Security Guide iOS. https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

[4] Noah Apthorpe, Danny Yuxing Huang, Dillon Reisman, Arvind Narayanan, and Nick Feamster. 2018. Keeping the Smart Home Private with Smart (er) IoT Traffic Shaping. https://arxiv.org/pdf/1812.00955.pdf

[5] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. http://arxiv.org/abs/1708.05044

[6] David Barrera, Ian Molloy, and Heqing Huang. 2017. IDIoT: Securing the Internet of Things like it's 1994. https://arxiv.org/abs/1712.03623

[7] Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic. 2013. *Mobile ad hoc networking: Cutting edge directions*. Vol. 35. John Wiley & Sons, Hoboken, NJ.

[8] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. 2018. Behavioral Fingerprinting of IoT Devices. In *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security (ASHES '18)*. ACM, New York, NY, USA, 41–50.

[9] Nick Bilton. 2016. Nest Thermostat Glitch Leaves Users in the Cold. https://nyti.ms/1Or0eH0

[10] Nellie Bowles. 2018. Thermostats, Locks and Lights: Digital Tools of Domestic Abuse. *The New York Times* (2018).

[11] Guillaume Celosia and Mathieu Cunche. 2018. Detecting Smartphone State Changes Through a Bluetooth Based Timing Attack. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec '18)*. ACM, New York, NY, USA, 154–159.

[12] Qi Alfred Chen, Yucheng Yin, Yiheng Feng, Z Morley Mao, and Henry X Liu. 2018. Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control. In *Network and Distributed Systems Security (NDSS) Symposium 2018*. Internet Society, San Diego, CA, 1–15.

[13] Google Cloud. 2018. Overview of Internet of Things: Data Storage. https://cloud.google.com/solutions/iot-overview

[14] Thomas d'Otreppe. 2019. Aircrack-ng. https://www.aircrack-ng.org

[15] Jeffrey Hagins and Alexander Hawkinson. 2016. Distributed control scheme for remote control and monitoring of devices through a data network. US Patent 9,462,041.

[16] Jason Hanna and Stella Chan. 2018. The murder suspect denies it. The victim's Fitbit tells another story, police says. https://www.cnn.com/2018/10/04/us/california-fitbit-killing/

[17] Marshall Honorof. 2017. Amazon Cloud Cam, Key Flaws Could Let in Burglars. https://www.tomsguide.com/us/amazon-key-cloud-cam-hack,news-26132.html

[18] CleverLoop Inc. 2018. Smart Home Security System: FAQs Before Purchase. https://www.cleverloop.com/faqs.html

[19] Information Sciences Institute. 1981. IETF RFC 793: Transmission Control Protocol (TCP). https://www.ietf.org/rfc/rfc793.txt

[20] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Zhuoqing Morley Mao, Atul Prakash, and Shanghai JiaoTong Unviersity. 2017. ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms.

[21] Andrew J Kerns, Daniel P Shepard, Jahshan A Bhatti, and Todd E Humphreys. 2014. Unmanned aircraft capture and control via GPS spoofing. *Journal of Field Robotics* 31, 4 (2014), 617–636.

[22] Subhash Lakshminarayana, Jabir Shabbir Karachiwala, Sang-Yoon Chang, Girish Revadigar, Sristi Lakshmi Sravana Kumar, David K.Y. Yau, and Yih-Chun Hu. 2018. Signal Jamming Attacks Against Communication-Based Train Control: Attack Impact and Countermeasure. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec '18)*. ACM, New York, NY, USA, 160–171.

[23] Marc Liberatore and Brian Neil Levine. 2006. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*. ACM, New York, NY, USA, 255–263.

[24] M. Malekzadeh, R. G. Clegg, and H. Haddadi. 2018. Replacement AutoEncoder: A Privacy-Preserving Algorithm for Sensory Data Analysis. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, Orlando, FL, 165–176. https://doi.org/10.1109/IoTDI.2018.00025

[25] Ramya Jayaram Masti, Claudio Marforio, Aanjhan Ranganathan, Aurélien Francillon, and Srdjan Capkun. 2012. Enabling Trusted Scheduling in Embedded Systems. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC '12)*. ACM, New York, NY, USA, 61–70.

[26] Molly McHugh. 2015. How NestDoor and Nest Cams are helping cops solve crimes. https://www.wired.com/2015/12/nextdoor-crime-nest-cams/

[27] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. R. Sadeghi, and S. Tarkoma. 2017. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, IEEE, Atlanta, GA, 2177–2184.

[28] Maxime Montoya, Simone Bacles-Min, Anca Molnos, and Jacques J.A. Fournier. 2018. SWARD: A Secure WAke-up RaDio Against Denial-of-Service on IoT Devices. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec '18)*. ACM, New York, NY, USA, 190–195.

[29] Mariella Moon. 2016. Software bug forced Nest thermostats offline. http://engt.co/1Oskqsf

[30] Lily Hay Newman. 2018. Turning an Echo Into a Spy Device Only Took Some Clever Coding. https://www.wired.com/story/amazon-echo-alexa-skill-spying

[31] Johannes Obermaier and Martin Hutle. 2016. Analyzing the Security and Privacy of Cloud-based Video Surveillance Systems. In *Proceedings of the 2Nd ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS '16)*. ACM, New York, NY, USA, 22–28.

[32] Sankardas Roy, Mauro Conti, Sanjeev Setia, and Sushil Jajodia. 2014. Secure data aggregation in wireless sensor networks: Filtering out the attacker's impact. *IEEE Transactions on Information Forensics and Security* 9, 4 (2014), 681–694.

[33] Savio Sciancalepore, Gabriele Oligeri, and Roberto Di Pietro. 2018. Strength of Crowd (SOC) Defeating a Reactive Jammer in IoT with Decoy Messages. *Sensors* 18, 10 (2018), 3492.

[34] Vitaly Shmatikov and Ming-Hsiu Wang. 2006. Timing analysis in low-latency mix networks: Attacks and defenses. , 18–33 pages.

[35] Sandra Siby, Rajib Ranjan Maiti, and Nils Ole Tippenhauer. 2017. IoTScanner: Detecting Privacy Threats in IoT Neighborhoods. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS '17)*. ACM, New York, NY, USA, 23–30.

[36] Amit Kumar Sikder, Hidayet Aksu, and A Selcuk Uluagac. 2017. 6thsense: A context-aware sensor-based attack detector for smart devices.

[37] Amit Kumar Sikder, Giuseppe Petracca, Hidayet Aksu, Trent Jaeger, and A Selcuk Uluagac. 2018. A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications. https://arxiv.org/pdf/1802.02041.pdf

[38] Saleh Soltan, Prateek Mittal, and H Vincent Poor. 2018. BlackIoT: IoT Botnet of high wattage devices can disrupt the power grid. In *Proc. USENIX Security*, Vol. 18. USENIX, Baltimore, MD, 15–32.

[39] A. S. Uluagac, V. Subramanian, and R. Beyah. 2014. Sensory channel threats to Cyber Physical Systems: A wake-up call. In *2014 IEEE Conference on Communications and Network Security*. IEEE, San Francisco, CA, 301–309.

[40] Mathy Vanhoef and Frank Piessens. 2014. Advanced Wi-Fi Attacks Using Commodity Hardware. In *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC '14)*. ACM, New York, NY, USA, 256–265.

[41] Mathy Vanhoef and Frank Piessens. 2015. All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS. , 97–112 pages.

[42] Mathy Vanhoef and Frank Piessens. 2016. Predicting, Decrypting, and Abusing WPA2/802.11 Group Keys. , 673–688 pages.

[43] Mathy Vanhoef and Frank Piessens. 2017. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, USA, 1313–1328.

[44] Mathy Vanhoef and Frank Piessens. 2018. Release the Kraken: New KRACKs in the 802.11 Standard. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. ACM, New York, NY, USA, 299–314.

[45] WikiLeaks. 2018. Vault 7: Detailed Notes regarding Samsung F8000 Smart TV networking. https://wikileaks.org/ciav7p1/cms/page_13205592.html

[46] Matthias Wilhelm, Ivan Martinovic, Jens B. Schmitt, and Vincent Lenders. 2011. Short Paper: Reactive Jamming in Wireless Networks: How Realistic is the Threat?. In *Proceedings of the Fourth ACM Conference on Wireless Network Security (WiSec '11)*. ACM, New York, NY, USA, 47–52.

[47] Daniel Wood, Noah Apthorpe, and Nick Feamster. 2017. Cleartext Data Transmissions in Consumer IoT Medical Devices. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy (IoT S&P '17)*. ACM, New York, NY, USA, 7–12.

[48] Charles V Wright, Scott E Coull, and Fabian Monrose. 2009. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis.

[49] Kevin Wu and Brent Lagesse. 2019. Do You See What I See? Detecting Hidden Streaming Cameras Through Similarity of Simultaneous Observation. https://faculty.washington.edu/lagesse/publications/SSO.pdf

[50] Hiroto Yasuura, Chong-Min Kyung, Yongpan Liu, and Youn-Long Lin. 2018. Smart sensors at the IoT frontier.

[51] Nan Zhang, Soteris Demetriou, Xianghang Mi, Wenrui Diao, Kan Yuan, Peiyuan Zong, Feng Qian, XiaoFeng Wang, Kai Chen, Yuan Tian, Carl A. Gunter, Kehuan Zhang, Patrick Tague, and Yue-Hsun Lin. 2017. Understanding IoT Security Through the Data Crystal Ball: Where We Are Now and Where We Are Going to Be. http://arxiv.org/abs/1703.09809