On the State of Internet of Things Security: Vulnerabilities, Attacks, and Recent Countermeasures

DEVKISHEN SISODIA, University of Oregon

In this paper, we review the state of Internet of Things (IoT) security research, with a focus on recent countermeasures that attempt to address vulnerabilities and attacks in IoT networks. Due to the fact that IoT encompasses a large range of significantly distinct environments, each of which merits their own survey, our survey focuses mainly on the smart home environment. Based on the papers surveyed, we pinpoint several challenges and open issues that have yet to be adequately addressed in the realm of IoT security research. Lastly, in order to address these open issues, we provide a list of future research directions on which we believe researchers should focus.

CCS Concepts: • **Security and privacy → Mobile and wireless security**.

Additional Key Words and Phrases: Internet of Things (IoT); security; vulnerabilities; attacks; countermeasures

1 INTRODUCTION

The Internet of Things (IoT) is a computer networking paradigm that refers to scenarios where network connectivity and computing capability extends to embedded sensors and everyday devices, allowing them to generate, exchange, consume, and act upon data with minimal to no human intervention. This paradigm is possible due to recent advancements in the miniaturization of electronics, networking capabilities, and computing power. Even in its relative infancy, IoT is already impacting our everyday lives in profound ways, from healthcare to home automation. In 2018, 7 billion devices were connected to the Internet [1] and this number is expected to increase to 19.8 billion by 2023 [2].

While IoT devices and traditional machines suffer from the same types of attacks, IoT devices tend to be harder to secure due to some unique properties. IoT devices are often harder to patch and update due to largely non-existent automatic update systems. Also, they tend to have scarce CPU and memory resources, and limited battery capacity, if not plugged into an external power source. IoT devices can have anywhere from a few gigabytes to a few kilobytes of memory. Furthermore, with many different types of IoT devices, IoT networks are far more diverse and heterogeneous than traditional networks. These unique properties, which differentiate IoT devices from traditional machines, hinder the deployment of existing security mechanisms in IoT environments.

Due to the prevalence of IoT and the security threats facing it, we survey the area of IoT security. The goal of this survey is to provide a holistic-view of the vulnerabilities, attacks, and recent countermeasures in the Internet of Things. This survey is particularly focused on countermeasures proposed in the last five years to capture the more recent landscape of IoT security, especially smart home security. We hope to provide the reader with a view on what security-related topics the IoT research community has been focusing on, mainly in the last five years. We also discuss several challenges facing IoT security research, and missing gaps that we believe have not been sufficiently addressed. We conclude by proposing several future research directions that could help to address the pinpointed challenges and missing gaps.

What follows is a road map detailing the structure of this survey.

- Section 2 explains the scope of the survey, the methodology used to collect the papers, and the taxonomy used to categorize the papers.
- Section 3 describes a few of the most cited surveys in the last decade relating to IoT.
- Section 4 analyzes various IoT vulnerabilities.
- Section 5 analyzes various IoT attacks related to the aforementioned vulnerabilities.

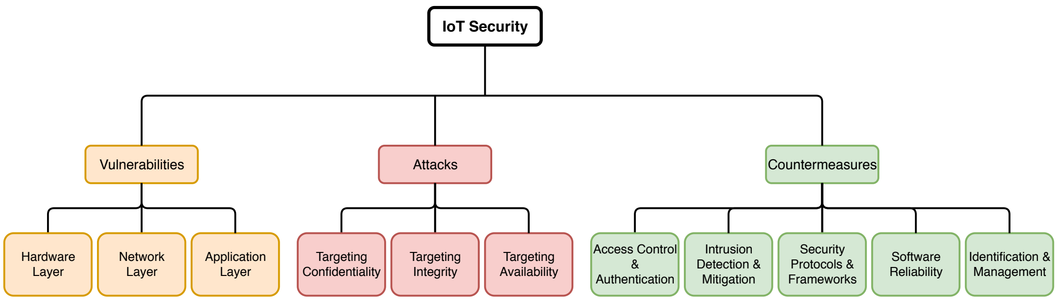


Fig. 1. The taxonomy used to organize this survey.

- Section 6 analyzes various IoT countermeasures addressing the aforementioned vulnerabilities and attacks.
- Section 7 discusses open Issues and possible future work directions.
- Section 8 provides our concluding remarks.

2 BACKGROUND

This section will explain the scope of the survey and the methodology used to find the papers. We further provide a detailed explanation of the taxonomy used in the survey to organize the papers.

2.1 Scope

The scope of this survey is to provide a holistic-view of the vulnerabilities, attacks, and recent countermeasures in the Internet of Things. We begin by detailing vulnerabilities in IoT, then explain attacks that can exploit those vulnerabilities, and finally, describe countermeasures that attempt to address those attacks. The survey mainly focuses on smart home networks, with less of an emphasis on industrial IoT (IIoT), supervisory control and data acquisition (SCADA) networks, smart power grids, and other large-scale IoT networks.

2.2 Methodology

In order to capture the more recent landscape of IoT security, we focus on papers related to countermeasures proposed in the last five years. In order to do so, we examined papers in top conferences and journals from 2015 to the end of 2019, which make up most of our surveyed papers. Other papers included in this survey were found mainly by reading the related work of the recent papers, and were those that we thought were most relevant to the area, and highly cited (especially, among the recent papers).

Below is a list of conferences and journals where the majority of the surveyed papers are published:

- USENIX Security
- IEEE Symposium on Security and Privacy (S&P)
- ACM Annual International Conference on Mobile Computing and Networking (MobiCom)
- ACM Conference on Computer and Communications Security (CCS)
- ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI)
- Network and Distributed System Security Symposium (NDSS)
- Computer Networks
- IEEE Communications Surveys & Tutorials

2.3 Taxonomy

Figure 1, shows the taxonomy this survey uses to organize the reviewed papers.

2.3.1 Vulnerabilities.

In the **Vulnerabilities** category, we survey IoT vulnerabilities in each of the three main IoT layers.

- **Hardware Layer**: In IoT, the hardware layer (also known as the perception layer) is primarily responsible for sensing the environment by capturing cyber-physical data. In this subcategory, we will survey vulnerabilities relating to the hardware of IoT devices.
- **Network Layer**: The network layer is responsible for providing device-to-device communication, and acts as a bridge between the hardware and application layers. In this subcategory, we will survey vulnerabilities relating to the communication protocols used by IoT devices.
- **Application Layer**: The application layer is responsible for generating and displaying analytics from data obtained from the hardware layer, and allows users to interface with the IoT devices. In this subcategory, we will survey vulnerabilities relating to the firmware and software of IoT devices.

2.3.2 Attacks.

In the **Attacks** category, we survey various IoT attacks targeting each of the three main security requirements of the CIA triad (confidentiality, integrity, availability).

- **Targeting Confidentiality**: The confidentiality requirement is intended to protect data and devices from unauthorized access, and is enforced by access control, authentication, and encryption methods. Therefore, attacks targeting confidentiality essentially aim to break privacy assurances. In this subcategory, we will survey attacks targeting confidentiality.
- **Targeting Integrity**: The integrity requirement is intended to protect data from unauthorized modifications, and is enforced in-part by intrusion detection and mitigation methods, and assuring software reliability through updates and patches, along with access control, authentication, and encryption. Therefore, attacks targeting integrity essentially aim to break data non-alteration assurances. In this subcategory, we survey attacks targeting integrity.
- **Targeting Availability**: The availability requirement is intended to guarantee reliable access to data and devices, and is enforced in-part by monitoring devices and network infrastructure to ensure services are available, and access control to prevent malicious entities from exhausting device and network resources. Therefore, attacks targeting availability essentially aim to break reliable and timely access assurances. In this subcategory, we survey attacks targeting availability.

2.3.3 Countermeasures.

In the **Countermeasures** category, we survey various recent countermeasures, and group them into five categories that best describe them.

- **Access Control & Authentication**: Access control and authentication methods are used to address a number of IoT vulnerabilities and attacks. In fact, access control and authentication can be employed to prevent attacks targeting each of the three main security requirements (confidentiality, integrity, and availability). In this subcategory, we survey recent papers related to access control and authentication in IoT environments.
- **Intrusion Detection & Mitigation**: Intrusion detection systems (IDSes) are employed to detect malicious behavior, and possibly mitigate the effects of such behavior on IoT networks. IDSes can also be used to detect and mitigate attacks targeting confidentiality, integrity and

availability. In this subcategory, we survey recent papers related to intrusion detection and mitigation in IoT environments.

- **Security Protocols & Frameworks**: In recent years, researchers have proposed security protocols and frameworks to address the unique challenges of securing IoT environments. In this subcategory, we survey recent papers related to security protocols and frameworks for IoT environments.
- **Software Reliability**: Many real-world IoT attacks can be traced to vulnerabilities of IoT software. Therefore assuring software reliability through code verification, automatic updates, and rapid patching is paramount in protecting vulnerable devices. In this subcategory, we will survey recent papers related to software reliability for IoT environments.
- **Identification & Management**: The difficulty for users to identify and troubleshoot mismanaged or misconfigured IoT devices leads to vulnerable and attack-prone IoT networks. Therefore, the identification and management of IoT devices and networks is an important step in securing IoT environments. In this subcategory, we will survey recent papers related to identification and management of IoT devices and networks.

3 RELATED SURVEYS

In this section, we briefly describe a few of the most cited surveys in the last decade relating to IoT, and explain how our survey fits among them. Because there is a plethora of recent IoT security surveys [3–23], we focus only on a handful of surveys that are most related to this paper. Most of the security surveys we found can be categorized into five categories:

- vulnerabilities,
- security of communication protocols,
- access control,
- intrusion detection, and
- application & middleware security.

3.1 Vulnerabilities

A few surveys focus mainly on vulnerabilities in IoT environments. By surveying a wide-range of vulnerabilities, Neshenko et al. [3] concluded that most IoT attacks are possible because of two main vulnerabilities in IoT:

- unnecessarily open ports, and
- weak programming practices coupled with improper software update capabilities.

The authors further point out that insufficient IoT access controls and audit mechanisms enable attackers to generate IoT-centric malicious activities in a highly stealthy manner.

Alrawi et al. [4] evaluated 45 popular real smart home devices for various known vulnerabilities. The results of their survey show that

- 40% of the devices had security issues,
- of the mobile applications used to control the devices, 64% were overprivileged, 44% leaked sensitive data, and 48% incorrectly used cryptographic protocols,
- 51% of the cloud endpoints used by the devices and applications had TLS/SSL configuration issues, and finally,
- 46% of the devices were susceptible to man-in-the-middle (MITM) attacks, and 60% of the devices used communication channels that had partial or no encryption.

3.2 Security of Communication Protocols

Instead of surveying a broad range of IoT vulnerabilities like Neshenko et al. [3], Granjal et al. [5] focused mainly on vulnerabilities and attacks targeting existing communication protocols used in IoT networks. They also provided insights on ongoing work attempting to secure those protocols. The authors find various research challenges for security at each layer. At the physical layer, protocols (e.g., IEEE 802.15.4) do not specify a key model (i.e., a model for generating, distributing, storing, and replacing cryptographic keys), because it depends largely on the resources available on the IoT devices to support key management operations. At the network layer, routing protocols (e.g., Routing Protocol for Low-Power and Lossy Networks or RPL) offer security against external attacks only, and are not resilient against internal attacks. And finally, at the application layer, protocols (e.g., Constrained Application Protocol or CoAP) lack appropriate key management mechanisms for multicast communication.

3.3 Access Control

A recent survey conducted by Ouaddah et al. [6] is on access control mechanisms for IoT. The authors made two important conclusions:

- access control mechanisms for the traditional Internet cannot be directly applied to IoT, and therefore these mechanisms need to either be adapted for IoT, or new mechanisms need to be conceived with IoT specific requirements in mind, and
- access control in IoT can be categorized into centralized or distributed access control, each having their own advantages and drawbacks.

3.4 Intrusion Detection

Zarpelao et al. [7], Mosenia et al. [8], and Gendreau et al. [9] all surveyed intrusion detection in IoT. Many IDS-based security papers for IoT not only focus on attack detection and mitigation, but also placement of security resources. The careful placement of IDSes is more critical in IoT networks than in traditional networks due to

- limited computational/memory/battery capacity at IoT devices, and
- device-to-device communication or mesh networking.

All three surveys conclude that detection and prevention are difficult in the realm of IoT due to the limited computational power of IoT devices. A recent survey by Chaabouni et al. [10] focused on machine learning-based network IDSes for IoT — a research area still in its infancy.

3.5 Application Security

Celik et al. [11] studied privacy and security issues related to IoT program analysis. They analyzed a plethora of analysis systems for five major IoT programming platforms (Samsung's SmartThings, OpenHAB, Apple's HomeKit, Google's Android Things, and Amazon AWS IoT). The authors found five open issues when it comes to IoT program analysis:

- Program analysis techniques often focus on smart homes, and as a result, are not responsive to the unique characteristics and constraints of different IoT domains.
- Current IoT analysis systems are not scalable, and therefore may not be feasible for IoT systems where large-scale programs are developed, such as automobiles and industrial IoT.
- Analysis systems need to be aware of the potential security issues that may result from the interactions between the program and physical process, because safety and security issues are more extreme in IoT environments due to the authority given to IoT systems over the physical world.
- Analysis systems often do not assess their impact on the system resources.

- When a security violation occurs, analysis approaches need to consider taking the right course of action — in many cases, simply blocking a device state or asking for user approval could be dangerous.

3.6 Other IoT Security Surveys

Some of the IoT security-related surveys we found do not fit well into any of the aforementioned five categories. For example, instead of studying security solutions themselves, Hellaoui et al. [12] studied the techniques that allow for energy-efficient security solutions. To the best of our knowledge, this is the first and only survey to approach IoT security from an energy-efficiency standpoint. Weber et al. [13] looked into legal frameworks for dealing with various IoT security threats, and come to the conclusion that it is extremely difficult to craft frameworks that are flexible and innovative enough to keep up with the rapidly evolving threat landscape inherent to IoT, and therefore, IoT security should not rely solely on legal and regulatory approaches. Finally, there are a few surveys that study how emerging technologies can be applied in the realm of IoT security. For example, Khan et al. [14] surveyed blockchain solutions for identification and governance, authentication and integrity, and secure communications, while Farris et al. [15] surveyed software-defined networking (SDN) and network function virtualization (NFV) solutions for providing advanced security mechanisms for IoT environments, such as dynamic flow control, decoupling security software from hardware, and security service chaining.

In conclusion, we found that the surveys tended to look at IoT security as a collection of separate parts, and not as a whole, interconnected system. For example, some surveys focused mainly on vulnerabilities or attacks, while others focused mainly on a specific form of countermeasure (e.g., access control or intrusion detection). While the focus of our paper is on countermeasures, we provide a more holistic view of IoT security by first identifying the vulnerabilities that especially affect IoT devices and networks, next describing the attacks that can result from those vulnerabilities, and finally categorizing and comprehensively analyzing recent countermeasures for the aforementioned IoT vulnerabilities and attacks.

4 VULNERABILITIES

In this section, we describe various IoT vulnerabilities, and categorize them by the three IoT layers in which they originate. In general, IoT architectures consist of three layers:

- hardware layer,
- network layer, and
- application layer.

This section in no way presents an exhaustive list of IoT vulnerabilities, as this would merit a survey all to itself [3]. Instead, we mainly focus on vulnerabilities that have peaked the interest of the IoT security research community in recent years.

4.1 Hardware Layer

4.1.1 Lack of Restricted Access to Hardware:

Unlike our phones, laptops, and personal computers, many IoT devices operate in an unattended fashion, making it easier for attackers to physically tamper with them and go undetected. Wurm et al. [24] presented one of the earliest works to empirically analyze hardware vulnerabilities, which can be exploited by physical tampering, in consumer and industrial IoT devices. Specifically, the authors studied the hardware, software, and network security of home automation and smart meter devices, and identified several backdoors. A major hardware vulnerability they found in both consumer and industrial devices was the unrestricted access to the universal asynchronous receiver-transmitter (UART) which allowed them to modify the devices' boot sequences. By modifying the boot sequences, they were able to gain low-level access to the device, and thereby extract login information. Another major vulnerability the authors found in the devices was the lack of encryption. The lack of encryption allowed them to directly modify the filesystem of each device.

4.1.2 *Lack of Necessary Power for Cryptographic Primitives:*

While it is clear that encryption can help to address some of the vulnerabilities presented in [24], complex cryptographic functions, such as those found in the Advanced Encryption Standard (AES), can result in large overhead for resource-constrained IoT devices. As a result, there is a growing interest in ultra lightweight, but secure encryption algorithms optimized for low-powered hardware. However, as Singh et al. [25] pointed out, hardware-based encryption engines have a significant vulnerability: the power dissipation of the hardware can be measured while performing encryption, and later statistically analyzed to recover the secret key, thus compromising the device. Many countermeasures have been proposed to address this vulnerability in AES engines. Unfortunately, these countermeasures incur significant power and performance overheads, and therefore are not suitable for lightweight cryptographic primitives.

4.1.3 *Lack of Verification of Abnormal Stimuli:*

Vulnerabilities in sensors used by IoT devices can cause unexpected behavior which could lead to various attacks. More recently, researchers have shown that various virtual assistants used to control smart home devices are susceptible to different types of voice-command injection attacks, which we will elaborate on in subsection 4.3.6 when we discuss vulnerabilities at the application layer. However, Sugawara et al. [26] recently (November 2019) discovered a new hardware-based vulnerability which can be exploited to cause a laser-based audio injection attack. Essentially, the authors discovered that the microelectro-mechanical systems (MEMS) microphones used in smart home devices unintentionally respond to light as if it were sound. In order to exploit this effect, the authors inject "sound" (via laser light) into the microphones of 17 popular smart home devices that use Amazon's Alexa, Apple's Siri, Facebook's Portal, and Google Assistant, by simply modulating the amplitude of a laser light. By doing so, the authors show that they can unlock front doors, open garage doors, and even locate, unlock, and start various vehicles.

4.2 Network Layer

4.2.1 *Lack of Encryption:*

IoT devices are vulnerable to the same security threats faced by any Internet-connected machine. As Alrawi et al. showed in [4], many popular IoT devices do not encrypt their communication channels, and lack endpoint verification. Furthermore, many devices leak sensitive information by using third-party recursive DNS servers. These vulnerabilities clearly can also apply to non-IoT machines. However, IoT devices are also susceptible to IoT-specific vulnerabilities caused by the low-power communication protocols used almost exclusively in IoT environments, such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [27], ZigBee [28], and Bluetooth Low Energy (BLE) [29].

4.2.2 *Security Flaws in Various Communication Protocols:*

Before we present the vulnerabilities of 6LoWPAN, ZigBee, and BLE, we provide a brief overview of each. 6LoWPAN uses the Constrained Application Protocol (CoAP) [30] as its application layer protocol, UDP with DTLS at the transport layer, IPv6 over LoWPAN with Routing Protocol for Low-Power and Lossy Networks (RPL) [31] as the routing algorithm at the network layer, and finally IEEE 802.15.4 at the data link and physical layers. ZigBee on the other hand uses ZigBee-specific protocols at the application, transport, and network layers, with a routing algorithm similar to Ad hoc On-Demand Distance Vector Routing (AODV) [32], but like 6LoWPAN, also uses IEEE 802.15.4 as the underlying data link and physical layer protocol. Unlike 6LoWPAN and ZigBee, BLE supports its own physical (BLE PHY) and data link layer (BLE MAC) protocols, but like ZigBee, also supports its own application, transport, and network layer protocols.

Granjal et al. [5] showed that Neighbor Discovery (ND) and mesh routing mechanisms in IEEE 802.15.4 environments, such as 6LoWPAN and ZigBee, are susceptible to security threats, such as routing and data leakage attacks. While RPL has a self-healing mechanism to deal with routing topology failures, link failures, and node failures, Wallgren et al. [33] showed that this mechanism cannot self-correct networks under certain selective forwarding attacks. We elaborate more on these attacks in subsection 5.3.1.

ZigBee devices use symmetric-key encryption to establish secure communications. However, if keys are not pre-installed on devices that want to communicate, they are transmitted unencrypted, making it feasible for attackers to not only gain sensitive information, but control over the devices, as demonstrated by Vidgren et al. [34]. Furthermore, studies by Zillner [35] and Morgner et al. [36] showed that the ZigBee Light Link (ZLL) protocol, which is used in smart home lighting systems to set up the connection between smart lights and the hubs responsible for controlling them, had a security flaw that allowed control of smart lights to be taken from their hubs and given to adversaries. As we explain in subsection 5.3.3, this vulnerability can be exploited to launch crippling distributed-denial-of-service (DDoS) attacks against smart cities.

BLE also has its share of vulnerabilities. As shown by Ryan [37], a flaw in the key-exchange protocol for Bluetooth allows an attacker to passively recover session keys. Ho et al. [38] showed how door locks could be opened by launching relay attacks on BLE — an attacker can capture a lock's BLE authentication challenge message, relay the message (possibly over Wi-Fi for long-range communication) to an accomplice who is several meters away from the lock's rightful owner; the accomplice would then broadcast the challenge message and capture the response from the rightful owner's device, and relay the response back to the attacker who can now use the response to unlock the door. Even more worrying is that home-based IoT devices have vulnerable legacy versions of low energy protocols implemented in hardware, thereby limiting their mitigation options, as described by Alrawi et al. [4].

4.2.3 Open Ports:

Lastly, a major concern to the security of IoT networks is the significant number of devices with unnecessarily open ports. Czyz et al. [39] showed that a large number of IoT devices are only reachable over IPv6, and various IoT protocols are more accessible over IPv6 than over IPv4 (e.g., 6LoWPAN). They discovered that a given IPv6 port is almost always more open than the same port is in IPv4. For example, IPv6 had 5% more open SSH ports, and 46% more open Telnet ports as compared to IPv4. They also concluded that there was a systemic failure in organizations to deploy consistent security policies for their devices as it pertains to port blocking. Lastly, the authors debunked the belief that the security threat of open ports in IPv6 is dampened due to the infeasibility of IPv6 network-wide scanning by discovering high-value hosts through scanning alone.

4.3 Application Layer

4.3.1 Security Flaws in Firmware Images:

We begin by taking a look at firmware vulnerabilities in IoT. Costin et al. [40] performed simple static analysis on 32,000 embedded firmware images, and discovered 38 previously unknown vulnerabilities in almost 700 firmware images. They were also able to verify that some of those vulnerabilities are affecting at least 140,000 devices on the Internet. Specifically, the authors were able to extract private RSA keys and their self-signed certificates used in an estimated 35,000 online devices (many of these devices were surveillance cameras). The authors were also able to extract 100 distinct hard-coded password hashes, and, for 58 of them, recover the original passwords. However, simple static analysis alone is insufficient to discover all of the vulnerabilities in a given firmware image, and therefore the number and severity of the vulnerabilities discovered through only simple static analysis should be very worrying.

4.3.2 *Lack of Reliable Patching and Update Mechanisms:*

In order to minimize the number of attack vectors, operating systems, firmware, and applications should be patched regularly. However, many manufacturers either do not regularly maintain security patches, or do not have automated update mechanisms in place [3]. Some devices themselves may lack software updating capabilities, while others may outlive the limited period for which they receive updates. Even if updates are available, they may be challenging to apply, as some devices may require users to actively install updates, while those that automatically install might need to restart to update their firmware, causing gaps in availability [41]. Furthermore, even available update mechanisms lack integrity guarantees, rendering them vulnerable to man-in-the-middle and modification attacks.

4.3.3 *Weak Credentials and Lack of Strong Authentication Mechanisms:*

Weak credentials and lack of strong authentication mechanisms is a major concern for IoT devices. In 2010, Cui et al. [42] conducted Internet-scale probing and uncovered more than half a million embedded devices with default credentials. Most of these devices belonged to government organizations, large enterprises, Internet Service Providers (ISPs), and educational institutions. Two years later, in 2012, the Carna botnet revealed that there were more than 1.2 million devices online with no or default credentials [43]. Weak credentials have in fact led to large-scale, real-world attacks [43–46]. We elaborate on these attacks in subsection 5.1.2.

4.3.4 *Exposure of Sensitive Data:*

IoT applications are also prone to data leakage vulnerabilities. Celik et al. [47] conducted static taint analysis on 230 SmartThings applications, and found that 138 of the applications exposed at least one piece of sensitive data via the Internet or messaging services. Furthermore, the authors showed that half of the analyzed applications leak at least three different sensitive data sources, such as device info, devices state, user input, etc., to the Internet or messaging services.

4.3.5 *Lack of Data Flow Control in Trigger-Action Platforms:*

In recent years, researchers have begun studying the potential vulnerabilities of trigger-action IoT platforms. The trigger-action programming paradigm provides a simple and intuitive abstraction for non-technical users to automate IoT devices, and as a result, is commonly used for home automation. Essentially, a trigger-action program contains a set of trigger-action rules that specify that when a certain trigger event occurs, such as motion is detected, one or more actions, such as turn on the lights, should be subsequently executed.

Wang et al. [48] comprehensively analyzed the interactions between trigger-action rules in order to identify inter-rule vulnerabilities. They identified six different types of inter-rule vulnerabilities Note, to explain the inter-rule vulnerabilities, we use example rules in the format *If A, then C*, where *A* is the condition (e.g., user is home), and *C* is the event (e.g., turn on the lights). This six different types of inter-rule vulnerabilities are listed below.

- Condition Bypass: Consider the rules *If A and B, then C*, and *If A, then C*. In this case, condition *B* will be surpassed by rule *If A, then C*.

- Condition Block: Consider the rules *If A and B, then C*, and *If A, then not B*. In this case, the rule *If A, then not B* prevents the rule *If A and B, then C* from ever being satisfied.
- Action Revert: Consider the rules *If A, then B*, *If B, then C*, *If C, then D*, and *If D, then not B*. In this case, the rule *If A, then B* is reverted by the rule *If D, then not B*.
- Action Conflict: Consider the rules *If A, then C*, and *If B, then not C*. In this case, if conditions *A* and *B* are met, then there will be a conflict between actions *C* and *not C*.
- Action Loop: Consider the rules *If A, then B*, *If B, then not A*, *If not A, then not B*, and *If not B, then A*. In this case, the conditions cause an infinite loop of actions.
- Action Duplicate: Consider the rules *If A, then C*, and *If A, then C*. In this case, a single instance of condition *A* could cause multiple duplicate actions *C*.

Some of these vulnerabilities, if exploited, could lead to serious damage. For example, an attacker could exploit the action loop vulnerability to continuously turn on and off a light potentially inducing seizures, or the action duplicate to inject medicine multiple times potentially causing a fatal reaction.

Once the authors identified the inter-rule vulnerabilities, they then analyzed 315,393 applications from a popular trigger-action programming platform, IFTTT (acronym for If This Then That), and found that 66% of them had potential for inter-rule vulnerabilities. Due to the popularity of trigger-action platforms, coupled with the aforementioned serious vulnerabilities, there has been a push within the research community to find appropriate countermeasures to these vulnerabilities. We detail some of these countermeasures in subsection 6.1.5.

4.3.6 Lack of Verification in Virtual Personal Assistant Services:

Lastly, we describe speech recognition and voice-injection vulnerabilities that are present in virtual personal assistant (VPA) services running on IoT devices. Given a voice commands, such as "will it rain today?", VPA services can run a function, or a skill, to respond to the command. Most VPA services, such as Amazon's Alexa and Google Assistant, allow third-party skills to be uploaded to their skills market to further enrich the user experience when interacting with the VPA services. Unfortunately, Kumar et al. [49] found a vulnerability, which was later termed invocation confusion [50], that when exploited could cause a voice squatting attack (VSA). This vulnerability stems from the fact that VPA services allow for different skills to be initiated by same or similar sounding commands. Therefore, an attacker could exploit this vulnerability to create a malicious skill that can be initiated unknowingly by a victim. The authors give the example of a malicious skill, which attempts to phish credit card information, that is initiated by the words "Capital Won". In this case, the victim may initiate the malicious skill when he/she says the words "Capital One", which in normal circumstances (in absence of the malicious skill) would initiate the legitimate skill (which opens the authentic Capital One application). Speech recognition and voice-injection vulnerabilities in VPA services can cause more interesting attacks, some of which we discuss in subsection 5.2.1.

5 ATTACKS

In this section, we describe various IoT attacks, and categorize them by the security requirements they target. We focus on the three main requirements:

- confidentiality,
- integrity, and
- availability.

Most of the attacks we cover in this section exploit the vulnerabilities we described in section 4.

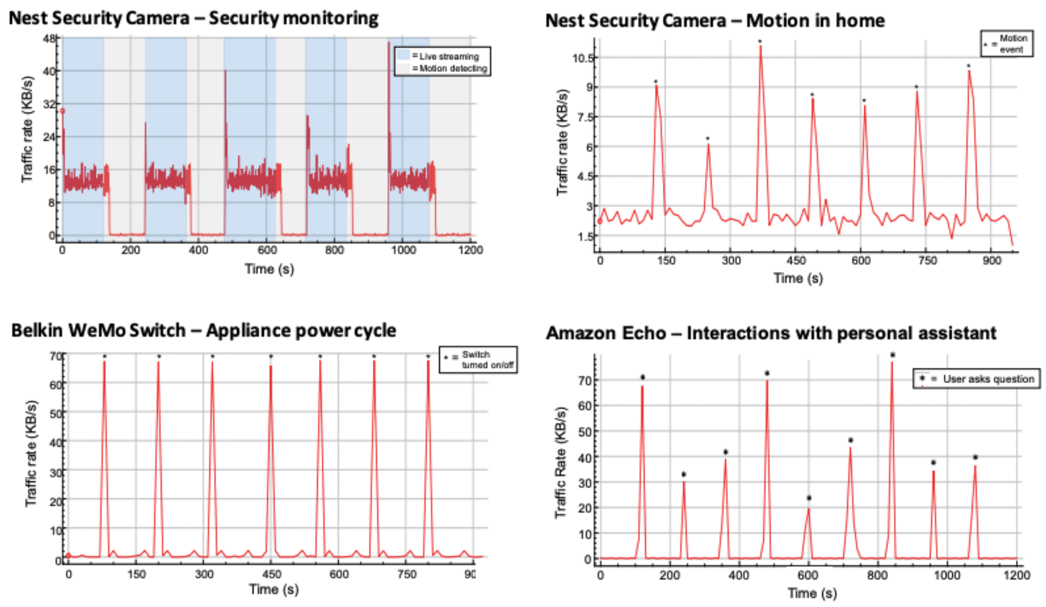


Fig. 2. Network traffic rates (send/receive) of 4 popular smart home devices [53].

5.1 Targeting Confidentiality

5.1.1 Side-Channel Attacks:

Attackers can exploit data leakage vulnerabilities to gain sensitive information about users. One such example was a side-channel attack described in 2015 [51]. The goal of the attack was to discover the keystrokes of a victim typing on a keyboard from their wrist position inferred by accelerometer and gyroscope data sensed from a smartwatch. The authors tested the attack on eight real users, each of whom was asked to type 300 different English words from a dictionary of 5000 words while wearing a smartwatch on their left wrist. The results showed that the keystroke detection rates for the left-most keys on the keyboard (Q, W, E, R, T, A, S, D, F, G, Z, X, C, V, B) were from 88% to 99% accuracy, while the keystroke detection rates for the right-most keys on the keyboard (Y, U, I, O, P, H, J, K, L, N, M) were from 3% to 5%. These keystroke detection rates allowed the authors to have a 50% chance to narrow down each possible word typed by 99.5% (in other words, for each word the user typed, the attack had a 50% chance of accurately eliminating 4976 words from the 5000-word dictionary of possible words). Unlike similar previous papers that explored side-channel attacks from sensor data leakage of smartphones, this paper was one of the first to present an attack from sensor data leakage of wearable devices with relative success, without requiring any training from the victim. In 2016, Wang et al. [52] improved the accuracy of the attack and tested it on real ATM machines with over a 93% success rate.

Alternatively, sensitive data can also be leaked through traffic analysis attacks. Copos et al. [54] presented a traffic analysis attack on encrypted smart home network traffic. The authors analyzed the network traffic of Nest Thermostat and Nest Protect (smoke and carbon monoxide detector) devices, and showed that potentially sensitive information about the state of the smart home could be discerned from the encrypted traffic. Specifically, the authors captured traffic to and from the Nest devices within the smart home network, and decrypted the WPA encrypted traffic so that they could identify the hosts that the devices frequently contact. However, the packets are still encrypted with SSL/TLS. By analyzing the types of requests (e.g., HTTPS, NTP, DNS, etc.) and the frequency of requests, the authors were able to discover when the devices switched between *Home* (home is occupied) and *Away* (home is unoccupied) modes. For example, there was a discrepancy in the frequency of NTP requests generated between when the Nest Thermostat was operating in *Home* mode and when it is in *Away* mode. From this information, the authors created a simple Support Vector Machine (SVM)-based learning model that achieved 81% accuracy (with zero false positives) in predicting the mode of the Nest Thermostat.

Apthorpe et al. [53] extended the work done in [54] by demonstrating that ISPs or other network observers could infer privacy sensitive in-home activities by simply analyzing the traffic rates generated from smart homes containing commercially-available IoT devices, even when those devices used encryption. As shown in Figure 2, the changes in network traffic rates (send/receive) of 4 popular smart home devices correspond with the devices' states. Clearly, an attacker aware of this correlation between smart home traffic rates and device states could easily distinguish user activities within the smart home. The authors present a traffic shaping defense method for preventing such traffic analysis attacks, which we detail in subsection 6.5.5.

Covert channel attacks presented in works [55] and [56] show that an attacker can use IoT devices to bypass security mechanisms, such as firewalls, traffic monitors, and information flow control systems, by routing sensitive data through covert channels. Yang et al. [55] presented a covert communication method called NICScatter, where a mobile device, infected with malware, backscatters surrounding radio frequency (RF) signals at varying intensities to transmit sensitive information retrieved from the device's memory to the attacker's phone. In order to launch this attack, the malware controls the impedance of a device's wireless network interface card (NIC). While the authors only tested their attack on laptop and smartphone NICs, this attack is very much possible in an IoT environment. In fact, Ronen et al. [56] showed that it was possible to create a covert channel with smart LED lights. The authors were able to misuse smart LEDs' APIs to cause the lights to oscillate between different intensities, in such a way as to be indistinguishable to the human eye, to surreptitiously transmit data to a light sensor.

While we discuss various countermeasures for side-channel attacks in subsection 6.2.1, here we point out that empirical testing of individual IoT devices and networks is a necessary first-step in mitigating side-channel attacks. To this end, Sachidananda et al. [57] presented an IoT testbed for various devices, such as smart home and smart wearables, to test them against a set of security requirements by performing customizable security tests. The testbed tests both hardware and software components of the IoT devices, and offers different types of testing environments which allows the authors to test various sensors in the devices (e.g., accelerometer, gyroscope, GPS, light, etc.). While the authors did not use the testbed to test how susceptible devices were to side-channel attacks, this type of holistic testing environment is crucial to preventing attacks that exploit data leakage vulnerabilities.

5.1.2 Brute-Force Attacks:

Brute-force, or dictionary attacks, can easily be launched against IoT devices, especially those using weak credentials. As we discussed in subsection 4.3.3, there are a significant number of open IoT devices with no, default, or weak credentials, a vulnerability that can be exploited fairly easily. One of the most notorious examples of such exploitations is the Mirai malware which at one point infected over 300,000 IoT and embedded devices all over the world [44]. The botnet that was formed from the infected devices was used to launch devastating DDoS attacks against multiple targets (Krebs on Security, OVH, and Dyn), where the Krebs on Security attack exceeded 600 Gbps in volume, making it the largest DDoS attack recorded at the time. A study conducted

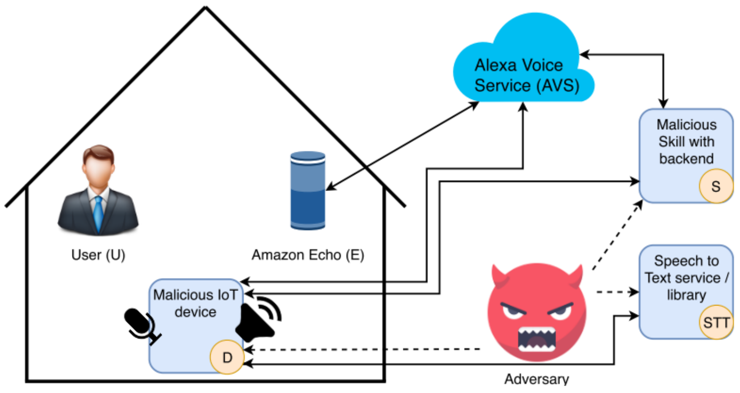

Alexa Voice Service (AVS)
Malicious Skill with backend
S
User (U)
Amazon Echo (E)
Speech to Text service / library
STT
Malicious IoT device
D
Adversary

Fig. 3. The main components of the Lyexa attack [60].

by Cetin et al. [58] showed that while Mirai could be removed by rebooting an infected device, simply rebooting the device will not fix the underlying problem as the device remains vulnerable to infections once it comes back online. As the authors note, removing the underlying problem would require affected users to change default passwords, or update the firmware — measures that are much more complicated than a mere reboot. In recent years, the Mirai malware has been evolving into potentially much more dangerous strands, such as the Hajime malware [46], whose real purpose remains a mystery as it has no attacking code for launching DDoS attacks.

5.2 Targeting Integrity

5.2.1 Voice-Command Injection Attacks:

In recent years, the IoT security research community has focused much attention on voice-command injection vulnerabilities. As a result, many interesting attack papers exploiting these vulnerabilities have been published, especially in the last three years. In this subsection, we present three such papers, one on inaudible command attacks [59], and two on skill squatting attacks [49, 60].

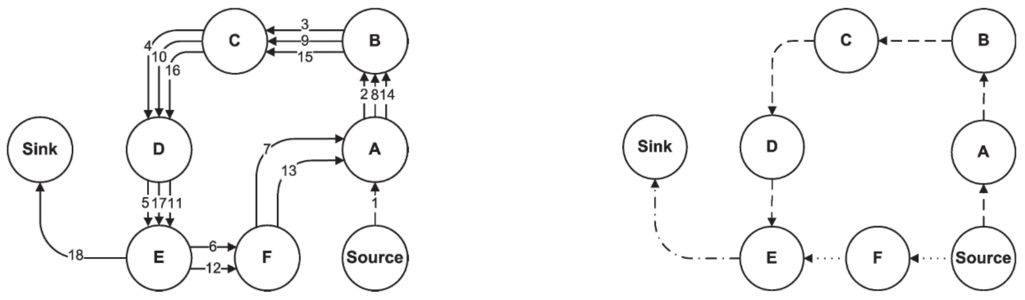
In 2017, Zhang et al. [59] designed a completely inaudible attack, which they called the DolphinAttack, that modulates voice commands on ultrasonic frequencies (e.g., $f > 20$ kHz) using the amplitude modulation technique (AM) to achieve inaudibility. By leveraging the nonlinearity of electret condenser microphone (ECM) and MEMS microphone circuits, the modulated low-frequency audio commands can be successfully demodulated, recovered, and interpreted by the speech recognition systems. The authors showed that with the DolphinAttack, an adversary can inject malicious and inaudible voice commands into major speech recognition systems, including Siri, Google Now, and Alexa.

Kumar et al. [49] and Mitev et al. [60] presented skill squatting attacks against Amazon Alexa devices. As we explained in subsection 4.3.6, Kumar et al. not only discovered the voice squatting attack (VSA), but also created a variant of the attack, called spear skill squatting, which targeted the accents and speech patterns of specific demographic groups. Mitev et al. created a skill squatting-based man-in-the-middle attack called Lyexa. This attack consists of four main components, as depicted in Figure 3:

(1) Commands issued by a user U are inaudibly jammed and simultaneously recorded by a malicious device D. When user U speaks the wake word (e.g., "Alexa"), both the benign Alexa-enabled device E and malicious device D are activated. Benign device E starts listening for subsequent commands and malicious device D starts jamming this command with ultrasound modulated noise. Therefore, the command issued by user U cannot be understood by benign device E.
(2) When user U finishes speaking the command, D immediately stops jamming and inaudibly injects a skill invocation command with malicious skill S's invocation name to benign device E, which is still listening for a command. E will forward the injected audio command to the Alexa Voice Service (AVS), which interprets it and invokes malicious skill S. Here the authors assume that Amazon has verified and accepted malicious skill S into its skill repository for all Alexa users to use. Malicious skill S is now running and can return arbitrary text to be echoed through benign device E in Alexa's voice. The inaudible command injection is realized by utilizing the amplitude modulation technique, as described in the aforementioned DolphinAttack.
(3) In order to hide the attack, malicious device D may want to fetch the benign data from the benign skill that user U originally wanted to invoke with his command. To do so, D begins a new session with AVS and forwards user U's recorded command to it, as if it was a benign Alexa-enabled device. AVS will then invoke the requested skill, and pass it user U's command. Subsequently, the benign skill will return a textual representation of the response to AVS, which will then use the Alexa text-to-speech (TTS) engine to create a voice audio file out of the response, and forward it to malicious device D. D will send the audio file containing the response to a speech-to-text (STT) service for it to be transcribed back into text. Now malicious device D knows what data the victim wanted to fetch from the requested skill.
(4) Lastly, malicious device D may modify the response data and echo it back to user U via benign device E. After arbitrarily modifying the response data received from the benign skill in the previous step, D passes this text to the backend of malicious skill S. S then passes the received text to the already established AVS session. AVS will then create an audio file with the help of Alexa's TTS engine. This audio file is passed to benign device E to be played back to user U, thus completing the attack.

5.2.2 Event Spoofing Attacks:

Event spoofing is another attack targeting the integrity of IoT networks. In an event spoofing attack, an attacker can create and propogate a seemingly legitimate event to devices in the home, causing them to react in some way that benefits the attacker. Fernandes et al. [61] analyzed the SmartThings framework to determine how susceptible it is to event spoofing attacks. They showed that the SmartThings framework does not enforce access control around raising events, and does not offer a way for triggered SmartThings applications to verify an event's integrity or origin. To prove this, the authors launched two event spoofing attacks. In the first attack, they wrote a malicious SmartThings application that spoofed the location of the smart home owner, causing the SmartThings Hub, which controlled the smart home's lights, to turn off *Vacation* mode. The authors were able to launch this attack because SmartThings does not have any security controls around the *sendLocationEvent* API. In the second attack, they wrote a malicious SmartThings application that spoofed a carbon monoxide detection event, causing the legitimate SmartThings alarm application to sound a siren alarm.



(a) An honest route would exit the loop immediately from node E to Sink, but a malicious packet makes its way around the loop twice more before exiting.

(b) Honest route is dotted while malicious route is dashed. The last link to the sink is shared.

Fig. 4. Examples of the carousel (a) and stretch (b) attacks [62].

5.3 Targeting Availability

5.3.1 Selective-Forwarding Attacks:

In a selective-forwarding attack, a malicious device, which is located on the route between the sources and destinations of certain connections, launches a denial-of-service (DoS) attack by forwarding only a subset of the packets that it receives to the destination [33]. Although this attack is primarily targeted to disrupt routing paths, it can be used to filter any type of traffic, no matter the protocol. For example, an attacker could forward all RPL control messages (thereby preventing devices from detecting routing inconsistencies), and drop all other traffic. In this case, RPL's self-healing mechanisms will not be able to detect the DoS attack, as the control messages are still being forwarded.

Selective-forwarding attacks can be coupled with sinkhole attacks to become even more powerful. In sinkhole attacks, a malicious device advertises an artificial, but attractive routing path, thereby causing many nearby devices to route traffic through it [33].

It is generally fairly difficult to defend against all selective-forwarding attacks. However, there are some security mechanisms that can minimize the impact of these types of attacks. For example, through encryption, legitimate devices can ensure that an attacker cannot distinguish between different types of traffic, thus forcing the malicious device to either forward all or none of the traffic it receives. To this end, devices can use IPsec to secure RPL control messages, which we explain in more detail in subsection 6.3.1. Additionally, analysis of application-layer traffic can help detect if any application traffic is lost, and devices may report such losses to the underlying RPL system in order to improve path quality by finding a route that bypasses the malicious device.

5.3.2 Battery-Draining Attacks:

Battery-draining, or energy-depleting attacks are another way attackers can target the availability of IoT devices. Vasserman et al. [62] present various battery-draining attacks, called Vampire attack, which target wireless ad hoc sensor networks. Vampire attacks use routing protocols to permanently disable networks by depleting devices' battery power. These attacks do not depend on particular protocols or implementations, but rather rely on the properties of many popular classes of routing protocols. These properties include link-state, distance-vector, source routing, and geographic and beacon routing. For example, Figure 4 depicts two types of Vampire attacks: the carousel and stretch attack. The authors show through simulation that depending on the location of the adversary in the network, network energy expenditure increases from between 50 to 1000 percent. While the authors present Vampire attacks on ad hoc sensor networks, smart home networks could just as easily fall prey to such attacks, as many smart home devices, many of which are mobile and battery-powered, use routing protocols similar to those used in ad hoc sensor networks, such as AODV (used by ZigBee devices).

Chiariotti et al. [63] also analyze battery-draining attacks from a game-theoretic perspective. The authors use game theory to model a smart IoT device defending itself from a similarly energy-constrained jammer. Both devices are assumed to be rational actors, and their interactions can be modeled as a zero-sum game The mathematical properties of this zero-sum game are exploited to find an effective defense solution.

5.3.3 *DDoS Attacks:*

Lastly, we describe unique DDoS attacks presented in recent years against IoT devices and networks. In 2018, Soltan et al. [64] and Ronen et al. [65] presented DDoS attacks that could potentially disrupt entire cities or countries. Soltan et al. demonstrated that a botnet of high wattage IoT devices, such as air conditioners and heaters, gives an adversary the unique ability to launch large-scale coordinated attacks on a city's or country's power grid. By synchronously switching on/off compromised high wattage IoT devices, and thereby manipulating the total power demand, an adversary could significantly disrupt a power grid's normal operation. Through simulation of the attack on Poland's power grid, the authors show that this attack, in the extreme case, could cause large-scale, country-wide blackouts. With a botnet of only 210 devices, the attack could initiate a cascading failure resulting in a power outage covering 86% of the country.

Ronen et al. described a new type of malware worm that infects IoT devices that are in close proximity with one other, thereby rapidly spreading over large areas, given that the density of compatible IoT devices exceeds a certain amount (i.e., critical mass). The authors tested the malware on the popular Philips Hue smart lamps. The worm spreads by jumping directly from one lamp to its neighboring lamps, using only the built-in ZigBee wireless mesh connectivity. The infection can begin spreading by a user plugging in a single infected bulb anywhere in a city, and the worm can catastrophically spread throughout the city within minutes. The malware enables an attacker full control of the city's smart lamps – he/she can turn all of the lights on or off, permanently brick them, or use them in a massive DDoS attack. To make such an attack possible, the authors had to find a way to:

(1) remotely break the connection between already installed lamps and the hubs controlling them (i.e., remove them from their networks), and
(2) perform over-the-air firmware updates.

As we mentioned in subsection 4.2.2, the authors exploited the known vulnerability in the ZLL protocol to overcame the first problem. To solve the second problem, they developed a side-channel attack to extract the global AES-CCM key (for each device type) that Philips uses to encrypt, and authenticate new firmware.

Two examples of real-world DDoS attacks targeting IoT devices are the BrickerBot [66] malware, and the attack on Finland's central heating and warm water circulation systems [67]. In the span of about one year (November 2016 to December 2017), the BrickerBot malware managed to permanently destroy over 10 million IoT devices, in the hopes, according to its author, to prevent those bricked devices from being infected with the Mirai malware and launch DDoS attacks themselves. Also in November 2016, a DDoS attack halted heating distribution in two buildings in Finland. While the attack did not target any IoT devices directly, but rather the computers that controlled the devices responsible for central heating and warm water circulation, this attack prevented the devices from functioning normally. Thankfully the attack was easily mitigated by limiting network traffic, and no one was hurt. However, these real-world attacks that target critical infrastructure should make us wary about the potential damage that IoT-based attacks can cause.

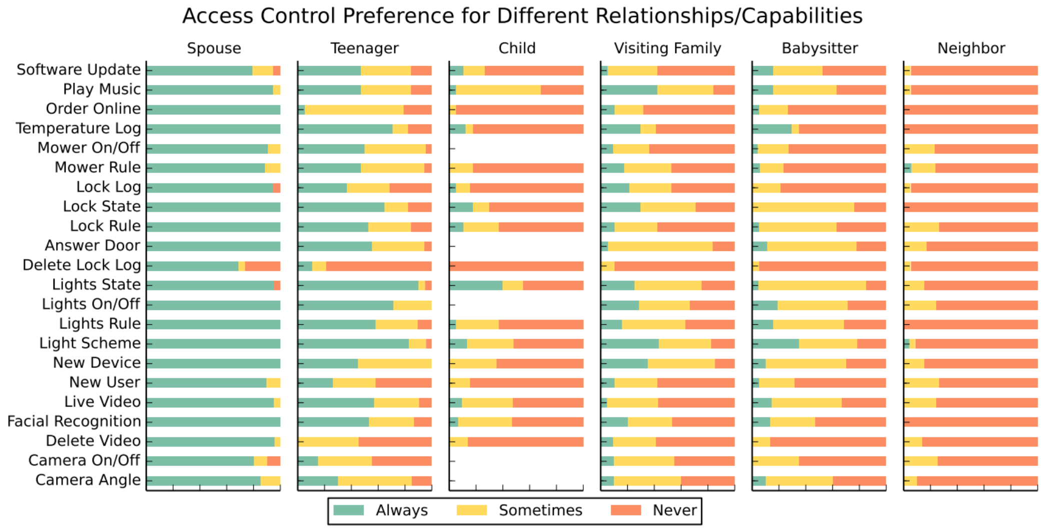


Fig. 5. Desired access-control policies given different capabilities and relationships [68].

6 COUNTERMEASURES

In this section, we describe various recent countermeasures to the aforementioned IoT vulnerabilities and attacks, and group them into five categories:

- access control & authentication,
- intrusion detection & mitigation,
- security protocols & frameworks,
- software reliability, and
- identification & management.

Most of the countermeasures we cover in this section attempt to prevent, detect, and/or mitigate the attacks we described in section 5.

6.1 Access Control & Authentication

6.1.1 User Studies Related to Access Control:

In recent years, researchers have found the need for reenvisioning access control and authentication for multi-user-per-device environments, like the smart home. As He et al. [68] noted in their 2018 study of access control and authentication in the home IoT, current authentication methods for the home IoT appear transplanted from smartphone and desktop paradigms, which for the most part, assume a single-user-per-device environment. However, in IoT environments, such as the smart home, multiple users with complex social relationships may interact with a single device. Therefore, He et al. conducted a 425-participant online user study and found major differences in the participants' desired access-control policies for different capabilities within a single device (e.g., updating software, turning lights on/off, turning cameras on/off, adding new user, etc.), as well as based on who is trying to use that capability (e.g., spouse, teenager, child, visiting family, babysitter, neighbor, etc.). Figure 5 summarizes the participants' desired access-control policies given different capabilities and relationships. This study allowed them to pinpoint various contextual factors (e.g., time of day, location of user, location of device, who is near by, etc.) that, along with capabilities and relationships, dictate the specification of more complex, yet desired, access-control policies.

One year later, Zeng et al. [69] applied the complex access-control policies derived from [68], among other design principles from other studies, to create an access control system (in the form of a mobile application) for the smart home. The application included four types of access controls:

- Role-Based Access Control: Each user is assigned a role — admin, child, or guest. Only admins are allowed to change access control policies, add new users, and organize the devices.
- Location-Based Access Control: Users can be restricted from using devices if they are not physically near the device.
- Supervisory Access Control: Allows a user who may be restricted from using a device, to use the device, if and only if another (authorized) user is nearby.
- Reactive Access Control: If a user attempts to use a device they do not have permission to use, the application will ask a more privileged user for permission in real-time, by sending a notification asking them to approve or deny the request.

The authors further tested their application in seven households in the Seattle metropolitan area. Although the users expressed that they had a strong need for certain access controls in their homes before the study, only a few ended up using the application. While usability was one factor that went into this, most participants were either unconcerned about restricting access to mundane devices, or that existing social norms and trust in their households checked against bad behavior. For example, children were trusted to follow rules, roommates respected each others' spaces, and for the most part, people were not concerned about information revealed by the smart home when it matched their household's privacy norms. The users were also willing to accept (multi-user) security and privacy risks posed by usage of devices because of the convenience and utility lost by using the access control mechanisms. The study reemphasized the notion that the design of security and privacy features for a smart home must work with, and not limit, a user's primary use case for the smart home.

6.1.2 *Access Control of Overprivileged Applications:*

As described in [61], IoT home automation platforms have overprivilege problems. For example, in the SmartThings platform, an application may ask for a set of capabilities (device functionalities the application needs to use), and the user must choose the devices that can perform those capabilities and give the application access to those devices. However, this type of implicit authorization lacks fine granularity and context awareness — the application is automatically granted unlimited access to the entire device (even if it asked for only a subset of all the capabilities provided by the device), and allowed to subscribe to all of the device's events (the application will know when other applications use any capability provided by the device).

Many papers in the last three years, such as [70–73], have introduced access control mechanisms to tackle the overprivilege problem and the risks associated with this problem. Tian et al. [70] proposed the SmartAuth authorization mechanism which used insights from code analysis, and natural language processing (NLP) of app descriptions to provide a new solution to the overprivilege problem. They designed a new policy enforcement mechanism, compatible with current home automation frameworks, that enforced complicated, context-sensitive security policies with low overhead. Essentially, SmartAuth automatically collects security-relevant information from an application's code and description, and generates a user-friendly authorization interface, through which a user can specify specific capabilities the application has access to per device. Similarly, Jia et al. [71] proposed ContexIoT, a context-based permission system that provides contextual integrity by supporting fine-grained context identification for sensitive actions performed by applications, and runtime prompts with detailed context information to help users perform effective access control. To support existing applications, the authors developed an application patching mechanism that can convert unmodified commodity applications to ContexIoT-compatible applications. Schuster et al. [72] developed a system called environmental situation oracles (ESOs) to enforce IoT access control for situational constraints for IoT environments. This system, like ContextIoT, also supports fine-grained context identification, and considers the state of the device, along with the location of the users, to determine a valid access request. Rahmati et al. [73] presented Tyche, which introduced the notion of risk-based permissions for IoT systems. With risk-based permissions, device operations are grouped into units of similar risk. Users can thereby grant applications access to devices at that risk-based granularity.

The major drawback to each of these systems is their lack of realistic policy enforcement assumptions. Each paper assumes that their access control systems has access to each application's source code and can automatically patch each application [70, 71], or can coordinate with the developers of each application to rewrite their code to more easily enable more fine-grained access control [73]. If these assumptions do not hold in reality (which they most likely will not as most home automation platforms and applications are closed-source), then the access control systems presented in the papers will be somewhat useless.

6.1.3 Default-Off Access Control:

Hong et al. [74] argued that IoT device communications should be default-off and only explicitly enabled when absolutely needed because IoT applications and devices, unlike traditional applications and computers (e.g., web browser or laptop), serve narrowly defined purposes. The authors proposed, Bark, a policy language and runtime for specifying and enforcing minimal access permissions in IoT networks by whitelisting access for IoT applications when communication is default-off. Bark constructs access control policies through natural questions (e.g., who, what, where, when, and how), and transforms them into transparently enforceable rules for IoT application protocols. However the authors assume that the gateway, where policies are being enforced, can observe all traffic between devices, applications, and remote servers. This means that the system would not be able to enforce rules for device-to-device communication in protocols that allow for mesh networking (e.g., 6LoWPAN, ZigBee, Bluetooth).

6.1.4 Access Control with Delegation:

The transfer of device usage and/or capability access permissions between users is another challenge related to access control in the smart home environment. Le et al. [75] attempted to tackle the problem of delegating permissions by proposing a lightweight authorization protocol that allows a user to easily transfer his/her access rights to smart home devices. The protocol works by transferring access rights to a device in the form of a Bloom filter with the help of secured hashing to prevent the permission from being forged. Due to the properties of Bloom filters, items cannot be removed from the Bloom filter, and therefore, a user cannot recreate a permission higher that what he/she is holding, but can still transfer lower permissions to other users.

6.1.5 Data Flow Control:

As discussed in subsection 4.3.5, the lack of data flow control is a significant vulnerability in smart home environments. We analyze five recent works [76–80] related to controlling how data flows between sources and sinks.

Fernandes et al. [76] introduced a method called FlowFence for restricting an application's access to sensitive IoT data. The authors noted that while IoT frameworks use permission-based access control for data sources and sinks, they do not control *flows* between the authorized sources and sinks. To explain this notion further, let's consider the example of an application that unlocks a door based on the person's face. It extracts features from the person's face and compares them to the features belonging to authorized people, checks the current state of the door, unlocks the door, and sends a notification to the home owner over the Internet. The user may have given this application permission to use the camera, unlock the door, and access the Internet, however, there is no way for the user to ensure that the application does not leak camera data to the Internet. To be supported by FlowFence, an application developer must split his/her application into two parts — one part is code that deals with non-sensitive data, and the other that does. The code that deals with sensitive data is "quarantined". FlowFence then runs the application in a sandbox to generate a data flow graph, which is analyzed to determine if any data flowing through the quarantined code violates any user defined flow rules (e.g., no data flowing through quarantined code should be sent over the Internet).

Fernandes et al. applied rule-based flow tracking and control properties for action integrity for trigger-action IoT platforms [77] as an extension of their work in [76]. They noted that IFTTT, and other trigger-action platforms, can have overprivileged access to smart home devices, which can cause serious attacks if the authorization tokens, which give the platforms the ability to control the devices, are compromised. For example, a user could allow a IFTTT application to change the color of the smart lights in their living room if a friend posts a picture on FaceBook. In order to create this type of application, the setup process would include the user allowing the smart lights' action service (which could be located on the devices themselves, the hub controlling the devices, or in the cloud) to provide an authorization token to the application. Although the application should only control the lights' color, with the authorization token it has full control over the lights. Therefore, if the application accidentally leaks the authorization token or if the platform itself is compromised, this would potentially give an attacker full control over the lights. In [77], Fernandes et al. present a security principle that ensures that an attacker who controls a compromised trigger-action platform can only invoke actions and triggers that are specified in the rules that users have created (e.g., can only control the color of the lights), can invoke actions only if it can prove to an action service that the corresponding trigger occurred in the past within a reasonable amount of time, and cannot secretly tamper with any data passing through the platform.

Similarly to [76] and [77], Bastys et al. [78] and Celik et al. [79] evaluate the runtime execution of applications to enforce policy-based information flow control for trigger-action platforms. Bastys et al. studied a dataset of 279,828 IFTTT applets and found that that 30% of the applets were at risk of violating privacy. To tackle this problem, they argued and demonstrated that information flows should and can be secured in applications by state-of-the-art information flow tracking techniques. Celik et al. [47] presented IoTGuard, a dynamic, policy-based enforcement system for IoT. The purpose of IoTGuard is to protect users from unsafe and insecure device states by monitoring the behavior of trigger-action platform applications. It requires extra logic to be added to an application's source code to collect runtime information. Similarly to [76], that runtime information is used to create a dynamic model that represents the runtime execution behavior of the application. Lastly, it finally enforces relevant safety and security policies on the dynamic model of the individual application or sets of interacting applications.

Most of these solutions [47, 76, 78] are somewhat impractical for the real-world due to the fact that they require applications to be modified. Furthermore, the aforementioned solutions also require applications to be analyzed thoroughly before they are used, in order to discern their runtime execution behavior. However, they do provide an important first-step in addressing the problem of lack of data flow control in IoT environments.

Melara et al. [80] introduced an access control system that did not require the modification of applications. Because most IoT applications include third-party libraries that may contain vulnerabilities, the authors aimed to provide intra-process access control for applications, especially for those that generate and analyze highly privacy-sensitive data. The access control system, called Pyronia, enforces function-granular resource access policies via runtime and kernel modifications.

Therefore, while Pyronia does not need to modify the applications themselves, each device looking to deploy Pyronia needs to run a custom Linux kernel.

6.1.6 *Recently Proposed Authentication Mechanisms:*

While conducting this survey, we found that recently proposed authentication mechanisms mostly dealt with biometric factors. For example, many papers [81–85] developed unique touch-based authentication mechanisms for wearable or smart home devices. Alternatively, Lin et al. [86] presented a continuous authentication system based on geometric and non-volitional features of cardiac motion.

Of all the interesting authentication methods we found, the continuous authentication mechanism for voice assistants presented by Feng et al. [87] is the most relevant to our survey. The authors present VAuth, the only system that we found that provides continuous authentication for voice assistants. VAuth collects the body-surface vibrations of a user, and matches it with the speech signal received by the voice assistant's microphone. VAuth can fit inside things that people normally wear, such as eyeglasses, earbuds, and necklaces, Such a system can guarantee that the voice assistant only executes the commands that originate from the voices of authorized users. The authors evaluated the system on 18 users and 30 voice commands, and achieved a detection accuracy of 97% with less than 0.1% false positives, regardless of VAuth's position on the user's body, the user's language, the user's accent, or the user's mobility. This authentication mechanism would help prevent against the inaudible voice-command injection attacks we described in subsection 5.2.1.

6.2 Intrusion Detection & Mitigation

6.2.1 *Detecting Side-Channel Attacks:*

We discussed side-channel attacks in subsection 5.1.1, and present a recent solution to this type of attack. Chaman et al. [88] analyzed stealthy eavesdroppers that employ passive receivers that only listen and never transmit any signals. The authors show that even passive receivers leak RF signals over the wireless medium. As a result, they were able to detect passive receivers present in a network, even when the leaked RF signals were buried under ongoing transmissions.

6.2.2 *Detecting the Execution of Malicious Processes:*

In subsection 5.1.2, we discussed brute force attacks, and how they can lead to the creation of large-scale botnets. Breitenbacher et al. [89] presented a host-based anomaly detection system called HADES-IoT, to detect if unauthorized processes, such as those belonging to IoT malware, are attempting to execute. HADES-IoT has proactive detection capabilities, and provides tamper-proof resistance. It is installed inside a device's kernel space, and can be deployed on a wide range of Linux-based IoT devices. HADES-IoT is first pre-compiled and delivered to the device. The kernel's initialization file is modified accordingly to ensure that HADES-IoT is always executed when the device is booted. Once executed, HADES-IoT monitors and collects information about all calls to the *execve* system call. It only allows a set of whitelisted processes, which it learns of during a profiling phase, to call the *execve* system call. The authors deployed HADES-IoT on seven IoT devices and achieved 100% effectiveness in detecting recent IoT malware strains, such as VPNFilter and IoTReaper, while on average, requiring only 5.5% of available memory, but in some cases incurs almost 40% extra CPU usage.

6.2.3 *Detecting Routing Attacks:*

While in the past four decades there has been a plethora of papers focused on the detection and mitigating of routing attacks (which we described in subsection 5.3.1) in wireless sensor networks (WSNs), we focus on a more recent IDS for 6LoWPAN routing attacks. Raza et al. [90] presented SVELTE which detected sinkhole attacks in 6LoWPAN networks that occur through

RPL rank manipulation. It has two main modules running on the border router of a 6LoWPAN network: 6LoWPAN Mapper (6Mapper) that gathers information about the network, and an intrusion detection module that checks the rank inconsistency in data obtained by 6Mapper to detect sinkhole attacks. The 6Mapper sends *probing* messages to nodes in the entire network at regular intervals (e.g., 2 minutes). Each node then sends a *response* message to the 6Mapper, which includes its node ID, node rank, parent ID, and all of its neighbors' IDs and ranks.

Unfortunately, SVELTE's probing mechanism can increase the network overhead, device power consumption, and the latency of detecting sinkhole attacks. Every probe from the border router increases the network overhead. Every response from a device consumes more power. Worst of all, SVELTE has a dilemma in choosing the probing interval: a short interval leads to a low latency in detecting sinkhole attacks, but a large overhead due to frequent probing and responding; a long interval results in a low overhead, but a high latency in detecting sinkhole attacks.

6.2.4 Detecting Event Spoofing Attacks:

Sikder et al. [91] and Birnbach et al. [92] introduced detection methods to address event spoofing attacks, which we described in subsection 5.2.2. Sikder et al. proposed a context-aware intrusion detection system which observes changes in sensor data for different user tasks, and created a contextual model to distinguish between benign and malicious sensor behavior. The detection system utilizes three different machine learning-based detection mechanisms to detect malicious behavior associated with sensors. Alternatively, Birnbach et al. proposed a detection method that attempts to verify physical events using data from an ensemble of sensors that are commonly found in smart homes. The authors show that this approach both protects against event sensor faults, and sophisticated attacks.

6.2.5 Detecting Voice-Command Injection Attacks:

While we described an authentication mechanism to prevent inaudible voice-command injection attacks in subsection 6.1.6, He et al. [93] instead presented a detection and mitigation system to tackle such attacks. Specifically, the system they presented detects inaudible voice commands by analyzing the frequency of voice commands picked up by the microphone. Once an inaudible voice command is detected, the authors apply active noise cancellation techniques to negate the frequency at which the inaudible command is carried.

6.2.6 Detecting Attacks via Encrypted Traffic Analysis:

As described in subsection 5.1.1, sensitive data can be leaked through traffic analysis, even if that traffic is encrypted. et al. [94] employed the same techniques used in traffic analysis attacks to create an anomaly detection system. The goal of the system, called HoMonit, is to detect misbehaving applications. Specifically, HoMonit leverages the leakage of packet size and inter-packet timing to infer device activities. The inferred activities are compared to the expected program logic of the application to identify anomalies. and then compares the inferred event sequences with the expected program logic of the application to identify misbehaviors (e.g., event spoofing). At the core of HoMonit is a Deterministic Finite Automaton (DFA) matching algorithm. The authors argue that every application follows a certain DFA model, in which each state represents the status of the application and the corresponding smart devices, and the transitions between states indicate interactions between the application and the devices. HoMonit includes techniques to extract the program logic from an application's source code, and automate the DFA construction process. HoMonit applies the DFA matching algorithm to compare the inferred device activities with the expected DFA transitions of each application. If the DFA matching fails, a misbehaving application is detected.

6.2.7 Detecting Hidden Inter-Application Interactions:

We discussed inter-rule vulnerabilities in subsection 4.3.5, which are essentially the same as the inter-application interaction vulnerabilities that Ding et al. [95] analyzed. Unlike in traditional networks, in an IoT environment, an application can control a device to change the physical environment, which may in turn trigger another application to command another device to react to that change. Therefore, if an application is not aware of all of its possible interactions with other applications, unexpected interactions could be exploited and triggered by attackers. The authors proposed a framework called IoTMon that captures all potential physical interactions across applications, and allows users to create safe interaction controls. IoTMon performs intra-application interaction analysis using static program analysis and NLP to extract necessary application information, such as conditions, actions, and devices, for generating inter-application interaction chains. After identifying all interaction chains, IoTMon uses a risk analysis mechanism, similar to [48], to evaluate the risk of identified inter-application interaction chains. The authors evaluated their framework on 185 official SmartThings applications, and found that 162 hidden interaction chains exist among these applications, where 37 of them are highly risky and could be potentially exploited to launch serious attacks.

In the same vein, Wang et al. [96] argued that with the rise in popularity of home automation, devices and applications may be chained together in long sequences of trigger-action rules. These chains could be so complex that from an observable symptom (e.g., an unlocked door), it may be impossible to identify the distantly removed root cause (e.g., a malicious app). Thus, the authors presented ProvThings, an auditing system that performs efficient automated instrumentation of IoT apps and device APIs in order to capture complex chains of interdependencies between different apps and devices.

6.3 Security Protocols & Frameworks

6.3.1 Improvements to 6LoWPAN Security:

As we described in subsection 4.2.2, 6LoWPAN enables the use of IP in IEEE 802.15.4 low power and lossy wireless networks, and it uses CoAP as its application-layer protocol, which sits on top of UDP with DTLS at the transport-layer. While DTLS supports a wide range of cryptographic primitives for peer authentication and payload protection, it was originally designed for networks where machine resources and message length were not critical constraints. Therefore, Raza et al. [97] proposed Lithe — CoAP with a compressed version of DTLS. They employed the same techniques used in 6LoWPAN to compress IP headers. Through DTLS header compression, the authors argued that they achieved energy efficiency by reducing the overall IP/UDP datagram size (due to the fact that transmitting packets requires more energy than computation). They also argued that they minimize 6LoWPAN fragmentation that is applied when the size of a datagram is larger than the link layer MTU (127 bytes for the case of IEEE 802.15.4), thereby making 6LoWPAN less vulnerable to fragmentation attacks. Similarly, Raza et al. [98] also proposed to use IPsec to secure communication between devices through IPsec compression. However, it is unclear from these papers ([97, 98]) if compressing the respective security protocols reduces the overall security of a network and opens it up to potential attacks. Furthermore, neither of the protocols presented in these papers have been standardized as of yet, and therefore, it is unlikely that they are being used in the real-world.

6.3.2 Recent Encryption Protocols:

IoT applications often store and share data in the cloud, and therefore, sensitive data about users could be observed by cloud operators, or leaked if the cloud infrastructure is not appropriately secured. In order to tackle the problem of sensitive data leakage in the cloud, Shafagh et al. [99]

designed a data protection framework, called Talos, where the cloud stores encrypted data while allowing applications to perform queries over the encypted data, without having to first download and decrypt the data. The proposed solution utilized crpytographic techniques, such as Partial Homomorphic Encryption (PHE) and order-preserving encryption, to allow computations to be carried out on encrypted data. Furthermore, the authors later extended Talos to not only be able to query encrypted data, but to also share it [100].

Due to the resource constrained nature of some IoT devices, and the resource consumption incurred by cryptographic primitives, Li et al. [101] proposed a key-free communication method for IoT networks, which they called HlcAuth. Essentially, HlcAuth utilized challenge-response mechanisms for mutual authentication between the gateways and smart devices without key management. Through real-world evaluation, the authors showed that HlcAuth can defend against replay attacks, message-forgery attacks, and man-in-the-middle attacks. However, for HlcAuth to work, the authors assumed that attackers are not within a certain range (at least 4.2 meters) of the gateway node.

6.4 Software Reliability

6.4.1 Reliable Patching and Update Mechanism:

Along with insecure firmware images, one of the major vulnerabilities facing IoT devices is the lack of reliable patching and update mechanisms, as described in subsections 4.3.1 and 4.3.2. Mainly to address the issue of patching in IoT environments, Simpson et al. [41] proposed a central security manager that can be installed on top of a smart home's gateway router. The security manager is aware of the status of all devices in the home, by observing all network traffic leaving and entering the network. By having knowledge of each devices status, and keeping track of reported vulnerabilities, the security manager could potentially intervene as needed to deter or alleviate many types of security risks. Modules can be built on top of the manager to offer convenient installation of software updates, filter traffic that might be malicious, and strengthen authentication for legacy devices. As the authors elaborate, the proposed modules can help improve security for the IoT devices at different stages in the patching process:

(1) Stage 1: Vulnerability discovery but no patch yet. In this case, the goals of the security manager are to identify the affected devices, filter attack flows to these devices, reduce the impact of default passwords, mitigate cross-site request forgeries, secure vulnerable SSL/TLS connections, block vulnerable APIs on the affected devices, and alert the user.
(2) Stage 2: Acquiring the patch. In this case, the main goal of the security manager is to apply the patch as quickly as possible by alerting the user. If the device is offline or in use, the security manager will pre-fetch the patch and apply it later, while continuing to protect the device as described in Stage 1.
(3) Stage 3: Applying the patch. In this case, the goal of the security manager is to simply apply patches when it is safe to do so.
(4) Stage 4: Device compromised. In this case, the goal of the security manager is to prevent the compromised device from harming other devices in the network.

The security manager is a proof-of-concept, as implementation details of the security manager and modules are not discussed in the paper.

6.4.2 Software Testing Through Fuzzing:

One method to discover security flaws in software is through fuzz testing. Chen et al. [102] presented IoTFuzzer, an automatic, blackbox fuzzing framework designed specifically for detecting memory-corruption flaws in IoT firmware, without the need to access firmware images. Unlike prior approaches, IoTFuzzer runs a protocol-guided fuzz that does not need to reverse-engineer the protocol. Instead, it performs dynamic analysis to identify the content inside the application that forms the messages to be delivered to the target device, and automatically mutates such content during the runtime so as to use the application's program logic to produce meaningful test cases for probing the target firmware. The main drawback with this paper is that it cannot precisely locate software flaws, but simply reports the presence of a problem (through a crash in the application triggered by a test case).

6.5 Identification & Management

6.5.1 Identification Through Traffic Analysis:

The discovery and identification of IoT devices is a necessary first-step to monitor and protect those devices. Sivanathan et al. [103] analyzed the problem of classification of IoT devices based on their network traffic characteristics. The authors first instrumented a real-world IoT environment consisting of 28 devices. The devices included smart cameras, lights, plugs, motion sensors, appliances, and health-monitors. They then collected data from this environment for a period of 6 months, which they made available to the IoT research community. Next, they identified key statistical attributes such as activity cycles, port numbers, signaling patterns, and cipher suites, which gave them insights into the underlying network traffic characteristics of the IoT devices. The authors further developed a multi-stage machine learning-based classification algorithm, and demonstrated its ability to identify, with high accuracy, specific IoT devices based solely on their network behavior.

Alternatively, as Feng et al. [104] argued, manual device annotation impedes large-scale discovery, and device classification based on machine learning requires large training data with labels. Therefore, the authors proposed an Acquisitional Rule-Based Engine (ARE), which can automatically generate rules for discovering and annotating IoT devices without the need of any training data. ARE creates these rules by using application-layer responses from IoT devices and product descriptions from websites for device annotations. The authors define a transaction as a pair of textual units, consisting of the application-layer data of an IoT device, and the corresponding description of an IoT device from a website. To collect the transaction set, ARE uses the association algorithm to generate rules of IoT device annotations in the form of (type, vendor, and product). In testing ARE in the real-world for a 2-month period, the authors were able to discover nearly 2,000 compromised IoT devices among 12,928 IP addresses.

6.5.2 Secure Logging:

Logging is a popular method used by network operators to manage devices. However, large amounts of personal information could be generated in IoT environments, such as smart homes. Logging such information could potentially be dangerous if not secured properly. However, such information could be valuable in detecting and mitigating attacks. Nguyen et al. [105] proposed LogSafe, which employs Intel Software Guard Extensions (SGX) to store logs from IoT devices efficiently and securely. As the authors noted, SGX provides security guarantees that allow LogSafe to run in an untrusted cloud infrastructure, and satisfies confidentiality, integrity, and availability (CIA) properties.

6.5.3 Identification of Hidden Cameras:

Recently, an interesting ideas was proposed by Cheng et al. [106], which attempted to detect hidden wireless cameras in an environment. The authors proposed DeWiCam, a lightweight detection mechanism using smartphones. Unlike existing detection methods for unauthorized wireless cameras, DeWiCam does not require any additional dedicated devices. The basic idea of DeWiCam is to utilize the intrinsic traffic patterns from wireless cameras, without the need to access encrypted information in the packets. Specifically, DeWiCam exploits the way wireless cameras transmit data. First, video and audio frames are compressed, and each frame is fragmented into a series of packets, so as to fit in the MTU size. Then, the fragmented video and audio packets are combined to be transmitted. The final traffic profile resembles a series of large packets interleaved by small packets, which the authors defined as the packet length distribution (PLD) pattern. Together with the duration, bandwidth stability, and PLD stability features, DeWiCam can detect wireless cameras.

6.5.4 *Management of Compromised Devices:*

Instead of focusing solely on device identification, Xu et al. [107] presented an idea more related to IoT device management. The authors argued that large IoT networks, consisting of many nearly identical devices, are especially attractive targets to attackers. However, recovery from compromises by conventional means is costly and slow, especially if the devices are distributed over a large geographical area, where network administrators or operators would have to travel to the devices to manually recover them. The authors presented CIDR, a system that can recover compromised IoT devices within a short amount of time, even if attackers have taken root control of every device in the network. Once a network administrator identifies a compromise and creates an updated firmware image, he/she can direct CIDR to force the devices to reset and to install the patched firmware. The authors call the ability of one computing device (i.e., the server on which CIDR is running) to control the software configuration of another computing device *dominance*. Unfortunately, CIDR requires hardware modifications, making it difficult to be applied in the real-world.

6.5.5 *Traffic Shaping to Prevent Unauthorized Identification:*

While identification is an important step in securing IoT devices, unauthorized analysis of IoT traffic for malicious purposes, as we described in subsection 5.1.1, should be prevented. Apthorpe et al. [53], who demonstrated that network observers, such as ISPs, could infer privacy sensitive in-home activities by simply analyzing the traffic rates generated from encrypted traffic of smart home devices, argued that current defenses to such traffic analysis attacks, such as blocking traffic, or tunneling traffic through a virtual private network (VPN), are ineffective. Specifically, developers would be disincentivized to allow users to block all or some traffic from leaving the smart home, and while tunneling smart home traffic through a VPN would increase the attack difficulty, it would not provide the user with any actual proof of privacy protection. In fact, the authors showed that an adversary can still fingerprint devices using the rate of VPN traffic alone. Therefore, the authors suggested that traffic shaping is the only solution that can prevent the leak of rate information, and thus render such an attack impossible. By shaping traffic rates to match a predetermined rate or schedule, users can prevent exposing information about device behavior to an adversary. Unfortunately, slowing down devices' traffic rates to match the predetermined rate can have significant negative performance impacts (e.g., voice assistants not answering questions, devices losing connectivity to their mobile applications, etc.), and adding extra cover traffic to match the predetermined rate can incur significant data usage costs.

7 DISCUSSION ON OPEN ISSUES & FUTURE WORK

While surveying the area of IoT security, we found several missing gaps and challenges yet to be sufficiently addressed, such as a lack of large-scale identification methods for insecure and compromised devices, lack of security protocol standardization, lack of sufficient regulatory frameworks, lack of user interest in or knowledge of security and privacy concerns in their homes, and lack of real-world deployment. However, from our perspective, there are four main challenges that are critical to IoT security, and need to be strongly considered in future work related to this area:

(1) **Lack of distributed solutions**: Most of the papers we surveyed proposed defense mechanisms that were placed in a central location, most likely the gateway/border router of the IoT network. Unfortunately, due to the device-to-device communication and mesh networking capabilities of many IoT protocols (e.g., 6LoWPAN, ZigBee, Bluetooth Mesh), centralized solutions cannot capture a large portion of intra-network traffic, and therefore, may be useless against in-network attacks.
(2) **Strong assumptions related to device and application modification**: Of the few distributed solutions we surveyed, most of them assume that devices or applications running on the devices can be modified. We argue that this is a strong assumption. Many of the devices already connected to the Internet are closed systems, thereby making modifications at any layer almost impossible. Furthermore, the location of IoT devices can also make modifications very difficult, and in some cases, infeasible (e.g., devices implanted in the body). Lastly, most everyday users probably do not possess the technical knowledge to make security modifications to their devices.
(3) **Lack of concern for energy consumption**: We were surprised to find that almost none of the papers published in the last five years that we surveyed treated energy consumption as a primary constraint when designing their security solutions. This may be because we mainly surveyed papers related to smart home security, and the research community is less concerned with energy consumption in such networks. We disagree. While it is true that many smart home devices may be "plugged-in" and do not rely on a finite energy source (e.g., a battery), there are some devices, such as cameras, smoke detectors, and various wearables, that are not always plugged-in. Also, an ideal security solution should be able to be deployed in various different IoT environments, not just smart homes. Furthermore, a solution that requires a seemingly moderate amount of energy may not be a significant issue for an individual, plugged-in device, but if that solution were to scale to the billions of IoT devices all over the world, its total energy consumption would be anything but moderate, which could arguably have a negative impact on our environment.
(4) **Lack of concern for IoT-enabled DDoS**: Considering the recent rise of large-scale IoT-enabled DDoS attacks, we were surprised to find a lack of interest among the IoT security research community in studying this issue in more depth. We only found a few solutions, such as [89], that focused on detecting recent malware strains specifically targeting IoT devices. We suspect the research community is not interested in this topic because many assume that the large body of work dealing with worm propagation and botnet detection for the traditional Internet can be applied directly to the IoT environment. However, we believe that the mesh networking capabilities, coupled with energy (and potentially memory) consumption constraints of IoT devices makes direct application somewhat unsuitable, especially for potentially new IoT malware strains that exploit those unique properties and constraints of IoT.

To address the first challenge, we propose a distributed defense ecosystem that is able to handle the unique capabilities of IoT networks. This ecosystem will include in-network security nodes that will be able to observe device-to-device communication. While each node will be responsible for observing parts of the network (e.g., a node in each room of a smart home), all nodes will be able to share information with each other to gain a complete view of the network. Nodes can also be mobile, moving to different locations on-demand, depending on the security needs of the network. While there are many challenges to overcome in order to make such an ecosystem a reality, such a distributed defense system would better suit an IoT environment, as compared to a centralized system.

To address the second challenge, we simply argue that future work should not be dependent on the assumption that devices and/or applications can be modified *at all*. In fact, there are a few examples of previous work that do not make this assumption, but can still provide security for IoT devices, such as work done by Gollakota et al. [108] and more recently, Malekzadeh et al. [109].

To address the third challenge, we recently proposed a security framework for smart homes with a primary focus on limiting resource consumption [110]. The proposed solution utilizes a two-mode adaptive security model that allows an IoT device to be in regular mode for most of the time which incurs a low resource consumption rate, and only when suspicious behavior is detected, switch to vigilant mode which potentially incurs a higher overhead. However, this framework also makes the assumption that devices can be modified, and therefore, can be further improved.

Lastly, to address the fourth challenge, we plan on improving previous worm detection solutions for the traditional Internet, such as work done by Li et al. [111], so that they can be effective in detecting IoT worms in IoT environments. Traditional solutions, which tend to be deployed at a central location, may not be able to detect potentially new forms of IoT worms. For example, new, unique IoT worms that exploit the mesh nature of IoT could spread (in the same fashion as the attack presented in [65]) unnoticed if detection is conducted solely at the network's border router. Also, due to resource constraints at IoT devices, distributing a traditional solution may not be a straightforward process. Furthermore, we have proposed a solution for mitigating IoT-enabled DDoS traffic at the source [112], which we can further improve.

8 CONCLUSION

The staggering growth of the Internet of Things (IoT) raises serious security concerns. We therefore conduct this survey in order to analyze the vulnerabilities plaguing IoT devices and networks, the attacks that can exploit those vulnerabilities, and the recent countermeasures proposed to address these security concerns. By conducting this survey, we found several challenges and open issues that have not been sufficiently explored in current literature. Some of these include a lack of distributed solutions, strong assumptions related to device and application modification, lack of concern for energy consumption, and lack of concern for IoT-enabled DDoS. By identifying these missing gaps, we are able to gain a clearer perspective of the future directions the research community should pursue.

REFERENCES

[1] Knud Lasse Lueth. 2018. State of the IoT 2018: Number of IoT Devices Now at 7B – Market Accelerating. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b. (2018).

[2] 2018. Ericsson Mobility Report. https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf. (2018).

[3] Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum, and Nasir Ghani. 2019. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Communications Surveys and Tutorials*.

[4] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. *IEEE Symposium on Security and Privacy (S&P)*.

[5] Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. 2015. Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. *IEEE Communications Surveys and Tutorials*.

[6] Aafaf Ouaddah, Hajar Mousannif, Anas Abou Elkalam, and Abdellah Ait Ouahman. 2017. Access Control in the Internet of Things: Big Challenges and New Opportunities. *Computer Networks.*

[7] Bruno Bogaz Zarpelão, Rodrigo Sanches Miani, Cláudio Toshio Kawakani, and Sean Carlisto de Alvarenga. 2017. A Survey of Intrusion Detection in Internet of Things. *Journal of Network and Computer Applications.*

[8] A. Mosenia and K. Niraj. 2017. A Comprehensive Study of Internet of Things. *IEEE Transactions on Emerging Topics in Computing.*

[9] Audrey A. Gendreau and Michael Moorman. 2016. Survey of Intrusion Detection Systems Towards an End to End Secure Internet of Things. *International Conference on Future Internet of Things and Cloud (FiCloud).*

[10] Nadia Chaabouni, Mohamed Mosbah, Akka Zemmari, Cyrille Sauvignac, and Parvez Faruki. 2019. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Communications Surveys and Tutorials.*

[11] Z. Berkay Celik, Earlence Fernandes, Eric Pauley, Gang Tan, and Patrick Mcdaniel. 2019. Program Analysis of Commodity IoT Applications for Security and Privacy: Challenges and Opportunities. *ACM Computing Surveys.*

[12] Hamed Hellaoui, Mouloud Koudil, and Abdelmadjid Bouabdallah. 2017. Energy-Efficient Mechanisms in Security of the Internet of Things: A Survey. *Computer Networks.*

[13] Rolf H. Weber and Evelyne Studer. 2016. Cybersecurity in the Internet of Things: Legal Aspects. *Computer Law and Security Review.*

[14] Minhaj Ahmad Khan and Khaled Salah. 2018. IoT Security: Review, Blockchain Solutions, and Open Challenges. *Future Generation Computer Systems.*

[15] Ivan Farris, Tarik Taleb, Yacine Khettab, and Jaeseung Song. 2019. A Survey on Emerging SDN and NFV Security Mechanisms for IoT Systems. *IEEE Communications Surveys and Tutorials.*

[16] Francesca Meneghello, Matteo Calore, Daniel Zucchetto, Michele Polese, and Andrea Zanella. 2019. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. *IEEE Internet of Things Journal.*

[17] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials.*

[18] Noah Apthorpe, Sarah Varghese, and Nick Feamster. 2019. Evaluating the Contextual Integrity of Privacy Regulation: Parents' IoT Toy Privacy Norms Versus COPPA. *USENIX Security Symposium.*

[19] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini. 2015. Security, Privacy and Trust in Internet of Things: The Road Ahead. *Computer Networks.*

[20] Rodrigo Roman, Jianying Zhou, and Javier Lopez. 2013. On the Features and Challenges of Security and Privacy in Distributed internet of Things. *Computer Networks.*

[21] Anne H. Ngu, Mario Gutierrez, Vangelis Metsis, Surya Nepal, and Quan Z. Sheng. 2017. IoT Middleware: A Survey on Issues and Enabling Technologies. *IEEE Internet of Things Journal.*

[22] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar. 2019. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access.*

[23] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2017. Understanding the Internet of Things: Definition, Potentials, and Societal Role of a Fast Evolving Paradigm. *Ad Hoc Networks.*

[24] Jacob Wurm, Khoa Hoang, Orlando Arias, Ahmad Reza Sadeghi, and Yier Jin. 2016. Security Analysis on Consumer and Industrial IoT Devices. *Asia and South Pacific Design Automation Conference (ASP-DAC).*

[25] Arvind Singh, Nikhil Chawla, Jong Hwan Ko, Monodeep Kar, and Saibal Mukhopadhyay. 2019. Energy Efficient and Side-Channel Secure Cryptographic Hardware for IoT-Edge Nodes. *IEEE Internet of Things Journal.*

[26] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. 2019. Light Commands: Laser-Based Audio Injection Attacks on Voice-Controllable Systems.

[27] N. Kushalnagar, G. Montenegro, and C. Schumacher. 2007. RFC 4919: IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. *Internet Engineering Task Force.*

[28] ZigBee Alliance. 2010. Zigbee alliance. *WPAN Industry Group, http://www. zigbee. org/. The industry group responsible for the ZigBee standard and certification.*

[29] Carles Gomez, Joaquim Oller, and Josep Paradells. 2012. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors.*

[30] Zach Shelby, Klaus Hartke, Carsten Bormann, and B. Frank. 2014. RFC 7252: The constrained application protocol (CoAP). *Internet Engineering Task Force.*

[31] T. Winter, P. Thubert, A. Brandt, J. Hui, and R. Kelsey. 2014. RFC 6550: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. *Internet Engineering Task Force.*

[32] C. Perkins, E. Belding-Royer, and S. Daas. 2003. RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing. *Internet Engineering Task Force.*

[33] Linus Wallgren, Shahid Raza, and Thiemo Voigt. 2013. Routing Attacks and Countermeasures in the RPL-Based Internet of Things. *International Journal of Distributed Sensor Networks.*

[34] Niko Vidgren, Keijo Haataja, José Luis Patiño-Andres, Juan José Ramírez-Sanchis, and Pekka Toivanen. 2013. Security Threats in ZigBee-Enabled Systems: Vulnerability Evaluation, Practical Experiments, Countermeasures, and Lessons Learned. *Hawaii International Conference on System Sciences (HICSS).*

[35] Tobias Zillner and S. Strobl. 2015. ZigBee Exploited: The Good, The Bad and The Ugly. *Black Hat USA.*

[36] Philipp Morgner, Stephan Mattejat, and Zinaida Benenson. 2016. All Your Bulbs are Belong to Us: Investigating the Current State of Security in Connected Lighting Systems. *arXiv preprint arXiv:1608.03732.*

[37] Mike Ryan. 2013. Bluetooth Smart: The Good, The Bad, The Ugly ... and The Fix. *Black Hat USA.*

[38] Grant Ho, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. 2016. Smart Locks: Lessons for Securing Commodity Internet of Things Devices. *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[39] Jakub Czyz, Matthew Luckie, Mark Allman, and Michael Bailey. 2017. Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy. *Network and Distributed System Security Symposium (NDSS).*

[40] Andrei Costin, Jonas Zaddach, and Aurélien Francillon. 2014. A Large Scale Analysis of the Security of Embedded Firmwares. *USENIX Security Symposium.*

[41] Anna Kornfeld Simpson, Franziska Roesner, and Tadayoshi Kohno. 2017. Securing Vulnerable Home IoT Devices with an In-Hub Security Manager. *International Workshop on Pervasive Smart Living Spaces (PerLS).*

[42] Ang Cui and Salvatore J. Stolfo. 2010. A Quantitative Analysis of the Insecurity of Embedded Network Devices: Results of a Wide-Area Scan. *Annual Computer Security Applications Conference (ACSAC).*

[43] Yin Minn, Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2015. IoTPOT: Analysing the Rise of IoT Compromises. *USENIX Workshop on Offensive Technologies (WOOT).*

[44] Manos Antonakakis, Tim April, Michael Bailey, Matthew Bernhard, Ann Arbor, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Ann Arbor, Luca Invernizzi, Michalis Kallitsis, Merit Network, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, Yi Zhou, Manos Antonakakis, Tim April, Michael Bailey, Matthew Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the Mirai Botnet. *USENIX Security Symposium.*

[45] Artur Marzano, David Alexander, Osvaldo Fonseca, Elverton Fazzion, Cristine Hoepers, Klaus Steding-Jessen, Marcelo H.P.C. Chaves, Italo Cunha, Dorgival Guedes, and Wagner Meira. 2018. The Evolution of Bashlite and Mirai IoT Botnets. *IEEE Symposium on Computers and Communications (ISCC).*

[46] Stephen Herwig, Katura Harvey, George Hughey, Richard Roberts, and Dave Levin. 2019. Measurement and Analysis of Hajime, A Peer-to-Peer IoT Botnet. *Network and Distributed System Security Symposium (NDSS).*

[47] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Amit K Sikder, Gang Tan, Patrick Mcdaniel, and A Selcuk Uluagac. 2018. Sensitive Information Tracking in Commodity IoT. *USENIX Security Symposium.*

[48] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A. Gunter. 2019. Charting the Attack Surface of Trigger-Action IoT Platforms. *ACM Conference on Computer and Communications Security (CCS).*

[49] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. 2018. Skill Squatting Attacks on Amazon Alexa. *USENIX Security Symposium.*

[50] Nan Zhang, Xianghang Mi, Xuan Feng, Xiaofeng Wang, Yuan Tian, and Feng Qian. 2019. Dangerous Skills: Understanding and Mitigating Security Risks of Voice-Controlled Third-Party Functions on Virtual Personal Assistant Systems. *IEEE Symposium on Security and Privacy (S&P).*

[51] He Wang, Ted Tsung Te Lai, and Romit Roy Choudhury. 2015. MoLe: Motion Leaks Through Smartwatch Sensors. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[52] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. 2016. Friend or Foe? Your Wearable Devices Reveal your Personal PIN. *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[53] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic. *arXiv preprint arXiv:1708.05044.*

[54] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. 2016. Is Anybody Home? Inferring Activity from Smart Home Network Traffic. *IEEE Symposium on Security and Privacy Workshops (SPW).*

[55] Zhice Yang, Qianyi Huang, and Qian Zhang. 2017. NICScater: Backscater as a Covert Channel in Mobile Devices. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[56] Eyal Ronen and Adi Shamir. 2016. Extended Functionality Attacks on IoT Devices: The Case of Smart Lights. *IEEE European Symposium on Security and Privacy (Euro S&P).*

[57] Vinay Sachidananda, Jinghui Toh, Shachar Siboni, Suhas Bhairav, Asaf Shabtai, and Yuval Elovici. 2017. Let the Cat out of the Bag: A Holistic Approach Towards Security Analysis of the Internet of Things. *ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS).*

[58] Orcun Cetin, Carlos Ganan, Lisette Altena, Takahiro Kasama, Daisuke Inoue, Kazuki Tamiya, Ying Tie, Katsunari Yoshioka, and Michel van Eeten. 2019. Cleaning Up the Internet of Evil Things: Real-World Evidence on ISP and Consumer Efforts to Remove Mirai. *Network and Distributed System Security Symposium (NDSS).*

[59] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. DolphinAttack: Inaudible voice commands. *ACM Conference on Computer and Communications Security (CCS).*

[60] Richard Mitev, Markus Miettinen, and Ahmad Reza Sadeghi. 2019. Alexa Lied to Me: Skill-Based Man-in-the-Middle Attacks on Virtual Assistants. *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[61] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security Analysis of Emerging Smart Home Applications. *IEEE Symposium on Security and Privacy (S&P).*

[62] Eugene Y. Vasserman and Nicholas Hopper. 2013. Vampire Attacks: Draining Life from Wireless Ad Hoc Sensor Networks. *IEEE Transactions on Mobile Computing.*

[63] Federico Chiariotti, Chiara Pielli, Nicola Laurenti, and Andrea Zanella. 2019. A Game-Theoretic Analysis of Energy-Depleting Jamming Attacks with a Learning Counterstrategy. *ACM Transactions on Sensor Networks.*

[64] Saleh Soltan, Prateek Mittal, H Vincent, and H Vincent Poor. 2018. BlackIoT: IoT Botnet of High Wattage Devices Can Disrupt the Power Grid. *USENIX Security Symposium.*

[65] Eyal Ronen, Adi Shamir, Achi Or Weingarten, and Colin Oflynn. 2018. IoT Goes Nuclear: Creating a Zigbee Chain Reaction. *IEEE Symposium on Security and Privacy (S&P).*

[66] Catalin Cimpanu. 2017. BrickerBot Author Retires Claiming to Have Bricked over 10 Million IoT Devices. https://www.bleepingcomputer.com/news/security/brickerbot-author-retires-claiming-to-have-bricked-over-10-million-iot-devices/.

[67] Janita. 2016. DDoS Attack Halts Heating in Finland Amidst Winter. https://metropolitan.fi/entry/ddos-attack-halts-heating-in-finland-amidst-winter.

[68] Weijia He, Roshni Padhi, Jordan Ofek, Maximilian Golla, Markus Dürmuth, Earlence Fernandes, and Blase Ur. 2018. Rethinking Access Control and Authentication for the Home Internet of Things (IoT). *USENIX Security Symposium.*

[69] Eric Zeng and Franziska Roesner. 2019. Understanding and Improving Security and Privacy in Multi-User Smart Homes: A Design Exploration and In-Home User Study. *USENIX Security Symposium.*

[70] Yuan Tian, Nan Zhang, Yueh-hsun Lin, Xiaofeng Wang, Blase Ur, Yuan Tian, Nan Zhang, Yueh-hsun Lin, Xiaofeng Wang, Blase Ur, and Xianzheng Guo. 2017. SmartAuth: User-Centered Authorization for the Internet of Things. *USENIX Security Symposium.*

[71] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Z Morley Mao, and Atul Prakash. 2017. ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms. *Network and Distributed System Security Symposium (NDSS).*

[72] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2018. Situational Access Control in the Internet of Things. *ACM Conference on Computer and Communications Security (CCS).*

[73] Amir Rahmati, Earlence Fernandes, Kevin Eykholt, and Atul Prakash. 2018. Tyche: A Risk-Based Permission Model for Smart Homes. *IEEE Cybersecurity Development Conference (SecDev).*

[74] James Hong, Amit Levy, Laurynas Riliskis, and Philip Levis. 2018. Don't Talk Unless I Say So! Securing the Internet of Things with Default-Off Networking. *ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI).*

[75] Tam Le and Matt W. Mutka. 2019. Access Control with Delegation for Smart Home Applications. *ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI).*

[76] Earlence Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. 2016. FlowFence: Practical Data Protection for Emerging IoT Application Frameworks. *USENIX Security Symposium.*

[77] Earlence Fernandes, Amir Rahmati, Jaeyeon Jung, and Atul Prakash. 2018. Decentralized Action Integrity for Trigger-Action IoT Platforms. *Network and Distributed System Security Symposium (NDSS).*

[78] Iulia Bastys, Musard Balliu, and Andrei Sabelfeld. 2018. If This Then What? Controlling Flows in IoT Apps. *ACM Conference on Computer and Communications Security (CCS).*

[79] Z. Berkay Celik, Gang Tan, and Patrick McDaniel. 2019. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. *Network and Distributed System Security Symposium (NDSS).*

[80] Marcela S Melara, David H Liu, and Michael J Freedman. 2019. Pyronia: Redesigning Least Privilege and Isolation for the Age of IoT. *arXiv preprint arXiv:1903.01950.*

[81] Ben Hutchins, Michael Zhou, Anudeep Reddy, Ming Li, Wenqiang Jin, and Lei Yang. 2018. Beat-PIN: A User Authentication Mechanism for Wearable Devices Through Secret Beats. *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[82] Viet Nguyen, Mohamed Ibrahim, Hoang Truong, Phuc Nguyen, Marco Gruteser, Richard Howard, and Tam Vu. 2018. Body-Guided Communications: A Low-Power, Highly-Confined Primitive to Track and Secure Every Touch. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[83] Xiaopeng Li, Fengyao Yan, Fei Zuo, Qiang Zeng, and Lannan Luo. 2019. Touch Well Before Use: Intuitive and Secure Authentication for IoT Devices. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[84] Zhenyu Yan, Qun Song, Rui Tan, Yang Li, and Adams Wai Kin Kong. 2019. Towards Touch-to-Access Device Authentication Using Induced Body Electric Potentials. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[85] Wenqiang Chen, Lin Chen, Yandao Huang, Xinyu Zhang, Lu Wang, Rukhsana Ruby, and Kaishun Wu. 2019. Taprint: Secure Text Input for Commodity Smart Wristbands. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[86] Feng Lin, Chen Song, Yan Zhuang, Wenyao Xu, Changzhi Li, and Kui Ren. 2017. Cardiac Scan: A Non-Contact and Continuous Heart-Based User Authentication System. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[87] Huan Feng, Kassem Fawaz, and Kang G. Shin. 2017. Continuous Authentication for Voice Assistants. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[88] Anadi Chaman, Jiaming Wang, Jiachen Sun, Haitham Hassanieh, and Romit Roy Choudhury. 2018. Ghostbuster: Detecting the Presence of Hidden Eavesdroppers. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[89] Dominik Breitenbacher, Ivan Homoliak, Yan Lin Aung, Nils Ole Tippenhauer, and Yuval Elovici. 2019. HADES-IoT: A Practical Host-Based Anomaly Detection System for IoT Devices. *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[90] Shahid Raza, Linus Wallgren, and Thiemo Voigt. 2013. SVELTE: Real-Time Intrusion Detection in The Internet of Things. *Ad Hoc networks.*

[91] Amit Kumar Sikder, Hidayet Aksu, and A. Selcuk Uluagac. 2017. 6thSense: A Context-aware Sensor-based Attack Detector for Smart Devices. *USENIX Security Symposium.*

[92] Simon Birnbach, Simon Eberz, and Ivan Martinovic. 2019. Peeves: Physical Event Verification in Smart Homes. *ACM Conference on Computer and Communications Security (CCS).*

[93] Yitao He, Junyu Bian, Xinyu Tong, Zihui Qian, Wei Zhu, Xiaohua Tian, and Xinbing Wang. 2019. Canceling Inaudible Voice Commands Against Voice Control Systems. *ACM International Conference on Mobile Computing and Networking (MobiCom).*

[94] Wei Zhang, Yan Meng, Yugeng Liu, Xiaokuan Zhang, Yinqian Zhang, and Haojin Zhu. 2018. Homonit: Monitoring Smart Home Apps from Encrypted Traffic. *ACM Conference on Computer and Communications Security (CCS).*

[95] Wenbo Ding and Hongxin Hu. 2018. On the Safety of IoT Device Physical Interaction Control. *ACM Conference on Computer and Communications Security (CCS).*

[96] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. 2018. Fear and Logging in the Internet of Things. *Network and Distributed System Security Symposium (NDSS).*

[97] Shahid Raza, Hossein Shafagh, Kasun Hewage, Rene Hummen, and Thiemo Voigt. 2013. Lithe: Lightweight Secure CoAP for the Internet of Things. *IEEE Sensors Journal.*

[98] Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, and Utz Roedig. 2011. Securing Communication in 6LoWPAN with Compressed IPsec. *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS).*

[99] Hossein Shafagh, Anwar Hithnawi, Andreas Dröscher, Simon Duquennoy, and Wen Hu. 2015. Talos: Encrypted Query Processing for the Internet of Things. *ACM Conference on Embedded Networked Sensor Systems (SenSys).*

[100] Hossein Shafagh, Anwar Hithnawi, Lukas Burkhalter, Pascal Fischli, Hossein Shafagh, Anwar Hithnawi, Lukas Burkhalter, Pascal Fischli, Simon Duquennoy Secure Shar, Hossein Shafagh, Lukas Burkhalter, Pascal Fischli, and Simon Duquennoy. 2017. Secure Sharing of Partially Homomorphic Encrypted IoT Data. *ACM Conference on Embedded Networked Sensor Systems (SenSys).*

[101] Chaohao Li, Xiaoyu Ji, Xinyan Zhou, Juchuan Zhang, Jing Tian, Yanmiao Zhang, and Wenyuan Xu. 2018. HlcAuth: Key-free and Secure Communications via Home-Limited Channel. *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[102] Jiongyi Chen, Wenrui Diao, Qingchuan Zhao, Chaoshun Zuo, Zhiqiang Lin, XiaoFeng Wang, Wing Cheong Lau, Menghan Sun, Ronghai Yang, and Kehuan Zhang. 2018. IoTFuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing. *Network and Distributed System Security Symposium (NDSS).*

[103] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2018. Classifying IoT Devices in Smart Environments using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing.*

[104] Xuan Feng, Qiang Li, Haining Wang, and Limin Sun. 2018. Acquisitional Rule-based Engine for Discovering Internet-of-Things Devices. *USENIX Security Symposium.*

[105] Hung Nguyen, Radoslav Ivanov, Linh T.X. Phan, Oleg Sokolsky, James Weimer, and Insup Lee. 2018. LogSafe: Secure and Scalable Data Logger for IoT Devices. *ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI).*

[106] Yushi Cheng, Xiaoyu Ji, Tianyang Lu, and Wenyuan Xu. 2018. DeWiCam: Detecting Hidden Wireless Cameras via Smartphones. *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[107] Meng Xu, Manuel Huber, Zhichuang Sun, Paul England, Marcus Peinado, Sangho Lee, Andrey Marochko, Dennis Mattoon, Rob Spiger, and Stefan Thom. 2019. Dominance as a

New Trusted Computing Primitive for the Internet of Things. *IEEE Symposium on Security and Privacy (S&P).*

[108] Shyamnath Gollakota, Haitham Hassanieh, Benjamin Ransford, Dina Katabi, and Kevin Fu. 2011. They Can Hear Your Heartbeats: Non-Invasive Security for Implantable Medical Devices. *ACM Special Interest Group on Data Communications (SIGCOMM).*

[109] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. 2019. Mobile Sensor Data Anonymization. *ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI).*

[110] Devkishen Sisodia, Samuel Mergendahl, Jun Li, and Hasan Cam. 2018. Securing the Smart Home via a Two-Mode Security Framework. *International Conference on Security and Privacy in Communication Systems (SecureComm).*

[111] Jun Li and Shad Stafford. 2014. Detecting Smart, Self-Propagating Internet Worms. *IEEE Conference on Communications and Network Security (CNS).*

[112] Samuel Mergendahl, Devkishen Sisodia, Jun Li, and Hasan Cam. 2018. FR-WARD: Fast Retransmit as a Wary but Ample Response to Distributed Denial-of-Service Attacks from the Internet of Things. *International Conference on Computer Communication and Networks (ICCCN).*