11-8-2021

Novel Attacks and Defenses for Enterprise Internet-of-Things (E-IoT) Systems

Luis C. Puche Rondon
Florida International University, lpuch002@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

NOVEL ATTACKS AND DEFENSES FOR ENTERPRISE INTERNET-OF-THINGS (E-IOT) SYSTEMS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Luis C. Puche Rondon

2021

To: Dean John Volakis
College of Engineering and Computing

This dissertation, written by Luis C. Puche Rondon, and entitled Novel Attacks and Defenses for Enterprise Internet-of-Things (E-IoT) Systems, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Kemal Akkaya

Alexander Perez-Pons

Bogdan Carbunar

A. Selcuk Uluagac, Major Professor

Date of Defense: November 08, 2021

The dissertation of Luis C. Puche Rondon is approved.

Dean John Volakis
College of Engineering and Computing

Andres G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2021

DEDICATION

To my parents, Luis Puche del Portillo and Adriana Puche

and my brother, Jose Puche.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. A. Selcuk Uluagac for his valuable guidance and tireless support throughout the entire doctoral process. His mentoring, insights, encouragement, and experience have motivated this research throughout the doctoral journey. My deepest gratitude to the committee members; Dr. Kemal Akkaya for his collaboration and feedback on many publications; Dr. Bogdan Cabunar for his role as an educator in essential security mechanisms; and Dr. Alexander Perez-Pons being my introduction to the cybersecurity field.

A sincere thank you to all my colleagues at the Cyber-Physical Systems Security Lab for making this journey a memorable experience throughout all these years. My gratitude also goes to our post-doctoral researcher Dr. Ahmet Aris, whose feedback, collaboration, and support has been invaluable for many of our publications. I would also like to thank Dr. Leonardo Babun for his constant feedback and guidance on several papers. Also, a thank you to Dr. Abbas Acar and Dr. Amit Kumar Sikder for their feedback and suggestions throughout this dissertation. I would also like to thank Florida International University for being my university since 2011. I would like to acknowledge the NSF Scholarship-for-service and the Koerner Family Foundation for their invaluable financial support throughout my doctoral efforts.

Finally, I would like to thank my parents Luis Puche and Adriana Puche for their unwavering support and encouragement throughout my entire education, my brother Jose Puche for his motivation, and the rest of my family for their heartfelt encouragement.

This dissertation is partly based upon work supported by the National Science Foundation (Award Numbers: NSF-CAREER-CNS-1453647, NSF-1663051).

ABSTRACT OF THE DISSERTATION

NOVEL ATTACKS AND DEFENSES FOR ENTERPRISE INTERNET-OF-THINGS (E-IOT) SYSTEMS

by

Luis C. Puche Rondon

Florida International University, 2021

Miami, Florida

Professor A. Selcuk Uluagac, Major Professor

This doctoral dissertation expands upon the field of Enterprise Internet-of-Things (E-IoT) systems, one of the most ubiquitous and under-researched fields of smart systems. E-IoT systems are specialty smart systems designed for sophisticated and high-end automation applications (e.g., multimedia control, security, lighting control). E-IoT systems are often closed source, costly, require certified installers, and are more robust for their specific applications. This dissertation begins with an analysis of the current E-IoT threat landscape and introduces three novel attacks and defenses for under-studied software and protocols heavily linked to E-IoT systems. For each, we review the literature for the threats, attacks, and countermeasures. Based on the systematic knowledge we obtain from the literature review, we propose three novel attacks and countermeasures to protect E-IoT systems. In the first attack, we present PoisonIvy, several attacks developed to show that malicious E-IoT drivers can be used to compromise E-IoT. In response to PoisonIvy threats, we describe Ivycide, a machine-learning network-based solution designed to defend E-IoT systems against E-IoT driver threats. As multimedia control is a significant application of E-IoT, we introduce is HDMI-Walk, a novel attack vector designed to demonstrate that HDMI's Consumer Electronics Control (CEC) protocol can be used to compromise multiple devices through a single connection. To defend devices from this threat, we introduce HDMI-Watch, a standalone intrusion detection system (IDS) designed to defend HDMI-enabled devices from HDMI-Walk-style attacks. Finally, this dissertation evaluates the security of E-IoT proprietary protocols with LightingStrike, a series of attacks used to demonstrate that popular E-IoT proprietary communication protocols are insecure. To address LightningStrike threats, we introduce LGuard, a complete defense framework designed to defend E-IoT systems from LightingStrike-style attacks using computer vision, traffic obfuscation, and traffic analysis techniques. For each contribution, all of the defense mechanisms proposed are implemented without any modification to the underlying hardware or software. All attacks and defenses in this dissertation were performed with implementations on widely-used E-IoT devices and systems. We believe that the research presented in this dissertation has notable implications on the security of E-IoT systems by exposing novel threat vectors, raising awareness, and motivating future E-IoT system security research.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

The introduction of smart consumer electronics has led to the widespread adoption of smart devices, with over 45 million of smart home components distributed and deployed worldwide [ABC$^+$18, IoT18]. Many researchers and users are familiar with commodity, off-the-shelf smart systems that can be purchased and installed by the average end-user without specialized training. However, for more complex applications, where reliability in smart solutions is needed, Enterprise Internet-of-Things (E-IoT) have become an accepted solution. E-IoT systems offer customized deployments, with numerous use-cases and applications, offering users a broad set of compatible devices, custom-programmed behavior, user interface customization, and proprietary protocols. Further, E-IoT systems have been increasingly popular in smart installations, with Crestron growing to 1.5 billion dollars of annual revenue in 2018 and Control4 deploying over 15 million smart products in over 400,000 installations worldwide [Mar18, Con20b]. With many of these systems present in high-profile locations such as luxury smart homes, yachts, smart buildings, smart offices, and secure conference rooms the security of E-IoT is of utmost importance.

Nonetheless, while recent works have examined well-known components (e.g., software, apps) from widely-available smart systems [LBAU17, KBAU18, SBAU19, CBS$^+$18a, AADB17, AFA$^+$20, AAUA18, SAU20, SPA$^+$18, USB, CBS$^+$18b, NSRU19, NSRU20, NSBU20, CMT$^+$, SBC$^+$20, BAR$^+$20, MBY$^+$19, DBU20], very little research has been conducted on E-IoT systems and their associated components. The lack of research into E-IoT systems has led many users and researchers to overlook E-IoT system vulnerabilities and mistakenly assume that these systems are secure. There are several challenges related to E-IoT research. First, the closed-source nature and availability of E-IoT software, technical documentation, and components makes research into E-IoT threats challenging. Further, without access to API/system call hooking, source code, or special permissions, many traditional defense mechanisms are not viable for E-IoT. As such, proposed solutions must not intrude with E-IoT operation and consider the limitations of working with E-IoT systems without vendor support. To address this research gap, we first identify three new attacks to common E-IoT components (e.g., protocols, software) and demonstrate how these novel threats can be used by malicious actors to compromise devices. Finally, we account for the limitations of closed-source E-IoT architecture and propose novel defense solutions to address these new-found threats.

The Landscape of E-IoT Threats and Defenses

The evolution of smart technology has yielded to incremental improvements and advancements in the field of E-IoT since their inception. However, as systems grow more complex, this added complexity creates more threat vectors for the target system. Thus, numerous attacks, threats, and vulnerabilities have been discovered, creating several novel vectors of attack against E-IoT systems. For instance, attackers can attempt to compromise E-IoT through supply chain attacks, software vulnerabilities, or side-channel attacks. To better defend E-IoT, the threats and defenses that can impact E-IoT security must be identified. However, while prior research may be applicable, E-IoT systems are unique in their design, architecture, components, and challenges. As such, known threats and defenses as applied to E-IoT can create different implications and should be treated as a separate field of study. For instance, if a large E-IoT system relies on Zigbee communication for some components, other integrated components may be affected. As a result, an analysis of current threats and defenses in the context of E-IoT is needed.

E-IoT Drivers

The need for E-IoT systems to become compatible with third-party devices has led to the need for software modules known as E-IoT Drivers. E-IoT drivers are used to give E-IoT systems a higher level of customizability and they easily integrate software services and hardware to an E-IoT system. As such, drivers are programmable components in E-IoT that include all the relevant information (e.g., model, commands, inputs, protocol, outputs) needed to integrate a device into an E-IoT system. However, while most drivers originate from trusted sources such as the E-IoT vendor or device manufacturer, there are many forums and external third-party sites where unverified drivers can be found [C4F]. Unverified drivers can be developed by unknown third parties and then later installed in E-IoT systems for several reasons. For instance, unverified drivers may be installed if no verified drivers are available or verified drivers show to be costly. Prior research on IoT apps has shown that any programmable module for a larger system can act as a threat vector for an attacker. However, earlier works have never researched the possibility of malicious E-IoT drivers as a threat vector, or proposed any defenses.

E-IoT Multimedia Control

A common use-case of E-IoT is complete multimedia control in spaces such as offices, conference rooms, and home theaters. In such cases, E-IoT systems will often integrate multimedia devices (e.g., amplifiers, switchers, receivers, theater systems, speakers). With multimedia control, users and guests may control conference room systems, audio, and video of multiple rooms at the press of a button without the need to understand the underlying operation of the multimedia system. This abstraction is necessary, as E-IoT systems may manage increasingly complex multimedia systems, some extending to control over one-hundred displays (e.g., televisions, projectors), video sources, and audio zones. Thus, a user or guest cannot be expected to understand the underlying multimedia system. For many of these systems, the High Definition Multimedia Interface (HDMI) is the backbone and the de-facto standard for A/V connections between video-enabled devices. An important component in HDMI is the Consumer Electronics Control (CEC) protocol, which allows HDMI devices that share an HDMI connection to communicate and interact with each other [Hol05]. However, as widespread as E-IoT systems and distributed HDMI networks are, current research has not examined possible threats with CEC, or proposed any defense mechanisms. An attacker that uses CEC to compromise devices may remain completely undetected from traditional network intrusion mechanisms and cause undesired operation to HDMI-enabled devices.

Proprietary Communication Buses

The need of custom E-IoT solutions has led to the development of custom devices (e.g., controllers, interfaces) by E-IoT vendors. Similarly, when protocols are not available, E-IoT vendors will often develop their own protocols to fit their purpose. For instance, E-IoT vendors such as Lutron created RadioRA as a proprietary wireless solution, while Cresnet was developed by Crestron for wired communication between Crestron devices [Bla20, Cre17a]. A common use case of proprietary technology are communication buses, used to establish communication between multiple E-IoT devices (e.g., touchscreens, keypads). These buses rely on proprietary protocols such as Cresnet to relay information from connected devices to the E-IoT controller. For instance, when a keypad button is pressed, the keypad will transmit a packet through the communication bus notifying the system controller that a button has been pressed. The E-IoT controller with then actuate the programming associated with that button press (e.g, turn on a light). Thus, E-IoT systems rely heavily on proprietary protocols for communication and functionality. However, no current research has investigated how secure these closed-source protocols are. Further, if any vulnerabilities are found, no defense mechanisms exist for proprietary E-IoT serial-based protocols.

1.1 Research Purposes

This doctoral dissertation introduces novel threats and defense mechanisms for E-IoT systems and relevant E-IoT sub-components. With this dissertation, we cover six fundamental contributions towards E-IoT security: (1) the introduction of E-IoT drivers as a novel threat vector; (2) a smart, machine learning-based solution to new-found E-IoT driver threats; (3) the demonstration of HDMI's CEC protocol as a viable threat vector; (4) a novel HDMI-based defense mechanism to defend devices against CEC-based attacks; (5) the demonstration of E-IoT proprietary serial-based protocol vulnerabilities and threat vectors; (6) a defense framework to protect E-IoT against newly discovered proprietary protocol threats. For each contribution, this dissertation introduces novel proof-of-concept attacks and defense systems that can be realistically applied to live E-IoT system and raise the overall awareness of the current state of E-IoT security.

1.2 Research Problems

The research problems for this dissertation have four components:

- *Identification of E-IoT Threats:* E-IoT is a novel field of research, with many components that make E-IoT a unique threat vector. Thus, it is fundamental to first find understand the architecture for E-IoT. Further, known threats and defenses must evaluated in the context of E-IoT architecture.
- *E-IoT Driver Threats and Defenses:* Drivers are frequently utilized in E-IoT systems to easily integrate third-party devices and services. However, drivers

are used in E-IoT systems without any consideration for malware threats. As drivers are programmable and modular, an attacker can easily create a malicious driver to compromise an E-IoT system. Thus, PoisonIvy is presented to show that an external attacker can use E-IoT drivers to compromise an E-IoT system. To address the threat of malicious drivers, a network-based IDS is introduced called Ivycide.

- *E-IoT Multimedia Control Threats and Defenses:* As E-IoT integrates a wide variety of devices and employs countless protocols, some overlooked threat vectors can severely impact E-IoT system performance. In effect, E-IoT's emphasis on multimedia control makes E-IoT systems susceptible to threats from multimedia-based protocols such as HDMI. We show that CEC, an integral part of HDMI, can be used to assume arbitrary control of HDMI-enabled devices using HDMI-Walk. To solve this research problem, HDMI-Watch, a defense mechanism is proposed to protect HDMI-enabled devices and distribution networks.

- *E-IoT Communication Bus Threats and Defenses:* In E-IoT environments, communication buses are used to connect devices such as interfaces in an effective and reliable manner. Moreover, E-IoT vendors will create proprietary communication protocols for these communication buses and their devices. However, many of these proprietary protocols rely solely on security-through-obscurity, as the design and implementation are unknown. To verify if major protocols are secure, LightningStrike is introduced, showing that an E-IoT system can be compromised due to weak proprietary protocols. To defend against novel threats, a custom defense called LGuard is introduced.

In general, to understand E-IoT as a threat vector and properly mitigate future threats, components need to be evaluated individually. As such, finding attacks on E-IoT components is the first step to securing E-IoT devices against threats and attackers. Once threats are identified, new defense mechanisms need to be proposed to mitigate attacks, considering the limitations of working with E-IoT (e.g., lack of vendor support, closed-source design).

1.3 Significance of the Study

Currently, E-IoT systems see widespread deployments, with common usage in protected locations (e.g., smart buildings, luxury smart homes, expensive yachts, classrooms, meeting rooms, government offices, and business establishments). These systems and their associated protocols may integrate multimedia, lighting control, motorization, security, sensors, and other sensitive systems. However, the closed-source nature of E-IoT makes researching E-IoT and its associated components a challenge. This lack of research has led many users to mistakenly believe that E-IoT systems and their components are secure. Thus, investigating threats that can affect millions of E-IoT systems and finding defense mechanisms for these threats, is of utmost importance. In this dissertation, we aim to resolve this research gap by first investigating existing, unexplored threat vectors closely tied to E-IoT. We then acknowledge the limitations of E-IoT and find defense mechanisms that do not alter the original closed-source systems as a defense strategy. Specifically, we focus on three main areas: (1) E-IoT-specific software (e.g., drivers) threats and defenses; (2) Multimedia control and complex HDMI distributions threats and defenses; and (3) E-IoT proprietary communication (e.g., serial buses) threats and defenses.

1.4 Organization

The rest of this dissertation is organized as follows. In Chapter 2, we present background information essential to the understanding of this dissertation. In Chapter 3, we discuss the related work. Then, in Chapter 4 we overview E-IoT systems, threats, and defenses at each layer. In Chapter 5, we evaluate the security of drivers with PoisonIvy and introduce Ivycide as a defense mechanism for these novel threats. Later, in Chapter 6, we demonstrate HDMI-Walk attacks and the HDMI-Watch defense mechanism. Further, Chapter 7 investigates E-IoT communication bus protocol security with LightningStrike and introduces a defense mechanism, LGuard to mitigate these threats. Finally, we conclude this dissertation and propose future research paths in Chapter 8.

CHAPTER 2

BACKGROUND

2.1 Enterprise Internet-of-Things

In this section we introduce concepts on E-IoT systems essential for this dissertation.

2.1.1 General E-IoT

The need for automation in smart buildings, luxury homes, commercial, and industrial applications has existed since the 70's [Sul]. As such, there are many different use-cases where E-IoT is the best solution for automation and integration of multiple devices. Automation may be done in a single-room systems (e.g., a theater, a conference room) or have multiple rooms or floors under the same system. The expandable nature of E-IoT systems allows for small or large smart systems and integration between these systems. Figure 2.1 highlights applications of E-IoT in smart buildings. For instance, a smart office may be automated with CCTV systems, lighting control systems, and access control components with an E-IoT system. As such, E-IoT systems are customized for each specific application and deployment. We highlight some E-IoT use-cases on smart buildings, where these use cases can work together under a single E-IoT system. As such, if the E-IoT system is compromised, the integrated devices may also be compromised.

Lighting Control. Any control of physical lighting or electrical loads by an E-IoT system (e.g., lights, fans, outlets). E-IoT systems may be used in this use-case to schedule light events, program independent keypads, and allow remote control of lighting functions. E-IoT allows users to control their lights remotely, schedule light

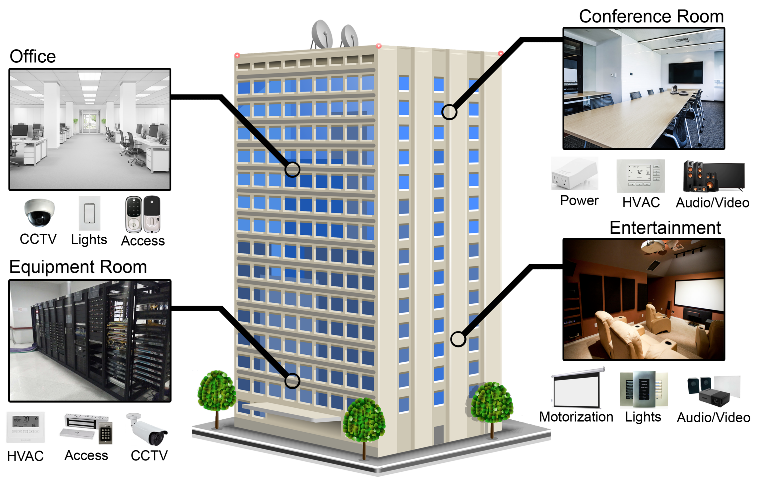


Figure 2.1: Common use-cases of of E-IoT systems.

events (e.g., wake up, turn outdoor lights on sundown, and trigger light-based events from other devices.

Security and Safety. E-IoT systems are often integrated to control security components. This integration grants authorized users the ability to control security aspects of a location (e.g., CCTV systems, access control systems, motion sensors, fire alarms, security alarm systems). As such, E-IoT systems allow for remote access, control, and camera activation based on motion sensor triggers. E-IoT allows users to integrate other components such as lights with security systems. For example, an E-IoT system may start flashing lights when the alarm system is triggered.

Advanced Media Control. The control and management of media and audio/video (A/V) components with E-IoT systems (e.g., projectors, televisions, video distributions, HDMI networks, audio matrix management). E-IoT systems manage complex audio/video distribution networks from a single interface through audio/video zones, audio switchers, video switchers, and amplifiers. With the complexity of many A/V systems, E-IoT is a reliable method of control through a single user interface.

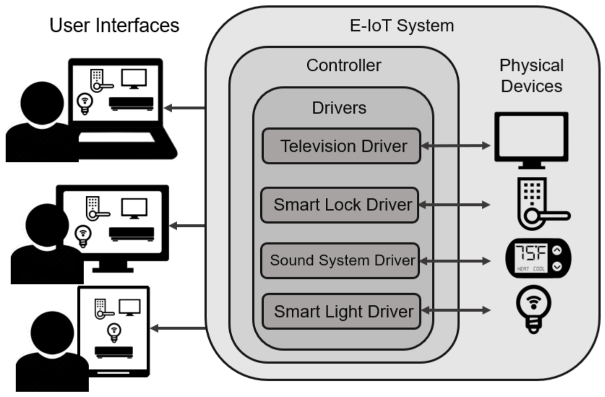


Figure 2.2: Architecture of a typical E-IoT system with user interfaces, controller, and physical devices.

2.1.2 Architecture of E-IoT Systems

Figure 2.2 depicts the general architecture of an E-IoT system. E-IoT systems have unique design and deployment practices that discriminate them from regular consumer IoT systems. In its most basic form, the E-IoT solution contains four core components: the physical devices, the controller, user interfaces, and drivers. As all installations are custom-made, E-IoT system deployments vary from system to system. The first component of E-IoT systems is the *physical devices*, which include any device integrated into the central system (e.g., sensors, televisions, lighting modules). To integrate physical devices, E-IoT systems use *drivers*, which provide the system with all the necessary information to integrate a device to an E-IoT system. Drivers contain information such as model number, protocol type, code, commands, and physical connections. Each device requires a driver to be integrated. In E-IoT systems, the *controller* serves as the central processing unit and stores all the drivers as well as user-specific custom programming required for the E-IoT system (e.g., scheduled events). Finally, *user interfaces* serve as the main point of interaction between users and the E-IoT system. After any third-party devices are integrated, the end-user can use user interfaces such as tablets, phones, and remotes to control integrated devices. For instance, if an E-IoT user wants to turn a light on, he/she may use a phone app as the interface to communicate with the controller. The controller then uses a smart light driver of an integrated E-IoT light to toggle the light at a user's request. As such, with any E-IoT actions many components are involved (e.g., hardware, networking, drivers, proprietary, wireless).

As designed, E-IoT systems may fulfill different purposes. One such purpose is specialization, such as centralized lighting control systems designed to control electrical loads in locations such as yachts or offices [Con14, oE18]. Another purpose of E-IoT systems is integrating previously separate components into a smart system (e.g., Savant, Crestron, and Control4); components integrated can then work together and interact as a single system [Aud]. For instance, integrating an alarm system with a lighting system allows a use case such as turning off all the lights when the alarm is activated. As such, E-IoT systems require trained installation and come at a higher cost than standard off-the-shelf systems. This added functionality over commodity systems has led E-IoT systems to become popular in expensive locations such as yachts, classrooms, smart offices, conference rooms, and luxury smart homes. Further, the installation of an E-IoT system is done by an *integrator*, a certified installer that performs the physical and software configuration for such a system. The configuration process requires specialized training and tools, which are provided by the system vendor to the integrator [Crea, Con10b]. However, hardware and software (e.g., integrated devices and drivers) used in E-IoT may also come from unverified third-party vendors and sources [Bla08].

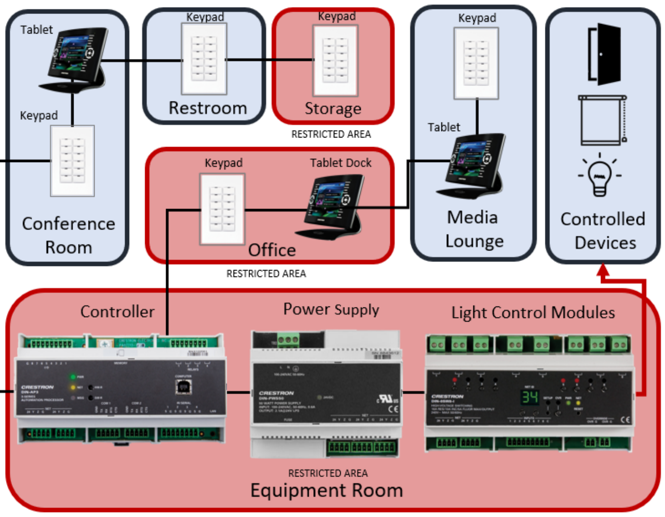


Figure 2.3: An example E-IoT system with wired bus communication and two daisy-chain paths. Restricted areas highlighted in red, common areas in blue.

2.1.3 E-IoT Lighting Control Systems

One specialized application of E-IoT systems are Lighting Control Systems (LCSs). LCSs are primarily used in custom-wired smart deployments to control and manage high and low-voltage wiring (e.g., lighting, motors, dimming, relays, magnetic locks). A generalized implementation of an LCS is shown in Figure 2.3, which consists of E-IoT components deployed in different rooms in a smart environment such as a business office in a smart building. The equipment room usually contains the core LCS E-IoT components, such as a controller, a power supply, and light control modules. In a similar manner to standard E-IoT systems, the *controller* is the core processing unit of an LCS and contains the execution logic for user actions on controlled devices (e.g., pressing button 1 on a keypad opens a security door, pressing button 6 turns off all the lights). This highly-programmable component is configured by the integrator during deployment or maintenance stages of an E-IoT LCS according to a user's specification. The *power supply* powers keypads and other interfaces integrated into the core system as well as the controller and light

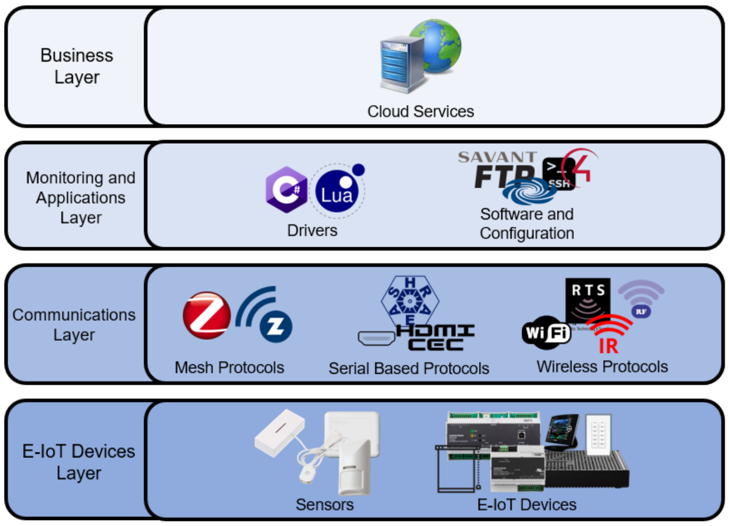


Figure 2.4: E-IoT system four-layer model used in this dissertation.

control modules. *The lighting control modules* are the physical high-voltage and relay based interfaces between the system and *controlled devices*. Controlled devices are any light fixture, shade, relay-operated door, or any physical device controlled by the LCS system. Finally, *the communication buses* are the daisy-chain lines that traverse through different equipment, rooms, and multiple *connection endpoints* where devices such as keypads, touchscreens, and other user interfaces connect to the communication bus. Such interfaces can be accessible by general users while other interfaces are only accessible in restricted locations. As Figure 2.3 shows, the daisy chain wiring saves integrators the need to wire all interfaces back to the main equipment room. With daisy chain, the physical wiring can connect from device to device instead of requiring that every individual device is wired back to the equipment room, saving in labor and wiring costs.

2.1.4 The Layered Architecture of E-IoT

As depicted in as depicted in Figure 2.4, the layered E-IoT solution as described includes four distinct layers: (1) E-IoT Devices Layer, (2) Communications Layer, (3) Monitoring and Applications Layer, and (4) Business Layer. The lowest layer, *E-IoT devices layer* includes the integrated E-IoT devices, physical interfaces used by devices, sensors, and any physical components of E-IoT systems. Next comes the *communication layer*, which possesses all the communication protocols (e.g., open-source and proprietary) used by integrated devices in the E-IoT devices layer. To manage communication, configuration, software, and programmed events in E-IoT systems, the *monitoring and applications layer* contains all software-based components (e.g., drivers, E-IoT applications, and configuration software) of E-IoT systems used by integrators and users. Finally, the *business layer* includes cloud components of an E-IoT system, for instance, remote services or remote storage used by an E-IoT system. The combination of these layers creates a unique technology solution that is highly customizable to any user's need. For instance, with an E-IoT system, a user can configure events such as a good morning timer which simultaneously plays a specific song, opens the shades, and turns on the lights every morning or a panic button to call the police, blare the alarms, and flash all the lights integrated to a system. Additional details on the four layers are as follows:

E-IoT Devices Layer. The E-IoT devices layer consists of all physical components of E-IoT systems. A physical component may be physical wiring, sensors, physical interfaces, or connection endpoints. E-IoT systems use many physical devices as part of their systems (e.g., motorized lifts, HVAC, sensors). These devices may be integrated for different applications. In some cases, they may be simply controlled by the system such as motorized projector lifts. Other cases may be external sensors such as water leak sensors to automatically shut off water valves prevent flooding.

Communications Layer. The communications layer contains all protocols, interfaces, and communication services used by E-IoT systems. This includes protocols in any component of E-IoT systems. To integrate a wide range of smart devices into the central system, E-IoT systems must support a multitude of communication protocols (e.g., Zigbee, Cresnet, Serial) used by smart devices. For instance, to integrate alarm systems to larger E-IoT systems, integrators will often use serial-based adapters, or an available IP interface [ADI20, Hon01].

Monitoring and Applications Layer. The applications and monitoring layer contains software-based components of an E-IoT system. For instance, E-IoT system configuration, drivers, firmware, or programmable behavior all can be considered part of the application layer. E-IoT systems must have the capability to be customized for every installation. As all deployments may be different and fit for different purposes, custom applications are a large part of E-IoT systems.

Business Layer. The topmost layer for E-IoT systems is the business layer, which handles all external cloud services used by E-IoT solutions. While not used in all implementations, some E-IoT systems rely on cloud computing and online services for features and integration. For instance, some E-IoT system use-cases require 'always offline' configuration after being deployed (e.g., yachts, remote locations, secure locations). Cloud services provide E-IoT systems with expanded capabilities, remote connections, and other services. For instance, E-IoT systems with Closed-Circuit Television (CCTV) components may use cloud storage services to store video feed in case the local video recorder is damaged or stolen [Cam20].

2.1.5 E-IoT Proprietary Protocols

E-IoT supports a variety of protocols, while some supported protocols are widely-known and well-documented (e.g., Zigbee, Z-wave, and TCP/IP), other protocols used by E-IoT systems are entirely proprietary in nature. As E-IoT system vendors need protocols designed for their specific purposes, they may modify existing known protocols or design entirely new protocols. Specifically, user interfaces such as keypads and touchscreens use wired and wireless protocols for communication purposes. For instance, in 2013, before Zigbee's rise in popularity, Control4, a vendor that offers E-IoT solutions, used a version called Embernet as a wireless solution [Con10c]. Lutron, a vendor that focuses on E-IoT lighting control systems, implemented a proprietary wireless communication known as Somfy's Radio-Technology Somfy (RTS) [Som20, Lut20]. *E-IoT systems also use proprietary wired protocols that use E-IoT communication buses.* For instance, Litetouch smart systems use a proprietary protocol for user interfaces [Lit06]. For similar purposes, Control4 employs a proprietary communication protocol [Con13a]. Savant uses communication buses and proprietary protocols for interfaces [Sav14]. Finally, Crestron, one of the most prolific E-IoT vendors, uses Cresnet, a form of a proprietary protocol over communication buses for interfaces and other components [Cre06]. *The technical specifications of these highlighted protocols are not publicly available, and thus their security, if any, is largely unknown.* Since the communication is simple, reliable, and allows daisy-chain wiring between interfaces, this communication is very prevalent in E-IoT. In comparison to protocols such as Z-wave and ZigBee, wired communication buses are preferred for E-IoT devices for three reasons. First, communication buses often provide power to the connected devices through the same communication line [Cre20a]. Second, communication buses are seen as more reliable than wireless protocols over long distances where mesh networking has range limitations (60 feet)

[Hen18]. Third, wired E-IoT communication is not as susceptible to interference as wireless communication, creating a more reliable system [WG06]. However, communication buses require physical cabling. As such, mesh wireless may still be used in E-IoT for smaller or retrofit deployments, where physical wiring is not a possibility.

2.1.6 E-IoT Drivers

One of the primary components of many E-IoT systems is the inclusion of *drivers* which may have different names depending on the manufacturer (e.g., Crestron modules, Control4 drivers). Drivers provide all the information and software modules necessary to integrate a device into a centralized E-IoT system. For instance, to integrate a Sony television into an E-IoT system, the controller must know what the protocol of communication is (e.g., IR, Serial, IP, Zigbee, Zwave), the physical inputs (HDMI ports, analog ports, etc), and the vendor-specific proprietary commands to interface with a device. Drivers are not limited to simply integrating physical devices and they also integrate services such as Weatherbug to add more functionality to an existing system [Con10a, Con].

Vendor Drivers. Drivers are inserted and configured during programming or maintenance stages of a smart environment by the integrator. Integrators may obtain drivers in three different ways: (1) they may get drivers directly from the E-IoT system software (pre-loaded drivers), (2) directly from a catalog hosted by the manufacturer of the E-IoT system devices, or (3) download from a third-party site in the Internet (from a third-party vendor or a developer). Vendors of E-IoT systems often validate drivers distributed in their platforms for functionality such as Control4's certified drivers [Con19]. However, with millions of different devices to be integrated, certifying every driver is not possible. *In effect, integrators may*

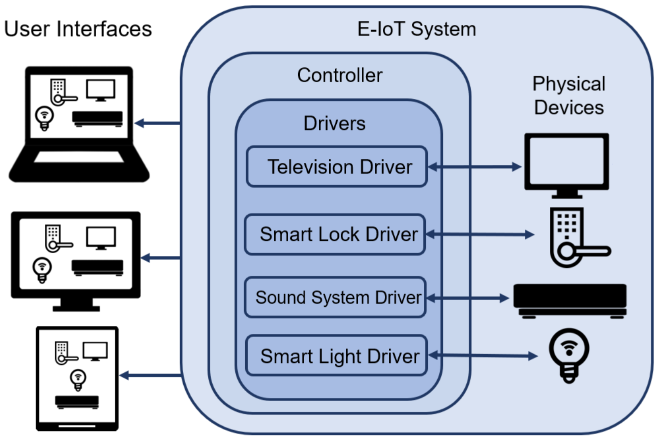


Figure 2.5: E-IoT system with four different control drivers, controller, and user interfaces. Individual devices are controlled through the user interfaces after being integrated.

be forced to use third-party drivers for their installations if no drivers are available for their specific solutions from the vendor or manufacturer. In this chapter, we focus on unverified drivers, or drivers available on third-party sites that have not been checked for malicious content.

Driver Verification Mechanisms. Many operating systems and platforms offer signature and verification mechanisms to guarantee the authenticity and integrity of software components. Microsoft uses digital signatures to guarantee the integrity of Microsoft drivers [Mic20]. Apple requires XCode and developer ID certificates to sign software available for MAC computers [App20]. In Linux, the kernel module signing facility secures Linux modules with signatures before installation [Ker20]. Further, Android developers have the ability to sign apps to guarantee the integrity of installed applications [And20a]. *In contrast to these well-documented practices, E-IoT vendors do not offer validation for E-IoT drivers for their systems. As E-IoT drivers often operate strictly on the proprietary software, traditional hardware-level driver defenses do not apply to application-layer based E-IoT drivers. As such, integrators are forced to trust unverified software which may be malicious in nature.*

Unverified Drivers. Integrators may opt for unverified, third-party drivers due to several reasons:

- *Driver Availability.* Verified drivers may not always be available to an integrator. Therefore, the only recourse to integrate a third-party device to an E-IoT system may be with an unverified driver from an untrusted source. Additionally, to integrate less-known devices, the driver has to be made by the manufacturer, who may be untrusted and their code closed-source. For instance, available integrator forums, offer a floury of unverified drivers for projectors, televisions, and other devices [C4F]. Additionally, many vendors do not offer E-IoT drivers for their devices, leading to third-party developers to offer their own drivers.
- *Cost.* Developers may charge for verified drivers (e.g., Atlona HDMI Switcher drivers for 110 USD), which in turn has to be paid by the integrator and end-user [dri20]. Integrators may be tempted to use free unverified drivers available on forums and online storefronts. Further, while paid drivers may be made by trusted developers, they are not necessarily verified by the E-IoT vendor.
- *Compatibility.* Devices may change commands and specifications when their firmware is updated [Pin19]. As such, verified drivers need to be updated to remain compatible with the latest models and firmware. To get a system running quickly after an update, an integrator may use an unverified driver that claims to run perfectly with a newer firmware version of the device when a verified driver is not available.
- *Phishing.* It is possible that an integrator may install an untrusted driver through a phishing link offering a "driver update" or a tampered vendor web-

site. It is possible to receive drivers through email attachments, impersonating a trusted vendor.

2.1.7 Consumer IoT vs. E-IoT

As commodity IoT smart systems have some limitations (e.g., scale, compatibility), E-IoT offers a solution for complex and reliable deployments. In this subsection we highlight the differences and benefits of E-IoT and why E-IoT solutions are chosen over commodity IoT. As such, E-IoT has some unique security concerns and threats. We outline some of these differences in Table 2.1.

Compatibility. As smart systems grow in scale, a user must determine the best solution to easily control many different devices. While commodity systems are limited in scale and compatible products, there are fewer limitations on what can be integrated into E-IoT. As E-IoT vendors offer components such as drivers, which are used to integrate third-party devices with E-IoT systems, many third-party devices are compatible with E-IoT systems. However, from a security standpoint, broad support of protocols can pose a threat as an attacker may be able to attack through many available protocols. This is true as more diverse systems have more possible points of failure.

Complexity. Commodity smart systems are designed to handle small deployments of IoT devices. While this use case is sufficient for most consumers, commodity smart systems are not a viable solution for large, complex deployments. For instance, multi-room video and audio distribution is one of the more complex applications of E-IoT. With audio/video switchers that can control up to 164 inputs and outputs, E-IoT becomes a reliable way to manage large systems and deployments. E-IoT systems also allow for a high degree of flexibility and customization. A number of

Table 2.1: Commodity IoT vs. E-IoT Solutions.

Commodity IoT Solutions	**E-IoT Solutions**
Simpler, easily deployable solutions	More complex, diverse smart solutions
Less compatible, approved devices	More compatible 3rd-party devices
Lower cost of installation and maintenance	High cost of installation, maintenance, and programming
User-deployed and maintained smart systems	Installer-deployed and maintained smart systems
More often open-sourced documentation available publicly	Closed-source systems, with no technical documentation available
Often cannot be deployed as completely offline systems	Must be deployed as always-offline systems in some use-cases

protocols and modes of communication are supported with drivers and expandable hardware components [dri20]. As a result, E-IoT can integrate more devices than consumer systems. All in all, the unprecedented level of complexity can mean that more vulnerabilities may occur at more stages and sectors of the E-IoT system in comparison to commodity IoT systems.

Delegation. As the installation of E-IoT components is often complex; many users opt to have installation and maintenance of E-IoT systems delegated to a dedicated contractor. As such, in a similar manner to electricians, plumbers, and other specialists, E-IoT integrators are contracted only for the installation and maintenance of E-IoT systems. In fact, the end-user does not need to understand the technical details of the E-IoT system, but he only needs to know how to operate the system, removing layers of complexity for any visitors. The delegation of installation and maintenance of E-IoT means that in addition to technical expertise, integrators must consider the security aspects of E-IoT systems. Thus, clients depend on their hired integrators for the security of their systems. As such, if an integrator is careless, or does not keep security in mind, the E-IoT system will be insecure without the owner's knowledge.

Cost. As E-IoT requires specialized integrators, custom programming, proprietary hardware, and dedicated technical support, the systems come at a higher cost. Further, the physical installation of E-IoT often involves fully rack-mounted, cable-

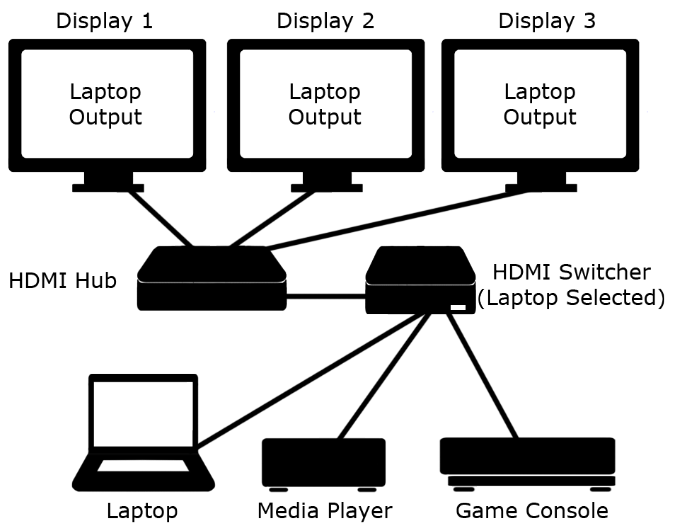


Figure 2.6: Example HDMI device distribution network including three displays sharing the same source image (Laptop). Usually, in bars and conference rooms, displays are chained via the HDMI cables.

managed systems throughout a building or home. While consumer IoT solutions are designed be affordable by end-users, E-IoT installations may be valued at hundreds of thousands of dollars depending on the complexity [Aud18]. The high cost of E-IoT systems may lead some users to wrongly assume their systems are secure.

2.2 Overview on Multimedia Communication

In this section, we present some necessary concepts about the Consumer Electronics Control (CEC) protocol and distributed HDMI-based multimedia.

2.2.1 The High Definition Multimedia Interface (HDMI)

The High Definition Multimedia Interface or HDMI, was developed with the purpose of digital Audio/Video transfer with seamless integration of communication features through the same connection [HDM09]. Through the 19-pin connector, HDMI transfers Audio, Video, Network, and CEC communication signals [HDM18]. With almost ten billion HDMI-capable devices distributed around the globe, HDMI has found a place in countless residences, offices and secure facilities and has become one of the most widely-deployed protocols worldwide [Wri18]. In its current form, HDMI is primarily used in high-bandwidth, video distribution applications between vastly different types of devices (e.g., Televisions, Bluerays, Media Centers).

2.2.2 HDMI Distribution Networks

HDMI deployments are not limited to one-to-one connections. Similar to Ethernet networks, there are many devices which control the HDMI signal flow and distribute signal in a controlled and organized manner. For instance, in Figure 2.6 the user maintains the same visual image over three displays, and switches between three source devices. This figure also shows the laptop selected as the active source over multiple displays. Depending on the device setup, there is a distribution of CEC through the same connection. We note the following components in an HDMI distribution and will refer to them during our dissertation.

Displays: Any device with a primary purpose of being an end-display such as a television or a projector.

Hubs/Splitters: Any device which primarily allows multiple video signals to be split to various displays from a single video input without switching.

Switches: Any device with a primary purpose of allowing various source device inputs to one or more display device outputs. They also perform switching between these sources to a different output(s).

Source Devices: Any device which is primarily an HDMI output-only devices such as a Chromecast or a laptop.

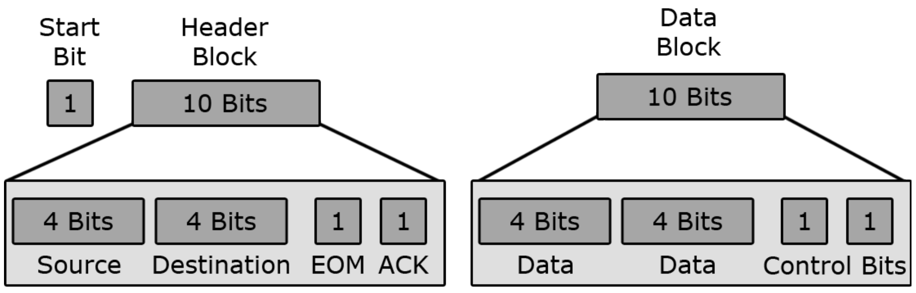


Figure 2.7: The CEC stack and structure as used in HDMI

2.2.3 The Consumer Electronics Control (CEC) Protocol

CEC was developed in to enable interoperability between HDMI devices, with full specification in 2005 [Hol05]. CEC signals are carried through Pin 13 as part of the HDMI interface [HDM18]. The communications in CEC are divided into 10-bit blocks that include a header, opcode, and data blocks. The flow of information is dictated by the header, the first eight bits note the source and destination. Message Destination may refer to a specific device by logical address or broadcast. Figure 2.7 shows how the CEC header allows for 16 unique IDs (4 bits). IDs 0-E specify device addresses while the last logical address (F) is reserved for broadcast within the HDMI distribution. This logical address assignment usually follows certain device-type guidelines. For example, displays are usually assigned to the logical address (0) and additional displays self assign to “free use” address (E).

CHAPTER 3

LITERATURE REVIEW

In this chapter, we present related work that is closely related to the research presented in this dissertation.

3.1 Attacks and Defenses on Smart Devices

Attacks and defenses against smart devices have been an ongoing topic of research in recent years.

3.1.1 Attacks on Alternative Threat Vectors

As early as 2013, works have highlighted various threats in smart devices and how attackers are in constant search of new threat vectors to infect and compromise smart devices [AP13, CBS$^+$18a, BCMU19, LBAU17, KBAU18, BAU19, SBAU19, ALAM19]. Further, research in alternative threat vectors such as USB shows how an attacker can easily compromise devices using insecure protocols [DEB$^+$19, DEBU19]. Very little research exists on the specific vulnerabilities of E-IoT systems or proprietary protocols. Coverage referring to such systems often comes in the form of vendor guarantees for security on traditional network attacks (e.g., TCP/IP components) [Cre20g]. Research on proprietary smart system protocols and threats has been mostly reserved to reverse engineering of protocols or encryption such as Somfy RTS [Pus16a, Pus16b]. Specifically for Crestron, the Cresnet protocol is closed-source; thus, the only prior research we identified is an attempt at creating a Cresnet protocol monitoring tool [Ste15]. Prior research on E-IoT lighting control systems (LCSs) by the U.S. Department of Energy has highlighted some security risks that come from LCSs [oE18].

3.1.2 Surveys on IoT System Security

The security of IoT smart devices have been an ongoing topic of research in the recent years. As such, a number of IoT security surveys have been conducted [ARC18, LYZ$^+$17, AOHA17, H$^+$19, HCS$^+$19, ODO17, DV17, BKP15, ZG13, KT15, YWY$^+$17, PG16, JVW$^+$14, BMV17]. Most of these surveys cover attacks, defenses, security challenges and general counter-measures in IoT, others are more specific. For instance, a survey by Hassan et al., highlights current research trends in IoT security [H$^+$19]. Other work has focused on IoT security aspects, such as the survey by Deogirikar et al., which focused specifically on known IoT attacks [DV17]. A survey by Yan et al. also covered the topic of IoT trust management, which may be applicable to E-IoT systems in some deployments [YZV14]. Individually, as early as 2013, works have highlighted threats in smart devices, and how attackers always search for new, unexplored threat vectors [CBS$^+$18a, BCMU19, LBAU17, KBAU18, BAU19, BAU18, Bab19, SBAU19]. We refer readers to the surveys Aris et al. [AOV18] for the intrusion detection and mitigation mechanisms applicable to 6LoWPAN-based IoT networks. For existing ML-based and traditional network intrusion detection systems (NIDS) we refer to the survey by Chaabouni at al, [CMZ$^+$19]. For research on attacks and defenses on wireless sensor networks, we refer to a survey by Butun et al, on emerging sensor threats and security [BÖS19]. Further, for works on distributed IoT devices, attack techniques, and defenses are covered in a survey by Vishwakarma et al, [VJ20]. Finally, individual defense mechanisms such as the one proposed by Forzin et al, investigates the use of Snort on Raspberry Pis to create an IDS for IoT systems [SMCB16]. Another notable example is Flowguard, an edge-defense mechanism proposed by Jia et al, to mitigate against IoT DDoS attacks [JZA$^+$20].

Some topics covered in this dissertation have had dedicated surveys examine attacks, defenses and threats for each topic. For instance, several surveys have covered jamming attacks, and defenses against wireless communication [BPP13a, MRR18, MGKP09, GLY14]. Surveys on communication are also relevant, with a number of surveys covering attacks, risks, and defense mechanisms for Bluetooth communication [LZ20, DG17, MT12, Dun10]. Sensory channels are also an active subject of research and relevant to E-IoT. As such, related surveys on the security of sensory channels touch upon subjects beyond E-IoT applications such as WSNs and large-scale sensor deployments [XSJ+10, SJRB17, SPA17, PS+09, MG10, SSS11, BT11, MOG11, VDM13, WS04, RM08]. Surveys such as the survey by Zhang et al. cover advances on the current security issues of industrial cyber-physical systems (ICPSs) [ZWF+21]. Further, as cloud computing is an active topic of research, a number of relevant surveys have also covered cloud computing threats and challenges [DDGD+19, LSR+15, Rya13, Sha14, SK11, GWS10, MPB+13, SJP16, FSG+14, PAS14, SC17, XX12, AADV15, HRFMF13]. Other works focused on cloud defense mechanisms, applying to different use cases beyond the scope of this survey and E-IoT applications [FSG+14, Rya13, SC17, AADV15, MPB+13, CDG+20]. Finally, some works have focused on security mechanisms for lesser-known IoT threat vectors, providing insight and defense strategies in fields such as medical implants [WDK+17], vehicular networks [SDWV20], mobile networks [NZV20], and smart grids [BSDV21].

3.2 Threats and Vulnerabilities on Multimedia Devices

Ongoing research in compromising A/V through unconventional means has been a topic of discussion among researchers. Within the scope of Smart TVs, Oren

& Keromytis describe a method of compromising connected Smart TVs through Hybrid Broadcast-Broadband Television (HbbTV) and web-based code injections [OK14]. Related work from Niemietz et al., on Smart TVs explores the attacks on Smart TVs through app-based approach [NSMS15]. Parts of this dissertation center on TV embedded applications, and the security flaws which may come from vendor-specific apps. On the topic of HDMI, research related to HDMI systems and their security issues has remained a relatively uninvestigated avenue or not systematically investigated by the research community. The most relevant work in HDMI systems is a 2012 work published by NCC Group, which focused on vulnerabilities with fuzzing [Dav13]. Similarly, Smith et al., presented HDMI-CEC as an avenue of attacks through fuzzing [Smi15]. Finally, a BlackHat presentation by Smith et al., covered CECSTeR, a fuzzing framework designed specifically to compromise HDMI-enabled devices through the HDMI-CEC communication bus [Dav12].

3.3 Industrial Communication Bus Threats and Security

Industrial Bus Security. In terms of industrial bus security, several researchers have proposed works in industrial control networks, in-vehicle networks, and other serial-based networks. Well-known industrial protocols, such as Modbus, DNP3, S7comm, and IEC 60870-5 employ serial-based communication buses for industrial devices. Industrial networks can be targeted by several threats such as man-in-the-middle (MITM). In this regard, the survey of Conti et al. [CDL16] highlighted MITM attacks. In terms of the studies aiming to protect industrial networks, the works of Dudak et al. [DGS$^+$19] and Wilson [Wil18] aimed to incorporate confidentiality, integrity, and authenticity to industrial protocols against threats such as MITM attacks. As a standardization effort to ensure the security of industrial protocols, including serial-based communication buses, the IEEE 1711.2 working group proposed the Secure SCADA Communications Protocol [IEE20]. A comprehensive review of security challenges regarding both serial and non-serial-based communication buses used by the industrial protocols can be found in the study of Volkova et al. [VNBd19]. Further, solutions were proposed by researchers to detect attacks targeting serial-based communication buses. To name a few, Eigner et al. [EKT16] proposed an ML-based defense approach using K-nearest neighbors towards detecting MITM attacks against industrial control networks (i.e., Modbus). Similarly, Lan et al. [LZSL20] proposed a method of classifying S7comm traffic to detect data tampering caused by MITM attacks. Controller Area Network (CAN) bus used in in-vehicle networks employs serial communication [SKK16]. CAN bus security has been a very active topic of research, and an extensive analysis of intrusion detection systems in this regard can be found in the work of Young et al. [YZOB19]. In the work of Buttigieg et al. [BFM17], the researchers investigated security issues and executed MITM attacks against a CAN network. Morgner et al. [MPSB18] proposed a novel attack that is based on third-parties deploying a malicious implant that tampers with the serial communication of the target hardware. In their study, the malicious implant is controlled by a remote attacker via IoT communication protocols and is used to conduct various attacks.

CHAPTER 4

ANALYSIS OF CURRENT E-IOT LANDSCAPE

4.1 Introduction

The introduction of modern smart consumer electronics has led to the widespread adoption of smart devices, with over 45 million smart home components sold worldwide [ABC+18, IoT18]. Most users are familiar with commodity systems, off-the-shelf smart systems that are easily installed by the average end-user without specialized training (e.g., Samsung SmartThings, Google Home) [Sul, BSAU18]. However, in more complex installations, where robust, secure, and reliable smart solutions are needed, Enterprise Internet-of-Things (E-IoT) systems (e.g, Crestron, Control4, Savant, RTI) are accepted solutions. In contrast to commodity systems, E-IoT offers customized deployments, with more use-cases and applications. Offering users a broad set of compatible devices (e.g., sensors, Audio/Video equipment, interfaces), protocols (e.g., Zigbee, Z-wave, IP, proprietary protocols), custom programmed behavior, and system User Interface (UI) customization. As such, E-IoT systems are found in locations such as smart offices, smart buildings, luxury smart homes, yachts, and secure conference rooms (as illustrated in Figure 6.1).

While the security of many emerging commodity systems is well-understood due to prior research and mainstream knowledge, the security of E-IoT systems has been largely overlooked [BAU18, Bab19, BAU19, DEBU19, BCMU19, DEB+19, LBAU17, KBAU18, SBAU19, CBS+18a, AADB17, AFA+20, AAUA18, SAU20, SPA+18, USB, CBS+18b, NSRU19, NSRU20, NSBU20, CMT+, SBC+20, BAR+20, MBY+19, DBU20]. As such, the lack of research and awareness coupled with the cost of devices and installation of E-IoT has led many users to mistakenly assume that E-IoT systems are completely secure. As E-IoT systems follow a unique design with specialty devices, proprietary software, and a large number of compatible protocols, there is a need to research unique threats and security of E-IoT systems. Further, E-IoT systems have been increasingly popular in smart installations, with Crestron growing to 1.5 billion dollars of annual revenue in 2018 and Control4 deploying over 15 million smart products in over 400,000 installations worldwide [Mar18, Con20b]. With many of these systems present in high-profile locations, understanding threats and defense strategies for E-IoT systems should be of great importance. However, no current research focuses on E-IoT system components, attacks, threats, and relevant defenses of E-IoT systems. We believe that this research gap in the literature is notable considering the prevalence of E-IoT deployments and ever-increasing attacks against smart systems. To address this research gap and analyze the security of E-IoT systems, we first divide E-IoT into four distinct layers: E-IoT Devices Layer, Communications Layer, Monitoring and Applications Layer, and Business Layer. As such, we consider E-IoT components at each layer, the associated threats, attacks, and defense mechanisms. Additionally, we present key observations in E-IoT security and provide a list of open issues that require further research.

Although there are existing studies on IoT systems, this chapter focuses solely on relevant threats and solutions to E-IoT systems. This study aims to provide users with adequate information on E-IoT system components, vulnerabilities, attacks, and defenses. With this chapter, we also aim to encourage further research and development from the research community on the topic of E-IoT systems. For instance, this chapter highlights widely-used E-IoT proprietary technologies that have seen no security scrutiny and thus have relied on security through obscurity for decades. This chapter may be valuable to researchers, E-IoT vendors, users, installers, and manufacturers that want to improve their security practices. Further, users who do not know about E-IoT concepts may find this study a beneficial resource. Ultimately, this chapter sheds light on the security implications of E-IoT systems and raises awareness of security practices, protocols, and viable threats against E-IoT systems.

The contributions of this chapter are as follows:

- We highlight popular E-IoT system platforms and identify security challenges in these systems.
- We categorize and analyze E-IoT components, threats, attacks, and defenses by dividing E-IoT systems into four distinct layers.
- We present the need for further research in E-IoT systems and a number of proprietary technologies used in E-IoT.
- We open discussion on the security of E-IoT systems, and related defense mechanisms.

4.1.1 Differences from Existing Works.

This chapter differs from previously discussed works as it focuses on the insecurities, possible threats, and defenses applicable to E-IoT. To the best of our knowledge, this is the first study that focuses solely on E-IoT systems and their security, categorizing E-IoT systems into four unique layers. Specifically, we categorize E-IoT components into four distinct layers, (1) the E-IoT Devices Layer, (2) the Communications Layer, (3) the Monitoring and Applications Layer, and (4) the Business Layer. We take this approach as E-IoT system architecture differs from many IoT systems. Further, we present a threat model for each distinct layer of E-IoT, as each layer may present different threats and require different capabilities from attackers.

4.2 Taxonomy and Scope

In this section, we firstly explain the taxonomy we employed for the categorization of the security issues and the associated defenses for E-IoT systems. Following that, we highlight the topics covered in this chapter. While covering the topics that are closely related to E-IoT, we do not consider the topics that are not directly related to E-IoT or that are common to general computer systems in the scope of this study.

4.2.1 Taxonomy

In this chapter, we divide E-IoT into four distinct layers: E-IoT Devices Layer, Communications Layer, Monitoring and Applications Layer, and Business Layer. Based on the layers of E-IoT, we consider E-IoT components and we categorize the associated threats, attacks, and defense mechanisms accordingly.

4.2.2 Scope

Scope of E-IoT Devices Layer. For the E-IoT devices layer section, we cover attacks (e.g., sensory attacks, node theft, battery exhaustion) and defense mechanisms that target components at the E-IoT devices layer. Included in these topics are E-IoT devices, supply chain attacks, and physical access attacks relevant to E-IoT systems. Topics outside of the scope of E-IoT devices layer are attacks on chipsets (e.g., processor side-channel attacks) and other physical devices that are either widely researched or not unique to E-IoT.

Scope of Communications Layer. This chapter covers communication interfaces, publicly-documented protocols, proprietary protocols, and other relevant communication components as part of the communications layer. As such, this chapter covers jamming attacks and other well-known attacks against public and proprietary protocols used in E-IoT. Finally, communication protocols such as TCP/IP, cellular communication, long-range radio protocols such as LoRaWAN, and their respective attacks are outside of this chapter's scope as they are not common in E-IoT use-cases.

Scope of Monitoring and Applications Layer. Topics in the monitoring and applications layer include E-IoT software, configuration, and software services. Topics outside of this layer's scope are operating systems as they are a common topic of research and not exclusive to E-IoT systems. For instance, Linux-based operating systems are common in E-IoT and other smart systems, making Linux a common topic of research. Also, outside of this layer's scope, web-based Distributed Denial-of-Service (DDoS) attacks, mobile application threats, ransomware, firmware attacks, and common software vulnerabilities.

Scope of E-IoT Business Layer. Relevant topics to the E-IoT business layer include remote access cloud services, maintenance services, and CCTV data storage. As the E-IoT business layer is not employed by all E-IoT systems, and cloud security is a diverse field, some topics are not covered. Topics outside of this chapter's scope are encrypted storage access, computation of stored E-IoT content in cloud environments, online microservices, advanced persistent threats, virtualization technologies, general data storage, and other cloud concepts that are uncommon for E-IoT.

4.3 E-IoT Devices Layer: Components and Security

In this section, we cover the E-IoT Devices layer, threats, defenses, and their implications. First, we introduce components of the E-IoT devices layer, and then cover threats and attacks. Finally, we give an overview of possible defense and mitigation mechanisms.

4.3.1 Elements of the E-IoT Devices Layer

E-IoT Devices. Many devices, such as sensors and lighting controllers, are integrated into E-IoT systems to expand the use cases and functionality. Integrated devices may serve specific purposes (e.g., television, media player) or be a part of larger use-cases such as power control modules for lighting control systems.

Sensors. E-IoT and E-IoT-integrated devices will very often have sensors used to trigger programmed actions in an E-IoT system. Sensors may play a role in E-IoT in several different ways. For instance, individual sensors (e.g., glass break, motion, contact) can be integrated directly into an E-IoT system thanks to the official support of E-IoT vendors for several protocols (e.g., Zigbee, Bluetooth, Z-wave) [Con20c, Sav20a, Cre20c]. In addition, an external system, such as an alarm system, can be configured to work with an E-IoT deployment. For instance, an E-IoT deployment with water leak detection sensors and automated valves can be configured to close a leak, inform the user via text, and display a message on E-IoT interfaces about the issue [Con15b].

4.3.2 Threat Model for E-IoT Devices Layer

For this chapter, we consider Mallory to be an attacker that targets E-IoT systems through different E-IoT layers. In the E-IoT Devices Layer, Mallory compromises the E-IoT system solely through physical access to interfaces, devices, cabling, and unattended equipment. To compromise an E-IoT system through the E-IoT devices, Mallory is assumed to have physical access to devices during the manufacturing, installation, operation, or maintenance stages of the E-IoT system. Mallory is capable of this, as security for device-layer components in E-IoT environments relies on the specific devices and the integrator's installation practices (e.g., directional antennas, access restriction, tamper-proofing). We explain Mallory's possible actions at different stages of the E-IoT devices as follows:

Manufacturing and Transportation. In the manufacturing or transportation stages of E-IoT equipment, Mallory may have several opportunities to compromise a device. During these stages, insiders (e.g., manufacturing workers, delivery drivers, packaging personnel) all have direct access to the E-IoT device before a device is installed, making supply chain attacks possible for Mallory, who may be in the role of an insider attacker. Further, Mallory could be an employee of outsourced manufacturing, and as such, it may be particularly difficult to prosecute Mallory during the manufacturing and transportation stages. In this role, Mallory may target E-IoT devices specifically if she has prior knowledge that E-IoT components may be installed in sensitive locations (e.g., secure conference room, access control, enterprise network).

Deployment, Operation, and Maintenance. E-IoT installations may see visitors such as presenting guests or maintenance workers that have direct access to E-IoT equipment. As such, Mallory as a visitor may perform a node capture attack and further compromise an E-IoT system. Additionally, if Mallory is a more knowledgeable attacker, she may perform sensory channel and side-channel attacks. In other roles, such as a role where Mallory is an IT professional, she could compromise devices in the same manner.

4.3.3 E-IoT Devices Layer: Attacks and Vulnerabilities

In the following subsection, we cover attacks and vulnerabilities relevant to the E-IoT Devices Layer. These attacks can be performed by Mallory as highlighted in the threat model.

Supply Chain Attacks. Even before E-IoT devices reach integrators, installers, and consumers, devices may be compromised during manufacturing and distribution stages. Several articles have highlighted supply chain threats and provided examples of how systems in different industries (e.g., medical, banking) have been targeted and compromised through supply chain attacks [Mil13, Nat20a, Edw20, FH18, Mar19, YMF20]. As specific industries have been targeted, it is reasonable to assume that E-IoT systems may be a future target for supply chain attacks. With the price-point of E-IoT systems and high-profile clients, attackers may find E-IoT systems an attractive target for supply-chain attacks. Work by Farooq et al. analyzed the risks and research challenges in IoT supply chain security [FZ19]. Their work highlighted three types of interactions in the supply chain: device-supplier interactions, supplier-supplier interactions, and device-device interactions. In device-supplier interactions, a supplier provides maintenance, security patches, and upgrades to devices. Supplier-supplier interactions are when suppliers use different companies to distribute devices. Finally, device-device interactions occur due to the inter-connectivity of devices in the supply chain, that is, communication between devices (e.g., configuration) in the supply chain. As such, an attacker could compromise a device at any of these interactions. The UK's National Cyber Security Centre highlighted several attacks that can occur from supply chain interactions [Nat20b]. For instance, malware inserted into vendor websites or devices can "trojanize" devices before the devices leave the supply chain. As compromised software is very difficult to detect at the source, target companies may not suspect the software is altered or illegitimate. Supply chain threats also extend to embedded hardware such as chipsets, unauthenticated parts, and counterfeit components inserted in the supply chain. These counterfeit components may impact systems by being of lower quality [Gor12]. In other cases, hardware threats extend to hardware trojans, which have been an ongoing topic of research [BR15, TK10, KTC$^+$08]. In this case, malicious chipsets and electronic components are inserted into devices, usually during manufacturing stages, compromising the integrity of the device. These types of attacks have been observed, in a notable case where Chinese manufacturers infiltrated 30 large U.S companies using malicious hardware components embedded in networking devices [RR18]. As such, E-IoT can easily become a target to a variety of supply chain attacks, as distribution, manufacturing, and installation stages of E-IoT provide wide opportunity to compromise E-IoT devices.

Physical Attacks. In any E-IoT deployment, E-IoT devices will be found throughout the location or establishment. Some of these devices may be installed in private, unsupervised areas (e.g., a keypad in a closet, an empty conference room). As such, it may be possible for visiting attackers to interact with physical devices integrated into E-IoT systems. As several vulnerabilities against physical devices rely on physical access to E-IoT devices and interfaces (e.g., node capture, tampering, button resets, theft). Physical access to devices and E-IoT components may allow an attacker to perform malicious actions on E-IoT devices, enabling programming mode, hard resets, or otherwise, change the configuration in E-IoT devices that can render them inoperable. For instance, "button sequences" may present a vulnerability to E-IoT devices. Reset sequences are used for purposes such as changing a device's configuration, resetting a device to factory settings, or even gain information about devices [Cre20h, Con13b]. As such, an attacker can use these sequences to alter physical devices' configuration, gather information, or otherwise cause E-IoT components to become unavailable to the E-IoT system.

Physical access to E-IoT devices allows malicious actors to perform *node capture attacks*, where devices are physically captured (or stolen) to gather sensitive information about a system [BTJS12]. Although there is no study on node capture attacks in E-IoT, attacks applied on related domains may be applicable to E-IoT as well. In this respect, work by Wang et al. covered the implications of node capture attacks in wireless sensor networks (WSNs), which are relevant to wireless E-IoT devices (e.g., sensors, interfaces, remotes) as they often share the similar communication technologies [WWT+20]. The authors of the work identified ten unique vulnerabilities that can be exploited through node capture attacks affecting session keys, users, sensor nodes, gateways, and availability of the network. As such, the attacks could acquire communication keys, eavesdrop on messages, impersonate devices, track user activity, and impersonate users. Several other pieces of literature have discussed node capture attacks that exploit vulnerabilities to gather keys from connected devices [BBP10b, BBP10a, DLD09, KSP07, MT11, RAL+17]. The work of Lin et al. focused more on the efficiency of node capture attacks and introduced the full graph attack (FGA), with two optimal algorithms for this attack [LWYY15]. The attack specializes in compromising relationships between nodes and paths. As such, the attacks reportedly increased the efficiency by 50% compared to previously proposed attacks.

Side-Channel and Sensory Channel Attacks. *Side-channel attacks* are threats against the implementation of computer systems, rather than inherent weaknesses. These attacks allow attackers to compromise a system or component through an indirect channel (e.g., timing information, power consumption, electromagnetic leaks, auditory channels) [Sta10]. A number of E-IoT components may be vulnerable to side-channel attacks through electromagnetic (EM) approaches. For instance, a study by Smulders et al. on serial-based communication suggested that electromagnetic radiation can be used to eavesdrop on physical cables and serial-based communication as a type of side-channel attack [Smu90]. These methods take advantage of a known fact that most electronic equipment emits electrical radiation, and bit amplitude in serial-based communication is relatively larger than other signals [HW88]. Their tests performed with a standard AM/FM receiver antenna allowed intercepting and reading signals going through the wire. The work concluded that data signals transmitted over serial-based communication could be intercepted from several meters away. Further, this work noted that the equipment required to perform these scans is inexpensive and readily available, as such, similar attacks may be possible in similar unsecured networks with improved equipment and techniques. Legacy systems, or systems without authentication or encryption may be especially vulnerable to these or similar attacks. Electromagnetic attacks are not limited to wiring, as work published as early as 1986 by Eck et al. noted that electromagnetic radiation eavesdropping attacks are possible in video display units [VE85]. Further work by Kuhn et al. noted that while technology has changed, electromagnetic eavesdropping can work on more modern LCD displays [Kuh04]. Researchers have found other ways to compromise systems that may be relevant to E-IoT. For instance, Savage et al. showed that with recorded video (e.g., from a CCTV system, intercom systems), an attacker could use passive sound recovery to eavesdrop on conversations [Sav15]. Further work by Davis et al. demonstrated that an attacker could also use vibrations on object surfaces for eavesdropping under certain conditions (e.g., visible glass or water)[DRW$^+$14].

As E-IoT may control smart lights, light-emitting devices, and light sensors, threats posed by visible-light side-channels may affect E-IoT deployments. Information leakage through optical side-channels has been an active topic of research. For instance, Xu et al. created a video recognition attack where they were able to identify a video being watched on a television using the light emitted by the television through a window [XFM14]. Similar works as presented by Schwittmann et al. used ambient light sensors on smartphones and smartwatches to perform similar attacks [SBM$^+$17, SMW$^+$16]. Alternatively, Light Ears, presented by Maiti et al., proposed a new attack vector designed to infer a user's private data and preferences from smart lighting media visualization features [MJ19]. Based on this research, researchers used the light and sound intensity of smart lights to infer ongoing audio and video. Alternatively, covert optical channels have been researched, with Loughry et al. providing the first call of attention to possible information exfiltration attacks on air-gapped systems by using LED light indicators [LU02]. Similar data-exfiltration attacks have been demonstrated using LCD displays [GHKE16], infrared [ZZY18], security camera infrared lights [GB19], air-gapped systems [GE18], and smart lights [RS16].

As E-IoT systems rely on sensors for accurate measurements and to trigger preprogrammed events, physical sensor threats are a concern for E-IoT. Sensor threats and security have been an active topic of research with multiple surveys. However, most of these surveys focus on sensor communication and wireless sensor networks [XSJ$^+$10, SJRB17, SPA17, PS$^+$09, MG10, SSS11, BT11, MOG11, VDM13, WS04, RM08]. As sensors are a vast research topic, different attacks and vulnerabilities on sensors have been discovered that can be applicable to E-IoT. Analog threats such as sound waves can maliciously influence an accelerometer's output and cause unintended effects in an E-IoT system configured to respond to specific readings [KF18]. Other proposed attacks, such as DolphinAttack, target microphones through inaudible voice commands, can be effective against E-IoT systems that integrate voice recognition and microphones [ZYJ+17]. With many sensors lacking security mechanisms, E-IoT systems may be particularly vulnerable to sensor attacks. Work presented by Uluagac et al. summarized several sensory channels in cyber-physical systems (CPS) and devices that can be targeted by an attacker [USB14]. These channels are the light, seismic, acoustic, and infrared channels. The *light channel* functions through light sensors and ambient light temperatures. The light channel may be used in E-IoT to trigger programmed events at nighttime. *Seismic channels* are vibrational channels that can be detected by devices such as accelerometers that detect the physical movements of a device. *Acoustic channels* are based on sound waves and can be comparable to sonar technologies. Finally, *infrared channels* use infrared emitters for navigation assistance and can present a covert side-channel for attacks as it is not visible to the human eye. Further, this work highlighted that these sensory channels can all be used to trigger malware (e.g., keyloggers, DoS, spyware) already implanted into devices by attackers and that traditional security mitigation strategies do not defend against sensory channel attacks.

Other physical attacks on sensors rely on multiple sensors to function. One of the most researched examples is keystroke inference on devices with unprotected sensors [Spr14, CC12, AHIN13, HGC+19, OHD+12, MVCT11, NSN14, LS19, XBZ12, MVBC12, Ngu15, HB18, LCY+18, RRC16, VP09]. While keystroke inference research centers around mobile devices, it may be relevant to E-IoT. Many E-IoT interface devices (e.g., dedicated touchscreens, keypads, remotes) have similarities with mobile devices as they possess several sensors and receive user input. Many of these keystroke inference attacks rely on multiple sensors in different sensor channels to infer sensitive information (e.g., what a user is typing from sensor activity). For instance, a work presented by Han et al. introduced PitchIn, a method for exploiting non-acoustic sensors used in smart environments that can allow an attacker to perform speech reconstruction attacks [HCT17]. With this method, multiple sensors (e.g., geophones, accelerometers, gyroscopes) were used to reconstruct audio and perform word recognition in the mentioned work.

Battery Exhaustion Attacks. As a number of E-IoT devices are battery-powered (e.g., remotes, interfaces, sensors, etc.), an attacker could use battery exhaustion attacks to impact the operation of E-IoT systems negatively. Battery Exhaustion attacks are a type of DoS attack that aims to deplete the batteries of devices by forcing the device to perform an excess amount of tasks [SKR17, BCM15]. Moyers et al. presented the effects of wireless and Bluetooth battery depletion attacks on mobile devices [MDMT10]. This work classified three distinct types of battery exhaustion implementations, *service request power attacks*, *benign power attacks*, and *malignant power attacks*. For service request power attacks, attackers target devices by making repeated requests to these devices and exhaust power through the wireless network interface card. In benign power attacks, victims are forced to perform repeated tasks (e.g., data processing, diagnostics) and consume large amounts of power. Finally, malignant power attacks are usually implemented with malware designed to increase power consumption in a device (e.g., increasing the CPU clock). Other work by Martin et al. highlighted the effects of these attacks on wireless devices, noting that damage caused by battery exhaustion attacks may also cause long-term damage to battery life in addition to a DoS condition when a device becomes unavailable [MHDK04].

4.3.4 Mitigation of E-IoT Devices Layer Attacks

In this subsection, we highlight possible mitigations to E-IoT devices-layer threats.

Supply Chain Defenses. A few solutions were proposed in the literature to defend against supply chain attacks. In order to secure the device endpoints, Yang et al. proposed an RFID-based solution that authenticates devices once they are deployed [YFT15]. This work was taken further with the introduction of ReSC by the same authors, a solution proposed to defend against the theft of authentic smart devices, and the insertion of counterfeit malicious devices [YFT18, YFT17]. Another approach by Chamekh et al. proposed the use of a Merkle tree management framework applied to supply chain architecture to provide a more trusted system and defend against supply chain attacks [CHEK18]. Alternatively, Chao Lin et al. proposed a blockchain-based framework to protect and guarantee anonymous authentication, auditability, and confidentiality for the supply chain [LHH^{+}18]. Another solution proposed by Jangirala et al. proposed the use of blockchain-enabled RFID-based authentication protocol (LBRAPS) for supply chain security [JDV19]. During transportation stages, tamper-proof and tamper-evident packages and equipment may also prevent unauthorized attackers from tampering with devices before they reach a client [EEEE07, Rob19]. The European Union Agency for Cybersecurity (ENISA) provided comprehensive guidelines for IoT supply chain security [Eur20]. These guidelines divide defense strategies into several relevant stages relevant to E-IoT: product design, component assembly and embedded software, device programming, platform development, distribution and logistics, technical support & maintenance, and device recovery & repurpose. For product design, guidelines dictate that secure software libraries and cryptographic practices, sabotage prevention, tamper-resistant software and hardware, and chain of trust are design practices that may prevent supply-chain attacks. Further the ENISA guidelines highlight that vendors can take some preventative measures such as, working with suppliers that provide security guarantees, maintaining transparency, having a skilled workforce, promoting security awareness, and developing novel trust models.

One of the largest topic of research is counterfeit components inserted in the supply chain, as such best practices and solutions have been proposed. For instance, ENISA guidelines highlight that parts used during manufacturing should be authenticated to prevent counterfeit components from entering the supply chain. Further, to prevent defective components, ENISA also advises for quality control and testing of parts to prevent defective components [Eur20]. Surveys conducted on the topic of counterfeit devices and hardware Trojans have suggested several solutions [BR15, TK10]. First, optical inspection-based detection relies on reverse engineering to detect Trojans. As such, techniques such as scanning optical microscopy, scanning electron microscopy, and pico-second imaging circuity analysis are used. Images captured with these techniques are then compared to benign chipsets provided by the designer. Testing-based detection techniques use functional testing to detect Trojans. As such, a functional set of vectors need to be designed for each chipset. Side-channel detection approaches rely on factors such as power consumption, EM emissions, and time delays to detect anomalies. Such approaches can also be used to detect Trojans. For instance Agarwal et al. used Principle Component Analysis to create a side-channel fingerprint of a circuit and compare it to a known, benign model [ABK$^+$07]. Run-time detection approaches are also used, usually combining hardware and software to detect Trojans. For instance, DEFENSE is a proposed monitoring framework that operates at device run-time to detect hardware anomalies and Trojans [AB09]. Finally, invasive techniques modify integrated circuit's structures to avoid the insertion of hardware Trojans. The studies have shown that hardware obfuscation methods can prevent Trojan insertion and assist other detection methods [CB08, CB09a, CB09b].

Physical Security. Physical security of cabling and devices is an important part of E-IoT deployments as E-IoT devices can be stolen, tampered with, or otherwise damaged. Vendors implement some physical mitigations and best practices for many of their devices. Additionally, E-IoT systems make an effort towards tamper-proofing their systems and offer suggestions on physical installation. For instance, Control4 released an exterior installation security best practices document [Con12]. This document highlights several important points on exposed devices such as door stations used for gate access and intercom. First, installers are encouraged to use standard tamper-resistant security screws shipped with devices to prevent opportunists from stealing or tampering with devices. Second, relays used to open security gates should not be connected at the door station itself and instead to a relay inside the building. Relays' endpoints should be in a secure location as physical attackers may compromise devices by tampering with relays and gain unauthorized access to locations. Finally, they acknowledge the risks associated with the network cable running to public interfaces (e.g., door stations, intercoms) and highlight solutions such as network isolation, MAC address filtering, and wireless door station access as possible solutions. In some instances, E-IoT components may only be removed with custom tools to prevent theft and tampering. For instance, touchscreens may come with a special tool so that an unprepared attacker cannot easily remove the interface [Con15a]. Finally, integrators and users should take advantage of monitoring tools (e.g., wireless monitoring, IP monitoring) to identify devices that fall offline to know if they have been tampered with. Practices used for loss prevention may also be useful for E-IoT. Concepts such as beacons, smart tags, and geo-fencing may prevent node capture attacks and alert integrators before an attack occurs [Ing19]. Integrators may also take certain steps in the installation to make sure that E-IoT devices are secure. For instance, installers should follow best practices, place sensors in places where they are not easily reachable and do not leave any exposed wiring in installations. As noted earlier, physical access to exposed wiring and devices would make it trivial for an attacker to compromise an E-IoT system in public and unmonitored areas. Further, installers and users should consider physical access control to prevent access by unauthorized users.

Side Channel and Sensory Channel Defenses. There exist a number of defense solutions against side-channel attacks. For instance, for EM and many side-channel eavesdropping attacks, physical security and encryption provides a level of defense. For attacks that rely on sound, AuDroid is a policy-based framework for smart devices proposed by Petracca et al. [PSJA15]. AuDroid controls information flow in audio channels and notifies users when audio access is requested. Access control frameworks such as these may present a viable solution for side-channel attacks where sensory and audio channels can be abused. A number of defense mechanisms proposed for sensors and wireless sensor networks may be applicable to E-IoT against side-channel attacks. For instance, for sensors in mobile devices such as phones, security mechanisms have been an ongoing topic of research [SZLJ14, WYZ+15, EGH+14, WH14, XZ15]. However, many of these proposed solutions rely on software-based approaches to defend against sensor-based attacks. Alternatively, solutions such as frameworks and intrusion detection systems have been proposed for wireless sensor networks and may apply to large E-IoT deployments configured to rely more heavily on sensors for programmed events [Str07, IDF07, FKWL13, PAL+08, YT08]. One example, 6thSense, a sensor-based defense mechanism by Sikder et al. takes a machine learning approach to detect malicious behavior occurring in smart devices [SAU20, SAU19, SAU17]. The proposed solution relies on sensor co-dependence, sensor sampling, and real-time monitoring. Since E-IoT systems may share some similarities to proposed solutions (e.g., multiple sensors, centralized design), these defense mechanisms may apply to E-IoT against side-channel and sensory channel attacks. While many of these solutions may protect against side-channels, some side-channel attacks (e.g., LightEars) do not have direct solutions proposed beyond physical security and require future research.

Battery Exhaustion Defenses. A number of mitigation strategies have been proposed to combat battery exhaustion attacks on wireless devices. The solution for E-IoT may be entirely dependent on the type of the system. For instance, battery exhaustion defenses may be different in a Zigbee vs another wireless-based deployment. Buennemeyer et al. proposed Battery-Sensing Intrusion Protection System (B-SIPS) that focuses on small mobile hosts and correlates power consumption with wireless activity [BGMT07]. Moyers et al. proposed an intrusion detection system (IDS) to protect against malicious activities [MDMT10]. The proposed Multi-vector Portable Intrusion Detection System (MVP-IDS) works by monitoring electrical current changes and correlating this with malicious traffic. Other IDSs have been developed, such as the one proposed by Nash et al. that uses CPU load and disk access to estimate power consumption and detect if battery exhaustion attacks are occurring [Nas05]. In situations where devices may be homogeneous, defenses against battery exhaustion attacks can be based on comparing these devices to create a realistic baseline and find anomalies that may be effective in wireless sensors and interfaces [UKTB15]. Finally, work by Hristozov et al. using rate limiting approaches to defend against battery exhaustion attacks reported to be successful for devices supporting RESTful services [HHS19].

4.4 Communications Layer

In this section, we firstly cover components of the E-IoT Communications Layer such as interfaces and protocols. We follow up with the threat model for this layer. Moreover, we introduce E-IoT Communications Layer threats and attacks. Finally, we highlight mitigations and security mechanisms applicable to the E-IoT Communications Layer.

4.4.1 Elements of the E-IoT Communications Layer

Ethernet. Internet Protocol (IP) communication has become one of the most widely deployed standards in internal and external networks. Often, modern homes and offices already have the physical Ethernet wiring and infrastructure for Internet Protocol. As such, an E-IoT system installer can use both standards (IPv4 or IPv6) for Ethernet-based communication [FS18]. Additionally, with IP, integrators have the flexibility to divide traffic flow of connected devices with subnetting and virtual LANs (VLANs). For instance, an integrator can divide a larger network into segmented sections with subnetting, determining the maximum number of devices in each segment through network configuration [MP85]. Similarly, VLANs are used to improve information flow, security and better manage an IP network [Sup20]. For instance, Pakedge, a vendor of E-IoT-centered network solutions, encourages VLANs for E-IoT installations and network segmentation [Pak20b]. As IP is popular and widely supported by many vendors, E-IoT systems will often use IP communication in some of their components. Ethernet provides the advantage of a superior level of reliability and speed compared to the wireless counterpart. Further, Ethernet can power devices through Power-over-Ethernet (PoE) technology [Ver16]. As such, integrators only need to cable a PoE-capable connection to a device, such as a touchscreen, to provide data and power through a single connection. Physical cabling has proven to be a reliable communication method between smart components and remains popular for high-bandwidth, high-reliability applications. For instance, Ethernet may be used to control devices in the equipment rack such as IP-capable A/V receivers, Ethernet-powered IP cameras, or hardwired touchscreens [Mar14]. Moreover, Ethernet offers different networking topologies (e.g., star, ring, single-switch), which grant integrators the flexibility needed for custom E-IoT installations [MLM$^+$99].

WiFi. Wireless Fidelity (WiFi) is a frequently used communication protocol for smart devices where Ethernet cabling endpoints are not viable. Various modes within IEEE 802.11 have allowed for increased speeds and frequencies. The main advantage of wireless communication is that E-IoT devices (e.g., thermostats, controllers, A/V) may use a wireless connection without requiring an extra physical connection to integrate into an existing system. Similarly to Ethernet Category cables: 802.11 generations b, a, g, n provide different levels of data rates, as well as operate in 2.4 GHz or 5.0 GHz[Int17]. In many E-IoT systems, WiFi serves different purposes due to its widespread nature. Many smart device vendors enable wireless network connections natively on their devices, making such devices easy to integrate into E-IoT systems. Examples of WiFi usage in E-IoT systems may include interfaces (e.g., phones, touch screens, tablets) and physical devices (e.g., displays, receivers, projectors). In terms of WiFi security, a number of configurations are available for accepted WiFi security standards, such as the Wireless Equivalent Standard (WEP) which is obsolete now or WiFi Protected Access (WPA), with the latest release being WPA3 security [BSK04, AMA20a]. Furthermore, in larger and more complex network deployments, enterprise solutions exist and are usually installed by trained integrators [Li 18]. As such, a number of different configurations are possible with WiFi communication dependent on the equipment, level of security, and installation requirements of an E-IoT deployment.

Zigbee and Z-wave. Two of the most popular mesh-network protocols for smart devices are Zigbee and the proprietary Z-wave [Z-W18, Zig18]. Various vendors have embedded radio communication hardware on their thermostats to connect their devices to more extensive mesh networks. While Zigbee and Z-wave are different protocols, they are used for similar purposes in E-IoT systems. For instance, these protocols are often used in low-bandwidth applications to integrate devices such as thermostats, light dimmers, relays, and sensors to a larger system. Mesh networking allows users to retrofit existing installation by replacing existing components such as light switches for wireless-enabled components. For Zigbee, usually, there are three types of devices within the Zigbee mesh network: a coordinator, routers, and end devices [RSP11]. The Zigbee coordinator is the root of the Zigbee network and manages components necessary for Zigbee to operate (e.g., security keys, access control, security policies, stack profile). The Zigbee Router relays information and routes Zigbee packets among devices. Some Zigbee routers may also have the functions of end-devices. Finally, the end-devices send and receive communication from parent nodes and are usually designed for a specific purpose (e.g., door locks, light bulbs, sensors). Z-wave follows a similar device architecture with three basic device types, controllers, routers, and slaves. These devices fulfill similar purposes as their Zigbee counterparts [Dav20].

Bluetooth/BLE is a wireless standard for data exchange between portable and fixed devices. A short-wavelength protocol, Bluetooth operates from the 2.4 to 2.485 GHz range [Blu20a]. Additionally, Bluetooth may operate as Bluetooth Low-Energy (BLE) or Bluetooth Mesh, which allow for more varied applications to the protocol [Blu20b]. With the number of Bluetooth devices in the market, E-IoT systems are compatible with the protocol for different purposes. For instance, Savant may use Bluetooth Low Energy for their smart lighting solutions, while other systems use Bluetooth for connecting mobile devices and stream music to the central system [Sav]. Bluetooth networks, commonly known as piconets, follow a master and slave architecture where up to seven active slave devices can be connected to a master device [SP08].

IR Infrared (IR) is a wireless optical communication medium used to control devices over short, line-of-sight ranges [Tec20]. While limited, as it cannot penetrate through walls and the short transmission rate, IR remains popular in many consumer devices (e.g., A/V, televisions remotes, motorized components). As such, because of this widespread support, IR sees common use in many E-IoT systems that need to integrate these devices into centralized E-IoT systems. E-IoT systems integrate these devices using IR flashers placed on physical devices; these flashers relay messages directly to the receiving device [Sna20]. As some devices can only be controlled through IR, E-IoT makes widespread use of IR communication.

Proprietary Wireless. Not all protocols used by E-IoT systems are well-known or open-source. Proprietary wireless communication protocols are often used in E-IoT systems and have not seen much research. For instance, the Radio Technology Somfy (RTS), is used by Somfy, one of the major vendors of E-IoT motorized blinds [Som]. Similarly, popular system vendors such as Lutron, Levitron, Legrand, and Crestron also use proprietary wireless protocols that have remained mostly unexplored [Bla20, Creb, Leg19, Lev]. Table 4.1 highlights some proprietary wireless protocols used by E-IoT systems and their usage in E-IoT.

Serial-based. Serial-based communication is a precursor to several modern device communication standards. While many may consider the use of serial-based communication as deprecated, various E-IoT systems and connected devices officially support serial-based communication for system-to-device integration. Further, some E-IoT systems have built their systems on top of existing serial-based communication for proprietary devices. For instance, since the accepted inception in 1969 [fCo02], Recommended Standard 232 (RS-232) has been a well-known medium for device-to-device communication. This standard is often used in E-IoT environments for communication between devices. Some of these devices include thermostats, projectors, A/V receivers, A/V switchers, motorized lifts, displays, pool controllers, motorized drapery, and alarm systems that interface with other devices directly through serial-based links. A more specific example is the Carrier Infinity Series systems module for HVAC units. This module allows an E-IoT system to communicate with Carrier HVAC systems through serial interface or allow for remote access using a physical Ethernet connection [Car13]. In many cases, serial-based communication is wired in a "daisy-chain" bus configuration where the cabling goes from device-to-device instead of each device is individually wired to the E-IoT controller.Such a wiring configuration is a common practice in E-IoT, as daisy-chain is easier to wire and saves the integrators and users in labor and wiring costs.

The use of serial-based protocols for a variety of use-cases is widespread among E-IoT vendors. For instance, Crestron's Cresnet has become a ubiquitous name in residential, marine, and commercial installations [Cre17a]. Cresnet uses RS-485 half-duplex communication used for communication between devices (e.g., interfaces, components, keypads) and the controller [Cre17b]. Similarly, vendors such as Control4 [Con13a], LiteTouch [Lit06], and Savant [Sav14] use proprietary serial-based protocols to communicate with interfaces. These connections usually are daisy-chained together and work with multiple lines. In addition to these examples, many product vendors manufacture devices with native serial communication to where many devices and systems are integrated into E-IoT through serial communication.

For instance, shades [Som20], advanced audio receivers [Mar14], televisions [Sam20], and alarm systems [ADI20] can all be integrated into E-IoT using serial-based communication. The technical specifications of many of these highlighted serial-based communication protocols are not publicly available, and thus any security mechanisms remain largely unknown.

Another type of serial-based communication are building automation protocols such as BACnet. BACnet was designed specifically to meet the requirements for automation and control within corporate offices, buildings, and other commercial establishments. BACnet can be integrated into some E-IoT systems, with many devices available. The protocol is also used for communication in sensors, security systems, energy management, lighting control, physical access systems, and elevator controls [ASH16]. BACnet operates on top of RS-485 and RS-232 to provide application and networking layers for device operation. BACnet implements four layers: Application Layer, Network Layer, Data Link Layer, and Physical Link Layer. In this protocol, RS-232 is used for point to point communication while RS-485 handles Master/Slave Token Passing [Tex14]. Since BACnet is an open protocol, it has been adopted by various device vendors and manufacturers as a form of external control.

HDMI. The High Definition Multimedia Interface (HDMI) is one of the core components of audio/video systems. It acts as the main physical connection between multiple devices (e.g., televisions, projectors, video players, receivers). As such, HDMI is one of the most common interfaces used worldwide, with billions of compatible devices in the wild [Ven15, Wri18, Tsu08]. Per HDMI design, communication transmitted is not limited to audio and video, as HDMI transmits control and information signals through the cabling through the 19-pin connector [HDM18]. Further, HDMI can be a part of distribution networks with switchers, splitters, and other interconnects that allow multiple HDMI-enabled devices to share A/V sig-

Table 4.1: Examples of E-IoT system proprietary RF protocols.

Vendor	Protocol	Product Lines
Lutron	Clear Connect Technology RF [Bla20]	Lighting, Shades, Interfaces
Somfy	Radio Technology Somfy [Som]	Lighting, Shades, Interfaces
Levitron	LevNet RF [Lev]	Lighting, Shades, Interfaces
Legrand	TopDog RF [Leg19]	Lighting, Shades, Interfaces
Crestron	infiNET EX/ER [Creb]	HVAC, Lighting, Shades, Interfaces

nals and communicate. As A/V distribution is an important part of E-IoT, HDMI serves a major role in E-IoT systems [Con20d, Cre20d, Sav20b]. Further, some E-IoT systems use communication protocols embedded in HDMI to control and integrate devices into an E-IoT system [Cre20b]. For instance, the HDMI connection includes the Consumer Electronics Control (CEC) to expand the functionality of HDMI systems [HDM09]. The CEC protocol is a component of HDMI communication and was developed to enable interoperability between HDMI devices. CEC is a low-bandwidth protocol with a maximum of 16 devices and functions in a bus architecture. Some E-IoT systems use CEC to control A/V devices such as receivers, televisions, and projectors. Thus, many vendors implement CEC features on their devices under different trade names, including Anynet+ (Samsung), Aquos Link (Sharp), BRAVIA Link/Sync (Sony), CEC (Hitachi), CE-Link and Regza Link (Toshiba), SimpLink (LG), VIERA Link (Panasonic), EasyLink (Philips), Realink (Mitsubishi) [Goo18].

4.4.2 Threat Model for E-IoT Communications Layer

In this layer, we consider Mallory compromising an E-IoT system through the communications layer, targeting the confidentiality, availability, and integrity of the system. Thus, Mallory compromises the E-IoT system through communication components, often without the need of physical access. Attacks on this layer may benefit weak protocols, protocol vulnerabilities, flaws in implementation, and other similar factors. As such, Mallory, in this case, is knowledgeable in communication vulnerabilities and has the equipment necessary to compromise E-IoT. For instance, Mallory may carry sniffers and the software necessary to eavesdrop on communication channels and inject messages into E-IoT communication. We explain Mallory's possible roles in attacking E-IoT communication layer as follows:

Visitors and Unprivileged Users. Some users (e.g., visitors, insiders) may not have sufficient privileges to interact with all of the components of a deployed E-IoT system. Mallory, as a malicious unprivileged user, may use protocol vulnerabilities to gain unauthorized access to devices near her. As such, attacks on serial-based protocols, short-range wireless, and HDMI are feasible. An unprivileged user may just need some preliminary knowledge of the protocols used.

IoT Hackers. Malicious actors such as hackers may target E-IoT systems specifically in public locations (e.g., presentation rooms, bars, campuses). In this scenario, Mallory as a malicious hacker, may choose to perform reconnaissance of an E-IoT deployment without direct physical access to the system. Additionally, more sophisticated attackers may attempt to compromise a system, gain unauthorized access, cause DoS attacks, or otherwise disrupt E-IoT operations through the communications layer. In this case, Mallory only has unauthorized access to all E-IoT system components.

4.4.3 Communication Layer: Attacks and Vulnerabilities

In this subsection, we give an overview of the attacks and vulnerabilities of the E-IoT Communications Layer. Specifically, we cover attacks to serial-based protocols, wireless protocols, HDMI-based protocols, and building automation protocols.

Serial-based Protocol Attacks. One of the challenges of properly evaluating serial-based protocols in E-IoT is the proprietary nature of many of these protocols. Many proprietary protocols are long-lived and do not advertise any form of security mechanism to the communication. Information on many of these protocols (e.g., Cresnet) is sparse. These protocols rely largely on security through obscurity as many of these protocols were designed for functionality but not security in mind. Even with this lack of research, online communities and integrators have explored E-IoT protocols and managed to create sniffers to capture serial-based communication for debugging [Ste15]. As such, these sniffers work without any form of authorization beyond physical access and expose possible threats to E-IoT serial-based protocols. In relation to industrial control protocols, a comprehensive review of the security channels for industrial control protocols can be found in a study by Volkova et al. [VNBd19]. This study highlighted that aging serial-based communication technologies such as Modbus can be attacked (e.g., credential theft attacks, replay attacks, Man-in-the-middle attacks) by a knowledgeable attacker.

An analysis in 2003 by the Department of Commerce found some threats to building automation protocols such as BACnet. While most systems were not connected to the Internet, there was still backdoor access via modem connections to controllers [Hol03]. The study also noted various attacks on passwords, confidentiality, integrity, DoS, spoofing, and eavesdropping within a BACnet installation. Gasser et al. discussed research on Internet-exposed BACnet systems [GSD$^+$17]. BACnet is often an integral part of connected Industrial Control Systems (ICS); these are critical infrastructural systems for any size business and offices [MMA$^+$16]. BACnet operates on UDP ports 47808-47823 by default [ASH16]. Researchers used a pre-rendered BACnet payload in conjunction with Zmap [DWH13] to scan for devices in the IPv4 address space for valid responses. Using this methodology, researchers managed to confirm a total of 15,429 exposed BACnet devices on the Internet. A notable characteristic of BACnet/IP UDP protocol is that it is both stateless and does not require handshake nor authentication. The previously mentioned characteristics of BACnet make it susceptible to Amplification Attacks, a DoS attack where a response payload is larger than the request payload [GSD+17].

WiFi Attacks. WiFi communication has been an active topic of research due to its broad appeal and uses in many connected devices. Many WiFi attacks have been covered in different publications, surveys, and technical documents. Additionally, attacks may be dependent on installed hardware, firmware, security used (e.g., WEP, WPA, WPA2, WPA3), and specific implementation. A survey by Lashkari et al. highlighted weaknesses to security mechanisms in WiFi communication [LDS09b]. Specifically, this work notes that WEP is susceptible to attacks (e.g., packet forgery, replay attacks, de-authentication) and vulnerabilities such as improper key-management and problems with the RC-4 algorithm. Other work from Borisov et al. goes further into the insecurities of the WEP protocol and how poor security practices (e.g., keystream reuse, key management) allows an attacker to compromise WiFi with WEP security [BGW01]. Specifically, WEP is vulnerable to eavesdropping attacks, message modification, message injection, message decryption, authentication spoofing, and reaction attacks against WEP. While considered more secure, WPA vulnerabilities also exist. Lashkari et al. note that WPA/WPA2 has definite security improvements over WEP, such as the use of the Advanced Encryption Standard (AES) and the Temporal Key Integrity Protocol (TKIP) [LDS09b].

However, even with improvements, WPA and WPA2 can be susceptible to attacks (e.g., brute force attacks, dictionary attacks). A related attack for WPA/WPA2 is a handshake capture attack. An attacker can capture the communication handshake and attempt to perform brute force attacks or dictionary attacks against the captured handshake [KKAA14]. An attack proposed by Vanhoef et al. introduces key re-installation attacks against WPA/WPA2 where attackers can force a WiFi network to reuse old keys and compromise confidentiality in the network [VP17]. As such, key re-installation attacks would allow Mallory to perform actions such as packet replay, decryption, and forging in some implementations, severely impacting the confidentiality and integrity of WiFi communications. Other attacks such as the Reaver and Pixie-Dust attacks also target WPA-based security, specifically exploiting the WiFi Protected Setup (WPS) protocol in routers [Kod18]. Finally, as a newer security mechanism, some weaknesses have been found in WPA3 [KH18]. As such, DoS attacks [LZ19a], connection deprivation attacks [LZ19b], and handshake attacks [VR20] can compromise WiFi communication with WPA3 security. As many E-IoT devices use WiFi communication, any WiFi attacks could compromise the confidentiality, integrity, and availability of E-IoT and E-IoT-integrated components.

Zigbee and Z-Wave Attacks. Wireless technologies are common in E-IoT systems in many different use cases and have been an active topic of research in the security community. As described in [KHS17], various communication protocols (e.g., Zigbee, Z-Wave) can be attacked, negatively impacting E-IoT systems by directly affecting user interfaces. There have been known security breaches in Zigbee devices. The Zigbee Light Link (ZLL) standard was designed with easy client integration, and installation in mind [Wan13]. One known breach in 2015 involved the leakage of the master key for light-based Zigbee devices. This leak made ZLL devices insecure [ZS15]. It must be noted that there are variations between Zigbee systems, software, hardware, and chipsets; not all attacks may be effective on all Zigbee systems even if the Zigbee stack is an accepted standard.

Energy depletion attacks such as Ghost-in-Zigbee [CSC+16] may prove to be effective against battery-powered E-IoT components. In addition to depleting Zigbee devices' power, it can facilitate threats such as DoS and replay attacks on a Zigbee network. The attack method involves sending false messages to nodes within a Zigbee network to trigger processor-intensive computations (e.g., cryptographic operations). These unnecessary computations cause power and performance losses on the affected device. Ghost-in-Zigbee attacks also demonstrate three unique types of DoS attacks. First is a computational load attack, which can be done by sending numerous messages at the same time to trigger the depletion of a node's energy. However, such an attack could be easily detected with abnormality detection. The second type of DoS is referred to as MAC misbehavior, which takes advantage of Zigbee channel sensing. When a targeted node receives continuous traffic, all nodes within that region will not communicate through that node. The third is a replay attack in which a malicious attacker may use frame counters greater than valid values in their message. Since Zigbee keeps an Access Control List (ACL) table, this table will be updated to match the malicious counter values. Any legitimate node trying to make contact after the alteration will be rejected due to their frame counter values being less than the altered values, leading to a malicious spoofing attack. The article [KHS17] mentions a third attack on Zigbee spanning from hardware implementations. Going further in-depth, in [RSWO18], researchers attacked an implementation of an Atmel chip used with Phillips Hue bulbs and Zigbee Light Link (ZLL) mode. In this attack, the researchers created a custom circuit board to target the igbee chipset used with smart bulbs and created a worm to spread the infection among light nodes.

In [OHA+14] three types of attacks on igbee were demonstrated using the KillerBee toolkit [Riv17]. The first attack takes advantage of Zigbee's discovery process and mimicked a legitimate device to gather information about other devices within the Zigbee network. This information spans various channels and will yield responses from Zigbee nodes within a channel. The second attack is the interception of packages. This attack functions on the basis that some Zigbee networks use weak or no encryption. As such, an attacker can eavesdrop on communication using the toolset and a USB adapter to capture traffic on a given channel. As the third proposed attack, if the previous two attacks are successful, an attacker can intercept and record Zigbee traffic. As such, an attacker can replay previously recorded packets and have Zigbee devices accept sent messages. Z-Wave vulnerabilities may depend on implementation practices, firmware, and hardware. Using reverse engineering methods, Fouladi et al. in [FG13] provided some examples of available exploits that could compromise entire devices. The attack used Z-force, a packet interception, and injection tool, to reset the established network key and take advantage of the protocol's steps. The researchers describe the issue as a lack of 'state validation' in some Z-Wave devices. An attacker can use packet injection to force Z-Wave devices to overwrite their current shared network key with an attacker-specified key. They demonstrated a successful attack on a connected door lock. While follow-up publications note that some of the attacks described have been patched, devices that have not been updated and usage of older firmware may be vulnerable to these attacks [KHS17].

The research by Fuller et al. explored vulnerabilities of rogue controllers within Z-Wave established networks ranges [FR15]. This work introduced an attack that used a malicious Z-Wave controller to attack unsecured devices. To begin, the authors established a Home Automation Network (HAN) using Z-Wave devices such as connected door locks, smart lights, and connected water valves. The attacker must first gain access to the local WLAN network to perform this attack, assuming the network is improperly secured. Once access to the network has been granted, an attacker can scan the network and retrieve the address of the Z-Wave gateway and any other gateways. The researchers then took advantage of known gateway vulnerabilities and, in this case, attacked a VeraEdge Z-Wave controller. Further, they retrieved and saved a backup file for the entire system. With this information, the researchers could then duplicate a legitimate Z-Wave controller with a malicious one in the same network. This rogue controller could then communicate to Z-Wave devices within that network, compromising all of the available devices. The study also noted that with this backup file, there is the possibility that sensitive information and activity can be retrieved. Further, log files could also prove valuable to an attacker gathering information in usage or future attacks.

Bluetooth/BLE Attacks. Other popular short-range wireless solutions are Bluetooth and BLE. As mentioned earlier, Bluetooth is used by some E-IoT systems during standard operation and device configuration. Due to mobile devices, IoT, and other common use-cases of Bluetooth, attacks on Bluetooth have been widely documented, with a number of surveys written on the topic of Bluetooth security [SW05, DG17, MT12, Dun10, JSSN18]. Relevant to E-IoT, attacks highlighted in these surveys include man-in-the-middle attacks that can occur by compromising Bluetooth's Secure Simple Pairing (SSP) to impersonate trusted parties [HH07, SMS18, HT10, HT08, HH08, BWM12]. Further, another attack relevant to E-IoT is Bluesniping, which uses specialized antennas to sniff Bluetooth communication beyond the expected Bluetooth range [Her04]. Bluesniffing attacks may also be a concern, as attackers may be able to infer E-IoT activity from sniffing packets coming from Bluetooth-based interfaces and devices [SB07]. Disruption attacks such as Bluechopping, Bluecutting, and Bluedepriving may also affect the availability of E-IoT devices as these attacks all work to disrupt Bluetooth communication through different approaches [LZ20]. For instance, for bluechopping, an attacker spoofs the identity of a connected Bluetooth device to cause a DoS condition. Bluecutting, an attack that disrupts Bluetooth communication by spoofing a Bluetooth device and requesting a target device begins re-pairing. As attacker then discards the stored link key and pairing cannot be performed [LZ18]. Finally, bluedepriving interrupts Bluetooth communication by causing a conflict between a spoofed device and a legitimate device so that this legitimate device cannot pair through Bluetooth connection [AK18]. It must be noted that similar to other protocols, many Bluetooth attacks are dependent on implementations, software versions, and use cases of Bluetooth devices. The differences between Bluetooth and BLE have led to some unique attacks for BLE. For instance, a work by Lounis et al. demonstrated that the "Just Works" pairing mode for BLE can be exploited to perform interception, interruption, fabrication, and modification attacks on BLE devices. Another notable example by Wu et al. is BLE Spoofing Attacks (BLESA)[WNK$^+$20b]. In this work, the authors demonstrated that with BLESA, an attacker can impersonate a BLE devices and provide spoofed data to previously-paired devices. In another work, Zhang et al. showed that BLE is susceptible to a downgrading attack with Android-based systems, where devices can be forced to run in an insecure communication mode without a user's knowledge [ZWD$^+$20].

IR Attacks. IR communication is used in E-IoT in the form of IR flashers to control integrated devices (e.g., displays, projectors, blinds). As such, most of these systems use simple, line-of-sight receivers without any form of authentication from the remote. Many of the controlling codes are available from online sources in websites such as remote central [Rem20]. As such, it is trivial for an attacker to capture or emit IR commands through line-of-sight [Adm11]. A malicious attacker could simply use an IR blaster to control IR-enabled devices and disrupt the operation of E-IoT systems [Ind08]. In other cases, attackers may be able to reconfigure IR-enabled devices as if they had the original device remote. In terms of E-IoT, if a device is reconfigured or reset, an E-IoT system may not be able to communicate with these devices.

General Wireless Attacks. In this category, we cover any attacks that can apply to wireless in the Industrial, Scientific, and Medical (ISM) frequency bands and are not unique to any communication protocol. Jamming attack can negatively impact E-IoT system communication in multiple modes of communication and falls under a specialized DoS attack. Specifically, jamming presents a major threat to wireless networks and any E-IoT device that uses wireless networks (e.g., interfaces, sensors, relays), causing the devices to fall offline. Several works and surveys have covered jamming attacks against wireless communication that can be relevant to E-IoT systems [BPP13a, MRR18, MGKP09, GLY14]. Specifically, these surveys highlight several proven jamming techniques against wireless networks (e.g., spot jamming, sweep jamming, barrage jamming, deceptive jamming). Further, jamming attacks are often cheap, easy to perform, and difficult to mitigate. The capabilities of more elaborate jamming attacks such as reactive jamming are covered by Wilhelm et al., highlighting the dangers of reactive jamming in wireless networks, where jamming techniques can target specific packets in wireless communication [WMSL11]. While reactive jamming may have limitations due to cost, demonstrations of jamming attacks show that an attacker can target specific wireless communication (e.g., Zigbee) with some technical knowledge and widely-available low-cost devices [Bas19].

HDMI-Based Protocol Attacks. HDMI is one of the core connections of video distribution and contains various protocols that can pose a threat to E-IoT systems. In HDMI-Walk, Puche et al. demonstrated that the CEC protocol can be used to gain arbitrary control of CEC-supported device functions [RBAU19a]. Specifically, the authors demonstrated that CEC can be used with HDMI distributions to attack multiple HDMI devices. The HDMI-Walk attacks further showed that an attacker might control devices, transfer information, cause DoS conditions, eavesdrop, and otherwise harm HDMI networks through a single point of connection or compromised device. For all of the attacks, the researchers inserted a device into an HDMI-capable distribution. The first attack used the inserted device to gather information about all of the connected HDMI devices, returning details such as the language, model number, power state, and running version. Two more attacks proved that eavesdropping and facilitation of existing attacks are possible with CEC. The authors showed that CEC could be used for unauthorized data transfer by transferring audio information and WPA handshakes from one end of the distribution to another rogue device. Finally, there were two DoS attacks demonstrated in HDMI-Walk. On the first attack, the attacker device was configured to identify televisions powering on through CEC broadcast and shutting the displays down before they initiated. The second DoS attack abused television input change and overwhelmed displays through CEC, causing them to become inoperable. Further, the authors of HDMI-Walk noted that CEC propagation is not obvious and difficult to mitigate, creating networks without the user's awareness. Other relevant work on HDMI sub-protocols was published by the NCC group identified on CEC-based fuzzing vulnerabilities through CEC, and other viable threats through HDMI [Dav13]. Specifically, the NCC Group identified that HDMI's HEC channel could be used for corporate boundary breach, endpoint protection circumvention, and unauthorized network extension. Similar work presented by Smith et al. contributed to further CEC-based fuzzing with the development of the tool CECSTeR, used to execute CEC-based fuzzing attacks on CEC-supported devices [Smi15, Dav12].

4.4.4 Mitigation of Communication Layer Attacks.

Serial-based Communication Defenses. While not specific to E-IoT, research in serial-based communication defense mechanisms may apply to E-IoT. Studies by Dudak et al. [DGS+19], and Wilson et al. [Wil18] provide insight into securing serial-based protocols and considerations that must be taken to design protocols securely. Further, as standardization may help secure serial-based communication in ICS, the IEEE 1711.2 working group's efforts have focused on creating the Secure SCADA communications protocol [IEE20]. A similar approach has not been taken for proprietary E-IoT communication protocols yet, but could guarantee interoperability and secure protocol design in the future. In a survey by Volkova et al. highlighting attacks and defenses [VNBd19], the authors noted that network security, best practices, and software updates may help mitigate threats to Modbus and similar serial-based protocols. However, the authors noted that even with existing mitigation strategies, there are vulnerabilities that have to be mitigated by the protocol specifications. Finally, many proprietary protocols may require physical access to compromise, so controlling physical access may be a viable mitigation strategy.

For building automation protocols, vulnerabilities are often dependent on the implementation and installation. ASHRAE, the compendium behind BACnet, has released a security architecture to its initial construction for the deployment of a security layer for BACnet networks [ASH17]. In the addendum, ASHRAE acknowledges the need to update the 56-bit DES cryptographic standard used for communication since 2004 to AES-128 bits. As several threats have been found in DES encryption, protocol updates are needed. Further, the BACnet specification explicitly notes that BACnet security encryption is optional and dependent on an integrator to be deployed [NBGT19]. If the integrator does not choose to deploy the BACnet security encryption, then the BACnet deployment will be insecure. As such, an improperly configured system may be insecure without the user's knowledge. To keep E-IoT systems and related components secure, integrators should configure systems to use available encryption. Further, entities that create and maintain communication standards must update their protocols to newer cryptographic standards.

WiFi Defenses. In a similar manner to many technologies, one of the best solutions to defend against WiFi and other wireless vulnerabilities is ensuring that the most secure protocol implementations are in place in E-IoT devices. For instance, attacks and vulnerabilities such as Reaver have been patched in many modern routers [Kod18]. Literature also references other solutions such as experimental defense mechanisms (e.g., custom key generation practices, modified WiFi standards); however, as most vendors and integrators cannot realistically implement these mechanisms, they are outside of the scope of this chapter [WJZ10]. In a similar manner to the individual network configuration of devices, integrators should follow accepted best practices when configuring WiFi security, such as the ones suggested by the United States Federal Trade Commission [Fed15]. For instance, access to a network should be limited, and routers should be secured with strong passwords, custom SSID names, with management features. Strong passwords practices can help mitigate handshake cracking and brute force attacks on WiFi. Further, using WEP is considered insecure and outdated, and as such, it should be avoided unless completely necessary. Other best practices were also highlighted by the Cybersecurity & Infrastructure Security Agency (CISA) [Cyb20]. Some defenses proposed include installing firewalls, maintaining anti-virus software, frequent networking equipment updates, and following wireless configuration recommendations from manufacturers. Several attacks can be prevented through best practices and proper configuration in WPA/WPA2 devices. For instance, disabling features such as WPS in routers may be a good practice to prevent threats such as the Reaver and Pixie-dust attacks [SNN13]. Surveys conducted on WiFi security also suggest that if it is possible, users should update their systems to the latest WPA3 security standard, however acknowledge that this is not ideal in all cases [LZ20, LDS09b]. Further, these surveys note that proper configuration of WPA3 can prevent key cracking attacks.

Zigbee/Z-Wave Defenses. One of the best solutions to Zigbee and Z-Wave protocol vulnerabilities is verifying that vendors use the latest and the most secure protocol implementations. Further, E-IoT integrators should follow the best practices offered by E-IoT vendors and manufacturers. A survey by Lounis et al. highlighted how updated protocols have resolved many attacks for short-range wireless protocols [LZ20]. This chapter also highlights that network administrators (integrators - in E-IoT systems) should monitor and verify that devices are properly configured and updated. However, as users and integrators of E-IoT systems rely on E-IoT device manufacturers and vendors, solutions for vulnerabilities will come from vendor updates and best practices. For instance, manufacturers of E-IoT controllers must make sure that short-range wireless nonces are not reused to prevent key generation attacks [WAM14]. Additionally, a work by Benzaid et al. highlighted that polling messages and responses should also be authenticated to prevent spoofing attacks on short-range wireless networks [BLAN$^+$16]. An article published in 2006 on Z-Wave security highlighted the main differences between Zigbee and Z-Wave security [Kni06]. The article noted that Z-Wave protocol encryption is optional and for that reason, encryption should always be enabled as a security measure. The study also noted that older Z-Wave systems are open to various attacks, especially if encryption has not been enabled. As such, maintaining systems properly updated and securely configured should be a priority for E-IoT communication.

Bluetooth/BLE Communication Defenses. In a similar manner to other wireless defenses, one of the best solutions for Bluetooth attacks are updates and making sure that best practices are followed in Bluetooth configuration. A set of Bluetooth-specific best practices have been proposed in [Lyn07]. For instance, disabling Bluetooth functions when they are not in use, disabling device ID broadcast, strong passwords, and verifying incoming transmissions have been suggested to mitigate Bluetooth-based threats. Further, a survey by Lounis et al. [LZ20] notes that Bluetooth software updates are necessary to defend against well-known Bluetooth attacks (e.g., bluechopping, bluecutting, bluedepriving). Similarly, some defense mechanisms have been proposed for BLE-specific threats. For instance, to address Android-based downgrading attacks against BLE, Zhang et al. proposed some modifications to the Secure Connection Only (SCO) configuration for BLE [ZWD$^+$20]. Another example by Wu et al. is BlueShield, a monitoring system designed to detect spoofing attacks on BLE communication [WNK$^+$20a].

IR Communication Defenses. As IR communication is line-of-sight, physical security may be one of the best defense approaches. With very little literature on IR communication and defenses, it may be an idea for integrators to cover IR receivers when not in use to prevent attackers from tampering with devices. Further, access control may prevent unauthorized users from disrupting the operation of E-IoT-controlled devices using IR emitters. As IR requires line-of-sight, it may be easy to discern when an attacker is meddling with a device. Further, as CCTV can display the IR spectrum, it may be possible to use cameras to identify an attacker using IR to communicate with devices [Car20].

General Wireless Communication Defenses. Securing wireless communication from jamming attacks has been a topic of research with a number of different approaches suggested. Numerous surveys are available on wireless jamming defenses and counter-measures [BPP13a, MRR18, MGKP09, GLY14, LKP07, BPP13b, MS12, OAH18]. As such, a solution for E-IoT deployments will depend on the wireless technology used and the particular wireless use cases. A survey on this topic by Aristides et al. divides anti-jamming approaches into three different types: proactive, reactive, and mobile agent-based solutions [MGKP09]. Proactive counter-measures in the background cannot be initiated, stopped, or resumed on demand and require prolonged implementation time and high implementation cost. An example of proactive measures are software and hardware-based solutions that detect jamming attacks before they occur (e.g., DEEJAM) [WSZ07]. Reactive anti-jamming approaches reduce computation energy costs compared to proactive counter-measures. Reactive jamming defenses rely on active jamming attacks and aim to mitigate attacks (e.g., JAM) [WSS03]. However, in the case of some jamming attacks, reactive-anti-jamming may have some detection delays. Finally, mobile agent-based solutions employ anti-jamming agents that move between hosts to detect jamming attempts. For different protocols, there exist different jamming approaches, subject to surveys of their own. Vendors of E-IoT should consider the best defenses for their supported devices and implement them in their systems.

HDMI-based Communication Defenses. While HDMI sub-protocols are usually secured through restrictions to physical access; other options have been explored. For instance, in work proposed by Puche et al., the authors created a passive intrusion detection system framework designed to protect against HDMI-based threats [PBAU20]. The framework uses features in CEC communication to build a machine learning classifier and does not require modification to the original protocol, as a modification to the protocol is problematic, with billions of HDMI devices distributed worldwide. Physical defenses against these attacks involve the use of CEC-less adapters, which can prevent CEC signal from propagating over large distributions [Ama20b]. As such, an integrator may use a CEC-less adapter to prevent public, easily-reachable HDMI endpoints from receiving CEC communication.

4.5 Monitoring and Applications Layer

In this section we highlight the monitoring and application layer of E-IoT systems. First, we cover elements of the E-IoT monitoring and applications layer then introduce the threat model at this layer. Next, we cover monitoring and application-layer threats and attacks. Finally, we provide relevant defenses and mitigation mechanisms.

4.5.1 Elements of the Monitoring and Applications Layer

E-IoT Monitoring and Applications Layer consists of E-IoT drivers, E-IoT software and services, and E-IoT configuration.

E-IoT Drivers. As introduced in Chapter 2.1.6, drivers are an important part for E-IoT system functionality. As a software-based component of E-IoT systems, drivers provide all the information necessary for an E-IoT system to integrate a device or web service into the system. As such, E-IoT drivers are not standardized from system-to-system and may be known under a different name (e.g., Crestron modules, Control4 Drivers) [RBA$^+$20]. Drivers are inserted and configured in an E-IoT system during programming or maintenance by integrators. Thus, drivers can be obtained in three different ways. (1) Drivers may be acquired directly from the E-IoT configuration software. (2) Drivers may be acquired from a catalog of drivers provided by the main E-IoT vendor. (3) Drivers can be downloaded from third-party sites (e.g., forums, device vendors, third-party developers). However, while many vendors will validate drivers acquired from their software or repositories, drivers from third-party sources are often not checked for malicious content. Additionally, some drivers are not free which may tempt integrators to use a free, unverified driver with malicious code available online [Bla08].

E-IoT Software Services. E-IoT systems use several software services for configuration and maintenance. Beyond proprietary tools used by E-IoT vendors, such as Control4's composer and Crestron's Simpl, E-IoT uses common application services [Con10a, Cre20e]. Available software services may vary from system to system. While E-IoT systems may have well-known, documented software services such as File Transfer Protocol (FTP), Secure Shell (SSH), and Telnet communication, E-IoT solutions may also run unknown proprietary services. With the closed-source nature of many E-IoT systems, documentation and details of these proprietary services remain mostly unavailable. As such, operating manuals available online and troubleshooting guides are among the few sources of information on these services. In contrast, well-known and commonly-used services are easier to identify. For instance, file transfer is necessary for E-IoT tasks such as firmware upgrades, image uploads, and vendor software configuration. As such, one of the accepted file-transfer services is FTP, and for more secure communication, Secure FTP (SFTP) [Jon20, Edu20]. Another requirement for E-IoT is diagnostics and configuration; thus, integrators need to communicate directly to the E-IoT system. Secure shell services may be used for diagnostics and configuration as integrators use secure shell clients such as PuTTy to connect to, diagnose, and configure E-IoT system and system components through services such as Telnet or SSH [Nit13, Sim20]. Another use-case of software services is webservers and web interfaces using HTTP or HTTPS. E-IoT systems may host webservers and web interfaces to allow integrators to configure, diagnose, or monitor devices. For instance, CCTV systems host a web interface to configure cameras, view recordings, view a live feed, and manage CCTV systems

[Mar20]. Finally, software suites such as Busybox are common in IoT and E-IoT alike, as BusyBox provides many common UNIX utilities in a compact executable with size optimization and a modular design [Eru20, Wel00]. Due to the convenient design, E-IoT vendors such as Control4 run BusyBox on their main controllers and devices [Joh14].

E-IoT Configuration. Beyond software, configuration of E-IoT systems can impact the overall security of the system. Some E-IoT users may need to access E-IoT system features remotely. Additionally, remote access aids integrators, as it allows them to provide remote technical support, especially in moving installations such as yachts. As such, E-IoT vendors and integrators permit remote access through a variety of different methods. While the configuration is different for each system, most E-IoT systems are accessed remotely through subscription services, virtual private networks (VPNs), or port forwarding. First, some vendors offer subscription services, creating a secure and easy way for clients and integrators to connect remotely to an E-IoT system (e.g., Control4's 4Sight) [Con20a]. VPNs are another popular solutions recommended by many vendors, granting users remote access to the E-IoT network and equipment. For this reason, vendors will recommend routers with VPN functionality to integrators. Finally, as E-IoT devices (e.g., controllers, CCTV NVRs) often use ports for control and configuration, integrators often port forward these devices to allow remote access [Ver20, Mar20, Cre20g].

4.5.2 Threat Model for E-IoT Monitoring and Applications Layer

For Monitoring and Application Layer threats, we consider Mallory, an attacker knowledgeable on configuration and software vulnerabilities of E-IoT systems. As such, an attacker on this layer compromises E-IoT functionality and may gain access to unauthorized resources without any physical contact with E-IoT systems. For this layer, an attacker needs technical knowledge of E-IoT systems and software-based attacks. Mallory can be in the roles of malicious users, integrators, or remote attackers.

Users. Mallory, in the role of a frequent or visiting malicious user, could attack E-IoT systems through the Monitoring and Applications Layer. As malicious actors, these users may attempt privilege escalation, modify E-IoT systems, or otherwise try to cause unintended operation. As regular users are meant to operate E-IoT systems and not alter any configuration, vulnerabilities may allow Mallory to compromise E-IoT system components as an unprivileged user. Further, in an improperly configured network, if Mallory has network access and proprietary configuration software, she may modify E-IoT software, remote access configuration, or compromise an E-IoT system through software (e.g., malicious drivers).

Integrators. Integrators often will have full access to E-IoT systems. As a malicious integrator, Mallory may become the attacker in certain situations (e.g., bribed or disgruntled employees [And20b]). In this scenario, Mallory already has the proprietary tools and access to one or many E-IoT systems through maintenance software. Mallory could inadvertently compromise multiple systems using malicious drivers or remote access tools. Further, Mallory may target wealthy or famous clients and eavesdrop on information for personal gain or otherwise disrupt E-IoT system operation.

Remote Attackers. Mallory may be a remote attacker seeking systems to compromise. She may find E-IoT systems exposed to the Internet. If Mallory is a more capable attacker, she may use configuration tools and manuals used by E-IoT vendors to gain complete access to E-IoT systems, install malicious E-IoT drivers, and otherwise compromise exposed systems.

4.5.3 Monitoring and Applications Layer Attacks and Vulnerabilities

In this subsection, we cover Monitoring and Application Layer attacks and vulnerabilities.

E-IoT Driver Attacks. E-IoT drivers contain all the programming necessary for E-IoT systems to integrate third-party components such as devices, APIs, and web services [Con10a, Con]. Research on the topic of E-IoT drivers by Puche et al. demonstrated that drivers can be used to compromise E-IoT systems [RBA^{+}20]. Specifically, the authors performed a DoS attack, maliciously expended system resources, and assumed control of the E-IoT controller's networking functions through malicious E-IoT drivers. The authors note that an integrator may inadvertently compromise an E-IoT system by downloading unverified drivers from third-party vendors, forums, or any external site. Since there is no verification mechanism for drivers in E-IoT controllers, an attacker can gain the control of the E-IoT system.

E-IoT Software Service Attacks. E-IoT systems will run a combination of proprietary services and well-known services in their devices. As such, some vulnerabilities have been exposed by researchers on E-IoT systems. For instance, in Defcon 26 (2017), Lawshae et al. presented several Crestron controller vulnerabilities [Ric17]. Specifically, Crestron controllers could be compromised through the CTP console, a Telnet-like interface for Crestron E-IoT systems used to configure and diagnose Crestron devices. This interface also allowed Lawshae to have direct chip communication, browser remote control, UI interaction, and microphone recording capabilities.

Further, as of the time of this writing, CVE Details show over twenty vulnerabilities for Crestron devices and six for Savant systems [CVE20a, CVE20b]. For Savant and Crestron systems, these vulnerabilities include de-authentication code overflow, authentication bypass, remote code execution, directory traversal, cross-site request forgery, and DoS. Vulnerabilities have also been discovered in presentation devices and systems. For instance, Crestron presentation devices, Barco wePresent, and Extron ShareLink presentation systems have had numerous vulnerabilities discovered (e.g., stack overflows, unauthenticated command injection) as they all share underlying code [Lin19]. Vulnerability research by Synack, a company that specializes in security research, tested the now discontinued SR-250 Control4 controllers and found several unpatched vulnerabilities described as unauthenticated management vulnerabilities [Pau15a]. Moreover, improper implementation of encryption could threaten the confidentiality and integrity of E-IoT data. Practices such as 'rolling your own encryption' (e.g., implementing self-made cryptographic functions and algorithms) have left products from companies (e.g., Dualcom, Telegram) vulnerable to attackers [Sus19]. For instance, Dualcom alarm signaling products were demonstrated to be vulnerable and susceptible to cracking attacks due to improper use of encryption mechanisms [And15]. As such, improperly implemented encryption can open up E-IoT components to a great number of attacks (e.g., malicious sniffing, brute-force, man-in-the-middle, replay).

E-IoT Configuration Attacks. One of the most notable examples of a failure in IoT security was made abundantly clear with the Mirai botnet, which overwhelmed high profile targets through DDoS attacks. The malware hijacked exposed IoT devices and used them to create a botnet. How the Mirai malware grew to a peak of six-hundred thousand infections so quickly is one of the reasons why users should be wary of the security of their connected E-IoT systems and other Internet-facing devices[ABB$^+$17]. Research on this botnet revealed issues with the current state of exposed IoT and E-IoT devices. Mirai created a bank of targeted devices with 46 unique passwords. Most of these passwords targeted exposed systems such as security cameras, CCTV video recorders, routers, and printers. Initially, Mirai used this bank of default passwords to brute force through Telnet and SSH authentication. Future iterations of Mirai altered themselves to attack through known exploits in targeted systems. Attacks such as these could also take advantage of known backdoors, such as those seen in Dahua DVRs and IP cameras, where a firmware had to be released for all installed devices due to the found vulnerabilities [Chr17]. An attacker could compromise an E-IoT system through port forwarded devices. As of the writing of this chapter, a search in Shodan.io, a search engine for exposed devices connected to the Internet reveals over 30,000 E-IoT devices exposed online from major vendors (Control4, Crestron, Savant, and Lutron) [Sho20].

4.5.4 Mitigation of Monitoring and Applications Layer Attacks

Driver Defenses. E-IoT vendors will often provide a number of drivers or validate drivers developed by third-parties. As such, integrators should try to use validated drivers to prevent driver-based threats. The work that presented driver-based attacks highlighted that vendors should approach drivers in a similar manner to untrusted software [RBA$^+$20]. Further, without standardization, drivers are implemented differently in each system; thus, security mechanisms that are viable for one E-IoT system may not be viable for others. A proposed solution is a permission system for drivers, based on the function and what a driver should be allowed to do (e.g., a serial-based controlled device should not have a driver with network connectivity) [RBA+20]. Finally, many users and integrators may not be aware that malicious code could exist in drivers and thus, awareness of this possible threat is one of the best and only current defenses.

E-IoT Software and Services Defenses. In a similar manner to any smart system; vendors, users, and integrators should follow patching and firmware best practices. As E-IoT vendors will note and often patch vulnerabilities with later releases, integrators should install the latest software and firmware versions. Moreover, users should schedule frequent product updates [Jac19]. Following frequent updates and patching in E-IoT systems can help mitigate known service vulnerabilities from software services. Further, overall awareness on running services and versions can help integrators gauge the risk of exposing E-IoT components to a network. It may be possible to anticipate unpatched vulnerabilities and prevent an attack before it occurs. As such, integrators may want to isolate E-IoT systems from other networked systems (e.g., guest-accessible networks) and enable proper network-based mechanisms to prevent unauthorized access. Additionally, legacy and discontinued equipment that cannot be upgraded or updated presents a major threat to many smart systems beyond E-IoT, especially Internet-facing systems. Integrators need to be aware of legacy equipment and make sure that their clients are aware of the risks of legacy equipment. Finally, E-IoT developers should avoid mistakes during development such as improper encryption mechanisms by using the latest libraries, avoiding custom encryption, and following verification processes [Sus19].

E-IoT Configuration Defenses. E-IoT vendors will often release security best practices, and integrators should follow these best practices for configuring E-IoT systems [Cre20g]. Moreover, installers and users should avoid weak and insecure passwords as Internet-facing devices with weak password practices have allowed attackers to compromise devices in previous large-scale automated attacks [ABB+17].

A whitepaper published by Synack provided an outline relevant to E-IoT, and professionally-installed systems [Syn15]. Proposed best practices from this guide highlighted that vendors and manufacturers should not rely on users for security. Basic password strength requirements should be enforced, as compromising a remote access account could give an attacker access to an E-IoT system. Users should also receive notifications when device statuses change or when sessions are initiated. Finally, the whitepaper notes that vendors should avoid SSL pinning, self-signed certificates, and custom encryption. One other source of vulnerabilities is port-forwarding, which exposes devices to the Internet. E-IoT vendors have always advised dealers and users not to port forward devices as some devices were not designed to be exposed directly to the Internet [Pau15b]. Instead, integrators and users should opt for VPN configuration or vendor remote access services.

4.6 Business Layer

In this section, we highlight the E-IoT Business Layer and common security concerns. Specifically, we first address common Business Layer components of E-IoT systems. Second, we highlight possible threats and attacks. Finally, we cover possible defense and mitigation mechanisms.

4.6.1 Elements of the E-IoT Business Layer

In this subsection, we highlight common elements of the E-IoT Business Layer.

Security Cloud Services. E-IoT CCTV systems usually record camera footage in local hard disk storage in a Digital Video Recorder (DVR) or a Network Video Recorder (NVR) for analog or IP cameras respectively [Swa18]. However, if a DVR or NVR is damaged or stolen in traditional systems, all video recordings are lost.

Moreover, CCTV systems have limited storage space and will often overwrite old recordings with new footage once storage runs out. As a solution to this issue, vendors offer online cloud storage solutions designed specifically for security cameras and CCTV. In addition to cloud storage, security cloud services allow integrators and users to manage multiple E-IoT deployments from a single hub. For instance, services beyond cloud storage include health checks, remote access, and remote camera control for the end-user. Another feature of security cloud services is machine-learning-based tagging and human activity recognition on recorded video, with providers such as Camio providing these features [Cam20]. With this feature, recorded video data can be labeled by events (e.g., van passing, pizza delivery, red shirt), allowing users to search for a specific events in stored recordings easily [HKMA14, BKH$^+$17].

Vendor Services. E-IoT vendors will often offer cloud services for monitoring and maintenance of E-IoT systems. These services serve a variety of purposes as maintenance is an important part of E-IoT deployments. First, maintenance services monitor integrated devices in the E-IoT system network. As such, an integrator can know when a device falls offline and can address the issue before a client notices. Second, maintenance services can send integrators and clients phone and email notifications on needed updates, unplanned downtime, Internet failure, and ongoing network issues. Services such as Pakedge's Bakpak are designed to work with E-IoT and provide vendors with many features. As such, they allow integrators to provide both remote support, monitoring, and maintenance to multiple user E-IoT systems [Pak20a]. Cloud services are also used in E-IoT for secure remote access to E-IoT systems. While not all of E-IoT systems offer this service, major E-IoT vendors or device makers always offer some form of remote access solution. For instance, services such as Control4's 4sight offer users remote access through mobile apps and cloud support. Further, 4sight also allows integrators to service a specific E-IoT system remotely through a secure connection.

4.6.2 Threat Model for E-IoT Business Layer

For this layer, we consider Mallory compromising an E-IoT system through the E-IoT Business Layer. As such, Mallory is knowledgeable about cloud and remote services. Specifically, Mallory knows how to use Business Layer services (e.g., security cloud, vendor maintenance) to compromise one or multiple E-IoT systems remotely. As an attacker, in addition to knowledge on integrated services, Mallory must possess an Internet connection, knowledge on business services, and capabilities to perform phishing attacks, dictionary attacks, or web-based attacks. Mallory can be in the roles of hackers or integrators to target the Business Layer of E-IoT.
Hackers. In this scenario, Mallory may be a remote attacker with or without prior knowledge of E-IoT systems. Mallory may target E-IoT cloud and vendor services and disrupt these systems. If Mallory is a more knowledgeable attacker, she may be aware that E-IoT systems can be compromised through management services. Mallory may acquire sensitive information from E-IoT systems such as CCTV recordings, schedules, and E-IoT usage patters. Further, Mallory can also use phishing techniques (e.g., texting, email, apps) to obtain a user's or integrator's credentials and compromise one or multiple systems.
Malicious Integrators. Mallory may be a malicious integrator or an insider in an integrator company with access to user accounts and credentials for remote support. As such, Mallory can become a malicious actor (e.g., disgruntled employees, insiders) and compromise a user's E-IoT system to disrupt or for personal gain. Additionally, as Mallory is an integrator or an employee, she may know E-IoT user's financial

or societal status. This may tempt Mallory to sell information (e.g., passwords, accounts, CCTV footage) of users to external attackers for financial gain.

4.6.3 Business Layer Attacks and Vulnerabilities

Cloud Attacks. As cloud services are a part of E-IoT, cloud service threats and attacks are relevant to E-IoT systems. While architectures may vary from system to system and service to service, threats to integrated cloud services could negatively impact E-IoT system security. As an active topic of research, several surveys have highlighted threats, attacks, and best practices in cloud computing [DDGD+19, LSR+15, Rya13, Sha14, SK11, GWS10, MPB+13, SJP16, FSG+14, PAS14, SC17, XX12, AADV15, HRFMF13]. Relevant to E-IoT, surveys by Liu et al. [LSR+15], Ryan [Rya13], Shazad [Sha14], and Zhou [ZCDV17] have highlighted several key challenges to cloud security. For instance, these studies suggest that cloud components are susceptible to DDoS attacks and that encryption solutions will not protect sensitive data if the cloud provider cannot be trusted due to insiders. A comprehensive survey by Fernandes et al. raised additional issues which may occur with cloud computing [FSG+14]. Issues related to unreliable computing, data storage, availability, cryptography, sanitation, and malware can arise from systems that rely on cloud services. Further, this chapter highlights how keyloggers, phishing, malicious redirects, URL-guessing attacks, browser vulnerabilities, cross-site scripting, XML/SAML wrapping attacks, and MitM attacks may impact cloud services. Finally, a survey by Kumar et al. highlights some of the common cloud security threats, such as data breaches, weak access control, insecure APIs, application vulnerabilities, account hijacking, malicious insiders, advanced persistent threats (APT), data loss, nefarious use of cloud services, DoS, and DDoS [KG19]. In terms of E-IoT, these threats could mean that E-IoT users may lose access to their accounts, face information theft, or experience system downtime from integrated or vendor-provided cloud services.

4.6.4 Mitigation of Business Layer Attacks

Cloud Defenses. Several defenses have been proposed for threats that can impact cloud-based services. In this respect, several surveys have been conducted on the topic of cloud security, often highlighting attacks, defenses, and mitigation mechanisms relevant to E-IoT cloud services [FSG$^+$14, Rya13, SC17, AADV15, MPB$^+$13, KG19, SJP16, YLDL17, YZG$^+$19]. Majority of these works note that there are many ways to attack different types of cloud systems. As such, different mitigation strategies exist for each threat. For instance, LAM-CIoT was proposed by Wazid et al. as a lightweight authentication system to protect cloud-based IoT environments [WDBV20]. Alternatively, to defend against data breaches, properly implemented encryption should be used by cloud services. Vendors should require strong passwords and authentication practices in their cloud services to address weak access control that could compromise users, integrators, and E-IoT systems. Further, as accounts may be compromised, some articles have suggested that two-factor authentication may add an additional layer of security to cloud services [Ed 17]. As browser vulnerabilities such as XSS and redirection attacks can impact web-based GUI interfaces, users should update browsers, have malware protection, and follow best practices to prevent web-based vulnerabilities. Surveys by Fernandes et al. [FSG$^+$14] and Kumar et al. [KG19] specify mitigation strategies against cloud threats. For instance, APIs should be protected with good sanitation practices, secure development standards, signatures, and encryption. To prevent intrusion in cloud systems, the authors highlight that network-based, host-based, distributed, and hypervisor intrusion detection systems can be helpful. DDoS mitigation can improve the overall reliability of cloud services in case of volumetric attacks. Specifically, DDoS mitigation strategies may take the form of rate-limiting, proxy filtering, load balancing, crypto puzzles, and flexible network configuration depending on the cloud system and use case. As such, E-IoT vendors and manufacturers should follow these practices to guarantee the security of cloud-based components used in E-IoT.

Cloud hosting can also benefit from privacy-preserving techniques to protect a user's information. For instance, cloud service providers can provide an additional layer of mitigation by applying Homomorphic Encryption (HE) concepts to stored information [AAUC18]. With HE, concepts such as Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SWHE), and finally Fully Homomorphic Encryption (FHE) can be applied to improve data privacy. Specifically, PHE, SWHE, and FHE allow for a number of operations (depending on the type) on encrypted data without the need to decrypt the data for these operations. This allows users to store encrypted information, without the risk of exposing sensitive information to untrusted cloud providers. While these approaches are experimental and require further research, they should be considered for cloud services for E-IoT systems and storage.

4.7 Lessons Learned and Open Issues

In this chapter, we analyzed the threats and defenses concerning individual layers. However, an attacker can follow a cross-layer approach, which means he/she can attack multiple layers at once. So the security of E-IoT systems should be considered holistically. In this section, we cover lessons learned and open issues in state-of-the-art E-IoT security research.

4.7.1 Lessons Learned

Customized Deployments. E-IoT deployments are diverse and complex. There may be unique threats from deployment to deployment, especially with the numerous use cases in E-IoT. For instance, a lightning control E-IoT environment will be different from a smart media management system in terms of vulnerabilities. Specifically, a lighting system may rely more heavily on serial-based communication interfaces in every room than media management that relies on touchpanels and mobile interfaces. Further, even in a lighting system, a purely serial-based system will have different threats and vulnerabilities than a lighting control system with Zigbee/Z-Wave interfaces. Many attacks (e.g., node-capture attacks, sensory channel attacks) may have unique system-to-system consequences. Namely, if safety or motorization devices rely on E-IoT sensors, an attacker may create a much bigger safety issue if these sensors are compromised. The degree of customization in E-IoT means that one deployment's solutions and security guidelines are not applicable for all deployments.

Legacy Systems. Legacy systems are systems considered to be outdated, discontinued, and that no longer receive software or security updates. With companies such as Crestron established since the 70's, it is expected that there are legacy E-IoT systems installed worldwide [Mar18]. As these systems may not receive updates for several reasons. For instance, an E-IoT system may simply be discontinued or the manufacturer may no longer exist (e.g., Litetouch lighting control systems) [Lit06]. Alternatively, in systems with frequent updates, a user may choose not to update due to the additional costs (e.g., new devices, software costs, labor costs). For example, if an entire building is wired to function with a legacy system, it requires hiring an integrator for re-wiring and re-programming. This added labor may be a costly endeavor as opposed to simply keeping an older E-IoT system and using legacy equipment. Updated software, such as drivers, may also cost money for the end-user [RBA+20]. In other cases, discontinued devices may not work on newer systems (e.g., driver availability) and a user might choose to keep the current E-IoT system without updating to keep control of these integrated devices. E-IoT needs unique solutions that can provide protection to legacy E-IoT systems which cannot be updated.

Reliance on Integrators. In E-IoT systems, consumers rely upon integrators. This reliance on integrators may create attack scenarios where an E-IoT is compromised because of this trust. For instance, bad account management could allow attackers to compromise one or multiple systems purely due to integrators and remote support tooling. Additionally, as integrators handle devices before installation, they can be considered part of the supply chain. This adds another stage to the deployment process where devices may be compromised by an attacker or a malicious employee for an installation company. Integrators maintain and diagnose E-IoT devices in case of any issues, working directly with the client. As such, there is very little oversight on how well systems are configured. As poorly-configured devices pose a threat to E-IoT systems, a method for auditing E-IoT system security may be necessary to guarantee systems are properly configured. Further, more research into E-IoT security can assist vendors in evaluating existing best practices for integrators and developing new best practices. Finally, as E-IoT relies very heavily on integrators, new solutions are needed to protect end-users against attacks that may come from malicious insiders or poor configuration.

(Near) Future E-IoT Attacks. Attackers have always been in constant search of new types of attacks, including nation-state attackers with unprecedented attack capabilities that may target E-IoT systems. Attacks have already been observed that target E-IoT devices among other devices with Mirai being one of the most well-known. In Mirai, research has shown that CCTV systems, specifically DVRs and NVRs were targeted in the password banks [ABB+17]. These devices were possibly configured with default passwords in many cases and reflects the need for auditing and better research on E-IoT. If research is not done, E-IoT systems may end up being used in large-scale attacks, such as DDoS. Attacks would not be limited to DDoS, ransomware attacks may be different in E-IoT, both rendering a system inoperable and requiring an integrator to repair the affected system. In more advanced attacks, it may be possible for an attacker to compromise touchscreens, keypads, controllers, and other devices for cryptomining through malicious firmware or a malicious controller. Another recent and notable example of an attack has been coined the "SolarWinds" attack, where thousands of devices were compromised through vendor tool updates [Ana20, Thr20]. It may be possible for E-IoT to be affected by similar attacks in the future if precautions are note taken. This shows that trusted software must also be held to high scrutiny. Finally, in a similar manner to IoT, there are privacy concerns with E-IoT (e.g., sensitive data, usage data, occupancy, subscriptions). For instance, E-IoT systems may process sensitive data that may be used to infer a user's systems usage, home occupancy, and daily activities. An attacker can target E-IoT systems in various ways (e.g., malware, malicious drivers, etc.) and obtain such sensitive information. Additionally, E-IoT systems integrate with cloud and online services (e.g., Netflix, Spotify) that an attacker can leverage to learn more about an E-IoT user through sniffing the network flows and using metadata information.

4.7.2 Open Issues

Proprietary Communication. E-IoT supports a diversity of protocols, from publicly-known to proprietary. However, proprietary protocols in E-IoT are often closed-source, with no public specification. Additionally, E-IoT hardware and software are often unavailable to the public. Researching E-IoT communication can be difficult without vendor cooperation as much of the protocols and practices need to be reverse-engineered. As a result, many E-IoT components and protocols have not been properly investigated, and many attacks have not been discovered and addressed. For instance, serial-based proprietary protocols such as Cresnet are used extensively in Crestron systems; however, little to no security research exists on this protocol. This case is also valid for many wireless protocols as highlighted in Section 4.4 such as RadioRa, TopDog RF, and infiNET, where the security methodology used is unknown. It is a realistic assumption that vulnerabilities must exist with the age and lack of oversight of many of these proprietary protocols. The absence of known vulnerabilities is not due to strong design but through security through obscurity. Unfortunately, once adversaries find vulnerabilities in these protocols, it may lead to easily-compromised systems as security through obscurity provides very little legitimate protection. Much more research and collaboration with vendors are needed to assess the security of these protocols and develop security tools (e.g., monitoring, auditing).

Proprietary Software. In a similar manner to protocols, research on E-IoT software is a challenge as much of E-IoT development is closed-source with minimal external resources available. Further, even if research is done on one E-IoT system, different systems will follow different integration and configuration practices. For instance, components like drivers are different in every system, and the implementation may allow for completely different attacks in each system. We found that many

E-IoT systems have operated under security through obscurity for their software in addition to communication protocols, a practice that is currently insufficient. As such, it may be necessary to find flaws in E-IoT software components and correct these flaws before malicious actors compromise E-IoT installations. It may be a good idea for vendors to cooperate with research and academic communities and provide closed-source configuration software to prevent attacks before they occur. In comparison to more open industries, in E-IoT, an attacker that acquires source code for E-IoT devices may have a running start in compromising these devices before security researchers have even acknowledged the problem.

Honeypots as a Defense Strategy. It may be possible for honeypots to offer insight and warnings on possible attackers against E-IoT systems and complement existing security mechanisms. Litchfield et al. noted that honeypots can vary between different applications, highlighting that high-interaction honeypots are not suitable in some applications [LFR$^+$16]. Other solutions may be possible, for instance, Conpot, a honeypot system developed by the Honeynet Project supporting industrial protocols such as BACnet, EtherNet/IP, and Modbus. These developments are applicable in E-IoT installations as honeypot frameworks may be expanded to work with less-known proprietary E-IoT protocols [hon20, Luk20]. For instance, Mays et al. proposed a solution to defend home and building automation systems using decoy networks [MRR$^+$17]. In this work, researchers created a honeypot network on the smart automation Insteon protocol to hide communication using a dummy network and hide genuine network traffic from attackers. Such approaches could apply to E-IoT and other custom systems that rely heavily on physical components, and hopefully to understand the behavior of attackers, thus secure E-IoT systems.

Third-Party Components. While E-IoT manufacturers often practice closed-source and limited software access, there are third-party resources for software modules and hardware devices that integrate with E-IoT systems. As such, the security of these third-party components could directly impact the overall security of an E-IoT system. For instance, for E-IoT systems such as Control4 and Crestron, third-party software is required to integrate third-party devices (e.g., Televisions, receivers) [dri20, Pin19, C4F]. Device manufacturers and third-party developers will create these software modules. However, the software modules developed by third-party developers can have vulnerabilities which can leave E-IoT deployments open to attacks. In addition, since there is no security assessment for such software, malicious actors can also participate such software market places and try to distribute their benign-looking malicious software as well. For these reasons, such third-party software resources need to go through a thorough security assessment process and the integrators should not blindly trust such resources.

E-IoT Malware and Exploits. With any type of a smart system, software and hardware can be compromised through malware and exploits. As such, E-IoT vendors should participate more actively in hackathons and bug bounties for their own systems. The research community should collaborate with E-IoT vendors to find new bugs and exploits before they are compromised. As E-IoT systems integrate high-profile and sensitive locations (e.g., government offices, schools and hospitals), the security of these systems should be of utmost importance. Malware, such as a ransomware attack, may disable an entire E-IoT system, preventing employees from working and general usage of an E-IoT system. A spyware targeting E-IoT systems can cause leakage of sensitive data in such high-profile locations. Further, as E-IoT systems are often designed for centralization, an attacker who compromises an E-IoT system may now have access to all other devices (e.g., streaming boxes, alarm systems, door access systems, lighting control systems) integrated into the E-IoT deployment.

Privacy Issues. In terms of privacy issues, E-IoT users can have multiple factors to worry about. First, integrators have access to sensitive information from E-IoT systems they install (e.g., authentication codes, remote support). Clients may carelessly give information out and trust that the integration companies and their employees handle sensitive information in a secure manner. Further, the liability of integrators that do not follow proper privacy practices may need to be regulated in a similar manner that healthcare professionals must protect patient data. Secondly, E-IoT vendors who provide the E-IoT system to the users need consideration. Although vendors have privacy policies, no study has analyzed the privacy policies of E-IoT vendors in detail. Third, third-party components (apps, drivers, devices) and their privacy implications need to be considered. Such components can (un)intentionally leak sensitive data of E-IoT users. Fourth, cloud-integrated E-IoT systems can store sensitive data. Such data should be kept encrypted and processed while still encrypted in such environments in order to prevent leakage. Finally, adversaries can try to obtain sensitive data by attacking the E-IoT system and/or sniffing the network traffic.

E-IoT Security Assessment. Although vulnerability discovery tools exist for traditional computing systems and IoT environments, discovery of vulnerabilities is an unexplored topic of research for E-IoT at the moment. Due to the proprietary protocols, proprietary software, the closed nature of the E-IoT system, and the lack of research interests, vulnerability discovery and security assessment have been challenging aspects of E-IoT security. Although already existing tools for well-known protocols (e.g., Zigbee, Z-wave, Bluetooth) can be extended to work with proprietary protocols used by E-IoT, there is still a need for fuzzing, exploitation, and penetration testing tools developed specifically for E-IoT environments.

E-IoT Forensics. With existing forensic solutions have been offered for IoT systems, few solutions are catered specifically to E-IoT due to the closed-source nature of these systems [BSAU18]. Further research should aim to create forensic tools and mitigation frameworks in case E-IoT systems are compromised. Ideally, forensic frameworks should be created in collaboration to E-IoT vendors, due to limitations that span from black-box testing and closed-source projects. Additionally, E-IoT forensics frameworks need to consider integrators with multiple clients and E-IoT systems under their management. If an integrator's remote access to E-IoT systems is compromised, forensic tools should exist to know if client systems were affected and the best strategies to effectively mitigate any possible breach. Thus, there is a need for forensic frameworks catered specifically for E-IoT and E-IoT integrators.

4.8 Conclusion

The rising popularity of smart systems has led to millions of users worldwide interacting with smart devices on a day-to-day basis. Many of these devices are commodity, off-the-shelf systems (e.g., Google Home, Samsung SmartThings), easily maintained and installed by the average end-user in small deployments. However, while easy to install, commodity systems are limited and do not provide a viable solution for more sophisticated applications. For more extensive installations, E-IoT systems provide a custom-installed solutions to fit a client's needs. As such, systems such as Crestron, Control4, RTI, and savant offer a solution for more sophisticated applications (e.g., complete lighting control, A/V management, managed CCTV security), where commodity systems are insufficient. For this reason, E-IoT systems are common in locations such as high-end smart homes, government and private offices, yachts, and conference rooms. In contrast to commodity systems, E-IoT systems are usually proprietary, costly, closed-source, and more robust for their configured use cases. However, even with their popularity, very little research has focused on the overall security of E-IoT systems. Namely, no study provides a complete overview of E-IoT systems, their components, threats, and relevant vulnerabilities in the literature. To address this research gap, motivate further research, and raise awareness on E-IoT insecurities, this chapter focused solely on E-IoT systems. Specifically, we discussed E-IoT components, vulnerabilities, and their security implications. To provide a better analysis of E-IoT, we divided E-IoT systems into four layers: E-IoT Devices Layer, Communications Layer, Monitoring and Applications Layer, and Business Layer. We considered E-IoT components at every layer, the associated threat models, attacks, and defense mechanisms. We also presented critical observations on E-IoT security and provided a list of open research problems that require further research. We believe this study will raise awareness on E-IoT and E-IoT security, and motivate further research.

CHAPTER 5

DRIVER-BASED NOVEL ATTACKS AND DEFENSES FOR E-IOT SYSTEMS

5.1 Introduction

The introduction of modern commodity IoT devices has changed the everyday lives of users with the deployment of millions of smart environments (e.g., smart buildings, offices, homes) worldwide [IoT18]. While many IoT systems are easily installed by average end-users via Do-it-Yourself (DIY) applications, Enterprise Internet-of-Things (E-IoT) systems exist as an automation solution for professional settings. As such, E-IoT systems are used exclusively for applications such as smart buildings, luxury smart homes, expensive yachts, classrooms, meeting rooms, government offices, and business establishments. In these professional settings, proprietary E-IoT systems (e.g., Crestron, Control4, and Savant) introduce a robust, reliable, and custom solutions catered to meet an enterprise client's needs. As such, E-IoT systems require professional installation and specialized training to deploy. Additionally, maintenance, upgrades, and service of E-IoT systems is handled by specialized integrators and not the end-users.

Although many consumer-grade commodity IoT systems are well-understood due to their mainstream popularity, very little security research exists on E-IoT systems' design, development, verification processes, and vulnerabilities. The lack of research on these systems has led many users to overlook E-IoT systems as possible attack vectors and assume that these systems are secure. With many of these professional systems present in high-profile locations, evaluating threats for E-IoT systems should be of utmost importance. In this chapter, we systematically explore E-IoT system vulnerabilities and insecure development practices, specifically, the usage of drivers as an attack mechanism. In order to demonstrate that malicious actors can easily attack E-IoT systems, we introduce PoisonIvy, a collection of novel attacks that leverages E-IoT system vulnerabilities to an attacker's benefit. Specifically, we attack one of the integral components of E-IoT systems: drivers, which contain all of the necessary software to integrate external software and devices into E-IoT ecosystems. To show that E-IoT systems may be attacked through drivers, we analyze the highly-programmable nature of drivers and the associated vulnerabilities. With PoisonIvy, we show that it is feasible to use malicious code in drivers to perform attacks using E-IoT systems. As many third-party devices do not have verified drivers, installers must sometimes opt for unverified drivers with no method to guarantee their safety, making PoisonIvy a real and viable threat against E-IoT systems.

To raise awareness on the (in)secure development of the drivers that control E-IoT systems, we perform PoisonIvy attacks in a realistic E-IoT system testbed in a smart building setting. For this, we show how with PoisonIvy an attacker can use E-IoT system drivers to assume arbitrary control of device functions in E-IoT systems remotely. Specifically, with PoisonIvy, an attacker may remotely (1) perform DoS attacks on E-IoT system, (2) assume control E-IoT systems as an effective botnet, and (3) use E-IoT system resources to perform illicit activities (e.g., bitcoin mining, distributed password cracking). As drivers are outside of any traditional protection mechanisms, there are no defense mechanisms against attacks in E-IoT systems.

Securing an E-IoT system against PoisonIvy attacks presents distinct challenges as E-IoT systems are closed-source. E-IoT systems cannot be modified without the help of the vendor to monitor the running processes or API/system calls to detect the activities of the malicious drivers in PoisonIvy. Therefore, a passive network monitoring solution remains as an applicable methodology to detect such driver-based attacks based on the network traffic they create. Hence, to defend against PoisonIvy-style threats we present Ivycide; a passive network monitoring-based intrusion detection system designed to protect E-IoT deployments against PoisonIvy-like threats using machine learning (ML) and signature-based classification. As PoisonIvy attacks rely on network communication to communicate with the attacker or apply the attack, Ivycide operates as a standalone framework for E-IoT systems, passively monitoring network traffic for unexpected and malicious behavior. Ivycide first classifies individual incoming and outgoing network packets into four types of distinct behaviors caused by PoisonIvy attacks using ML-based techniques. Ivycide then uses these individual network packets and signature-based classification to determine the type of PoisonIvy attack occurring. To test Ivycide's performance against PoisonIvy attacks, we conducted a set of evaluations in a realistic E-IoT testbed using real E-IoT devices. Our results show that Ivycide achieves an average accuracy of 97% and precision of 94% against PoisonIvy-style attacks without any operational overhead or modification to the E-IoT system.

The contributions of this chapter are as follows:

- We demonstrate that E-IoT system drivers are a viable attack vector for smart buildings by introducing PoisonIvy, a series of novel attacks against E-IoT systems.
- We test and evaluate PoisonIvy attacks in a real E-IoT system and leverage malicious drivers to cause undesired behavior in a smart building on behalf of a remote attacker.
- We articulate the effects and implications of insecure E-IoT systems, their secure development, verification, and we open the discussion to the best practices and potential countermeasures to PoisonIvy attacks.

- We propose Ivycide, a novel intrusion detection system to protect E-IoT systems against driver-based threats. Ivycide monitors active E-IoT network traffic and detects unexpected network traffic generated by the malicious drivers of PoisonIvy.
- We evaluate Ivycide in a realistic E-IoT environment with a variety of E-IoT devices, achieving an overall accuracy of 97% and precision of 94%.

5.2 Differences from Existing Works

Our work differs from the previously works as PoisonIvy focuses on the insecurity of E-IoT system drivers, an attack vector which has been largely unexplored. In contrast to injecting malicious code into an operating system, our attacks rely entirely on weaknesses available through E-IoT system design and lack of secure development practices. We focus on the exploitation of drivers to a remote attacker's benefit and create proof-of-concept implementations of a malicious driver. Specifically, we present three specific threats that are possible to implement with a malicious driver: (1) DoS attacks on the host system, (2) remote control of a target E-IoT system, and (3) the malicious farming of system resources for unauthorized activities (e.g., bitcoin mining). With PoisonIvy, we demonstrate it is possible for an attacker to assume control of the E-IoT system in a malicious nature, solely through the use of drivers. To address these threats, we introduced Ivycide, a defense mechanism accounting for E-IoT system design and tailored specifically to E-IoT systems. Furthermore, Ivycide poses no modification or overhead to the original E-IoT system, and defends with a passive two-step network traffic defense framework.

5.3 Problem Scope and Threat Model

This section presents the problem scope and threat model for PoisonIvy attacks.

5.3.1 Problem Scope

This work assumes the existence of an E-IoT system installed in a smart building. Indeed, such E-IoT systems have experienced a rapid increase in popularity in smart buildings, luxury smart homes, expensive yachts, classrooms, meeting rooms, government offices, and business establishments. The E-IoT system's controller is connected to a network and the Internet. The attacker, named Mallory, is a malicious actor with knowledge of E-IoT systems and their weaknesses. In this scenario, Mallory develops a malicious driver for a popular device and advertises the driver through user boards such as online forums [C4F] to integrators. Mallory also creates fake accounts to give good reviews on the driver and mislead integrators. The driver advertised is not available by the manufacturer or through verified drivers, making Mallory's driver the only way to integrate a particular device into an E-IoT system. With this malicious driver, Mallory assumes the control of E-IoT system controllers and uses her machine to execute remote attacks.

Additionally, we assume that an integrator uses Mallory's unverified malicious driver for the E-IoT system deployment, introducing it into the system without issues as there are no security mechanisms in place. These assumptions are realistic as online drivers from third-party sites are not verified, and smart systems require Internet connectivity for many of their services (e.g., remote access, music streaming, movies) [Lan18]. Anyone can upload a driver to public forums easily. As integrators may download unverified drivers from any website, an attacker can create an attractive driver for integrators to download and install in their systems. For instance, unverified drivers may be offered at a third-party website which advertises them. In our current scenario, Mallory compromises E-IoT system devices indirectly through the use of a downloaded unverified malicious driver.

The consequences of driver-based attacks depend on the capabilities of drivers in a specific E-IoT system. Drivers with network capabilities may be used to attack servers and other devices when multiple controllers are infected by the driver. Additionally, access to system resources would allow Mallory to perform cryptographic operations in infected controllers. In effect, Mallory could use infected devices to mine cryptocurrency or perform other cryptographic-based operations (e.g., password cracking) [Set19]. Moreover, Mallory could simply use a driver in an attempt to overwhelm the host system, causing a local DoS condition. Any devices integrated into the E-IoT system may become unreachable through user interfaces. It is also notable that drivers may act as a "bridge" between traditional IP networks and other protocols. A driver may have the capability to communicate with devices with embedded protocols (e.g., HDMI's Consumer Electronics Control protocol (CEC), Serial, InfraRed (IR)) making previously unreachable devices reachable. For instance, work on the topic of HDMI-CEC has demonstrated that arbitrary CEC control makes attacks on multiple connected HDMI devices viable [RBAU19b].

5.3.2 Threat Model

In this chapter, we consider the following powerful adversaries as part of the threat model.

Threat 1: Denial-of-Service. This threat considers DoS attacks where Mallory disrupts the availability of an E-IoT system through the use of a malicious driver.

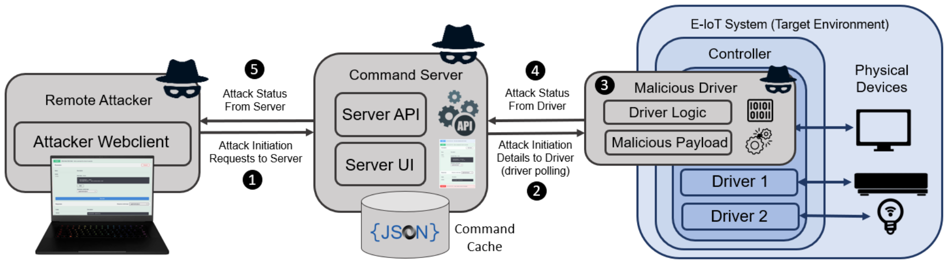


Figure 5.1: General end-to-end implementation for PoisonIvy-based attacks. Attack-related components are highlighted in gray, E-IoT system components are in blue.

Threat 2: Remote Control. This threat considers a case where Mallory assumes the control of E-IoT system devices to execute DoS attacks on local/remote devices or webservers.

Threat 3: Malicious System Resource Farming. In this threat, Mallory uses local system resources in a compromised device to perform unauthorized processor-intensive actions to her benefit (e.g., cryptocurrency mining [Set19], password cracking [Dev13])

Note that this chapter does not consider attacks that focus on traditional Linux or other mainstream operating system malware. Similarly, this chapter does not consider protocol-based vulnerabilities (e.g., Zigbee vulnerabilities).

5.4 PoisonIvy Architecture

To demonstrate E-IoT drivers as a viable threat vector, we developed PoisonIvy, a series of driver-based attacks. In this section, we detail the end-to-end implementation of PoisonIvy attacks, which become practical and applicable due to E-IoT component implementations not built with security in mind. Such implementation involves the interaction of four modules: *remote attacker, command server, malicious driver, and the target environment.*

5.4.1 PoisonIvy Overview

The proposed end-to-end implementation of PoisonIvy is highlighted in Figure 5.1. In this architecture, the integrator has unknowingly installed the malicious unverified driver, and the E-IoT system controller has been compromised. As explained earlier, this could be achieved through a forum post advertising a malicious driver as benign. The attack begins with the remote attacker (e.g., Mallory) initiating an attack with a webclient such as a laptop by communicating with the command server ❶. The command server grants Mallory an intermediary point of communication between her device and infected controllers, and includes three components: the *server API*, *server UI*, and the *command cache*. The server API represents the primary endpoints (e.g., REST Architecture) of the server, which can be requested by Mallory or the malicious driver. Mallory uses the server UI component executed by the webclient, which grants her a visual interface, to initiate attacks and view the attack's status. As the last component of the command server, the command cache stores attack initiation requests fetched by malicious drivers through the server API endpoints.

Once the command server receives initiation requests from Mallory, the malicious driver can now query the command server for new attack details ❷. The Malicious driver is the core of PoisonIvy attacks and contains the *driver logic* and the *malicious payload*. As such, the driver logic controls a smart device in an expected manner, which allows the driver to appear as a benign driver. In contrast, the malicious payload contains the attacker's malicious code for the execution of the attacks ❸. Finally, the target environment contains the smart system's controller and the drivers of the smart environment. Mallory takes control of the target environment's functions through the malicious driver. As attacks complete, the malicious driver sends back the attack status to the command server ❹. The attack status includes feedback to the attacker from the driver, such as hashing results, errors, or success codes, and can be then queried by Mallory from the command server ❺.

Remote Attacker. In PoisonIvy, the remote attacker is the malicious actor of the attacks. The primary purpose of the remote attacker is to send commands to the command server to be executed by a driver-compromised controller. In this case, Mallory uses the attacker webclient, which is any web-enabled device such as a laptop, tablet, or phone that is used to initiate the attack. Additionally, the remote attacker receives information from the command server such as attack results, hashing status, or available controllers to use for attacks.

Command Server. The command server module acts as an intermediary communication point between remote attacker and malicious driver. Controlled devices query the command server for new attacks to execute. Additionally, the command server is divided into three components: the *server API logic*, the *server UI*, and the *command cache*. The server API component contains all the API programming logic and REST paths needed for communication between the remote attacker and the malicious driver. The server UI component, in contrast, is an interface for Mallory to interact with and view reports of devices. These reports may contain results on successful hashing attempts, controlled device status, or the current status of an attack. The server UI component, depending on the type of attack performed, could be implemented as a fully-dynamic website (to view reports on attacks) or as a simple command-line interpreter for simplicity. Finally, the command cache holds the last executed commands and other responses from controlled devices. This cache allows multiple compromised smart controllers to fetch the same execution message stored in the command server. Additionally, the command cache allows Mallory to disconnect while an attack is active to retrieve the responses from an attack at a later time.

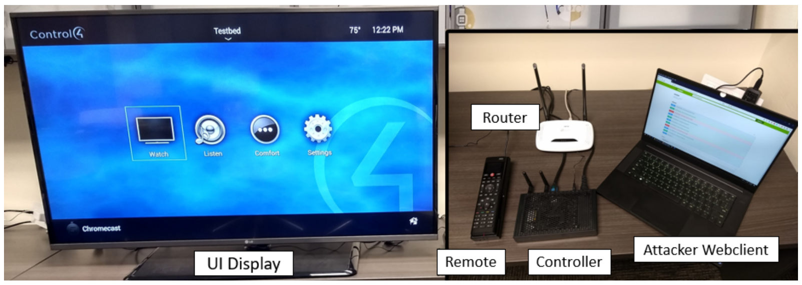


Figure 5.2: E-IoT system testbed used to implement PoisonIvy attacks in a smart building setting.

Malicious Driver. The malicious driver serves as the attack vector and performs the bulk of malicious operations in PoisonIvy. When the malicious driver is active, the driver contacts the command server for new attacks to execute, multiple controlled devices may contact the same server. The malicious driver module contains two components: the *driver logic* and the *malicious payload*. The driver logic can be seen as a standard operating code to allow the malicious driver to appear and operate as a non-malicious driver. A malicious driver must appear as if it is benign, providing all standard operations a legitimate driver offers. The malicious payload component contains all the code required to execute attacks. The malicious payload may cause memory leaks, perform malicious requests to servers, eavesdrop, and otherwise execute any operation beneficial to Mallory.

Target Environment. Serving as the host of malicious drivers, the target environment is the E-IoT system itself. These include any system in a smart building, luxury home, or office, which may be compromised by a malicious driver. The target environment is connected to the Internet and contains all of the devices of the affected smart system, including the centralized controller. As part of the PoisonIvy's end-to-end implementation, the target environment is one of the targets compromised by the remote attacker during attack activation.

Table 5.1: Hardware & software used in PoisonIvy attacks implementation and testing.

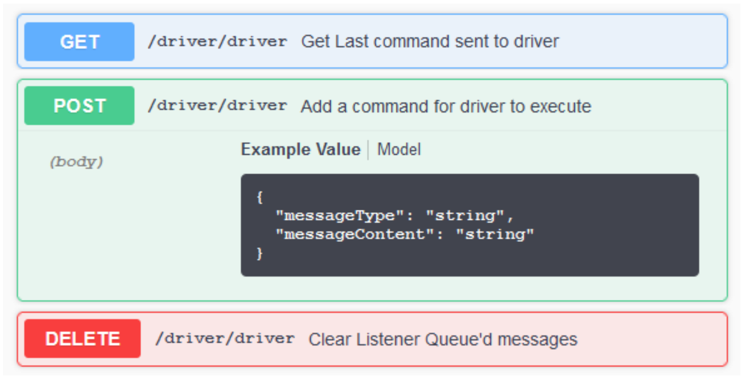
Hardware	Software
Control4 EA-1 Controller	Microsoft Visual Studio Code 1.4.11
Control4 SR-260	Control4 Driver Editor 3.0.1
LG 49LX570H	Control4 Composer 2.10.6
TP-Link TL-WR841N Router	Jersey JAX-RS with Swagger.io
Razer Blade 15 Laptop	Amazon AWS Elastic Beanstalk

5.5 Evaluation and Realization of PoisonIvy Attacks

In this section, we demonstrate the implementation of PoisonIvy attacks on our realistic E-IoT testbed. Further, we evaluated the effects of PoisonIvy attacks on the E-IoT system in detail.

5.5.1 PoisonIvy Implementation on Real E-IoT Devices

We created a malicious television E-IoT driver (as detailed in Figure 5.1) and an E-IoT system testbed with real Control4 devices as shown in Figure 5.2. Control4 was selected as it is one of the most popular E-IoT systems available in the market, named a leading brand in E-IoT for five years in a row [Con18]. The testbed included vendor-specific devices and is configured to function as a small E-IoT system (Table 5.1). We utilized Driver Editor, a tool available for the development of drivers in Control4 [C4D14]. Additionally, we used LUA, an open-source programming language which is the core development platform of Control4 drivers [Zap17]. We configured the E-IoT system using Control4's Composer 2.10.6 and with an EA-1 as the main controller. To grant Internet access to the devices included in the testbed, we configured a network with the TP-Link TL-WR841N Router. We verified the running version of LUA in Control4 devices as LUA 5.1 programmatically (executing a script which returned the running LUA version). To implement Poi-


GET
/driver/driver Get Last command sent to driver
POST
/driver/driver Add a command for driver to execute
(body)
Example Value | Model
DELETE
/driver/driver Clear Listener Queue'd messages

```
{
  "messageType": "string",
  "messageContent": "string"
}
```

Figure 5.3: Swagger interface for PoisonIvy remote attack execution with JSON object. The *messageType* field determines the attack type and *messageContent* for extra parameters.

sonIvy realistically, we created a command and control webserver with a RESTful API in JAX-RS hosted in Amazon AWS. The Swagger-based web interface for the server can be seen in Figure 5.3.

Execution JSON Object. A JSON object is used by PoisonIvy modules to exchange attack details. The JSON response object consists of two fields. The *messageType* field is the attack type to be executed. The *messageContent* field contains additional information such as the target URL to attack.

Command and Control Webserver. To implement PoisonIvy realistically, we created a webserver that could be queried by the malicious driver for attack commands. The server hosts a RESTful API implemented using JAX-RS and Swagger add-on as a UI interface. The Swagger-based web interface for the server can be seen in Figure 5.3. Primarily, the webserver has one endpoint /DRIVER/DRIVER with POST, GET, and DELETE. In our implementation, the REST request types were used as follows:

- *GET:* Fetches the last JSON object received by the API, and is used by the driver to poll for the last command received.

- *POST:* Submits a new JSON object to be stored by the API, overwriting the previous values.
- *DELETE:* Clears the stored object fields, setting both the *messageType* and *messageContent* to NULL.

5.5.2 Software Modules

To execute PoisonIvy attacks and implement the malicious driver, we created a number of LUA software modules.

1) Remote Polling Module: The remote polling module awaits commands from an attacker-managed server which issues commands to execute specific PoisonIvy-based attacks. As with all software modules, the remote polling module was written in LUA. Pseudocode to demonstrate the polling and selection process can be seen in Algorithm 1. As a traditional REST client, the first initialization request is "DELETE: [URL]/driver/driver" which clears the command cache in the webserver (Line 2). Once initialized, the module executes as a loop every three seconds. The command server's address is polled with the request "GET: [URL]/driver/driver" and the response JSON object stored into a local cache (Line 4). This newly received command is compared to the last command received if the command is different (Line 5), then the attack specified is initiated (Line 6). After the execution, the local cache and the server JSON messages are cleared (Line 7). After the execution is finished, the loop waits and initiates again.

2) LUA Hashing Module: A notable challenge for the development of PoisonIvy was the creation of a hashing module to perform cryptographic hashing (SHA-256) operations in a LUA-based driver. LUA 5.1 does not support the bitwise operations from the standard libraries; thus, the hashing algorithm had to be adapted for

Algorithm 1 PoisonIvy attack polling algorithm

```
1: Initialization;                                   // Initializes driver variables
2: DELETE: [URL];                                    // Clear server cache
3: while true do                                     // Operation loop
4:     LocalCache ¡- GET: [URL];                     // Get server cache
5:     if New Command in LocalCache then
6:         Execute Attack Specified;
7:         DELETE: [URL];                            // Clear server cache
8:     end if
9:     Wait 3 Seconds;
10: end while
```

this version of LUA. We utilized several sources of code for the implementation of SHA-256, commonly used for password hashing and cryptocurrency mining [MO14]. The SHA-256 hashing algorithm was implemented in pure LUA for this module to effectively test cryptography-based threats in PoisonIvy.

3) Memory Exhaustion Module: To perform attacks, we created a software module that would allow an attacker to expend system resources in the controller. This module was implemented as a LUA table data type and a loop that iterated over itself, adding content to the table to expend system resources. This code caused a DoS condition in the target system.

4) Network Request Module: The network request module was created so that PoisonIvy could perform GET requests to remote URLs. The module was implemented using Control4 specific API command C4:urlGet() to execute a GET request to a given endpoint. The request is placed in a loop, in effect, this allows the attacker to perform a set number of requests or continue making requests indefinitely. While the API command returns the fetched data, the data is only used for confirmation of a successful query.

5.5.3 PoisonIvy-based Attacks

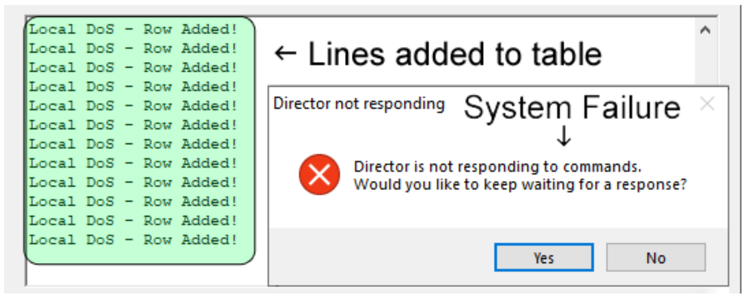
In this sub-section, we realize the PoisonIvy attacks and discuss the results and implications of each attack. All of the attacks presented begins with the initiation of the driver and by polling the server. The attacks were executed using an EA-1 controller with a malicious driver querying the AWS-hosted server. The Razer Blade 15 laptop was used for the remote execution of the attacks.

Attack 1: Denial-of-Service. This attack was developed to demonstrate that Threat 1 is possible through PoisonIvy. This attack implements a DoS condition in a local system by causing memory exhaustion in the host controller.

Step 1 - Activation. The activation of the driver was executed remotely through the attacker's web interface. With this web interface, the MESSAGETYPE field of the JSON object was set to "DOS" to initiate a local DoS attack.

Step 2 - DoS Payload. As the driver polls the server with the remote polling module, the activation message was successfully interpreted by the driver, and the attack was initiated. The action activated the memory exhaustion module and begins to consume system resources in the target device.

Evaluation: This attack was entirely successful as the device hosting the driver (the controller) was rendered inoperable within five seconds of activation, affecting the controller in two ways. First, any configuration software connected to the main controller lost connection and was locked up. Figure 5.4 shows the configuration software losing connection with the controller during our attack. Second, any communication with the central controller was interrupted, meaning that a user would have no way to use the E-IoT system once this attack was active. On-screen interfaces (e.g., Television UI interface, computer interface) and handheld remotes lost communication with the main controller, preventing access to any of the other smart devices integrated with the controller. With no form of disabling the loop, the only



```
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
Local DoS - Row Added!
```

Director not responding

Director is not responding to commands.
Would you like to keep waiting for a response?

Figure 5.4: Attack 1 (Memory Exhaustion) implementation results. Figure shows items being inserted into a LUA table, creating resource exhaustion.

option to re-establish the device was to power cycle. It is even possible to run this attack on the controller's initiation, effectively rendering the device inoperable even after rebooting.

Attack 2: Remote Control. The remote control attack serves as a way to demonstrate the feasibility of Threat 2. Primarily, we show that a remote attacker (Mallory) can take control of one (or many) devices and command them to make continuous malicious requests to a specific server, negatively impacting the critical servers. We follow Figure 5.5 for the following steps.

Step 1 - Activation. This attack was activated through the use of the available web interface by the remote attacker laptop. The JSON object MESSAGETYPE field was set to "BOT" and MESSAGECONTENT to "www.pucherondon.com" to initiate a repeated querying to the target site.

Step 2 - Execution. As we did not want to disrupt the functionality of the target webserver, we used the network request module with a loop of ten requests to the target webserver. Once the JSON object was received, the requests were made to an external website "www.pucherondon.com". All of the requests were successful on the target webserver.

Evaluation. We evaluate this attack by the success of remote attack activation and the requests to the target site. The attack received the remote commands from

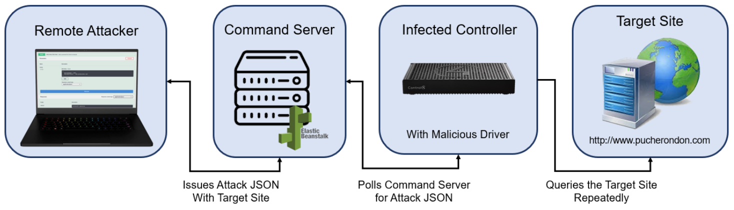

Remote Attacker
Command Server
Elastic Beanstalk
Infected Controller
With Malicious Driver
Target Site
http://www.pucherondon.com
Issues Attack JSON With Target Site
Polls Command Server for Attack JSON
Queries the Target Site Repeatedly

Figure 5.5: Implemented botnet attack model for Attack 2. The remote attacker initiates the attack as shown in this figure.

the remote attacker laptop and then performed web requests upon the target website without any issues. Thus, the commands issued by the remote attacker were executed on a target site. While the request loop was kept to ten executions, one can easily increase to any number of requests. The purpose of this test was to demonstrate that remote activation and querying of a page is possible via PoisonIvy attacks. Additionally, if there are multiple controllers with the malicious driver, infected devices could perform a more effective distributed attack on a target webpage by increasing the number of requests, creating a Distributed Denial-of-Service (DDoS) attack. The goal of this implementation is to demonstrate the remote attacks are possible, which was proven by our attacks. Additionally, scaling is very straightforward, which can be done by increasing the number of compromised controllers with malicious drivers available to the attacker. It is possible that complex E-IoT deployments may be compromised with drivers and used for DDoS attacks in a similar manner to Mirai with E-IoT systems if not properly secured [AAB^{+}17].

Attack 3: Malicious Resource Farming. Resource farming attack was developed to demonstrate that system resources may be used to Mallory's benefit for a purpose such as bitcoin mining. Currently, bitcoin uses a double hash SHA256 operation (Equation 5.1) where, B represents recent transactions, N represents a nonce, and T is the target value [MO14].

$$T > SHA256(SHA256(B.N)) \tag{5.1}$$

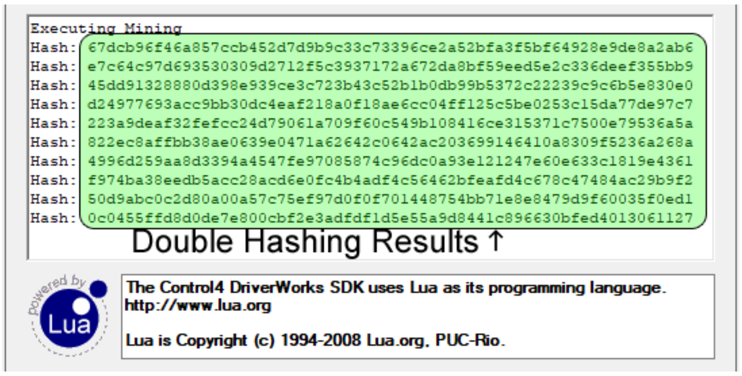
For PoisonIvy, we performed the required cryptographic operations used in bitcoin mining. To demonstrate that such operations can be done within a driver, we executed multiple hashing operations in the infected device.

Step 1 - Activation: This attack was activated similar to the previously introduced attacks, using the web interface and a JSON object request. To initiate this attack, the MESSAGETYPE field was set to "MIN" in the outgoing JSON object from the client computer, the driver interpreting the change as a request to perform mining-based cryptographic operations.

Step 2 - Execution: After the internal remote polling module processed by the driver, the MESSAGETYPE field, the driver calls the LUA hashing module which executes ten hashing operations using controller resources. Similarly to cryptocurrency mining, the hashing operations were performed with a static B value and random nonce values for T in each iteration.

Evaluation: The driver managed to perform all hashing operations successfully. Figure 5.6 shows a sample of ten mining operations executed in the malicious driver. In the case of multiple devices infected with malicious drivers, the number of machines performing hashing operations on the attacker's behalf could be increased. This type of resource farming attack could negatively impact the performance of an E-IoT system depending on how many cryptographic operations are executed per minute. The number of hashing operations per minute can also be adjusted to avoid detection.

Discussion and Findings As PoisonIvy attacks were developed and tested, we demonstrate that insecure software development, lack of built-in security, and untrusted drivers result in malicious activities. We have found drivers to be a



```
Executing Mining
Hash: 67dcb96f46a857ccb452d7d9b9c33c73396ce2a52bfa3f5bf64928e9de8a2ab6
Hash: e7c64c97d693530309d2712f5c3937172a672da8bf59eed5e2c336deef355bb9
Hash: 45dd91328880d398e939ce3c723b43c52b1b0db99b5372c22239c9c6b5e830e0
Hash: d24977693acc9bb30dc4eaf218a0f18ae6cc04ff125c5be0253c15da77de97c7
Hash: 223a9deaf32fefcc24d79061a709f60c549b108416ce315371c7500e79536a5a
Hash: 822ec8affbb38ae0639e0471a62642c0642ac203699146410a8309f5236a268a
Hash: 4996d259aa8d3394a4547fe97085874c96dc0a93e121247e60e633c1819e4361
Hash: f974ba38eedb5acc28acd6e0fc4b4adf4c56462bfeafd4c678c47484ac29b9f2
Hash: 50d9abc0c2d80a00a57c75ef97d0f0f701448754bb71e8e8479d9f60035f0ed1
Hash: 0c0455ffd8d0de7e800cbf2e3adfdf1d5e55a9d8441c896630bfed4013061127
```

The Control4 DriverWorks SDK uses Lua as its programming language.
http://www.lua.org

Lua is Copyright (c) 1994-2008 Lua.org, PUC-Rio.

Figure 5.6: Hashing process as executed by PoisonIvy attacks.

viable threat vector; thus, we have coordinated and shared of our findings with Control4 for further discussion. Without any form of verification, an integrator may download a compromised driver and allow a malicious actor to compromise an E-IoT system to her benefit. All of the proposed attacks were implemented successfully, the implications which could negatively impact E-IoT systems. In Attack 1, we demonstrate that an entire system can be rendered unusable at the command of an attacker and is possible due to the ability of drivers to consume system resources without limitations. The attack presents a viable method of disabling access to security systems, gates, doors, or any other system which is integrated into an E-IoT system. For instance, if gate access or panic button is controlled purely through an E-IoT system, a user will not be able to operate the gate access or a panic button while a DoS attack is active. Attack 2 is made possible due to the lack of limitations on connections to external websites and shows how an attacker can perform DDoS-type of attacks on target webpages using multiple controllers.

There have been documented cases of malware purposely accessing illegal websites to frame the system owners [New09]. An attacker with a compromised E-IoT system may request illegal websites and frame the owners for illicit activity. In this chapter, we cited an example of one use-case of cryptographic operations as cryptocurrency mining. These results also imply that an attacker may also perform any other hardware-intensive actions such as password cracking. Ultimately, our implementations show that drivers as attack vectors have many possibilities. Attack 3, is possible due to a lack of restrictions in the LUA implementation and unfettered access to system resources. Further, with processor-intensive operations, a compromised controller could also be used for cracking hashed passwords. An attacker with a list of passwords to crack could use the processing power of compromised controllers to attempt to reverse password hashes, a very similar operation to cryptocurrency mining. As PoisonIvy-style attacks present a substantial negative impact on E-IoT systems, acknowledging these threats and finding solutions should be of utmost importance. We believe that a security verification mechanism is needed for E-IoT systems that verify the integrity and origin of the drivers. In addition, an E-IoT system controller needs inherent security mechanisms that limit external communications, resource consumption, and access to E-IoT system resources.

5.6 Attack Discussion

In this section, we discuss the implications of these attacks and possible defense mechanisms for PoisonIvy attacks.

PoisonIvy-based Attacks. With PoisonIvy, we explored possibilities of attacks that could be implemented through E-IoT smart device drivers. Depending on the capabilities of the driver, in addition to the attacks demonstrated in this chapter, it is possible for a driver to act as a keylogger, capturing key-presses relayed to a device from any interface. For instance, if a user has a media device with login credentials for web services (such as Netflix in an integrated AppleTV) an attacker may be able to capture those credentials. Specifically, if a user uses an infected driver to communicate to the media device and enter their password with the arrow keys on an on-screen keyboard, a malicious driver could intercept the key-presses and capture the user's credentials. Another possible attack, depending on the driver implementation, may involve weak script interpreter implementations. If there are weaknesses to the interpreter, an attacker may be able to perform injections through a driver using known vulnerabilities.

Challenges in Standardization. One of the biggest challenges in E-IoT systems and IoT as a whole is standardization. There are countless companies, protocols, and implementations of many technologies depending on the vendor. Drivers are no different; how drivers are implemented from system to system are different. As attackers become more sophisticated, manufacturers cannot rely on a closed-source system for security. However, having multiple vendors agree in a standard to interface with devices is not a challenging task. An effort to standardize how drivers and how they are implemented would be the first step towards security. Further, with many E-IoT systems deployed in the world, legacy systems present a problem to developing defense mechanisms against any new threats. By definition, many of these legacy systems can not be upgraded to the latest security practices [Sta19]. A great number of systems may not longer be supported or their vendor is no longer in business (e.g., Litetouch, X10-technology) [Jul15]. As such, there are many legacy E-IoT systems which may be too costly or too impractical to upgrade. Legacy E-IoT raises the issue that these systems cannot be patched easily and be compromised by a knowledgeable attacker. Defense mechanisms must consider the limitations that come with legacy systems and how to secure them.

Risk Awareness. Most vendors have documented best practices for the installation of their devices, discouraging risky configurations such as port forwarding directly to their controllers. However, installers will still port-forward their devices, exposing them to the Internet as it is the easiest solution. As many controllers were not designed to be connected directly to the Internet, they could become compromised by an attacker if exposed. First, the usage of VPNs needs to be documented in a proper manner for remote access to devices. Second, E-IoT system integrators should be wary of drivers on the Internet and favor trusted drivers provided by E-IoT system vendors. We hope this chapter besides motivating further research on protecting E-IoT systems from novel types of attacks, can raise awareness for integrators on what malicious code is able to do, and allow them to evaluate the risks of using unverified drivers. Second, integrators must be aware that because one version of a driver is verified, updated versions may not. This could create a false sense of security, as an attacker may be able to verify a benign driver, then link to their own page for an updated, malicious version of a driver.

Comprehensive Driver Validation. As driver development for every vendor is different, vendor certification of drivers is the most effective step towards the security of the E-IoT system drivers. As of now, the development and distribution of unverified drivers come without any form of source control, standards, or code analysis. Existing driver certification needs to evaluate beyond functionality and consider that code could be implemented maliciously. Additionally, E-IoT system vendors could allow for the submission of drivers and perform code analysis to drivers submitted to their platform. Such an idea would create a larger number of drivers available to vendors. Vendors should then highlight that unverified drivers should be used at the integrator's own risk.

5.7 Ivycide Architecture

To address PoisonIvy attack threats, we introduce Ivycide, a passive network-based intrusion detection system (IDS), easily configurable to detect traffic anomalies in E-IoT controller network traffic.

5.7.1 Design Considerations and Challenges

In this section, we first include the distinct challenges of E-IoT systems that make it difficult to protect against PoisonIvy attacks and require a specialized solution such as Ivycide. The design considerations of the Ivycide framework are driven by these challenges.

Closed-Source E-IoT. E-IoT systems are very often closed-source that makes many accepted defense strategies very difficult without vendor cooperation. Further, software for configuration is not available to researchers and consumers. As such, a defense strategy must be designed with closed systems in mind. In the case of Ivycide, mechanisms must be created using features available to integrators and consumers. Without special permissions, source code, API/system call hooking, performance analysis, and other features, a defense system is a notable challenge to outside developers. Thus, Ivycide is needed as current defense systems may not work with the limited access of E-IoT systems and the lack of support from E-IoT vendors.

System-to-System Differences. There are many different E-IoT system vendors, each E-IoT system often with their type of configuration. As such, traffic will vary from E-IoT system to system, even if they are from the same vendor. For instance, a system that controls a single room (e.g., conference room, theater) will be vastly different in traffic than a large-scale system (e.g., whole home, smart office, yacht complete integration). Further, systems may differ from the services integrated. For instance, some users may opt for a fully-offline system, while other users may request a system that integrates music services (e.g., Spotify, TuneIn, Weatherbug). Since all systems are custom, a custom solution needs to be proposed for PoisonIvy attacks as existing solutions may not consider or be too costly for complex. A solution for E-IoT needs to be flexible, affordable, and needs to consider that systems may be updated and modified by the integrator.

E-IoT Traffic. In contrast to E-IoT devices, E-IoT controllers have some unique characteristics in E-IoT environments due to their role in E-IoT systems. First, the controller is the hub of all communication, as such, integrated devices (e.g., keypads, touchscreens, televisions) communicate with the controller. Second, E-IoT controllers will often have audio and video interfaces, such as audio out, for streaming services and internet radio (e.g., Spotify, Rhapsody, TuneIn). Thus, in some systems, the E-IoT controller will handle the streaming service communication traffic. Finally, the E-IoT controller often communicates with the vendor's web services and configuration software. In most cases, this means that the only way to modify the system (and drivers) for both benign or malicious purposes will be through the E-IoT controller and a network connection. As the E-IoT controller acts as the central communication hub, monitoring the network traffic of only the controller instead of all of the E-IoT devices can be useful to detect PoisonIvy attacks.

Constraints of E-IoT on the PoisonIvy Attacker. While PoisonIvy attacks demonstrate the capabilities of attackers using malicious drivers, there are limitations of E-IoT systems on PoisonIvy attacks that can be of use by a defense system such as Ivycide. First, a PoisonIvy attacker is limited on how they may communicate to the Internet. Namely, an attacker must rely on the driver's API to communicate

Table 5.2: Examples of expected network traffic by device type.

Device/Service Type	Examples	Expected Traffic
Displays	Televisions, projectors	Power state, device state, volume state, version, media metadata.
A/V Equipment	Media Centers, A/V Switchers, Receivers	Power state, device state, volume state, authentication.
User Interfaces	Touchscreens, Keypads	Device state, user input, external communication, authentication.
Sensors	Motion sensors, humidity sensors, alarms	Power state, device state, sensor data, user input.
Software Services	Rhapsody, Spotify	Media metadata, volume state, external communication, authentication.
Lighting Control	Lighting modules, dimmers, switches, relays	Power state, device state, light levels, user input.
Motorization	Motorized blinds, projector lifts	Power state, device state.

remote servers. Second, an attacker must rely on this form of external communication for the core of Attack 2 and Attack 3 (Section 5.5). Finally, traffic from a malicious driver originates from the controller. Thus, an attacker using the malicious drivers has only one device they can establish communication to external servers (e.g., CnC server, target servers) and cannot execute attacks from other devices integrated in the E-IoT system. Knowing these limitations, a defense solution such as Ivycide can rely on network communication from the E-IoT controller to identify and detect malicious activities originating from a malicious driver.

5.7.2 E-IoT Devices, Drivers, and Expected Traffic

Modifications to E-IoT systems are not done frequently for several reasons. First, there are costs associated with contracting an integrator and purchasing new drivers after initial installations. Additionally, if a device integrated into an E-IoT system needs a replacement (e.g., damages, upgrades), an integrator will often replace the device with a similar device to fulfill the same purpose. As such, it may not be necessary to retrain a learned model for an E-IoT system with similar replacements. Devices integrated into E-IoT have expected communication traffic dependent on the type of driver and device. In Table 5.2, we show some examples of how device type defines the communication traffic of each integrated device. For instance, a display (e.g., television, projector) will communicate with the E-IoT system with information such as power state, firmware version, and volume levels. Further, depending on the E-IoT devices and their drivers, network traffic will be different. For instance, a driver for a device controlled through Zigbee by the E-IoT system should not create any additional IP network traffic. We highlight some examples of driver types as follows:

Driver types: We highlight three types of drivers used for devices in E-IoT, and how each type of driver affects the E-IoT network traffic.

- **Non-IP Drivers.** Drivers (e.g., Zigbee drivers) that do not use any IP network communication to control integrated devices. These drivers should not add any additional traffic to the E-IoT system. Since PoisonIvy attacks require Internet connection, they will not function as this type of driver.
- **IP Drivers.** Drivers that use IP network communication to connect to devices or services (e.g., IP TV driver, Spotify Drivers). These drivers will create network traffic relevant to the device or service. More information on expected network traffic is highlighted in Table 5.2.
- **Drivers with Remote Validation.** Drivers that require online validation, such as a licensed driver that must validate a license key with the developer of the driver. These drivers will have communication with a remote server.

5.7.3 Terminology

In this sub-section, we provide terminology necessary to introduce Ivycide.

Expected Operation. We define the expected operation of an E-IoT system, as usage of an E-IoT system in a manner that is benign such as selecting video sources, listening to music, and menu navigation. Activity occurring from malicious drivers is outside of expected operation.

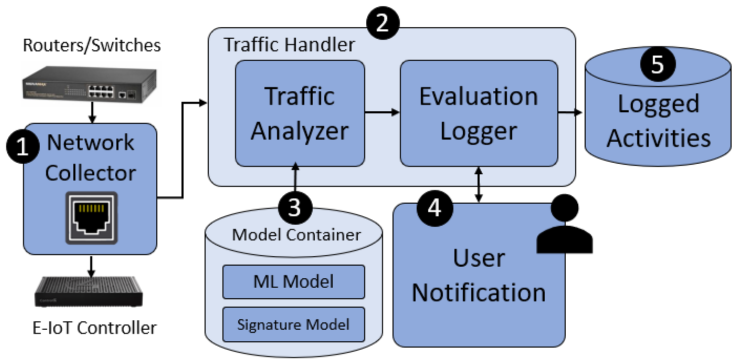


Figure 5.7: Architecture of Ivycide, modules numbered.

Expected Traffic. We define benign traffic as any IP network communication which is caused by the expected operation of the E-IoT system.

Unexpected Network Behavior. Unexpected behavior occurs from unexpected network traffic due to active PoisonIvy attacks in the E-IoT system. For the scope of this chapter, unexpected behavior is observed through the IP network communication.

5.7.4 Ivycide Overview

Ivycide is designed to protect E-IoT systems from PoisonIvy-based threats. It aims to detect PoisonIvy attacks via passive network monitoring and a two-step classification approach. In the first step of the classification, individual attack patterns are detected via a ML-based classifier, whereas in the second step, series of patterns are checked against attack signatures and the type of the attack is determined via a signature-based classifier.

The proposed Ivycide architecture is composed of five different modules as seen in Figure 5.7. The first module is the *Network Collector* which captures network traffic incoming and outgoing from the E-IoT controller ❶. Further, the Network Collector pre-processes E-IoT network traffic and forwards it to the Traffic Handler.

The *Traffic Handler* evaluates incoming traffic and logs suspicious network activity using two sub-modules: the *Traffic Analyzer* and *Evaluation Logger* ❷. The Traffic Analyzer sub-module is used as the first step, performing ML-based classification of individual E-IoT network traffic packets. These packets are classified into the four types of behaviors; benign, Unexpected External Request (UER), Command-and-Control (CnC), and Activation. As PoisonIvy attacks are composed of a series of such behaviors, as the second step, the Traffic Analyzer applies a signature-based classification on a set of classified packets within a time window to determine the type of attack occurring. The Evaluation Logger sub-module is then used to forward suspicious network packets and analysis results to the user notification and logged activities modules. The *Model Container* stores the ML model and Signature Model used by Ivycide's traffic analyzer ❸. The *User Notification* module is used to alert and notify the user on warnings and suspicious activities ❹. Finally the *Logged Activities* module stores all the the suspicious packets and classification results from the Traffic Handler ❺. This logged information may be queried later for reference, or further analysis.

5.7.5 Network Collector

The Network Collector allows Ivycide to passively collect incoming and outgoing traffic to the E-IoT controller. As such, Network Collector only captures TCP/IP network traffic relevant to the E-IoT controller. Additionally, this capture is passive, as packet manipulation of E-IoT traffic may cause undesired operation for the E-IoT system. Further, the captured traffic data that is irrelevant to Ivycide is then filtered out (e.g., internal LAN communication). Relevant packets to Ivycide (e.g., communication from controller to external servers) are parsed and features are

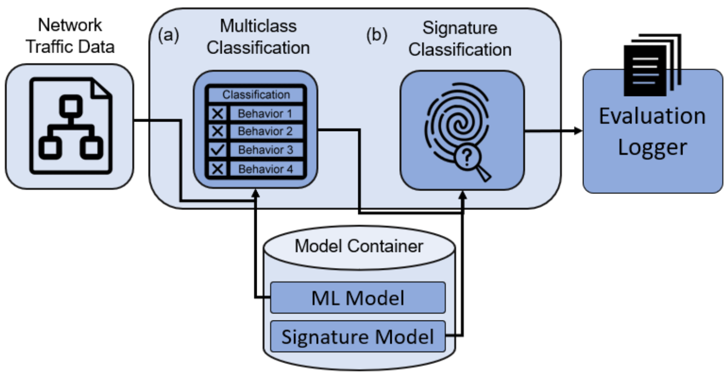


Figure 5.8: Ivycide classification process.

extracted by the Network Collector for further processing. Filtered and formatted network data coming out of Network Collector includes all packet information and additional attributes necessary for Ivycide training and classification (e.g., timestamp, data length, TCP or UDP flags).

5.7.6 Traffic Handler

The Traffic Handler acts as the classification stage for Ivycide and is composed of two sub-modules; the Traffic Analyzer and the Evaluation Logger.

Traffic Analyzer.

The Traffic Analyzer is one of the core components of Ivycide and performs the ML multiclass classification of incoming/outgoing data to the controller. Additionally, the Traffic Analyzer performs signature-based classification using a series of attack patterns/behaviors to determine the type of the attack. This two-stage process is required since the type of the PoisonIvy attacks cannot be identified from a single malicious packet or a single attack behavior. Further, classifying behaviors per packet can yield to more flexibility and types of signatures for future attacks. We refer to Figure 5.8, for the ML and signature-based classification processes employed by Ivycide. The first step of the Traffic Analyzer is the Multiclass Classifier. In this step, Ivycide attempts to classify network traffic as benign or as different types of unexpected behaviors/patterns (unexpected external request, activation, malicious CnC). Once network traffic is classified with ML, signature-based classification can take place. However, if a packet is classified as benign, no further classification is needed. For signature-based classification Ivycide follows a set of rules and attempts to determine the type of attack that occurred depending on the unexpected activity observed within a set timeframe. The resulting classification and timestamps are then forwarded to the Evaluation Logger and user notification modules.

Multiclass Classifier. The Traffic Analyzer uses ML classification to infer the type of network activity occurring with the E-IoT controller. As Ivycide is designed to be flexible and better fit the heterogeneous nature of E-IoT systems, different ML algorithms and models may be used for better accuracy. For Ivycide we defined four distinct types of behaviors for E-IoT systems and PoisonIvy attacks. Specifically, attacks can be identified by a combination of these behaviors. While we define four distinct types of behaviors associated with E-IoT systems and attack behaviors, more types of behaviors may be learned and added with newer threats. A more detailed description of behaviors classified during this stage are highlighted below:

- **Benign.** Benign behavior is expected network traffic and does not raise any flags for Ivycide. Benign behavior is dismissed from further analysis.
- **Unexpected External Request.** Traffic classified as unexpected external requests (UER) is unauthorized traffic from the E-IoT controller to external servers. Usually these requests are repetitive during a short span of time and can be associated with PoisonIvy DoS attacks.

- **Malicious CnC Requests.** CnC requests are unexpected network traffic used by a malicious PoisonIvy driver to communicate with the command server. As such, malicious CnC requests are associated with an infected E-IoT system actively communicating with a command server.
- **Activation.** Activation requests are unexpected network traffic used by PoisonIvy to initiate attacks. When activation requests are detected, Ivycide can determine that an attack was initiated. Thus, these requests may be used to determine the type of traffic that occurs after an attack.

Signature Classifier. The Traffic Analyzer uses signature-based classification to infer the type of attack occurring from unexpected behavior found during the multiclass classification stage. First, the signature classifier stage will determine if the threshold of malicious activity is reached within a given window timeframe to begin classification. If this is the case, the classifier will refer to the Signature Model, a set of rules that define the behaviors that make up the PoisonIvy attacks. Ivycide will then determine the type of attack that occurred in ongoing traffic using the Signature Classifier. As such, it is possible to configure Ivycide to classify for future attacks using additional rulesets. For the purpose of this chapter, we only consider PoisonIvy driver-based attacks.

Evaluation Logger.

The Evaluation Logger receives the evaluation results and any relevant packet data from the Traffic Handler. As such, the Evaluation Logger acts as a middle stage between the evaluation and the data logs, caching and formatting data into a database compatible format. Essentially, this module allows Ivycide users to view prior warnings, see ongoing network communication, and review activity that was deemed to be suspicious.

5.7.7 Model Container

Ivycide's model container stores the classification model for the E-IoT system. The model container uses several packet attributes as the features to classify E-IoT network activity and is divided into two sub-modules, the ML model and the signature-based model.

ML Model.

The ML model is one of the core sub-components used by the traffic analyzer for multiclass classification. The ML model should be learned from an active E-IoT system, or generated from expected network traffic. As such, the ML model includes several common features in IP communication (e.g., packet size, TCP flags, UDP flags, TCP Source/Destination port). Additionally, Ivycide includes two custom features described as follows.

- **Packet Rate.** The number packets an IP source communicates with an IP destination within a 0.1 second time window before and after the given packet. For instance, if the E-IoT controller requests information from a Spotify service at time `t`, the frequency value will show the number packets were sent from the E-IoT controller to the Spotify servers were from time `t-0.1 seconds to t+0.1 seconds`.
- **External Origin.** A boolean value is set to true if the packet originates from an external source to the E-IoT controller.

Signature-based model.

Ivycide uses a signature-based model to infer the type of attack occurring from traffic classified in the multiclass classification stage. The signature-based model contains the signatures of each attack. For instance, the signature-based model would dictate that an activation command, followed by a large number of unauthorized requests to an external target address is likely to be a PoisonIvy DoS attack. While PoisonIvy attacks are the focus of this chapter, there may be many more attacks in the future, as new attacks become known, the signature model can be updated to include new attacks. As such, the Signature Model should be flexible, and easily expandable to include current and future attack signatures.

5.7.8 User Notification

The User Notification module is used to notify a user or the network administrator on warnings and messages from Ivycide. After traffic is analyzed, the Notification Module will detail the traffic logs and give the user all the information necessary to evaluate a possible breach of security. Additionally, the user should receive a notification (e.g., mobile notification) that suspicious activity is occurring in the controller so that they may take action and prevent further issues.

5.7.9 Logged Activities

The Logged Activities module acts as a storage database for any information found during Ivycide monitoring. The administrator queries the logged activities module and can view the suspicious packets and activity detected by Ivycide. Logged Activities only includes packets and data deemed to be of interest to the administrator as well as Ivycide's evaluation of the traffic data stored. This module acts as the final stage of Ivycide and acts as a point of reference for any network administrator that needs to review logged information and prior events.

Table 5.3: Hardware and Software used in Ivycide implementation and testing.

Hardware and Software	
Hardware	Hak5 Plunder Bug
	Acer GX-785 Desktop
Software	Wireshark 3.4.3
	Python 3.9
	Python Scikit-learn
	Visual Studio Code 1.55.2
	Control4 Composer 2.10.6
	Python Scapy
	JupyterLab 3.0.12

5.8 Ivycide Implementation

To implement Ivycide's necessary modules, we used open source, freely-available software and libraries. We detail software and hardware used for Ivycide in Table 5.3. Our testing environment is identical to the PoisonIvy attacks implementation, with the addition of the Hak5 Plunder bug as an active network sniffer between the E-IoT controller and the network router. We assume that the attacker executes the PoisonIvy attacks in the same manner as discussed in Section 5.5, receiving the attack initiation commands from the remote command server and executing the attacks on the local E-IoT system.

5.8.1 Network Collector Implementation

The implementation of the Network Collector required the use of the Hak5 Plunder Bug, Wireshark, and Python scripts to process incoming network data. For the Network Collector, the Hak5 Plunder Bug was placed between the E-IoT controller and the network router. The Plunder Bug was then connected the Acer GX-785 desktop for data collection. Data was collected using Wireshark and then pre-processed using `Scapy`, a Python-based library used to manipulate and extract data from Wireshark

.pcap files. This data was then passed through our pre-processing software and exported as a comma-delimited string that extracted all of each packet's relevant data (e.g., TCP/UDP ports, TCP/UDP flags, timestamp, source/destination IP, packet size). Additionally, our software added additional attributes.

5.8.2 Traffic Handler Implementation

The Traffic Handler and both sub-modules were implemented using Python with JupyterLab and Visual Studio Code.

Traffic Analyzer.

The Traffic Analyzer was implemented using JupyterLab and the Python Scikit-learn library used for ML applications. The Traffic Analyzer receives traffic data formatted by the Network Collector and performs classification on each individual packet using KNN, Decision Tree and Random Forest classifiers using the ML Model sub-module. Packets are tagged by the Traffic Handler as four distinct types of network activity (e.g., benign, unexpected external request, activation, or malicious CnC request) as highlighted in Section 5.7. Packets marked as one of the three types of malicious behaviours are sent sequentially to the signature-based classification stage of the Traffic Analyzer. In the signature-classification stage, the Traffic analyzer refers to the Signature-Based Model for attack classification. All classified packets within the cache window are converted into a string. For instance, two activation packets and seven unauthorized request packets translate to the string 'aauuuuuuu'. If this string is beyond a threshold (e.g., seven malicious packets per window) and falls under the known signatures of attacks in the signature-based model, the activity is classified as one of the three well-known PoisonIvy attacks.

This classification is then passed to the Evaluation Logger. We chose a three minute cache window as PoisonIvy attacks take less than three minutes to execute. Reducing the speed of the attack network throughput (e.g., less packets per second for a remote DoS attack) would greatly reduce an attack's effectiveness.

Evaluation Logger Implementation

The Evaluation Logger was implemented as a Python-based console notification that provides the classification of E-IoT traffic data.

5.8.3 Model Container Implementation

The Ivycide Model Container contains two models used for classification purposes: the *ML Model* and the *Signature-Based Model*. In this subsection we overview both models and the data collection process used to implement these models.

ML Model Implementation

The ML Model was stored as a Python object in Jupyterlabs as a list of fitted models. Each model was then queried by the program to process each incoming packet sequentially. For the implementation we evaluated several classifiers including: Nearest Neighbors, Decision Tree, and Random Forest classifiers. We chose the Decision Tree classifier for the final implementation as it provided adequate classification accuracy and precision for the purposes of Ivycide. For better classification, we also introduced the following features:

- **Frequency.** The frequency was calculated using a sliding window during data collection. Essentially, Ivycide stores packets for a given time window (100 ms), then calculates how many packets share the same source and destination address within the time window.

- **External Origin.** The external origin feature was created by comparing the known IP address of the E-IoT controller and setting this flag to ’True’ if the E-IoT controller was the destination of the packet.

Signature-Based Model Implementation

The Signature-Based model was implemented as a set of rules in our custom Python software. We apply these rules to the signature string generated from the first stage of classification. We highlight the signature rule table as follows:

- **Benign.** A set of traffic data is classified as Benign if the attack pattern does not contain activation commands, indicating no attack was initiated and malicious communication was infrequent.
- **DoS.** A set of traffic data is classified as DoS if the final commands (last five) in attack pattern are classified under “activation”. This behavior indicates that the E-IoT controller became unavailable after a an activation command was received from the attacker.
- **Remote Control.** Traffic data is classified as Remote Control if there is a high frequency of packets classified as UERs in the Multiclass stage. This signature string indicates that the E-IoT controller is making multiple UERs in a short timeframe.
- **Malicious Resource Farming.** A set of traffic data is classified as Malicious Resource farming if CnC requests are observed after activation without unauthorized external requests. This behavior indicates an attack was activated, however, the E-IoT controller is still functional after attack activation.

Data Collection

To train Ivycide, we collected daily usage data from the E-IoT environment by performing expected operation with the E-IoT system as defined in Section 5.7. Expected operation involved the use of the E-IoT environment for streaming media, volume control, menu navigation, and any use consistent with an expected smart system usage. Benign data was collected from the E-IoT environment over the span of two weeks, where the system was allowed to idle, turn on, turn off, and otherwise operate in a manner consistent with expected operation. Malicious data was captured as detailed in the PoisonIvy attacks. In total, we collected 525,705 packets from the E-IoT system for testing and training. To train the model, we followed a supervised learning approach, requiring labeled data for the training. We found that it was necessary to use supervised learning to properly categorize the four types of network activity from the E-IoT controller and evaluate Ivycide.

5.8.4 Other Implementations

The User Notification module was implemented using the Python `ctypes` library to create a notification on the machine running the core Ivycide software. The Logged Activities module was implemented as direct text file exports on the local machine, allowing for future reference of the attack logs and providing any relevant information of the Ivycide analysis.

5.9 Performance Evaluation

In this section, we evaluate the performance of Ivycide in detecting PoisonIvy attacks. Specifically, we attempt to answer the following research questions:

RQ1: Malicious Behavior Evaluation. How do different ML classification algorithms perform in detecting malicious activity based on individual network packets? (Sub-section 5.9.2)

RQ2: Malicious Activity Type. How effective is Ivycide in classifying between different PoisonIvy attacks with signature-based detection? (Sub-section 5.9.3)

5.9.1 Attack Data Collection

Based on the previously mentioned PoisonIvy attacks, we performed the attacks as specified in Section 5.5. To evaluate the Ivycide data classification, we collected communication packets involving the E-IoT controller. The activities collected for our evaluation included expected and unexpected traffic, as defined in Section 5.7. The collection resulted in a total of 60 datasets of attack data, 20 for each attack. Additionally, we recorded 20 datasets of expected network traffic from the E-IoT system as defined in Section 5.7 for a total of 80 datasets. All of the PoisonIvy attacks were executed as highlighted in Section 5.5. For AT_1, we performed a remote activation of DoS attack, disabling the E-IoT controller. For AT_2, our attacker issued a CnC request to the malicious driver to perform unauthorized requests to a target webserver. Finally, for AT_3, we issued a CnC request for the E-IoT controller to begin performing resource-intensive calculations on behalf of the attacker.

Performance Metrics

Performance metrics in this chapter follow the accepted parameters: accuracy, precision, F-score, recall, True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), and False Negative Rate (FNR).

True Positive Rate (TPR). denotes the total number of correctly identified benign traffic within the test environment.

True Negative Rate (TNR). denotes the total number of correctly identified malicious traffic within the test environment.

False Positive Rate (FPR). denotes the total number of cases where malicious traffic was mistaken as being benign.

False Negative Rate (FNR). denotes the total number of cases where benign traffic is mistaken as malicious.

$$RecallRate = \frac{TNR}{TNR + FPR}, \tag{5.2}$$

$$PrecisionRate = \frac{TPR}{TPR + FPR}, \tag{5.3}$$

$$Accuracy = \frac{TPR + TNR}{TPR + TNR + FPR + FNR}, \tag{5.4}$$

$$F1 = \frac{2 * RecallRate * PrecisionRate}{RecallRate + PrecisionRate}. \tag{5.5}$$

5.9.2 Ivycide Performance for Different Classifiers (RQ1)

As part of *RQ1*, we evaluate different ML-based classifiers and their performance on individual network traffic packets. As highlighted in Section 5.7, Ivycide may use the most effective classifier to classify between behavior types. For *RQ1* Ivycide evaluation we implemented several classifiers, highlighting *Decision Tree*, *Nearest Neighbors*, and *Random Forest* with the best performance. We refer to Table 5.4 for the performance of each classifier used with Ivycide. In these results we show how different classifiers perform against E-IoT network traffic in terms of accuracy

Table 5.4: Multiclass classification of malicious E-IoT traffic behaviors.

Legend : DT = Decision Tree, KNN = Nearest Neighbors, RF = Random Forest

BEN = Benign, CnC = Command & Control, ACT = Activation, UER = Unauth. External Request

Model	Class	TPR	TNR	FPR	FNR	ACC	PREC	REC	F1
DT	BEN	0.98	0.91	0.08	0.02	0.96	0.97	0.98	0.98
	CnC	0.92	0.97	0.01	0.09	0.98	0.97	0.92	0.94
	ACT	0.99	0.96	0.00	0.00	1.00	1.00	1.00	1.00
	UER	0.84	0.97	0.01	0.18	0.99	0.82	0.85	0.83
KNN	BEN	0.98	0.85	0.14	0.01	0.96	0.95	0.99	0.97
	CnC	0.84	0.98	0.00	0.19	0.96	0.98	0.84	0.91
	ACT	0.94	0.95	0.00	0.06	1.00	0.96	0.94	0.95
	UER	0.92	0.96	0.00	0.09	0.99	0.90	0.92	0.91
RF	BEN	0.99	0.92	0.08	0.01	0.97	0.97	0.99	0.98
	CnC	0.93	0.98	0.01	0.07	0.98	0.95	0.94	0.95
	ACT	0.99	0.97	0.00	0.01	1.00	0.99	0.99	0.99
	UER	0.85	0.98	0.01	0.17	0.99	0.87	0.85	0.86

and precision. For all of the covered classifiers, we observed accuracy and precision rates averaging higher than 90%.

UER (Unauthorized External Requests) were particularly challenging to classify. In most cases, UER was misclassified as CnC attacks. This is possibly due to the fact that the internal programming functions to perform UER requests in the attack code are identical to CnC attacks. The Ivycide architecture (Section 5.7) highlights that the multiclass classification is the first step for Ivycide. We note that perfect classification accuracy on individual packets is not required for effective signature classification because attack signatures have some matching tolerance given the rulesets. Further, the configurable design of Ivycide, means that evaluating different classifiers yields to valuable information. For instance, some classifiers may have more success at classification on some E-IoT deployments and configuration than others. As such, since E-IoT systems are highly heterogeneous, Ivycide can be adapted with one or multiple classifiers to provide better accuracy and precision for individual deployments.

Table 5.5: Signature-based classification of malicious E-IoT attacks.

Legend : BEN = Benign, BOT = Botnet, DOS = Denial-of-Service, MRF = Resource Farming

Class	TPR	TNR	FPR	FNR	ACC	PREC	REC	F1
BEN	1.0	1.0	0.00	0.00	1.0	1.0	1.0	1.0
BOT	1.0	0.98	0.03	0.00	0.98	0.91	1.0	0.95
DOS	0.75	1.00	0.00	0.25	0.94	1.00	0.75	0.86
MRF	0.95	0.95	0.05	0.05	0.95	0.86	0.95	0.90

5.9.3 Ivycide Signature Classification Performance (RQ2)

We refer to Table 5.5 for Ivycide's signature-based classification performance in terms of accuracy, precision, recall, and F1 metrics for each attack type. As such we note that Ivycide achieved an overall accuracy of 97% and precision of 94%. More notable, is that no malicious cases were classified as benign, as such, even if an attack is misclassified as another attack, the administrator will still be alerted of suspicious traffic. Specifically, we found that three DoS attacks were misclassified as malicious resource farming attacks. This may be due to both PoisonIvy DoS (memory exhaustion on the controller) and resource farming attacks low network throughput.

In some cases, we found that the music streaming service TuneIn, caused false positives. Ivycide improperly classified some benign data from the streaming service as unauthorized requests. We believe that the addition of whitelisting to approved IP addresses may further improve the accuracy of Ivycide since attackers cannot spoof addresses using the driver API. However, even without whitelisting, the number of unauthorized requests in our proof-of-concept attacks were limited as legal limitations with the target Amazon Web Services hosted website do not allow for DDoS attacks. Specifically, Amazon Web Services explicitly prohibits any type of DDoS testing what would put any stress on their servers. Traffic-based DoS attacks performed by attackers without legal concerns would create many more observable unauthorized requests within a given timeframe from an E-IoT controller and, as such, become easier to identify using Ivycide.

We note that the classification performance was accomplished using only *black-box integration and with no modification to the E-IoT controller, drivers, or system code.* While some attacks were misclassified as other attacks, all malicious instances of attacks were detected as suspicious activities. Similarly, benign activity was properly classified in all cases, greatly reducing the number of false alarms by the signature-based classification. As such, in any system implementation, network administrators would have been alerted for all attacks, been able to investigate attacks further, and take action against an infected E-IoT controller.

5.9.4 Detection Time and Overhead

How quickly attacks are detected is dependent on the attacks and the attack payload through the network. For instance, a remote control (DoS) attack is much more noticeable in network traffic than a DoS attack. As such, the maximum time it would take for an attack to be detected is the time window given for Ivycide. We measured the CPU usage and memory consumption of Ivycide for each stage with 4.5% CPU usage and 11.6 MB of peak usage for the multiclass classification stage; and a 0.3% CPU and 19.2 MB peak usage for the signature classification stage. We must note that this overhead is only applied to the computer running the Ivycide system (16 GB RAM and i7-700 3.6 GHz) and not to the E-IoT system controller.

5.10 Ivycide Benefits and Discussion

Ivycide is designed as a defense solution for E-IoT. In this section, we highlight Ivicyde's benefits and further discuss their implications.

Passive Monitoring. Ivycide is based on passive network monitoring. This has two advantages. First, Ivycide will not affect incoming or outgoing network traffic, as such, the quality-of-service of the E-IoT system will not be affected. Second, with passive monitoring, no changes need to be made to the original E-IoT system, a process that may not be viable in some older systems. As legacy systems may be too costly or impractical to replace, Ivycide can provide an alternative that will evolve with newer threats.

Black-box integration. Ivycide solves one of the biggest issues with E-IoT systems, lack of available source code and technical documentation. Ivycide does not require knowledge of more technical aspects of an E-IoT system. As such, Ivycide can be trained from a live analysis of an E-IoT system at the configuration phase that uses known certified drivers. Alternatively, if drivers and communication of the E-IoT system are known, models can be reused and configured for an E-IoT system without the need to retrain Ivycide for every E-IoT system.

Flexible Design. It is possible to adjust Ivycide for any E-IoT deployment. This is a necessity as most E-IoT systems are custom-built, a custom-built solution to novel threats is needed. As such, the design of Ivycide allows to save and load custom models depending on the components integrated into the E-IoT system and expected operation as highlighted in Section 5.7. Alternatively, it may be possible for Ivycide to ignore traffic originating from known vendor services and approved addresses (e.g., whitelisting). These actions could reduce the number of false positives and reduce the processing requirements for Ivycide in larger systems.

Independent IDS. The E-IoT controller and Ivycide are independent systems. As such, in the case of controller failure (e.g., malicious DoS) there will be no effect on the Ivycide system. Additionally, this means that no overhead is added to the E-IoT controller or to E-IoT operations.

Model Re-usability. Ivycide relies on the ML model for accuracy and detection, it may not be necessary to retrain the model for every E-IoT system. First, if two E-IoT systems are identical deployments (e.g., similar conference rooms in a building as separate E-IoT systems), the same model will work for both systems. Further, Ivycide is affected by drivers with networking capabilities. If non-IP drivers between E-IoT systems are different, this does not affect the model used by Ivycide. Second, devices may behave similarly. As highlighted by Table 5.2, different device types have similar expected network traffic. As such, if similar devices are added (e.g., replacing a faulty television) the model may not need to be re-trained as the expected traffic is the same. Third, drivers verified can supply the traffic logs of their drivers and that can be added to the model. As such, Ivycide may be able to offer pre-trained models if all of the drivers in the E-IoT system are known and have vendor support.

5.11 Conclusion

Recent years have seen a dramatic rise in IoT systems and applications that enabled billions of commodity IoT devices to empower smarter settings in buildings, offices, and homes. Although commodity IoT devices are employed by ordinary end-users in small-scale environments, more reliable, complex, customized, and robust solutions are required for enterprise customers. Those solutions called E-IoT are offered by dedicated vendors. With the higher price, customization, robustness, and scalability of E-IoT systems, they are commonly found in settings such as smart buildings, government or private smart offices, academic conference rooms, luxury smart homes, and hospitality applications. As E-IoT systems require specialized training, software, and equipment to deploy, many of these systems are closed-source and proprietary in nature. This has led to very little research investigating the security of E-IoT systems and their components. In effect, E-IoT systems in professional smart settings (e.g., smart buildings) present an unprecedented and unexplored threat vector for an attacker. In this chapter, we explored E-IoT system vulnerabilities and insecure development practices, specifically, the usage of drivers as an attack mechanism. We implemented an E-IoT system testbed in a smart building setting and introduced PoisonIvy, a novel attack mechanism to show that it is possible for a malicious actor to easily attack and command E-IoT system controllers using malicious drivers. Specifically, with PoisonIvy, an attacker may cause DoS conditions, take control of E-IoT system controllers, and remotely abuse the resources of the such systems for illegal activities (e.g., bitcoin mining). With this chapter, we raise awareness on the (in)secure development of the drivers that control E-IoT systems, the consequences of which can largely impact E-IoT systems as a result. Additionally, we discussed the (in)security of these drivers, security implications, and possible counter-measures. To defend against these threats, we introduced Ivycide, a novel, configurable defense mechanism designed specifically for E-IoT systems. As Ivycide operates as a standalone framework it provides no additional overhead to E-IoT systems. Finally, we evaluated the Ivycide performance on a realistic E-IoT system. Our analysis showed that Ivycide achieved 97% in accuracy and 94% precision for attack type identification.

CHAPTER 6

NEW HDMI ATTACKS AND DEFENSES FOR E-IOT SYSTEMS

6.1 Introduction

Audio/Video (A/V) devices have always witnessed a wide range of adoption as consumer electronics. The High Definition Multimedia Interface (HDMI) is used primarily for the distribution of A/V signals and has become the de-facto standard for this purpose [Tsu08]. For instance, in many applications such as concert halls or sporting events, large displays are chained together via HDMI to show concert images and gameplay. Figure 6.1 shows possible use-cases of HDMI distributions. Indeed, as of this writing, there have been close to 10 billion HDMI devices distributed, making HDMI one of the most highly deployed systems worldwide[Wri18]. With the requirement to merge control and communication over a single connection, the HDMI Consumer Electronics Control (CEC) protocol was specified with the release of the HDMI v1.2a [Hol05]. CEC provides control and communication between HDMI devices through HDMI cabling. This has led many vendors to implement CEC features on their devices under different trade names, including: Anynet+ (Samsung), Aquos Link (Sharp), BRAVIA Link/Sync (Sony), CEC (Hitachi), CE-Link and Regza Link (Toshiba), SimpLink (LG), VIERA Link (Panasonic), EasyLink (Philips), Realink (Mitsubishi) [Goo18]. The adoption of CEC has become a means of control for well-known household devices (e.g., Google Chromecast, Apple TV, Sony A/V Receivers, Televisions). This rapid adoption has made CEC into an ubiquitous protocol in many A/V installations and the adoption of CEC enabled devices in conference rooms, homes, offices, government, and secure facilities. Given the popularity and the penetration of HDMI-based devices, their security is of utmost importance.

Nonetheless, CEC is outside the reach of the traditional networking mechanisms, and most importantly, current security mechanisms provide no protection to CEC-based threats. This creates a widely-available, unprotected, and unexplored threat vector in locations (e.g., homes, government, offices) without mainstream user awareness. Unprotected HDMI networks give malicious entities an attractive medium of attack from which they can remain undetected. CEC allows them to perform activities over an HDMI device distribution network such as information gathering, device control, and attack facilitation. In effect, an attacker can retrieve and alter the power state of all HDMI-Capable devices without physical or traditional network access. While there has been abundant research on the security of traditional networks, this protocol has remained an under-researched communication component in the realm of cybersecurity.

As HDMI distributions are non-traditional components of smart network systems, current security mechanisms do not offer any protection against to HDMI-based attacks. Thus, CEC remains as a widely-available, unprotected, and unexplored attack surface without mainstream user awareness. To defend against these threats, we propose HDMI-Watch; a novel passive smart intrusion detection system that protects HDMI distributions against CEC-based attacks. HDMI-Watch operates as a standalone framework in HDMI distributions, passively monitoring CEC traffic for CEC malicious behavior. HDMI-Watch leverages CEC command types and machine learning techniques to detect unexpected activities in CEC communication. Additionally, HDMI-Watch accounts for expected command lengths, associating CEC command types to their acceptable message lengths to improve detection. To test HDMI-Watch performance, we performed an extensive set of evaluations in a realistic HDMI testbed with a variety of consumer HDMI-capable devices and against HDMI-Walk attacks. Our results show that HDMI-Watch per-

(a) Conference room with multiple displays and points of HDMI connection.

(b) Airport information kiosks with multiple displays and HDMI connections.

(c) Concert displays, where used may be HDMI for multiple displays.

(d) Sports bars where multiple displays are shared by a single video source.

Figure 6.1: Possible examples of HDMI distribution use cases where HDMI-Walk could present a novel threat.

formance achieves an average accuracy and precision of 98%, detecting unexpected activities without any form of operational overhead or modification to HDMI devices.

The contributions of this chapter are as follows.

- We introduce HDMI-Walk, a novel attack vector against HDMI distributions to demonstrate that arbitrary control of CEC devices is feasible for an attacker using this method.
- We implemented five unique attacks to HDMI distributions. Specifically, we performed topology inference, DoS attacks, eavesdropping, targeted device attacks, and facilitate existing attacks.
- We demonstrate the threat of HDMI-Walk with a specific testbed of commonly used HDMI equipment (e.g., Google Chromecast and Sharp Smart TV) for the evaluation of HDMI-Walk attacks.
- We propose HDMI-Watch, a novel intrusion detection system that protects HDMI distributions against CEC-based threats in HDMI distributions. HDMI-Watch monitors CEC communication and detects unexpected CEC behavior occurring in an HDMI distribution.

- We evaluate HDMI-Watch in a realistic HDMI testbed with a variety of consumer devices (e.g., Google Chromecast and Sharp Smart TV) achieving an average accuracy and precision of 98%.

6.1.1 Differences from Existing Works.

This chapter differs from other works as follows. We introduce a novel attack method called HDMI-Walk to HDMI devices. Our scope is entirely through CEC as the main vector of attack and does not rely on any custom applications, software vulnerabilities, fuzzing, buffer-overflows, vendor-specific attacks, or traditional network connectivity. We focus on the exploitation of the CEC protocol in both local and remote attacks. We demonstrate proof-of-concept implementations of five different types of attacks; specifically, (1) malicious device Scanning, (2) eavesdropping, (3) facilitation of attacks (e.g,. WPA Handshake theft), (4) information theft, and (5) denial of service through HDMI. Finally, we introduce HDMI-Watch, the first IDS designed specifically for CEC threats over HDMI connections, achieving overall accuracy and precision of 98%

6.2 Problem, Assumptions, and Threat Model

In this section, we present the assumptions, definitions, and the threat model for HDMI-Walk-based attacks.

6.2.1 Problem Scope

This sub-section denotes an HDMI distribution network within a conference room which may be used for confidential presentations. The topology of this distribution network includes common HDMI distribution equipment such as switches, hubs as well as HDMI devices such as displays and sources. The attacker is an invited guest presenter Mallory, who has a small amount of time to prepare in the conference room without any supervision. Mallory either compromises an existing HDMI device through malicious apps, or hides a malicious HDMI-capable device within the distribution (e.g., connected behind a television). We later elaborate compromising devices through malware in further detail. This is realistic, as A/V systems are very rarely inspected by users, and are physically accessible by visitors. Mallory connects her own laptop to auxiliary ports on the podium prior and during the presentation and perpetrates the HDMI-Walk attacks. After presenting, Mallory leaves. Sometime after her departure, further security policies are enacted and unsupervised access to the conference room is disallowed to visitors. Mallory's only avenue of attack is to access her hidden device indirectly, locally or remotely.

Compromising Devices: We note that Mallory may compromise an HDMI distribution without direct access to the HDMI network. Malware (e.g., firmware, app-based) could compromise an existing device to Mallory's benefit, acting as a link between the distribution and their machine. For instance, privileged malware applications in an Android-based A/V device could make use of the `HdmiControlManager` functionality which is available to transmit and receive arbitrary messages [Dev19]. An attacker can therefore compromise a system with a malicious app installed by a user, or by a visitor.

Possible Payloads: CEC attacks can provide access to devices which may have been believed secure or isolated in a conference room. Conference rooms may serve varying purposes from unrestricted to confidential usage. When a space is in unrestricted usage, an attacker may disrupt operation, damage equipment and prevent normal usage of a conference room through CEC attacks. If the space is used for confidential purposes, an attacker may gather data about a system, gather restricted information within a conference room, or simply facilitate more complex attacks (e.g., wireless handshake theft, eavesdropping). In both of these cases, an attacker may avoid traditional means of detection through the use of CEC.

Attack Mode 1 (Local Communication): Mallory only has local access when connecting directly to the HDMI distribution network as a presenter. This case is independent of any form of network access, it relies on Mallory's ability to connect to the auxiliary connection on the conference room podium. Local communication from her laptop through the HDMI distribution with HDMI-Walk and to the hidden or compromised device.

Attack Mode 2 (Remote Communication): In this case, Mallory has found an open guest network connection during her first visit or later gained unauthorized internet access. This allows Mallory to enable remote access to her hidden device. Furthermore, this allows Mallory to perform specific attacks.

6.2.2 Definitions

In this sub-section we denote definitions for concepts used in this section.

Definition 1 - Isolated Device: An HDMI device which has no network connectivity to traditional IP networks in any manner.

Definition 2 - Limited Access User: A limited-access user is primarily described as a user with temporary physical access to a location and limited IP network connectivity. This user can be a temporary visitor such as a presenter.

Definition 3 - Attacker (Temporary Visitors): An attacker is any limited-access user which attempts malicious access to unauthorized resources. The attacker's motivations are to disrupt, gather information, gain unauthorized access, learn user behavior, and perpetrate the attacks listed in the threat model below. In our case, the attacker may be a temporary visitor with limited access to the facilities (e.g., a presenter, Mallory).

6.2.3 Assumptions

To perform the HDMI-Walk attacks, we have the following assumptions.

CEC Propagation: HDMI-Walk assumes full CEC protocol propagation over the distribution of HDMI devices. Some devices tested had no function to disable CEC propagation, even if CEC control was disabled. In testing performed on devices with multiple HDMI ports, we found 80% of devices provided some form of propagation.

CEC Control: We assume CEC control is active on connected devices in the distribution. This is a realistic scenario, as we found that in all CEC-capable devices tested, CEC functionality was enabled by default. We also observed that many devices revert to default settings after a firmware update.

Access to HDMI Components: We also assume that Mallory has access to some HDMI components (or endpoints) in the distribution. This is a realistic assumption as A/V components are often not as secure as networking components. Display inputs and outputs are often visible and available to presenters. Presenters are often given enough time to prepare and free access to A/V equipment in a conference room without supervision or suspicion is expected. In some cases, we have found displays (often used for information purposes) outside conference rooms which could act as another connection point to an HDMI distribution inside a conference room.

6.2.4 Threat Model

HDMI-Walk assumes the following five threats as part of the threat model.

Threat 1: Malicious CEC Scanning: This threat considers the malicious use of *scanning* features through CEC and exposed HDMI ports to gather information about the connected devices. For instance, Mallory can create a topology of available HDMI devices to control and use this information to perform further attacks.

Threat 2: Eavesdropping: In this threat, Mallory is not present but actively eavesdrops on CEC communication through an implanted device.

Threat 3: Facilitation of attacks: This threat eliminates time and physical access limitations in wired and wireless attacks. HDMI-Walk facilitates many of these attacks so that they become more viable or more difficult to detect. For example, Mallory installs a device to passively capture WPA handshakes, avoid detection, and control through CEC remotely.

Threat 4: Information Theft: This threat considers information theft as a form of data transfer which Mallory may find valuable. For example, information about available HDMI devices or wireless handshake capture which would enable future attacks.

Threat 5: Denial of Service: This threat considers DoS attacks where Mallory disrupts the availability of a system through an HDMI connection. These attacks may be targeted to a specific device or broadcast to multiple devices. For example, Mallory prevents the use of a television through the repeated broadcast of HDMI control commands.

Note that HDMI-Walk does not consider attacks which focus entirely on IP networks; data injection attacks through CEC such as buffer overflows over CEC or setting manipulation attacks. Similarly, other protocols such as USB or Bluetooth are entirely outside the scope of this chapter.

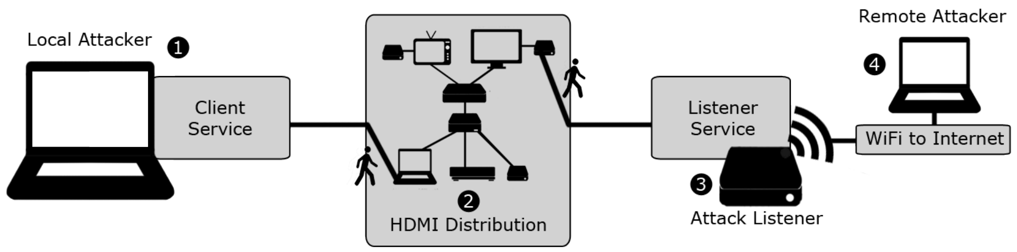


Figure 6.2: General architecture for HDMI-Walk-based attacks.

6.3 HDMI-Walk

In this section, we present the details of the HDMI-Walk based attacks. Figure 6.2 depicts the general architecture of HDMI-Walk which comes with four main components: *local attacker, HDMI Distribution, attack listener, and remote attacker.*

The first component of HDMI-Walk is the Local Attacker which runs the Client Service in their local machine. This local hardware is temporarily connected to the HDMI distribution. The client service contains any required modules for communication to the listener and facilitates the attacks through HDMI-Walk (❶). The second part is the HDMI Distribution, which is the core of our attacks and allows for end-to-end communication between devices through HDMI as a medium. The user may scan the distribution for addressed CEC devices, as well as communicate bidirectionally with other devices (❷). The third part of the architecture involves the Attack Listener. The attack listener is the physical attacker device and hosts the Listener Service. The listener service includes all the required modules for HDMI-Walk communication and listener-run attacks. This service also includes a remote access module to enable communication to the remote client if a connection is available (❸). Finally, we have the Remote Attacker, which communicates directly through a remote connection to the attack listener if remote access is possible (❹).

Local Attacker: A local attacker establishes communication with the listener device through CEC and the HDMI Distribution. The local attacker places their client device in an exposed HDMI port such as an auxiliary connection in a presentation room or a side input of a television. In our case, the client device can be a laptop with a CEC capable adapter. The client's main purpose is to establish communication with the listener and serves as the main interface for an attacker to issue commands and receive data from the listener device. The local client communicates to the listener through HDMI-Walk derived control of the distribution. Additionally, the client device hosts the client service. This service contains all necessary software modules for specific actions within the scope of CEC such as the ability for file transfer, arbitrary CEC communication, and CEC scanning.

HDMI Distribution: This allows for the core concepts of this chapter is the nature of the CEC protocol which allows propagation and control. These are not inherently equal or mutually exclusive; for instance, a device may be able to both control and propagate CEC commands through auxiliary HDMI ports. In contrast, a different device, such as an HDMI hub, may allow propagation but offer no CEC based control. The inherent design of CEC allows for any device to transmit and request information to and from any other device within the same distribution. During our evaluations, we found that CEC commands propagate from device to device, passing through different 'hops' in a similar fashion to a bus network while allowing individual devices to further propagate communication to their own branched connections. This is a requirement in 'scanning' behavior, which allows for any device to query others by logical address for a name, type, language, OSD string, vendor, power status, CEC version, and source status. With this, the querying device is able to build a map of available CEC devices within the distribution. Since the headers signify a broadcast or a message to a specific device by logical address, this becomes useful for targeting specific devices or broadcasting to all devices.

Attacker Listener: The listener device awaits client commands. Ideally, the listener is hidden by the attacker in a location such as behind a television, in an equipment cabinet or anywhere where there is a connection to the HDMI distribution. The listener may establish communication with CEC-enabled isolated devices (see Section IV) through HDMI-Walk. In the attack model, once the listener receives expected commands from the attacker client (local or remote), it will enact actions in the HDMI distribution on behalf of the attacker. In our proposed HDMI-Walk, the attack listener performs the core actions for our attacks and runs all the separate modules required for each attack. Additionally, the listener hosts all the software modules required by the attacker for CEC communication, CEC file transfer, CEC scanning, microphone access, wireless access, and remote access.

Remote Attacker: The remote attacker maintains a remote web interface to a listener device. Commands and messages are relayed bidirectionally from the listener and the remote attacker. In contrast to the local attacker, the remote attacker operates in a remote web server, has no direct CEC connectivity and only hosts remote communication modules. The remote attacker's server is polled via an Internet connection by the listener for new commands. This allows the attacker to perform remote execution of CEC actions using the listener. These actions may involve CEC information gathering, targeted attacks, DoS, or any attack module within the listener device.

Table 6.1: Hardware and software used in HDMI-Walk.

Hardware	Software
Sharp Smart TV.	Pulse Eight LibCEC 4.0.2
Samsung UN26EH4000F	Python 3.6.1
Monoprice Blackbird 3x1 HDMI Switch	Aircrack-ng 1.2-rc4
Wyrestorm - 1x4 HDMI 1.3b Splitter	Eclipse IDE
Chromecast NC2-6A5	PyAudio v0.2.11
Sony STR-ZA2100ES	Jersey JAX-RS
Raspberry 3 Model B x2	Raspbian Version 9
TP-Link TL-WN722N V1 Adapter	Swagger.io
Motorola G5 Plus Phone	Java 1.8
TP-Link TL-WR841N Router	AWS Elastic Beanstalk

6.4 Evaluation and Realization of HDMI-Walk Attacks

In this section, we describe and evaluate the HDMI-Walk attacks in detail. The purpose of the attacker is to leverage the HDMI-Walk capabilities to discover, manipulate, control, and cause undesired operation to devices within an HDMI distribution. The adversary also aims to use CEC as the primary medium for their attacks. This is achieved via a connection to the listener through a remote client or through a local HDMI connection. As explained in earlier in Sections IV and V, in all of the attacks, the attacker plants the listener device somewhere within the CEC distribution (e.g., behind a television).

6.4.1 The implementation of HDMI-Walk

In order to ensure the attacks are implemented in a realistic HDMI environment, we created a CEC capable testbed with standard and widely available commodity HDMI devices presented in Table 6.1. Here we included two displays, an HDMI switcher, an HDMI hub, a source and the attacker devices as depicted in Figure 6.3. We utilized LibCEC, an open-source CEC implementation [Pul18]. This library provides Python modules which we used to create both the client and the listener services. Due to readily available CEC support in Raspberry Pi v3 devices, we used

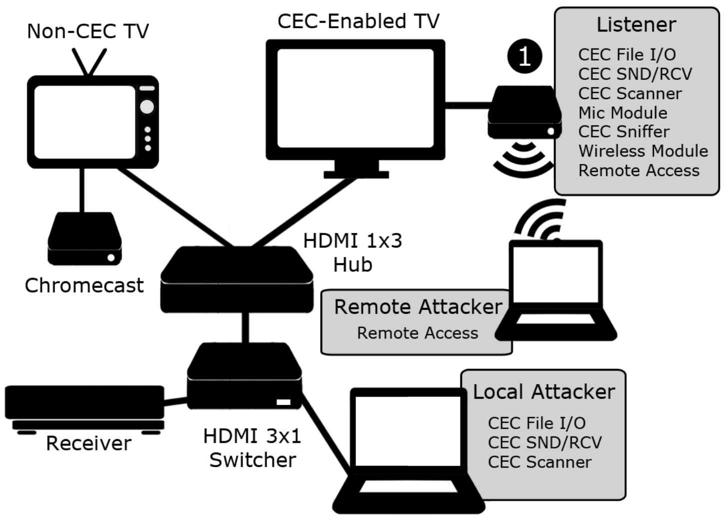


Figure 6.3: HDMI-Walk testbed implemented with various commodity HDMI devices.

two Pis, one as the listener and one as the local client to perform the attacks and evaluations. To test WiFi (handshake) and remote attacks, we created a network with SSID `Portabox`. Note that even with a non-CEC-addressed TV, the TV simply propagates any CEC commands through additional ports.

6.4.2 Software Modules

CEC File I/O

This module facilitates sensitive data transfer within the CEC protocol. We leveraged HDMI-Walk for data transfer between client and listener devices within the CEC distribution. This module can be subdivided into three sections: serialization, transmission, and deserialization. We break down a file transfer from the listener (sender) to the attacker (receiver) below:

- *Serialization:* LibCEC allows the transfer of CEC packets through the distribution. The attacker device (hosting client service) first begins the file request with the “aa:aa:aa:aa” packet. The data is then imported into the running service and converted into hexadecimal values. This serialized file is stored locally within the buffer of the current sender (i.e., the attack listener).

- *Transmission:* The buffer is segmented into hexadecimal strings of length 28 in preparation for the file transfer by the sender. Each segment of the buffer is sent with the data header “xx:00” over CEC to the receiver. Finally, once all segments are exhausted, the transmission ends with “ee:ee:ee:ee”. Any packet received without these headers are dismissed by the receiving device.
- *Deserialization:* packets are received in order by the receiver (i.e., the attacker laptop), cropped and then stored locally in a clean data buffer. With transmission finished, the client now deserializes the stored buffer into the original file.

CEC SND/RCV

This module sends and receives custom CEC messages through the alteration of the header (destination) and data blocks. These may be used to activate listener conditions, attacks, or request a file transfer from the listener. We achieve sending and receiving of custom CEC messages through the use of the libCEC Python module. This library provides the communication method which allows the creation and transmission of CEC commands within specifications. This function is used as part of File I/O transfer or to transmit specific commands to a device over the HDMI distribution.

CEC Scanner

This module scans a distribution to identify CEC devices. The CEC scanner implements the standard LibCEC `scan` command which queries all possible devices within a distribution and records all valid responses. HDMI-Walk captures the available devices and provides the attacker information on each device and the logical address of their listener.

Microphone Module

Used to record and store anything captured by the embedded microphone on the listener device for the purpose of audio eavesdropping. Microphone access and recording were achieved through the use of the PyAudio library. PyAudio allows local storage of audio data within a Python operation at pre-determined length and bandwidths. We created this module to activate the microphone in the listener device.

CEC Sniffer

CEC sniffer allows the listener to passively monitor all the commands and data passing through the CEC distribution. Targeted attacks may use this feature to trigger commands upon the action of a device. This is implemented through the command callback in the LibCEC library which allowed us to handle any command received through the bus. We analyze every packet for specific calls during the attack phases. With this form of detection, attacks may target specific devices based on their power state change.

Wireless Module

The wireless module comes in two parts. The first part provides standard wireless access or the capability to connect to an Internet-enabled network for remote support to the attacker. The second part implements Aircrack-ng to allow for sniffing, capture, and final cleaning of WPA/WPA2 handshakes for further cracking. We use

Table 6.2: Module utilization per attack

Legend : ◓ = Local Attack, ◒ = Remote Attack, ○ = Neither, ● = Both

	Topol. Infer.	CEC Eaves.	Handshk. Theft	Targeted Attack	Broadcast DoS
File I/O	◒	◒	◒	○	○
SND/RCV	●	◒	◒	●	●
Scanner	●	◒	◒	●	◒
Mic Mdl.	○	◒	○	○	○
CEC Sniffer	○	○	○	●	○
Wireless Mdl.	○	○	◒	○	○
Remote Mdl.	◓	○	○	◓	◓

a `monitor mode` capable adapter with this module (TP-Link) and Python calls to automate the process in the target WiFi network.

Remote Access Module

The remote access module is utilized to allow for remote requests to the listener device through a valid Internet connection. It is divided into two parts: Server Component and the Listener Web Component.

Server Component: We hosted a RESTful API running Swagger GUI as the remote client and server component within AWS' Elastic Beanstalk service. We create two string caches reachable with the paths `/cec/listener` and `/cec/webclient` each with GET and POST methods. The attacker accesses this server component and submits their commands through `POST: /cec/listener` with a JSON object containing the desired command to execute remotely.

Listener Web Component: We implemented the web component using Python threading and polling requests to our server. The listener polls `GET: /cec/listener` every two seconds for new commands submitted for remote execution. This listener component posts to `POST: /cec/webclient` for later retrieval by the attacker.

6.4.3 Attacks

In this sub-section, we realize HDMI-Walk attacks and discuss its implications. We also present individual uses of every module aforementioned for HDMI-Walk attacks in Table 6.2.

Attack 1: Topology Inference Attack (Local and Remote) This attack is a demonstration of Threat 1 (Malicious CEC Scanning) possible through CEC in online and offline scenarios. We use the HDMI-Walk architecture to move through the distribution and gather information about every device available with malicious intent. This attack can be executed through the local or remote client.

Step 1 - Activation: Upon initial placement within the HDMI distribution, the listener automatically connects and begins the information gathering process with remote and local execution of HDMI-Walk scans.

Step 2 - Information Gathering: The listener begins to perform a "walk" over all of the devices using the CEC scanner module. This easily yields information about HDMI device type, device, logical address, physical address, active source, vendor, CEC version, device name, and power status from available devices in the distribution. Once this has been processed, the listener stores the data locally.

Step 3 - Leakage: For a local client, the data is ready to be retrieved through the File I/O module upon local client request. For the remote client, the listener performs a call to `POST: /cec/webclient` with all the captured information. The data is submitted to the remote server in the form of a JSON object to be retrieved by a remote attacker.

Evaluation: With this attack, we used the scanning functionality to "walk" and gather more information on the controllable devices available. The attack was entirely successful and allowed us to learn information both locally and remotely about each accessible device. As seen in Table 6.3, we gather information such as

Table 6.3: Attack 1–Information gathered through HDMI-Walk.

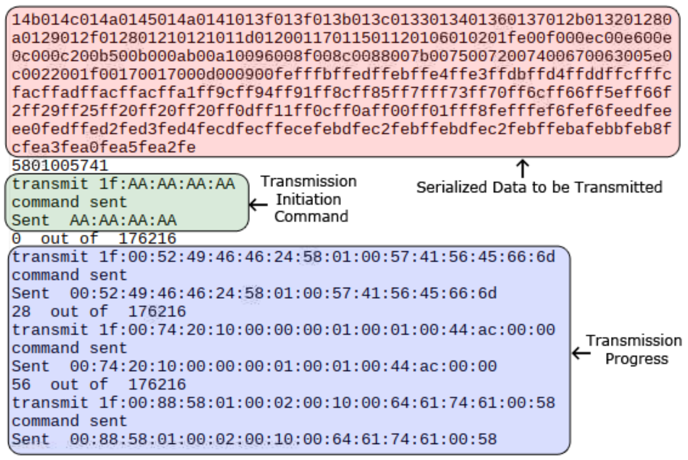
Info	Addr 00	Addr 01	Addr 02	Addr 04	Addr 05
P. Addr	0.0.0.0	f.f.f.f	4.0.0.0	3.0.0.0	1.0.0.0
Active	No	Yes	No	No	No
Vendor	Unk	Unk	Pulse-Eight	Google	Sony
OSD Str	TV	RPI	CECTestr	Chromecast	STR-ZA2100
CEC Ver	1.4	1.3a	1.4	1.4	1.4
Pow Status	ON	ON	ON	ON	Standby
Language	Eng.	Eng.	Eng.	Unk	Unk

the device logical/physical address, active source state, Vendor name, CEC Version, OSD Name, and power status. With this information, an attacker may as well infer usage from the power state of the equipment. For example, an attacker may be able to infer that a room is in use when the power state of the displays is `on` or perform more vendor and device-specific attacks with more research on specific devices.

Attack 2: CEC-Based Eavesdropping (Local) We perform this attack to demonstrate Threat 2 (Eavesdropping) and Threat 4 (Information Theft). In this local attack, an attacker has access only to the HDMI port for communication with the listener device. The attacker walks the HDMI distribution and forwards messages to the listener to activate and record audio using the Microphone Access Module. This audio data is stored locally in the listener device. The audio data is then transferred to the client at a later date through the use of the File I/O module.

Step 1 - Listener: The attacker first places a listener device in the CEC distribution as noted by the architecture. The listener device awaits attacker commands from another location in the HDMI distribution.

Step 2 - Listener Activation: The attacker sends the request to performs an HDMI-walk to scan the devices and identifies the listener device in the CEC distribution. We note the logical address of the listener device and activate the Microphone module with "bb:bb:bb:bb" command received by the listener. The listener device records audio and stores the data locally.



```
14b014c014a0145014a0141013f013f013b013c0133013401360137012b013201280
a0129012f012801210121011d01200117011501120106010201fe00f000ec00e600e
0c000c200b500b000ab00a10096008f008c0088007b007500720074006700630 05e0
c0022001f00170017000d000900fefffbffedffebffe4ffe3ffdbffd4ffddffcfffc
facffadffacffacffa1ff9cff94ff91ff8cff85ff7fff73ff70ff6cff66ff5eff66f
2ff29ff25ff20ff20ff20ff0dff11ff0cff0aff00ff01fff8fefffef6fef6feedfee
ee0fedffed2fed3fed4fecdfecffecefebdfec2febffebdfec2febffebafebbfeb8f
cfea3fea0fea5fea2fe
5801005741
transmit 1f:AA:AA:AA:AA
command sent
Sent  AA:AA:AA:AA
0  out of  176216
transmit 1f:00:52:49:46:46:24:58:01:00:57:41:56:45:66:6d
command sent
Sent  00:52:49:46:46:24:58:01:00:57:41:56:45:66:6d
28  out of  176216
transmit 1f:00:74:20:10:00:00:00:01:00:01:00:44:ac:00:00
command sent
Sent  00:74:20:10:00:00:00:01:00:01:00:44:ac:00:00
56  out of  176216
transmit 1f:00:88:58:01:00:02:00:10:00:64:61:74:61:00:58
command sent
Sent  00:88:58:01:00:02:00:10:00:64:61:74:61:00:58
```

Transmission Initiation Command
Serialized Data to be Transmitted
Transmission Progress

Figure 6.4: Attack 2–File I/O Module transfer of audio data.

Step 3 - Client Request: The client requests a file transfer using the File I/O module and the command “aa:aa:aa:aa” to the listener. The listener receives this command via the CEC distribution and serializes the stored audio data as the client awaits the data transfer. Once the audio file is serialized the File I/O module transmission begins.

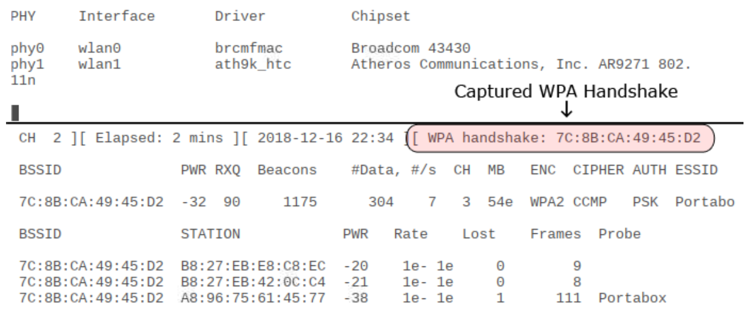
Step 4 - Client processing: The audio data is transmitted from the listener device to the client service through the File I/O module. Once this is finished the client saves the audio file locally, making it available to the attacker.

Evaluation: For this attack, we had success at every stage of the attack. Tests performed in different locations of the HDMI distribution proved successful. Script activation began and a recording was saved locally. The listener device successfully received the activation command from the client and a recording was successfully stored locally within the listener device. At a later time, the client requested the audio data from the listener device through the assigned message. The listener successfully confirmed the receipt of this message and began the data transfer over the CEC network to the client as seen in Figure 6.4. The client successfully stored and deserialized this data into a valid file format. This further opens the possibility to a listener which could await keywords such as “password” passively or use voice-to-text technology to transfer days of conversations to an adversary.

Attack 3: WPA/WPA2 Handshake Theft (Local) This attack was specified in order to demonstrate the concepts of Threat 3 (Facilitation of Attacks) and Threat 4 (Information Theft). In this local attack, the attacker uses HDMI-Walk to facilitate WPA/WPA2 handshake capture and prevent detection by a security system in place. In traditional handshake theft attacks, an attacker has to wait for a handshake to occur, this can take an indefinite amount of time as the WPA handshake is only transferred in specific cases [LDS09a]. If there is a time constraint, the attacker must attempt forced de-authentication [Dor17]. This raises the issue that forced de-authentication may be detected through a network scanner such as Wireshark or through more complex IDS [BADA15]. In this attack, we facilitate such a threat through the removal of time constraints.

Step 1 - Initial Configuration: The attacker must be especially careful about the listener placement. The listener must be able to reach wireless network connections and must also come equipped with a wireless adapter capable of “monitor mode” for packet capture.

Step 2 - Client Trigger: The client triggers the listener’s service wireless attack module. This activates the wireless adapter in monitor mode with airmon-ng, then begins the capture with airodump-ng using the wlan1 interface and BSSID “7C:8B:CA:49:45:D2” in the listener device. Airodump-ng process is opened in separate terminal using Python’s `os` import command. This places the listener in a passive state which awaits handshakes to naturally occur without forced de-authentication. The attacker is not needed for the duration of this capture. At a later time, an authorized user connects and the handshake is captured passively.


```
PHY     Interface       Driver          Chipset

phy0    wlan0           brcmfmac        Broadcom 43430
phy1    wlan1           ath9k_htc       Atheros Communications, Inc. AR9271 802.
11n
```

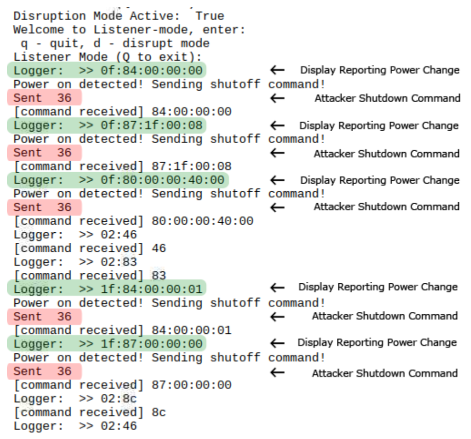
```
 CH  2 ][ Elapsed: 2 mins ][ 2018-12-16 22:34 ][ WPA handshake: 7C:8B:CA:49:45:D2

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 7C:8B:CA:49:45:D2  -32  90     1175      304    7   3  54e  WPA2 CCMP   PSK  Portabo

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 7C:8B:CA:49:45:D2  B8:27:EB:E8:C8:EC  -20    1e- 1e     0        9
 7C:8B:CA:49:45:D2  B8:27:EB:42:0C:C4  -21    1e- 1e     0        8
 7C:8B:CA:49:45:D2  A8:96:75:61:45:77  -38    1e- 1e     1      111  Portabox
```

Figure 6.5: Attack 3–Running handshake capture with Aircrack-ng.

Step 3 - Handshake Retrieval: The attacker reconnects with the client and requests the handshake from listener. The listener first cleans the capture `.cap` file using `wpaclean`. This greatly reduces the file size and the transfer begins. The attacker can finally receive the cleaned capture through the CEC File I/O module.

Evaluation: Local CEC client triggers for the activation of this attack proved entirely successful. Activation of the wireless module, Airodump-ng, and cleanup functions succeeded as seen in Figure 6.5. With the capture size reduced, the handshake was transferred to the local client successfully. This process would allow the attacker to retrieve the handshake at a later date and use more computing resources to attempt to crack the handshake and gain unauthorized access to the network. This would then allow the attacker to enable remote functionality to their own listener.

Attack 4: Targeted Device Attack (Local and Remote) This attack was developed to demonstrate Threat 5 (Denial of Service) through arbitrary sniffing and control of a device. In this attack, the attacker uses functionality from the Python-based listener service to target a specific device in the HDMI distribution. She also takes advantage of the nature of CEC to sniff and detect when a device has been turned on. This attack can be divided into three main steps.



```
Disruption Mode Active:  True
Welcome to Listener-mode, enter:
 q - quit, d - disrupt mode
Listener Mode (Q to exit):
Logger:  >> 0f:84:00:00:00           ←  Display Reporting Power Change
Power on detected! Sending shutoff command!
Sent  36                             ←     Attacker Shutdown Command
[command received] 84:00:00:00
Logger:  >> 0f:87:1f:00:08           ←  Display Reporting Power Change
Power on detected! Sending shutoff command!
Sent  36                             ←     Attacker Shutdown Command
[command received] 87:1f:00:08
Logger:  >> 0f:80:00:00:40:00        ←  Display Reporting Power Change
Power on detected! Sending shutoff command!
Sent  36                             ←     Attacker Shutdown Command
[command received] 80:00:00:40:00
Logger:  >> 02:46
[command received] 46
Logger:  >> 02:83
[command received] 83
Logger:  >> 1f:84:00:00:01           ←  Display Reporting Power Change
Power on detected! Sending shutoff command!
Sent  36                             ←     Attacker Shutdown Command
[command received] 84:00:00:01
Logger:  >> 1f:87:00:00:00           ←  Display Reporting Power Change
Power on detected! Sending shutoff command!
Sent  36                             ←     Attacker Shutdown Command
[command received] 87:00:00:00
Logger:  >> 02:8c
[command received] 8c
Logger:  >> 02:46
```

Figure 6.6: Attack 4–TV Power state change and execution of targeted attack.

Step 1 - Activation: The listener awaits attack activation. It awaits commands either from the local client (through a walk) or from a remote client to activate the targeted attack. Once a command is received, the listener activates the attack.

Step 2 - Sniffing: The listener is set within an HDMI distribution and monitors CEC packets flowing through the distribution. We particularly listen to the data commands "84:00:00:00", "87:1f:00:08" and "80:00:00:30:00" from any incoming source. These values, usually signify a device broadcasting to HDMI distribution devices that its power state has changed and has been turned on. More specifically, 84 reports physical address, 87 reports vendor id, and 80 reports a routing change. In this particular attack, the attacker targets a CEC enabled display, the Sharp television.

Step 3 - DoS attack: Once the attack is active the listener awaits commands associated with power state change within the HDMI distribution. Once the power state change is detected it sends the CEC shutoff command "20:36" to the display (ID: 0) in the distribution. This automatically powers off the display as soon as it is powered on.

Evaluation: The listener began in an inactive state as expected with passive listening of the CEC commands. Powering the display did not cause any changes in this inactive state. The listener successfully received the activation command over remote and local clients, activating the attack mode. With this mode active, the display was manually powered on. The module in our service successfully identified the power state change in the display and provided the shutoff command as seen in Figure 6.6. The display received the shutoff commands and immediately powered off as expected. No matter which method of powering on, the attack could not be avoided, successfully executing the DoS attack. We additionally had another notable finding while performing this attack. That is, during DoS, the user was prevented from disabling CEC control within the system. Additionally, this attack may prove difficult to detect as it may be mistaken for a malfunctioning display.

Attack 5: Display Broadcast DoS (Local and Remote) We developed this attack to demonstrate Threat 5 (Denial of Service) through broadcast functionality. This attack abuses the broadcast function in CEC to cause a DoS condition in any display within a given HDMI distribution. This attack specifically targets displays by producing standard CEC commands for source and input control. We divide this attack into three steps.

Step 1 - Insertion of Attacker Listener: The listener device is placed in any location of the HDMI distribution. The device then awaits instructions from a client service to begin the attack. In the case of an available wireless connection, the listener's Remote Access Module becomes active.

Step 2 - Activation Phase: The listener activates in two different methods: (1) the listener receives a direct command from a client service to begin the attack. (2) the listener receives through a remote client with the `DOS1` command.

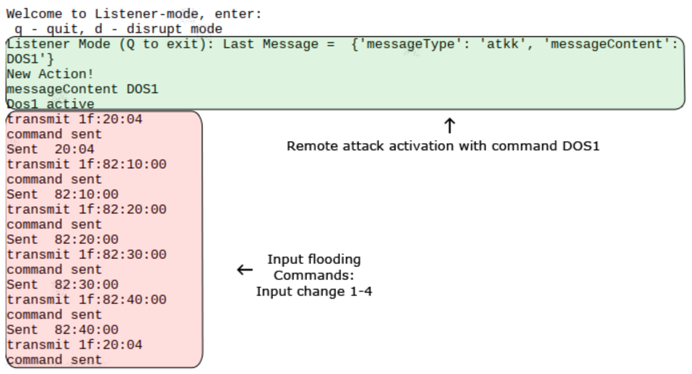

```
Welcome to Listener-mode, enter:
 q - quit, d - disrupt mode
Listener Mode (Q to exit): Last Message =  {'messageType': 'atkk', 'messageContent':
DOS1'}
New Action!
messageContent DOS1
Dos1 active
transmit 1f:20:04
command sent
Sent  20:04
transmit 1f:82:10:00
command sent
Sent  82:10:00
transmit 1f:82:20:00
command sent
Sent  82:20:00
transmit 1f:82:30:00
command sent
Sent  82:30:00
transmit 1f:82:40:00
command sent
Sent  82:40:00
transmit 1f:20:04
command sent
```
Remote attack activation with command DOS1
Input flooding Commands: Input change 1-4

Figure 6.7: Attack 5–Input-change induced DoS attack. Executed by remote attacker with command `DOS1`.

Step 3 - DoS attack phase: After activation conditions are reached, the listener device begins broadcast of various *display input change* commands. These are standard CEC commands accepted by enabled televisions to adjust the active source on the display device. The CEC distribution is flooded with a broadcast loop: power on ("20:04"), input 1 ("82:10:00"), input 2 ("82:20:00"), input 3 ("82:30:00"), and input 4 ("82:40:00"). This renders the displays unusable by the user, effectively creating a DoS attack.

Evaluation: In this attack, the listener began in an idle state as intended in the distribution. The listener successfully received the activation command over remote and local clients. Then, it initiated the DoS broadcast loop over the entire distribution as depicted in Figure 6.7. The attack first powered on the display if it was powered off. The loop then began rapid input change over all inputs on the display. The display began to flash rendering it unusable. We noticed faster switching between inputs than if compared with manual input change. Another effect of this condition is that it made it impossible for the user to alter any settings in the display to disable external control after activation.

Summary and findings: During testing of HDMI-Walk attacks, we identified a vendor-specific vulnerability, and are currently coordinating to report this finding to the product's respective manufacturer. HDMI-Walk can identify specific device information to develop further attacks. We have proven arbitrary control over HDMI devices which could be used to an attacker's advantage. Also, we enabled control of the TV volume and Amplifier volume with devices in our testbed. This control is completely feasible in an HDMI distribution with the concepts of HDMI-Walk. We find these attacks critical as they occur over a medium without any form of security mechanisms or existing techniques for mitigation. Via Attack 4, we found that the input change control could become a viable form of a visual attack. With these functions, display input changes could be used to trigger seizures (e.g., television epilepsy) with the rapid flickering of a display switching between inputs [JIS13]. We also consider volume control to an Amplifier device. A remote attacker with the control of a distribution can easily adjust the volume of devices with CEC commands. Extended playback at high volumes is known to damage sound equipment [Hey13]. An implementation of Attack 1 would first allow an attacker to infer room occupancy via power state. Combining this with Attack 4, the attacker could peak the volume output in a room when nobody is present and cause gradual damage to the sound system, which cause a notable financial cost to the user. Combination of HDMI-Walk and targeted device attacks such as Attack 4 could also allow a malicious person to assume control of menu functions in specific HDMI devices. This would allow the attacker to change menu settings, make purchases, or update firmware through device-tailored command sequences. With attackers in constant search for new vectors of attack, disruption, data leakage, behavioral leakage, and any type of information leakage could present catastrophic outcomes to an organization. A conference room while in confidential use can be a target to eavesdropping and handshake theft, giving attackers a chance to acquire passwords, access codes, and confidential information. In normal usage, inferring devices and disrupting func-

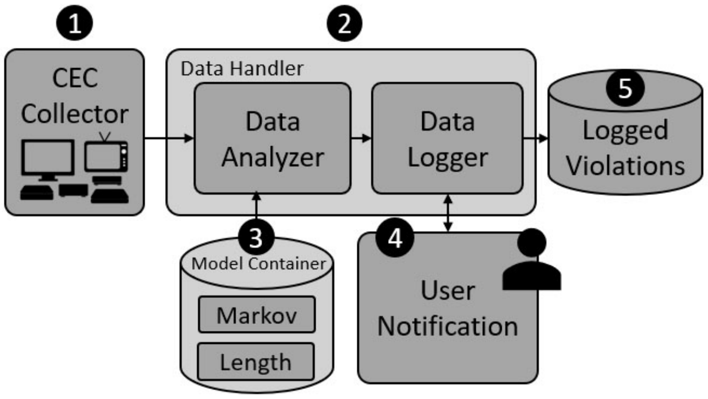


Figure 6.8: Architecture of HDMI-Watch. Each module numbered.

tionality is possible and may present a threat which many users have not considered or anticipated.

6.5 HDMI-Watch Architecture

To address CEC-based threats, we introduce HDMI-Watch, a passive, easily configured intrusion detection system. In this section, we detail the different modules of the HDMI-Watch architecture.

6.5.1 HDMI-Watch Overview

The proposed architecture of HDMI-Watch is divided into five different modules, as seen in Figure 6.8. The first module is the *CEC collector* which captures CEC packets from an HDMI distribution and supplies them to the data handler ❶. The data handler evaluates and logs the incoming CEC traffic utilizing the two sub-modules: the *data analyzer* and the *data logger* ❷. The data analyzer is a sub-module used for classification, applying a machine learning model over incoming CEC traffic to perform both binary (malicious or benign) and signature-based (scanning, data transfer, power change, or input control abuse) classification. The *data logger* is then used to forward this processed data (classification results, and violations) to both the logged violations and the user notification module. The *model container* stores a machine learning model of expected communication behavior for CEC-enabled devices, which is used by the data analyzer sub-module to evaluate CEC data ❸. Any incoming CEC data flagged as a violation by the data analyzer is forwarded to the data logger. The data logger sends the flagged violations to the user notification module which notifies the user on unexpected CEC activity which occurs over the distribution ❹. Finally, the logged violations module stores all the flagged violations and relevant data found by HDMI-Watch ❺. The logged information may be queried later for reference, or further analysis.

6.5.2 CEC Collector

The CEC collector provides HDMI-Watch the CEC traffic necessary to operate over an HDMI distribution. Due to the design of CEC as a bus architecture, a single point of connection allows HDMI-Watch to monitor all active CEC communication from devices within the same HDMI distribution. Additionally, the CEC collector parses the raw data received into a format that other modules of the HDMI-Watch architecture can interpret. Formatted messages out of the collector include all information necessary for evaluation: timestamp, CEC command type, and CEC packet length. If needed for logging purposes, the entire CEC packet is also included. The command type is the main feature and later used by the Markov model, while the length model uses the packet length during the binary classification stage.

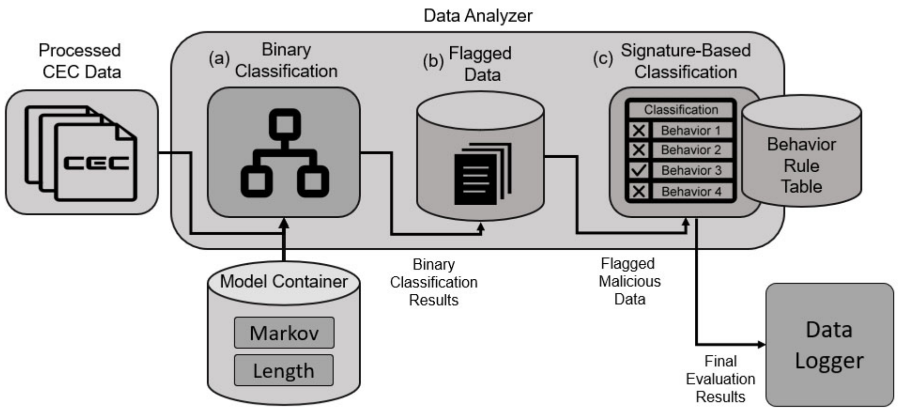


Figure 6.9: HDMI-Watch classification process.

6.5.3 Data Handler

The data handler acts as the evaluation stage for HDMI-Watch. We divide its functionality into two main sub-modules; the Data Analyzer and the Data Logger.

Data Analyzer

The data analyzer is the core of HDMI-Watch, making the distinction on whether incoming CEC data is from malicious or benign activities. Additionally, the data analyzer performs signature-based classification of malicious activity into different types of attack behaviors (see Section 6.4).

We refer to Figure 6.9 for the classification process performed by the data analyzer. The first step in HDMI-Watch classification is binary classification, which attempts to classify CEC activity as benign or malicious. To perform binary classification, the data analyzer refers to the model container which contains the Markov and length model used for binary classification. Any violation found from the incoming data by the models is cached locally by the data analyzer into lists as flagged data. The data analyzer can determine the number of violations for the number of messages received. If the number of flagged violations exceeds the detection threshold, the activity is deemed malicious. Once a malicious activity has been identified, HDMI-Watch begins signature-based classification, inferring the specific type of activity in the flagged data (scanning, data transfer, power control, or input control abuse). To perform signature-based classification, HDMI-Watch uses a behavior rule table where command types associated with different types of activities. For instance, malicious activity found during the binary classification stage will be labeled as power change abuse if most of the violations in the flagged data are power control commands. The resulting evaluations from both binary and signature-based classification along with the flagged data are then passed to the data logger module for logging and to the user notification module.

Binary Classification. The data analyzer uses binary classification in HDMI-Watch to infer if activities within the HDMI distribution are benign or malicious. HDMI-Watch is a flexible system with adjustments to improve classification accuracy and better fit the likely heterogeneous HDMI distributions where HDMI-Watch is deployed. Thus, HDMI-Watch employs a configurable *violation threshold* as the acceptable number of violations for a number of received CEC messages. In addition, the sample of received CEC messages may also be adjusted to improve the quality of classification. Binary classification in HDMI-Watch classifies sets of violations as malicious CEC activity if the number of violations exceeds the violation threshold for a number of received messages.

Signature-Based Classification. HDMI-Watch uses the data analyzer to perform signature-based classification of malicious CEC behavior. We create a set of rules for each behavior (scanning, data transfer, power control, or input control abuse) and classify violation sets as a second step to the initial binary classification. HDMI-Watch uses a predefined ruleset to infer the type of behavior occurring in a malicious set of data. These behaviors were selected as they are related to HDMI-Watch attack behaviors. It is possible to configure HDMI-Watch to classify other behaviors with additional rules. We narrow down to four different unauthorized behaviors from five attacks using command types:

- **Scanning.** Scanning is heavily associated with Attack 1. Scanning is required for an attacker to gather information about a CEC distribution and devices.
- **Data Transfer.** Attempting file transfer over a CEC bus is not standard CEC operation and is strongly associated with Attack 2 and 3. To the best of our knowledge, data transfer capabilities through CEC are not commonly used by any manufacturer.
- **Power Control.** While power control commands may be issued by devices in a benign manner. We can associate the unexpected use of power control to Attack 4.
- **Input Control.** Input change may be issued by devices in standard operation. However, abuse of these commands is associated with Attack 5.

Data Logger

The data logger module receives evaluation results, violations, and relevant data found during the data analyzer stage. The data logger serves a storage endpoint to process these results into a database-compatible format. Additionally to formatting, the logger is responsible for storing this data into the logged violations database. This module, essentially allows for the users of HDMI-Watch to refer to past events and view logs on activity which may have been deemed suspicious.

6.5.4 Model Container

The model container stores the HDMI-Watch models used to evaluate CEC traffic in an HDMI distribution. This module uses the command type of CEC packets as its main feature, with length as a secondary attribute and is used by the data analyzer module to predict unexpected behaviors and also specific types of malicious activities. Specifically, the model container is divided into two parts, the *Markov model* and the *length model.* HDMI-Watch uses the Markov model to determine if a command type in CEC traffic is expected after the last message received. Additionally, HDMI-Watch uses the length model to determine if the length of a CEC packet matches expected lengths for the command type. Packets which violate these models are flagged as *violations.*

Markov Model

The Markov model is the core module of the model container and is used by the data analyzer for CEC data evaluation. Since vendors and attackers have the ability to create CEC packets of any command type, HDMI-Watch must consider all possible command types as states. In effect, this yields to a total of 256 (00-FF) states for the HDMI-Watch machine learning model. The data analyzer refers to this model to analyze CEC traffic and infer if received command type follows expected behavior. Any CEC packet which does not follow expected behavior is marked as a *violation* and acts against the *detection threshold*, causing CEC activity to be deemed malicious after being reached.

Mathematical Foundations: We build a Markov-Chain-based model to perform binary classification of the CEC behavior within the HDMI distribution. With the Markov chain model, we evaluate the probability of changes in CEC command

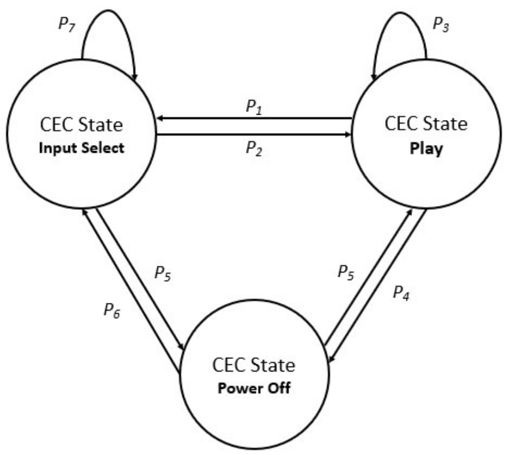


Figure 6.10: Three command types are shown as states in a CEC Markov Model. Probabilities given (P_1 to P_x) as the possibility one command type following another command type.

behavior over time. The Markov chain model serves as the core classification mechanism for HDMI-Watch.

We represent the probabilistic condition of CEC state changes in Equation 6.1 where X_t denotes the CEC command as a Markov chain state at time t. Figure 6.10 illustrates a simplified version of a CEC behavior with three command types defining Markov Chain states and probabilities (P_1,...,P_7) of transitions between these states. For instance, P_5 being the probability of a CEC "Power Off" command being sent after an "Input Select" message.

$$\begin{aligned} P(X_{t+1} = x | X_1 = x_1, X_2 = x_2..., X_t = x_t) = \\ P(X_{t+1} = x | X_t = X_t) \\ when, P(X_1 = x_1.X_2 = x_2..., X_t = x_t) > 0. \end{aligned} \tag{6.1}$$

In HDMI-Watch, we observe the commands transmitted by a set of devices over time. Let us assume that C denotes a set which represents a set which contains all transmitted command types over a CEC bus, such that $C = \{C_1,\ C_2,\ C_3,...,C_n\}$, where C_1, C_2, C_3,...,C_n = any CEC standard command value type (e.g., 36, 8f, 00). For the function of time, t, we consider the command during time t as the state in our model. If we consider the number of total number of unique CEC commands, there are a total of 256 possible commands which must be considered.

If we assume that the distribution's states are X_0, X_1,..., X_T. at a given time of $t = 0, 1, \ldots, T$. We can then represent the transition probability P_{ij} as shown below:

$$P_{ij} = N_{ij}/N_i,$$

where N_{ij} is the number of transitions from X_t to X_{t+1}, with $X_t = i$ and $X_{t+1} = j$; N_i being the total number of transitions from state i. Initial probability distribution of this Markov Chain is represented as follows:

$$Q = [q_1, q_2, q_3, ..., q_m],$$

this model denotes q_m as the probability that the model is in state m at time 0. The probability of observing a sequence of states (X_T) at a given time T, can be computed as shown:

$$P(X_1, X_2, ..., X_t) = q_{x1} \prod_{2}^{T} P_{X_{t-1}X_t}.$$

For HDMI-Watch, instead of predicting future states, we determine the probability of a transition between states at a given time. We train our Markov Model with a dataset collected from active CEC devices and create a prediction model to calculate the probability accordingly. Any packet with a probability of zero from a previous state is deemed unexpected and therefore a violation of the Markov model.

Length Model

HDMI-Watch uses a length model to determine if a command received is of the expected length. This model contains mappings of associated lengths to command types found during the training phase of HDMI-Watch, taking advantage of the association between length and type of CEC packets in CEC communication. For instance, messages with the command type "36" shut off should not contain additional data and the length model serves as a method to detect discrepancies in cases such as these. The data analyzer refers to the length model in the binary classification stage to determine if CEC packets are *violations* of the expected length model. Any CEC packet which is deemed a violation of the length model acts against the *detection threshold* and may cause incoming data to be classified as malicious.

6.5.5 User Notification

The user notification module is the primary form of notification to a network administrator using HDMI-Watch. After CEC traffic is analyzed, the user notification module details any violations to an administrator. This administrator is shown the complete set of packets including packet source, packet destination, command type, data, length, timestamp, violation type, and possible attack type. Ideally, an administrator receives a text message or an email when a violation or a set of violations occur. This information is also archived by logged violations module of HDMI-Watch.

6.5.6 Logged Violations

The logged violations module acts as a storage database for any violations found during HDMI-Watch monitoring. The administrator queries this module to view

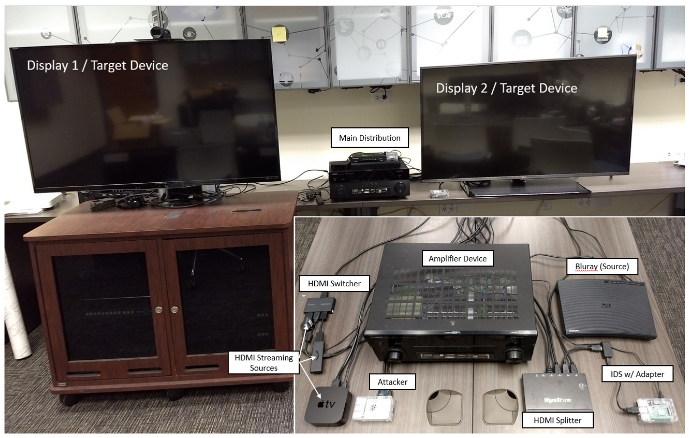


Figure 6.11: HDMI testbed, including two targeted displays, an attacker device, and the HDMI-Watch device within the same distribution.

a history of violations and react accordingly to any threat. This enables proper mitigation by the administrator and allows for simple tracking of past suspicious behavior. Logged violations only include commands deemed malicious, as well as evaluations of what type of action is occurring with a set of these violations based on HDMI-Walk attacks. This acts as the final stage of HDMI-Watch and as a point of reference for any network administrator authorized to view HDMI-Watch logging.

6.6 Implementation of HDMI-Watch

To implement HDMI-Watch's necessary modules, we used a modified version of Pulse-Eight's LibCEC library [Pul18] and Python extensions [Pyt19]. All the software used is open source and freely available online. We refer to Figure 6.11 as the testing environment for HDMI-Watch. Our testing environment uses several commodity HDMI devices (A/V receiver, switcher, source devices, displays), an attacker client, the attack listener, and a device hosting HDMI-Watch. We assume the attacker executes all the HDMI-Walk attacks as covered in Section 6.4 of this manuscript, receiving execution commands from an attacker with a connection to the attacking device.

6.6.1 CEC Collector

Implementing the CEC collector required modification to the original LibCEC library. The original LibCEC code filters some incoming CEC packets, which was not desirable for HDMI-Watch. For the CEC collector, we removed all forms of filtering from the original LibCEC source code. The removal of such allowed HDMI-Watch to monitor all ongoing CEC communication in an HDMI distribution, no matter the source or destination. Live CEC data was received using our modified LibCEC library. The received data was then pre-processed with Python into a comma-delimited string that included the CEC packet's timestamp, command type, length, and the CEC packet itself. Once data was formatted, data was passed to the data handler module.

6.6.2 Data Handler

The data handler includes two sub-modules, the data analyzer and the data logger. Both sub-modules used by the data handler were instrumented using Python libraries.

Data Analyzer

The data analyzer was implemented as a Python-based sub-module operating within HDMI-Watch. Internally, the data analyzer fetches each attribute from the received data (timestamp, packet, and length). The data analyzer takes note of the command type of each incoming packet. For instance, the CEC packet "02:89:01" was marked

Table 6.4: signature-based evaluation by behavior type rule set. Malicious activity is tagged as the case with the most occurrences by command type.

Command	Behavior	Characteristics
83	Scanning	Scanning, eavesdropping behavior
00	Data Transfer	Multiple occurrences data transfer
36	Power Control	Possible abuse in power control
20	Input Control	Possible input control abuse

as command type "89" and length of "8". This sub-module then queries the model container in search of violations in incoming data. Incoming CEC packets deemed suspicious are flagged and logged. This flagged data was then used to infer the type of CEC activity occurring in an HDMI distribution given the specific rules from Table 6.4.

Data Logger

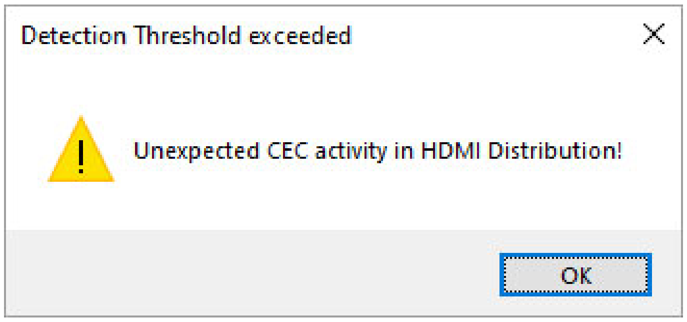
The data logger uses standard Python I/O libraries to convert data into a comma-delimited format. This information was then exported as an external document containing all the flagged data, classification results, and relevant information.

6.6.3 Model Container

The model container was comprised of the two models used by HDMI-Watch for classification purposes, the *Markov model* and the *length model*. In this subsection, we overview the Markov model, the length model, and the data process to implement these models.

Markov Model

The Markov model was stored as a map of key pairs and probabilities created from the training data. This model was a comma-delimited document that was imported


Detection Threshold exceeded

Unexpected CEC activity in HDMI Distribution!

Figure 6.12: User notification shown when detection threshold is exceeded in HDMI-Watch.

into the HDMI-Watch at runtime. For instance, an association of command 9D to 90 was stored as follows “9D-90,0.5”. The first value expresses a command type pair (90 received after 9D). The second value was the probability of the command type pair occurs in this trained model.

Length Model

The length of each packet was associated with the command type gathered during the training phase. The length model was stored as a serialized Python dictionary derived from the training data and functions as a complementary feature to the Markov model. The length model stored the associated lengths in key-list pairs to their respective command type. For instance, if the command type “87” has the expected length as 11 and 14, this was stored as “’87’: [14, 11]” in the length model.

Data Collection and Training

To train the HDMI-Watch classifier, we collected daily data usage from an HDMI environment. We performed normal operation of the HDMI equipment as defined in Section 6.5. Normal operations involved using the environment for music, movies, videos, and any manner consistent with an HDMI distribution system. Data from the environment was collected over the span of two weeks, where users performed normal operations using the testbed. This collected data was then used as the training data for the model container module using a python-based software designed for HDMI-Watch. In total, we collected 61,765 benign CEC communication packets during the standard operations of the HDMI testbed as the training data. To train the model, we followed an unsupervised learning approach that did not require labeled data for training. We found that the unsupervised learning approach with the Markov model provided more flexibility for HDMI-Watch and required fewer system resources during the learning process.

6.6.4 User Notification

The user notification module in HDMI-Watch was implemented to notify the user when the detection threshold was exceeded and thus, malicious activity was detected. The current implementation functioned as a popup notification on the local machine using the Python module Tkinter [Pyt19]. Figure 6.12 shows the notification which occurs when violations exceed the detection threshold. Implementations of email, texting and other notifications are a very straightforward task which can be very easily implemented.

6.6.5 Logged Violations

Violations are logged directly into a comma-delimited document stored in the device hosting the HDMI-Watch system. This file is created using standard Python I/O libraries, appending messages to the end of the output document. This comma-delimited document contains all relevant information to the violation: timestamp, data, packet length, violation type, and possible attack-type of each violation.

6.7 Performance Evaluation

In this section, we evaluate our solution against HDMI-Watch attacks. Specifically, we aim to answer the following research questions:

RQ1: Threshold Evaluation. How does HDMI-Watch's detection threshold affect the binary classification results in HDMI-Watch? (Section 6.7.1)

RQ2: Malicious Activity Type. How effective is HDMI-Watch in classifying between different types of malicious behaviors? (Section 6.7.2)

Attack Implementation

Based on previously mentioned HDMI-Watch attacks, we perform the attacks as per Section 6.4 specifications. We assume that the attacker perpetrates the attacks following a Normal distribution. As such, we assume there is an equal probability that any attack will occur at any given time and a valid model to emulate a single attack executed at the time, which we consider a realistic approach to emulate HDMI-Watch attacks. Considering *t=[0,T]* as the timeframe of the attack, we present the vector of the attacks as follows, with five types of attacks:

$$AT_i = [AT_1, AT_2, AT3, AT_4, AT_5], \tag{6.2}$$

such that the probability of having an attack occurring:

$$P(AT) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(AT-\mu)^2/2\sigma^2}, \tag{6.3}$$

where µ is the mean number of attacks per given timeframe and σ being the standard deviation of attacks occurring within a given timeframe.

Attack Data Collection.

To evaluate the data classification capabilities of HDMI-Watch, we collected data from the interaction between all CEC devices in the distribution. The activity collected for the evaluation included regular CEC activity as defined in Section 6.5 and malicious activities from HDMI-Walk attacks. During the malicious data collection, we executed each attack 30 times within the HDMI test environment. The collection resulted in a total of 150 datasets of attack data, 30 for each attack. Additionally, we recorded 30 datasets of CEC traffic from the expected operations of the HDMI testbed as defined in Section 6.5. All of the attacks were executed as noted in Section 6.4. For AT_1, our attacker device performed CEC-based scans over the distribution. For AT_2 and AT_3, we performed file transfer of the captured data to the attacker device. For AT_4, we activated the attack and attempted to power on the display under attack conditions, recording the attacker device shutting off the display. In AT_5, we attacked a display in the distribution via rapid input change.

Performance Metrics

For performance metrics, we utilized the standard parameters: accuracy, True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), False Negative Rate (FNR), recall, precision, and F-score. For instance, in the case of benign evaluation True Positive Rate (TPR) denotes the total number of correctly identified benign CEC activity within the test environment. True Negative Rate (TNR) denotes the total number of correctly identified malicious CEC activity within the test environment. False Positive Rate (FPR) denotes the total number of cases where malicious CEC activity was mistaken as being benign. False Negative Rate (FNR) denotes the total number of cases where benign CEC activity is mistaken as malicious.

Table 6.5: Binary classification performance evaluation of HDMI-Watch on varied violation thresholds T

T	Binary	TPR	TNR	FPR	FNR	ACC	PREC	REC	F1
1	Benign	0.87	1.0	0.0	0.13	0.87	1.0	0.87	0.93
	Malicious	1.0	0.87	0.13	0.0	0.97	0.97	1.0	0.99
2	Benign	0.90	1.0	0.0	0.1	0.9	1.0	0.9	0.947
	Malicious	1.0	0.9	0.1	0.0	0.98	0.98	1.0	0.99
3	Benign	0.97	0.99	0.01	0.03	0.94	0.97	0.97	0.97
	Malicious	0.99	0.97	0.03	0.01	0.99	0.99	0.99	0.99
4	Benign	0.97	0.96	0.04	0.03	0.81	0.82	0.97	0.89
	Malicious	0.96	0.97	0.03	0.04	0.95	0.99	0.96	0.98
5	Benign	0.96	0.93	0.07	0.03	0.73	0.74	0.97	0.84
	Malicious	0.93	0.97	0.03	0.07	0.92	0.99	0.93	0.96
6	Benign	0.96	0.91	0.10	0.03	0.66	0.67	0.97	0.79
	Malicious	0.91	0.97	0.03	0.10	0.90	0.99	0.91	0.95

$$RecallRate = \frac{TNR}{TNR + FPR}, \tag{6.4}$$

$$PrecisionRate = \frac{TPR}{TPR + FPR}, \tag{6.5}$$

$$Accuracy = \frac{TPR + TNR}{TPR + TNR + FPR + FNR}, \tag{6.6}$$

$$F1 = \frac{2 * RecallRate * PrecisionRate}{RecallRate + PrecisionRate}. \tag{6.7}$$

6.7.1 Performance of HDMI-Watch Classification for Different Violation Thresholds (RQ1)

As part of *RQ1*, we evaluate binary classification and the effects of the violation threshold as defined in Section 6.6, we processed a total classification of 180 datasets (30 malicious for each attack and 30 for benign cases) with HDMI-Watch using different threshold values. As highlighted in Section 6.5, part of HDMI-Watch classification involves the violation threshold per number of packets received. We refer to the classification results in Table 6.5 for different detection threshold values (per 400 packets received). In these results, we show how different threshold values affect the classification results in terms of accuracy and precision.

Table 6.6 presents the binary classification results for T=2 against malicious and benign behavior with accuracy, precision, recall, and F1 metrics for each case. We use this value for T as it presents no false negatives for malicious test cases. For this case, we obtain an accuracy of 90% and precision of 100% for benign detection, with only three benign cases misclassified as malicious out of 30 cases. In the case of malicious activity, we achieve an overall accuracy and precision of 98%.

With the configurable design of HDMI-Watch, evaluating the behavior on different thresholds is important and yields some interesting results dependent on attack behavior. We observed that the violation threshold is particularly important for attacks with fewer violations (scanning and power control). The proposed scanning and power control attacks require less CEC packets to execute than attacks involving rapid input switching or data transfer. As a result, the number of violations is much less for scanning and power control behaviors, making such attacks less noticeable. Behaviors such as data transfer in CEC were more easily detectable (higher number of violations) than power control, as an attack involving data transfer or input change spam involves many more packets than abusing shut-off commands. Therefore, if the threshold is too high then attacks with fewer violations may be missed by HDMI-Watch. Inversely, a threshold that is too low, benign behavior may be improperly classified as malicious, reducing the accuracy of HDMI-Watch. During testing, A more interesting case was a benign case with 16 violations. This case occurred during a manual power cycling of an AppleTV in the distribution.

Table 6.6: Binary performance evaluation for HDMI-Watch. Classifying expected vs unexpected behavior for T=2.

Binary	TPR	TNR	FPR	FNR	ACC	PREC	REC	F1
Benign	0.90	1.0	0.0	0.1	0.9	1.0	0.9	0.947
Malicious	1.0	0.9	0.1	0.0	0.98	0.98	1.0	0.99

Media centers such as these are “always-on” devices thus power cycling was benign, but unexpected operation detected by HDMI-Watch.

6.7.2 Classification of Malicious CEC Behavior (RQ2)

As part of *RQ2*, we refer to Table 6.7 for HDMI-Watch’s effectiveness in classifying different malicious CEC-based behaviors. For HDMI-Watch testing, we classified malicious activities into four different types of behaviors (scanning, file transfer, input control, or power control) based on the command type of the violation. Our results show that HDMI-Watch achieves an average accuracy of 98% and an average precision of 99% for signature-based classification. Additionally, our results show that transfer and input change behavior detection achieved perfect classification with HDMI-Watch. In the case of power command abuse, HDMI-Watch mislabeled one case as scanning behavior, impacting the individual accuracy for scanning and power behavior classification.

One of the observations made is that some attacks were more easily distinguishable than others. We highlight that data transfer and input behaviors were classified with better accuracy as they involve many CEC packets from the attacker. For instance, in the case of file transfer, the serialization of packets and then transmission of those packets yields to many violations of command type “00”. This command type is matched to values on the table and the activities identified as data transfer. Similarly, rapid input requests require the use of CEC command “20”, allowing

HDMI-Watch to easily distinguish the type of behavior being executed. In contrast, other behaviors with a smaller footprint (less CEC packets required) such as scanning and power control do not rely on a large number of CEC commands to execute. In effect, these behaviors were more difficult to detect by HDMI-Watch compared to other types of behaviors.

6.7.3 Benefits and Discussion

There are notable benefits to using HDMI-Watch.

Complete Passive Monitoring. HDMI-Watch is based on complete, passive monitoring. This has two main advantages. First, HDMI-Watch does not affect CEC performance in an HDMI distribution. Second, our method does not require any changes to the overall protocol or to existing devices in a distribution.

Complete Black-box integration. HDMI-Watch resolves one of the biggest issues of HDMI-based devices, the lack of technical documentation. HDMI-Watch does not require knowledge of source code or operation of any device. HDMI-Watch may learn from live analysis during the training phase of a system.

Flexible Design. It is possible to adjust HDMI-Watch to specific deployments. For instance, the threshold used to classify if an activity is malicious or benign can be adjusted. In addition to the threshold, the number of packets on which the threshold is applied to may also be easily configured to fine-tune HDMI-Watch classification. Additionally, the signature-based classification ruleset may be altered to include different types of behaviors not covered under this chapter.

Privacy. HDMI-Watch only requires command type and length to operate. With the ability to strip communication data from packets HDMI-Watch requires no sensitive information to operate. HDMI-Watch can improve the privacy of an HDMI

Table 6.7: Signature-based classification performance evaluation of the proposed IDS. Detecting behavior by type.

Behavior	TPR	TNR	FPR	FNR	ACC	PREC	REC	F1
Scan	1.0	1.0	0.008	0.0	0.968	0.968	1.0	0.984
Transfer	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0
Input	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0
Power	0.967	1.0	0.0	0.033	0.967	1.0	0.967	0.983

system by detecting the insertion of new devices and preventing malicious CEC scanning. Even though inferring sensitive information within an HDMI distribution from malicious scanning or passive sniffing is possible, a more formal privacy analysis on HDMI is necessary to fully evaluate CEC information leakage. As CEC behavior varies from different devices and device-to-device interaction, a future study on privacy issues from HDMI behavior is needed.

Detection Time. HDMI-Watch attack detection is based on how quickly CEC packets are issued during the attack, most attacks were detected before the attacks had finished. As such, attacks that issue more packets in a shorter span of time are going to be detected quicker. The time of detection is also difficult to guarantee due to several factors affecting the attack behavior. For instance, an attacker may send CEC packets at a slower rate to accomplish the same attack over a longer span of time. Additionally, data-transfer behaviors may vary depending on the amount of data transmitted and the rate of transmission. As attacks are detected through the detection threshold, the threshold can be adjusted to detect slower attacks by reducing the detection threshold. However, while this may raise the number of false positives, attacks would also be slower and more ineffective.

Scalability. As with many machine learning systems, data must be gathered on every testbed and on the addition of every new device. This is not unreasonable, as new devices might not be added often within HDMI distributions. As such, the design of HDMI-Watch allows to save and load models. It may be possible to create models of combinations of devices, and just load the right model into HDMI-Watch when a new device is added. As devices of the same make and model should have similar behavior, models may be reused in multiple distributions. Additionally, it is possible for HDMI-Watch to ignore certain devices temporarily while a new model is built. Such actions could reduce false positives for devices which were not in the testbed before. Another solution is to add a CEC-less adapters to new devices, this would keep them from entering the CEC bus while allowing all other HDMI functionality.

6.8 Conclusion

Today there are close to 10 billion High Definition Multimedia Interface (HDMI) devices in the world and HDMI has become the de-facto standard for the distribution of A/V signals in smart homes, office spaces, sports events, etc. A component of this widely-deployed interface is the CEC protocol which is used to control devices using the HDMI interface. With no currently known security solutions in place or security implementations in the CEC protocol design, CEC opens a realm of possibilities to attackers. In this chapter, we highlighted HDMI-Walk, a novel attack surface against HDMI distribution networks and presented five different attacks using this vector. We studied how current insecure CEC protocol practices and HDMI distributions may grant an adversary a viable attack surface against HDMI-enabled devices. Using HDMI-Walk, we analyzed the CEC propagation and implemented a series of local and remote CEC based attacks as a proof-of-concept design. Specifically, we used HDMI-Walk to perform malicious analysis of devices, eavesdropping, DoS attacks, targeted device attacks, and facilitate existing attacks through HDMI. As current network security mechanisms only protect traditional networks and components, CEC-based threats are outside of their scope. To defend against these threats, this manuscript proposed HDMI-Watch, a novel easily-configured security mechanism tailored specifically for the classification of CEC-based abnormal behavior. HDMI-Watch operates as a passive, standalone framework in an HDMI distribution and provides no additional overhead to CEC communication. Finally, We evaluated the performance of HDMI-Watch in an HDMI testbed under realistic conditions. HDMI-Watch evaluation results showed levels of over 90% in accuracy and precision for binary and signature-based classification.

CHAPTER 7

SERIAL-BASED ATTACKS AND DEFENSES FOR E-IOT COMMUNICATION BUSES

7.1 Introduction

The rapid adoption of specialty smart systems has changed the lives of millions of users worldwide [IoT18]. As part of these ecosystems, Enterprise Internet of Things (E-IoT) are smart systems designed to allow users to integrate and control very complex installations at a higher cost than off-the-shelf IoT systems. As such, E-IoT systems grant users a robust, reliable, and accepted solution for smart installations and complex deployments. Many vendors such as Savant, LiteTouch, Crestron, and Control4 offer E-IoT solutions, which are then deployed and configured to a user's specification by trained installers. In effect, E-IoT systems are often found in smart settings where security oversight is critical (e.g., smart buildings, hotels, government and private offices, smart homes, businesses, yachts, colleges).

While the security of numerous off-the-shelf IoT smart systems is well-understood due to prior research and mainstream knowledge, very little research exists on E-IoT and their proprietary technologies. With many of these E-IoT systems deployed in high-profile locations (e.g., government and enterprise offices, colleges, conference rooms, hospitals), evaluating possible threats for these E-IoT smart systems should be of utmost importance. However, many E-IoT systems use proprietary communication protocols that rely solely on security through obscurity. The motivation of this paper is to shed light upon the security of E-IoT systems and uncover possible vulnerabilities of E-IoT proprietary communication protocols that can affect millions of E-IoT deployments. To address this open research problem and determine if E-IoT systems are susceptible to attacks, we focus on one of the core E-IoT components, E-IoT communications buses. E-IoT communication buses are used by E-IoT proprietary communication protocols to carry out fundamental internal communication functions such as interactions between user interfaces and the central controller. Specifically, communication buses are used to trigger programmed E-IoT events on integrated devices. In this work, we take a look at Crestron's Cresnet, a proprietary communication bus protocol used by one the major E-IoT system vendors. Crestron is a great example of a globally accepted E-IoT system with billions in sales, deployments in over 90% of Fortune 500 companies, and thousands of independent installers [Mar18]. To demonstrate that it is feasible for attackers to compromise E-IoT systems through insecure communication protocols, we propose LightningStrike, a series of novel attacks created to leverage communication buses against insecure communication protocols, namely Cresnet, to an attacker's advantage. With LightningStrike, we demonstrate that an attacker with limited resources can (1) cause Denial-of-Service (DoS) conditions in an E-IoT system, (2) maliciously eavesdrop system communication, (3) execute replay attacks to cause undesired behavior (e.g., open a door), and (4) impersonate other E-IoT devices.

Our evaluations with a realistic E-IoT testbed show that LightningStrike provides attackers with effective, practical, and covert mechanism to compromise E-IoT systems and cause great damage which can broadly impact millions of current and future E-IoT deployments. For this reason, protecting E-IoT systems against such attacks are imperative. However, E-IoT systems have distinct challenges. For instance, E-IoT systems and underlying protocols are closed-source and cannot be modified by third-parties. Further, modifications to existing protocols would require upgrading or replacing E-IoT systems at a great cost to the system users. These limitations make traditional defense strategies inadequate for such threats. Thus, a new defense mechanism is needed that considers the mentioned challenges and utilizes existing system resources. Thus, we present LGuard, a defense system designed specifically to protect E-IoT deployments against LightningStrike threats. LGuard first increases the difficulty of eavesdropping by obfuscating E-IoT traffic through the insertion of redundant traffic in the E-IoT communication bus. Further, LGuard uses passive traffic monitoring to identify E-IoT device tampering against impersonation attacks and voluminous traffic to detect LightningStrike-style DoS attacks. Finally, LGuard detects replay attacks using computer vision techniques and the video captures of existing CCTV system. To test LGuard's performance, we implemented LGuard and created a realistic E-IoT testbed. Our extensive evaluations show that LGuard achieves an average accuracy and precision of 99% in detecting LightningStrike-style DoS, impersonation, and replay attacks without operational overhead or modification to the E-IoT system. In addition, LGuard effectively increases the difficulty of extracting valuable information for eavesdroppers via E-IoT traffic obfuscation.

The contributions of this chapter are as follows:

- We introduce LightingStrike, a set of attacks against E-IoT proprietary communication protocols.
- We demonstrate that communication buses used by major E-IoT vendors (e.g., Cresnet) can be used as an attack vector against E-IoT systems using LightingStrike.
- We test LightingStrike attacks in a realistic E-IoT Crestron testbed and leverage communication buses to cause undesired behavior on behalf of an attacker.
- We propose LGuard, a novel defense system designed to protect E-IoT deployments against LightingStrike-style threats.

- We evaluate the performance of LGuard in a realistic E-IoT testbed and show that it achieves an overall accuracy and precision of 99% in detecting DoS, impersonation, and replay attacks while mitigating eavesdropping attacks via obfuscating the E-IoT traffic.

7.2 Differences from Existing Works

While prior works highlight threats against off-the-shelf IoT systems through well-known attack vectors (e.g., TCP/IP, WiFi, Zigbee, Z-Wave), LightingStrike is the first in the literature that uncovers the insecurities of E-IoT by focusing solely on proprietary protocols used in E-IoT. By this way, we shed light upon security of proprietary E-IoT communication through unconventional attack vectors. In order to analyze the security of such systems and demonstrate realistic attacks, we created a testbed utilizing real E-IoT devices of one of the most popular E-IoT systems, namely Crestron. We demonstrated four attacks, specifically two distinct types of DoS, eavesdropping, and replay attacks. The scope of our attacks relies on proprietary communication, and does not rely on any software-based vulnerabilities, overflows, traditional network connectivity, or fuzzing. To address these threats, we introduced LGuard, a defense mechanism tailored specifically to protect E-IoT communication buses against LightingStrike-style threats. Furthermore, LGuard functions without modification or overhead to the E-IoT system, targeting each threat individually with high precision and accuracy.

7.3 Problem Scope and Threat Model

In this section, we present the problem scope and the threat model for LightingStrike-based attacks.

7.3.1 Problem Scope

This chapter assumes the existence of an E-IoT system with a communication bus network within a smart building, with electric loads integrated to the E-IoT system. Full integration is a realistic assumption as the purpose of E-IoT systems is to integrate many devices into common interfaces. The topology of the communication network includes common components such as lighting modules, switches, magnetic relays, lights, and user interfaces. As such, users have communication bus interfaces (e.g., keypads, touchscreens) available throughout the building to control the lights, physical access, and other smart E-IoT functions. The attacker is Mallory, a visitor with authorized access only to public areas of the smart building. With security policies enacted on all traditional networks (e.g., TCP/IP, WiFi), Mallory's only avenue of attack is through indirect means through an available communication bus in a smart building.

With many wired communication bus interfaces, Mallory finds an unsupervised wired device such as a touch screen docking station. This is a viable assumption as it is unrealistic that every communication bus endpoint and interface in the smart building (i.e., restroom, private office) is being supervised by the building security. Mallory may easily find an empty room with a touchscreen or a keypad and compromise the communication bus by inserting a device such as a compact computer with a communication adapter into a daisy-chain (communication bus) line. The inserted device physically connects to the bus network and grants Mallory the ability to eavesdrop and inject messages into the network bus. Compromising the communication is possible as network buses often do not have any form of security monitoring, as noted in previous sections. An inserted device will not be detected as no intrusion detection mechanisms exist for communication buses in E-IoT. Additionally, bus-based communication is often unencrypted and accessible to all devices that use the same bus. *This behavior allows Mallory to monitor and broadcast arbitrary messages to all devices into the communication bus.* As such, with the compact computer (e.g., Raspberry Pi) inserted, Mallory can hide her inserted device and begin executing her attacks elsewhere.

Attack Practicality. While we proposed an example scenario where an attacker can compromise an E-IoT deployment, there can be many other practical scenarios.

- *Third-party contractors.* Repair and maintenance services often require external contractors (e.g., electricians, plumbers, painters, external I.T.) with unsupervised temporary access to facilities such as smart buildings. In some scenarios, such as re-painting walls or repairing damages, contractors must remove fixtures and mounted E-IoT devices (e.g., keypads, touchscreens). An attacker can be a part of the contractors or can bribe an employee to insert a malicious device in the communication bus line.
- *Rented Rooms.* Some locations may opt to rent conference rooms, allowing outsiders to gain frequent access to parts of the facility, with services such as LiquidSpace [Liq21]. Conference rooms need E-IoT interfaces for the users to control projects, lightning, screens, and A/V required for a presentation. If the communication bus wiring is shared between the rented room and other areas of the facilities, it would be trivial for attackers to insert their devices in the line and later perform attacks.

- *Neighbors.* Locations with E-IoT systems may have neighboring offices or other locations for rent. Cases of attackers using their proximity to their target have occurred in the past [Dav11], as such, E-IoT systems can be attacked in a similar manner. An attacker may temporarily rent a location (e.g., store, office) adjacent to the target E-IoT system as a way to gain physical access to the communication bus wiring of target location through a shared wall or a shared low-voltage junction box. Once the attacker inserts a malicious device into the communication bus, the boxes and the walls can be closed up and the E-IoT system can be compromised.

7.3.2 Definitions

In this sub-section we cover essential definitions for the concepts used in the LightingStrike attacks.

Limited-Access User. A limited-access user is any user, such as a temporary visitor, with guest access to any facility. As such, he/she has restricted access and limited permissions to a facility.

Attacker. The attacker is any user (e.g., temporary visitor) with limited access to the facilities that attempts to gain access to unauthorized resources. The attacker's motivations are to disrupt, gather information, learn user behavior, gain unauthorized access, and perpetrate attacks.

Interface Devices. An interface device is a device that a user can use to interface and operate a smart system (e.g., keypads, touchscreens, buttons, tablets, phones, remotes).

7.3.3 Threat Model

LightingStrike considers the following powerful threats as part of the threat model.

Threat 1: Denial-of-Service. This threat considers DoS attacks where Mallory disrupts an E-IoT system's availability through a communication bus connection. These attacks may target specific devices or affect multiple devices. For instance, Mallory can prevent the usage of multiple keypads by causing conflicts in the communication bus or flooding the bus with redundant messages. Hence, ordinary users cannot use E-IoT interfaces to open/close magnetic doors, operate window shades, trigger lights, trigger emergency panic buttons in case of an emergency situation.

Threat 2: Malicious Eavesdropping. This threat considers Mallory monitoring the communication bus maliciously. As an unauthorized user, this threat allows Mallory to maliciously gather potentially sensitive information about an E-IoT system such as usage, button sequences, and user activity.

Threat 3: Impersonation. This threat considers Mallory maliciously impersonating devices connected to the communication bus. For instance, Mallory altering the identification number of a device to impersonate or cause an undesired E-IoT system behavior.

Threat 4: Replay Attack. This threat considers Mallory replaying messages captured through the communication bus to cause undesired behavior on connected devices. For instance, Mallory can replay a button press to unlock a door relay controlled by a lighting system, turn all or specific lights on/off as frequently as she wants, generate fake emergency button presses, and affect the quality of the working/living environment in various ways.

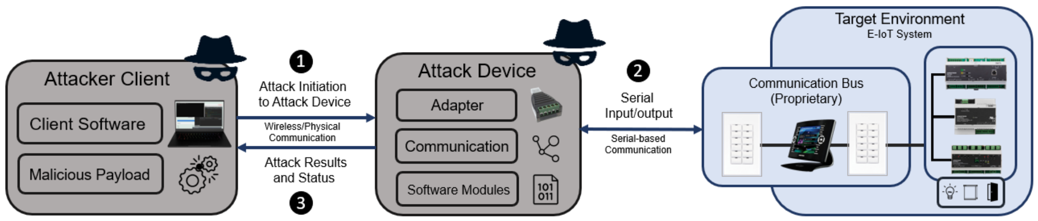


Figure 7.1: General end-to-end implementation for LightingStrike-based attacks. Attack-related components are highlighted in gray, E-IoT components are in blue.

7.4 LightingStrike Architecture

In this section, we describe the LightningStrike architecture and the end-to-end implementation of LightingStrike-based attacks which involves the interaction of three unique components: *attacker client, attack device*, and *target environment.*

7.4.1 LightingStrike Overview

We highlight the architecture of LightingStrike in Figure 7.1. In this architecture, Mallory (the attacker) has compromised the E-IoT communication bus with the insertion of a malicious device (e.g., Attack Device). The attacks against the proprietary E-IoT communication protocol begin with Mallory, using LightingStrike's *attacker client*, such as a tablet, phone, or laptop, to communicate with the attack device and initiate the attacks with the client software ❶. In our case, Mallory sends the *malicious payload* and all the information necessary to initiate the attacks to the attack device using her client software. Communication between the attacker client and the attack device may be wireless (e.g., cellular, Bluetooth, WiFi), using a command-line interface or a VNC connection. The target environment is the E-IoT system being attacked and contains the *communication bus* sub-components. As such, attack device's *adapter* sub-component acts as the physical connection between the communication bus and the attack device. The *software modules* sub-component is the software necessary to interface with the communication bus and attack the proprietary communication protocol. With communication in place, Mallory begins the LightingStrike attacks, transmitting attack-specific commands (using the payload) to the target environment ❷. Finally, the status and results of ongoing attacks are returned to Mallory's attack client with attack device's communication sub-module as the attacks are executed ❸. We further detail the components of the LightingStrike end-to-end architecture.

Attacker Client. The attacker client is any device Mallory uses to execute the attacks, such as a phone, laptop, computer, tablet, or any device capable of running the necessary sub-components and communicating wirelessly with the attack device. Mallory uses this component to initiate, stop, and monitor ongoing attacks against the target environment. As such, to execute the LightingStrike attacks, the attacker client includes two sub-components, the *client software* and the *malicious payload*. The client software is the primary means of communication with the attack device, and may be a command-line interface, VNC client, IDE, or any piece of software that can command and control the attack device. The malicious payload contains all the necessary information to initiate the attacks; for instance, if Mallory wants to execute a DoS attack, the payload specifies the attack type and target. In effect, Mallory runs the client software in the attacker client, submitting her malicious payload to the attack device through her client software, where responses and attack statuses are displayed.

Attack Device. In the proposed architecture, the attack device is a compact device connected to the communication bus. For instance, the attack device is connected physically to the physical wires behind an unattended keypad, or hidden under a docking station. The attack device is designed to act as the intermediary communi-

Table 7.1: Hardware & software used in LightingStrike attacks implementation and evaluation.

Hardware	Software
Crestron DIN-PWS50	Eclipse IDE 2020-03
Crestron C2N-DB12W	Crestron D3 Pro
Crestron DIN-EN-2X18	Crestron Toolbox
Crestron DIN-AP3	Java RX-TX Library
Crestron DIN-8SW8-I	Java 8 SDK
Razer Blade 15 Laptop	VNC Viewer 6.20.529
Acer GX-785 Desktop	TightVNC 2.8.27
GearMo Mini USB to RS485	-

cation point between the attacker client and the target environment. As such, the attack device sends/receives messages through the communication bus to/from the target environment on behalf of Mallory. Additionally, the attack device receives the malicious payload from the attacker client software and executes the attacks. The attack device is comprised of three sub-components, 1) the *communication*, 2) the *adapter*, and 3) the *software modules* sub-components. The communication sub-component allows the attacker client to communicate with the attack device wirelessly (e.g., Bluetooth, WiFi, cellular) or through wired communication. The adapter sub-component, such as a USB-to-Serial interface, acts as the attack device's physical connection and is directly connected to the E-IoT system's communication bus. Finally, the software modules component contains all software logic necessary to monitor, interface, and attack the proprietary protocol through the communication bus. It is configured to the baud rate and communication specifications of the communication bus. We elaborate on independent software modules as follows:

- *Monitoring Module.* The monitoring module passively eavesdrops on active bus communication without transmitting messages. As bus-connected devices broadcast messages in bus architectures, the monitoring module is able to capture all messages transmitted.

- *Injection Module.* The injection module injects arbitrary messages from Mallory into the communication bus.
- *Flooding Module.* The flooding module is designed to cause DoS conditions in the communication bus, by maliciously flooding the communication bus.
- *Re-addressing Module.* The re-addressing module has the ability to re-address and modify the configuration in devices that use the communication bus. As such, it may allow devices to impersonate others by setting identification numbers and other configurations.
- *Filtering Module.* The filtering module filters communication received by the monitoring module to allow Mallory only to view the information requested and make incoming data easier to interpret.

Target Environment. The target environment is the E-IoT system compromised by Mallory using the LightingStrike attacks. The target environment integrates several physical devices (e.g., lighting, shades, relays, magnetic door access, fans) into a central system. With all the devices integrated, they are operated through any user interface such as a keypad or a touch screen connected to the communication bus. These interfaces are all connected through the *communication bus* sub-component, which is used by the target environment.

7.5 LightingStrike Attacks Implementation

In this section, we describe the LightingStrike attacks implementation. We use LightingStrike to attack the Cresnet E-IoT proprietary communication protocol. As noted earlier in Section 7.3 and Section 7.4, Mallory compromises an E-IoT system through the insertion of a malicious attacker device into the communication bus. To ensure that LightingStrike attacks are demonstrated and evaluated realistically, we created an attack suite and a realistic E-IoT testbed. The Attacker client was implemented as the Acer GX-785 desktop and the attack device as the Razer Blade 15 laptop with the attached Gearmo Mini USB-to-RS485 adapter. We established the connection from the attacker client to the attack device using a VNC client/server.

LightingStrike Testbed. As most E-IoT systems are proprietary systems that define their own specifications for protocols using communication buses, we had to select a proprietary protocol which is a representative of E-IoT systems for this chapter. To ensure that we evaluate LightingStrike attacks realistically, we selected Crestron for implementation and evaluation. Crestron represents one of the most flexible and highly-deployed E-IoT systems available with 1.5 billion dollars in annual revenue [Mar18]. Further, 90% of Fortune 500 companies have some form of Crestron solution in their facilities. In addition to being one of the largest players in smart installations, Crestron highlights a commitment to security [Cre20f]. As such, Crestron is expected to be at or above industry security standards for E-IoT systems. Specifically, we chose Crestron's Cresnet for analysis, a proprietary serial-based communication protocol used in E-IoT integration and other smart use-cases.

We created an E-IoT testbed using common Crestron devices highlighted in Figure 7.2. Crestron proprietary software was used for testbed deployment and configuration. To configure the Crestron system, we utilized D3 Pro and Toolbox, the software used by integrators to deploy Crestron lighting systems. We utilized a Crestron DIN-AP3 processor as the primary logic unit of the testbed and a Crestron DIN-8SW8-I as the lighting module (8 controlled loads) for the testbed. Crestron devices utilize a proprietary communication protocol, namely Cresnet. To create a single Cresnet daisy-chain, we implemented two C2N-DB12W keypads using Cresnet

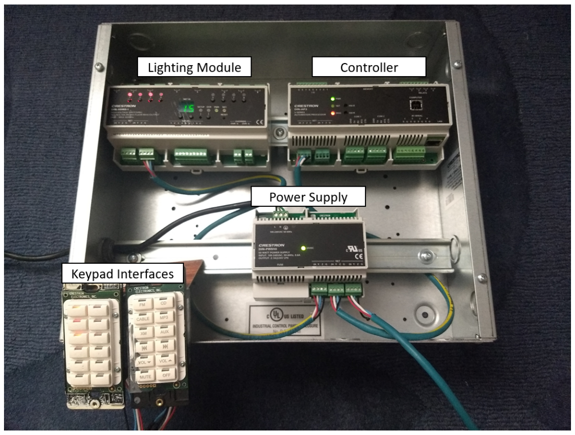


Figure 7.2: Testbed used to implement LightingStrike attacks, including a controller, power supply, smart modules, and keypad interfaces.

wiring. The Cresnet-based 12-button keypads and similar interfaces are common in public areas such as restrooms and are used as user interfaces to control lights, door access, relays, and any programmed functions. Finally, the entire system is powered by a Cresnet-based DIN-PWS50 power supply.

The Cresnet Bus. With the need to interface multiple devices together, proprietary communication protocols using communication buses are commonly used in E-IoT. Cresnet is a widely-used proprietary protocol of the popular Crestron E-IoT smart systems. Based on RS-485 communication, Cresnet is used to power, control, update, identify, and configure Cresnet-enabled devices (e.g., controllers, lighting modules, keypads, touchscreens). As a proprietary protocol, not much information is available on the technical specification of the Cresnet protocol. Cresnet comes in a 4-wire configuration, with 24 (red, power), Y (white, TX/RX+), Z (blue, TX/RX-), and G (black, ground) [Cre20a]. While there is no public specification given, we found the information to support that Cresnet communication operates at a 38400 baud rate and half-duplex operation [Cre06].

Cresnet ID. Devices in a Cresnet network are addressed with a network ID. The IDs are hexanumerical and range from 00 to FF. Cresnet IDs identify individual

Cresnet devices in the network. The integrator assigns unique Cresnet IDs to each Cresnet device during the configuration.

Cresnet Protocol Analysis. As Cresnet is largely undocumented, we faced a number of challenges in the analysis stage of this protocol. With no available documentation on the protocol, we referred to some existing open-source software and legacy manuals to find the proper baud rate and specification of the Cresnet protocol. While the data structure of Cresnet communication is not publicly accessible; our analysis shows that Cresnet devices place their Cresnet ID on packet headers. Further, our analysis of the protocol showed that Cresnet packets end in the hex code "02:00". Additionally, we found that Cresnet has two distinct types of traffic which we refer to as *idle traffic* and *active traffic.* Idle traffic is caused by packets that are transmitted when no user interactions are occurring. For instance, the Crestron controller periodically queries the existing Cresnet devices in the E-IoT system every 500ms. Active traffic occurs when users interact with a Cresnet device. Actions such as button presses or disconnecting a keypad generate such active traffic.

Software Module Implementation. In order to execute the LightingStrike attacks, we developed a number of software modules. The attacks were implemented using open-source tools available online. We used Java as the base language with RX-TX libraries for serial-based communication [RXT20]. Java was chosen as it is compatible with many platforms. Modules requiring communication rate specifications were configured with a 38400 baud rate, eight data bits, one stop bit, and no parity. The software modules are implemented into the attack device, as highlighted in our end-to-end implementation (Section 7.4).

1) Monitoring Module. The monitoring module was implemented in Java and the RX-TX library for RS-485-based communication. As such, the module executes as a loop that listens to Cresnet communication with the serial settings specified.

2) Injection Module. The injection module was implemented using RX-TX's `write()` function. The `write()` function allows us to inject any message as a hex string over the Cresnet bus.

3) Flooding Module. The flooding module is implemented using a Java loop and RX-TX broadcasting RS-485 packets over the Cresnet bus. This code was effective in causing a DoS condition in the target E-IoT system.

4) Re-addressing Module. To perform an attack, we used existing tools to allow an attacker to modify the configuration of Cresnet devices. This module is implemented through Crestron's D3 Pro proprietary tools, allowing us to reconfigure Cresnet IDs in interfaces.

5) Filtering Module. The filtering module was implemented as a Java character array ArrayList and a filtering component in the monitoring module. While software such as Wireshark exists, programmed filtering was sufficient for testing purposes since the protocol is proprietary.

7.6 LightingStrike Attacks Evaluation

In this section, we realize the LightingStrike attacks and analyze their effects on E-IoT systems. Additionally, we discuss the results and implications of individual attacks.

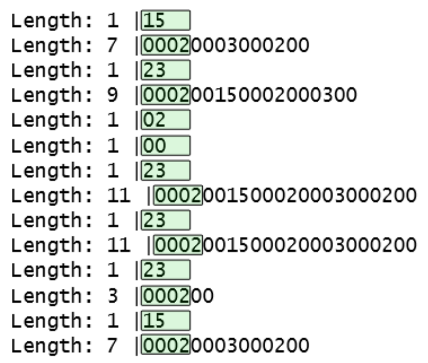
Attack 1: Flooding Denial-of-Service. This attack was created to demonstrate that Threat 1 (DoS, Section 5.3) is viable through LightingStrike by overwhelming the communication bus with messages.

Step 1 - Activation. Activation of this attack was executed through the attacker's client interface. This initiated the attack condition with the attack device and began the execution.

Step 2 - DoS Payload. As the attack executed, the attacker's adapter flooded the Cresnet bus with repeated RS-485 messages to overwhelm communications. This was accomplished with LightingStrike's flooding module which injected messages that did not even need to follow any special format to flood the Cresnet bus.

Evaluation. This attack was a complete success as all Cresnet communication was rendered inoperable just in a few seconds. This caused several notable negative impacts to the system. First, all keypads connected to the communication bus were inoperable, thus any control to any light or programmed event in the Crestron system became inaccessible. Second, the attack is not easily traceable; there were no messages or feedback from the system to notify a user or an integrator that the system was being attacked. The quick activation allows the attacker to easily control the availability of the E-IoT system on activation and de-activation. As a DoS attack is not easily identified, a user or technician may believe that there is a faulty component in the system. This DoS attack is presented as a proof-of-concept which can act as a part of more complex attacks.

Attack 2: Malicious Eavesdropping. Attack 2 demonstrates that Threat 2 (Malicious Eavesdropping, Section 5.3) is viable on the Crestron testbed. This attack used the monitoring and filtering modules to observe and infer information from Cresnet packets.



```
Length: 1  |15
Length: 7  |00020003000200
Length: 1  |23
Length: 9  |000200150002000300
Length: 1  |02
Length: 1  |00
Length: 1  |23
Length: 11  |0002001500020003000200
Length: 1  |23
Length: 11  |0002001500020003000200
Length: 1  |23
Length: 3  |000200
Length: 1  |15
Length: 7  |00020003000200
```

Figure 7.3: A snapshot of the captured periodic query-response traffic during Attack 2 (Malicious Eavesdropping) where the Cresnet IDs are highlighted.

Step 1 - Activation. With the attacker's device inserted into the Cresnet bus, the attack began through the attacker's client machine. It started with the activation of the Java client and the monitoring module began to sniff packets over the bus.

Step 2 - Monitoring. Using the monitoring module active, the attacker eavesdropped on active communication occurring through the Cresnet bus. The monitoring module operated in a loop, in an independent thread and captured Cresnet communication.

Step 3 - Analysis and Filtering. During this step, we analyzed the ongoing communication and performed traffic filtering using the filtering module. This allowed us to filter the periodic query-response traffic between the Crestron controller and the devices. Such filtered messages allowed us to capture the unique Cresnet IDs 00, 02, 03, 15, and 23 in the network.

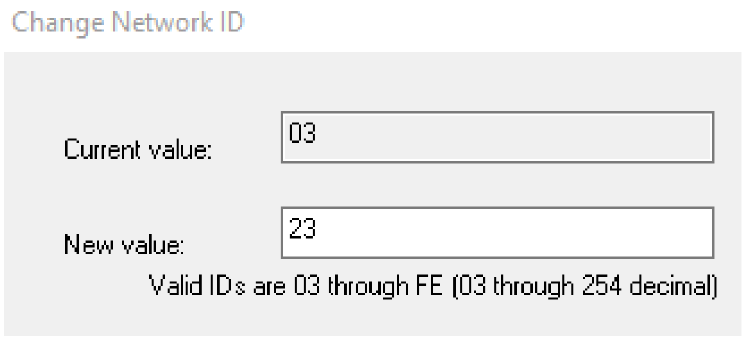
Step 4 - Final Analysis. After the periodic query-response traffic was filtered, we focused on the remaining E-IoT traffic and we observed spikes in data packets when user actions were occurring. The spikes signal Cresnet activity such as button presses, disconnected keypads, or activity occurring in other rooms.

Evaluation. The attack monitored and gathered information from the Cresnet bus successfully due to Cresnet's lack of encryption. We highlight a snapshot of the captured Cresnet communication in Figure 7.3, where communication is in the order it was received. First, monitoring the Cresnet bus easily allowed us to gather Cresnet IDs. As such, an attacker can easily know how many Cresnet devices are connected to the communication bus through their unique IDs. Our eavesdropping revealed at least four unique devices: the keypads, the lighting module, and the controller. We could observe spikes of activity when keypad buttons were pressed and other actions were executed on the bus. An attacker can use this information to infer building occupancy by identifying keypads in specified locations and listening for events originating from the associated Cresnet ID. As the attack was performed through passive monitoring, no alarms, or any unexpected behavior occurred in the Cresnet bus or any of the dependent devices (e.g., keypads, controller).

Attack 3: Impersonation-based Denial-of-Service Attack. This attack was designed to demonstrate another form of DoS attack using Threat 1 and Threat 3 (DoS and Impersonation, Section 5.3) through LightingStrike. As such, we accomplished a DoS condition by creating an ID conflict between devices. The attack takes advantage of Cresnet's identification phase when a new device is added to the system. Our research showed that new devices broadcast several packets upon connection and Cresnet relies solely on the Cresnet ID to identify the individual devices, button presses, and other actions.

Step 1 - Removal. We initiated the attack by disconnecting a keypad (Cresnet ID: 03) from the network bus. Disconnection only disables that keypad as it is offline. Disconnection is trivial as it can be done by simply removing a keypad from the wall with a screwdriver and disconnecting the physical Cresnet connector.

Step 2 - Modification. Using a spare controller and software, we altered the Cresnet ID of the keypad to "23" without any form of validation as shown in Figure


Change Network ID

Current value: 03

New value: 23

Valid IDs are 03 through FE (03 through 254 decimal)

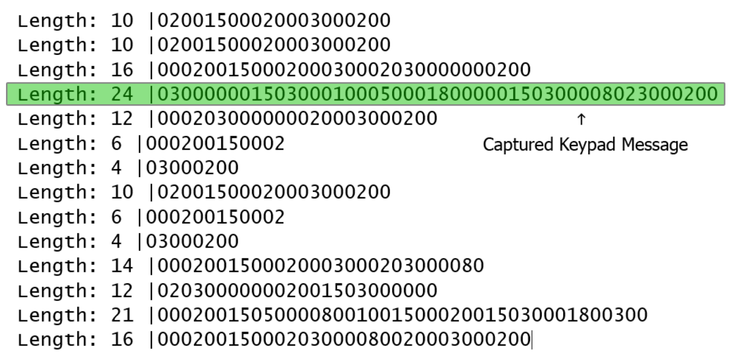
Figure 7.4: Cresnet ID change window in D3 Pro Crestron software during Attack 3 (Impersonation-based DoS).

7.4. We changed the ID of a keypad with the ID of an existing keypad to purposely cause an ID conflict in the Cresnet bus. Additionally, this was also accomplished by physically replacing the original keypad with an identical keypad with a pre-assigned ID.

Step 3 - Re-connection. The altered keypad was reconnected to the Cresnet bus, where the keypad attempted to advertise itself as a new keypad with ID “23”. This caused an internal conflict in the Cresnet network between both keypads, causing the keypads to fall offline.

Evaluation. The attack was completely successful. When the conflicting keypad was connected the Cresnet bus, both keypads caused conflicts and stopped functioning. As such, this can act as a form of targeted attack over the communication bus. Additionally, an advantage is that our attack device in this case could also be an existing keypad and would only require re-addressing an available keypad to cause a DoS condition.

Attack 4: Replay Attack. This attack was created to demonstrate that Threat 4 (Communication Replay, Section 5.3) is possible on the Cresnet bus. Further, we highlight the implications of replay attacks on E-IoT systems.



```
Length: 10 |02001500020003000200
Length: 10 |02001500020003000200
Length: 16 |00020015000200030002030000000200
Length: 24 |030000001503000100050001800000150300008023000200
Length: 12 |000203000000020003000200
Length: 6 |000200150002
Length: 4 |03000200
Length: 10 |02001500020003000200
Length: 6 |000200150002
Length: 4 |03000200
Length: 14 |0002001500020003000203000080
Length: 12 |020300000002001503000000
Length: 21 |000200150500008001001500020015030001800300
Length: 16 |00020015000203000080020003000200
```

Figure 7.5: A snapshot of the messages captured from Cresnet bus during Attack 4 (Replay Attack).

Step 1 - Activation. With the attack device connected to the Cresnet bus, the attack was initiated through the attacker's client using the monitoring module.

Step 2 - Monitoring. With the monitoring module active, we awaited keypad press commands relayed through the Cresnet bus. The Cresnet communication was recorded for later analysis.

Step 3 - Analysis. After the initial capture, we determined the packets issued from the device with Cresnet ID 03. We successfully identified the Cresnet packet transmitted during a button press to activate a programmed event (powering on light 1 in the lighting module). The specific packet was identified from the header "0300", which notes the message from ID 03 (keypad) to ID 00 (controller).

Step 4 - Replay. With the packet captured, the same packet was replayed using a user interface in the attack client. As such, the message was then injected into the Cresnet bus. The testbed reacted by powering on light 1 on the DIN-8SW8-I as if the button had been pressed.

Evaluation. We evaluated this attack on the success of replaying button presses from a Cresnet keypad. As such, the attack was entirely successful in replaying button presses on the Cresnet bus. The initial monitoring phase for messages was successful as the packets were captured during physical button presses. Step 2 of the attack can be observed in Figure 7.5. In this step we could easily observe spikes in activity on button presses. With the messages captured, our attempts to replay a button press (button 1) were successful, with the testbed reacting by turning on a light associated with button 1 as if the button had been pressed. Further, we could use the same captured packet to turn on the light again, demonstrating there is no replay protection in the Cresnet bus. The implications of this attack show that an attacker could capture a button press to unlock an Crestron-controlled door with the same captured code, or ultimately assume control of integrated devices by replaying the associated button presses.

Summary As LightingStrike attacks were developed and tested, we demonstrated that insecure proprietary protocols can have many negative impacts on the security of E-IoT. From our attacks, we concluded that without any form of security beyond obscurity, a knowledgeable attacker can easily compromise E-IoT to their benefit. All of the proposed attacks were implemented successfully, the implications of which clearly show the potential of LightingStrike attacks. In Attack 1 and Attack 3, we demonstrated that multiple Cresnet-based interfaces can be disabled by an attacker. This is a viable form of preventing access to any user-controlled systems through a DoS attack. As E-IoT manages light control, gate access, and other essential components, an authorized user can be prevented from operating a connected system through the attack proposed in our examples. Further, programmed events such as panic buttons will not execute while a DoS attack is active on the affected interfaces. In Attack 2 we showed that an attacker can capture communication between multiple devices from a single point of connection, the implications of which can be abused by an attacker. With Attack 3, creating a Cresnet ID conflict would be no issue for attackers as all the source and destination addresses are broadcasted over the communication bus. Further, if an attacker has an idea of which Cresnet IDs belong in which locations, they can infer which room is occupied. As button presses and messages are broadcasted no matter where the keypad is located, an attacker can infer information on unauthorized locations and query equipment unreachable via traditional means (e.g., TCP/IP, WiFi, Bluetooth). As Attack 4 (replay attack) was successful, we show that an attacker can severely compromise the security of Crestron systems. For instance, if an attacker manages to re-address a keypad using a replay attack, it is possible to reprogram a number of devices. Understanding the Cresnet protocol through further reverse engineering may allow future attacks through generating Cresnet packets without the need for capture and replay.

7.7 Attack Discussion

In this section, we outline the findings and contributions, and discuss the possible defense mechanisms and the corresponding challenges.

Findings. In this chapter, we explored the security of E-IoT and found some inherent vulnerabilities of proprietary communication protocols in E-IoT. Specifically, we focused on Crestron, one of the most popular E-IoT vendors. We found that the widely-used Cresnet protocol does not have any security mechanisms to ensure confidentiality, authenticity, integrity, or access control. As such, we found some notable vulnerabilities in the Cresnet protocol. First, we found that Cresnet has no encryption in the protocol, allowing any device in the Cresnet bus to capture and monitor ongoing traffic. Thus, it is trivial for an attacker to observe communication, collect IDs, and infer user interactions. Second, there are no rate-limiting functions, allowing an attacker to easily flood the communication bus and cause DoS conditions. Third, Cresnet message integrity is not guaranteed. Without timestamps or signatures, an attacker can replay any message anytime as the protocol does not reject older messages. An attacker can easily abuse the protocol and replay malicious packets with the E-IoT system accepting them as legitimate. Finally, Cresnet devices do not have protections against unauthorized modification. For instance, re-addressing a keypad for Crestron E-IoT can be done without any form of authorization. An attacker can reconfigure devices to cause a DoS, or simply disrupt E-IoT operation by altering the IDs of multiple devices. Once the devices have been altered, the Crestnet-based system cannot self-recover. In addition, the end-users cannot fix this issue and will be forced to contact their integrator at a cost of time and money. LightingStrike demonstrates that it is trivial for an attacker to use communication buses in order to compromise E-IoT through proprietary protocols. These threats could be critical, as E-IoT systems such as Crestron, control emergency equipment and physical access, the consequences of which may be as costly and dangerous as well-known attacks with a low level of effort and knowledge from an attacker.

Contributions. Our research provided several contributions towards the security of E-IoT systems. As mentioned in Section 7.5, we emphasized that the Crestron brand is currently one of the most popular E-IoT systems available worldwide with 1.5 billion dollars in annual revenue [Mar18]. With this chapter, we demonstrated that unsecured proprietary communication protocols used in E-IoT systems can lead to downtime, a breach of privacy, and a breach of physical security. Additionally, this chapter aimed to raise awareness on lesser-known but widely-used protocols in E-IoT systems such as Cresnet. As such, we highlighted vulnerabilities found during our research, and how the lack of common security mechanisms in proprietary-protocols can easily lead to critical vulnerabilities in E-IoT systems. We used the LightingStrike proof-of-concept attacks to expose these vulnerabilities, and presented several practical attacks that can contribute towards more complex, larger attacks. LightingStrike introduced a new threat vector, so that future iterations of E-IoT systems and their proprietary protocols can be designed with secure practices in mind for communication buses. Since systems such as Crestron have been deployed for decades, we expect that this chapter can motivate further research on E-IoT attacks, security mechanisms, proprietary protocol security, highly-deployed threat vectors, and other popular E-IoT systems that have not received any form of security scrutiny.

Possible Defense Mechanisms and Challenges. Security for E-IoT must go beyond ensuring the confidentiality, integrity, and availability but also must consider the challenges associated with E-IoT design, proprietary architecture, physical security, and software security. One of the biggest challenges in securing proprietary E-IoT communication is that most of these systems are closed-source that use custom protocols. Without specifications available on most proprietary protocols, the packet structure, exception cases, and communication process have to be reverse engineered. Further, depending on the system and software versions, the implementation of proprietary protocols can differ. Additionally, many E-IoT systems require backward compatibility, making some traditional solutions that patch existing protocols with security mechanisms (e.g., encryption, signatures, timestamps) difficult to deploy on older systems. Ideally, proprietary communication protocols should follow a secure communication standard stack and implement vendor-specific functions. However, a standard would require an agreement between the top E-IoT vendors. For newer E-IoT systems, eavesdropping can be remedied by enabling encryption in the network. Additionally, newer protocols should protect from impersonation and replay attacks through the use of timestamps and signed messages. Physical-based mitigation strategies can also be helpful as there are physical actions an attacker must take to compromise E-IoT devices. For instance, E-IoT systems can use broadcast messages from devices to identify when a keypad is removed or tampered and notify administrators before an attack occurs. Such a design could expand into a live-mode where any modifications to any devices notify administrators as tampering. Further, it can be possible to segment daisy-chain lines depending on the location of interfaces (e.g., all devices of a sensitive location on one line, all devices in public locations on another line). Sensor-based solutions can also prevent physical tampering, as well as to provide a context-aware solution to button presses. For instance, certain messages should only be received if there is a sensor activity near the user interface.

7.8 LGuard

To secure E-IoT systems against LightingStrike threats, we introduce LGuard, an easily-configurable defense system designed to protect E-IoT communication buses using traffic analysis, computer vision, and traffic obfuscation. In this section, we firstly discuss the design considerations and challenges for LGuard. We then define the necessary terminology and introduce our additional findings on the Cresnet protocol over our prior study [RBA+21]. Finally, this section details the LGuard architecture and its individual components.

7.8.1 Design Considerations and Challenges

In this sub-section, we explain the distinct features of E-IoT systems that make it challenging to employ existing schemes to protect against threats over the E-IoT communication buses, therefore necessitate a specialized solution like LGuard.

Closed-Source E-IoT. E-IoT systems are very often closed source, with no source code or technical documents available to the end-user or integrators. Thus, a defense strategy must work without relying on the source code constructs, system hooking, or modification to the E-IoT system. LGuard is designed with these limitations in mind on top of an existing E-IoT system. In this manner, an integrator or an end-user would be able to deploy LGuard on an E-IoT system without modification to the underlying code.

Legacy Systems. As E-IoT systems have existed for decades, there are many deployments that are either outdated and unsupported. As buildings and homes were pre-wired for many legacy E-IoT systems, modification (e.g., rewiring) may be costly or impossible for users looking to install new systems. For instance, homes wired for panelized lighting are wired differently than traditional homes, as they often have high voltage running to interfaces (e.g., keypads, touchpanels). Converting a system such as this to traditional electrical wiring would require substantial labor and cost. As such, a defense mechanism is needed to protect these systems without the need for costly and impractical solutions.

Bus Architecture. The architecture of E-IoT communication buses allows attackers to easily eavesdrop and compromise the communication from any single point. An attacker can easily replay or spoof packets which causes a defense mechanism fail to distinguish the origin of a packet. Indeed, E-IoT protocols such as Cresnet rely on bus architecture. Although it causes challenges for defense systems, the same attributes of bus-based communication can also be used to a defense system's advantage. For instance, an attacker's replayed or spoofed messages destined to any E-IoT device will be received by all the devices connected to the bus. Therefore, a defense system like LGuard can monitor all messages in the E-IoT bus from a single point.

Modification Costs. As the aforementioned characteristics of E-IoT allow an attacker to compromise E-IoT systems through communication buses, defense solutions may consider relying on additional data sources such as motion, proximity, or touch sensors to detect the attacks. Although this is a viable option, it comes with additional device and labor costs for end users. In addition, an attacker can also compromise such additional devices to evade the detection system. However, E-IoT systems are often installed with other integrated components such as Closed-Circuit Television (CCTV) and alarm systems. Especially, systems such as these may be integrated if the E-IoT system is installed in a sensitive location. A defense mechanism may leverage access to integrated security systems to defend against LightningStrike attacks.

Security Systems. LGuard takes two assumptions. First, that there is a CCTV system present in the location the E-IoT system is installed. This is a valid and feasible assumption as E-IoT systems are often designed to integrate Closed-Circuit Television (CCTV) systems and other security systems. For instance, some of the largest E-IoT system vendors, advertise CCTV control and integration as one of the core capabilities of their systems with several compatible CCTV systems [Con21b, Con21a, Cre21a, Cre21b]. Further, CCTV systems are expected to be present to be installed in secure locations. Second, that at least one of the CCTV cameras can see the E-IoT interface keypads. This is a viable and practical assumption, as security cameras can be adjusted to face different keypads. Further if no cameras are available, wireless cameras may be installed for LGuard installed in sensitive locations.

7.8.2 Terminology

This sub-section introduces the necessary terminology to understand LGuard.

Interface Interaction. We define interface interaction as any interaction between a user and an E-IoT interface. For example, a user pressing a button to turn on the light.

Tampering. We define tampering as any manipulation or modification of an E-IoT device (e.g., interface) or communication bus wiring such as the unauthorized removal of a keypad.

7.8.3 LGuard Overview

LGuard is a novel defense system against LightingStrike-style attacks in E-IoT deployments. It consists of tailored solutions against individual LightingStrike attacks. First, LGuard identifies the removal of E-IoT interfaces and excessive network traffic on the bus to detect impersonation and DoS attacks. Second, LGuard benefits from the existing CCTV system to detect replay attacks. Specifically, using computer vision, it performs pose estimation on the CCTV video footage to determine if messages received from an E-IoT interface were caused by a replay attack. Finally, LGuard mitigates eavesdropping attacks by obfuscating Cresnet communication with the insertion of redundant communication packets from non-existing E-IoT devices.

The proposed LGuard architecture consists of five distinct components as shown in Figure 7.6. The first component is the Serial Collector ❶, which connects directly to the communication bus of the E-IoT system. It inserts the redundant traffic generated by the Obfuscator component to the bus and also collects the traffic and feeds it to the Data Handler. The Obfuscator component ❷ is used by LGuard

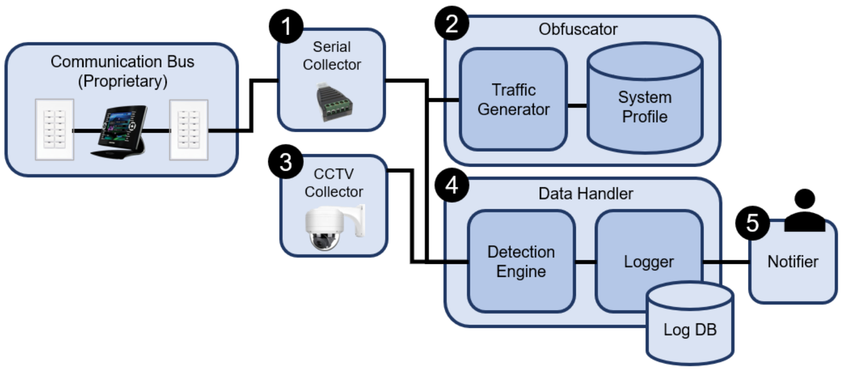


Figure 7.6: LGuard architecture, components numbered.

to generate redundant communication packets to obfuscate the E-IoT traffic. It obfuscates the E-IoT traffic using two sub-components: the System Profile and the Traffic Generator. The System Profile sub-component contains the E-IoT system information and the device details which are used by the Traffic Generator sub-component to generate redundant traffic. The generated traffic is fed to the Serial Collector to be inserted to the communication bus. The third component of LGuard is the CCTV Collector ❸. This component connects to the CCTV system and transfers video captures for LGuard upon a request from the Data Handler. The Data Handler ❹ is the core of LGuard and is used for detection of the LightingStrike attacks. It consists of Detection Engine and Logger sub-components. The Detection Engine obtains the traffic of the E-IoT communication bus via Serial Collector and detects DoS, impersonation, and replay attacks via three tailored solutions. When an attack is detected by the Detection Engine, the details with the attack are passed to Logger sub-component. The Logger stores the relevant attack data to its Log DB and also forwards attack information to the Notifier component ❺, which finally notifies the users about the attack.

7.8.4 Serial Collector

The Serial Collector allows LGuard to collect and transmit data packets to the communication bus, acting as the main interface between LGuard and the E-IoT system's communication bus. As such, to capture and transmit packets, the Serial Collector includes a physical interface connected directly to the communication bus. The Serial Collector is also responsible for pre-processing communication packets into a format that LGuard can use. It contains a message buffer to process concatenated, partial, and invalid packets received on the bus and format them. These formatted packets include the raw packet information, timestamp, length, and any other attributes necessary for LGuard to detect the LightingStrike attacks. Serial Collector also transmits the redundant E-IoT traffic generated by Obfuscator component as needed.

7.8.5 Obfuscator

The Obfuscator component obfuscates the E-IoT traffic against eavesdropping attacks for LGuard and includes two sub-components: the System Profile and the Traffic Generator. As an eavesdropper aims to obtain valuable information by listening the E-IoT communication bus traffic, Obfuscator component intends to make the job of the attacker harder by generating and inserting redundant traffic to the bus. The logic of Obfuscator is if the bus has the traffic of n real E-IoT devices, then the Obfuscator generates the same amount of redundant traffic to show that there are $2n$ devices in the E-IoT deployment. Hence the probability of obtaining valuable information for the eavesdropper reduces, which means the adversary cannot easily discriminate legitimate traffic from redundant traffic.

System Profile

The System Profile sub-component contains all the information (e.g., IDs of the existing E-IoT devices, reserved device IDs) necessary for the Traffic Generator to generate packets. If the E-IoT system has n devices, then this sub-component creates n additional Cresnet IDs representing non-existing E-IoT devices. The System Profile sub-component always includes the IDs of all the existing E-IoT devices communicating over the bus, and must be updated when new devices are added to the deployment. As such, the System Profile sub-component should be flexible and easily-modifiable by the administrator or integrator that perform the original configuration.

Traffic Generator.

The Traffic Generator sub-component aids LGuard in transmitting redundant traffic into the E-IoT system's communication bus. This additional traffic intends to make eavesdropping more difficult for an attacker. The Data Handler refers to the System Profile sub-component to generate data packets then transmit them to the E-IoT system's communication bus through Serial Collector. Packets are only be transmitted when LGuard detects user activity. To mimic interface activity, the Traffic Generator first loads a random Cresnet ID from a non-existent device and generates idle and active traffic (sub-section 2). To do so, the Traffic Generator first generates redundant to mimic the controller pinging dummy device. LGuard then generates redundant communication packets to mimic keypad-to-controller communication as active packets every 500ms. These packets are ignored by the E-IoT controller for two reasons: first, the devices do not exist, thus the controller has no programming for these devices. Any packet originating from an invalid device is dropped. Second,

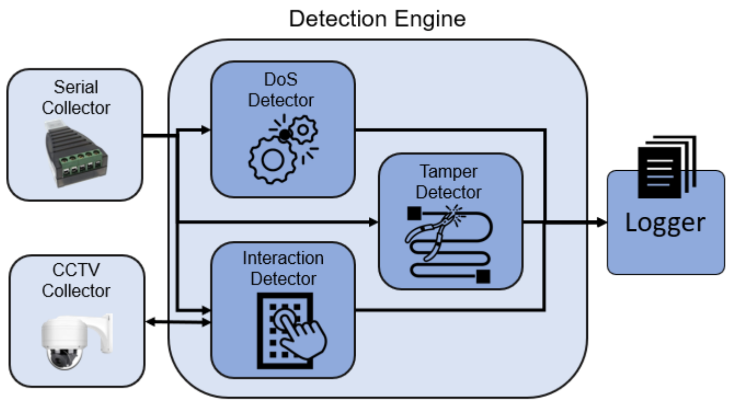


Figure 7.7: Architecture of the Detection Engine component of LGuard that detects DoS, impersonation, and replay attacks via DoS Detector, Tamper Detector, and Interaction Detector modules respectively.

E-IoT systems do not allow new keypads to be added without re-configuring the controller, as such new device ID's cannot be added by an attacker.

7.8.6 CCTV Collector

The CCTV Collector acts as the main interface between LGuard and the E-IoT CCTV system. The CCTV Collector receives requests for video capture data and passes this information to the Data Handler for evaluation. Specifically, this video data is then used by to LGuard to perform all necessary valuation of E-IoT communication bus traffic.

7.8.7 Data Handler

The Data Handler performs the communication bus traffic analysis and attack detection for LGuard. It is composed of two sub-components: the Detection Engine and the Logger.

Detection Engine

The Detection Engine is one of the core sub-components of the LGuard defense system and performs the bulk of the detection process. As shown in Figure 7.7, the Detection Engine includes three defensive modules each specifically designed to address different LightingStrike attacks. As the name of it implies, the first defense module, DoS Detector, aims to detect DoS attacks. The second module is the Tamper Detector which intends to address the impersonation attacks. The third module is the Interaction Detector that performs replay attack detection. Details with the modules are as follows.

DoS Detector. Detects LightingStrike DoS attacks over the E-IoT communication bus by monitoring the network traffic and checking the size and number of packets received in a short time interval. As LightingStrike DoS attacks depend on high throughput, anomalies in the size and number of packets received indicate that a DoS attack is being attempted.

Tamper Detector. Detects impersonation attacks via identifying tampering (e.g., removal) of an E-IoT interface. It achieves this via examining the E-IoT communication bus traffic for messages transmitted when E-IoT devices fall offline. For instance, the Cresnet controller will request a response from all interfaces at 500ms intervals. If a device does not respond, the controller starts to constantly query the related E-IoT device. Tamper Detector identifies the impersonation attacks via detecting these packets in the E-IoT bus.

Interaction Detector. Receives both the communication bus traffic and the CCTV data necessary to determine if communication packets are legitimate or created through a replay attack. As an attacker cannot physically interact with a targeted E-IoT interface at the time of replay attacks and CCTV cameras are common in secure locations, Interaction Detector uses computer vision and pose estimation to determine if E-IoT traffic is legitimate or replay. For instance, if packets are detected from keypad 23, CCTV information should display physical interactions with keypad 23 within seconds. As such, Detection Engine monitors Cresnet packets in the communication bus, identifying messages transmitted during events such as button presses. Once an event is detected, the Detection Engine requests a video capture from the CCTV Collector of the moment the interface should have been touched by a user. The observed packets are deemed benign when a video capture shows that a user interacted with the E-IoT interface at the time the message from that interface was received.

Logger

The Logger component receives detection results, CCTV video captures, and the related packet data from the Detection Engine. As such, the Logger component acts as an intermediary step between the LGuard data and Log DB. The Logger component is responsible for formatting important information from LGuard (e.g., detected attacks, errors and caught exceptions with LGuard) and storing this information in the Log DB. Finally, the Logger component allows LGuard administrators and users to view a history of occurrences and ongoing network communication, and enable them to review any activity that was deemed to be attack by LGuard. The Log DB sub-component acts as the primary storage database for data and information for LGuard monitoring. A user or administrator can query the Log DB component to view the malicious activity detected by LightingStrike. Thus, the Log DB component should store communication packets and video feed used during the evaluation process that may be relevant for the administrator.

Table 7.2: Hardware and Software used in LGuard implementation and testing.

Hardware	Software
Raspberry Pi 3b	Python 3.9
Raspberry Pi Camera	OpenCV Python 4.5.3
GearMo Mini USB to RS485	Visual Studio Code 1.55.2
Razer Blade 15 Laptop	VNC Viewer 6.20.529
Acer GX-785 Desktop	Google MediaPipe 0.8.7.3
	Redis 3.2

7.8.8 Notifier

The Notifier component notifies users or administrators about suspicious activities using warnings and notifications from LGuard. After traffic analysis is conducted, the Notifier component notifies the user if any attack activity has been found. Thus, the Notifier component should give users all the information necessary to evaluate and act upon malicious activity occurring in the communication bus. Finally, the Notifier component is responsible for mobile (e.g., text, in-app) notifications sent to the user.

7.9 LGuard Implementation

To implement the necessary components for LGuard we used open-source libraries and easily obtainable hardware. We detail the software and hardware used by LGuard in Table 7.2. Our testing E-IoT environment is identical to the environment used for the LightingStrike implementation. We assume that the attacker executes the LightingStrike attacks by inserting a malicious device into the E-IoT bus and executing the attacks. The implementation details with the components of LGuard are as follows.

7.9.1 Serial Collector Implementation

The Serial Collector was implemented using the Gearmo Mini USB adapter and the Python serial library to collect raw E-IoT communication bus data. Using Python, this information was then pre-processed and added to a data buffer. The data buffer is then processed with the end of packet delimeter '02:00' to separate individual packets. The Serial Collector shares the preprocessed data and current size of the buffer with the Data Handler.

7.9.2 Obfuscator Implementation

The Obfuscator component contains two sub-components, the Traffic Generator and the System Profile, that are used to insert redundant traffic into the E-IoT communication bus to obfuscate the legitimate E-IoT traffic. In this sub-section we cover the implementation of these sub-components.

System Profile Implementation

The System Profile was implemented using a table of the Cresnet IDs of the existing E-IoT devices and also the reserved Cresnet IDs (e.g., 00, 01, 02). As the test environment has four real Cresnet devices (03, 13, 15, 23), the table has the IDs of these devices and an additional four unused Cresnet ID's (06, 16, 26, 36) that are used for the Traffic Generator's packet generation. This table is easily expandable to add or remove Cresnet devices in the E-IoT system.

Traffic Generator Implementation

The Traffic Generator sub-component was implemented using the Python serial library to receive and transmit Cresnet packets. Traffic Generator implementation generates redundant traffic including both idle E-IoT traffic and active E-IoT traffic. For the redundant idle traffic, Traffic Generator generates idle traffic for every non-existing E-IoT device ID in every 200ms. In this respect, first, the Traffic Generator generates idle packets for each non-existent device (e.g., 33:00:02:00 for device ID 33) in the System Profile list and passes the packets to Serial Collector. In terms of redundant active traffic generation, Traffic Generator follows a probabilistic packet generation approach. Specifically, in 200ms intervals, Traffic Generator decides to generate a redundant active traffic according to a probability function. With 0.2 probability, it generates a redundant active traffic for a non-existing E-IoT device ID. We selected this probability value because our analysis showed that the redundant active traffic generated using this value does not disrupt the legitimate E-IoT communication. If the conditions are met, the Traffic Generator then generates redundant activity traffic packets for a non-existing E-IoT device ID in the list, mimicking button presses and controller responses. The redundant packets are passed to Serial Collector to be sent to the E-IoT communication bus. We would like to note that, the generated redundant activity traffic does not cause any issues as the controller ignores such messages.

7.9.3 CCTV Collector Implementation

The CCTV Collector was implemented using a Raspberry Pi 3b, with an integrated camera and Python scripts to act as a CCTV source for LGuard. The CCTV Collector communicates with the Data Handler through an intermediary Redis server. As such the CCTV Collector polls the Redis server for new requests to record using a Python script. Once a request to record is received from the Data Handler (e.g., from a button press detected by the Data Handler), the CCTV Collector initiates

Table 7.3: Observed Cresnet Communication

Packet Start	Description
ID:00:02:03:00:...	Button press/release on keypad ID
ID:00:00:FF	Keypad ID removed and being queried
ID:00:02:00	Idle traffic from ID

recording and saves a twenty-second video capture locally as a '.h264' video file at an average of 25 frames per second. These video captures are then passed to the Data Handler for analysis.

7.9.4 Data Handler Implementation

In this sub-section we detail the implementation of the Data Handler and its sub-components: the Detection Engine and the Logger.

Detection Engine Implementation

To implement the Detection Engine component, several external Python libraries were used for computer vision and image processing such as `MediaPipe` and `OpenCV` [LTN+19, Bra00]. The Detection Engine first identifies predetermined activities occurring in the communication bus as highlighted in Table 7.3. For instance, a packet starting with 23:00:02:03:00 will be observed when a button is pressed or released on Cresnet keypad ID 23. Using these known packet features, the Detection Engine can execute three types of detectors: *DoS Detector*, *Tamper Detector*, and *Interaction Detector*.

DoS Detector. As LightningStrike DoS attacks depend on large volume of invalid messages, LGuard detects DoS attacks by examining the size of incoming data packets. We performed extensive analysis of E-IoT Cresnet communication and our observations showed that benign Cresnet traffic does not exceed 1024 bytes per second, even under frequent usage of interfaces. The DoS detector was thus implemented by detecting packet rates larger than 1024 bytes per second. When transmission exceeds 1024 bytes, the DoS detector reports an attempted DoS attack.

Tamper Detector. Impersonation LightingStrike attacks occur by tampering with the E-IoT devices. In this regard, Tamper Detector aims to detect such activities through passive analysis of Cresnet traffic. In Cresnet, the controller periodically queries Cresnet devices. If a device does not reply to the query, the controller queries again using a specific Cresnet packet header. For instance, for the E-IoT device with ID 13 that does not respond to the query of the controller, the controller starts to send queries starting with 13:00:00:FF. Hence, Tamper Detector detects that the E-IoT device with ID 13 has been removed from the communication bus.

Interaction Detector. Interaction Detector aims to detect replay attacks. To implement Interaction Detector, Detection Engine monitors Cresnet packets in the communication bus, identifying traffic events such as button-presses. When an event is detected LGuard intends to determine whether it is a legitimate event or a replay attack. To answer this question, LGuard takes the timestamp of the observed event packet and forwards a message to the CCTV Collector, requesting a video capture at the given time of the interface interaction. When CCTV Collector sends the video capture, Interaction Detector uses computer vision techniques to determine if a person is touching the interface in the video captures. This process requires Interaction Detection to have a prior knowledge of the X and Y coordinates of the interface in each CCTV frame. We assume that the administrator can enter the X and Y coordinates of the E-IoT interfaces in the CCTV video frames to LGuard during setup. Since the location and position of E-IoT interfaces and CCTV rarely change in E-IoT deployments, this one-time process can be performed by administrators. Having the prior knowledge of X and Y coordinates of the E-IoT interfaces

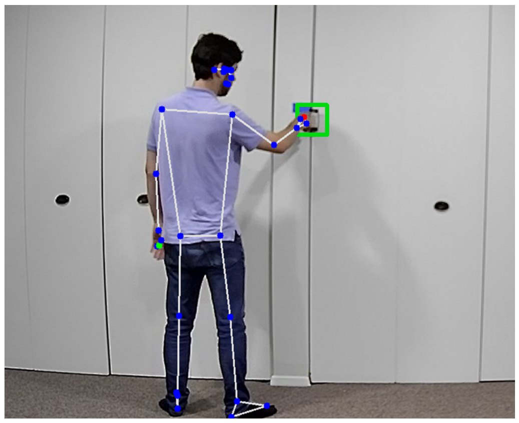


Figure 7.8: LGuard pose detection on a keypad from CCTV footage. Red highlighting right hand, green point highlighting left hand. The green square highlights the interface location.

in video frames, Interaction Detector performs the following steps to determine if the event is a replay or legitimate:

1. For each frame in the CCTV capture, the Interaction Detector first identifies the pose vertices of any person in the current frame, specifically the left and right hand vertices using the Google MediaPipe pose recognition library. This process is depicted in Figure 7.8 where the green square highlights the E-IoT interface location and red highlighting signifies the right hand.
2. The proximity of left and right hand vertices to the known keypad X and Y coordinate locations are calculated. If the distance is less than the predefined tolerance value, the Interaction Detector notes this frame as a user-to-interface interaction.
3. The number of interactions are counted for every frame. If the number of interactions are greater than a threshold value, the received event packets are deemed benign. Otherwise, a replay attack is detected.

As explained in steps of the detection process, Interaction Detector requires a a tolerance value for proximity of left and right hand vertices to the known coordinates of the E-IoT interface. The tolerance should be adjusted to the relative pixel-size of the interface. For instance, if a tolerance is set to 10 pixels, an interface will be considered 'touched' if the left or right hand vertices come within 10 pixels of the known X and Y coordinates of the interface. For our implementation, the pixel tolerance was configured to 25 pixels.

Logger

The logger sub-component was implemented as a local software buffer collecting all the logs and warnings from the LGuard system and detection process. These logs are then exported using Python JSON serialization to the Log DB. The Log DB sub-component was implemented using the Python-based JSON serialization and IO libraries to export plaintext logs on the running machine. The stored plaintext logs can then be viewed for future reference and contain all the relevant information for LGuard and detection results (e.g., timestamps, packets, warnings). We would like to note that, Log DB can be implemented using known database schemes such as MySQL, Redis, MongoDB.

7.9.5 Notifier Implementation.

The Notifier component was implemented using the Python-based `ctypes` library to create a notification window on the LGuard computer. These notifications can be modified depending on the detected attack.

7.10 Performance Evaluation

We evaluate the performance of LGuard in detecting LightingStrike attacks and answer the following research questions:

RQ1: DoS Detection. How effective is LGuard's performance in LightingStrike DoS attack detection (Sub-section 7.10.2)?

RQ2: Impersonation Detection. How well does LGuard identify tampering in the communication bus through passive monitoring (Sub-section 7.10.3)?

RQ3: Replay Attack Detection. How effective is LGuard at identifying replay attacks using pose estimation and traffic monitoring (Sub-section 7.10.4)?

RQ4: Traffic Obfuscation. How well does traffic obfuscation mitigate LightingStrike-type eavesdropping (Subsection 7.10.5)?

In this section, we firstly explain the attack data collection process. Afterwards, we answer each research questions and finally evaluate the detection time and overhead of LGuard.

7.10.1 Attack Data Collection

For LGuard evaluation we applied the LightingStrike attacks as specified in Section 7.5. We collected Cresnet traffic over the E-IoT communication bus. The activities collected for our evaluation included Cresnet traffic caused by the LightingStrike attacks and benign traffic generated by expected usage of the E-IoT system. Additionally, traffic data collected includes the CCTV recordings of the keypads for the duration of the logging. We would like to note that we considered different CCTV views and light conditions in our evaluations that are depicted in Figure 7.9. Details with the executed attacks are as follows:

- 20 replay attacks and 20 benign cases in bright light conditions (front-view).
- 20 replay attacks and 20 benign cases in low-light conditions (front-view).
- 20 replay attacks and 20 benign cases in bright light conditions (side-view).
- 20 DoS and 20 impersonation attacks.

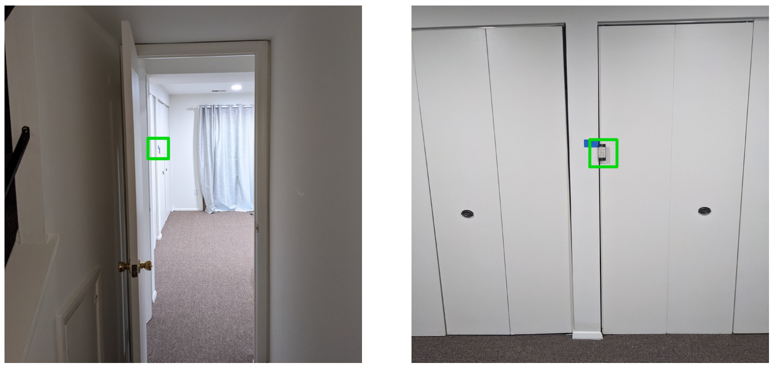

(a) Side-view of Cresnet E-IoT interface. (b) Front-view of Cresnet E-IoT interface.

Figure 7.9: Side and front views of CCTV used for LGuard evaluation, keypads highlighted in green. Different angles were tested to evaluate pose estimation efficacy.

- 20 LGuard logs with obfuscated traffic (2000 packets each).
- 20 logs without obfuscated traffic in which each log consists of 2000 packets.

Performance Metrics

Performance metrics are measured with the following parameters: accuracy, precision, F-score, recall, True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), and False Negative Rate (FNR).

True Positive Rate (TPR). denotes the total number of correctly identified benign test cases within the test environment.

True Negative Rate (TNR). denotes the total number of correctly identified malicious test cases within the test environment.

False Positive Rate (FPR). denotes the total number of cases where malicious test cases was mistaken as being benign.

False Negative Rate (FNR). denotes the total number of cases where benign test cases is mistaken as malicious.

$$RecallRate = \frac{TNR}{TNR + FPR}, \tag{7.1}$$

$$PrecisionRate = \frac{TPR}{TPR + FPR}, \tag{7.2}$$

$$Accuracy = \frac{TPR + TNR}{TPR + TNR + FPR + FNR}, \tag{7.3}$$

$$F1 = \frac{2 * RecallRate * PrecisionRate}{RecallRate + PrecisionRate}. \tag{7.4}$$

7.10.2 DoS Detection Performance (RQ1)

As part of *RQ1*, we evaluate LGuard's performance in detecting LightingStrike-style DoS attacks. In all 20 test cases, LGuard detected active DoS attacks with 100% precision and accuracy. As DoS is a high-throughput attack over an E-IoT communication bus, the attacks were easily identified and reported as active DoS attacks.

7.10.3 Impersonation Detection Performance (RQ2)

To answer *RQ2*, we determine if LGuard can effectively detect physical tampering of E-IoT Cresnet keypads, thus impersonation attacks. Traffic analysis of Cresnet packets performed by LGuard to identify tampering of E-IoT devices yielded no false positives and LGuard correctly identified keypad removal with 100% precision and accuracy. As keypads are repeatedly queried by the E-IoT controller at about 500ms intervals, tampering detection by LGuard occurred as soon as the keypad

Table 7.4: LGuard performance evaluation on replay attacks.

Legend : BEN = Benign, MAL = Malicious (Replay Attack)

Class	TPR	TNR	FPR	FNR	ACC	PREC	REC	F1
BEN	0.98	1.0	0.00	0.02	0.99	1.0	0.98	0.99
MAL	1.0	0.98	0.02	0.00	0.99	0.98	1.0	0.99

was removed from the E-IoT communication bus. As such, with LGuard, an administrator or technician may quickly receive alerts of tampering or faulty devices almost immediately and react accordingly.

7.10.4 Replay Detection Performance (RQ3)

To answer *RQ3*, we evaluated the performance of LGuard against LightningStrike replay attacks. The results are outlined in Table 7.4. As noted, LGuard performance showed an overall accuracy and precision of 99% in identifying replay attacks over the Cresnet communication. Further, while two false positives were observed in two benign cases (CCTV capture in bright-light), there were no false negatives for any of the malicious cases. In one of these false positive cases, pose vertices were not calculated properly by the Google MediaPipe library. In the other false positive case, vertices were misplaced in the video capture and touch detection could not be processed accurately. Other notable cases such as, one benign (CCTV capture in bright-light) and malicious (CCTV capture in low-light) test case, we found that the MediaPipe classifier misclassified some frames of the recording and placed all the vertices incorrectly in the person's head due to their proximity to the camera. These vertices were then corrected in later frames and yielded to some false positive interface interactions during the mislabeled frames. However, these mis-classified frames did not affect the overall detection by LGuard.

7.10.5 Traffic Obfuscation Performance (RQ4)

To address *RQ4*, we found that manual analysis of Cresnet traffic became more difficult for an attacker when redundant traffic is inserted to the bus via LGuard as traffic obfuscation. First, as four non-existing devices were mimicked by LGuard, the attacker's probability in identifying the real devices was reduced by a factor of two. Further, as random activity is generated, it becomes harder for an external attacker to determine which traffic originates from real user activity and which active traffic was generated by LGuard. To demonstrate the effect of traffic obfuscation, we considered to evaluate the success ratio of a LightningStrike eavesdropper in identifying real E-IoT activity by listening the bus traffic with and without traffic obfuscation. Without the traffic obfuscation, we found that an attacker can identify activities in the E-IoT communication bus (e.g., button presses) with a 100% success ratio since all the activity traffic on the bus is real. However, with traffic obfuscation, we found that the same activities could only be identified with 19% success ratio on average by the attacker. As a result, traffic obfuscation makes the identification of real E-IoT activity harder for the attacker, hence mitigates the effect of LightningStrike eavesdropping attacks. We would like to note that traffic obfuscation applied by LGuard does not cause any issues with the E-IoT system as the E-IoT controller ignores such messages sent by non-existing devices.

7.10.6 Detection Time and Overhead

Our evaluations show that the detection time is dependent on each attack. For DoS attacks, as the rate calculated as incoming data is received, DoS attacks were detected in under 1000ms. Impersonation attacks that tamper with the communication bus and keypads were also detected within 500ms, as the polling messages observed from the E-IoT controller to continuously query the removed interfaces are transmitted within 500ms. Finally, replay attacks were detected within 10 seconds, which is the length of each CCTV video capture. This duration can be further reduced by using better processing hardware, multi-threading, and shorter video captures.

While LGuard operates as a standalone system without modification of the E-IoT deployment, we measured overhead on the host machine (16 GB RAM and I7-700 3.6 GHz Processor). We measured idle (average) monitoring usage of LGuard at a 15% CPU usage and 113 MB RAM peak usage. While processing replay attacks and video recordings, we measured a peak of 25% CPU and 300 MB RAM usage of LGuard.

7.11 Benefits and Discussion

LGuard is designed as a defense solution for E-IoT LightingStrike-style attacks. In this section, we highlight LGuard's benefits and further discuss their usage implications.

Independent Framework. LGuard and all the associated components function as an independent defense system to the E-IoT deployment. Thus, in the case of failure of any LGuard component, LGuard can be adjusted or repaired without any effect on the E-IoT system. Further, in any case that the E-IoT system is damaged or disabled (e.g., due to DoS), LGuard will continue to function, log, and alert the user on any attempted attacks during downtime.

Black-box integration. LGuard addresses some of the biggest limitations of securing E-IoT systems. Namely, no information on the protocols is available to any outside third parties. With LGuard, we achieve a high level of accuracy and precision, while leaving the E-IoT system intact of modifications. Further, LGuard does not provide any overhead to E-IoT operation, and allows for a reliable operation of older systems.

Backwards Compatibility. Legacy and pre-wired systems are an issue for many smart systems, including E-IoT. In the case of E-IoT, replacing older systems for newer versions may be extremely costly (e.g., physical rewiring, hardware, software, labor). As such, LGuard can be configured to support the E-IoT system and older protocols that are common but no longer updated.

CCTV Lighting Conditions. CCTV lighting can vary from deployment to deployment and during the time of the day. During nighttime and in dark locations, CCTV cameras will often have infrared (IR) illuminators, specially for professional security systems. IR illuminators activate on dark conditions for better visibility during CCTV recordings. It is logical that any location where security is a concern, will use cameras with illuminators in the dark. Further, standalone CCTV illuminators can be installed for additional lighting. For these reasons, it is reasonable to assume that LGuard will always have adequate lighting in secure locations as the CCTV systems also depend on this lighting for proper functionality.

Cost and Expandability. The cost of LGuard is minimal as it uses existing smart system components such as CCTV feed and it requires an external computer and low-cost adapters. In contrast to replacing devices, hiring external programmers, purchasing a new system, or re-wiring, LGuard provides an affordable solution for any administrator that wants to secure an E-IoT system. Further, LGuard is flexible and can operate with large and small systems once configured. An administrator may even select which interfaces need to be protected (e.g., access control keypads) and which do not need LGuard's protection. Additionally, if CCTV cameras are not available, wireless cameras may be added for LGuard only as needed at a reasonable price (e.g., IP cameras available under 30 dollars at the time of this writing). Similarly, illuminators are affordable, and can be installed to improve lighting conditions if needed.

7.12 Conclusion

The widespread adoption of smart systems has changed the lives of millions of users worldwide. In these smart ecosystems, E-IoT allows users to control lighting fixtures, relays, shades, door access, and scheduled events. E-IoT systems from various vendors in huge quantities can be found in smart buildings, conference rooms, government or smart private offices, hotels, and similar professional settings. One of the core E-IoT components are proprietary communication protocols that are used for the communication between E-IoT devices. In contrast to well-known communication protocols, very little research exists that investigates the security of these communication protocols. For this reason, users wrongly assume that E-IoT systems and their proprietary components are secure. To investigate the security of E-IoT, we proposed LightingStrike, a series of attacks that leverage insecure E-IoT communication practices and vulnerabilities to an attacker's advantage. Specifically, with LightingStrike we showed that it would be very easy for an attacker with a low level of effort and knowledge to compromise an E-IoT system through communication buses. Specifically, we demonstrated that E-IoT is susceptible to DoS, eavesdropping, impersonation, and replay attacks due insecure communication practices. As traditional defense mechanism cannot mitigate LightingStrike threats due to the distinct characteristics of E-IoT systems, we introduced LGuard as a novel defense system against LightingStrike-style threats. LGuard uses CCTV footage and computer vision techniques to detect replay attacks. LGuard identifies impersonation and DoS attacks by detecting E-IoT tampering and excessive bus traffic. LGuard also obfuscates the E-IoT traffic via adding redundant traffic to the bus in order to mitigate eavesdropping attacks. Finally, we evaluated the performance of LGuard on a realistic E-IoT system. Our analysis show that LGuard achieves an overall accuracy of 99% in detecting DoS, impersonation, and replay attacks and effectively increases the difficulty of extracting useful information through eavesdropping attacks.

CHAPTER 8

CONCLUDING REMARKS AND FUTURE WORK

In this dissertation, we approached the under-researched field of E-IoT system security. We introduced E-IoT as a unique field of research and investigated three novel threat vectors heavily linked to E-IoT. First, we presented a detailed study of E-IoT threat landscape and current defenses. We then introduced three unique threat vectors, proving that E-IoT Drivers, CEC, and E-IoT communication buses can be used to compromise E-IoT systems and their components. Further we proposed a defense mechanism to defend each novel vector against unauthorized and malicious attacks, accounting for the unique design of E-IoT systems.

To first evaluate the current state of E-IoT, we investigated known threats in current E-IoT ecosystems. As such, we researched how attackers with different capabilities may compromise E-IoT. To do so, we divided E-IoT into four layers and for each layer we proposed a novel threat model and highlighted the threats, attacks, and defenses at each E-IoT layer. Finally we detailed how the design and proprietary technologies of E-IoT presents challenges to researchers and current defense strategies.

We explored how E-IoT system drivers can be used as an attack mechanism against E-IoT systems. To do so, we introduced PoisonIvy, a series of novel attacks that leverage malicious drivers against E-IoT system vulnerabilities. With PoisonIvy we demonstrated that an attacker can easily assume control of E-IoT system functions using malicious drivers. Specifically, we show that an attacker can perform DoS attacks, gain remote control, and maliciously abuse system resources through malicious E-IoT drivers. With PoisonIvy, we raise awareness in vulnerable system components that can largely impact the security, privacy, reliability and performance of E-IoT systems worldwide. To defend E-IoT against these threats, we introduced Ivycide, an IDS designed to detect unexpected network traffic from malicious E-IoT drivers. We implemented PoisonIvy and Ivycide in a realistic E-IoT testbed, with several real E-IoT devices. Our detailed evaluation showed that Ivycide achieved an overall accuracy of 97% and precision of 94% without any form of operational overhead or modification to the existing E-IoT system.

To demonstrate how devices can be compromised through HDMI connections, we introduced HDMI-Walk, a novel attack vector, that can be used by an attacker to gain arbitrary control of HDMI devices and perform malicious analysis of devices, eavesdropping, DoS attacks, targeted device attacks, and even facilitate other well-known existing attacks through HDMI. To defend against these newly discovered threats on smart systems, we further proposed HDMI-Watch, a novel intrusion detection system used to detect malicious CEC-based activity within HDMI distributions. HDMI-Watch operates as a standalone IDS within an HDMI distribution, passively monitoring and thus imposing no additional overhead to CEC communication. To test HDMI-Watch's performance, we performed our evaluations in a realistic HDMI testbed with a variety of consumer HDMI devices. Our extensive evaluation results showed that the proposed defense system achieved an average of 98% in accuracy without any modifications required to connected devices.

Finally, we investigated E-IoT vulnerabilities by targeting core E-IoT components. Namely, we investigated vulnerabilities of E-IoT proprietary protocols used in E-IoT communication buses. To do so, we introduced LightningStrike, a series of proof-of-concept attacks that demonstrate several weaknesses in proprietary communication used by E-IoT systems. Specifically, with LightningStrike we demonstrated that popular E-IoT proprietary communication protocols are susceptible to DoS, eavesdropping, impersonation, and replay attacks. As these threats cannot be mitigated by traditional defense mechanisms due to limitations posed by

E-IoT, we introduced LGuard, a novel defense system designed to defend E-IoT systems against LightningStrike attacks. Namely, LGuard uses video captures and network traffic monitoring to detect replay attacks. To detect impersonation and DoS attacks, LGuard detects device tampering and excessive network traffic. Finally, LGuard addresses eavesdropping by obfuscating E-IoT communication bus traffic. Our detailed evaluations showed that LGuard did not incur any operational overhead and achieved an overall accuracy and precision of 99%.

We present several key directions for future research.

- In this dissertation, we presented several viable HDMI-based attacks that can easily compromise integrated devices. However, all of the attacks were designed to use the CEC protocol without any fuzzing or more advanced attack techniques. While the discovered impacts are impactful, we believe that further research is needed into CEC vulnerabilities and other protocols embedded into multimedia devices.

- Drivers as a threat vector are a new discovery and some threats were dependent on the software implementation of vendor drivers. We believe that there is room for further research in this topic, ideally with collaboration from E-IoT vendors. One future research direction could tackle the problem of standardization of E-IoT drivers and offer a solution of a single, secure, driver model that multiple E-IoT vendors can use to expand the functionality of their own systems.

- Throughout this dissertation, we noted the challenges of working and investigating proprietary protocols. However, while this presents a challenge, security-through-obscurity is insufficient. A new research direction needs to be taken in collaboration with E-IoT vendors to secure proprietary E-IoT protocols before they are deployed in countless systems worldwide.

BIBLIOGRAPHY

[AAB$^+$17] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th {USENIX} Security Symposium*, pages 1093–1110, 2017.

[AADB17] M. A. N. Abrishamchi, A. H. Abdullah, A. David Cheok, and K. S. Bielawski. Side channel attacks on smart home systems: A short overview. In *IECON 2017 = 43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 8144–8149, Oct 2017.

[AADV15] Claudio A Ardagna, Rasool Asal, Ernesto Damiani, and Quang Hieu Vu. From security to assurance in the cloud: A survey. *ACM Computing Surveys (CSUR)*, 48(1):1–50, 2015.

[AAUA18] A. Acar, H. Aksu, A. S. Uluagac, and K. Akkaya. Waca: Wearable-assisted continuous authentication. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 264–269, May 2018.

[AAUC18] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):1–35, 2018.

[AB09] Miron Abramovici and Paul Bradley. Integrated circuit security: new threats and solutions. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, pages 1–3, 2009.

[ABB$^+$17] Manos Antonakakis Tim April, Michael Bailey, Matthew Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, et al. Understanding the mirai botnet. In *26th USENIX security symposium*, pages 1093–1110, 2017.

[ABC$^+$18] Hidayet Aksu, Leonardo Babun, Mauro Conti, Gabriele Tolomei, and A Selcuk Uluagac. Advertising in the iot era: Vision and challenges. *IEEE Communications Magazine*, 56(11):138–144, 2018.

[ABK$^+$07] Dakshi Agrawal, Selcuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, and Berk Sunar. Trojan detection using ic fingerprinting. In *2007*

IEEE Symposium on Security and Privacy (SP'07), pages 296–310. IEEE, 2007.

[ADI20] ADI. HoneyWell Serial Interface Module. https://www.adiglobaldistribution.us/Product/4100SM, 2020. Online: Accessed 25-September-2020.

[Adm11] Admin. How to hack into the TV remote control and understand the IR code. https://www.electrodragon.com/how-to-hack-into-the-tv-remote-control-and-understand-the-ir-code/, 2011. Online: Accessed 25-September-2020.

[AFA$^+$20] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '20, page 207–218. ACM, 2020.

[AHIN13] Ahmed Al-Haiqi, Mahamod Ismail, and Rosdiadee Nordin. Keystrokes inference attack on android: A comparative evaluation of sensors and their fusion. *Journal of ICT Research and Applications*, 7(2):117–136, 2013.

[AK18] Asma Alsaidi and Firdous Kausar. Security attacks and countermeasures on cloud assisted iot applications. In *2018 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 213–217. IEEE, 2018.

[ALAM19] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. Sok: Security evaluation of home-based iot deployments. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1362–1380, 2019.

[AMA20a] Intisar Shadeed Al-Mejibli and Nawaf Rasheed Alharbe. Analyzing and evaluating the security standards in wireless network: A review study. *Iraqi Journal for Computers and Informatics*, 46(1):32–39, 2020.

[Ama20b] Amazon. Lindy hdmi cec-less adapter. https://www.amazon.com/Lindy-HDMI-Adapter-Female-41232/dp/B00DL48KVI, 2020. Online: Accessed 18-July-2020.

[Ana20] Oxford Analytica. Fallout of solarwinds hack could last for years. *Emerald Expert Briefings*, (oxan-es), 2020.

[And15] Andrew Tierney. CSL Dualcom CS2300-R signalling unit vulnerabilities. https://cybergibbons.com/security-2/csl-dualcom-cs2300-signalling-unit-vulnerabilities/, 2015. Online: Accessed 10-November-2020.

[And20a] Android Developers. Sign your app. https://developer.android.com/studio/publish/app-signing, 2020.

[And20b] Andy Greenberg. A Tesla Employee Thwarted an Alleged Ransomware Plot. https://www.wired.com/story/tesla-ransomware-insider-hack-attempt/, 2020. Online: Accessed 10-November-2020.

[AOHA17] Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim Abaker Targio Hashem, and Faiz Alotaibi. Internet of things security: A survey. *Journal of Network and Computer Applications*, 88:10–28, 2017.

[AOV18] Ahmet Arıs, Sema F Oktug, and Thiemo Voigt. Security of internet of things for a reliable internet of services. 2018.

[AP13] A. Arabo and B. Pranggono. Mobile malware and smart device security: Trends, challenges and solutions. In *2013 19th International Conference on Control Systems and Computer Science*, May 2013.

[App20] Apple. Signing your apps for gatekeeper. https://developer.apple.com/developer-id/, 2020. Online: Accessed 20-May-2020.

[ARC18] Mahmoud Ammar, Giovanni Russello, and Bruno Crispo. Internet of things: A survey on the security of iot frameworks. *Journal of Information Security and Applications*, 38:8–27, 2018.

[ASH16] ASHRAE. BACnet®, the ASHRAE building automation and control networking protocol. http://www.bacnet.org/, 2016.

[ASH17] ASHRAE. BACnet Network Security Architecture. http://www.bacnet.org/Addenda/Add-2004-135g-PR1.pdf, 2017.

[Aud] AudioAdvice. Which smart home system is best? control4 vs. crestron vs. savant. https://www.audioadvice.com/videos-reviews/control4-vs-crestron-vs-savant/. Online: Accessed 10-December-2019.

[Aud18] Audrey Noble. A look inside the amazing smart-home systems that rich people use. https://www.businessinsider.com/smart-home-tech-that-rich-people-use-2018-7, 2018. Online: Accessed 20-June-2020.

[Bab19] Babun, Leonardo, Aksu, Hidayet, Uluagac, Selcuk A. Method of Resource-limited Device and Device Class Identification using System and Function Call Tracing Techniques, Performance, and Statistical Analysis. (10242193), March 2019.

[BADA15] N. Baharudin, F. H. M. Ali, M. Y. Darus, and N. Awang. Wireless intruder detection system (wids) in detecting de-authentication and disassociation attacks in ieee 802.11. In *2015 5th International Conference on IT Convergence and Security (ICITCS)*, pages 1–5, Aug 2015.

[BAR$^+$20] L. Babun, H. Aksu, L. Ryan, K. Akkaya, E. S. Bentley, and A. S. Uluagac. Z-iot: Passive device-class fingerprinting of zigbee and z-wave iot devices. In *2020 IEEE International Conference on Communications (ICC)*, pages 1–7, 2020.

[Bas19] Bastian Bloessl. Low-cost zigbee selective jamming. https://www.bastibl.net/reactive-zigbee-jamming/, 2019. Online: Accessed 18-July-2020.

[BAU18] Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. Detection of counterfeit and compromised devices using system and function call tracing techniques. (10027697), 7 2018.

[BAU19] Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. A system-level behavioral detection framework for compromised cps devices: Smart-grid case. *ACM Trans. Cyber-Phys. Syst.*, 4(2), nov 2019.

[BBP10a] T. Bonaci, L. Bushnell, and R. Poovendran. Node capture attacks in wireless sensor networks: A system theoretic approach. In *49th IEEE Conference on Decision and Control (CDC)*, pages 6765–6772, 2010.

[BBP10b] Tamara Bonaci, Linda Bushnell, and Radha Poovendran. Probabilistic analysis of covering and compromise in a node capture attack. *Network Security Lab (NSL), Seattle, WA, Techical Report*, 1, 2010.

[BCM15] Michael Bauer, Mark Coatsworth, and Justin Moeller. Nansa: A no-attribution nosleep battery exhaustion attack for portable computing devices. http://pages.cs.wisc.edu/ bauer/CS736Final.pdf, 2015. Online: Accessed 11-Feb-2021.

[BCMU19] Leonardo Babun, Z. Berkay Celik, Patrick McDaniel, and A. Selcuk Uluagac. Real-time analysis of privacy-(un)aware iot applications, 2019.

[BFM17] Robert Buttigieg, Mario Farrugia, and Clyde Meli. Security issues in controller area networks in automobiles. In *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 93–98, 2017.

[BGMT07] Timothy K Buennemeyer, Michael Gora, Randy C Marchany, and Joseph G Tront. Battery exhaustion attack detection with small handheld mobile computers. In *2007 IEEE International Conference on Portable Information Devices*, pages 1–5. IEEE, 2007.

[BGW01] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 180–189, 2001.

[BKH$^+$17] M. Babiker, O. O. Khalifa, K. K. Htike, A. Hassan, and M. Zaharadeen. Automated daily human activity recognition for video surveillance using neural network. In *2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, pages 1–5, 2017.

[BKP15] Ashvini Balte, Asmita Kashid, and Balaji Patil. Security issues in internet of things (iot): A survey. *International Journal of Advanced Research in Computer Science and Software Engineering*, 5(4), 2015.

[Bla08] Blackwire Designs. Control4 automation apps and drivers. https://www.blackwiredesigns.com/cat/automation-apps-and-drivers/control4_drivers/, 2008. Online: Accessed 18-May-2020.

[Bla20] Rich Black. Clear connect technology whitepaper. http://www.lutron.com/TechnicalDocumentLibrary/Clear_Connect_Technology_whitepaper.pdf, 2020. Online: Accessed 10-January-2020.

[BLAN$^+$16] Chafika Benzaid, Karim Lounis, Ameer Al-Nemrat, Nadjib Badache, and Mamoun Alazab. Fast authentication in wireless sensor networks. *Future Generation Computer Systems*, 55:362–375, 2016.

[Blu20a] Bluetooth. Bluetooth Core Specifications. https://www.bluetooth.com/specifications/bluetooth-core-specification/, 2020. Online: Accessed 1-March-2020.

[Blu20b] Bluetooth. The Global Standard For Connection. https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/, 2020. Online: Accessed 1-March-2020.

[BMV17] Samaresh Bera, Sudip Misra, and Athanasios V Vasilakos. Software-defined networking for internet of things: A survey. *IEEE Internet of Things Journal*, 4(6):1994–2008, 2017.

[BÖS19] Ismail Butun, Patrik Österberg, and Houbing Song. Security of the internet of things: Vulnerabilities, attacks, and countermeasures. *IEEE Communications Surveys & Tutorials*, 22(1):616–644, 2019.

[BPP13a] S. D. Babar, N. R. Prasad, and R. Prasad. Jamming attack: Behavioral modelling and analysis. In *Wireless VITAE 2013*, pages 1–5, 2013.

[BPP13b] S. D. Babar, N. R. Prasad, and R. Prasad. Jamming attack: Behavioral modelling and analysis. In *Wireless VITAE 2013*, pages 1–5, 2013.

[BR15] Shivam Bhasin and Francesco Regazzoni. A survey on hardware trojan detection techniques. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2021–2024. IEEE, 2015.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[BSAU18] Leonardo Babun, Amit Kumar Sikder, Abbas Acar, and A. Selcuk Uluagac. Iotdots: A digital forensics framework for smart environments. *CoRR*, 2018.

[BSDV21] Basudeb Bera, Sourav Saha, Ashok Kumar Das, and Athanasios V Vasilakos. Designing blockchain-based access control protocol in iot-enabled smart-grid system. *IEEE Internet of Things Journal*, 8(7):5744–5761, 2021.

[BSK04] Kwang-Hyun Baek, Sean W Smith, and David Kotz. A survey of wpa and 802.11 i rsn authentication protocols. 2004.

[BT11] Leela Krishna Bysani and Ashok Kumar Turuk. A survey on selective forwarding attack in wireless sensor networks. In *2011 International Conference on Devices and Communications (ICDeCom)*, pages 1–5. IEEE, 2011.

[BTJS12] M. V. Bharathi, R. C. Tanguturi, C. Jayakumar, and K. Selvamani. Node capture attack in wireless sensor network: A survey. In *2012 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–3, 2012.

[BWM12] Johannes Barnickel, Jian Wang, and Ulrike Meyer. Implementing an attack on bluetooth 2.1+ secure simple pairing in passkey entry mode. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 17–24. IEEE, 2012.

[C4D14] C4Drivers. Control4 driver programming, Oct, 2014. Online: Accessed 10-December-2019.

[C4F] C4Forums. Control4 forums files download. Online: Accessed 23-January-2020.

[Cam20] Camio. Ai for cost-effective remote video monitoring. https://camio.com/, 2020. Online: Accessed 18-July-2020.

[Car13] Carrier Enterprise. Carrier SAM Module. https://www.carrierenterprise.com/carrier-infinity-series-ethernet-cat-5-wired-broadband-remote-access-module-systxccrct01#tab-info, 2013.

[Car20] Carolina Staff. Make the Invisible Visible . https://www.carolina.com/knowledge/2020/02/20/make-the-invisible-visible, 2020. Online: Accessed 25-September-2020.

[CB08] Rajat Subhra Chakraborty and Swarup Bhunia. Hardware protection and authentication through netlist level obfuscation. In *2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 674–677. IEEE, 2008.

[CB09a] Rajat Subhra Chakraborty and Swarup Bhunia. Harpoon: an obfuscation-based soc design methodology for hardware protection.

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 28(10):1493–1502, 2009.

[CB09b] Rajat Subhra Chakraborty and Swarup Bhunia. Security through obscurity: An approach for protecting register transfer level hardware ip. In *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 96–99. IEEE, 2009.

[CBS$^+$18a] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. Sensitive information tracking in commodity iot. In *27th USENIX Security Symposium*, pages 1687–1704, 2018.

[CBS$^+$18b] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. Sensitive information tracking in commodity iot. In *27th Security Symposium (USENIX Security 18)*, Baltimore, MD, 2018.

[CC12] Liang Cai and Hao Chen. On the practicality of motion based keystroke inference attack. In *International Conference on Trust and Trustworthy Computing*, pages 273–290. Springer, 2012.

[CDG$^+$20] Sravani Challa, Ashok Kumar Das, Prosanta Gope, Neeraj Kumar, Fan Wu, and Athanasios V Vasilakos. Design and analysis of authenticated key agreement scheme in cloud-assisted cyber–physical systems. *Future Generation Computer Systems*, 108:1267–1286, 2020.

[CDL16] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. A survey of man in the middle attacks. *IEEE Communications Surveys Tutorials*, 18(3):2027–2051, 2016.

[CHEK18] M. Chamekh, M. Hamdi, S. El Asmi, and T. Kim. Secured distributed iot based supply chain architecture. In *2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 199–202, 2018.

[Chr17] Chris Brook. Dahua Patching Backdoor in DVRs, IP Cameras. https://threatpost.com/dahua-patching-backdoor-in-dvrs-ip-cameras/124119/, 2017.

[CMT$^+$] Z. B. Celik, P. McDaniel, G. Tan, L. Babun, and A. S. Uluagac. Verifying Internet of Things Safety and Security in Physical Spaces. *IEEE Security Privacy.*

[CMZ$^+$19] Nadia Chaabouni, Mohamed Mosbah, Akka Zemmari, Cyrille Sauvignac, and Parvez Faruki. Network intrusion detection for iot security based on learning techniques. *IEEE Communications Surveys & Tutorials*, 21(3):2671–2701, 2019.

[Con] Control4. Control4 driver search. https://drivers.control4.com/solr/drivers/browse. Online: Accessed 20-June-2019.

[Con10a] Control4. Control4 operating system os release notes. https://www.control4.com/files/dealers/TechDoc00046-ComposerProSoftware-Release-2.0.6-ReleaseNotes.pdf, 2010. Online: Accessed 20-June-2020.

[Con10b] Control4. Getting started with composer pro. https://www.control4.com/files/dealers/200-00168-ComposerProGettingStarted.pdf, Jun, 2010. Online: Accessed 10-December-2019.

[Con10c] Control4. Composer pro software release update instructions. https://www.control4.com/files/dealers/TechDoc00005_ComposerUpdate Instructions_1 8 2.pdf, Mar, 2010. Online: Accessed 10-April-2020.

[Con12] Control4. Exterior door station: Security best practices. https://www.control4.com/docs/product/security/best-practices/english/latest/security-best-practices-rev-a.pdf, 2012. Online: Accessed 22-September-2020.

[Con13a] Control4. Configurable Decora Wired Keypad Installation Guide. https://www.control4.com/docs/product/wired-keypad/installation-guide/english/revision/B/wired-keypad-installation-guide-rev-b.pdf, 2013. Online: Accessed 20-March-2020.

[Con13b] Control4. Control4 zigbee: The definitive magic button press guide. https://technet.genesis-technologies.ch/control4-zigbee-the-definitive-guide/, 2013. Online: Accessed 20-June-2019.

[Con14] Control4. Control4 panelized lighting: Reference guide for electricians. https://www.control4.com/docs/product/panelized-lighting/professional-reference-guide/latest/panelized-lighting-professional-reference-guide-rev-b.pdf, Feb, 2014. Online: Accessed 10-December-2019.

[Con15a] Control4. 7in and 10in t3 series in-wall touch screen installation guide. https://www.manualslib.com/manual/1436607/Control-4-C4-Wall7-Wh.htmlf, 2015. Online: Accessed 22-September-2020.

[Con15b] Control4. Keep Your House Dry, DAM-it! https://www.control4.com/blog/363/keep-your-house-dry-damit/, 2015. Online: Accessed 20-June-2020.

[Con18] Control4. Press release: Four years in a row, control4 named leading whole-house automation brand in cepro brand analysis, 2018. Online: Accessed 20-June-2020.

[Con19] Control4. Control4 driver search, Jan, 2019. Online: Accessed 10-December-2019.

[Con20a] Control4. Control4 4sight Services. https://www.control4.com/o/4sight-services, 2020. Online: Accessed 25-September-2020.

[Con20b] Control4. Control4: About our company. https://www.control4.com/company/, 2020. Online: Accessed 18-May-2020.

[Con20c] Control4. Control4 Sensors. https://www.control4.com/solutions/products/sensors/, 2020. Online: Accessed 20-June-2020.

[Con20d] Control4. Control4 Solution Multi-Room Audio. https://www.control4. com/solutions/multi-room-audio/, 2020. Online: Accessed 25-September-2020.

[Con21a] Control4. Control4 Camera Solutions. https://www.control4.com/solutions/products/cameras/, 2021. Online: Accessed 28-October-2021.

[Con21b] Control4. Control4 Driver Search - Camera. https://drivers.control4.com/solr/drivers/browse?&fq=primaryProxy%3A%22camera%22, 2021. Online: Accessed 28-October-2021.

[Crea] Crestron. Crestron technical institute. https://www.crestron.com/training. Online: Accessed 10-December-2019.

[Creb] Crestron. infinet ex® network and er wireless gateway. https://www.crestron.com/Products/Control-Hardware-Software/Wireless-Communications/Wireless-Gateways/CEN-GWEXER. Online: Accessed 10-January-2020.

[Cre06] Crestron. Crestron isys touchpanel operation guide, 2006. Online: Accessed 10-April-2020.

[Cre17a] Cresnet. Crestron Cresnet Monitor - Cresnet protocol analysis. https://archive.codeplex.com/?p=cresnet, 2017.

[Cre17b] Crestron. Crestron Automation Products. https://www.crestron.com/, 2017.

[Cre20a] Crestron. Cresnet wiring - cable types & lengths for connecting devices. https://support.crestron.com/app/answers/detail/a_id/1629/~/cresnet-wiring—cable-types, 2020. Online: Accessed 10-April-2020.

[Cre20b] Crestron. Crestron Database Release Notes. http://www.crestron.com/release_notes/crestron_database_31_05.html, 2020. Online: Accessed 25-September-2020.

[Cre20c] Crestron. Crestron Lightning and Environment Sensors. https://www.crestron.com/Products/Lighting-Environment/Sensors, 2020. Online: Accessed 20-June-2020.

[Cre20d] Crestron. Crestron Multiroom Audio. https://www.crestron.com/products/audio/multiroom-audio, 2020. Online: Accessed 25-September-2020.

[Cre20e] Crestron. Crestron software - downloading latest versions. https:// support.crestron.com/app/answers/detail/a_id/32/crestron-software—downloading-latest-versions, 2020. Online: Accessed 22-September-2020.

[Cre20f] Crestron. Crestron's Commitment to Security. https://www.crestron.com/About/commitment-to-security, 2020. Online: Accessed 20-October-2020.

[Cre20g] Crestron. Security at crestron. https://www.crestron.com/Security/Security-at-Crestron, 2020. Online: Accessed 18-May-2020.

[Cre20h] Crestron. Tsw model touchscreen manual. https://bit.ly/3mcRNrI, 2020. Online: Accessed 20-June-2020.

[Cre21a] Crestron. Crestron Application Market - Integrated Pan/Tilt. https://applicationmarket.crestron.com/integrated-pan-tilt-1/, 2021. Online: Accessed 28-October-2021.

[Cre21b] Crestron. Works with Crestron - Cameras. https://docs.crestron.com /en-us/8525/Content/CP4R/Appendix/Works-With-Crestron-Home/Cameras.htm, 2021. Online: Accessed 28-October-2021.

[CSC$^+$16] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen. Ghost-in-zigbee: Energy depletion attack on zigbee-based wireless networks. *IEEE Internet of Things Journal*, 3(5):816–829, Oct 2016.

[CVE20a] CVE Details. CVE Details Security Vulnerabilities Search: Crestron. https://www.cvedetails.com/vulnerability-list/vendor_id-15891/Crestron.html, 2020. Online: Accessed 25-September-2020.

[CVE20b] CVE Details. CVE Details Security Vulnerabilities Search: Savant. https://www.cvedetails.com/vulnerability-list/vendor_id-1231/Savant.html, 2020. Online: Accessed 25-September-2020.

[Cyb20] Cybersecurity & Infrastructure Security Agency. Securing Wireless Networks. https://us-cert.cisa.gov/ncas/tips/ST05-003, 2020. Online: Accessed 10-November-2020.

[Dav11] David Kravets. Wi-fi–hacking neighbor from hell sentenced to 18 years. https://www.wired.com/2011/07/hacking-neighbor-from-hell/, 2011. Online: Accessed 15-May-2021.

[Dav12] Andy Davis. HDMI : Hacking Displays Made Interesting, Mar, 2012.

[Dav13] Andy Davis. What the HEC? Security implications of HDMI Ethernet Channel and other related protocols, Aug, 2013.

[Dav20] David Mead. A Comprehensive Guide to Z-Wave. https:// linkdhome.com/articles/What-is-z-wave, 2020. Online: Accessed 25-October-2020.

[DBU20] Kyle Denney, Leonardo Babun, and A Selcuk Uluagac. Usb-watch: a generalized hardware-assisted insider threat detection framework. *Journal of Hardware and Systems Security*, 4(2):136–149, 2020.

[DDGD$^+$19] Michele De Donno, Alberto Giaretta, Nicola Dragoni, Antonio Bucchiarone, and Manuel Mazzara. Cyber-storms come from clouds: Security of cloud computing in the iot era. *Future Internet*, 11(6):127, 2019.

[DEB$^+$19] Kyle Denney, Enes Erdin, Leonardo Babun, Michael Vai, and Selcuk Uluagac. Usb-watch: A dynamic hardware-assisted usb threat detection framework. In *International Conference on Security and Privacy in Communication Systems*, pages 126–146. Springer, 2019.

[DEBU19] Kyle Denney, Enes Erdin, Leonardo Babun, and A. Selcuk Uluagac. Dynamically detecting usb attacks in hardware: Poster. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, page 328–329, 2019.

[Dev13] Jega Anish Dev. Usage of botnets for high speed md5 hash cracking. In *Third International Conference on Innovative Computing Technology (INTECH 2013)*, pages 314–320. IEEE, 2013.

[Dev19] Android Developers. Hdmi-cec control service. https://source.android.com/devices/tv/hdmi-cec, May, 2019.

[DG17] Seyed Mahdi Darroudi and Carles Gomez. Bluetooth low energy mesh networks: A survey. *Sensors*, 17(7):1467, 2017.

[DGS$^+$19] J. Dudak, G. Gaspar, S. Sedivy, P. Fabo, L. Pepucha, and P. Tanuska. Serial communication protocol with enhanced properties–securing communication layer for smart sensors applications. *IEEE Sensors Journal*, 19(1):378–390, 2019.

[DLD09] Pradip De, Yonghe Liu, and Sajal K Das. Deployment-aware modeling of node compromise spread in wireless sensor networks using epidemic theory. *ACM Transactions on Sensor Networks (TOSN)*, 5(3):1–33, 2009.

[Dor17] Brannon Dorsey. Crack WPA/WPA2 Wi-Fi Routers with Aircrack-ng and Hashcat. https://medium.com/@brannondorsey/crack-wpa-

wpa2-wi-fi-routers-with-aircrack-ng-and-hashcat-a5a5d3ffea46, Jul, 2017.

[dri20] drivercentral. Control4 drivers. https://drivercentral.io/platforms/control4-drivers/, 2020. Online: Accessed 20-May-2020.

[DRW$^+$14] Abe Davis, Michael Rubinstein, Neal Wadhwa, Gautham J. Mysore, Frédo Durand, and William T. Freeman. The visual microphone: Passive recovery of sound from video. *ACM Trans. Graph.*, 33(4), July 2014.

[Dun10] John Dunning. Taming the blue beast: A survey of bluetooth based threats. *IEEE Security & Privacy*, 8(2):20–27, 2010.

[DV17] Jyoti Deogirikar and Amarsinh Vidhate. Security attacks in iot: A survey. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 32–37. IEEE, 2017.

[DWH13] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Zmap: Fast internet-wide scanning and its security applications. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 605–620, Berkeley, CA, USA, 2013. USENIX Association.

[Ed 17] Ed Bott. Make your cloud safer: How you can use two-factor authentication to protect cloud services. https://www.zdnet.com/article/make-your-cloud-safer-how-you-can-use-two-factor-authentication-to-protect-cloud-services/, 2017. Online: Accessed 20-November-2020.

[Edu20] Educba. Ftp vs sftp. https://www.educba.com/ftp-vs-sftp/, 2020. Online: Accessed 22-September-2020.

[Edw20] Edward Kovacs. Iot devices at major manufacturers infected with malware via supply chain attack. https://www.securityweek.com/iot-devices-major-manufacturers-infected-malware-supply-chain-attack, 2020. Online: Accessed 25-September-2020.

[EEEE07] Jakob Ehrensvärd, Stina Ehrensvärd, Leif Eriksson, and Fredrik Einberg. Tamper evident packaging, January 30 2007. US Patent 7,170,409.

[EGH$^+$14] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):1–29, 2014.

[EKT16] Oliver Eigner, Philipp Kreimel, and Paul Tavolato. Detection of man-in-the-middle attacks on industrial control networks. In *2016 International Conference on Software Security and Assurance (ICSSA)*, 2016.

[Eru20] Eruc Andersen, Rob Landley, Denys Vlasenko. BusyBox: The Swiss Army Knife of Embedded Linux. https://busybox.net/about.html, 2020. Online: Accessed 10-November-2020.

[Eur20] European Union for Cybersecurity. Guidelines for Securing the Internet of Things. https://www.enisa.europa.eu/publications/guidelines-for-securing-the-internet-of-things, 2020. Online: Accessed 20-November-2020.

[fCo02] fCoder. History RS-232-C - Legacy Serial Connector. https://www.lookrs232.com/rs232/history_rs232.htm, 2002.

[Fed15] Federal Trade Commission. Securing your Wireless Network. https://www.consumer.ftc.gov/articles/0013-securing-your-wireless-network, 2015. Online: Accessed 10-November-2020.

[FG13] Behrang Fouladi and Sahand Ghanoun. Security evaluation of the z-wave wireless protocol. *Black Hat USA*, 24:1–2, 2013.

[FH18] Mayra Rosario Fuentes and Numaan Huq. Securing connected hospitals. https://www.key4biz.it/wp-content/uploads/2018/04/rpt-securing-connected-hospitals.pdf, 2018.

[FKWL13] Ashfaq Hussain Farooqi, Farrukh Aslam Khan, Jin Wang, and Sungyoung Lee. A novel intrusion detection framework for wireless sensor networks. *Personal and ubiquitous computing*, 17(5):907–919, 2013.

[FR15] J. D. Fuller and B. W. Ramsey. Rogue z-wave controllers: A persistent attack channel. In *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*, pages 734–741, Oct 2015.

[FS18] FS. Ipv4 vs ipv6: What's the difference? https://community.fs.com/blog/ipv4-vs-ipv6-whats-the-difference.html, 2018. Online: Accessed 10-January-2020.

[FSG$^+$14] Diogo AB Fernandes, Liliana FB Soares, João V Gomes, Mário M Freire, and Pedro RM Inácio. Security issues in cloud environments: a survey. *International Journal of Information Security*, 13(2):113–170, 2014.

[FZ19] Muhammad Junaid Farooq and Quanyan Zhu. Iot supply chain security: Overview, challenges, and the road ahead. *arXiv preprint arXiv:1908.07828*, 2019.

[GB19] Mordechai Guri and Dima Bykhovsky. air-jumper: Covert air-gap exfiltration/infiltration via security cameras & infrared (ir). *Computers & Security*, 82:15–29, 2019.

[GE18] Mordechai Guri and Yuval Elovici. Bridgeware: The air-gap malware. *Communications of the ACM*, 61(4):74–82, 2018.

[GHKE16] Mordechai Guri, Ofer Hasson, Gabi Kedma, and Yuval Elovici. An optical covert-channel to leak data through an air-gap. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 642–649. IEEE, 2016.

[GLY14] Kanika Grover, Alvin Lim, and Qing Yang. Jamming and anti–jamming techniques in wireless networks: a survey. *International Journal of Ad Hoc and Ubiquitous Computing*, 17(4):197–215, 2014.

[Goo18] Google. What is CEC? https://support.google.com/chromecast/answer/7199917?hl=en, 2018. Online: Accessed 10-January-2020.

[Gor12] Celia Gorman. Counterfeit chips on the rise. *IEEE Spectrum*, 49(6):16–17, 2012.

[GSD$^+$17] O. Gasser, Q. Scheitle, C. Denis, N. Schricker, and G. Carle. Security implications of publicly reachable building automation systems. In *2017 IEEE Security and Privacy Workshops (SPW)*, pages 199–204, May 2017.

[GWS10] Bernd Grobauer, Tobias Walloschek, and Elmar Stocker. Understanding cloud computing vulnerabilities. *IEEE Security & privacy*, 9(2):50–57, 2010.

[H$^+$19] Wan Haslina Hassan et al. Current research on internet of things (iot) security: A survey. *Computer networks*, 148:283–294, 2019.

[HB18] Duncan Hodges and Oliver Buckley. Reconstructing what you said: Text inference using smartphone motion. *IEEE Transactions on Mobile Computing*, 18(4):947–959, 2018.

[HCS$^+$19] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar. A survey on iot security: application areas, security threats, and solution architectures. *IEEE Access*, 7:82721–82743, 2019.

[HCT17] Jun Han, Albert Jin Chung, and Patrick Tague. Pitchln: Eavesdropping via intelligible speech reconstruction using non-acoustic sensor fusion. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '17, page 181–192, New York, NY, USA, 2017. Association for Computing Machinery.

[HDM09] HDMI Licensing LLC. HDMI Specification guide V1.4. http://d1.amobbs.com/bbs_upload782111/files_51ourdev_716302E34B9Q.pdf, Jun, 2009.

[HDM18] HDMI Licensing LLC. Inside an HDMI Cable. https://www.hdmi.org/installers/insidehdmicable.aspx, 2018. Online: Accessed 20-June-2020.

[Hen18] Josh Hendrickson. ZigBee vs. Z-Wave: Choosing Between Two Big Smarthome Standards. https://www.howtogeek.com/394567/zigbee-vs.-z-wave-choosing-between-two-big-smarthome-standards/, 2018. Online: Accessed 20-October-2020.

[Her04] J Hering. Bluetooth cracking gun: Bluesniper. https://www.defcon.org/html/links/dc_press/archives/12/esato_bluetooth cracking.htm, 2004. Online: Accessed 11-Feb-2021.

[Hey13] Cliff Heyne. AV Tip: How to Avoid Blowing Out Your Speakers. https://www.audioholics.com/home-theater-connection/avoid-blowing-speakers, Jan, 2013.

[HGC$^+$19] Yan Huang, Xin Guan, Hongyang Chen, Yi Liang, Shanshan Yuan, and Tomoaki Ohtsuki. Risk assessment of private information inference for motion sensor embedded iot devices. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(3):265–275, 2019.

[HH07] Konstantin Hypponen and Keijo MJ Haataja. “nino” man-in-the-middle attack on bluetooth secure simple pairing. In *2007 3rd IEEE/IFIP International Conference in Central Asia on Internet*, pages 1–5. IEEE, 2007.

[HH08] Keijo MJ Haataja and Konstantin Hypponen. Man-in-the-middle attacks on bluetooth: a comparative analysis, a novel attack, and countermeasures. In *2008 3rd International Symposium on Communications, Control and Signal Processing*, pages 1096–1102. IEEE, 2008.

[HHS19] Stefan Hristozov, Manuel Huber, and Georg Sigl. Protecting restful iot devices from battery exhaustion dos attacks. *arXiv preprint arXiv:1911.08134*, 2019.

[HKMA14] K. K. Htike, O. O. Khalifa, H. A. Mohd Ramli, and M. A. M. Abushariah. Human activity recognition for video surveillance using sequences of postures. In *The Third International Conference on e-Technologies and Networks for Development (ICeND2014)*, pages 79–82, 2014.

[Hol03] David G. Holmberg. Bacnet wide area network security threat assessment. *NIST, Department of Commerce*, July 2003.

[Hol05] Kasey Holman. HDMI Licensing LLC Announces Availability of HDMI 1.2a Specification, Dec, 2005.

[Hon01] Honeywell. Ethernet Interface User Manual. https://www.honeywell process.com/library/support/Public/Documents/51-52-25-96.pdf, 2001. Online: Accessed 25-September-2020.

[hon20] The honeynet project. https://www.honeynet.org, 2020. Online: Accessed 25-September-2020.

[HRFMF13] Keiko Hashizume, David G Rosado, Eduardo Fernández-Medina, and Eduardo B Fernandez. An analysis of security issues for cloud computing. *Journal of internet services and applications*, 4(1):5, 2013.

[HT08] Keijo Haataja and Pekka Toivanen. Practical man-in-the-middle attacks against bluetooth secure simple pairing. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–5. IEEE, 2008.

[HT10] Keijo Haataja and Pekka Toivanen. Two practical man-in-the-middle attacks on bluetooth secure simple pairing and countermeasures. *IEEE Transactions on Wireless Communications*, 9(1):384–392, 2010.

[HW88] Shian-Uei Hwu and Donald R Wilton. Electromagnetic scattering and radiation by arbitrary configurations of conducting bodies and wires. Technical report, San Diego State Univ Foundation CA, 1988.

[IDF07] Krontiris Ioannis, Tassos Dimitriou, and Felix C Freiling. Towards intrusion detection in wireless sensor networks. In *Proc. of the 13th European Wireless Conference*, pages 1–10. Citeseer, 2007.

[IEE20] IEEE. Ieee standard for secure scada communications protocol (sscp). *IEEE Std 1711.2-2019*, pages 1–37, 2020.

[Ind08] Indrek Mandre. Interfacing with a sensor, ir remotes. http://www.mare.ee/indrek/irtroll/, 2008. Online: Accessed 18-May-2020.

[Ing19] Ingram Micro. 4 innovations in theft and loss prevention. https://imaginenext.ingrammicro.com/iot/4-innovations-in-theft-and-loss-prevention, 2019. Online: Accessed 22-September-2020.

[Int17] Intel. Different Wi-Fi Protocols and Data Rates. https://www.intel.com/content/www/us/en/support/articles/000005725/network-and-i-o/wireless-networking.html, 2017.

[IoT18] IoTBusinessNews. The number of smart homes in europe and north america reached 45 million in 2017, Sept, 2018. Online: Accessed 10-December-2019.

[Jac19] Jacob Baines. Eight Devices, One Exploit. https://medium.com/tenable-techblog/eight-devices-one-exploit-f5fc28c70a7c, 2019. Online: Accessed 25-September-2020.

[JDV19] Srinivas Jangirala, Ashok Kumar Das, and Athanasios V Vasilakos. Designing secure lightweight blockchain-enabled rfid-based authentication protocol for supply chains in 5g mobile edge computing environment. *IEEE Transactions on Industrial Informatics*, 16(11):7081–7093, 2019.

[JIS13] MD Joseph I. Sirven. Photosensitivity and Seizures. https://www.epilepsy.com/learn/triggers-seizures/photosensitivity-and-seizures, Nov, 2013.

[Joh14] John Ehringer. Gaining Serial Console Access on the Control4 Mini Touch Screen. http://www.5khz.com/2014/07/22/gaining-serial-console-access-on-the-control4-mini-touch-screen/, 2014. Online: Accessed 10-November-2020.

[Jon20] Jon Martindale. What is ftp? https://www.digitaltrends.com/computing/what-is-ftp-and-how-do-i-use-it/, 2020. Online: Accessed 22-September-2020.

[JSSN18] Kang Eun Jeon, James She, Perm Soonsawad, and Pai Chet Ng. Ble beacons for internet of things applications: Survey, challenges, and opportunities. *IEEE Internet of Things Journal*, 5(2):811–828, 2018.

[Jul15] Julie Jacobson. Savant kills litetouch; lutron to the rescue; debating hardwired lighting control. https://www.cepro.com/news/savant_kills_litetouch_works_with_lutron_on_fix_hardwired_lighting_control/, 2015. Online: Accessed 20-June-2020.

[JVW$^+$14] Qi Jing, Athanasios V Vasilakos, Jiafu Wan, Jingwei Lu, and Dechao Qiu. Security of the internet of things: perspectives and challenges. *Wireless Networks*, 20(8):2481–2501, 2014.

[JZA$^+$20] Yizhen Jia, Fangtian Zhong, Arwa Alrawais, Bei Gong, and Xiuzhen Cheng. Flowguard: An intelligent edge defense mechanism against iot ddos attacks. *IEEE Internet of Things Journal*, 7(10):9552–9562, 2020.

[KBAU18] C. Kaygusuz, L. Babun, H. Aksu, and A. S. Uluagac. Detection of compromised smart grid devices with machine learning and convolution techniques. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2018.

[Ker20] Kernel Development Community. Kernel module signing facility. https://www.kernel.org/doc/html/v4.15/admin-guide/module-signing.html, 2020.

[KF18] Wenyuan Xu Kevin Fu. Risks of trusting the physics of sensors. hhttps://cacm.acm.org/opinion/articles/224627-risks-of-trusting-the-physics-of-sensors/fulltext, 2018. Online: Accessed 20-June-2019.

[KG19] Rakesh Kumar and Rinkaj Goyal. On cloud security requirements, threats, vulnerabilities and countermeasures: A survey. *Computer Science Review*, 33:1–48, 2019.

[KH18] Christopher P Kohlios and Thaier Hayajneh. A comprehensive attack flow model and security analysis for wi-fi and wpa3. *Electronics*, 7(11):284, 2018.

[KHS17] R. Krejci, O. Hujnak, and M. Svepes. Security survey of the iot wireless protocols. In *2017 25th Telecommunication Forum (TELFOR)*, pages 1–4, Nov 2017.

[KKAA14] Mahmoud Khasawneh, Izadeen Kajman, Rashed Alkhudaidy, and Anwar Althubyani. A survey on wi-fi protocols: Wpa and wpa2. In *International Conference on Security in Computer Networks and Distributed Systems*, pages 496–511. Springer, 2014.

[Kni06] M. Knight. Wireless security - how safe is z-wave? *Computing Control Engineering Journal*, 17(6):18–23, Dec 2006.

[Kod18] Kody. Hack WPA & WPA2 Wi-Fi Passwords with a Pixie-Dust Attack Using Airgeddon. https://null-byte.wonderhowto.com/how-to/hack-wpa-wpa2-wi-fi-passwords-with-pixie-dust-attack-using-airgeddon-0183556/, 2018. Online: Accessed 10-November-2020.

[KSP07] D. S. Kim, Y. K. Suh, and J. S. Park. Toward assessing vulnerability and risk of sensor networks under node compromise. In *2007 International Conference on Computational Intelligence and Security (CIS 2007)*, pages 740–744, 2007.

[KT15] Surapon Kraijak and Panwit Tuwanut. A survey on iot architectures, protocols, applications, security, privacy, real-world implementation and future trends. 2015.

[KTC$^+$08] Samuel T King, Joseph Tucek, Anthony Cozzie, Chris Grier, Weihang Jiang, and Yuanyuan Zhou. Designing and implementing malicious hardware. *Leet*, 8:1–8, 2008.

[Kuh04] Markus G Kuhn. Electromagnetic eavesdropping risks of flat-panel displays. In *International Workshop on Privacy Enhancing Technologies*, pages 88–107. Springer, 2004.

[Lan18] Kimberly Lancaster. Control4 delivers high-resolution audio and homeowner personalization enhancements to elevate the smart home experience, Sept 2018. Online: Accessed 10-December-2019.

[LBAU17] Juan Lopez, Leonardo Babun, Hidayet Aksu, and A Selcuk Uluagac. A survey on function and system call hooking approaches. *Journal of Hardware and Systems Security*, 1(2):114–136, 2017. Accessed: 11-17-2018.

[LCY$^+$18] Yi Liang, Zhipeng Cai, Jiguo Yu, Qilong Han, and Yingshu Li. Deep learning based inference of private information using embedded sensors in smart devices. *IEEE Network*, 32(4):8–14, 2018.

[LDS09a] Arash Habibi Lashkari, Mir Mohammad Seyed Danesh, and B. Samadi. A survey on wireless security protocols (wep, wpa and wpa2/802.11i). In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 48–52, Aug 2009.

[LDS09b] Arash Habibi Lashkari, Mir Mohammad Seyed Danesh, and Behrang Samadi. A survey on wireless security protocols (wep, wpa and wpa2/802.11 i). In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 48–52. IEEE, 2009.

[Leg19] Legrand. Legrand® announces ultra-secure wireless lighting controls platform. https://www.legrand.us/aboutus/press-room/news/legrand-announces-wireless-lighting-controls-platform.aspx, 2019. Online: Accessed 10-January-2020.

[Lev] Levitron. Levnet rf™: Self-powered wireless built on reliability. https://www.leviton.com/en/products/brands/levnet-rf. Online: Accessed 10-January-2020.

[LFR$^+$16] Samuel Litchfield, David Formby, Jonathan Rogers, Sakis Meliopoulos, and Raheem Beyah. Rethinking the honeypot for cyber-physical systems. *IEEE Internet Computing*, 20(5):9–17, 2016.

[LHH$^+$18] Chao Lin, Debiao He, Xinyi Huang, Kim-Kwang Raymond Choo, and Athanasios V Vasilakos. Bsein: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. *Journal of Network and Computer Applications*, 116:42–52, 2018.

[Li 18] Li Yinghua. What are the Differences Between Enterprise Wi-Fi and Home Wi-Fi? . https://e.huawei.com/en/eblog/enterprise-networking/wifi6/What-the-difference-between-corporate-Wi-Fi-and-home-Wi-Fi, 2018. Online: Accessed 10-November-2020.

[Lin19] Lindsey O'Donnell. Wireless presentation systems have an array of critical flaws. https://threatpost.com/bugs-wireless-presentation-systems/144318/, 2019. Online: Accessed 25-September-2020.

[Liq21] LiquidSpace. Liquidspace: Rent flexible office space. https://liquid space.com/, 2021. Online: Accessed 15-May-2021.

[Lit06] LiteTouch. LiteTouch Lighting Control Systems Installation and Troubleshooting Manual. http://sav-documentation.s3.amazonaws.com /Internal Documentation/LiteTouch and Savant Lighting/Troubleshooting Manual.pdf, 2006. Online: Accessed 20-March-2020.

[LKP07] Mingyan Li, Iordanis Koutsopoulos, and Radha Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 1307–1315. IEEE, 2007.

[LS19] Jessy Lin and Jason Seibel. Motion-based side-channel attack on mobile keystrokes. http://css.csail.mit.edu/6.858/2019/projects/lnj-jseibel.pdf, 2019.

[LSR$^+$15] Yuhong Liu, Yan Lindsay Sun, Jungwoo Ryoo, Syed Rizvi, and Athanasios V Vasilakos. A survey of security and privacy challenges in cloud computing: solutions and future directions. *Journal of Computing Science and Engineering*, 9(3):119–133, 2015.

[LTN$^+$19] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines, 2019.

[LU02] Joe Loughry and David A Umphress. Information leakage from optical emanations. *ACM Transactions on Information and System Security (TISSEC)*, 5(3):262–289, 2002.

[Luk20] Lukas Rist, Johnny Vesterngaard, Daniel Haslinger, Andrea Pasquale, and John Smith. Conpot ics/scada honeypot. https://www.compot.org, 2020. Online: Accessed 11-September-2020.

[Lut20] Lutron. Lutron Integration Protocol. http://www.lutron.com/TechnicalDocumentLibrary/040249.pdf, 2020. Online: Accessed 10-May-2020.

[LWYY15] Chi Lin, Guowei Wu, Chang Wu Yu, and Lin Yao. Maximizing destructiveness of node capture attack in wireless sensor networks. *The Journal of Supercomputing*, 71(8):3181–3212, 2015.

[Lyn07] Lynn Tan. Protect against Bluetooth threats. https://www.zdnet.com/article/protect-against-bluetooth-threats/, 2007. Online: Accessed 10-November-2020.

[LYZ$^+$17] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017.

[LZ18] Karim Lounis and Mohammad Zulkernine. Connection dumping vulnerability affecting bluetooth availability. In *International Conference on Risks and Security of Internet and Systems*, pages 188–204. Springer, 2018.

[LZ19a] Karim Lounis and Mohammad Zulkernine. Bad-token: denial of service attacks on wpa3. In *Proceedings of the 12th International Conference on Security of Information and Networks*, pages 1–8, 2019.

[LZ19b] Karim Lounis and Mohammad Zulkernine. Wpa3 connection deprivation attacks. In *International Conference on Risks and Security of Internet and Systems*, pages 164–176. Springer, 2019.

[LZ20] Karim Lounis and Mohammad Zulkernine. Attacks and defenses in short-range wireless technologies for iot. *IEEE Access*, 8:88892–88932, 2020.

[LZSL20] Haiyan Lan, Xiaodong Zhu, Jianguo Sun, and Sizhao Li. Traffic data classification to detect man-in-the-middle attacks in industrial control system. In *2019 6th International Conference on Dependable Systems and Their Applications (DSA)*, pages 430–434, 2020.

[Mar14] Marantz. AV Surround Receiver Web Manual. http://manuals.marantz.com/SR7009/EU/EN/HJWMSYmehwmguq.php, 2014. Online: Accessed 25-September-2020.

[Mar18] Mark N. Vena. How crestron paved the way for the smart home, and more. https://www.forbes.com/sites/moorinsights/2018/08/23/how-crestron-paved-the-way-for-the-smart-home-and-more/#397001f141f8, 2018. Online: Accessed 18-May-2020.

[Mar19] Maria Korolov. What is a supply chain attack? why you should be wary of third-party providers. https://www.csoonline.com/article/3191947/what-is-a-supply-chain-attack-why-you-should-be-wary-of-third-party-providers.html, 2019. Online: Accessed 22-September-2020.

[Mar20] Mark M. Types of Remote Access for DVRs. http://polarisusa.com/articles/8/types-of-remote-access-for-dvrs, 2020. Online: Accessed 27-September-2020.

[MBY^{+}19] J. Myers, L. Babun, E. Yao, S. Helble, and P. Allen. Mad-iot: Memory anomaly detection for the internet of things. In *2019 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2019.

[MDMT10] B. R. Moyers, J. P. Dunning, R. C. Marchany, and J. G. Tront. Effects of wi-fi and bluetooth battery exhaustion attacks on mobile devices. In *2010 43rd Hawaii International Conference on System Sciences*, pages 1–9, 2010.

[MG10] David Martins and Hervé Guyennet. Wireless sensor network attacks and security mechanisms: A short survey. In *2010 13th International Conference on Network-Based Information Systems*, pages 313–320. IEEE, 2010.

[MGKP09] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou. A survey on jamming attacks and countermeasures in wsns. *IEEE Communications Surveys Tutorials*, 11(4):42–56, 2009.

[MHDK04] T. Martin, M. Hsiao, Dong Ha, and J. Krishnaswami. Denial-of-service attacks on battery-powered mobile computers. In *Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Proceedings of the*, pages 309–318, 2004.

[Mic20] Microsoft. Driver signing, 2020. Online: Accessed 20-May-2020.

[Mil13] John F Miller. Supply chain attack framework and attack patterns. Technical report, MITRE CORP MCLEAN VA, 2013.

[MJ19] Anindya Maiti and Murtuza Jadliwala. Light ears: Information leakage via smart lights. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(3):1–27, 2019.

[MLM$^+$99] Martin W Murhammer, Kok-Keong Lee, Payam Motallebi, Paolo Borghi, and Karl Wozabal. *IP Network Design Guide.* IBM Corporation, 1999.

[MMA$^+$16] A. Mirian, Z. Ma, D. Adrian, M. Tischer, T. Chuenchujit, T. Yardley, R. Berthier, J. Mason, Z. Durumeric, J. A. Halderman, and M. Bailey. An internet-wide view of ics devices. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 96–103, Dec 2016.

[MO14] David Malone and K.J. O'Dwyer. Bitcoin mining and its energy footprint. pages 280–285, 01 2014. Online: Accessed 10-December-2019.

[MOG11] Majid Meghdadi, Suat Ozdemir, and Inan Güler. A survey of wormhole-based attacks and their countermeasures in wireless sensor networks. *IETE technical review*, 28(2):89–102, 2011.

[MP85] J. Mogul and J. Postel. Internet standard subnetting procedure. STD 5, RFC Editor, August 1985. http://www.rfc-editor.org/rfc/rfc950.txt.

[MPB$^+$13] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Avi Patel, and Muttukrishnan Rajarajan. A survey on security issues and solutions at different layers of cloud computing. *The journal of supercomputing*, 63(2):561–592, 2013.

[MPSB18] Philipp Morgner, Stefan Pfennig, Dennis Salzner, and Zinaida Benenson. Malicious iot implants: Tampering with serial communication over the internet. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 535–555, Cham, 2018. Springer International Publishing.

[MRR$^+$17] Caleb Mays, Mason Rice, Benjamin Ramsey, John Pecarina, and Barry Mullins. Defending building automation systems using decoy networks. In *International Conference on Critical Infrastructure Protection*, pages 297–317. Springer, 2017.

[MRR18] S. M. MirhoseiniNejad, A. Rahmanpour, and S. M. Razavizadeh. Phase jamming attack: A practical attack on physical layer-based key derivation. In *2018 15th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, pages 1–4, 2018.

[MS12] V. C. Manju and K. M. Sasi. Detection of jamming style dos attack in wireless sensor network. In *2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing*, pages 563–567, 2012.

[MT11] A. K. Mishra and A. K. Turuk. Adversary information gathering model for node capture attack in wireless sensor networks. In *2011 International Conference on Devices and Communications (ICDeCom)*, pages 1–5, 2011.

[MT12] Nateq Be-Nazir Ibn Minar and Mohammed Tarique. Bluetooth security threats and solutions: a survey. *International Journal of Distributed and Parallel Systems*, 3(1):127, 2012.

[MVBC12] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: your finger taps have fingerprints. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 323–336, 2012.

[MVCT11] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 551–562, 2011.

[Nas05] Daniel Charles Nash. *An Intrusion Detection System for Battery Exhaustion Attacks on Mobile Computers.* PhD thesis, Virginia Tech, 2005.

[Nat20a] Nate Lord. Supply chain cybersecurity: Experts on how to mitigate third party risk. https://digitalguardian.com/blog/supply-chain-cybersecurity, 2020. Online: Accessed 25-September-2020.

[Nat20b] National Cyber Security Centre. Supply chain security guidance. https://www.ncsc.gov.uk/collection/supply-chain-security/supply-chain-attack-examples, 2020. Online: Accessed 10-November-2020.

[NBGT19] Michael Nast, Björn Butzin, Frank Golatowski, and Dirk Timmermann. Performance analysis of a secured bacnet/ip network. In *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 1–8, 2019.

[New09] CBS News. Viruses frame pc owners for child porn, Nov 2009. Online: Accessed 10-December-2019.

[Ngu15] Trang Nguyen. Using unrestricted mobile sensors to infer tapped and traced user inputs. In *2015 12th International Conference on Information Technology-New Generations*, pages 151–156. IEEE, 2015.

[Nit13] Nitdroid. How to Access Control4 through Putty. https://nitdroid.wordpress.com/2013/07/30/how-to-access-control4-through-putty/, 2013. Online: Accessed 25-September-2020.

[NSBU20] AKM Iqtidar Newaz, Amit Kumar Sikder, Leonardo Babun, and A Selcuk Uluagac. Heka: A novel intrusion detection system for attacks to personal medical devices. In *IEEE Conference on Communications and Network Security (CNS)*, 2020.

[NSMS15] M. Niemietz, J. Somorovsky, C. Mainka, and J. Schwenk. Not so smart: On smart tv apps. In *2015 International Workshop on Secure Internet of Things (SIoT)*, pages 72–81, Sep. 2015.

[NSN14] Sashank Narain, Amirali Sanatinia, and Guevara Noubir. Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, pages 201–212, 2014.

[NSRU19] AKM Iqtidar Newaz, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A Selcuk Uluagac. Healthguard: A machine learning-based security framework for smart healthcare systems. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 2019.

[NSRU20] AKM Newaz, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A Selcuk Uluagac. A survey on security and privacy issues in modern healthcare systems: Attacks and defenses. *arXiv preprint arXiv:2005.07359*, 2020.

[NZV20] Jianbing Ni, Kuan Zhang, and Athanasios V Vasilakos. Security and privacy for mobile edge caching: challenges and solutions. *IEEE Wireless Communications*, 2020.

[OAH18] Opeyemi Osanaiye, Attahiru S Alfa, and Gerhard P Hancke. A statistical approach to detect jamming attacks in wireless sensor networks. *Sensors*, 18(6):1691, 2018.

[ODO17] Alma Oracevic, Selma Dilek, and Suat Ozdemir. Security in internet of things: A survey. In *2017 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE, 2017.

[oE18] U.S Department of Energy. Cyber Security for Lighting Systems. https://www.energy.gov/sites/prod/files/2018/06/f52/cyber_security_lighting.pdf, 2018. Online: Accessed 15-August-2020.

[OHA$^+$14] O. Olawumi, K. Haataja, M. Asikainen, N. Vidgren, and P. Toivanen. Three practical attacks against zigbee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned. In *2014 14th International Conference on Hybrid Intelligent Systems*, pages 199–206, Dec 2014.

[OHD$^+$12] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, pages 1–6, 2012.

[OK14] Yossef Oren and Angelos D. Keromytis. From the aether to the ethernet—attacking the internet using broadcast digital television. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 353–368, 2014.

[Pak20a] Pakedge. Bakpak remote management & monitoring. https://pakedge.com/bakpak/, 2020. Online: Accessed 18-July-2020.

[Pak20b] Pakedge. Pakedge zones. https://pakedge.com/technology/pakedge-zones.php, 2020. Online: Accessed 18-September-2020.

[PAL$^+$08] Kanthakumar Pongaliur, Zubin Abraham, Alex X Liu, Li Xiao, and Leo Kempel. Securing sensor nodes against side channel attacks. In *2008 11th IEEE High Assurance Systems Engineering Symposium*, pages 353–361. IEEE, 2008.

[PAS14] Fahad Polash, Abdullah Abuhussein, and Sajjan Shiva. A survey of cloud computing taxonomies: Rationale and overview. In *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*, pages 459–465. IEEE, 2014.

[Pau15a] Paul Lilly. Connected homes can be easy targets for hackers, says cybersecurity firm. https://www.techhive.com /article/2883246/connected-homes-can-be-easy-targets-for-hackers-says-cybersecurity-firm.html, 2015. Online: Accessed 25-September-2020.

[Pau15b] Paul Williams. Securing your Connected Life. https://www.control4.com/blog/113/securing-your-connected-life/, 2015. Online: Accessed 25-September-2020.

[PBAU20] L. C. PucheRondon, L. Babun, K. Akkaya, and A. S. Uluagac. Hdmi-watch: Smart intrusion detection system against hdmi attacks. *IEEE Transactions on Network Science and Engineering*, pages 1–1, 2020.

[PG16] Ankush B Pawar and Shashikant Ghumbre. A survey on iot applications, security challenges and counter measures. In *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, pages 294–299. IEEE, 2016.

[Pin19] Pinkoos. Apple tv tvos 13 killed my remote programming, 2019. Online: Accessed 20-May-2020.

[PS$^+$09] Dr G Padmavathi, Mrs Shanmugapriya, et al. A survey of attacks, security mechanisms and challenges in wireless sensor networks. *arXiv preprint arXiv:0909.0576*, 2009.

[PSJA15] Giuseppe Petracca, Yuqiong Sun, Trent Jaeger, and Ahmad Atamli. Audroid: Preventing attacks on audio channels in mobile devices. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 181–190, 2015.

[Pul18] Pulse-Eight. USB-CEC Adapter communication Library. https://github.com/Pulse-Eight/libcec/, 2018.

[Pus16a] Pushstack. Control4 driver decryption. https://pushstack.wordpress.com/2016/03/06/control4-driver-decryption/, 2016. Online: Accessed 18-May-2020.

[Pus16b] Pushstack. Somfy smoove origin rts protocol. https://pushstack.wordpress.com/somfy-rts-protocol/, 2016. Online: Accessed 18-May-2020.

[Pyt19] Python.org. tkinter — Python interface to Tcl/Tk, 2019. Online: Accessed 20-December-2019.

[RAL+17] A. Ramos, B. Aquino, M. Lazar, R. H. Filho, and J. J. P. C. Rodrigues. A quantitative model for dynamic security analysis of wireless sensor networks. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, 2017.

[RBA+20] Luis Puche Rondon, Leonardo Babun, Ahmet Aris, Kemal Akkaya, and A. Selcuk Uluagac. Poisonivy: (in)secure practices of enterprise iot systems in smart buildings, 2020.

[RBA+21] Luis Puche Rondon, Leonardo Babun, Ahmet Aris, Kemal Akkaya, and A. Selcuk Uluagac. Lightningstrike: (in)secure practices of e-iot systems in the wild. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '21, page 106–116, New York, NY, USA, 2021. Association for Computing Machinery.

[RBAU19a] Luis Puche Rondon, Leonardo Babun, Kemal Akkaya, and A. Selcuk Uluagac. Hdmi-walk: Attacking hdmi distribution networks via consumer electronic control protocol. In *35th Annual Computer Security Applications Conference*, 2019.

[RBAU19b] Luis Puche Rondon, Leonardo Babun, Kemal Akkaya, and A. Selcuk Uluagac. Hdmi-walk: Attacking hdmi distribution networks via consumer electronic control protocol. In *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019.

[Rem20] Remote Central. Index of Remote Control File Areas. http://files.remotecentral.com/index.html, 2020. Online: Accessed 25-September-2020.

[Ric17] Ricky Lawshae. Who Controls the Controllers - Hacking Crestron IoT Automation Systems. https://av.tib.eu/media/39726, 2017. Online: Accessed 25-September-2020.

[Riv17] RiverLoopSec. Framework and Tools for Attacking ZigBee and IEEE 802.15.4 networks. https://github.com/riverloopsec/killerbee, 2017.

[RM08] David R Raymond and Scott F Midkiff. Denial-of-service in wireless sensor networks: Attacks and defenses. *IEEE Pervasive Computing*, 7(1):74–81, 2008.

[Rob19] Rob Helmke. Tamper-Evident Packaging and Functionality. https://www.plasticingenuity.com/blog/tamper-evident-packaging-functionality, 2019. Online: Accessed 25-September-2020.

[RR18] Jordan Robertson and Michael Riley. The big hack: How china used a tiny chip to infiltrate us companies. *Bloomberg Businessweek*, 4, 2018.

[RRC16] Nirupam Roy and Romit Roy Choudhury. Listening through a vibration motor. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, page 57–69, New York, NY, USA, 2016. Association for Computing Machinery.

[RS16] Eyal Ronen and Adi Shamir. Extended functionality attacks on iot devices: The case of smart lights. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 3–12. IEEE, 2016.

[RSP11] C Muthu Ramya, M Shanmugaraj, and R Prabakaran. Study on zigbee technology. In *2011 3rd International Conference on Electronics Computer Technology*, volume 6, pages 297–301. IEEE, 2011.

[RSWO18] E. Ronen, A. Shamir, A. O. Weingarten, and C. O'Flynn. Iot goes nuclear: Creating a zigbee chain reaction. *IEEE Security Privacy*, 16(1):54–62, January 2018.

[RXT20] RXTX. RXTX - A Java Cross Platform Wrapper Library For The Serial Port. https://github.com/rxtx/rxtx, 2020. Online: Accessed 1-March-2020.

[Rya13] Mark D Ryan. Cloud computing security: The scientific challenge, and a survey of solutions. *Journal of Systems and Software*, 86(9):2263–2268, 2013.

[Sam20] Samsung. RS-232 on Samsung TV's. https://www.samsung.com /us/support/troubleshooting/TSG01201603/, 2020. Online: Accessed 25-September-2020.

[SAU17] Amit Kumar Sikder, Hidayet Aksu, and A Selcuk Uluagac. 6thsense: A context-aware sensor-based attack detector for smart devices. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 397–414, 2017.

[SAU19] Amit Kumar Sikder, Hidayet Aksu, and A Selcuk Uluagac. Context-aware intrusion detection method for smart devices with sensors, September 17 2019. US Patent 10,417,413.

[SAU20] A. K. Sikder, H. Aksu, and A. S. Uluagac. A context-aware framework for detecting sensor-based threats on smart devices. *IEEE Transactions on Mobile Computing*, 19(2):245–261, Feb 2020.

[Sav] Savant. Bulbs faq. https://www.savant.com/bulbs-faq. Online: Accessed 20-December-2019.

[Sav14] Savant. Savant Smart Lightning Deployment Guide. https://support .savant.com/pro, 2014. Online: Accessed 15-August-2020.

[Sav15] Neil Savage. Visualizing sound. *Communications of the ACM*, 58(2):15–17, January 2015.

[Sav20a] Savant. Savant Climate Control. https://www.savant.com/climate, 2020. Online: Accessed 20-June-2020.

[Sav20b] Savant. Savant Whole Home Audio. https://www.savant.com/whole-home-audio, 2020. Online: Accessed 25-September-2020.

[SB07] Dominic Spill and Andrea Bittau. Bluesniff: Eve meets alice and bluetooth. *WOOT*, 7:1–10, 2007.

[SBAU19] Amit Kumar Sikder, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. Aegis: A context-aware security framework for smart home systems. In *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019.

[SBC$^+$20] Amit Kumar Sikder, Leonardo Babun, Z. Berkay Celik, Abbas Acar, Hidayet Aksu, Patrick McDaniel, Engin Kirda, and A. Selcuk Uluagac. Kratos: Multi-user multi-device-aware access control system for the smart home. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020.

[SBM$^+$17] Lorenz Schwittmann, Christopher Boelmann, Viktor Matkovic, Matthäus Wander, and Torben Weis. Identifying tv channels & on-demand videos using ambient light sensors. *Pervasive and Mobile Computing*, 38:363–380, 2017.

[SC17] Ashish Singh and Kakali Chatterjee. Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*, 79:88–115, 2017.

[SDWV20] Jangirala Srinivas, Ashok Kumar Das, Mohammad Wazid, and Athanasios V Vasilakos. Designing secure user authentication protocol for big data collection in iot-based intelligent transportation system. *IEEE Internet of Things Journal*, 8(9):7727–7744, 2020.

[Set19] Shobhit Seth. What is botnet mining? https://www.investopedia.com /tech/what-botnet-mining/, 2019. Online: Accessed 23-January-2020.

[Sha14] Farrukh Shahzad. State-of-the-art survey on cloud computing security challenges, approaches and solutions. *Procedia Computer Science*, 37:357–362, 2014.

[Sho20] Shodan.io. Shodan: Analyze the internet in seconds. https://www .shodan.io/, 2020. Online: Accessed 22-September-2020.

[Sim20] Simon Tatham. PuTTY - a free SSH and telnet client for Windows. https://www.putty.org/, 2020. Online: Accessed 27-September-2020.

[SJP16] Saurabh Singh, Young-Sik Jeong, and Jong Hyuk Park. A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications*, 75:200–222, 2016.

[SJRB17] P. Sinha, V. K. Jha, A. K. Rai, and B. Bhushan. Security vulnerabilities, attacks and countermeasures in wireless sensor networks at various layers of osi reference model: A survey. In *2017 International Conference on Signal Processing and Communication (ICSPC)*, pages 288–293, 2017.

[SK11] Subashini Subashini and Veeraruna Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11, 2011.

[SKK16] H. M. Song, H. R. Kim, and H. K. Kim. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In *2016 International Conference on Information Networking (ICOIN)*, pages 63–68, 2016.

[SKR17] V. Shakhov, I. Koo, and A. Rodionov. Energy exhaustion attacks in wireless networks. In *2017 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, pages 1–3, 2017.

[SMCB16] Alessandro Sforzin, Félix Gómez Mármol, Mauro Conti, and Jens-Matthias Bohli. Rpids: Raspberry pi ids—a fruitful intrusion detection system for iot. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 440–448. IEEE, 2016.

[Smi15] Joshua Smith. High-Def Fuzzing : Exploring Vulnerabilities in HDMI-CEC. https://media.defcon.org/, Nov, 2015.

[SMS18] Da-Zhi Sun, Yi Mu, and Willy Susilo. Man-in-the-middle attacks on secure simple pairing in bluetooth standard v5. 0 and its countermeasure. *Personal and Ubiquitous Computing*, 22(1):55–67, 2018.

[Smu90] Peter Smulders. The threat of information theft by reception of electromagnetic radiation from rs-232 cables. *Computers & Security*, 9(1):53 – 58, 1990.

[SMW$^+$16] Lorenz Schwittmann, Viktor Matkovic, Torben Weis, et al. Video recognition using ambient light sensors. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. IEEE, 2016.

[Sna20] SnapAV. Episode® Electronics IR Flasher with LED Feedback. https://www.snapav.com/shop/en/snapav/episode-reg%3B-electronics-ir-flasher-with-led-feedback, 2020. Online: Accessed 20-June-2020.

[SNN13] Amirali Sanatinia, Sashank Narain, and Guevara Noubir. Wireless spreading of wifi aps infections using wps flaws: An epidemiological and experimental study. In *2013 IEEE Conference on Communications and Network Security (CNS)*, pages 430–437. IEEE, 2013.

[Som] Somfy. Revolutionizing home comfort control: Radio technology somfy. https://www.somfysystems.com/en-us/discover-somfy/technology/radio-technology-somfy. Online: Accessed 10-February-2020.

[Som20] Somfy. Control RTS Solutions with Most Automation Systems. https://www.somfysystems.com/en-us/products/1810872/universal-rts-interface, 2020. Online: Accessed 1-March-2020.

[SP08] Karen Scarfone and John Padgette. Guide to bluetooth security. *NIST Special Publication*, 800(2008):121, 2008.

[SPA17] Furrakh Shahzad, Maruf Pasha, and Arslan Ahmad. A survey of active attacks on wireless sensor networks and their countermeasures. *arXiv preprint arXiv:1702.07136*, 2017.

[SPA$^+$18] Amit Kumar Sikder, Giuseppe Petracca, Hidayet Aksu, Trent Jaeger, and A. Selcuk Uluagac. A survey on sensor-based threats to internet-of-things (iot) devices and applications. *CoRR*, abs/1802.02041, 2018.

[Spr14] Raphael Spreitzer. Pin skimming: Exploiting the ambient-light sensor in mobile devices. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 51–62, 2014.

[SSS11] Shio Kumar Singh, MP Singh, and Dharmendra K Singh. A survey on network security and attack defense mechanism for wireless sensor networks. *International Journal of Computer Trends and Technology*, 1(2):9–17, 2011.

[Sta10] François-Xavier Standaert. Introduction to side-channel attacks. In *Secure integrated circuits and systems*, pages 27–42. Springer, 2010.

[Sta19] Stacey McDaniel. What is a legacy system? https://www.talend.com/resources/what-is-legacy-system/, 2019. Online: Accessed 20-June-2020.

[Ste15] Stephen Genusa. Crestron cresnet monitor. https://pushstack.wordpress.com/somfy-rts-protocol/, 2015. Online: Accessed 18-May-2020.

[Str07] Andreas A Strikos. A full approach for intrusion detection in wireless sensor networks. *School of Information and Communication Technology*, 2007.

[Sul] Mohamed Sultan. Smart to smarter: Smart home systems history, future and challenges. Online: Accessed 10-December-2019.

[Sup20] NETGEAR Support. What is a vlan? https://kb.netgear.com/24720/What-is-a-VLAN, 2020. Online: Accessed 18-September-2020.

[Sus19] Susan Morrow. The Dangers of “Rolling Your Own” Encryption. https://resources.infosecinstitute.com/topic/the-dangers-of-rolling-your-own-encryption/, 2019. Online: Accessed 10-November-2020.

[SW05] Yaniv Shaked and Avishai Wool. Cracking the bluetooth pin. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 39–50, 2005.

[Swa18] Swann Security. Nvr vs. dvr – what’s the difference? https://www.swann.com/blog/dvr-vs-nvr-whats-the-difference/, 2018. Online: Accessed 18-July-2020.

[Syn15] Synack. Home Automation Benchmarking Results. https://www.synack.com/blog/home-automation-benchmarking-results/, 2015. Online: Accessed 25-September-2020.

[SZLJ14] Mingshen Sun, Min Zheng, John CS Lui, and Xuxian Jiang. Design and implementation of an android host-based intrusion prevention system. In *Proceedings of the 30th annual computer security applications conference*, pages 226–235, 2014.

[Tec20] Technopedia. Shipments of Products with HDMI Interface Nears 900 Million Devices in 2017; Total Installed Base Approaches Seven Billion. https://www.techopedia.com/definition/630/infrared-ir, 2020. Online: Accessed 20-June-2020.

[Tex14] Texas Instruments. Data communication protocol for control networks enabling automated buildings. http://www.ti.com /lit/wp/spry266/spry266.pdf, 2014.

[Thr20] Threat Intelligence Team. SolarWinds advanced cyberattack: What happened and what to do now. https://blog.malwarebytes.com/threat-analysis/2020/12/advanced-cyber-attack-hits-private-and-public-sector-via-supply-chain-software-update/, 2020. Online: Accessed 10-January-2021.

[TK10] Mohammad Tehranipoor and Farinaz Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE design & test of computers*, 27(1):10–25, 2010.

[Tsu08] Akihiro Tsutsui. Latest trends in home networking technologies. *IEICE transactions on communications*, 91(8):2470–2476, 2008.

[UKTB15] R. Upadhyay, S. Khan, H. Tripathi, and U. R. Bhatt. Detection and prevention of ddos attack in wsn for aodv and dsr using battery drain. In *2015 International Conference on Computing and Network Communications (CoCoNet)*, pages 446–451, 2015.

[USB] A.S. Uluagac, V. Subramanian, and R. Beyah. Sensory channel threats to cyber physical systems: A wake-up call. In *IEEE Conference on Communications and Network Security (CNS), 2014*, pages 301–309.

[USB14] A Selcuk Uluagac, Venkatachalam Subramanian, and Raheem Beyah. Sensory channel threats to cyber physical systems: A wake-up call. In *2014 IEEE Conference on Communications and Network Security*, pages 301–309. IEEE, 2014.

[VDM13] K Venkatraman, J Vijay Daniel, and G Murugaboopathi. Various attacks in wireless sensor network: Survey. *International Journal of Soft Computing and Engineering (IJSCE)*, 3(1):208–212, 2013.

[VE85] Wim Van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers & Security*, 4(4):269–286, 1985.

[Ven15] Steve Venuti. HDMI Interface Extends Exceptional Digital Quality with Single-Cable Simplicity to Over 4 Billion Consumer Devices. https://www.hdmi.org/press/press_release.aspx?prid=137, Jan, 2015. Online: Accessed 20-June-2020.

[Ver16] Veracity. Ipv4 vs ipv6: What's the difference? https://www.veracityglobal.com/resources/articles-and-white-papers/poe-explained-part-1.aspx, 2016. Online: Accessed 10-January-2020.

[Ver20] Verkada. Securing Your Video Surveillance Network. https://info.verkada.com/security/surveillance-network/, 2020. Online: Accessed 25-September-2020.

[VJ20] Ruchi Vishwakarma and Ankit Kumar Jain. A survey of ddos attacking techniques and defence mechanisms in the iot network. *Telecommunication systems*, 73(1):3–25, 2020.

[VNBd19] A. Volkova, M. Niedermeier, R. Basmadjian, and H. de Meer. Security challenges in control network protocols: A survey. *IEEE Communications Surveys Tutorials*, 21(1):619–639, 2019.

[VP09] Martin Vuagnoux and Sylvain Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *USENIX security symposium*, pages 1–16, 2009.

[VP17] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in wpa2. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1313–1328, 2017.

[VR20] Mathy Vanhoef and Eyal Ronen. Dragonblood: Analyzing the dragonfly handshake of wpa3 and eap-pwd. In *Proceedings of the 2020 IEEE Symposium on Security and Privacy-S&P 2020)*. IEEE, 2020.

[WAM14] Lindsey N Whitehurst, Todd R Andel, and J Todd McDonald. Exploring security in zigbee networks. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, pages 25–28, 2014.

[Wan13] Jianfeng Wang. Zigbee light link and its applicationss. *IEEE Wireless Communications*, 20(4):6–7, 2013.

[WDBV20] Mohammad Wazid, Ashok Kumar Das, Vivekananda Bhat, and Athanasios V Vasilakos. Lam-ciot: Lightweight authentication mechanism in cloud-based iot environment. *Journal of Network and Computer Applications*, 150:102496, 2020.

[WDK$^+$17] Mohammad Wazid, Ashok Kumar Das, Neeraj Kumar, Mauro Conti, and Athanasios V Vasilakos. A novel authentication and key agreement scheme for implantable medical devices deployment. *IEEE journal of biomedical and health informatics*, 22(4):1299–1309, 2017.

[Wel00] Nicholas Wells. Busybox: A swiss army knife for linux. *Linux J.*, 2000(78es):10–es, October 2000.

[WG06] Ryan Winfield and Mark Gerrior. Avoiding Interference in the 2.4-GHz ISM Band. https://www.eetimes.com/avoiding-interference-in-the-2-4-ghz-ism-band/, 2006. Online: Accessed 20-October-2020.

[WH14] Wen-Chieh Wu and Shih-Hao Hung. Droiddolphin: a dynamic android malware detection framework using big data and machine learning. In *Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems*, pages 247–252, 2014.

[Wil18] Paul Lawrence Wilson. ModSec: A Secure Modbus Protocol. Master's thesis, Georgia Institute of Technology, 2018.

[WJZ10] Ying Wang, Zhigang Jin, and Ximan Zhao. Practical defense against wep and wpa-psk attack for wlan. In *2010 6th international conference on wireless communications networking and mobile computing (WiCOM)*, pages 1–4. IEEE, 2010.

[WMSL11] Matthias Wilhelm, Ivan Martinovic, Jens B. Schmitt, and Vincent Lenders. Short paper: Reactive jamming in wireless networks: How realistic is the threat? In *Proceedings of the Fourth ACM Conference on Wireless Network Security*, WiSec '11, page 47–52, New York, NY, USA, 2011. Association for Computing Machinery.

[WNK$^+$20a] Jianliang Wu, Yuhong Nan, Vireshwar Kumar, Mathias Payer, and Dongyan Xu. Blueshield: Detecting spoofing attacks in bluetooth low energy networks. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020)*, pages 397–411, 2020.

[WNK$^+$20b] Jianliang Wu, Yuhong Nan, Vireshwar Kumar, Dave Jing Tian, Antonio Bianchi, Mathias Payer, and Dongyan Xu. {BLESA}: Spoofing attacks against reconnections in bluetooth low energy. In *14th {USENIX} Workshop on Offensive Technologies ({WOOT} 20)*, 2020.

[Wri18] Doug Wright. Shipments of Products with HDMI Interface Nears 900 Million Devices in 2017; Total Installed Base Approaches Seven Billion, Jan, 2018. Online: Accessed 20-June-2020.

[WS04] Anthony D Wood and John A Stankovic. A taxonomy for denial-of-service attacks in wireless sensor networks. *Handbook of sensor networks: compact wireless and wired sensing systems*, pages 739–763, 2004.

[WSS03] Anthony D Wood, John A Stankovic, and Sang Hyuk Son. Jam: A jammed-area mapping service for sensor networks. In *RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003*, pages 286–297. IEEE, 2003.

[WSZ07] Anthony D Wood, John A Stankovic, and Gang Zhou. Deejam: Defeating energy-efficient jamming in ieee 802.15. 4-based wireless networks. In *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 60–69. IEEE, 2007.

[WWT$^+$20] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang. Understanding node capture attacks in user authentication schemes for wireless sensor networks. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2020.

[WYZ$^+$15] Xiaolei Wang, Yuexiang Yang, Yingzhi Zeng, Chuan Tang, Jiangyong Shi, and Kele Xu. A novel hybrid mobile malware detection system integrating anomaly detection with misuse detection. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, pages 15–22, 2015.

[XBZ12] Zhi Xu, Kun Bai, and Sencun Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 113–124, 2012.

[XFM14] Yi Xu, Jan-Michael Frahm, and Fabian Monrose. Watching the watchers: Automatically inferring tv content from outdoor light effusions. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 418–428, 2014.

[XSJ$^+$10] Kai Xing, Shyaam Sundhar Rajamadam Srinivasan, Major Jose, Jiang Li, Xiuzhen Cheng, et al. Attacks and countermeasures in sensor networks: a survey. In *Network security*, pages 251–272. Springer, 2010.

[XX12] Zhifeng Xiao and Yang Xiao. Security and privacy in cloud computing. *IEEE communications surveys & tutorials*, 15(2):843–859, 2012.

[XZ15] Zhi Xu and Sencun Zhu. Semadroid: A privacy-aware sensor management framework for smartphones. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 61–72, 2015.

[YFT15] K. Yang, D. Forte, and M. M. Tehranipoor. Protecting endpoint devices in iot supply chain. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 351–356, 2015.

[YFT17] Kun Yang, Domenic Forte, and Mark M. Tehranipoor. Cdta: A comprehensive solution for counterfeit detection, traceability, and authentication in the iot supply chain. 22(3), April 2017.

[YFT18] Kun Yang, Domenic Forte, and Mark Tehranipoor. Resc: An rfid-enabled solution for defending iot supply chain. 23(3), February 2018.

[YLDL17] Yang Yang, Ximeng Liu, Robert H Deng, and Yingjiu Li. Lightweight sharable and traceable secure mobile health system. *IEEE Transactions on Dependable and Secure Computing*, 17(1):78–91, 2017.

[YMF20] Narges Yousefnezhad, Avleen Malhi, and Kary Främling. Security in product lifecycle of iot devices: A survey. *Journal of Network and Computer Applications*, page 102779, 2020.

[YT08] Zhenwei Yu and Jeffrey JP Tsai. A framework of machine learning based intrusion detection for wireless sensor networks. In *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*, pages 272–279. IEEE, 2008.

[YWY$^+$17] Yuchen Yang, Longfei Wu, Guisheng Yin, Lijie Li, and Hongbin Zhao. A survey on security and privacy issues in internet-of-things. *IEEE Internet of Things Journal*, 4(5):1250–1258, 2017.

[YZG$^+$19] Yang Yang, Xianghan Zheng, Wenzhong Guo, Ximeng Liu, and Victor Chang. Privacy-preserving smart iot-based healthcare big data storage and self-adaptive access control system. *Information Sciences*, 479:567–592, 2019.

[YZOB19] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom. Survey of automotive controller area network intrusion detection systems. *IEEE Design Test*, 2019.

[YZV14] Zheng Yan, Peng Zhang, and Athanasios V Vasilakos. A survey on trust management for internet of things. *Journal of network and computer applications*, 42:120–134, 2014.

[Z-W18] Z-Wave. Safer, Smarter, Zwave. http://www.z-wave.com/, 2018.

[Zap17] Zaphod. Why is lua used for control4 driver programming, May, 2017.

[ZCDV17] Jun Zhou, Zhenfu Cao, Xiaolei Dong, and Athanasios V Vasilakos. Security and privacy for cloud-based iot: Challenges. *IEEE Communications Magazine*, 55(1):26–33, 2017.

[ZG13] Kai Zhao and Lina Ge. A survey on the internet of things security. In *2013 Ninth international conference on computational intelligence and security*, pages 663–667. IEEE, 2013.

[Zig18] Zigbee Alliance. Zigbee. www.zigbee.org/, 2018.

[ZS15] Tobias Zillner and Sebastian Strobl. Zigbee exploited: The good, the bad and the ugly. *Black Hat – 2015 https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly. pdf (21.03. 2018)*, 2015.

[ZWD$^+$20] Yue Zhang, Jian Weng, Rajib Dey, Yier Jin, Zhiqiang Lin, and Xinwen Fu. Breaking secure pairing of bluetooth low energy using downgrade attacks. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 37–54, 2020.

[ZWF$^+$21] Dan Zhang, Qing-Guo Wang, Gang Feng, Yang Shi, and Athanasios V Vasilakos. A survey on attack detection, estimation and control of industrial cyber–physical systems. *ISA transactions*, 2021.

[ZYJ$^+$17] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 103–117, 2017.

[ZZY18] Zheng Zhou, Weiming Zhang, and Nenghai Yu. Irexf: Data exfiltration from air-gapped networks by infrared remote control signals. *arXiv preprint arXiv:1801.03218*, 2018.

VITA

LUIS C. PUCHE RONDON

2011-2016	B.S., Computer Science Florida International University Miami, Florida
2016-2017	M.S., Cybersecurity Florida International University Miami, Florida
2017-2021	Doctoral Degree Electrical and Computer Engineering Florida International University Miami, Florida

SELECTED PUBLICATIONS, PATENTS, AND INVENTION DISCLOSURES

L. Puche Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac. *"Survey on Enterprise Internet-of-Things Systems (E-IoT): A Security Perspective."*, Elsevier AdHoc Networks, 2021.

L. Puche Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac. *"LGuard: Securing Enterprise-IoT Systems against Serial-based Attacks via Proprietary Communication Buses"*, ACM DTRAP, Under Review.

L. Puche Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac. *"LightningStrike: (In)secure practices of E-IoT systems in the wild."*, ACM WiSec, 2021.

L. Puche Rondon, L. Babun, K. Akkaya, and A. S. Uluagac. *"PATENT: A Method for Detecting Unexpected HDMI Consumer Electronics Control Protocol Activities using Machine Learning and Packet Attribute Analysis"*, FIU, 2020.

L. Puche Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac. *"PoisonIvy: (In)secure Practices of Enterprise IoT Systems in Smart Buildings."*, ACM BuildSys, 2020.

L. Puche Rondon, L. Babun, K. Akkaya, and A. S. Uluagac. *"HDMI-Watch: Smart Intrusion Detection System Against HDMI Attacks"*, in IEEE TNSE, 2020.

L. Puche Rondon, L. Babun, K. Akkaya, and A. S. Uluagac. *"HDMI-Walk: attacking HDMI distribution networks via consumer electronics control protocol."* ACM

ACSAC, 2019.

L. Puche Rondon, L. Babun, K. Akkaya, and A. S. Uluagac. *"Attacking HDMI distribution networks: poster."*, ACM WiSec, 2019.

L. Puche Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac. *"Ivycide: Smart Intrusion Detection System against E-IoT Driver Threats."*, ACM TOPS, Under Review.