LGuard: Securing Enterprise-IoT Systems against Serial-Based Attacks via Proprietary Communication Buses

LUIS PUCHE RONDON, LEONARDO BABUN, AHMET ARIS, KEMAL AKKAYA, and A. SELCUK ULUAGAC, Florida International University

Enterprise Internet of Things (E-IoT) systems allow users to control audio, video, scheduled events, lightning fixtures, door access, and relays in complex smart installations. These systems are widely used in government or smart private offices, smart buildings/homes, conference rooms, schools, hotels, and similar professional settings. However, even with their widespread use, the security of many E-IoT systems and components has not been researched in the literature. To address this research gap, we focus on E-IoT communication buses, one of the core components used to connect E-IoT devices, and introduce LightningStrike attacks that demonstrate several weaknesses with E-IoT proprietary communication protocols used in E-IoT communication buses. Specifically, we show that popular E-IoT proprietary communication protocols are susceptible to Denial-of-Service (DoS), eavesdropping, impersonation, and replay attacks. As such threats cannot be mitigated through traditional defense mechanisms due to the limitations posed by E-IoT, we propose LGuard, a defense system to protect E-IoT systems against the attacks over communication buses. LGuard uses closed-circuit television footage and computer vision techniques to detect replay attacks. For impersonation and DoS attacks, LGuard utilizes traffic analysis. Finally, LGuard obfuscates the E-IoT traffic via inserting redundant traffic to the bus against eavesdropping attacks. We evaluated the performance of LGuard in a realistic E-IoT deployment, and our detailed evaluations show that LGuard achieves an overall accuracy and precision of 99% in detecting DoS, impersonation, and replay attacks while effectively increasing the difficulty of extracting valuable information for eavesdroppers. In addition, LGuard does not incur any operational overhead or modification to the existing E-IoT system.

CCS Concepts: • **Security and privacy** → **Systems security**; *Denial-of-service attacks*; **Network security;**

Additional Key Words and Phrases: Enterprise Internet-of-Things, E-IoT security, defense system, attacks, Cresnet

ACM Reference format:
Luis Puche Rondon, Leonardo Babun, Ahmet Aris, Kemal Akkaya, and A. Selcuk Uluagac. 2023. LGuard: Securing Enterprise-IoT Systems against Serial-Based Attacks via Proprietary Communication Buses. *Digit. Threat.: Res. Pract.* 4, 1, Article 10 (March 2023), 26 pages.
https://doi.org/10.1145/3555721

This work was partially supported by the U.S. National Science Foundation (Awards: NSF-CAREER CNS-1453647, NSF-1663051, and NSF-2219920) and a Microsoft Research Grant. The views are those of the authors only.

Authors' addresses: L. Puche Rondon (corresponding author), L. Babun, A. Aris, and A. S. Uluagac, Cyber-Physical Systems Security Lab, Florida International University, 11200 SW 8th St, Miami, FL 33199; emails: {lpuch002, lbabu002, aaris, suluagac}@fiu.edu; K. Akkaya, Advanced Wireless and Security Lab, Florida International University, 10555 W Flagler St, Miami, FL 33174; email: kakkaya@fiu.edu.

2576-5337/2023/03-ART10 $15.00
https://doi.org/10.1145/3555721

1 INTRODUCTION

The rapid adoption of specialty smart systems has changed the lives of millions of users worldwide [28]. As part of these ecosystems, **Enterprise Internet of Things (E-IoT)** systems 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).

Although 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. In addition, although E-IoT systems are widely used, the security of many E-IoT systems and components has not been researched in the literature. The motivation of this work is to shed light upon the security of E-IoT systems, uncover vulnerabilities of E-IoT proprietary communication protocols that can affect millions of E-IoT deployments, and propose practical solutions. Here, we aim to address this open research problem and determine if E-IoT systems are susceptible to attacks by focusing 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 of 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 [37]. 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 to an attacker's advantage. With LightningStrike, we show 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.

As LightningStrike provides attackers with an effective and practical mechanism to compromise E-IoT systems, 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 introduce LGuard, a defense system designed to protect E-IoT systems against communication bus attacks. 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 video captures of the existing **closed-circuit television (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.

Contributions. The contributions of this work are as follows:

- We introduce LightningStrike, a set of novel 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 LightningStrike.
- We test LightningStrike attacks in a realistic E-IoT testbed and leverage communication buses to cause undesired behavior on behalf of an attacker.
- We propose LGuard, a practical defense system designed to protect E-IoT deployments against LightningStrike-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.

Organization. The rest of this work is organized as follows. Section 2 provides background information on E-IoT, protocols, and the communication buses. Section 3 presents the definitions, problem scope, and the threat model. In Section 4, we cover the architecture, attack implementation, and evaluation of LightningStrike attacks. Section 5 covers the terminology, challenges, and architecture of LGuard. Section 6 covers the implementation of LGuard. In Section 7, LGuard is evaluated. Section 8 discusses the benefits of LGuard. In Section 9, related work is highlighted. Finally, we conclude the article in Section 10.

2 ENTERPRISE INTERNET-OF-THINGS

In this section, we highlight background information on E-IoT systems and protocols used in E-IoT communication.

2.1 E-IoT Systems

E-IoT systems are closed-source smart systems that follow unique design and deployment practices, separating them from off-the-shelf IoT systems [45]. Specifically, E-IoT systems generally come at a higher cost and are more complex than off-the-shelf IoT systems. E-IoT deployments require specialized training and proprietary tools, as they are customized and configured according to users' specifications [12, 14]. As a result, a trained programmer and installer, also known as an *integrator*, is needed to configure E-IoT systems. Integrators perform the physical installation, testing, device configuration, and future technical support of the E-IoT system for a client. Further, E-IoT equipment is not usually sold to the end user, and the user must rely on the integrator for any service or modification to the E-IoT system. This complexity and added functionality have led E-IoT systems to become popular in locations such as classrooms, conference rooms, smart buildings, smart offices, yachts, and luxury smart homes. We highlight common uses of E-IoT systems in Figure 1.

A generalized implementation of an E-IoT system is shown in Figure 2, 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 E-IoT components, such as a controller, a power supply, and light control modules. The *controller* is the core processing unit of an E-IoT and contains the execution logic for user actions on controlled devices (e.g., pressing button 1 on a keypad opens a security door, and pressing button 6 turns off all lights). This highly programmable component is configured by the integrator during deployment or maintenance stages of an E-IoT according to a user's specification. The *power supply* of an E-IoT system powers keypads and other interfaces integrated into the core system as well as the controller and light control modules. The *lighting control modules* are the physical high-voltage and relay-based interfaces between the E-IoT system and *controlled devices*. Controlled devices are any light fixture, shade, relay-operated door, or any physical device controlled by the E-IoT system. Finally, the *communication buses* are the daisy-chain lines that traverse through different

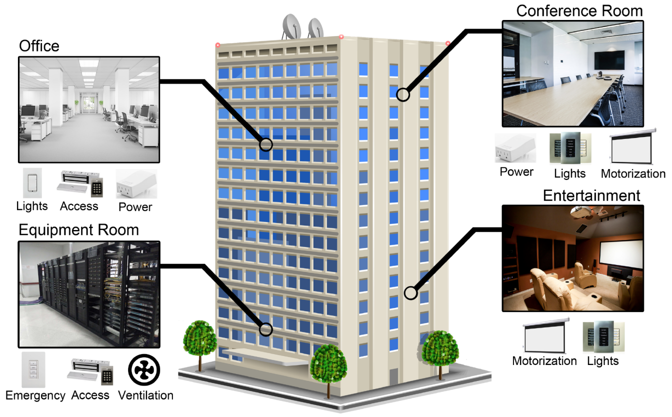


Fig. 1. Use cases of E-IoT systems.

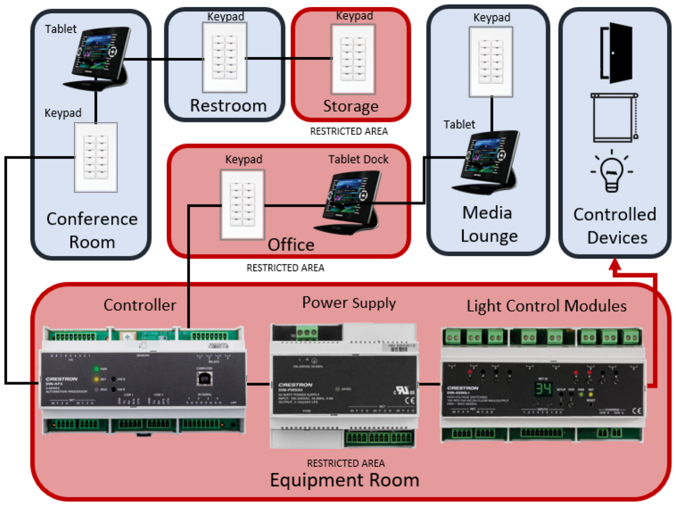


Fig. 2. An example E-IoT system with wired bus communication and two daisy-chain paths. Restricted areas are highlighted in red and common areas are in blue.

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, whereas other interfaces are only accessible in restricted locations. As Figure 2 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.

2.2 E-IoT Protocols

E-IoT supports a variety of protocols. Some supported protocols are widely known and well documented (e.g., Zigbee, Z-Wave, and TCP/IP), whereas 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 [13]. Lutron, a vendor that focuses on E-IoT lighting control systems, implemented a proprietary wireless communication known as Somfy's RTS (Radio Technology Somfy) [36, 48]. 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 [33]. For similar purposes, Control4 employs a proprietary communication protocol [9]. Savant uses communication buses and proprietary protocols for interfaces [46]. 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 [15]. 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 quite 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 cabling line [16]. Second, communication buses are seen as more reliable than wireless protocols over long distances where mesh networking has range limitations (60 feet) [26]. Third, wired E-IoT communication is not as susceptible to interference as a wireless one, creating a more reliable system [53]. 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.

3 PROBLEM SCOPE AND THREAT MODEL

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

3.1 Problem Scope

This work 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 via an available communication bus in a smart building.

With many wired communication bus interfaces, Mallory finds an unsupervised wired device such as a touchscreen 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 because 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. In addition, 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. We proposed an example scenario where an attacker can compromise an E-IoT deployment; however, 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 repainting 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 [32]. Conference rooms need E-IoT interfaces for the users to control projectors, 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 [21]; 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.

3.2 Definitions

In this section, we cover essential definitions for the concepts used in the LightningStrike 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 or 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 who 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).

3.3 Threat Model

LightningStrike 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, or 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

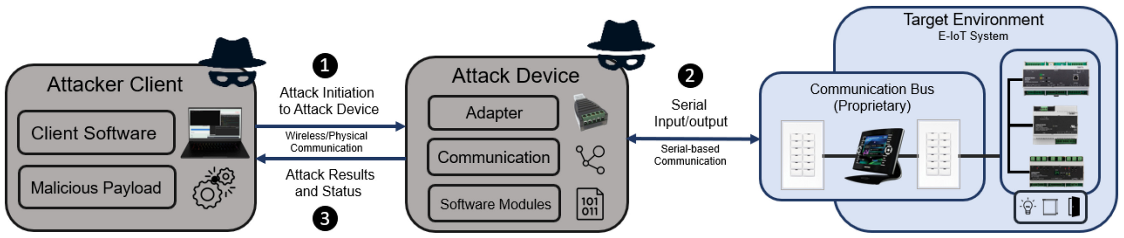


Fig. 3. General end-to-end implementation for LightningStrike-based attacks. Attack-related components are highlighted in gray and E-IoT components are in blue.

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.

Note that this work does not consider attacks that occur over TCP/IP networks, software vulnerabilities such as buffer overflows, or attacks on individual devices (e.g., keypad firmware exploits). Similarly, other protocols used in E-IoT such as Bluetooth, Ethernet, USB, and Zigbee are entirely outside the scope of this work.

4 LightningStrike ATTACKS

In this section we describe the architecture, end-to-end implementation, and evaluation of LightningStrike attacks. The complete details of LightningStrike can be found in our previous publication [44].

4.1 LightningStrike Attacks Overview

We highlight the architecture of LightningStrike in Figure 3. 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 LightningStrike'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 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, the 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 LightningStrike 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 the attack device's communication sub-module as the attacks are executed ❸.

4.2 Cresnet Protocol Analysis

As Cresnet is largely undocumented, we faced several 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. Although 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." In addition, 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

Table 1. Hardware and Software Used in LightningStrike Attacks Implementation and Evaluation

Hardware	Software
Crestron DIN-PWS50	Eclipse IDE 2020-03
Crestron C2N-DB12W x 3	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	–

Crestron controller periodically queries the existing Cresnet devices in the E-IoT system every 500 ms. Active traffic occurs when users interact with a Cresnet device. Actions such as button presses or disconnecting a keypad generate such active traffic.

4.3 LightningStrike Attacks Implementation

In this section, we describe the LightningStrike attacks implementation.

To evaluate LightningStrike 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 in annual revenue [37]. Specifically, we use LightningStrike to attack Crestron's Cresnet proprietary communication protocol. To ensure that LightningStrike attacks are demonstrated and evaluated realistically, we created an attack suite and a realistic E-IoT testbed as described in Table 1. 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.

Software modules implementation. To execute the LightningStrike attacks, we developed several software modules. The attacks were implemented using open-source tools available online:

(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. Although software such as Wireshark exists, programmed filtering was sufficient for testing purposes since the protocol is proprietary.

4.4 LightningStrike Attacks Evaluation

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

Attack 1: Flooding DoS. This attack was created to demonstrate that Threat 1 (DoS) is viable through LightningStrike by overwhelming the communication bus with messages. As the attack executed, the attacker's adapter flooded the Cresnet bus with repeated RS-485 messages to overwhelm communications with invalid packets. 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.

Attack 2: Malicious eavesdropping. Attack 2 demonstrates that Threat 2 (malicious eavesdropping) is viable on the Crestron testbed. This attack used the monitoring and filtering modules to observe and infer information from Cresnet packets. The attack monitored and gathered information from the Cresnet bus successfully due to Cresnet's lack of encryption. 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 three 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 DoS attack. This attack was designed to demonstrate another form of DoS attack using Threat 1 and Threat 3 (DoS and impersonation) through LightningStrike. 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. 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.

Attack 4: Replay attack. This attack was created to demonstrate that Threat 4 (communication replay) is possible on the Cresnet bus. Further, we highlight the implications of replay attacks on E-IoT systems. 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. 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. From our attacks, we concluded that without any form of security beyond obscurity, a knowledgeable attacker can easily compromise E-IoT to their benefit using the communication buses against insecure E-IoT proprietary protocols. All proposed attacks were implemented successfully, the implications of which clearly show the potential of communication bus 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. With Attack 3, creating a Cresnet ID conflict would be no issue for attackers, as all 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.

5 LGuard

To secure E-IoT systems against communication bus threats, we introduce LGuard, a defense system designed to protect E-IoT communication buses using traffic analysis, computer vision, and traffic obfuscation. In this section, we first discuss the design considerations and challenges for LGuard. We then define the necessary terminology. Finally, this section details the LGuard architecture and its individual components.

5.1 Design Considerations and Challenges

In this 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 necessitating 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 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 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 communication bus attacks.

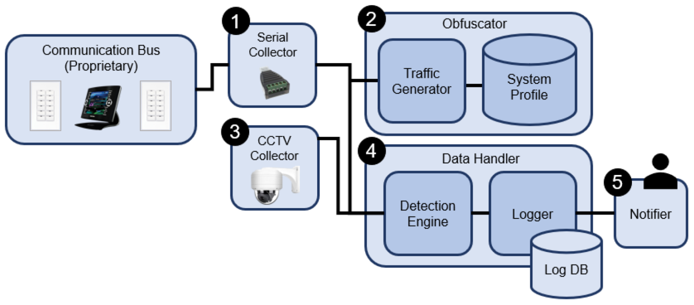


Fig. 4. The architecture of LGuard to defend E-IoT against attacks over E-IoT communication buses.

Security systems and assumptions. LGuard takes five assumptions. First, we assume that the E-IoT system is correctly configured and that all devices are functioning properly. Second, we assume 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 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 [10, 11, 18, 19]. Further, CCTV systems are expected to be present to be installed in secure locations. Third, we assume that at least one of the CCTV cameras can see the monitored E-IoT interfaces in secure locations. This is a viable and practical assumption, as security cameras can be adjusted to face different interfaces and not all interfaces require CCTV monitoring from LGuard. Further if no cameras are available, wireless cameras may be installed for LGuard installed in sensitive locations. Fourth, we make the assumption that the CCTV system is always recording. This is a realistic assumption, as CCTV systems are configured to always record when motion is detected. Finally, we assume that the CCTV system has not been compromised by the attacker, as it is in a separate network and compromising the CCTV system would require an added layer of difficulty than simply performing the LightningStrike attacks.

5.2 Terminology

This 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, such as 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.

5.3 LGuard Overview

LGuard is a novel defense system against attacks over communication buses in E-IoT deployments. It consists of tailored solutions against individual communication bus 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 using an 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 4. The first component is the Serial Collector ➊, 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 ➋ is used by LGuard 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 that 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 LightningStrike 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.

5.4 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 communication bus attacks. The Serial Collector also transmits the redundant E-IoT traffic generated by the Obfuscator component as needed.

5.5 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 for the E-IoT communication bus traffic, the Obfuscator component intends to make the job of the attacker harder by generating and inserting redundant traffic to the bus. The logic of the 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.

5.5.1 System Profile. The System Profile sub-component contains all 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 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 modifiable by the administrator or integrator performing the original configuration.

5.5.2 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 and then transmit them to the E-IoT system's communication bus through the Serial Collector. Packets are only transmitted when LGuard detects user activity. To mimic interface activity, the Traffic Generator first loads a random Cresnet IDs from non-existent Cresnet devices and generates idle and active traffic (Section 4.2). To do so, the

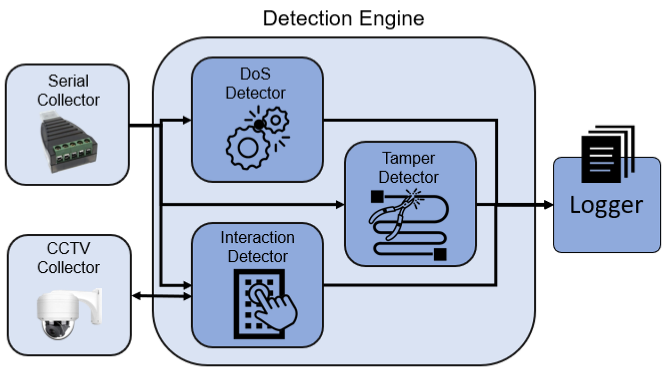


Fig. 5. Architecture of the Detection Engine component of LGuard that detects DoS, impersonation, and replay attacks via DoS Detector, Tamper Detector, and Interaction Detector modules, respectively.

Traffic Generator generates redundant communication packets to mimic keypad-to-controller communication every 500 ms. These generated packets should include idle traffic and active traffic as defined in Section 4. 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, E-IoT systems do not allow new keypads to be added without reconfiguring the controller, thus new device IDs cannot be added to the controller by the attacker.

5.6 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.

5.7 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.

5.7.1 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 5, the Detection Engine includes three defensive modules, each specifically designed to address different threats. As the name implies, the first defense module, DoS Detector, aims to detect high-throughput attacks such as DoS. The second module is the Tamper Detector, which intends to address impersonation attacks by detecting E-IoT interface tampering. The third module is the Interaction Detector used by LGuard for replay attack detection. Details with the modules are as follows.

DoS Detector. The DoS Detector detects DoS attacks over E-IoT communication by monitoring the network traffic and examining the amount of data received per second. As communication bus DoS attacks depend on high traffic volume, high throughput in the amount of data received indicates that a DoS attack is being attempted.

Tamper Detector. The 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 500-ms intervals. If a device does not respond, the controller starts to query the removed E-IoT interface with packets requesting a status response from the device. The Tamper Detector identifies attacks dependent on modifications to the E-IoT bus (e.g., impersonation) by identifying packets transmitted during the removal of an E-IoT device by the E-IoT controller. Since devices can malfunction in time and may not respond to the E-IoT controller without the any attempts of an adversary, the Tamper Detector also detects such situations and informs the network administrators.

Interaction Detector. The 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. At the time of a replay attack, the attacker replays the traffic of an E-IoT interface that the attacker has previously recorded. Hence, traffic seems to be generated by the E-IoT interface without any physical human activity on it. Based on this knowledge, the Interaction Detector uses computer vision and pose estimation to determine if E-IoT traffic is legitimate or part of a replay attack. For instance, if packets are detected from keypad 23, CCTV information should display physical interactions of a user and keypad 23. As such, the 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.

5.7.2 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 LGuard. Thus, the Log DB component should store communication packets and video feed used during the evaluation process that may be relevant for the administrator.

5.8 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 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.

6 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 2. Our testing E-IoT environment is identical to the environment used for the LightningStrike attacks implementation. We assume that the attacker executes the LightningStrike attacks in the same manner as defined in Section 4 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.

6.1 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 pre-processed data and current size of the buffer with the Data Handler.

Table 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

6.2 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 section, we cover the implementation of these sub-components.

6.2.1 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 IDs (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.

6.2.2 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, the Traffic Generator generates idle traffic for every non-existing E-IoT device ID in every 200 ms. 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 the Serial Collector. In terms of redundant active traffic generation, the Traffic Generator follows a probabilistic packet generation approach. Specifically, in 200-ms intervals, the 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 the 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.

6.3 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 recording and saves a 20-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.

Table 3. Observed Cresnet Communication Packets Used for LGuard, Where "ID" Is the Cresnet ID of the Device

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

6.4 Data Handler Implementation

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

6.4.1 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` [5, 35]. The Detection Engine first identifies pre-determined activities occurring in the communication bus as highlighted in Table 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: the 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 1,024 bytes per second, even under frequent usage of interfaces. The DoS Detector was thus implemented by detecting packet rates larger than 1,024 bytes per second. When transmission exceeds 1,024 bytes, the DoS Detector reports an attempted DoS attack.

Tamper Detector. Impersonation LightningStrike attacks occur by tampering with the E-IoT devices. In this regard, the 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, if an E-IoT device with ID 13 does not respond to the query of the controller, the controller starts to send queries starting with 13:00:00:FF. Hence, the Tamper Detector detects that the E-IoT device with ID 13 has been removed from the communication bus.

Interaction Detector. The Interaction Detector aims to detect replay attacks. To implement the Interaction Detector, the 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 the CCTV Collector sends the video capture, the Interaction Detector uses computer vision techniques to determine if a person is touching the interface in the video captures. This process requires the Interaction Detector 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 in video frames, the 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

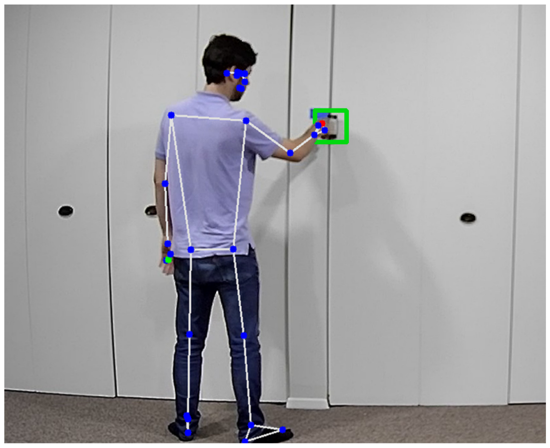


Fig. 6. LGuard pose detection on a keypad from CCTV footage. Red highlights the right hand, and the green point highlights the left hand. The green square highlights the interface location.

recognition library. This process is depicted in Figure 6, 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 pre-defined 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 the steps of the detection process, the 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.

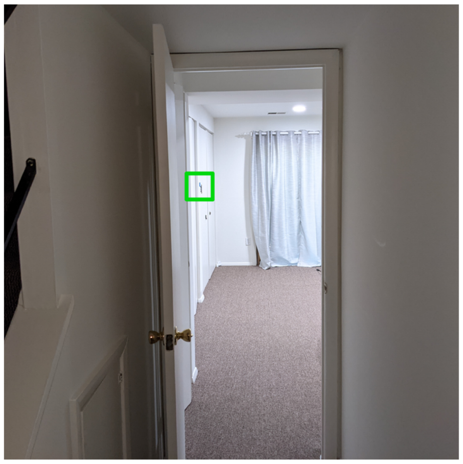
6.4.2 Logger. The Logger sub-component was implemented as a local software buffer collecting all logs and warnings from the LGuard system and detection process. These logs are then exported using Python JSON serialization to Log DB. The Log DB sub-component was implemented using the Python-based JSON 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 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, and MongoDB.

6.5 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 PERFORMANCE EVALUATION

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


(a) Side view of the Cresnet E-IoT interface in a smart living setting

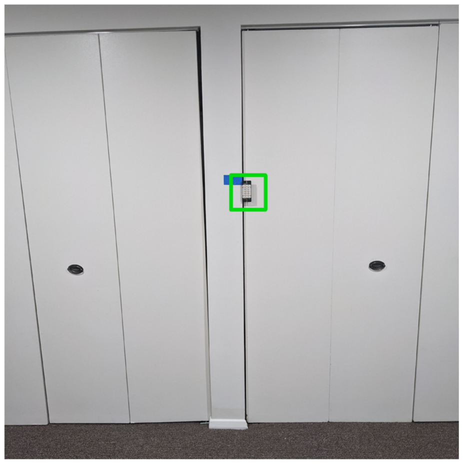

(b) Front view of the Cresnet E-IoT interface

Fig. 7. Side and front views of CCTV used for LGuard evaluation, with the keypads highlighted in green. Different angles were tested to evaluate pose estimation efficacy.

RQ1: DoS detection: How effective is LGuard's performance in LightningStrike Denial-of-Service attack detection (Section 7.2)?

RQ2: Impersonation detection: How well does LGuard identify tampering in the communication bus through passive monitoring (Section 7.3)?

RQ3: Replay attack detection: How effective is LGuard at identifying replay attacks using pose estimation and traffic monitoring (Section 7.4)?

RQ4: Traffic obfuscation: How well does traffic obfuscation mitigate LightningStrike eavesdropping (Section 7.5)?

In this section, we first explain the attack data collection process. Afterward, we answer each research question and finally evaluate the detection time and overhead of LGuard.

7.1 Attack Data Collection

For LGuard evaluation, we applied the LightningStrike attacks as specified in Section 4. We collected Cresnet traffic over the E-IoT communication bus. The activities collected for our evaluation included Cresnet traffic caused by the LightningStrike attacks and benign traffic generated by expected usage of the E-IoT system. In addition, 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. Details with the executed attacks are as follows:

- Twenty replay attacks and 20 benign cases in bright light conditions (front view of the interface)
- Twenty replay attacks and 20 benign cases in low light conditions (front view of the interface)
- Twenty replay attacks and 20 benign cases in bright light conditions (side view of the keypad)
- Twenty DoS attacks
- Twenty impersonation attacks in which keypad tampering takes place
- Twenty LGuard logs with obfuscated traffic in which each log consists of 2,000 packets
- Twenty logs without obfuscated traffic in which each log consists of 2,000 packets.

Table 4. LGuard Performance Evaluation on Replay Attacks

TP	TN	FP	FN	ACC	PREC	REC	F1
60	59	1	0	0.99	0.98	1.0	0.99

7.1.1 Performance Metrics. Performance metrics are measured with the following parameters: accuracy, precision, F-score, recall, **true positive (TP)**, **true negative (TN)**, **false positive (FP)**, and **false negative (FN)** [31]:

TP: TP denotes the total number of malicious cases correctly identified as malicious.
TN: TN denotes the total number of benign cases correctly identified as benign.
FP: FP denotes the total number of cases where a benign case is mistaken as malicious.
FN: FN denotes the total number of cases where a malicious case is mistaken as benign.

$$RecallRate = \frac{TP}{TP + FN} \tag{1}$$

$$PrecisionRate = \frac{TP}{TP + FP} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$F1 = \frac{2 * RecallRate * PrecisionRate}{RecallRate + PrecisionRate} \tag{4}$$

7.2 DoS Detection Performance (RQ1)

As part of *RQ1*, we evaluate LGuard's performance in detecting LightningStrike-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.

7.3 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 FPs, and LGuard correctly identified keypad removal with 100% precision and accuracy. As keypads are repeatedly queried by the E-IoT controller at about 500-ms intervals, tampering detection by LGuard occurred as soon as the keypad 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.

7.4 Replay Detection Performance (RQ3)

To answer *RQ3*, we evaluated the performance of LGuard against LightningStrike replay attacks. The results are outlined in Table 4. As noted, LGuard performance showed an overall accuracy and precision of 99% in identifying replay attacks over the Cresnet communication. Further, while two FPs were observed in two benign cases (CCTV capture in bright light), there were no FNs for any of the malicious cases. In one of these FP cases, pose vertices were not calculated properly by the Google MediaPipe library. In the other FP 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 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 FP interface interactions during the mislabeled frames. However, these mis-classified frames did not affect the overall detection by LGuard.

7.5 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 2. 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 for the bus traffic with and without traffic obfuscation. We recorded a total of 40 datasets (20 obfuscated and 20 unobfuscated). Throughout each dataset, each real keypad button (in total, we have three real keypads) was pressed four times at random times. Although the unobfuscated datasets employed only real traffic of keypads, the obfuscated dataset consisted of the real traffic of three keypads and the generated redundant traffic of four non-existing devices. When an attacker checks for the occurrence of button press/release packets (outlined in Table 3) on unobfuscated datasets, she can determine the activities happening in the communication bus with 100% certainty. However, when she checks the occurrence of button press/release packets on obfuscated datasets, she can identify the real activities in the communication bus with a 19% success ratio on average (total number of real button presses/(total number of real and non-real button presses)). 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.

7.6 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 1,000 ms. Impersonation attacks that tamper with the communication bus and keypads were also detected within 500 ms, as the polling messages observed from the E-IoT controller to continuously query the removed interfaces are transmitted within 500 ms. Finally, replay attacks were detected within 10 seconds, which is the length of each CCTV video capture. We note that the current LGuard testing had negligible delays in transferring video recordings (40 MB average) from the CCTV camera to the Detection Engine of LGuard in detecting replay attacks. This duration can be further reduced by using better processing hardware, multi-threading, and shorter video captures.

Although LGuard operates as a stand-alone system without modification of the E-IoT deployment, we measured the overhead on the host machine (16 GB of RAM and 17-700 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. Moreover, although additional packets are transmitted through the use of the Traffic Generator to obfuscate the traffic, the new messages provided no observable overhead or delays to the operation of the E-IoT system in our experiments.

8 BENEFITS AND LIMITATIONS

In this section, we highlight the benefits and discuss limitations of LGuard.

8.1 Benefits

Independent framework. LGuard and all associated components function as an independent defense system to 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.

Backward 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.

Implementation costs. The cost of LGuard is minimal, as it uses existing smart system components such as a 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 rewiring, LGuard provides an affordable solution for any administrator who 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. In addition, if CCTV cameras are not available, wireless cameras may be added for LGuard only as needed at a reasonable price (e.g., IP cameras are available for less than $30 at the time of this writing). Similarly, illuminators are affordable and can be installed to improve lighting conditions if needed.

Scalability. E-IoT systems may vary from small room-to-room deployments to large full-size deployments to multiple interconnected systems. As such, LGuard considers the needs for scalability to secure each and every type of system. Thus, because of the way that LGuard is designed, scalability is possible as long as each instance of LGuard has a physical connection to the E-IoT communication bus. Deployment services such as Docker may also allow for easy expandability of resources when multiple instances of LGuard are needed.

CCTV privacy. For LGuard, we take the assumption that the CCTV system has already been installed. Thus, there are no added privacy risks due to LGuard if the CCTV system already exists. However, although CCTV information is transmitted to LGuard, our proposed solution does not require privacy-concerning features such as facial recognition. Further, LGuard does not rely on transmitting CCTV recordings or information to external networks, thus CCTV recordings pose no risk of being exposed to external parties unless the network has already been compromised. Finally, LGuard does not store any CCTV data locally, reducing the impact of LGuard data being compromised as CCTV recordings would not be exposed in such a breach.

Reusability. Although it is difficult to determine how well LightningStrike and LGuard can be adapted to other E-IoT systems due to the lack of research in E-IoT protocols, LightningStrike and LGuard were designed to use features that are required for E-IoT communication buses. For LightningStrike attacks, we note that flooding, eavesdropping, impersonation, and replay attacks are all possible in any serial-based communication bus that uses insecure protocols. LGuard is designed to use components that should be compatible with other communication-bus networks (e.g., CCTV, passive monitoring, message obfuscation). For instance, for tamper detection, we assume that most E-IoT communication protocols have some method of monitoring if connected devices are active for the sake of normal operation. As such, although LGuard may need alterations or adaptations for each E-IoT communication bus protocol, most if not all concepts used in LGuard should function in all similar protocols.

8.2 Limitations

As LGuard is designed to function without modification to E-IoT device hardware or software, there are still some limitations with the design.

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 illuminators, especially for professional security systems. Infrared 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, stand-alone 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.

Malfunctioning devices. For the Tamper Detector module, it is not possible to distinguish between malicious tampering or a device falling offline (e.g., faulty wiring). Although we make the assumption that all devices are functioning at the deployment time of LGuard, it may be valuable for an integrator to know which devices might be tampered by an attacker or have fallen offline so that they may be replaced.

Obfuscator and the address space. As the Obfuscator provides n simulated devices on a Cresnet network with n real devices and Cresnet is limited to 252 devices [20], LGuard allows for a maximum of 126 real devices on a single Cresnet network. Although 126 real devices for small to medium E-IoT deployments may not be a limitation, for larger E-IoT deployments, 126 E-IoT devices may be a limitation. As large E-IoT deployments typically employ multiple Cresnet buses, it would be possible to run multiple instances of LGuard where each instance monitors a separate Cresnet bus and obfuscates the traffic in the assigned bus. Hence, large E-IoT deployments that have more than 126 real E-IoT devices can be supported by LGuard accordingly.

Effect of visual detection on detection time. We note that LGuard may be affected by delays in the CCTV equipment as part of the detection time. As video recordings need to be fetched, several variables can affect this retrieval time. Although current LGuard testing had negligible delays in transferring video recordings (40 MB average), other factors must be mentioned. For instance, the speed of processing of the recording device (NVR), the size of the recording, and the network speed may affect the overall retrieval time. Logically, if network devices are slow and large uncompressed video formats are used, the retrieval time will increase and affect LGuard detection time.

Performance of the Interaction Detector. The Interaction Detector that uses CCTV captures to detect replay attacks may have two possible issues when a room is populated. In one case, a person standing near an interface may register as if the interface is being touched at the time of a replay attack. In another case, a keypad may be obscured by a group of people, which may cause false results. Although these cases may present a problem, the risks are negligible for the following reasons. First, in a secure room, the remote attacker is unlikely to know when someone is standing in the vicinity of an interface. Second, interfaces mounted on walls are generally not at the same height as an individual's hand, making false touch detection more unlikely.

LGuard defense coverage. LGuard was designed as an addition for existing E-IoT systems to defend against communication bus based attacks. As such, LGuard does not defend against attacks that are executed through any other method (e.g., protocols, software, malware), unless it leverages the communication bus in some manner. In terms of communication bus attacks, LGuard aims to defend against the proposed attacks and attacks that rely on the methods proposed by LightningStrike. New attacks that rely on entirely new methods may not be detected by LGuard. However, LGuard is a modular defense framework, designed to be expandable to address future threats. Thus, LGuard allows for new algorithms to be implemented to defend against future attacks and newly discovered threats. For instance, if software injections are discovered for certain systems, adding a layer to detect possible communication bus based injections can be added as additional LGuard modules.

9 RELATED WORK

Smart device security. Attacks against smart devices has been an ongoing topic of research in recent years. 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 [1–4, 7, 29, 34, 47]. Further, research in alternative threat vectors such as USB and HDMI shows how an attacker can easily compromise devices using insecure protocols [22, 23, 42]. 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) [17]. Research on proprietary smart system protocols and threats has been mostly reserved to reverse engineering of protocols or encryption such as Somfy's RTS [40, 41]. 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 [50]. Prior research on E-IoT lighting control systems by the U.S. Department of Energy has highlighted some security risks that come from lighting control systems [39]. In the topic of E-IoT, Rondon et al. [43] covered driver-based attacks against E-IoT systems, compromising E-IoT through a software threat vector.

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. [8] highlighted MITM attacks. In terms of the studies aiming to protect industrial networks, the works of Dudak et al. [24] and Wilson [52] 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 [27]. 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. [51]. Further, solutions were proposed by researchers to detect attacks targeting serial-based communication buses. To name a few, Eigner et al. [25] proposed a machine learning based defense approach using K-nearest neighbors toward detecting MITM attacks against industrial control networks (i.e., Modbus). Similarly, Lan et al. [30] proposed a method of classifying S7comm traffic to detect data tampering caused by MITM attacks. The **Controller Area Network (CAN)** bus used in in-vehicle networks employs serial communication [49]. 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. [54]. In the work of Buttigieg et al. [6], the researchers investigated security issues and executed MITM attacks against a CAN network. Morgner et al. [38] 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.

How our work differs from prior works. Prior works highlight threats against off-the-shelf IoT systems through well-known attack vectors (e.g., TCP/IP, WiFi, Zigbee, Z-Wave), whereas LightningStrike is the first in the literature that uncovers the insecurities of E-IoT by focusing solely on proprietary protocols used in E-IoT. In this way, we shed light upon security of proprietary E-IoT communication through unconventional attack vectors. 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 LightningStrike-style threats. Furthermore, LGuard functions without modification or overhead to the E-IoT system, targeting each threat individually with high precision and accuracy.

10 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 LightningStrike, a series of attacks that leverage insecure E-IoT communication practices and vulnerabilities to an attacker's advantage. Specifically, with LightningStrike attacks, 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. In addition, we demonstrated that E-IoT is susceptible to DoS, eavesdropping, impersonation, and replay attacks due to insecure communication practices. As a traditional defense mechanism cannot mitigate LightningStrike threats due to the distinct characteristics of E-IoT systems, we introduced LGuard as a novel defense system against LightningStrike 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 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.

REFERENCES

[1] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security evaluation of home-based IoT deployments. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP'19).* 1362–1380. https://doi.org/10.1109/SP.2019.00013

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

[3] Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2019. A system-level behavioral detection framework for compromised CPS devices: Smart-grid case. *ACM Transactions on Cyber-Physical Systems* 4, 2 (Nov. 2019), Article 16, 28 pages.

[4] Leonardo Babun, Z. Berkay Celik, Patrick McDaniel, and A. Selcuk Uluagac. 2019. Real-time analysis of privacy-(un)aware IoT applications. arXiv:1911.10461 [cs.CR].

[5] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* 120 (2000), 122–125.

[6] Robert Buttigieg, Mario Farrugia, and Clyde Meli. 2017. Security issues in controller area networks in automobiles. In *Proceedings of the 2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA'17).* 93–98. https://doi.org/10.1109/STA.2017.8314877

[7] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A. Selcuk Uluagac. 2018. Sensitive information tracking in commodity IoT. In *Proceedings of the 27th USENIX Security Symposium.* 1687–1704.

[8] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. 2016. A survey of man in the middle attacks. *IEEE Communications Surveys Tutorials* 18, 3 (2016), 2027–2051. https://doi.org/10.1109/COMST.2016.2548426

[9] Control4. 2013. Configurable Decora Wired Keypad Installation Guide. Retrieved September 22, 2022 from https://www.control4.com/docs/product/wired-keypad/installation-guide/english/revision/B/wired-keypad-installation-guide-rev-b.pdf.

[10] Control4. 2021. Control4 Camera Solutions. Retrieved September 22, 2022 from https://www.control4.com/solutions/products/cameras/.

[11] Control4. 2021. Control4 Driver Search—Camera. Retrieved September 22, 2022 from https://drivers.control4.com/solr/drivers/browse?&fq=primaryProxy%3A%22camera%22.

[12] Control4. 2010. Getting Started with Composer Pro. Retrieved September 22, 2022 from https://www.yumpu.com/en/document/view/35663397/composer-pro-getting-started.

[13] Control4. 2010. Composer Pro Software Release Update Instructions. Retrieved September 22, 2022 from http://www.davidsonfamily.ca/wpcontent/uploads/asgarosforum/27894/TechDoc00033_RevB_ComposerUpdateQuickInstructions_MultipleControllers.pdf.

[14] Crestron. (n.d). Crestron Technical Institute. Retrieved September 22, 2022 from https://support.crestron.com/app/answers/detail/a_id/1000253/~/crestron-technical-institute---latest-courseofferings.

[15] Crestron. 2006. Crestron Isys Touchpanel Operation Guide. Retrieved September 22, 2022 from https://www.crestron.com/getmedia/e8833a2a-1d68-40ef-84af-0789e6de9f1d/mg_tpmc-4xg-b_1.

[16] Crestron. 2020. Wiring and Connectors. Retrieved September 22, 2022 from https://docs.crestron.com/en-us/9292/Content/Topics/Cresnet-Wiring.htm.

[17] Crestron. 2020. Security at Crestron. Retrieved September 22, 2022 from https://www.crestron.com/Security/Security-at-Crestron.

[18] Crestron. 2021. Crestron Application Market—Integrated Pan/Tilt. Retrieved September 22, 2022 from https://applicationmarket.crestron.com/integrated-pan-tilt-1/.

[19] Crestron. 2021. Works with Crestron—Cameras. Retrieved September 22, 2022 from https://docs.crestron.com/en-us/8525/Content/CP4R/Appendix/Works-With-Crestron-Home/Cameras.htm.

[20] Crestron. 2022. DIN Rail Cresnet®Distribution Hub. Retrieved September 22, 2022 from https://www.crestron.com/getmedia/f085dca1-aeab-4cff-b3c4-22a2989869a8/ss_din-hub_1.

[21] David Kravets. 2011. Wi-Fi–Hacking Neighbor from Hell Sentenced to 18 Years. Retrieved September 22, 2022 from https://www.wired.com/2011/07/hacking-neighbor-from-hell/.

[22] Kyle Denney, Enes Erdin, Leonardo Babun, and A. Selcuk Uluagac. 2019. Dynamically detecting USB attacks in hardware: Poster. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. 328–329.

[23] Kyle Denney, Enes Erdin, Leonardo Babun, Michael Vai, and Selcuk Uluagac. 2019. USB-Watch: A dynamic hardware-assisted USB threat detection framework. In *Proceedings of the International Conference on Security and Privacy in Communication Systems*. 126–146.

[24] J. Dudak, G. Gaspar, S. Sedivy, P. Fabo, L. Pepucha, and P. Tanuska. 2019. Serial communication protocol with enhanced properties—Securing communication layer for smart sensors applications. *IEEE Sensors Journal* 19, 1 (2019), 378–390.

[25] Oliver Eigner, Philipp Kreimel, and Paul Tavolato. 2016. Detection of man-in-the-middle attacks on industrial control networks. In *Proceedings of the 2016 International Conference on Software Security and Assurance (ICSSA'16)*.

[26] Josh Hendrickson. 2018. ZigBee vs. Z-Wave: Choosing between Two Big Smarthome Standards. Retrieved September 22, 2022 from https://www.howtogeek.com/394567/zigbee-vs.-z-wave-choosing-between-two-big-smarthome-standards/.

[27] IEEE. 2020. IEEE standard for secure SCADA communications protocol (SSCP). *IEEE Std 1711.2-2019* (2020), 1–37.

[28] IoTBusinessNews. 2018. The Number of Smart Homes in Europe and North America Reached 45 Million in 2017. Retrieved September 22, 2022 from https://iotbusinessnews.com/2018/09/24/20413-the-number-of-smart-homes-in-europe-and-north-america-reached-45-million-in-2017/.

[29] C. Kaygusuz, L. Babun, H. Aksu, and A. S. Uluagac. 2018. Detection of compromised smart grid devices with machine learning and convolution techniques. In *Proceedings of the 2018 IEEE International Conference on Communications (ICC'18)*. 1–6.

[30] Haiyan Lan, Xiaodong Zhu, Jianguo Sun, and Sizhao Li. 2020. Traffic data classification to detect man-in-the-middle attacks in industrial control system. In *Proceedings of the 2019 6th International Conference on Dependable Systems and Their Applications (DSA'20)*. 430–434. https://doi.org/10.1109/DSA.2019.00067

[31] Frank Liang. 2020. Evaluating the Performance of Machine Learning Models. Retrieved September 22, 2022 from https://towardsdatascience.com/classifying-model-outcomes-true-false-positives-negatives-177c1e702810.

[32] LiquidSpace. 2021. LiquidSpace: Rent Flexible Office Space. Retrieved September 22, 2022 from https://liquidspace.com/.

[33] LiteTouch. 2006. LiteTouch Lighting Control Systems Installation and Troubleshooting Manual. Retrieved September 22, 2022 from http://sav-documentation.s3.amazonaws.com/InternalDocumentation/LiteTouchandSavantLighting/TroubleshootingManual.pdf.

[34] Juan Lopez, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2017. A survey on function and system call hooking approaches. *Journal of Hardware and Systems Security* 1, 2 (2017), 114–136.

[35] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, et al. 2019. MediaPipe: A framework for building perception pipelines. arXiv:1906.08172 [cs.DC].

[36] Lutron. 2020. Lutron Integration Protocol. Retrieved September 22, 2022 from http://www.lutron.com/TechnicalDocumentLibrary/040249.pdf.

[37] Mark N. Vena. 2018. How Crestron Paved the Way for the Smart Home, and More. Retrieved September 22, 2022 from https://www.forbes.com/sites/moorinsights/2018/08/23/how-crestron-paved-the-way-for-the-smart-home-and-more/#397001f141f8.

[38] Philipp Morgner, Stefan Pfennig, Dennis Salzner, and Zinaida Benenson. 2018. Malicious IoT implants: Tampering with serial communication over the Internet. In *Research in Attacks, Intrusions, and Defenses*, Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis (Eds.). Springer International Publishing, Cham, Switzerland, 535–555.

[39] U.S. Department of Energy. 2018. Cyber Security for Lighting Systems. Retrieved September 22, 2022 from https://www.energy.gov/sites/prod/files/2018/06/f52/cyber_security_lighting.pdf.

[40] Pushstack. 2016. Control4 Driver Decryption. Retrieved September 22, 2022 from https://pushstack.wordpress.com/2016/03/06/control4-driver-decryption/.

[41] Pushstack. 2016. Somfy Smoove Origin RTS Protocol. Retrieved September 22, 2022 from https://pushstack.wordpress.com/somfy-rts-protocol/.

[42] Luis Puche Rondon, Leonardo Babun, Kemal Akkaya, and A. Selcuk Uluagac. 2019. HDMI-walk: Attacking HDMI distribution networks via consumer electronic control protocol. In *Proceedings of the 35th Annual Computer Security Applications Conference*.

[43] Luis Puche Rondon, Leonardo Babun, Ahmet Aris, Kemal Akkaya, and A. Selcuk Uluagac. 2020. PoisonIvy: (In)secure practices of enterprise IoT systems in smart buildings. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys'20)*. ACM, New York, NY, 130–139. https://doi.org/10.1145/3408308.3427606

[44] Luis Puche Rondon, Leonardo Babun, Ahmet Aris, Kemal Akkaya, and A. Selcuk Uluagac. 2021. 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)*. ACM, New York, NY, 106–116. https://doi.org/10.1145/3448300.3467830

[45] Luis Puche Rondon, Leonardo Babun, Ahmet Aris, Kemal Akkaya, and A. Selcuk Uluagac. 2021. Survey on Enterprise Internet-of-Things systems (E-IoT): A security perspective. arXiv:2102.10695 [cs.CR].

[46] Savant. 2014. Savant Smart Lightning Deployment Guide. Retrieved September 22, 2022 from https://sav-documentation.s3.amazonaws.com/Product%20Deployment%20Guides/SmartLightingControl_DeploymentGuide.pdf.

[47] Amit Kumar Sikder, Leonardo Babun, Hidayet Aksu, and A. Selcuk Uluagac. 2019. Aegis: A context-aware security framework for smart home systems. In *Proceedings of the 35th Annual Computer Security Applications Conference.*

[48] Somfy. 2020. Control RTS Solutions with Most Automation Systems. Retrieved September 22, 2022 from https://www.somfysystems.com/en-us/products/1810872/universal-rts-interface.

[49] H. M. Song, H. R. Kim, and H. K. Kim. 2016. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In *Proceedings of the 2016 International Conference on Information Networking (ICOIN'16)*. 63–68.

[50] Stephen Genusa. 2015. Crestron Cresnet Monitor. Retrieved September 22, 2022 from https://pushstack.wordpress.com/somfy-rts-protocol/.

[51] A. Volkova, M. Niedermeier, R. Basmadjian, and H. de Meer. 2019. Security challenges in control network protocols: A survey. *IEEE Communications Surveys Tutorials* 21, 1 (2019), 619–639.

[52] Paul Lawrence Wilson. 2018. *ModSec: A Secure Modbus Protocol*. Master's Thesis. Georgia Institute of Technology.

[53] Ryan Winfield and Mark Gerrior. 2006. Avoiding Interference in the 2.4-GHz ISM Band. Retrieved September 22, 2022 from https://www.eetimes.com/avoiding-interference-in-the-2-4-ghz-ism-band/.

[54] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom. 2019. Survey of automotive controller area network intrusion detection systems. *IEEE Design & Test* 36, 6 (2019), 48–55.

Received 5 November 2021; revised 3 July 2022; accepted 22 July 2022