LightningStrike: (In)secure Practices of E-IoT Systems in the Wild

Luis Puche Rondon, Leonardo Babun, Ahmet Aris, Kemal Akkaya, and A. Selcuk Uluagac
Florida International University - {lpuch002, lbabu002, aaris, kakkaya, suluagac}@fiu.edu

ABSTRACT

The widespread adoption of specialty smart ecosystems has changed the everyday lives of users. As a part of smart ecosystems, Enterprise Internet of Things (E-IoT) allows users to integrate and control more complex installations in comparison to off-the-shelf IoT systems. With E-IoT, users have a complete control of audio, video, scheduled events, lightning fixtures, shades, door access, and relays via available user interfaces. As such, these systems see widespread use in government or smart private offices, schools, smart buildings, professional conference rooms, hotels, smart homes, yachts, and similar professional settings. However, even with their widespread use, the security of many E-IoT systems has not been researched in the literature. Further, many E-IoT systems utilize proprietary communication protocols that rely mostly on security through obscurity, which has perhaps led many users to mistakenly assume that these systems are secure. To address this open research problem and determine if E-IoT systems are vulnerable, we focus on one of the core E-IoT components, E-IoT communication buses. Communication buses are used by E-IoT proprietary protocols to connect multiple E-IoT devices (e.g., keypads and touchscreens) and trigger pre-configured events upon user actions. In this study, we introduce LightningStrike, the implementation of four proof-of-concept attacks that demonstrate several weaknesses in E-IoT proprietary communication protocols through communication buses. With LightningStrike, we show that it is feasible for an attacker to compromise E-IoT systems using E-IoT communication buses. We demonstrate that popular E-IoT proprietary communication protocols are susceptible to Denial-of-Service, eavesdropping, impersonation, and replay attacks. As E-IoT systems control physical access, safety components, and emergency equipment, an attacker with a low level of knowledge and effort can easily exploit E-IoT vulnerabilities to impact the security and safety of users, smart systems, and smart buildings worldwide.

CCS CONCEPTS

• **Security and privacy → Denial-of-service attacks**; • **Networks → Cyber-physical networks**.

KEYWORDS

Enterprise Internet-of-Things, E-IoT Security, Attacks, Cresnet

WiSec '21, June 28-July 2, 2021, Abu Dhabi, United Arab Emirates

ACM ISBN 978-1-4503-8349-3/21/06…$15.00
https://doi.org/10.1145/3448300.3467830

ACM Reference Format:
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 *14th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21), June 28-July 2, 2021, Abu Dhabi, United Arab Emirates.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3448300.3467830

1 INTRODUCTION

The current, rapid adoption of specialty smart systems has changed the lives of millions of users worldwide [23]. As part of these smart ecosystems, Enterprise Internet of Things (E-IoT) are smart systems designed to allow users to integrate and control very complex installations at a higher cost than off-the-shelf IoT systems. As such, E-IoT systems are usually vendor-installed, and grant users a robust, reliable, and accepted solution for smart installations. 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, smart homes, offices, yachts, colleges).

While the security of numerous off-the-shelf IoT smart systems is well-understood due to prior research and mainstream knowledge, very little research exists on E-IoT and their proprietary technologies. With many of these E-IoT systems deployed in high-profile locations (e.g., government and enterprise offices, colleges, conference rooms, hospitals), evaluating possible threats for these E-IoT smart systems should be of utmost importance. However, many E-IoT systems use proprietary communication protocols that rely solely on security through obscurity. The motivation of this paper is to shed light upon the security of E-IoT systems and uncover possible vulnerabilities of E-IoT proprietary communication protocols that can affect millions of E-IoT deployments. To address this open research problem and determine if E-IoT systems are susceptible to attacks, we focus on one of the core E-IoT components, E-IoT communications buses. E-IoT communication buses are used by E-IoT proprietary communication protocols to carry out fundamental internal communication functions such as interactions between user interfaces and the central controller. As such, communication buses are used to trigger programmed E-IoT events on integrated devices. In this work, we take a look at Crestron's Cresnet, a proprietary communication bus protocol used by one the major E-IoT system vendors. Crestron is a great example of a globally accepted E-IoT system with billions in sales, deployments in over 90% of Fortune 500 companies, and thousands of independent installers [30]. In order to demonstrate how feasible it is for an attacker to compromise an E-IoT system through insecure communication protocols, we propose LightningStrike, a set of practical proof-of-concept attacks created to leverage insecure communication buses, namely Cresnet, to an attacker's advantage. Specifically,

with LightningStrike we demonstrate that it is feasible for an attacker with limited resources to easily compromise an E-IoT system through an E-IoT communication bus threat vector.

In this work, we execute LightningStrike attacks in a realistic Crestron E-IoT environment. Further, we show that with LightningStrike, an attacker can use proprietary communications vulnerabilities to take arbitrary control of E-IoT operations. Therefore, we demonstrate that an attacker 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 devices. As E-IoT buses used by proprietary communication protocols are outside the scope of any traditional networks (e.g., WiFi, TCP/IP), LightningStrike provides attackers with an effective, practical, and covert mechanism to compromise E-IoT systems. We believe that LightningStrike may act as the initiator of research on security of E-IoT communication, the consequences of which can broadly impact the security of millions of current and future E-IoT deployments. We hope that our work will raise the awareness in the community, and encourage further research in the field.

Contributions: The contributions of this work are as follows:

- We introduce LightningStrike, a set of proof-of-concept attacks against E-IoT proprietary communication protocols.
- We demonstrate that communication buses used by 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 real-life E-IoT Crestron testbed and leverage communication buses to cause undesired behavior on behalf of an attacker.
- We articulate the implications LightningStrike attacks on E-IoT security and any integrated component.

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 architecture for the proof-of-concept LightningStrike attacks. Section 5 describes the testbed configuration, software, hardware, and attack implementations used to evaluate LightningStrike attacks. Section 6 highlights the attack process and implications of LightningStrike-based attacks. In Section 7, we discuss findings, contributions, possible defense mechanisms, and challenges against LightningStrike attacks. In Section 8 related work is highlighted. Finally, we conclude the paper in Section 9.

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

Enterprise IoT (E-IoT) systems are closed-source smart systems that follow unique design and deployment practices, separating them from off-the-shelf IoT systems [37]. 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

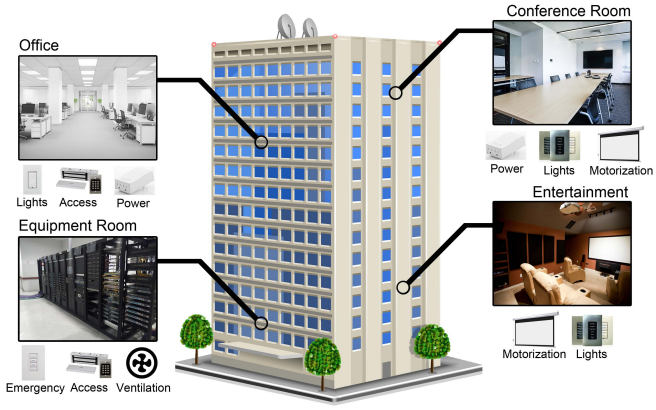

Figure 1: Use-cases of E-IoT systems.

to users' specifications [9, 11]. 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, pressing button 6 turns off all the lights). This highly-programmable component is configured by the integrator during deployment or maintenance stages of an E-IoT 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 equipment, rooms, and multiple *connection endpoints* where devices such as keypads, touchscreens, and other user interfaces connect to the communication bus. Such interfaces can be accessible by general users while other interfaces are only accessible in restricted locations. As Figure 2 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.

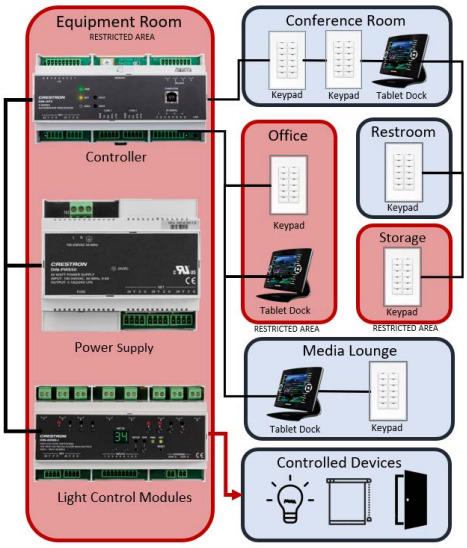

Figure 2: An example E-IoT system with wired bus communication and two daisy-chain paths. Restricted areas highlighted in red, common areas in blue.

2.2 E-IoT Protocols

E-IoT supports a variety of protocols, while some supported protocols are widely-known and well-documented (e.g., ZigBee, Z-wave, and TCP/IP), other protocols used by E-IoT systems are entirely proprietary in nature. As E-IoT system vendors need protocols designed for their specific purposes, they may modify existing known protocols or design entirely new protocols. Specifically, user interfaces such as keypads and touchscreens use wired and wireless protocols for communication purposes. For instance, in 2013, before Zigbee's rise in popularity, Control4, a vendor that offers E-IoT solutions, used a version called Embernet as a wireless solution [10]. Lutron, a vendor that focuses on E-IoT lighting control systems, implemented a proprietary wireless communication known as Somfy's Radio-Technology Somfy (RTS) [29, 41]. *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 [27]. For similar purposes, Control4 employs a proprietary communication protocol [8]. Savant uses communication buses and proprietary protocols for interfaces [39]. 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 [12]. *The technical specifications of these highlighted protocols are not publicly available, and thus their security, if any, is largely unknown.* Since the communication is simple, reliable, and allows daisy-chain wiring between interfaces, this communication is very prevalent in E-IoT. In comparison to protocols such as Z-wave and ZigBee, wired communication buses are preferred for E-IoT devices for three reasons. First, communication buses often provide power to the connected devices through the same communication line [13]. Second, communication buses are seen as more reliable than wireless protocols over long distances where mesh networking has range limitations (60 feet) [21]. Third, wired E-IoT communication is not as susceptible to interference as wireless communication, creating a more reliable system [46]. 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 through an available communication bus in a smart building.

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

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

- *Third-party contractors.* Repair and maintenance services often require external contractors (e.g., electricians, plumbers, painters, external I.T.) with unsupervised temporary access to facilities such as smart buildings. In some scenarios, such as re-painting walls or repairing damages, contractors must remove fixtures and mounted E-IoT devices (e.g., keypads,

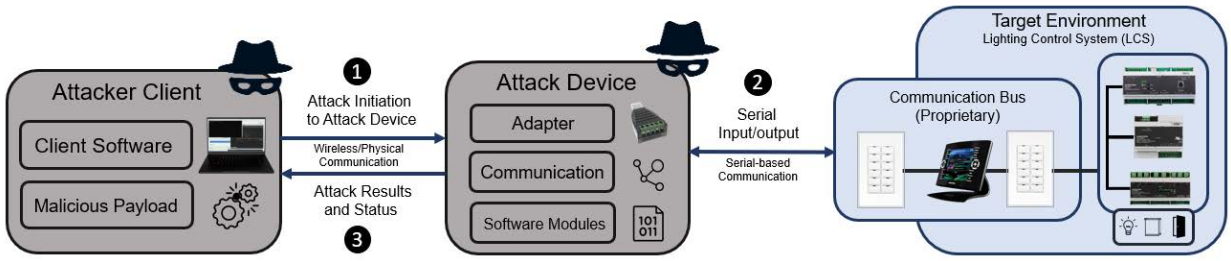

Figure 3: General end-to-end implementation for LightningStrike-based attacks. Attack-related components are highlighted in gray, E-IoT components are in blue.

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 [26]. Conference rooms need E-IoT interfaces for the users to control projects, lightning, screens, and A/V required for a presentation. If the communication bus wiring is shared between the rented room and other areas of the facilities, it would be trivial for attackers to insert their devices in the line and later perform attacks.
- *Neighbors.* Locations with E-IoT systems may have neighboring offices or other locations for rent. Cases of attackers using their proximity to their target have occurred in the past [16], 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 sub-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/she has restricted access and limited permissions to a facility.

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

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

3.3 Threat Model

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

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

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

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

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

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 paper.

4 LIGHTNINGSTRIKE ARCHITECTURE

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

4.1 LightningStrike 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

Table 1: Hardware & software used in LightningStrike attacks implementation and evaluation.

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

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

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

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

- *Monitoring Module.* The monitoring module passively eavesdrops on active bus communication without transmitting messages. As bus-connected devices broadcast messages in bus architectures, the monitoring module is able to capture all messages transmitted.
- *Injection Module.* The injection module injects arbitrary messages from Mallory into the communication bus.
- *Flooding Module.* The flooding module is designed to cause DoS conditions in the communication bus, by maliciously flooding the communication bus.
- *Re-addressing Module.* The re-addressing module has the ability to re-address and modify the configuration in devices that use the communication bus. As such, it may allow devices to impersonate others by setting identification numbers and other configurations.
- *Filtering Module.* The filtering module filters communication received by the monitoring module to allow Mallory only to view the information requested and make incoming data easier to interpret.

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

5 LIGHTNINGSTRIKE ATTACKS IMPLEMENTATION

In this section, we describe the LightningStrike attacks implementation. We use LightningStrike to attack the Cresnet E-IoT proprietary communication protocol. As noted earlier in Section 3 and Section 4, Mallory compromises an E-IoT system through the insertion of a malicious attacker device into the communication bus. To ensure that LightningStrike attacks are demonstrated and evaluated realistically, we created an attack suite and a realistic E-IoT testbed. The Attacker client was implemented as the Acer

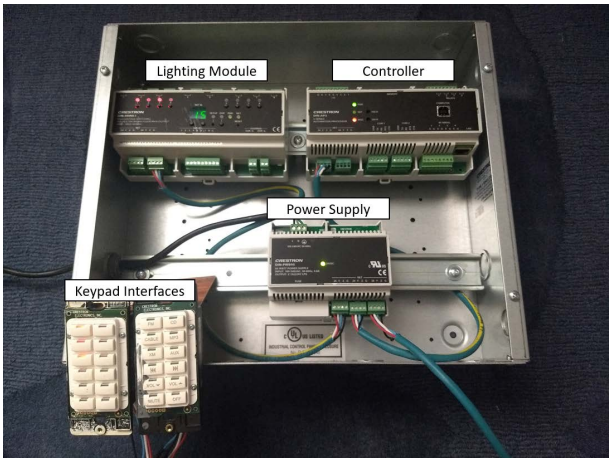

Figure 4: Testbed used to implement LightningStrike attacks, including a controller, power supply, smart modules, and keypad interfaces.

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.

LightningStrike Testbed. As most E-IoT systems are proprietary systems that define their own specifications for protocols using communication buses, we had to select a proprietary protocol which is a representative of E-IoT systems for this work. To ensure that we 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 dollars in annual revenue [30]. Further, 90% of Fortune 500 companies have some form of Crestron solution in their facilities. In addition to being one of the largest players in smart installations, Crestron highlights a commitment to security [14]. As such, Crestron is expected to be at or above industry security standards for E-IoT systems. Specifically, we chose Crestron's Cresnet for analysis, a proprietary serial-based communication protocol used in E-IoT integration and other smart use-cases.

We created an E-IoT testbed using common Crestron devices highlighted in Figure 4. Crestron proprietary software was used for testbed deployment and configuration. To configure the Crestron system, we utilized D3 Pro and Toolbox, the software used by integrators to deploy Crestron lighting systems. We utilized a Crestron DIN-AP3 processor as the primary logic unit of the testbed and a Crestron DIN-8SW8-I as the lighting module (8 controlled loads) for the testbed. Crestron devices utilize a proprietary communication protocol, namely Cresnet. To create a single Cresnet daisy-chain, we implemented two C2N-DB12W keypads using Cresnet wiring. The Cresnet-based 12-button keypads and similar interfaces are common in public areas such as restrooms and are used as user interfaces to control lights, door access, relays, and any programmed functions. Finally, the entire system is powered by a Cresnet-based DIN-PWS50 power supply.

The Cresnet Bus. With the need to interface multiple devices together, proprietary communication protocols using communication buses are commonly used in E-IoT. Cresnet is a widely-used proprietary protocol of the popular Crestron E-IoT smart systems. Based on RS-485 communication, Cresnet is used to power, control, update, identify, and configure Cresnet-enabled devices (e.g., controllers, lighting modules, keypads, touchscreens). As a proprietary protocol, not much information is available on the technical specification of the Cresnet protocol. Cresnet comes in a 4-wire configuration, with 24 (red, power), Y (white, TX/RX+), Z (blue, TX/RX-), and G (black, ground) [13]. While there is no public specification given, we found the information to support that Cresnet communication operates at a 38400 baud rate and half-duplex operation [12].

Table 2: Assigned Cresnet ID Table.

Device	Cresnet ID
Crestron DIN-AP3 (Controller)	Self-Addressed
Crestron DIN-PWS50 (Power Supply)	Self-Addressed
Crestron C2N-DB12W (Keypad)	03
Crestron DIN-8SW8-I (Lighting Module)	15
Crestron C2N-DB12W (Keypad)	23

Cresnet ID. Devices in a Cresnet network are addressed with a network ID. The IDs are hexanumerical and range from 00 to FF. Cresnet IDs identify individual Cresnet devices in the network. The integrator assigns unique Cresnet IDs to each Cresnet device during the configuration.

Proprietary Protocol Analysis. As Cresnet is largely undocumented, we faced a number of challenges in the analysis stage of this protocol. With no available documentation on the protocol, we referred to some existing open-source software and legacy manuals to find the proper baud rate and specification of the Cresnet protocol. With this basic information, we could begin the analysis of the protocol. While the data structure of Cresnet communication is not publicly accessible; our initial analysis shows that Cresnet devices place their Cresnet ID on packet headers. This was observed when connecting new devices to the communication bus and observing initiation packets after new devices were connected. Further, the controller periodically queries the devices in the Cresnet bus and expects responses. If these responses are not received, the Cresnet controller is observed to query until it receives a response. This behavior was observed while manually disconnecting keypads and monitoring active communication. In addition to this periodic query-response traffic in the Cresnet bus, Crestron devices also create E-IoT application traffic.

Software Module Implementation. In order to execute the LightningStrike attacks, we developed a number of software modules. The attacks were implemented using open-source tools available online. We used Java as the base language with RX-TX libraries for serial-based communication [38]. Java was chosen as it is compatible with many platforms. Modules requiring communication rate specifications were configured with a 38400 baud rate, eight data bits, one stop bit, and no parity. The software modules are implemented into the attack device, as highlighted in our end-to-end implementation (Section 4).

1) Monitoring Module. The monitoring module was implemented in Java and the RX-TX library for RS-485-based communication.

As such, the module executes as a loop that listens to Cresnet communication with the serial settings specified.

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

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

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

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

6 LIGHTNINGSTRIKE ATTACKS EVALUATION

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

Attack 1: Flooding Denial-of-Service. This attack was created to demonstrate that Threat 1 (DoS, Section 3) is viable through LightningStrike by overwhelming the communication bus with messages.

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

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

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

Attack 2: Malicious Eavesdropping. Attack 2 demonstrates that Threat 2 (Malicious Eavesdropping, Section 3) is viable on the Crestron testbed. This attack used the monitoring and filtering modules to observe and infer information from Cresnet packets.

```
Length: 1 |15
Length: 7 |00020003000200
Length: 1 |23
Length: 9 |000200150002000300
Length: 1 |02
Length: 1 |00
Length: 1 |23
Length: 11 |00020015000200030000200
Length: 1 |23
Length: 11 |00020015000200030000200
Length: 1 |23
Length: 3 |000200
Length: 1 |15
Length: 7 |00020003000200
```

Figure 5: A snapshot of the captured periodic query-response traffic during Attack 2 (Malicious Eavesdropping) where the Cresnet IDs are highlighted.

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

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

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

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

Evaluation. The attack monitored and gathered information from the Cresnet bus successfully due to Cresnet's lack of encryption. We highlight a snapshot of the captured Cresnet communication in Figure 5, where communication is in the order it was received. First, monitoring the Cresnet bus easily allowed us to gather Cresnet IDs. As such, an attacker can easily know how many Cresnet devices are connected to the communication bus through their unique IDs. Our eavesdropping revealed at least four unique devices: the keypads, the lighting module, and the controller. We could observe spikes of activity when keypad buttons were pressed and other actions were executed on the bus. An attacker can use this information to infer building occupancy by identifying keypads in specified locations and listening for events originating from the associated Cresnet ID. As the attack was performed through passive monitoring, no alarms, or any unexpected behavior occurred in the Cresnet bus or any of the dependent devices (e.g., keypads, controller).

Attack 3: Impersonation-based Denial-of-Service Attack. This attack was designed to demonstrate another form of DoS attack using Threat 1 and Threat 3 (DoS and Impersonation, Section 3) through LightningStrike. As such, we accomplished a DoS condition by creating an ID conflict between devices. The attack takes

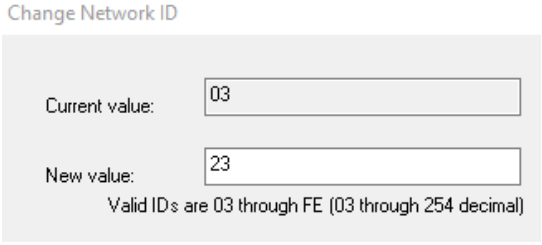

Figure 6: Cresnet ID change window in D3 Pro Crestron software during Attack 3 (Impersonation-based Denial-of-Service).

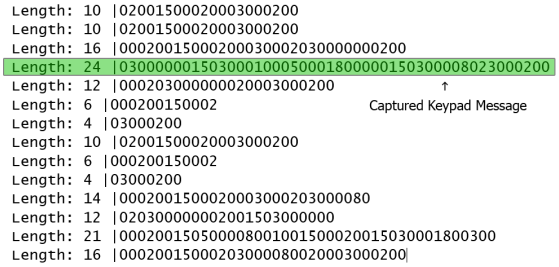

Figure 7: A snapshot of the messages captured from Cresnet bus during Attack 4 (Replay Attack).

advantage of Cresnet's identification phase when a new device is added to the system. Our research showed that new devices broadcast several packets upon connection and Cresnet relies solely on the Cresnet ID to identify the individual devices, button presses, and other actions.

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

Step 2 - Modification. Using a spare controller and software, we altered the Cresnet ID of the keypad to "23" without any form of validation as shown in Figure 6. We changed the ID of a keypad with the ID of an existing keypad to purposely cause an ID conflict in the Cresnet bus. Additionally, this was also accomplished by physically replacing the original keypad with an identical keypad with a pre-assigned ID.

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

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

Attack 4: Replay Attack. This attack was created to demonstrate that Threat 4 (Communication Replay, Section 3) is possible on the Cresnet bus. Further, we highlight the implications of replay attacks on E-IoT systems.

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

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

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

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

Evaluation. We evaluated this attack on the success of replaying button presses from a Cresnet keypad. As such, the attack was entirely successful in replaying button presses on the Cresnet bus. The initial monitoring phase for messages was successful as the packets were captured during physical button presses. Step 2 of the attack can be observed in Figure 7. In this step we could easily observe spikes in activity on button presses. With the messages captured, our attempts to replay a button press (button 1) were successful, with the testbed reacting by turning on a light associated with button 1 as if the button had been pressed. Further, we could use the same captured packet to turn on the light again, demonstrating there is no replay protection in the Cresnet bus. The implications of this attack show that an attacker could capture a button press to unlock an Crestron-controlled door with the same captured code, or ultimately assume control of integrated devices by replaying the associated button presses.

Summary As LightningStrike attacks were developed and tested, we demonstrated that insecure proprietary protocols can have many negative impacts on the security of E-IoT. From our attacks, we concluded that without any form of security beyond obscurity, a knowledgeable attacker can easily compromise E-IoT to their benefit. All of the proposed attacks were implemented successfully, the implications of which clearly show the potential of LightningStrike attacks. In Attack 1 and Attack 3, we demonstrated that multiple Cresnet-based interfaces can be disabled by an attacker. This is a viable form of preventing access to any user-controlled systems through a DoS attack. As E-IoT manages light control, gate access, and other essential components, an authorized user can be prevented from operating a connected system through the attack proposed in our examples. Further, programmed events such as panic buttons will not execute while a DoS attack is active on the affected interfaces. In Attack 2 we showed that an attacker can capture communication between multiple devices from a single point of connection, the implications of which can be abused by an attacker. With Attack 3, creating a Cresnet ID conflict would be no

issue for attackers as all the source and destination addresses are broadcasted over the communication bus. Further, if an attacker has an idea of which Cresnet IDs belong in which locations, they can infer which room is occupied. As button presses and messages are broadcasted no matter where the keypad is located, an attacker can infer information on unauthorized locations and query equipment unreachable via traditional means (e.g., TCP/IP, WiFi, Bluetooth). As Attack 4 (replay attack) was successful, we show that an attacker can severely compromise the security of Crestron systems. For instance, if an attacker manages to re-address a keypad using a replay attack, it is possible to reprogram a number of devices. Understanding the Cresnet protocol through further reverse engineering may allow future attacks through generating Cresnet packets without the need for capture and replay.

7 DISCUSSION

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

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

Contributions. Our research provided several contributions towards the security of E-IoT systems. As mentioned in Section 5, we emphasized that the Crestron brand is currently one of the most popular E-IoT systems available worldwide with 1.5 billion dollars in annual revenue [30]. With this work, we demonstrated that unsecured proprietary communication protocols used in E-IoT systems can lead to downtime, a breach of privacy, and a breach of physical security. Additionally, this work aimed to raise awareness on lesser-known but widely-used protocols in E-IoT systems such as Cresnet. As such, we highlighted vulnerabilities found during our research, and how the lack of common security mechanisms in proprietary-protocols can easily lead to critical vulnerabilities in E-IoT systems. We used the LightningStrike proof-of-concept attacks to expose these vulnerabilities, and presented several practical attacks that can contribute towards more complex, larger attacks. LightningStrike introduced a new threat vector, so that future iterations of E-IoT systems and their proprietary protocols can be designed with secure practices in mind for communication buses. Since systems such as Crestron have been deployed for decades, we expect that this work can motivate further research on E-IoT attacks, security mechanisms, proprietary protocol security, highly-deployed threat vectors, and other popular E-IoT systems that have not received any form of security scrutiny.

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

8 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, 6, 24, 28, 40]. Further, research in alternative threat vectors such as USB and HDMI shows how an attacker can easily compromise devices using insecure protocols [17, 18, 35]. 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) [15]. Research on proprietary smart system protocols and threats has been mostly reserved to reverse engineering of protocols or encryption such as Somfy RTS [33, 34]. 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 [43]. Prior research on E-IoT lighting control systems (LCSs) by the U.S. Department of Energy has highlighted some security risks that come from LCSs [32]. In the topic of E-IoT, Puche et al. [36] 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. [7] highlighted MITM attacks. In terms of the studies aiming to protect industrial networks, the works of Dudak et al. [19] and Wilson [45] 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 [22]. 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. [44]. Further, solutions were proposed by researchers to detect attacks targeting serial-based communication buses. To name a few, Eigner et al. [20] proposed an ML-based defense approach using K-nearest neighbors towards detecting MITM attacks against industrial control networks (i.e., Modbus). Similarly, Lan et al. [25] proposed a method of classifying S7comm traffic to detect data tampering caused by MITM attacks. Controller Area Network (CAN) bus used in in-vehicle networks employs serial communication [42]. 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. [47]. In the work of Buttigieg et al. [5], the researchers investigated security issues and executed MITM attacks against a CAN network. Morgner et al. [31] 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.

LightningStrike differs from prior works as follows: While prior works highlight threats against off-the-shelf IoT systems through well-known attack vectors (e.g., TCP/IP, WiFi, Zigbee, Z-Wave), our work is the first in the literature that uncovers the insecurities of E-IoT by focusing solely on proprietary protocols used in E-IoT. By this way, we shed light upon security of proprietary E-IoT communication through unconventional attack vectors. In order to analyze the security of such systems and demonstrate realistic attacks, we created a testbed utilizing real E-IoT devices of one of the most popular E-IoT systems, namely Crestron. We demonstrated four attacks, specifically two distinct types of DoS, eavesdropping, and replay attacks. The scope of our attacks relies on proprietary communication, and does not rely on any software-based vulnerabilities, overflows, traditional network connectivity, or fuzzing.

9 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 proof-of-concept attacks that leverage insecure E-IoT communication practices and vulnerabilities to an attacker's advantage. Specifically, with LightningStrike 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. To implement and test LightningStrike attacks in a realistic manner, we created an E-IoT testbed using real E-IoT devices. With LightningStrike, we analyzed the proprietary Cresnet communication protocol and implemented a series of attacks on the testbed. Specifically, we demonstrated that E-IoT is susceptible to Denial-of-Service, eavesdropping, impersonation, and replay attacks due insecure communication practices. LightningStrike attacks clearly demonstrated that millions E-IoT deployments in various professional environments are not secure. In addition, these threats could be very critical, as E-IoT systems may control emergency equipment and physical access, the consequences of which may be as costly and dangerous as well-known notorious attacks. As future work, we aim to develop a security mechanism to protect against LightningStrike-style attacks while considering the E-IoT proprietary protocols' use cases, designs, and limitations.

10 ACKNOWLEDGMENTS

This work is partially supported by the U.S. National Science Foundation (Awards: NSF-CAREER-CNS-1453647, NSF-1663051). The views are those of the authors only. We also thank the anonymous reviewers and the shepherd for their valuable feedback and time on this paper.

REFERENCES

[1] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *2019 IEEE Symposium on Security and Privacy (SP)*. 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 *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 Trans. Cyber-Phys. Syst.* 4, 2, Article Article 16 (nov 2019), 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:cs.CR/1911.10461

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

[6] 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 *27th USENIX Security Symposium*. 1687–1704.

[7] 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

[8] Control4. 2013. Configurable Decora Wired Keypad Installation Guide. https://www.control4.com/docs/product/wired-keypad/installation-guide/english/revision/B/wired-keypad-installation-guide-rev-b.pdf. Online: Accessed 20-March-2020.

[9] Control4. Jun, 2010. Getting Started with Composer Pro. https://www.control4.com/files/dealers/200-00168-ComposerProGettingStarted.pdf. Online: Accessed 10-December-2019.

[10] Control4. Mar, 2010. Composer Pro Software Release Update Instructions. https://www.control4.com/files/dealers/TechDoc00005_ComposerUpdate Instructions_1 8 2.pdf. Online: Accessed 10-April-2020.

[11] Crestron. [n.d.]. Crestron Technical Institute. https://www.crestron.com/training. Online: Accessed 10-December-2019.

[12] Crestron. 2006. Crestron Isys Touchpanel Operation Guide. https://www.crestron.com/getmedia/818150e6-4976-4982-8c05-ba7d2b33957b/mg_tps-12b_12w_15b_15w_17b_17w_1 Online: Accessed 10-April-2020.

[13] Crestron. 2020. Cresnet Wiring - Cable Types & Lengths for Connecting Devices. https://support.crestron.com/app/answers/detail/a_id/1629/~/cresnet-wiring—cable-types. Online: Accessed 10-April-2020.

[14] Crestron. 2020. Crestron's Commitment to Security. https://www.crestron.com/About/commitment-to-security. Online: Accessed 20-October-2020.

[15] Crestron. 2020. Security at Crestron. https://www.crestron.com/Security/Security-at-Crestron. Online: Accessed 18-May-2020.

[16] David Kravets. 2011. Wi-Fi–Hacking Neighbor From Hell Sentenced to 18 Years. https://www.wired.com/2011/07/hacking-neighbor-from-hell/. Online: Accessed 15-May-2021.

[17] 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.

[18] Kyle Denney, Enes Erdin, Leonardo Babun, Michael Vai, and Selcuk Uluagac. 2019. USB-Watch: A Dynamic Hardware-Assisted USB Threat Detection Framework. In *International Conference on Security and Privacy in Communication Systems*. Springer, 126–146.

[19] 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.

[20] Oliver Eigner, Philipp Kreimel, and Paul Tavolato. 2016. Detection of Man-in-the-Middle Attacks on Industrial Control Networks. In *2016 International Conference on Software Security and Assurance (ICSSA)*.

[21] Josh Hendrickson. 2018. ZigBee vs. Z-Wave: Choosing Between Two Big Smarthome Standards. https://www.howtogeek.com/394567/zigbee-vs.-z-wave-choosing-between-two-big-smarthome-standards/. Online: Accessed 20-October-2020.

[22] IEEE. 2020. IEEE Standard for Secure SCADA Communications Protocol (SSCP). *IEEE Std 1711.2-2019* (2020), 1–37.

[23] IoTBusinessNews. Sept, 2018. The number of smart homes in Europe and North America reached 45 million in 2017. Online: Accessed 10-December-2019.

[24] C. Kaygusuz, L. Babun, H. Aksu, and A. S. Uluagac. 2018. Detection of Compromised Smart Grid Devices with Machine Learning and Convolution Techniques. In *2018 IEEE International Conference on Communications (ICC)*. 1–6.

[25] 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 *2019 6th International Conference on Dependable Systems and Their Applications (DSA)*. 430–434. https://doi.org/10.1109/DSA.2019.00067

[26] LiquidSpace. 2021. LiquidSpace: Rent flexible office space. https://liquidspace.com/. Online: Accessed 15-May-2021.

[27] LiteTouch. 2006. LiteTouch Lighting Control Systems Installation and Troubleshooting Manual. http://sav-documentation.s3.amazonaws.com/Internal Documentation/LiteTouch and Savant Lighting/Troubleshooting Manual.pdf. Online: Accessed 20-March-2020.

[28] 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. Accessed: 11-17-2018.

[29] Lutron. 2020. Lutron Integration Protocol. http://www.lutron.com/TechnicalDocument Library/040249.pdf. Online: Accessed 10-May-2020.

[30] Mark N. Vena. 2018. How Crestron Paved The Way For The Smart Home, And More. https://www.forbes.com/sites/moorinsights/2018/08/23/how-crestron-paved-the-way-for-the-smart-home-and-more/#397001f141f8. Online: Accessed 18-May-2020.

[31] 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, 535–555.

[32] United States Department of Energy. 2018. Cyber Security for Lighting Systems. https://www.energy.gov/sites/prod/files/2018/06/f52/cyber_security_ lighting.pdf.

[33] Pushstack. 2016. Control4 driver decryption. https://pushstack.wordpress.com/2016/03/06/control4-driver-decryption/. Online: Accessed 18-May-2020.

[34] Pushstack. 2016. Somfy Smoove Origin RTS Protocol. https://pushstack.wordpress.com/somfy-rts-protocol/. Online: Accessed 18-May-2020.

[35] 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*. 10.

[36] 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 *(BuildSys '20)*. Association for Computing Machinery, New York, NY, USA, 130–139. https://doi.org/10.1145/3408308.3427606

[37] 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:cs.CR/2102.10695

[38] RXTX. 2020. RXTX - A Java Cross Platform Wrapper Library For The Serial Port. https://github.com/rxtx/rxtx. Online: Accessed 1-March-2020.

[39] Savant. 2014. Savant Smart Lightning Deployment Guide. https://support.savant.com/pro. Online: Accessed 15-August-2020.

[40] 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*. 14.

[41] Somfy. 2020. Control RTS Solutions with Most Automation Systems. https://www.somfysystems.com/en-us/products/1810872/universal-rts-interface. Online: Accessed 1-March-2020.

[42] 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 *2016 International Conference on Information Networking (ICOIN)*. 63–68.

[43] Stephen Genusa. 2015. Crestron Cresnet Monitor. https://pushstack.wordpress.com/somfy-rts-protocol/. Online: Accessed 18-May-2020.

[44] 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.

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

[46] Ryan Winfield and Mark Gerrior. 2006. Avoiding Interference in the 2.4-GHz ISM Band. https://www.eetimes.com/avoiding-interference-in-the-2-4-ghz-ism-band/. Online: Accessed 20-October-2020.

[47] C. Young, J. Zambreno, H. Olufowobi, and G. Bloom. 2019. Survey of Automotive Controller Area Network Intrusion Detection Systems. *IEEE Design Test* (2019).