Teaching IoT Security using CTF Problems

Submitted in partial fulfillment of the requirements for

the degree of

Master of Science

in

Information Security

Hugrun Hannesdottir

B.S., Computer Science, Reykjavik University

Carnegie Mellon University
Pittsburgh, PA

December, 2021

Acknowledgements

I want to express my gratitude to Maverick Woo for his endless support as an incredible thesis advisor. Thank you for our weekly meetings that helped me develop problems and understand concepts in-depth. I sincerely appreciate you inspiring me towards being a more qualified security professional. The outcome of this thesis would not have been possible without your input.

I would also like to acknowledge Hanan Hibshi for being a phenomenal teacher and academic advisor throughout my journey at Carnegie Mellon University. Thank you for being my thesis reader and for always having my best interest in mind.

I want to recognise CyLab and the picoCTF project for giving me the information and setup needed. The problems created by picoCTF were inspirational and motivated my work.

This research is supported in part by the Information Networking Institute at Carnegie Mellon University by providing me with financial aid to help me pursue this education. Without this support, I would not have been able to attend this University and gain my education.

I want to acknowledge my partner Sindri Pall Andrason for their love and support. Thank you for listening and encouraging me throughout the semester.

Last but not least, I want to thank my family for their endless support and inspiration throughout the years. Thank you for always lighting the candle during the walk down an unknown-dark path.

Abstract

In recent years, the *Internet of Things* (IoT) has become increasingly prevalent in modern societies. By integrating various sensors and logic controllers, IoT devices are utilized for automation in many different contexts, such as smart homes, agriculture, and manufacturing. Unfortunately, these devices are prone to have security vulnerabilities. This could be because these devices were designed with no bad actors in mind, or because their implementations contain security flaws. In order to provide more students with an opportunity to learn about potential IoT security risks, it would be highly desirable to provide hands-on exercises that are economical, scalable and safe to use. This thesis aims to develop new learning resources to support this goal, and it makes two contributions: (i) the design of a fictional factory that uses various IoT devices and protocols in an automated production line, showcasing an emerging paradigm known as *Industrial IoT* (IIoT); and (ii) the design and implementation of a set of Jeopardy-style CTF challenges that uses the IoT devices in this factory in a coherent storyline to test the skills and knowledge of students in various security concepts related to these devices.

Table of Contents

List of Tables

List of Figures

1

Introduction

In recent years, the *Internet of Things* (IoT) has become a popular technology concept in modern societies. At the functional level, IoT refers to devices that perform some form of automation using various controllers, sensors, or actuators. The most distinctive feature of IoT devices is that these devices can communicate and coordinate with each other *without* requiring a human operator. IoT devices currently use various protocols; some common examples include HTTP, Modbus, MQTT, Z-wave, and ZigBee. The actual protocol used by an IoT device depends on its purpose, as these protocols can vary in aspects such as implementation cost or throughput. In the consumer space, protocols such as ZigBee and Z-Wave are popular for *"Smart Homes"*, which is part of the IoT that is used in homes to simplify everyday lives. In the industrial space, protocols such as Modbus and MQTT are often used to automate manufacturing, where production devices are connected to the Internet and form the so-called *"Industrial IoT"* (IIoT).

Unfortunately, IoT devices are prone to contain security risks. One common reason is that the software on these devices may have implementation flaws similar to the software on any other modern computer. Another reason is that some of

these devices were initially designed to operate in a threat model that assumes no bad actors. This resulted in insecure devices that may lack validation of inputs, encryption or authentication of messages. Over the years, IoT attacks have become increasingly more severe, with many high-profile attacks including *Mirai* and *Stuxnet*. This suggests that if society is to continue to increase its reliance on the IoT, then prospective tech workers and leaders must learn how to build secure IoT devices.

1.1 Motivation

The work in this thesis aims to answer the following question: *"How can we help more IT Security students learn IoT Security given the typical education background of these students?"*. Our answer to this question is twofold.

First, we believe that there is currently a practical difficulty for students to learn IoT Security. Using ourselves as an example, the author of this thesis has spent three years studying for a Bachelor of Science in Computer Science at Reykjavik University and has been studying for a Master's degree in Information Security at Carnegie Mellon University (CMU) since Fall 2020. Upon reflection, it was noticeable that there had been limited opportunities in the curriculums for learning about cybersecurity in domains outside of traditional IT, such as IoT, Industrial Control Systems (ICS), and Operational Technology (OT). We believe a primary reason for this is because standard degree programs need to cover many topics to teach IT Security well, and thus little time can be spent on non-IT domains. This suggests that perhaps the most viable option for current students to learn non-IT Security is through *self-learning*.

Second, we believe that such self-learning can be facilitated by providing students with a set of well-designed Capture The Flag (CTF) problems in a non-IT domain. At its core, CTF is simply an assessment method. Compared to other assessment methods such as written exams, CTF excels in its quality to test students on their

possession of knowledge and their ability to apply the knowledge in a hands-on setting. However, when a set of related CTF problems are presented together, such as in a CTF competition, CTF problems can become an educational tool. This is best explained by the concept of *gamification* [1]. In essence, a set of related CTF problems presents students with a game to make progress by solving the CTF problems. The reward process in such games drives students to want to learn more when they do not know how to solve a problem, thus making them become self-learners. Indeed, many students already leverage CTF competitions to learn IT Security, as witnessed by the many CTF competitions every year and many participants in these competitions. We, therefore, believe that we will be successful in helping more students learn IoT Security by producing our own set of CTF problems in the domain of IoT.

1.2 Contributions

This thesis focuses on designing and implementing a set of CTF problems in the chosen domain of Industrial IoT, a sub-domain of IoT. The thesis has two main objectives: (i) design all the CTF problems using a single fictional factory setup, thus making them more coherent and engaging in context as a set, both of which facilitate learning, and (ii) attempt to cover as many fundamental cybersecurity concepts in IIoT as possible given the thesis timeframe, thus maximizing the educational value in the problems.

In addition to presenting the design of a fictional factory, this thesis also makes the following contributions. First, it shows that it is possible to create a set of CTF problems with elaborate problem descriptions that, when taken together, present a coherent storyline and cover various cybersecurity and domain-specific concepts in IIoT. Six CTF problems were designed and implemented on top of the picoCTF platform. For each problem, we will discuss the problem description and hints pre-

sented to the learners and the rationale behind the problem design, which includes the desired learning objectives.

Second, this thesis demonstrates how one may be able to avoid the need for actual hardware when designing CTF problems for the IIoT domain. Instead of using physical devices, our CTF problems were created using virtual devices, that is, programs that emulate their physical counterparts. This software-only approach means that our effort could be scaled to serve many learners economically and that the learners will remain safe because they would not be in contact with any physical danger. The latter is especially important because some of the CTF problems, by their very nature, involve having the learner attack the safety mechanisms in the targeted industrial devices.

Third, this thesis demonstrates how one may design more inclusive CTF problems. As a woman in the cybersecurity industry, the author has observed first-hand a gender imbalance in the industry, and many people inaccurately assume hackers are males only. In addition, there is also an increasing number of people who identify themselves as a different gender other than males and females, i.e., non-binary. Therefore, a deliberate choice was made that none of the problems created in this thesis would use any gender-specific pronouns that allow specifying the gender of the characters.

2

Background

2.1 Internet of Things

The term IoT was first coined by Kevin Ashton in 1999 while working at Procter and Gamble [2]. Ashton used this term to describe how the company could use *radio-frequency identification* (RFID) to track inventory [2]. The meaning of the term has changed since then; modern IoT devices are not necessarily RFID based but rather devices that are embedded with sensors and controllers to gather information about their environment [3].

Currently, these devices are used in various places such as homes and industries. IoT devices used in *Smart Homes* can make daily tasks easier. Devices in such households can range from lightbulbs to fridges [3]. IoT used for factories is known as *Industrial IoT* (IIoT). IIoT devices can help control automation and monitoring at scale [4]. Arguably, the usage of IIoT in the industry started even before the concept of IoT was invented. Modern IIoT dates back to as early as 1968, when Dick Morley invented the concept of *Programmable Logic Controller* (PLC) [5]. PLC are controllers connected to actuators and sensors to manage and control information in

a single or multiple manufacturing processes. The device uses different channels; the input channel for sensors and the output for actuators.

2.1.1 Protocols

IoT devices communicate with each other using network protocols that describe a predefined setup of how information should be transferred between hosts. As part of the research for this thesis, several protocols commonly used in IoT were studied, but the following section will contain a short review of them.

HyperText Transfer Protocol (HTTP) is an application layer protocol that was designed for a client-server model [6]. This protocol transfers hypermedia information between hosts, and therefore, it is widely used for Web Applications. Version 2 of HTTP was introduced in 2015 [7]. This version changed the structure of transferred data, such as by compressing the header [7]. However, this version is slow in adoption. The design of version 3 of HTTP is currently in progress and will contain changes to use QUIC instead of TCP/IP as the transport [8]. HTTP is used in various IoT devices to allow an online interface to interact with the device. For instance, *Smart Cameras* are cameras that can communicate over IP, and such devices usually expose a webserver to allow administrators to configure the device and view the camera footage over the Internet. Regarding security considerations, an administrator should consider how to handle sensitive information. If a sensitive application is to be implemented over HTTP, then *Transport Layer Security* (TLS) should be applied [6].

Constrained Application Protocol (CoAP) is a web application protocol that is specialised for constrained devices, known as nodes, and networks [9]. CoAP is recommended for low-power or lossy networks [9]. CoAP is a request-response protocol that allows for low overhead, asynchronous messages, caching capabilities, plus optional reliability requests [9]. The protocol was designed based on the REST architecture [9]. In regard to security concerns, CoAP should be bound to *Datagram*

Object	Access	Size (bits)
Coil	Read-Write	1
Discrete Input	Read-Only	1
Input Register	Read-Only	16
Holding Register	Read-Write	16

Table 2.1: Modbus Objects and Their Properties.

Transport Layer Security (DTLS) to provide a secure communication method [9].

Modbus is an application layer communication protocol that was created in 1979 [10]. Since its initial release, its popularity has increased in industrial settings as it is open-source and contains few restrictions. The protocol is commonly used for IIoT to communicate with PLCs. Unlike HTTP, Modbus is a request-reply protocol that uses function codes to offer services to users. Services include reading and writing to registers and how to gain diagnostic information. Modbus uses four different types of objects to help write to or read from channels. These objects are known as *Coils*, *Discrete Inputs*, *Input Registers*, and *Holding Registers*. Table 2.1 shows the properties of the different objects in Modbus [10]. Modbus also uses exception responses to indicate when a request has failed. Examples of such errors include illegal functions, access to addresses or values, device business, and gateway failures. Similar to HTTP, the Modbus documentation recommends using TLS to provide security over the transport layer [11]. Additionally, the Modbus documentation recommends mutual client and server authentication and authorisation via certificates.

Siemens Step 7 is a software used for both programming and configuring SIMATIC PLCs [5]. Its release in 1994 helped Siemens become a leader in automation technology. *S7 Communication* (S7comm) is a protocol used for communication between Siemens S7 and PLCs. This protocol is similar to Modbus in many ways, such as that it uses function and code commands.

Process Field Net (PROFINET) is commonly used in industrial companies [12].

PROFINET is easy to use, flexible and scalable. It uses a provider-consumer model, but data gets exchanged between controllers and IO devices. The controllers are known as *IO controllers*, and PLCs are an example of such controllers. PROFINET also uses IO Supervisors such as a computer or *Human Machine Interfaces* (HMI), which will be discussed later. The developers of PROFINET divided potential security threats into three classes [13]. The first is *Robustness*, which argues that it is crucial to seal the system from outside access. The second is *Integrity and Authentication*, which explains that it is vital to secure the data via the communication network. The third discusses the importance of data *Confidentiality*.

The *Message Queue Telemetry Transport* (MQTT) is a publish-subscribe protocol that was developed explicitly for IoT purposes [14]. The protocol is popular for many reasons, such as being lightweight, providing a reliable message delivery, bi-directional communication, and scalability. Industries use MQTT in many applications, such as manufacturing and transportation. In addition, Smart Homes tend to use this protocol as well, such as for monitoring patients [15]. MQTT architecture can contain multiple clients and servers. MQTT *broker* is the server that routes messages to and from clients based on a UTF-8 string called *topic* [14]. MQTT clients either subscribe or publish to a topic. Figure 2.1 shows an example MQTT deployment. The MQTT documentation states that the protocol should offer options such

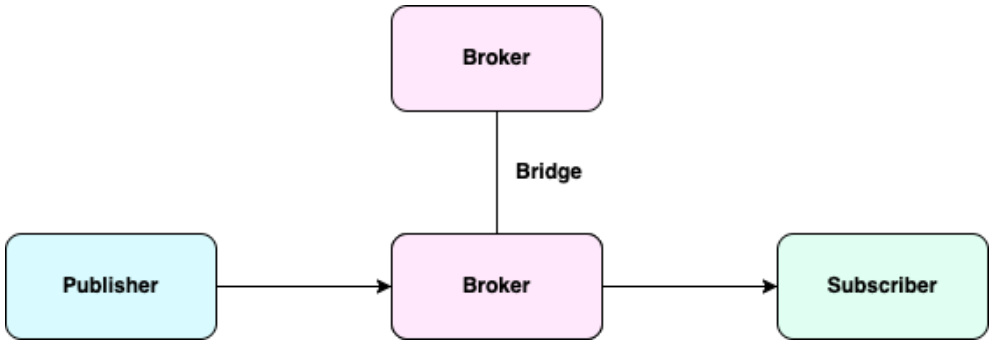

Figure 2.1: An Example MQTT Deployment.

as authentication and authorisation. Additionally, MQTT should use TLS for secure communication [14].

The *IEEE 802.15.4* standard defines low-rate wireless networks [16]. *ZigBee* is an example of such a protocol. It provides low-cost and low-power communication [16, 17]. ZigBee allows different types of network topologies, such as star and mesh. The architecture stack divides services into blocks called *layers*, and each layer supports the service layer above. ZigBee builds on the *Physical Layer* (PHY) and the *Media Access Control* (MAC) layer by creating the *ZigBee Network* (NWK) Layer. A *ZigBee coordinator* controls the NWK and devices on the network. Devices on the network are known as a *ZigBee Device Objects* (ZDO). Each ZDO is responsible for managing its device, including initialising the *Application Support Sublayer* (APS), NWK, and Security Service Providers (SSP). ZigBee uses *Device Profiles* to identify devices in the topology, and each profile expects a client-server topology. A client is the one that initiates the request to a server using a device identification message. ZigBee uses keys to provide a secure communication channel between devices. The ZigBee coordinator chooses the key that the network will use.

Z-Wave is another protocol that fits the IEEE 802.15.4 standard [18]. Z-Wave uses a master-slave architecture, as in one device controls the others. Device information needs to be stored in advance because the protocol uses a low-data link. Additionally, this protocol does not interfere with WiFi [19]. The protocol is commonly used for control and monitoring [19]. Similarly to ZigBee, Z-Wave uses a network key to secure communication messages [20].

2.1.2 Security Concerns

IoT devices can contain security vulnerabilities, such as if there is excessive trust between services or if the communication layer between hosts is not secured, leading to attacks such as Man-in-the-Middle and Denial of Service [21]. Due to this, numerous

IoT devices use *Transport Layer Security* (TLS) to provide a secure communication channel between hosts [22]. A handshake procedure is used between hosts to create a secure channel. During a handshake, keys are shared between the respective parties to allow for encryption on messages later in the same session.

OWASP is an organisation that promotes knowledge of known security vulnerabilities in the hope to improve future devices and networks. The top IoT vulnerabilities in 2018, according to OWASP, were the following [23].

1. *Weak passwords* can grant unauthorised users access to devices.
2. *Insecure network services* that can result in device services being accessible on the Internet.
3. *Insecure device interfaces* such as insecure web and mobile interfaces could lead to unauthorised access due to lack of authentication.
4. *Lack of updates* can result in known vulnerabilities being used to affect unpatched IoT devices.
5. *Outdated components* can result in a weak entry point that can lead to a total device compromise.
6. *Lack of privacy protection* can result in a breach of confidentiality.
7. *Unsafe transfer of data* can occur if the communication channel between devices is not encrypted.
8. *Lack of device management* can result in difficulties monitoring systems and detecting potential vulnerabilities.
9. *Default settings* should be updated to accommodate how the device is used.
10. *Inadequate physical hardening* can lead to more attackable surfaces.

This list of vulnerabilities will help guide the CTF problems designed in this thesis.

2.2 Industrial Control Systems

The term *Operational Technology* (OT) refers to systems that monitor and control the operations in an industrial setting [5]; in comparison, *Information Technology* (IT) are systems that transmit, store or process business data [5]. The term *Industrial Control Systems* (ICS) is used to describe control systems such as *SCADA* and *DCS* that can achieve objectives including automating manufacturing processes [24]. There are different kinds of manufacturing processes, categorized as either *Continuous Manufacturing Processes* or *Batch Manufacturing Processes*. Continuous Manufacturing Processes are continuously running, while Batch Manufacturing Process defines processes that have distinct steps [24].

The usage of IoT in industrial systems is slowly increasing because it allows for the possibility of automation of production lines [25]. Using a device with actuators and sensors, administrators can configure industrial controllers from miles away. Usually, a monitoring interface known as *Human Machine Interface* (HMI) is used for this purpose. In general, an HMI can allow both device configuration and monitoring [5].

Many factors need to be taken into considerations when designing an ICS, including the distribution of objects and devices, hierarchical structure, systems availability and safety requirements [5]. ICS systems are commonly partitioned into three different functions: view, monitor, and control. The *view function* sees the current state of the system; the *monitor function* verifies values match a predefined threshold; and the *control function* assists hardware such as actuators and valves to work as expected.

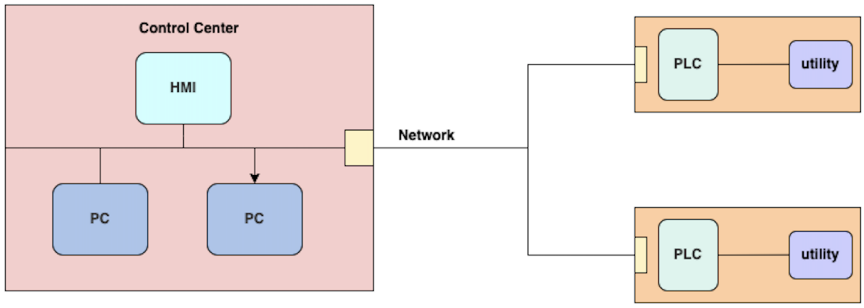

Figure 2.2: An Example SCADA Deployment.

2.2.1 Supervisory Control and Data Acquisition

Supervisory Control and Data Acquisition systems (SCADA) is an architecture consisting of controllers working together to monitor processes. Historically, SCADA was designed for geographically distributed systems [5]. SCADA uses *Remote Terminal Unit* (RTU) or *Programmable Logic Controllers* (PLC) to collect data. RTU is a computer that has network capabilities to communicate with a supervising controller [24]. Control servers, such as HMI, are located in a control center. Figure 2.2 shows an example of a SCADA deployment.

2.2.2 Distributed Control System

In many ways, *Distributed Control Systems* (DCS) are similar to SCADA. A DCS contains control servers such as HMI that collect information from RTU and PLC. However, unlike SCADA, DCS was created for systems in the same geographical locations, such as within a single factory. Figure 2.3 shows an example of a DCS deployment.

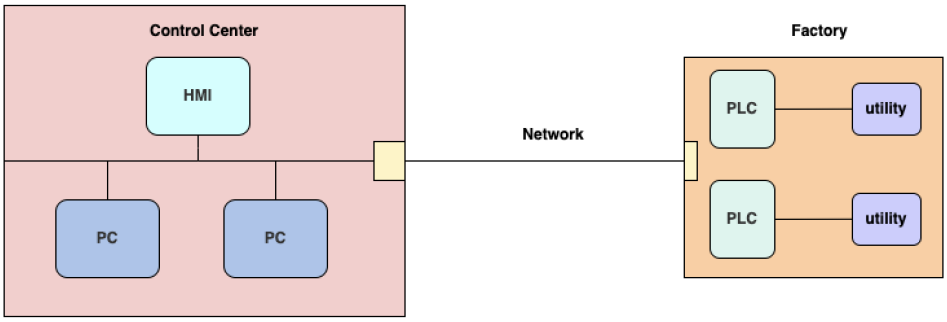

Figure 2.3: An Example DCS Deployment.

2.2.3 Architecture Models

The *Purdue Enterprise Reference Architecture* (PERA) is a modern enterprise architecture model [5]. This model was developed in the 1990s at Purdue University. It is often used to simplify ICS architecture design and description by partitioning a deployment into five levels.

- Level 0: The physical processes.
- Level 1: Devices that manipulate the physical processes.
- Level 2: Control systems such as an HMI.
- Level 3: Manufacturing systems.
- Level 4: Business logic.

In this thesis, we use a more refined model called the *Purdue* model. This model was adopted from the PERA model by the ISA-99 standards development committee. The Purdue model separates ICS components using a structure similar to the PERA model. The Purdue model contains three zones: Manufacturing, Demilitarized and Enterprise zone. Figure 2.4 shows the Purdue model structure in detail.

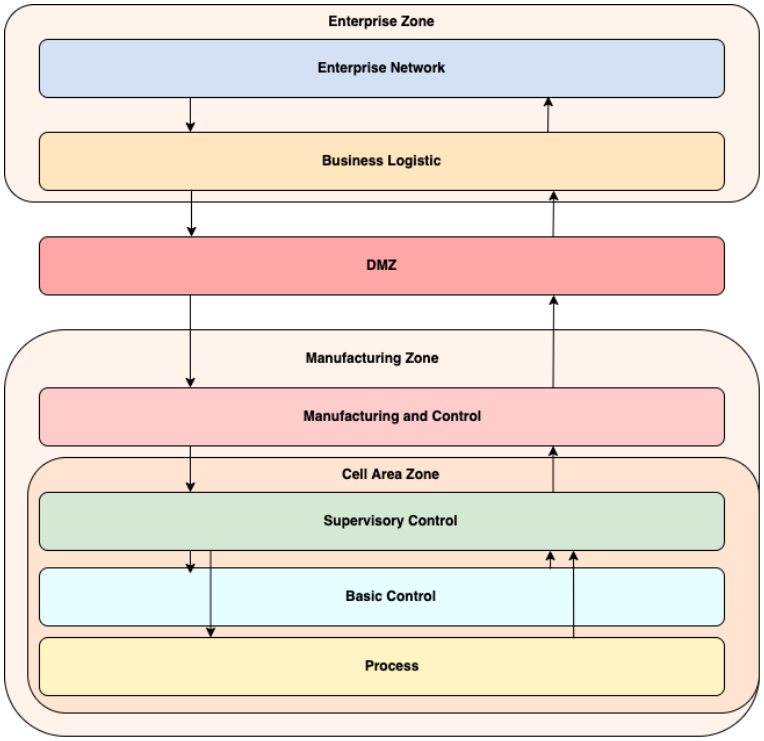

Figure 2.4: The Purdue Model of ICS Design.

In the Purdue model, the *Cell Area* zone contains the part of an ICS where the process occurs. This zone contains the following three levels:

- Level 0: The *Process* level contains the actual physical process that higher levels are controlling.
- Level 1: The *Basic Control* level contains systems such as motors, valves and sensors that can control physical processes.
- Level 2: The *Supervisory Control* level contains monitoring systems such as HMI.

The *Manufacturing* zone describes the part of an ICS about process management. This zone contains the levels in the Cell Area zone as well as an additional level:

- Level 3: The *Manufacturing and Control* level contains the necessary functions to help control the workflow of the process.

After comes the zone known as the *Industrial Demilitarized Zone* (DMZ). This zone sits between the Manufacturing zone and the Enterprise zone, and it is used for security measures to allow safe connections between the IT and the OT levels. Due to its nature, this zone often contains firewalls to protect private networks.

Finally, the model has an *Enterprise* zone, and it contains the following levels:

- Level 4: The *Business Logistics* level contains part of an ICS where administrators tasks are conducted.

- Level 5: The *Enterprise Network* level contains systems that use data produced by lower-level systems to generate reports about the current factory status.

2.2.4 Security Considerations

ICS systems often apply what is known as a *Defense in Depth* model [5, 24]. This is because no single solution can stop all attack vectors. *Physical security* of the systems is vital because no one should be able to gain access to a sensitive area. *Network security* is also necessary because even if users cannot access a restricted area, so long as they can gain access through the network, they could still get information about sensitive systems. Finally, *system security* is also essential, and all computers and applications in an ICS should be updated regularly to minimize the chance of being vulnerable to known cyberattacks.

2.3 Security Attacks

It would be desirable to understand attacks that have occurred in the past to understand the possible security threats that IoT devices and Industrial Control Systems face. In this section, a few recent high-profile attacks will be discussed.

2.3.1 Mirai

Denial of Service (DoS) attacks are attacks that aim to make a service unresponsive [26]. A distributed DoS attack refers to a DoS attack that uses high traffic to cause a service shut-down. In 2016, a DDoS IoT attack known as *Mirai* was unleashed, affecting multiple services, including Netflix, Twitter and Github [27]. This attack spread first by infecting devices such as cameras and routers. Then, the malware used brute force to figure out the credentials and the leaked source code of Mirai contained 62 username and password combinations. After gaining access to the device, the program informed a server, known as the botmaster, about the target before downloading a malware payload. The botmaster could then instruct its nodes to attack a chosen server in the future.

The Mirai attack showed multiple vulnerabilities IoT devices contained. The key mitigation approaches include changing the default credentials of a device at deployment time, closing unused ports and monitoring requests [28].

2.3.2 Stuxnet

In 2010, a new worm was discovered, and it aimed at vulnerable SCADA systems and was later named *Stuxnet* [29]. This attack was the first known attack aimed explicitly at Industrial Control Systems, but it is also an IoT attack [30]. This attack is believed to have been started with the use of a USB flash drive for the initial infection [29]. The attackers used stolen signed certificates to install a worm that can avoid detection [30]. Once the worm gained access to the operating system, it searched for specific slave variables that the Siemens S7 software would use and then sent the PLCs in the system malicious actions to perform [29]. On an infected system, the worm lives solely in memory, and thus no evidence of its existence can be found on the disks. Interestingly, the worm contained an un-install mechanism and had a strict worm scaling control mechanism [29].

This sophisticated attack showed that systems should apply a continuous update mechanism to improve systems security; in addition, removable storage should be allowed only with precaution and detection systems should be used [29].

2.3.3 WannaCry

The term "ransomware" refers to attacks that block users access to their data until the attackers receive payment [31]. The first ransomware is known as the PC Cyborg Trojan, and it was developed in 1989 [32]. Since then, there have been many others, including the high-profile *WannaCry* ransomware in 2017 [5]. The WannaCry ransomware resulted in the National Health Service employees being locked out of healthcare data [5, 32]. The vulnerability arose due to an older Microsoft vulnerability known as EternalBlue. This vulnerability allowed remote code execution with a crafted message to a Microsoft Server Message Block (SMB) server. The SMB file-sharing protocol allows applications to read and write data between different end-hosts. The WannaCry attack used this to pivot among computers and encrypted almost 300,000 computers, and demanded a bitcoin payment to decrypt the files. This attack showed that it is essential to apply security patches regularly and avoid using dated systems that have already reached their end-of-support because they will no longer receive patches.

2.3.4 Ukraine Power Grid Attack

In December 2015, the Ukrainian power companies were under the first known successful power grid ICS cyberattack [30]. The attackers ended up compromising and disrupting the electricity supply for around 230,000 people [33]. The attack began with spear-phishing emails that contained malware in an attachment. This malware is known as *BlackEnergy* and is said to have possibly given attackers access to legitimate credentials [30, 33]. As part of the attack, a *KillDisk* malware was exe-

cuted in the systems to erase selected files. This attack showed that it is essential to use trusted and updated hardware and software and know what is occurring in the network traffic.

2.3.5 Discussion

As can be seen, numerous IoT devices and ICS deployments have been compromised in the past. IoT devices have been compromised using default passwords and unused ports, such as in the Mirai attack. Past attacks on ICS systems occurred due to outdated software versions and a lack of detection. We believe that teaching students the importance of securing these systems is vital for the future prosperity of our societies.

2.4 Jeopardy-Style CTF

Capture the Flag (CTF) is a game-like education method to teach security. Like the outdoor game where the name CTF originally came from, the objective of a CTF is to have a user capture a digital flag. Usually, a *flag* is hidden by the creator on vulnerable software or devices. There are two major types of CTF competitions, namely Jeopardy-style and Attack-Defense [34]. *Attack-Defense* CTF are structured in a way that there are multiple competing teams. One team is attacking while the others are defending their service [34]. *Jeopardy-style CTF* is another form of CTF competition, where an individual or a team can play by themselves. Jeopardy-style CTF contains tasks categorised into different categories, including web, forensics and reverse engineering. The picoCTF platform created by CyLab at Carnegie Mellon University is an example of a CTF service that hosts such competitions [35]. Many CTF problems in the picoCTF service are intended to be more beginner-friendly. Each problem usually contains a description and hints to guide the player to find the hidden flag. Players that obtain these hidden flags get points in return.

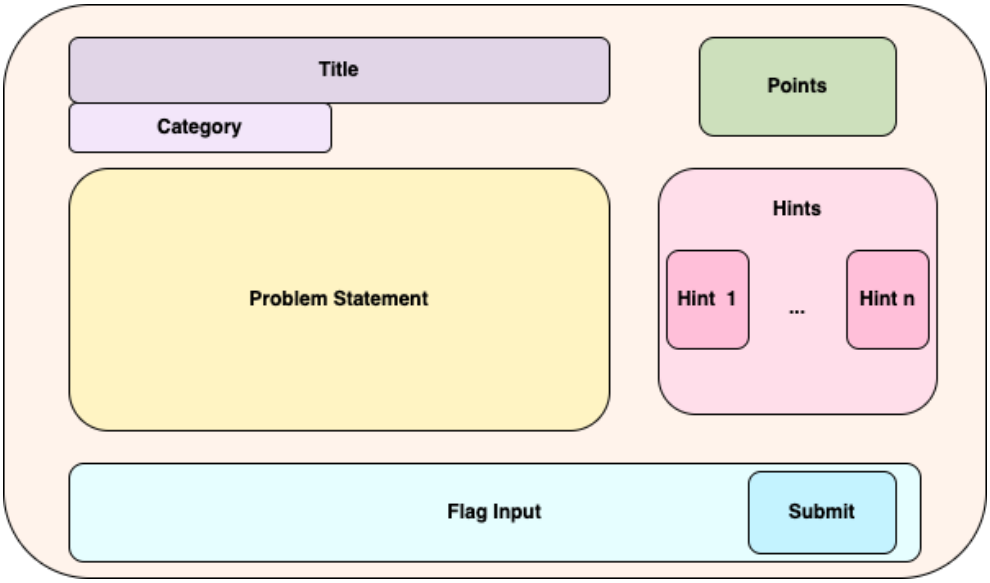

Figure 2.5: Components of a CTF Problem on the picoCTF Platform.

Our project was conducted in collaboration with picoCTF. In particular, the problem structure and the deployment method both follow other CTF problems found on the picoCTF platform. We hope that this will allow the CTF problems created in this thesis to be deployed one day in the future on the picoCTF platform to teach users about IoT vulnerabilities. Figure 2.5 illustrates the structure of a CTF problem on the picoCTF platform. Each problem is expected to contain the following items:

1. *Points* that can be given to users after submitting the correct flag for the problem.

2. *Title* that names the problem.

3. *Problem Statement* that presents the problem setting in the CTF question.

4. *Flag field* where the user can submit a flag for points.

5. *Hints* that the user can view to guide them along the way to solve the problem.

2.5 Related Work

CTF problems have been used before to teach students about different kinds of security weaknesses. For example, Carnegie Mellon University CyLab created the picoCTF platform for that purpose [35]. However, the CTF problems currently offered on the picoCTF platform focus on IT security, such as web, forensics, and reverse engineering. In contrast, this thesis aims to create CTF problems to teach IoT security, specifically in the area of IIoT.

IoT security-related CTF challenges have been created before. An example is the *IoT Village*, where users can access IoT devices in person and hack them to see the potential disasters that can occur [36]. However, unlike the IoT village, the problems this thesis aims to create are made without the need for actual hardware to increase our ability to scale to serve a large number of students simultaneously. Additionally, though IoT CTF problems have been created before, such as by *ph0wn*, this thesis aims to generate IoT CTF problems in an industrial setting with a focus on using a coherent storyline [37]. Similarly, though the *Collegiate Penetration Testing Competition* (CPTC) allows users to pen-test a fictional company, it does not guide the users throughout because of its competitive nature [38]. In this thesis, we aim to create CTF problems with sufficient hints to help users learn the knowledge and skills needed to solve the problems.

The book *Industrial Cybersecurity* comprehensively presents both the cyber attack and defense aspects of industrial systems [5]. Our research relied heavily on this book, and we highly recommend the reader to refer to it for more information on ICS security.

3

Fictional Factory Design

This thesis resulted in six CTF problems to teach common vulnerabilities in the IIoT setting. The problems revolve around the adventure of Pipi and Flash, who are arch-enemies that enjoy making life difficult for each other on a regular basis. For these problems, the story revolves around a cookie party that Flash is hosting, and Pipi wants to ruin it by destroying the order Flash made to the cookie factory Hug&Run.

The factory in Hug&Run accepts orders from an online web service. Once an order is received, the system will start creating the cookies, including mixing the batter, baking, packaging, and shipping. The company uses IoT devices to reduce manual costs and quicken the manufacturing speed of cookies. Its factory contains multiple PLCs, including those that add ingredients to the batter of the current order and controls the cookie oven. The company uses the Modbus protocol to communicate with many of the PLCs. However, the Modbus protocol used in the company often lacks the security features required to provide a secure communication channel. Unfortunately, such insecure protocol is used in the factory to communicate with the PLC that adds ingredients to the batter.

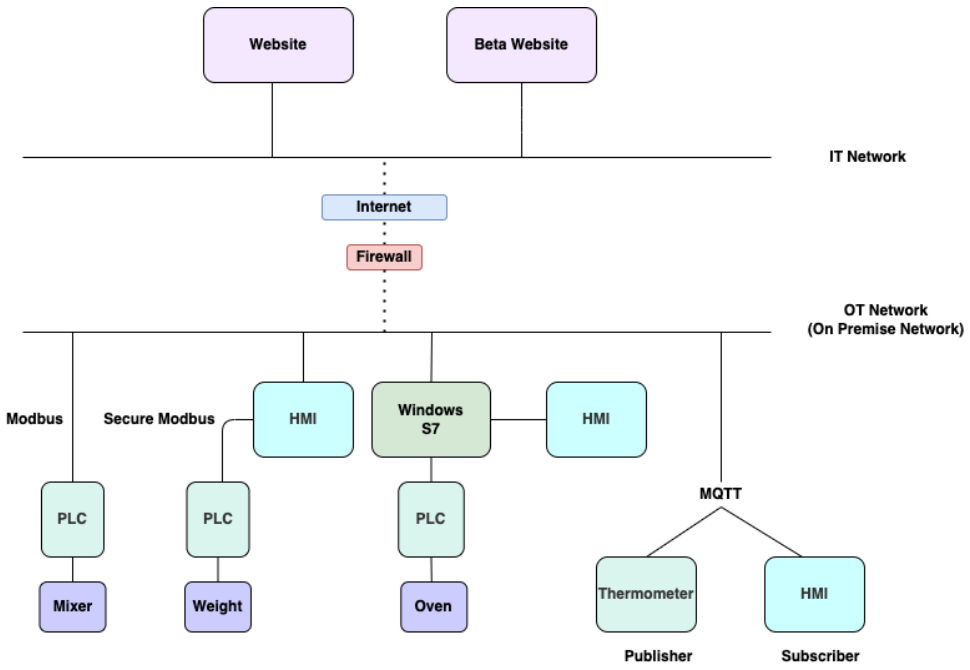

Figure 3.1: Network Diagram of the Fictional Factory.

After adding ingredients together, the weight of the batter is verified if it is as expected for quality assurance. This mechanism uses Modbus too, but due to upgrades, the communication between the sensor that reads from the scale and the PLC is secured with TLS. However, this scale has a web-based control panel that allows an operator to change the weight offset, which is intended to account for the springs in the scale that become loose over time. Unfortunately, this control panel was shipped with insecure credentials and those were not changed at installation, therefore making it simple for an attacker to change its calibration from a remote location.

The company also contains a temperature monitoring system that verifies the room temperature of the company ever so often to confirm that nothing out-of-the-ordinary is occurring. This system uses the MQTT protocol. However, similar to the previous problems, this protocol was used in a way that does not feature

authentication, encryption, or authorisation.

After the batter is complete, it goes into the oven for baking. This oven is controlled by another PLC that turns the oven on and off after twenty-five minutes have elapsed. The oven is controlled by the Siemens S7 Software. Using this software, the operators can communicate with the oven to change the baking temperature. The software also allows an HMI to read the oven's sensor information to verify the heat status. After the cookies have been baked, they are packaged and shipped to the customer.

Finally, as part of the factory upgrade, the company network contains a beta website that includes new features. Specifically, the current beta website contains a new network log that can be used for audit purposes. Usually, this beta website is known only to the programmer of the company. However, its existence can be detected using network traffic capture when the programmer tests the beta website.

4

CTF Problems Created

4.1 Problem 1: The Command Injection Attack

A *shell* is a program that takes action as input and then forwards it to the operating system to perform the action [39]. Our first CTF problem is about obtaining access to a shell using Command Injection. A *Command Injection* attack can occur when a host is vulnerable due to insufficient input validation, potentially allowing a bad actor to execute arbitrary commands on the server [40]. In this scenario, the cookie ordering website is vulnerable to such an attack in its mail field. The first problem will be solved once the user finds a hidden file after gaining shell access, and the file will contain the flag.

4.1.1 Problem Statement

Pipi and Flash are arch-enemies that enjoy making life difficult for each other regularly. Pipi noticed that Flash is hosting a cookie party and decides to ruin it by destroying Flash's order on the Hug&Run cookie website. To ruin the order, Pipi requires access to the internal network of the factory creating the cookies. However, Pipi only has access to the order website. Playing the role of Pipi, how will you gain

access to the internal network and find a flag hidden in one of the files?

4.1.2 Hints

1. Pipi reads online that lack of input validation is a common website vulnerability.

2. Pipi read online that PHP used to program the website is prone to have such a vulnerability.

3. Pipi notices a mail field using PHP that seems to contain a Command Injection vulnerability.

4. Pipi decides to use the command `cat` to output the data in a file.

4.1.3 Problem Reasoning

A Command Injection attack, though simple, can result in a significant breach if a user can create a reverse shell and access an internal network of a company. If an attacker gains access to the internal network of a company, it can lead to further exploits. This includes being able to communicate with IoT devices the factory owns. Keywords specifically added to the problem statement and hints are the following:

1. Input Validation

2. Internal Network

3. Command Injection

4. Hidden Flag File

The design of the website took into account other currently existing sites that are used to order items, such as Amazon, Insomnia Cookies and Whole Foods. The website itself contains a store overview, cart and a checkout option. There are only

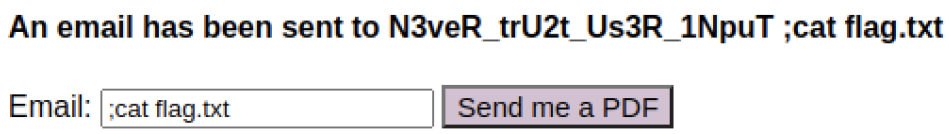

Figure 4.1: Successful Attack on the Cookie Ordering Website in Problem 1.

a few input boxes located on the website to not confuse the user as to which input box to exploit.

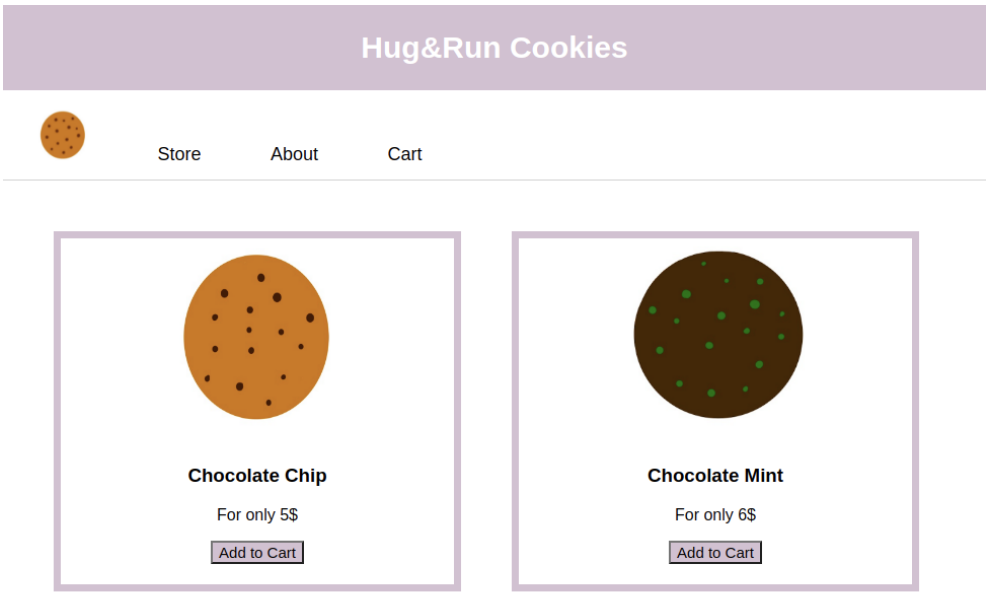

Figure 4.2: Homepage of the Cookie Ordering Website in Problem 1.

This problem teaches users about Command Injection vulnerabilities and the basics of a shell while making the case why users will, later on, be able to communicate with IoT devices the factory owns.

- The user will become more familiar with a Shell.
- The user will gain the ability to understand a Command Injection vulnerability.
- The user will understand the importance of input validation.

4.2 Problem 2: The Modbus Attack

Modbus is a popular communication protocol used in ICS [5]. The protocol was designed back in 1979, and thus it had a different threat model than today. This is why it can be vulnerable to multiple attacks nowadays [10]. Due to this, our second CTF problem is related to a Modbus vulnerability. In the fictional factory, Modbus is used to transfer information about the amount of flour to put into a cookie. However, the Modbus server of the batter mixer does not require authentication, thus allowing anyone to communicate with it.

This problem will have the user change the amount of flour of the cookies to teach them about this common vulnerability affecting Modbus. The user will have access to an HMI hosted on a website. The user can press a button to start the Modbus server. The users will then need to listen and write to a register on the Modbus server. A flag will be inserted into the input registers once the user changes the amount of flour to the expected malicious weight in the holding register.

4.2.1 Problem Statement

Pipi has successfully gained internal access via a reverse shell. This allowed for traffic monitoring of the internal network. While viewing this traffic, Pipi came across a Human Machine Interface (HMI) hosted on a website. After close examination, it seems this HMI interacts with a Modbus server to write information about the amount of flour to put into the cookie batter for each order. Pipi wonders if it is possible to increase the flour amount in Flash's cookie order from 45 grams per cookie to 60 grams to make them harder.

4.2.2 Hints

1. Pipi notices that there is no authentication on the Modbus server.

2. Pipi knows that Python contains libraries, such as pymodbus, that might help communicate with the server.

3. Pipi decides to read more about the difference between coil, discrete input, input register and holding register.

4. Pipi decides to update the information in a holding register to 600.

5. Pipi views the input registers out of curiosity and sees the numbers have changed. Perhaps they are characters?

4.2.3 Problem Reasoning

Modbus is one of the most popular IIoT protocols. Despite its usage in multiple companies, it is prone to security weaknesses as it was created in 1979 with no bad actors in mind [10]. Nowadays, it is recommended to implement Modbus with TLS, similar to how HTTP uses TLS, to prevent confidentiality breaches. Additionally, it is recommended to use certificates to verify both client and server before requests are authorised. Keywords specifically added to the problem statement and hints are the following:

1. Modbus

2. Coil

3. Discrete Input

4. Holding Register

5. Input Register

6. Human Machine Interface

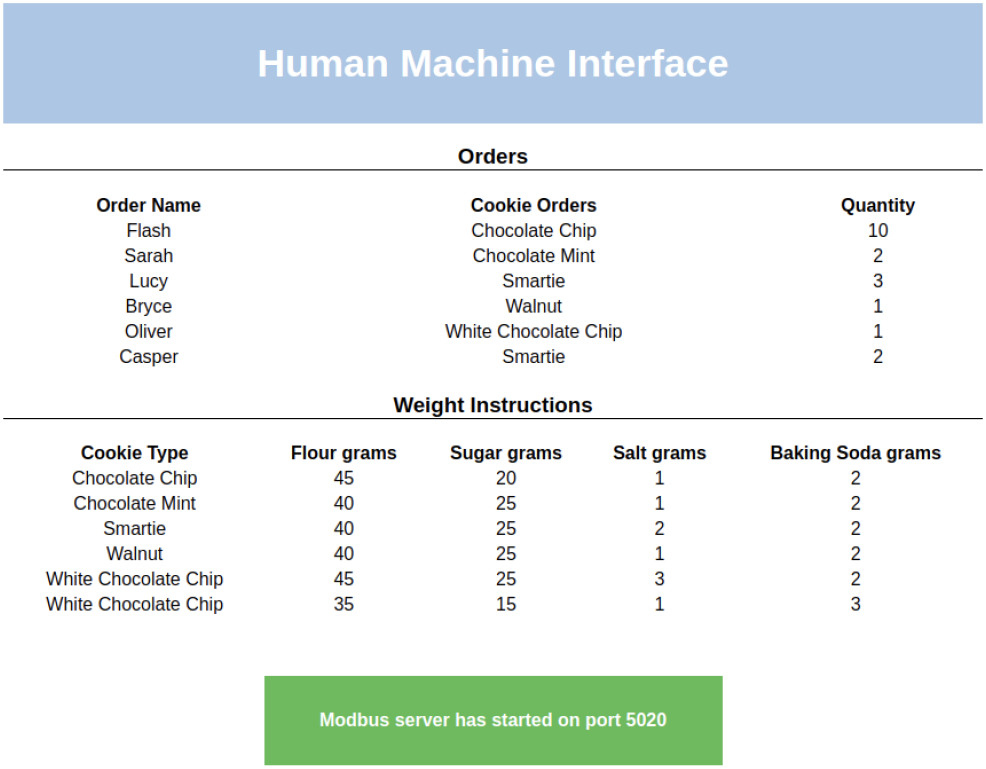

Figure 4.3: Web-Based HMI of the Cookie Production Line in Problem 2.

This problem contains both a Modbus server and an HMI to communicate with it. The structure of the HMI website was chosen to be simple not to overwhelm the user. The content shown on the HMI includes information of those who have recently ordered cookies at the company website, such as Flash. Information about the weight of the ingredients in each type of cookie is given to the user to make it easier to spot the information in the registers later on. The initial value for Flash flour weight will be 450 grams because Flash ordered ten chocolate chip cookies.

The button to start the Modbus server was inserted on the website for two reasons. First, it is to convince the user why an attacker can specifically target the order by Flash. Second, it informs the user which port the Modbus server can be found on.

This problem teaches users to spot Modbus weaknesses and exploit them.

- The user will become familiar with the common IoT protocol Modbus.
- The user will become familiar with a common Modbus vulnerability.
- The user will understand the importance of authentication in Modbus.

4.3 Problem 3: The Weight Scale Attack

The fictional company uses a weight scale to verify the expected weight of the cookie batter before baking. Once the weight scale has calculated the batter weight, a Modbus Server reads the information. Unlike in the previous problem, the Modbus server in this problem uses TLS for client authentication to prevent malicious activity. Due to this, the user will not be able to access it without the proper certificate. However, the weight scale itself turns out to be vulnerable because it can be administered using a web-based control panel using the correct credentials. This problem will be solved once the user gains access to the weight scale and changes the weight offset to prevent the changed batter weight from the previous problem from being flagged as incorrect.

4.3.1 Problem Statement

Pipi has successfully changed the total weight amount for the cookie batter in Flash's order from 680 grams to 830 grams. However, after closer examination of the network traffic, Pipi notices a weight scale checks the weight of the cookie batter. Pipi sees that all the Modbus traffic of the scale is secured with TLS, and when connecting to the corresponding Modbus server, that server requests a certificate for authentication. Almost losing hope, Pipi remembers the Mirai attack and, after a long search, sees that the weight scale has a web-based control panel.

4.3.2 Hints

1. Pipi wonders how the Mirai attack was possible and if this scale suffers a similar problem.
2. Pipi wonders if there are password lists available to be used.
3. Pipi spots the Mirai leaked password lists and wonders if any of those passwords would work as well.
4. Pipi has successfully gained access to the control panel! Pipi now wonders how to change the reported weight from 830 grams to 680 grams.

4.3.3 Problem Reasoning

This attack was created to teach users part of the Mirai attack. In the Mirai attack, cameras and devices were attacked with brute force methods to gain access to the systems [27]. Once access was gained, the system could be reconfigured and have malware installed on it [27]. Due to this, the password used in this challenge is one of the leaked passwords used in the Mirai attack. This will drive the user to read more about this attack. Keywords specifically added to the problem statement and hints are the following:

1. TLS
2. Certificate for Authentication
3. Mirai Attack
4. Leaked Passwords

The web-based control panel of the weight scale contains limited information based on what could be expected to see on this type of device. The website contains

Figure 4.4: Login Page of the Weight Scale Control Panel in Problem 3.

a login page, but the credential information it asks for, namely the username and password, was one of the default credentials found in the leaked list used in the Mirai attack.

Once the supplied credential has been validated, the website will contain information such as logs, settings and a home page. The home page contains information about a weight offset, but the hope is that the user navigates to the settings tab to change it.

This problem teaches users the importance of securing all entry points to a system. Furthermore, it teaches users that using weak or previously leaked credentials is a major security risk.

- The user will understand how Modbus can be secured with TLS and authentication.
- The user will understand the importance of securing all entry points.
- The user will understand the importance of changing the default password on

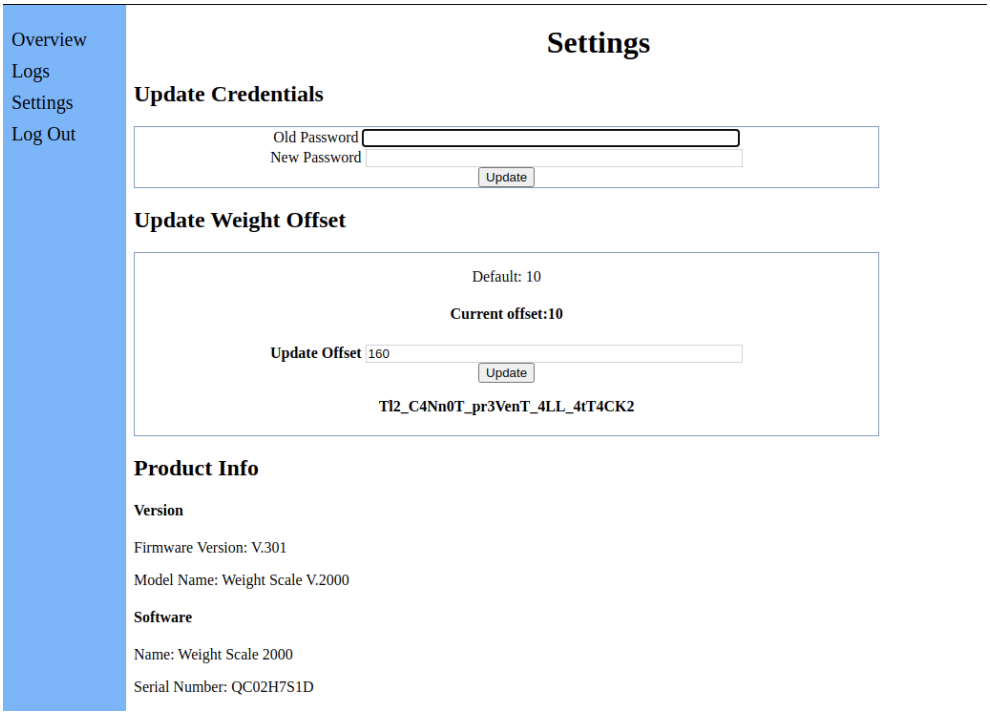

Figure 4.5: Settings Page of the Weight Scale Control Panel in Problem 3.

a device to a strong password.

4.4 Problem 4: The MQTT Attack

MQTT is a subscribe-publish protocol used to exchange messages between clients [14]. The messages are organised around the concept of "topics". In a secure deployment, the protocol should be protected with security features such as authentication, authorisation, and secure communication. However, if these features are not used, it could result in a malicious user being able to subscribe or publish to a topic.

Our fictional factory uses MQTT to report and monitor the temperature in the company. In the problem given to the user, the attacker wants to cause a denial of service to the subscriber(s) of the temperature topic. The problem can be solved by having the user spoof the actual client of the monitoring service. The flag will then be published on a different topic when this occurs.

4.4.1 Problem Statement

Pipi has successfully managed to get the tampered cookie batter to pass the assurance check at the weight scale. However, this is not enough to ruin the party. The cookies will need to be burnt! To do so, the oven temperature has to be tampered with, but this might result in smoke alarms activating. Pipi decides to look further into the network to see if any temperature alarms are connected. After further examination, Pipi notices the MQTT protocol in the network traffic and sees that it is used to publish temperature values of the factory between clients. Pipi wonders if it is possible to spoof a subscriber of the traffic as a form of denial of service attack.

4.4.2 Hints

1. Pipi wonders how to create an MQTT client that subscribes to topics.
2. Pipi wonders which client identifiers are already being used.
3. Pipi sees that one subscriber is using 1 as its identification number.
4. Pipi wonders which other topics are available.
5. Pipi wonders if there need to be two scripts running at the same time, one for attacking and one for listening.

4.4.3 Problem Reasoning

This problem teaches students about the MQTT protocol and its security vulnerabilities. Keywords specifically added to the problem statement and hints are the following:

1. MQTT
2. Subscriber/Publisher

Figure 4.6: Network Diagram Before the MQTT Attack Starts in Problem 4.

Figure 4.7: Network Diagram After the MQTT Attack Starts in Problem 4.

3. Denial of Service

4. Spoofing

The MQTT deployment in the fictional factory was deliberately made simple, with only one client and one server being used. There were two main topics created, both under the path *sensor* to make it easier to find. Before the attack occurs, one client starts publishing information at a regular interval to the topic *temperature*. At the same time, a subscriber with identification value 1 listens.

After the subscriber is kicked off the network once, they will begin publishing to the topic *flag*. Two topics were chosen to help the user learn that MQTT supports multiple topics.

Spoofing a client can result in a significant integrity breach. Unfortunately, this is a severe problem in many protocols because they were designed with no bad actors in mind. The spoofing attack in this problem demonstrated how a denial of service attack may be performed on another MQTT client.

- The user will understand how to use the MQTT protocol.
- The user will understand the basics of spoofing attacks.
- The user will gain experience in performing a Denial of Service attack.

4.5 Problem 5: The Oven Attack

Once the cookie batter is ready, it is inserted into the oven for 25 minutes. The temperature varies between cookies, but it cannot be higher than 375 degrees Fahrenheit. In the fictional factory, the oven contains an integer vulnerability because the temperature setting it receives is a signed integer, but the input validation only checks if the temperature setting is not higher than 375 degrees. However, if the temperature setting passes this safety check, the (signed) integer is placed into an unsigned variable to control the heating element of the oven.

In this scenario, the attacker can use a negative number to bypass the oven temperature checker as the setting will become a large integer after being placed in the unsigned variable. A flag will be given to the user if the temperature is increased to 400 degrees Fahrenheit, while the information the HMI reads is 375.

4.5.1 Problem Statement

Pipi has successfully managed to turn off the temperature alarm and now feels confident about destroying the cookies in the oven! Pipi sees that the cookies are placed in an oven for 25 minutes. After further investigation, Pipi notices that the software Siemens S7 communicates with the oven to change the temperature. Pipi decides that for the cookies to taste burnt, they should be baked at 400 degrees Fahrenheit instead of 375. After pen-testing for a while, it seems the oven will not take such a high value as the oven has been programmed to have a safety cutoff at 375 degrees. However, lower values do not produce an error. Pipi also knows that an HMI is used

to view the oven's temperature. Thus, the address the HMI reads from would need to specify the expected value of 375 degrees.

4.5.2 Hints

1. Pipi notices that the oven uses address 1 to see the updated temperature.
2. Pipi notices that the address oven uses takes in a signed integer but places it in an unsigned variable if the value is not higher than 375.
3. Pipi notices that the HMI uses address 5 to read the temperature of the oven.
4. Pipi wants the HMI to see 375 as opposed to 400 after attacking the oven.
5. Pipi always makes sure to read the updated address values to see if the attack worked.

4.5.3 Problem Reasoning

The Stuxnet attack occurred in 2010 [29]. The Stuxnet worm targeted Windows machines with the software S7 by manipulating the PLC registers. The worm tried to make the attack go undetected, such as by telling devices incorrect information about the status of the system. This problem is based on this attack as this was the first known attack against ICS systems [30]. Keywords specifically added to the problem statement and hints are the following:

1. Signed Integer
2. Unsigned Integer
3. Siemens S7 Software
4. PLC
5. HMI

Figure 4.8: Network Diagram in Problem 5.

The problem design has one S7 server that exposes two addresses, one for an oven and another for an HMI. The address space with identification number 1 was used for the oven temperature, while the HMI read from address 5. Sequential addresses were not chosen, such as 1 and 2, to make it slightly more difficult for the user to find the correct address to access.

The threshold the system uses to update the oven temperature is flawed in that it only makes sure that the value is not too high, but it uses a signed variable. Yet once the received value bypasses this check, it is stored in an unsigned variable. Additionally, the program uses only the first 9 bits as the other bits are used for configuration purposes and because the value should never need that many bits.

The flag the user receives is given once both attacks are successful. The flag is placed in one of the updated registers. This will test users if they understand how to read and write data in an S7 deployment.

This problem teaches students about similar attack methods used in Stuxnet.

- The user will become more familiar with integer vulnerabilities.
- The user will understand the Stuxnet attack better.
- The user will become more familiar with Siemens S7.

4.6 Problem 6: The Dogfood Attack

The company currently has two almost identical websites running, the regular ordering website and a beta website for testing new features. The beta website should be a hidden internal service that users should not know of. As it happens, developers often forget to safeguard these beta websites under the false impression that they are not intended to be accessible to the public.

In this problem, Pipi sniffs the network traffic and comes across another public website. The website itself seems identical to the ordering website, except it contains an extra tab labelled "admin". On this admin tab, it is possible to view and delete logs. The user will retrieve the final flag if they find the website in a given packet capture file (PCAP) and then delete the logs.

4.6.1 Problem Statement

Pipi has successfully ruined Flash's cookie party. However, Pipi wonders if there were any logs captured during the attack. Pipi uses Wireshark to see what type of hosts are on the network to help pivot between hosts. While watching the Wireshark traffic, Pipi notices something strange.

4.6.2 Hints

1. Pipi wonders if network segmentation is used in the company network.
2. Pipi notices an IP address in a segment not typically used in the company.
3. Pipi sees that IP address hosts a beta webserver.
4. Pipi manages to connect to that server and find a log file!

4.6.3 Problem Reasoning

A similar attack occurred in 2020 when Ezequiel Ereira found an internal Google Service in the Google Cloud by viewing logs [41]. This attack allowed him access to the Google bug tracker, leading Google to pay him a large bug bounty. This shows that such vulnerabilities can occur, making it essential for students to understand.

This problem was implemented for a few reasons. First, it teaches users that critical network traffic should be monitored with logs. Second, it pushes users to learn how to view and reason with IoT traffic because the PCAP file provided contains IoT protocols such as MQTT and Modbus. Third, it helps users understand how network segmentation is used in enterprise networks. Fourth, it gives our story an ending that is typical in cyberattacks, where the attackers erase their tracks. Keywords specifically added to the problem statement and hints are the following:

1. Wireshark
2. Network Segmentation
3. Network Logs and Audits

The PCAP file given to the users contains one public IP address that is meant for the website developer. The IP address of this website is set by a script when the problem is built on the picoCTF platform. All of the information shown in the PCAP file is traffic from the previous problems. This helps the user gain experience in understanding the network traffic generated by common IoT protocols.

The beta website was designed to have the same structure as the first problem, but an additional admin tab was added. If users played the expected storyline, they should notice that this is the only difference between the two websites. The admin tab itself only contains two buttons, one to view logs and another to delete them. If

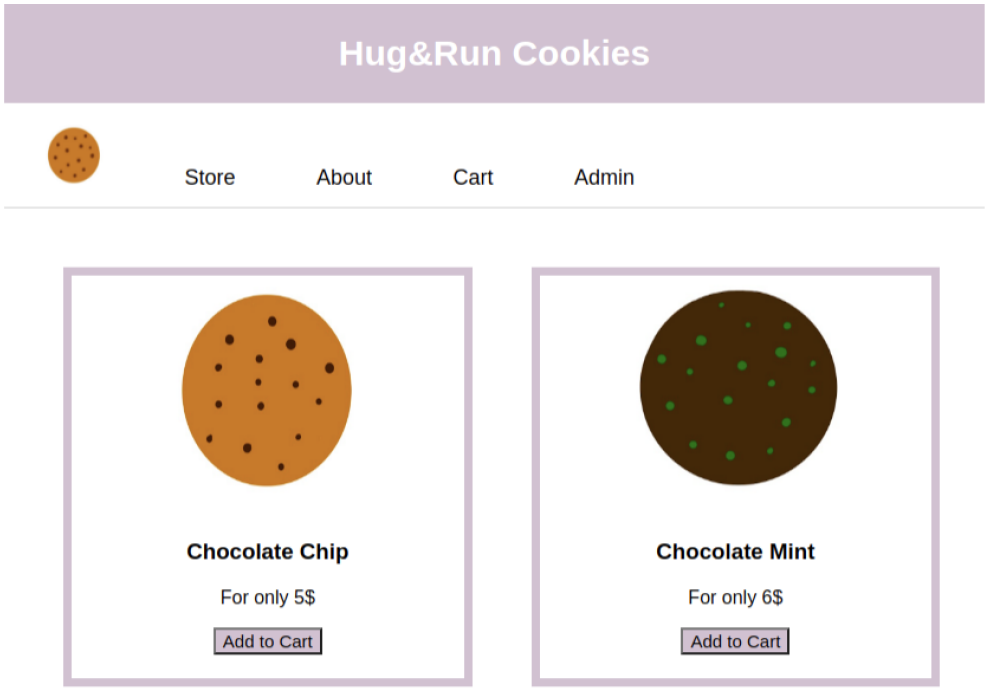

Figure 4.9: Homepage of the Beta Cookie Ordering Website in Problem 6.

the user deletes the logs, a flag will be put into the updated log file and the user will see the flag when the user verifies that the log has indeed been erased.

Our final problem will teach users the importance of securing internal services even when they are not meant to be used by normal users.

- The user will understand the importance of securing internal services.
- The user will understand the importance of log files and audits.

5

Personal Growth

5.1 Major Lessons Learned

The "Internet of Things" is a term that has become popular over the years. The most distinctive feature of IoT devices is that they do not require human assistance to operate or communicate with among each other. While numerous people learn about IoT due to their usage for "Smart Homes", it is highly beneficial for manufacturing too; therefore, the term "Industrial IoT" (IIoT) was introduced to describe the increasing inter-connectedness and automation among industrial devices. Unlike businesses that only deal with IT, industries often need a more complex network architecture to organize operations and coordinate between the business and the manufacturing side. That is why the Purdue model is often used to simplify the design and the description of the overall architecture. To reflect this trend, the CTF problems created in this thesis are based on a fictional factory that uses this model.

We found that the creation of CTF problems is more complex than initially anticipated. Developing problems that provide users with a meaningful learning objective relevant to modern societies forced us to read a large amount of documentation. We

believe that a careful balance is needed when deciding the amount of information presented to the users in a CTF problem. Having too much information in a problem could result in users being overwhelmed or not challenged. However, having too little information could result in users potentially quitting due to frustration. In our experience, it was quickly noticed that the problem creation process became more straightforward when keywords and learning objectives were determined before creating the problem statement.

5.2 Challenges Encountered

We noticed that understanding the structure of industrial companies without proper access to visit them proved to be quite difficult. Documents had to be read multiple times to properly understand standard ICS and IoT models, especially for a student with little prior background in these domains. Seeing an actual ICS deployment in real life could have helped to accurately design and present the fictional factory used in our CTF problems. Unfortunately, due to the COVID-19 pandemic, a visit to such a factory was not possible. However, this might have been a blessing in disguise. Our experience in this thesis demonstrates that creating IIoT CTF problems may be possible without physical access to industries; instead, books and online articles and videos provided sufficient information to make meaningful progress.

Writing the CTF problems in a coherent storyline was complicated, but the fictional factory had to represent an infrastructure accurately. Creating flags to fit the storyline and could be found by the users was incredibly challenging, such as in problems 2 and 5 that are based on network protocols. That is why some hints contained detailed information.

Problems had to be re-written multiple times, such as problem 3, which was initially supposed to be about a Modbus server that uses certificates. The hope was that it would show users what would occur if attempting to communicate with

a Modbus server without the proper credentials. However, it was quickly noticed that creating a correctly signed certificate is complex. In the end, the problem was modified as it did not fit the intended learning objective. This complication might explain why certificates might not be updated in practice as often as they should, as it can be challenging and time-consuming.

5.3 Other Revelations

It was surprising to notice how many common concerns IT and non-IT security share. This might be because both had designs developed with an outdated threat model that does not fit the circumstance in modern societies. However, this can also be because humans design both, and humans often tend to make similar mistakes. It is possible to see similar problems on both sides, such as the use of weak passwords or the lack of authentication altogether.

Another revelation is that ICS systems use a security model known as a *Defense in Depth*. This model reminds one of the stories told to children about a castle, where there is a ditch filled with water around the gates to secure the royalty inside. The moat would additionally contain a crocodile to prevent attackers from swimming it. Furthermore, if the attacker succeeds in circumventing the moat by chance, they are met with a gigantic wall and guards that are both difficult to avoid. A properly designed modern ICS system should be structured using a defense-in-depth model, where intruders are kept out with a demilitarized zone, as shown in the Purdue model. In addition, TLS and authentication should be used in case attackers gain internal access. Indeed, it is not wise to rely on only one security mechanism to prevent intrusion.

We found it surprising that modern industrial control systems are programmed similar to most other computers. As an IT student, the assumption was that it would not be possible to learn about ICS and IoT devices as they would not contain

any similarities to IT, which is based on programs. Thus, it was delightful to notice that the CTF problems created in this thesis were programmed like any other IT application.

Finally, we learned that use of certificates in an industrial setting could differ from the typical use of certificates that IT students learn in browser security. In industrial settings, certificates are used to verify both the server and the client, while in the browser settings, certifications are commonly used to verify only the server. After learning about this distinction, it was quickly understood why enterprises might prefer to manage SSH access using certificates instead of using an SSH public key. This could potentially avoid the management nightmare where every SSH server needs to keep an up-to-date list of each valid SSH public key in the company.

6

Concluding Remarks

The Internet of Things (IoT) are made up of devices with embedded sensors and controllers. Their usage is becoming increasingly popular over the years, both in homes and industries. Unfortunately, IoT devices are prone to numerous attacks because they were developed with an outdated threat model or contain implementation flaws. This thesis presents six Jeopardy-style CTF problems focusing on Industrial IoT (IIoT) security to teach more students about the potential security vulnerabilities of IoT devices and the emerging paradigm of IIoT.

The CTF problems created involve two arch-enemies and a fictional cookie factory. The factory created for this thesis is based on an industrial control systems architecture known as the Purdue model. Furthermore, it uses IoT devices for automation, including adding flour to a batter and weighing it before baking. Unfortunately, these devices have security vulnerabilities that an attacker can use to sabotage.

This thesis shows that it is possible to create a safe, scalable and economic environment for users to learn about IoT vulnerabilities. Our approach is safe because

the CTF problems revolve around a fictional factory, meaning that attacks on the systems cannot cause physical harm to anybody. It is also economic because our CTF problems depend on open-source software and do not require any actual hardware. Finally, it is also scalable because the CTF problems created, which are software-only, can be deployed on the picoCTF platform to serve many users simultaneously.

This thesis shows that teaching non-IT security to an IT student is possible, as this thesis was created and completed by a student with little prior knowledge of ICS and IoT systems. The problems created were developed during the COVID-19 pandemic, meaning that the creator could not visit industrial factories, which is why all of the problems created were developed by reading papers and viewing online material.

From the perspective of creating these CTF problems, it was noticeable that there are many common characteristics shared by IT and non-IT security. This includes insecure designs, using outdated components and unfortunate security misconfigurations. Since current security education seems to focus heavily on IT security, IT students may not realise that their education is already applicable to modern non-IT security. Hopefully, this thesis will encourage more IT students to explore non-IT security in the future.

6.1 Future

It would be interesting to see the CTF problems developed in this thesis become available to other students on the picoCTF platform. The possibility of receiving feedback on the CTF problems created could allow for a better understanding of challenges other IT students might face when learning non-IT security. Furthermore, it would be interesting to collect statistics on the time spent by students to finish the CTF problems and other feedback from the students.

We also believe that it would be interesting to use the feedback collected to create

another story regarding the revenge of Flash. This new story could then feature more types of physical devices to further expose students to the cyber-physical nature of modern societies.

Creating hints that fit the storyline while giving users the information required to finish the problems proved to be tricky. Most hints provided in the six CTF problems were deliberately long as we consider it more suitable for self-learners. It would be interesting to perform an experiment in the future by varying the number of hints and measuring the education outcomes and enjoyability of our CTF problems.

Lastly, assuming these CTF problems are deployed on the picoCTF platform publicly in the future, it would be interesting to observe if the number of different genders that play our CTF problems would be significantly different from that on the traditional CTF problems. This is because our storyline deliberately avoids including information about genders, thus allowing users to decide what they believe about the genders of Pipi and Flash. This may have a positive effect on inclusiveness since we avoid creating any prejudicious gender bias that can turn away potential audiences.

Bibliography

[1] A. Martens and W. Mueller, "Gamification - a structured analysis," in *Proceedings of the 16th International Conference on Advanced Learning Technologies (ICALT)*. IEEE, 2016, pp. 138–142.

[2] K. Ashton, "That 'Internet of Things' thing," Jun 2009, [Online; accessed 18. Nov. 2021]. [Online]. Available: http://www.itrco.jp/libraries/RFIDjournal-That%20Internet%20of%20Things%20Thing.pdf

[3] P. Dudhe, N. Kadam, R. M. Hushangabade, and M. S. Deshmukh, "Internet of Things (IoT): An overview and its applications," in *Proceedings of the 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, 2017, pp. 2650–2653.

[4] A. Karmakar, N. Dey, T. Baral, M. Chowdhury, and M. Rehan, "Industrial Internet of Things: A review," in *Proceedings of the 2019 International Conference on Opto-Electronics and Applied Optics (Optronix)*, 2019, pp. 1–6.

[5] P. Ackerman, *Industrial Cybersecurity: Efficiently secure critical infrastructure systems*. Packt Publishing, Oct 2017.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1," RFC Editor, RFC 2616, 1999, [Online; accessed 18. Nov. 2021]. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2616

[7] M. Belshe, R. Peon, and M. Thomson, "Hypertext transfer protocol version 2 (HTTP/2)," RFC Editor, RFC 7540, 2015, [Online; accessed 19. Nov. 2021]. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7540

[8] M. Bishop, "Hypertext transfer protocol version 3 (HTTP/3), draft-ietf-quic-http-34," RFC Editor, RFC, 2021, [Online; accessed 19. Nov. 2021]. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34

[9] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," RFC Editor, RFC 7252, June 2014, [Online; accessed 19. Nov. 2021]. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7252

[10] "Modbus application protocol specification v1.1b3," June 2004, [Online; accessed 19. Nov. 2021]. [Online]. Available: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[11] "Modbus/TCP security," Aug 2018, [Online; accessed 22. Nov. 2021]. [Online]. Available: https://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf

[12] "PROFINET system description: Technology and application," 2014, [Online; accessed 22. Nov. 2021]. [Online]. Available: https://www.profibus.com/index.php?eID=dumpFile&t=f&f=51714&token=4ea5554cbb80a066e805a879116ead2a759c23c3

[13] "PROFINET security," 2021, [Online; accessed 22. Nov. 2021]. [Online]. Available: https://www.profibus.com/technology/industrie-40/profinet-security#tab2-222211

[14] "MQTT version 5.0," Mar 2019, [Online; accessed 19. Nov. 2021]. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf

[15] IBM, "Telemetry use case: Home patient monitoring," 2021, [Online; accessed 19. Nov. 2021]. [Online]. Available: https://www.ibm.com/docs/en/ibm-mq/8.0?topic=cases-telemetry-use-case-home-patient-monitoring

[16] T. Kivinen and P. Kinney, "IEEE 802.15.4 information element for the IETF," RFC Editor, RFC 8137, May 2017, [Online; accessed 19. Nov. 2021]. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8137

[17] "ZigBee specification," Aug 2015, [Online; accessed 19. Nov. 2021]. [Online]. Available: https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf

[18] P. M. Linh An and T. Kim, "A study of the Z-Wave protocol: Implementing your own smart home gateway," in *Proceedings of the 3rd International Conference on Computer and Communication Systems (ICCCS)*, 2018, pp. 411–415.

[19] S. J. Danbatta and A. Varol, "Comparison of ZigBee, Z-Wave, Wi-Fi, and Bluetooth wireless technologies used in home automation," in *Proceedings of the 7th International Symposium on Digital Forensics and Security (ISDFS)*, 2019, pp. 1–5.

[20] Silicon Labs, "Z-Wave security," Nov 2021, [Online; accessed 30. Nov. 2021]. [Online]. Available: https://www.silabs.com/wireless/z-wave/specification/security

[21] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of Things: Security vulnerabilities and challenges," in *Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC)*, 2015, pp. 180–187.

[22] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," RFC Editor, RFC 5246, 2008, [Online; accessed 22. Nov. 2021]. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5246

[23] OWASP, "OWASP Internet of Things," 2021, [Online; accessed 22. Nov. 2021]. [Online]. Available: https://owasp.org/www-project-internet-of-things

[24] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to industrial control systems (ICS) security," NIST, Special Publication 800-82, Jun 2015.

[25] AWS IoT, "Industrial IoT | Digital Transformation," Nov 2021, [Online; accessed 5. Dec. 2021]. [Online]. Available: https://aws.amazon.com/iot/solutions/industrial-iot

[26] Cloudflare, "What is a distributed denial-of-service (DDoS) attack?" Dec 2021, [Online; accessed 3. Dec. 2021]. [Online]. Available: https://www.cloudflare.com/en-gb/learning/ddos/what-is-a-ddos-attack

[27] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[28] J. Margolis, T. T. Oh, S. Jadhav, Y. H. Kim, and J. N. Kim, "An in-depth analysis of the Mirai botnet," in *Proceedings of the 2017 International Conference on Software Security and Assurance (ICSSA)*, 2017, pp. 6–12.

[29] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *Proceedings of the 37th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2011, pp. 4490–4494.

[30] K. E. Hemsley and R. E. Fisher, "History of industrial control system cyber incidents," 12 2018. [Online]. Available: https://www.osti.gov/biblio/1505628

[31] CISA, "Stop ransomware," Dec 2021, [Online; accessed 3. Dec. 2021]. [Online]. Available: https://www.cisa.gov/stopransomware

[32] M. Satheesh Kumar, J. Ben-Othman, and K. G. Srinivasagan, "An investigation on WannaCry ransomware and its detection," in *Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 1–6.

[33] CISA, "Cyber-attack against Ukrainian critical infrastructure," https://us-cert.cisa.gov/ics/alerts/IR-ALERT-H-16-056-01, (Accessed on 11/20/2021).

[34] C.-K. Chen and S. Shieh, "CTF: Alternative training for offensive security," 2015, [Online; accessed 3. Dec. 2021]. [Online]. Available: https://rs.ieee.org/images/files/techact/Reliability/2015-08/2015-08-a05.pdf

[35] "picoCTF - CMU cybersecurity competition," Nov 2021, [Online; accessed 24. Nov. 2021]. [Online]. Available: https://picoctf.org

[36] "IoT Village - a security hacking event," 2021, [Online; accessed 5. Dec. 2021]. [Online]. Available: https://www.iotvillage.org

[37] "Ph0wn CTF - a capture the flag for smart devices," Dec 2021, [Online; accessed 5. Dec. 2021]. [Online]. Available: https://ph0wn.org

[38] "Global collegiate penetration testing competition (CPTC)," Dec 2021, [Online; accessed 5. Dec. 2021]. [Online]. Available: https://cp.tc

[39] "picoCTF primer," Oct 2021, [Online; accessed 25. Nov. 2021]. [Online]. Available: https://primer.picoctf.org

[40] MITRE, "CWE-78: Improper neutralization of special elements used in an OS command ('OS command injection') (4.6)," 2021, [Online; accessed 18. Sept. 2021]. [Online]. Available: https://cwe.mitre.org/data/definitions/78.html

[41] H. Sharma, "Announcing the winners of the 2020 GCP VRP prize," Mar 2021, [Online; accessed 22. Nov. 2021]. [Online]. Available: https://security.googleblog.com/2021/03/announcing-winners-of-2020-gcp-vrp-prize.html