2024 International Conference on Smart Applications, Communications and Networking (SmartNets) | 979-8-3503-8532-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/SMARTNETS61466.2024.10577747

Securing IoT Surveillance Airport Infrastructure

Catalina Aranzazu-Suescun, PhD.*, Luis Felipe Zapata-Rivera, PhD.†,
Owen Guerra-Mondragon Saenz†, and Jacob Michael Christensen*
*Embry-Riddle Aeronautical University, USA, Department of Cyber Intelligence and Security,
†Embry-Riddle Aeronautical University, USA, Department of Computer, Electrical and Software Engineering

***Abstract*—Over the last three decades, airports have adopted the Internet of Things (IoT) technology in their core services. Luggage scanning, ticketing, and security checkpoints are becoming increasingly automated, providing a more efficient and reliable process. However, IoT merchants disregard the importance of securing their devices, focusing only on the improvements following automation. This oversight has opened the door to numerous opportunities for attackers to exploit vulnerabilities in their system, resulting in damage to the company's reputation.**

Probable attacks are identified and ranked according to their respective damage to the system through threat modeling. After system implementation, cybersecurity professionals require periodic vulnerability assessments that outline potential shortcomings of the system's security to propose solutions to palliate the risks further.

This project presents a test implementation of an IoT-based camera system using open-source technologies and protocols commonly implemented in commercial solutions. The prototype monitors a given area in a remote location using servo motors that are connected to Raspberry Pis to support pan and tilt (PT) functionality for the cameras.

This project aims to apply forensics methodologies typically used in information technology (IT) enterprise networks in the context of IoT systems. OWASP's Risk Rating Methodology is used to identify and rate the likelihood and technical impacts of potential risks on the IoT surveillance testbed network. Well-known and trusted tools were used to assess vulnerabilities in the system's implementation. Based on the results of the vulnerability assessment, a series of attacks were performed on the system to compromise the confidentiality, integrity, and availability of the system and data. The findings from these attacks were used to propose recommendations to secure IoT systems in surveillance applications.

***Index Terms*—Internet of Things, Forensics Methodologies, OWASP, Surveillance, Vulnerability Assessment.**

I. Introduction

Internet of Things (IoT) systems are commonly used in almost every major industry including health, public infrastructure, and agriculture, among others. With millions of devices that are now connected in cyberspace, attackers have now many opportunities to exploit these systems. Most IoT system vendors do not consider implementing security mechanisms in the devices that they produce, whether it be due to restrictions in cost or limitations in available resources.

Securing IoT systems is becoming a critical task; statistics show that more than 50% of the IoT traffic is originated in the manufacturing and retail sectors. Every year, the malware attacks on IoT devices and servers increase by 400% [1].

68% of the devices are operating with critical or high-risk vulnerabilities. These devices are easy targets for attackers for many reasons including bad practices in device administration (such as default authentication credentials), improper configuration that leaves them open for being accessed from remote networks, or not activating security features such as encryption or logging [1].

26% of the IoT active devices are no longer supported by their manufacturers, which is also known as end-of-life. This poses greater risks of becoming the target of attacks by hackers [1].

Given the vast number of internet-connected devices, attackers find IoT applications appealing for exploiting security vulnerabilities. Manufacturers of IoT devices often prioritize cost reduction, targeting a broad market with inexpensive products. Consequently, incorporating security measures becomes a costly consideration that is deemed not worth the investment.

Threat modeling is a critical component of IoT security design. This process systematically identifies possible attacks against a system or device and then prioritizes the issues based on their severity. Once the system is implemented, vulnerability assessments are done periodically to identify potential exploits or weaknesses in a system. The result of the process allows cybersecurity experts to propose solutions that can mitigate these risks [2].

Nowadays, surveillance camera systems can be IoT-based due to the possibility of integrating cameras over an internet or local area network. Users can now remotely access and control the camera system using a computer or a mobile device. However, the security of these systems can be compromised, exposing private information and creating a risk to the rest of the network modules. In airports, IoT has been used to monitor passengers, anomalous events, and incidents inside the airport facilities, and to have access controls of the different facilities. In the line of surveillance, authors of [3] combine machine learning and CCTV surveillance cameras for detecting, identifying, and alerting to abnormal human behaviors in airports. Authors of [4] use edge computing and IoT systems for parking occupancy detection. Similarly, authors of [5] use artificial intelligence (AI), the Internet of Things, and cloud computing technologies for monitoring parking spots in airports. Authors of [6] also work in the development of an IoT surveillance system with cameras and drones to assist in detecting dangerous situations in public places.

This paper presents an overview of the implementation, threat modeling, vulnerability assessment, and results of the attacks on an airport surveillance system. The rest of the paper is organized as follows. Section II presents the main standards and protocols in IoT systems. Section III shows the design and implementation of a Surveillance system testbed, plus the threat modeling, vulnerability assessment, and results of a series of attacks. Section IV presents the discussion of the results given in the previous section and a list of recommendations for securing IoT systems. Finally, the conclusions are stated in section V.

II. IoT Background

A. IoT Standards and Projects

To have a unified set of definitions related to the design and operation of IoT devices, multiple initiatives have emerged, the most important is the Matter project [7], and the definitions made by IEEE with the IEEE Standard 2413-2019.

The primary goal of Project Matter is to develop a unified standard for IoT devices to ensure better compatibility, security, and ease of use for consumers. This initiative is a collaborative effort initiated by major technology companies, including Apple, Google, Amazon, and the Zigbee Alliance. These companies joined forces to create a standard that would work across various platforms and ecosystems. The project is open source, allowing developers to contribute and collaborate on the development of the standard. The objective of leaving the development as an open-source product is to encourage developers to adopt the project definitions.

The IEEE Standard 2413-2019 for an Architectural Framework for the Internet of Things (IoT). A framework description for the Internet of Things (IoT) This architecture framework addresses the concerns reported by stakeholders involved in IoT systems across diverse domains such as healthcare, transportation, and Smart Grid, among others. The foundation for the concept of ”things” within the IoT is introduced as well as a set of architectural viewpoints that compose the base of the framework description [8].

B. IoT Protocols

In the IoT communications world, multiple communication protocols and technologies are used, Some of the most popular are Thread [9], Zigbee [10], Z-Wave [11], and MQTT [12].

Thread is a low-power mesh protocol designed for battery-powered devices. It uses 6LoWPAN, which uses the IEEE 802.15.4 wireless protocol with mesh communication in the 2.4 GHz frequency spectrum.

Thread tackles challenges of IoT networks such as interoperability, range, security, energy efficiency, and reliability. It relies on existing technologies across all its layers, encompassing routing, packaging, security, and wireless radio technology to minimize the risk of compatibility issues with a diverse range of devices.

Zigbee is a wireless communication protocol designed for low-power, short-range communication between devices in a personal area network (PAN). It operates on the IEEE 802.15.4 standard and is commonly used in applications such as home automation, industrial control, and healthcare. Zigbee is characterized by its low power consumption, low data rate, and ability to form mesh networks, where devices can relay data for extended coverage. It uses a star or mesh topology and supports secure communication through encryption. Zigbee is widely adopted for its efficiency in connecting and controlling a variety of IoT devices within a local network.

Z-Wave is a wireless communication protocol used primarily in smart home networks, allowing smart devices to connect and exchange control commands and data with each other. With two-way communication through mesh networking and message acknowledgment, the Z-Wave protocol helps alleviate power issues and brings low-cost wireless connectivity to home automation, offering a lower-power alternative to Wi-Fi and a longer-range alternative to Bluetooth.

The MQTT protocol [12] started as a messaging transport protocol in 1999; it was originally designed for monitoring oil pipelines in conjunction with Supervisory Control and Data Acquisition (SCADA) industrial control systems. During the last 10 years, MQTT has gained widespread adoption in IoT systems such as Industrial IoT (IIoT) setups, and machine-to-machine (M2M) applications. The key features of MQTT include:

- Bandwidth efficiency due to low overhead
- Lightweight protocol suitable for constrained devices Scalability
- Bidirectional communication
- Data agnosticism

MQTT utilizes persistent Transmission Control Protocol (TCP) connections, keeping the connectivity between the device and the broker until necessary. Operating on a publish-subscribe architecture, MQTT involves three primary actors in its framework: the broker, the publisher, and the subscriber, as shown in figure 1.

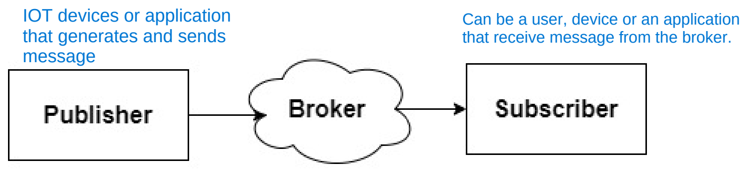

IOT devices or application that generates and sends message
Can be a user, device or an application that receive message from the broker.
Publisher
Broker
Subscriber

Fig. 1. MQTT Protocol architecture

The broker acts as the intermediary connecting the publisher and subscriber. The publisher is responsible for sending messages to the broker using a specified topic. The subscriber continuously listens to the broker for data published on a specific topic.

Figure 1 illustrates the architecture of the MQTT protocol, presenting the publisher-subscriber model. In the architecture, publishers and subscribers are unaware of each other's existence. Consequently, there is no direct communication between them, and they don't need to operate simultaneously. Operations on both sides can continue without interruption during the publishing or receiving process.

Software developers have implemented different MQTT libraries, some of which are open source, while others are proprietary [13]. Eclipse Mosquitto [14], is a widely used message broker that is part of the Eclipse Foundation [15].

III. Implementation

A. Design, Implementation, and Testing

The developed system's objective is for a local web page to provide live footage of three different cameras, simulating a typical airport surveillance camera system. Authorized persons can log into the locally maintained site to monitor different areas in the building. If passengers demonstrate suspicious behavior, the user can then report the problem to appropriate security guards or the police.

Through wireless communication, different nodes are connected in a network to pass information to each other. The blueprint for the system was constructed based on the MQTT protocol, the RTSP live video streaming protocol, and the Motion camera application.

As previously described, MQTT, developed by OASIS, constructs a wireless bridge from different machines. This protocol allows different IoT devices, general purpose or embedded, to exchange data and message each other through a publish/subscribe model [12]. Additionally, the Mosquitto software provides an efficient broker for this system.

Motion [16] is a video streaming service available in the Linux repository used to augment surveillance systems with capabilities of motion detection and video feed recording. It supports local and remote IP cameras with multiple configurations of resolutions and frames per second. In this implementation, motion service was used as a player for the collected RTSP video streams from each of the Pi cameras.

The Real-Time Streaming Protocol (RTSP) [17] allows for an efficient live stream of the camera footage on the web page as the web page code (HyperText Markup Language - HTML) would only need the stream link to display the video.

The elements of the system are as follows: a Raspberry Pi 4B [18] as the server for the broker and the web application, three camera nodes based on Raspberry Pi Zero 2 W's [19] as the subscribers and any authorized machine connected to the private network acting as the publisher. Figure 2 adequately depicts the different components attached to the Raspberry Pi Zero 2 W's that make up the camera node. The single-board computer communicates to an Arduino Nano v3 [20] through serial. Two servo motors are connected to the Arduino Nano to allow the user to adjust the angle at which the footage is being recorded on the website. The cameras used in the implementation are Raspberry Pi cameras, which are mounted with the servo motors on a 3D-printed stand.

The motive behind integrating the Arduino Nano is its internal support of a hardware clock that controls Pulse Width Modulation (PWM), removing the motors' jitteriness when directly connecting the motors to the Raspberry Pi Zero 2 W's General Purpose Input/Output (GPIO) pins.

Each Raspberry Pi Zero 2 W is equipped with a Python script that establishes serial communication with the Arduino Nano. The program also allows for subscriber connections to the MQTT broker. The Arduino Nano receives messages from the Pi Zero 2 W and shifts the angle of the camera according to the command. In addition, the Pi Zero 2 W will

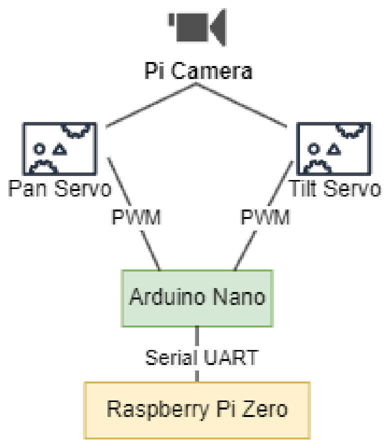
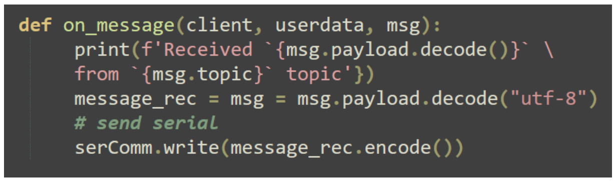


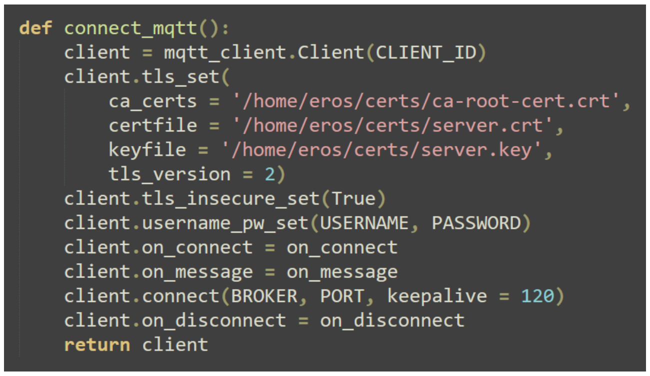
Fig. 2. Camera Node Configuration

automatically attempt to re-establish communication with the broker in case of unexpected disconnection. The functions presented in Figures 3 and 4 show the process of collecting the message from the broker and forwarding the content to the Arduino Nano for the execution of the requested command and the function that orchestrates the process of connection, communication, and disconnection.



```
def on_message(client, userdata, msg):
    print(f'Received `{msg.payload.decode()}` \
    from `{msg.topic}` topic'})
    message_rec = msg = msg.payload.decode("utf-8")
    # send serial
    serComm.write(message_rec.encode())
```

Fig. 3. Function on-message executed by the Pi Zero 2 W



```
def connect_mqtt():
    client = mqtt_client.Client(CLIENT_ID)
    client.tls_set(
        ca_certs = '/home/eros/certs/ca-root-cert.crt',
        certfile = '/home/eros/certs/server.crt',
        keyfile = '/home/eros/certs/server.key',
        tls_version = 2)
    client.tls_insecure_set(True)
    client.username_pw_set(USERNAME, PASSWORD)
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(BROKER, PORT, keepalive = 120)
    client.on_disconnect = on_disconnect
    return client
```

Fig. 4. Orchestrational function for MQTT messaging operations

The HTML code on the web page consists of displaying each of the camera's video footage as well as providing the ability to shift the recording angle of the cameras. The video is shown through iframes embedded with the respective RTSP stream links. When clicked, the buttons that control the movement of the cameras execute a PHP script that identifies which node is to be rotated, as well as its direction of motion. After identifying the behavior following the button press,

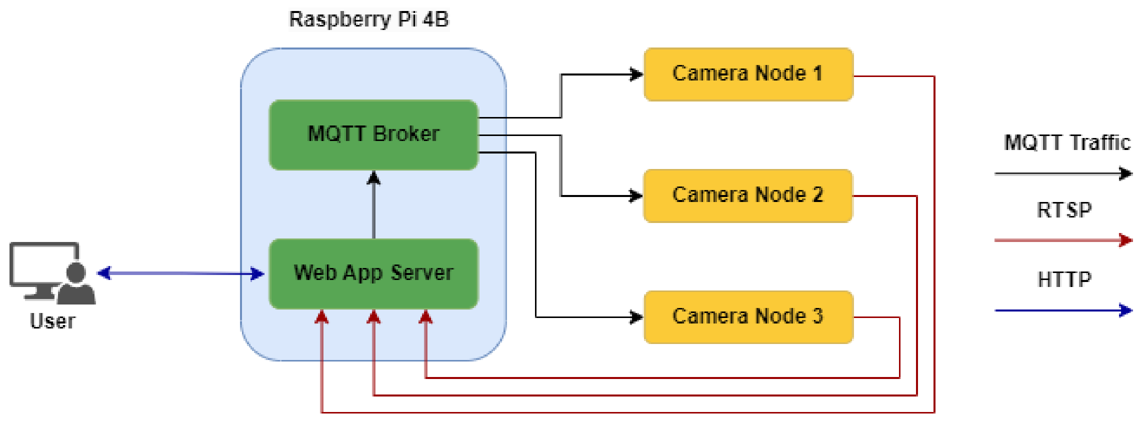


Fig. 5. Testbed System Architecture

the PHP code executes a shell command that publishes a Mosquitto MQTT message to the respective subscriber.

Figure 5 presents the Testbed System Architecture described previously in this section.

B. Threat Model

Using OWASP's Risk Rating Methodology, potential vulnerabilities were identified. A series of steps are done in the development of a threat model.

The first step is the identification of the vulnerabilities in the system. Table I lists these vulnerabilities, their exploits, as well as the resulting security impact on the system including what disruptions they can cause as per the Confidentiality, Integrity, and Availability (CIA) triad. This table covers all aspects of the system from MQTT transmissions to webpage exploitation to even physical destruction.

The second step is to rate the seriousness of these vulnerabilities based on factors such as how aware an attacker could discover that could penetrate the system and cover their tracks as well as the technical impacts on the cybersecurity standard of the CIA triad.

The result of this analysis shows that many of these attacks are not only trivial to perform but can also have a devastating impact on the system's functionality.

Table II analyzes the various factors of each vulnerability and assigns numerical values representing their overall severity. These numbers range from zero to nine, with the former meaning it is a non-factor and the latter indicating that it is a critical issue. Most of the vulnerabilities are easy for an attacker to discover and take advantage of; however, they tend to vary in how easy it would be for the site administrator to detect the problem. For instance, Distributed Denial of Service (DDoS) attacks would be immediately noticed since they would take down the system. Conversely, it would be almost impossible to detect an attacker passively sniffing network traffic.

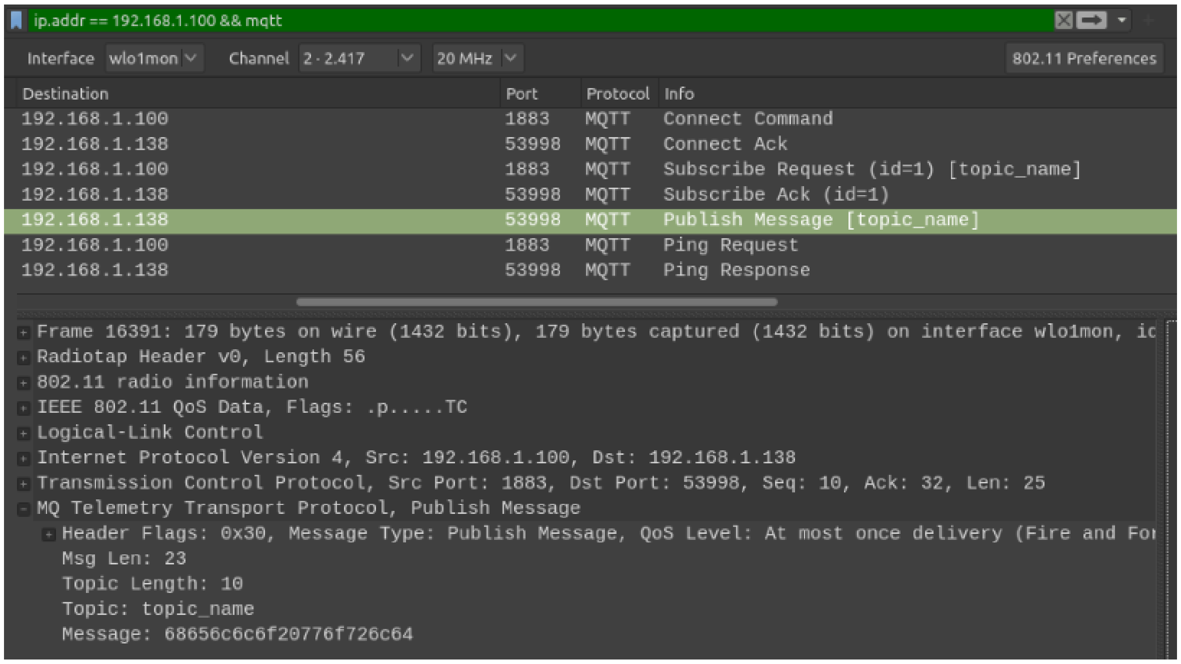
Table III presents a more granular look at the technical impacts of the CIA triad resulting from a successful exploitation of the presented vulnerabilities. Once again, the threats are labeled numerically to represent their level of severity. What is interesting to note is that each of them will result in extensive damage to at least one of the three main categories.

C. Vulnerability Assessment

An Nmap [21] network scan performed on the simulated testbed revealed the listening ports on the system including the services and versions on these ports. Easily accessible reconnaissance tools such as this can provide critical network topology and service information, which can then be used to compromise the system. For instance, this scan revealed that the Raspberry Pi 4B was listening on network port 1883, which is commonly used for insecure MQTT transmissions.

By default, MQTT transmissions are unencrypted. The program Wireshark [22] was used to perform packet analysis on the network traffic during communication between the Raspberry devices. This revealed information such as subscriber and publisher Internet Protocol (IP) addresses with their associated hardware addresses, transmission topics, and even transmission message content, as shown in Figure 6. Attackers can use this information to not only map the layout of the network but also reconfigure their own devices to spoof legitimate ones. This can be used to interrupt communication between the MQTT broker and clients and even send unauthorized commands. Even when authentication measures are used, such as usernames and passwords, this information can also be intercepted and decoded by an attacker, thus making those security measures practically useless without properly configuring Secure Socket Layer/Transmission Layer Security (SSL/TLS) encryption standards.

Experiments with Wireshark also revealed security issues when live-streaming video over the air. Captured packets could be reassembled locally on the attacking machine to recreate images from the Raspberry Pi Zero 2 W cameras. Again, SSL/TLS encryption as well as integrated access control lists can be used to remedy these issues.



```
Destination                 Port   Protocol Info
192.168.1.100               1883   MQTT     Connect Command
192.168.1.138               53998  MQTT     Connect Ack
192.168.1.100               1883   MQTT     Subscribe Request (id=1) [topic_name]
192.168.1.138               53998  MQTT     Subscribe Ack (id=1)
192.168.1.138               53998  MQTT     Publish Message [topic_name]
192.168.1.100               1883   MQTT     Ping Request
192.168.1.138               53998  MQTT     Ping Response

Frame 16391: 179 bytes on wire (1432 bits), 179 bytes captured (1432 bits) on interface wlo1mon, ic
Radiotap Header v0, Length 56
802.11 radio information
IEEE 802.11 QoS Data, Flags: .p.....TC
Logical-Link Control
Internet Protocol Version 4, Src: 192.168.1.100, Dst: 192.168.1.138
Transmission Control Protocol, Src Port: 1883, Dst Port: 53998, Seq: 10, Ack: 32, Len: 25
MQ Telemetry Transport Protocol, Publish Message
  Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Fo
  Msg Len: 23
  Topic Length: 10
  Topic: topic_name
  Message: 68656c6c6f20776f726c64
```

Fig. 6. MQTT Packet Analysis in Wireshark

IV. Discussion and Recommendations

Today, IoT-based systems are being used in several applications that traditionally were not Internet/Network connected, such as surveillance, luggage scanning, ticketing, security checkpoints, and control systems. With the inclusion of devices with constraints in the level of power, processing, and memory devices, new challenges arise in the line of management and security.

MQTT has been implemented and evaluated in this testbed, but other protocols commonly used on IoT systems would be evaluated to have a more holistic perspective in terms of cybersecurity threats and their defenses.

In the case of video surveillance systems, there is more than one point of vulnerability that can be possibly exploited. During the experience designing and implementing this testbed, more vulnerabilities had to be added to the list every time more capabilities were added to the system.

Combining or extending technologies within the same IoT implementation not only adds more functionality to the system but also creates challenges for integration and compatibility. If these challenges are not addressed adequately, security weak points will be added to the system. For instance, by adding to the security cameras capabilities for pan and tilt, an additional microcontroller with support of a hardware clock to control the Pulse Width Modulation (PWM) had to be included. With this new element, more ports and peripherals had to be disabled and protected due to the inflated attack surface.

After analyzing the results of the vulnerability assessment, some recommendations for securing IoT-based systems are:

- Use air-gapped networks that do not have access to the Internet
- Close ports and services that are not being used
- Use encryption mechanisms to store or transmit data
- Use MQTT protocol with authentication and encryption
- Use authentication methods for accessing the servers
- Use SSL/TLS protocol for certificate validation
- Use recognized and trusted manufacturers of IoT devices
- Only allow authorized people to physically access the devices
- Implement logging mechanisms to identify attacks and errors
- Include anti-malware software when it is possible
- Include security policies and access controls lists (ACL) for access permissions to critical control systems and programs

V. Conclusions and Future work

This work identified elements that provide insight into the future path of this research, including different methods for building the system and identifying vulnerabilities that, when addressed, provide system architects with reliable defenses in the realm of cyberspace.

With the implementation of a simulated camera network system manipulated through a web page, several major attack vectors have been identified as possible routes for exploitation. On each level of the system, from network communication to application configuration, it is necessary to ensure that proper steps are taken to secure the system from malicious actors.

Future work includes running a set of attacks and collecting data about the impact that the attacks had on the IoT system. Based on this information propose a securing methodology that helps developers to deploy more secure IoT systems.

TABLE I
VULNERABILITY IDENTIFICATION

VULNERABILITY	EXPLOIT	IMPACT
[A] Limited bandwidth for video streaming	DoS and DDos	Loss of availability and Financial extortion
[B] MQTT: Cleartext transmissions	Wireless sniffing, Evil subscriber and Credential stuffing to breach other systems	Loss of confidentiality
[C] MQTT: Lack of nonrepudiation mechanisms	Evil publisher, Main-In-The-Middle (MITM), Remote code execution, Malware injection, and Ransomware/Encryptors	Loss of message integrity, Loss of availability, Starting point for movement across the network, Financial extortion, and Total system compromised
[D] Motion: remote control port enabled	Remote code execution (motion user privileges), Unauthenticated access, Ransomware/Encryptors, and Malware injection	Loss of confidentiality, Loss of availability, Starting point for movement across the network, Financial extortion, Total system compromised, and Remote Control Camera movement
[E] Motion: SSL/TLS security disabled by default	Wireless sniffing to intercept video feed	Loss of confidentiality
[F] Motion: Stream authentication disabled by default	Evil client and Unauthorized system access	Loss of confidentiality and Loss of availability
[G] Motion: Static IPs of hosts/ clients	DoS and DDoS	Loss of availability and Financial Extortion
[H] Raspberry Pi: Lack of Operating System hardening/user management policies	Password cracking, Credential Stuffing to breach other systems, Privilege escalation, Total system compromised	Lack of availability, Lack of integrity, Lack of confidentiality, Financial extortion, and Starting point for movement across network
[I] Custom webpage for camera infrastructure	XSS Attacks, Code/malware injection, Session hijacking, and User impersonation	Loss of confidentiality
[J] Limited Memory	Pi Zero 2 W overloaded	Can slow/shutdown processes
[K] Physical Damage	Movement of servo motor throws off position threshold	Damaged or crippled camera movement, resulting in Lack of Availability

TABLE II
VULNERABILITY FACTORS

VULN	DISCOVERABILITY	EXPLOITABILITY	AWARENESS	INTRUSION DETECTION
[A]	7, easy	9, automated tools available	9, public knowledge	0, active detection
[B]	6, easy	9, trivial	8, obvious	9, passive, not logged
[C]	6, easy	9, trivial	8, obvious	7, likely logged without review
[D]	9, port scan	9, trivial	8, obvious	5, logged and reviewed
[E]	6, easy	9, automated tools available	8, obvious	9, passive, not logged
[F]	6, easy	9, trivial	8, obvious	7, likely logged without review
[G]	8, trivial	9, automated tools available	8, obvious	0, active detection
[H]	8, easy	9, trivial	8, obvious	5, logged and reviewed
[I]	5, easy	9, trivial	8, obvious	7, likely logged without review

TABLE III
TECHNICAL IMPACT FACTORS

VULN	CONFIDENTIALITY	INTEGRITY	AVAILABILTIY	ACCOUNTABILITY
[A]	0, no data disclosed	0, no data corrupted	9, all services lost	5, possibly traceable
[B]	7, extensive critical data disclosed	0, no data corrupted	0, no services interrupted	7, hard to trace
[C]	7, extensive critical data disclosed	7, extensive corrupted data	7, extensive primary services interrupted	1, traceable
[D]	8, extensive critical data disclosed	8, extensive corrupted data	8, extensive primary services interrupted	1, traceable
[E]	8, extensive critical data disclosed	0, no data corrupted	0, no services interrupted	7, hard to trace
[F]	7, extensive critical data disclosed	0, no data corrupted	7, extensive primary services interrupted	3, possibly traceable
[G]	0, no data disclosed	0, no data corrupted	9, all services lost	5, possibly traceable
[H]	9, all data disclosed	9, all data corrupt	9, all services lost	1, traceable
[I]	7, extensive critical data disclosed	5, some data corrupt	5, some services compromised	3, possibly traceable

REFERENCES

[1] C. Rouland, ”A Random Walk Through a BILLION Things: A conversation with millions of researched Cyber-Physical Systems. *Keynote speaker - IoT Security and Cyber Threat Intelligence (IoT SCTI) - ACSAC 2023 Conference*. Austin - Texas, USA, December 4, 2023.

[2] OWASP, ”Explore the world of cyber security”. (Online): https://owasp.org/ [Last Accessed: January 13, 2024]

[3] E. Elbasi, ”Reliable abnormal event detection from IoT surveillance systems”, *In proceedings of the 2020 7th International Conference on Internet of Things: Systems, Management, and Security (IOTSMS)*, 2020, Paris, France. DOI: 10.1109/IOTSMS52051.2020.9340162

[4] R. Ke and Y. Zhuang and Z. Pu and Y. Wang, ”A Smart, Efficient, and Reliable Parking Surveillance System With Edge Artificial Intelligence on IoT Devices”, *IEEE Transactions on Intelligent Transportation Systems*, volume 22, issue 8, Aug 2021. DOI: 10.1109/TITS.2020.2984197

[5] V. Bui, A. Alaei, and M. Bui, ”On the Integration of AI and IoT Systems: A Case Study of Airport Smart Parking”. *Internet of Things book series*. January 2022. (ITTCC)DOI: 10.1007/978-3-030-87059-1_16

[6] A. F. Santamaria, P. Raimondo, M. Tropea, F. De Rango, and C. Aiello, ”An IoT Surveillance System Based on a Decentralised Architecture”. *Sensors*, volume 19, issue 6, 2019. DOI: 10.3390/s19061469

[7] CSA - Connectivity Standards Alliance - Matter - The Foundation for Connected Things, (Online): https://csa-iot.org/all-solutions/matter/ [Last Accessed: January 13, 2024]

[8] IEEE SA - IEEE Standard for an Architectural Framework for the Internet of Things (IoT)”, (Online): https://standards.ieee.org/ieee/2413/6226/ [Last Accessed: January 13, 2024]

[9] Thread - Thread Certified Products, (Online): https://www.threadgroup.org/What-is-Thread/Thread-Benefits [Last Accessed: January 13, 2024]

[10] CSA - Connectivity Standards Alliance - Zigbee - The full stack Solution for All Smart Devices, (Online): https://csa-iot.org/all-solutions/zigbee/ [Last Accessed: January 13, 2024]

[11] ZWave - Smart home controls on one app, (Online): https://www.z-wave.com/” [Last Accessed: January 13, 2024]

[12] HiveMQ - MQTT Essentials, The Ultimate Guide to MQTT for Beginners, and Experts, (Online): https://www.hivemq.com/mqtt-essentials/ [Last Accessed: January 13, 2024]

[13] HiveMQ - Reliable Data Movement for Connected Devices, (Online): https://www.hivemq.com/ [Last Accessed: January 13, 2024]

[14] Mosquitto.org - Eclipse Mosquitto An open source MQTT broker, (Online): https://mosquitto.org/” [Last Accessed: January 13, 2024]

[15] Eclipse Foundation - About us, (Online): https://www.eclipse.org/org/foundation/ [Last Accessed: January 13, 2024]

[16] Motion Configuration, (Online): https://motion-project.github.io/motion_config.html [Last Accessed: January 13, 2024]

[17] Internet Engineering Task Force, ”Real Time Streaming Protocol (RTSP)”. (Online): https://www.ietf.org/rfc/rfc2326.txt [Last Accessed: January 13, 2024]

[18] Raspberry Pi 4 Model B, (Online): https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ [Last Accessed: January 13, 2024]

[19] Raspberry Pi Zero 2 W (Online): https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/ [Last Accessed: January 13, 2024]

[20] Arduino Nano, (Online): https://store-usa.arduino.cc/products/arduino-nano [Last Accessed: January 13, 2024]

[21] Nmap (Online): https://nmap.org/ [Last Accessed: January 13, 2024]

[22] Wireshark - The world’s most popular network protocol analyzer (Online): https://www.wireshark.org/ [Last Accessed: January 13, 2024]