JAIST Repository
https://dspace.jaist.ac.jp/

Title	Smart Building Control System Emulation Platform for Security Testing
Author(s)	翁, 暁琪
Citation	
Issue Date	2025-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/19784
Rights	
Description	Supervisor: BEURAN, Razvan Florin, 先端科学技術研究科, 修士 (情報科学)

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Master's Thesis

Smart Building Control System Emulation Platform for Security Testing

Xiaoqi Weng

Supervisor Razvan Beuran

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

March, 2025

Abstract

Smart buildings play a critical role in advancing the smartness of cities. With the continuous development of technology, the application of technologies such as automated control, smart sensors and communication networks in smart buildings is becoming increasingly complex. Smart buildings realize efficient management of equipment in buildings by integrating advanced automation control systems. At the same time, the application of smart sensors, elevators, robots and other devices enable buildings to respond to environmental changes in real time, optimize energy, reduce resource waste and bring much convenience to people's lives. However, in order to ensure the effectiveness and reliability of these technologies, continuous testing and system upgrades are required.

Testing and evaluating these technologies using real building environments often faces a number of challenges, such as the accompanying high testing costs, the risk of system crashes, limited resources, and lack of adaptability. In addition, the communication network, as the nerve center of smart buildings, ensures the control and information exchange between devices. However, along with the development of cyberattack techniques and the acceleration of digital transformation, the security threats to networks in smart buildings are increasing. Cyber attacks against smart buildings may threaten the data security and privacy of users, as well as cause damage to physical equipment in the building leading to disruption of operational services. These threats not only cause economic losses to building operators, but also have an impact on corporate reputation.

To address these challenges, we created the Smart Building Control System Emulator (SBCSE). SBCSE is a platform that emulates smart building control systems and was designed and developed based on real smart building log data. SBCSE simulates control systems and the movement of IoT devices such as robots and elevators, and uses an implementation of actual communication protocols, thus enabling emulation testing and evaluation of control systems and information interactions between devices in smart buildings. SBCSE also supports the emulation of various security scenarios, and analyzes and proposes countermeasures against potential threats, and verifies the effectiveness of the security measures.

Through this research, we aim to propose an innovative solution to the short-

comings of the current simulation platform for smart building control systems. By developing an smart building emulator, it provides an effective tool for testing and evaluating smart building systems. The simulator can not only provide users with an intuitive user interface and simplify the operation process, but also better help users conduct tests in different scenarios and help improve the reliability and performance of smart building systems. In this way, we attempt to solve the problem of the current lack of control system simulation platform for smart buildings. At the same time, for the network security problems existing in smart buildings, this study has specially designed a security test module, which can emulate a variety of attack scenarios and conduct a comprehensive security test. The module helps to identify potential security vulnerabilities in the smart building system and timely detect risks that may lead to system crash or data leakage. We have also analyzed relevant security measures that can be used to cope with the risks, which can provide effective protection solutions for the system. Through these simulations and tests, we can help designers and developers to identify and fix the security risks in the system, so as to avoid unnecessary losses in practical applications.

SBCSE not only improves the efficiency and security of smart building system testing, but also significantly reduces testing costs and provides powerful support for system design, optimization and maintenance. In addition, by emulating various cybersecurity attack scenarios, our security model helps to test and develop effective countermeasures against potential risks in smart building systems. SBCSE is a versatile and cost-effective tool for improving the reliability and security of smart buildings that facilitates the testing and application of new technologies in the growing field of smart cities.

Keywords: smart building, control system, security testing, emulation, simulation, MQTT.

Acknowledgment

First, I would like to express my sincere gratitude to my supervisor, Associate Professor Razvan Beuran. Although the two years have flown by in the blink of an eye, during this precious time, he has given me wholehearted guidance and support in both academic research and career activities. From my mentor, I have not only learned a wealth of knowledge and academic skills, but more importantly, I have learned how to pursue research with passion and persistence. This will forever inspire me to keep moving forward on my future academic path.

I would also like to express my special thanks to my second supervisor, Professor TAN Yasuo, for the valuable opinions and careful revisions he provided during the writing of my research proposal. His professional guidance helped me clarify my research ideas, allowing my research work to delve into deeper contemplation.

At the same time, I extend my sincere thanks to all the professors at JAIST. Thank you for creating an academic environment full of nurturing and support, in which I have thrived and made continuous progress. I would also like to express my deep gratitude to all members of the Beuran Laboratory. Their opinions and help have greatly facilitated the progress of my research. The spirit of collaboration and the atmosphere of knowledge sharing in the laboratory have not only enhanced my passion for research but also greatly expanded my professional horizons. Lastly, I would like to extend my deepest thanks to my family and friends. Thank you for your unconditional support and encouragement.

Finally, I once again express my heartfelt thanks to everyone who has helped and supported me.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Definition
BAC	Broken Access Control
BACS	Building Automation and Control System
BACnet	Building Automation and Control Network
BOS	Building Operating System
CMD	Command
DDoS	Distributed Denial of Service
ELV	Elevator
IoT	Internet of Things
MA	Malware
MITM	Man-in-the-Middle
MQTT	Message Queuing Telemetry Transport
MQTTS	Message Queuing Telemetry Transport over TLS
QoS	Quality of Service
RPF	Robot Platform
ROB	Robot
SBCSE	Smart Building Control System Emulator
SEA	Social Engineering Attack

Chapter 1

Introduction

This chapter provides an overview of the research context, motivations, significance, and structure of this thesis. Section 1.1 discusses the background of this research and highlighting its importance and potential challenges. Section 1.2 outlines the research questions we intend to address in order to define the objectives of the study. In Section 1.3, we summarize the significance and importance of this research. Finally, Section 1.4 describes the organization of the thesis.

1.1 Background

According to the report by the Association for Smarter Homes & Buildings (ASHB) on smart building technologies and market trends, the smart building industry is expected to continue growing in the coming years, primarily driven by IoT, automation systems, and environmental sustainability goals [1]. With the rapid development of the IoT technology, the concepts of smart city and smart home have emerged one after another. Smart buildings, as an important part of the smart city, have a pivotal role in the field of infrastructure in the future city. The functions and application equipment of smart buildings are also expanding at this stage.

In regard with smart buildings, the application of Building Automation Systems (BAS) [2], Building Management Systems (BMS) [3], smart sensors, and IoT devices is becoming increasingly complex. These technologies enable buildings to improve their operational efficiency by enabling device management and information integration through digital platforms. This not only helps building operators to efficiently integrate and analyze data within the building, but also to efficiently control the equipment in the building (e.g., elevators, robots, etc.) using automation systems. This integration not only optimizes the management of building equipment and improves operation and maintenance efficiency, but also saves en-

ergy to accelerate digital transformation and sustainable development, bringing convenience to people's production and life.

However, the effectiveness and reliability of these technologies and control systems depend on continuous improvement and testing. In contrast, evaluating these technologies in real-world building environments faces numerous challenges, including high costs, significant risks, limited resources, and insufficient adaptability. In this context, simulation tools and platforms offer a more flexible, cost-effective, and efficient solution, providing a reliable testing environment for technology development and validation.

In addition, with the acceleration of digital transformation, the need for communication and data interaction through networks has increased, and more and more devices are connected to networks,. However, the development of network technology is also accompanied by an increase in network vulnerabilities and cyber attacks. IoT devices and control systems in smart buildings are also becoming a key target for cyber attacks. These systems may face a variety of cybersecurity risks including data leakage, malicious activities, eavesdropping, and sabotage of IoT devices. According to a report by Kaspersky in the first half of 2019, 37.8% of computer systems used to control smart buildings suffered some form of malicious attack [4], which highlights the severity of this problem.

For operators of smart buildings, the security and privacy protection of buildings have become crucial issues that cannot be overlooked. The cybersecurity challenges faced by smart buildings are immense, making it particularly important to conduct comprehensive security testing and countermeasure analysis for smart buildings.

1.2 Objectives

This research aims to address the current lack of control system emulation platform for smart buildings and the cyber security challenges faced by smart buildings. The main objectives include:

- Design and implement a emulator for smart building control systems that integrates management systems. We will focus on two components: the network communication module and the device motion module. The IoT devices to be utilized include robots and elevators. This emulator will serve as a testing platform for smart buildings, facilitating configuration, information sharing, and management among various intelligent systems within the building.

- Add a security module to emulate attack scenarios. This module will emulate attack scenarios targeting the risks faced by the current components and

analyze the impact of these attacks to propose effective solutions.

1.3 Significance and Contributions

We propose an innovative solution to the shortcomings of the existing smart building control system simulation platforms. By developing the smart building control system emulator, we fill the gap of controllable emulation platform in the field of smart buildings. The emulator simplifies operation through an intuitive interface and supports multi-scenario testing, providing a powerful tool for smart building system testing and evaluation. In addition, with the rapid development of smart building technology, network security has become a major challenge facing intelligent buildings. The security testing module designed in this study can emulate multiple attack scenarios, identify security risks as well as visualize the impact through simulated data, and provide effective security measures against attacks.

The main contributions of this thesis are as follows. We have developed a smart building emulator that can emulate the network communication within smart building systems, as well as emulate the smart building control systems, and simulate the behavior of robots and elevators under various operating conditions. This tool provides an effective platform for testing and evaluating smart building systems, and helps to verify the performance and reliability of the system prior to actual deployment. At the same time, by providing an intuitive user interface and simplified operational processes, our emulator makes it easy for non-specialized users to test and evaluate smart building systems, thereby lowering the technical barrier and expanding the scope of smart building technology applications.

Moreover, in response to the growing cybersecurity threats to smart building systems, the emulator is able to emulate a number of different attack scenarios, enabling users to test the performance of intelligent building systems under different conditions in a secure environment, which is critical for system design optimization and performance enhancement. We not only design and implement attack scenarios for potential security risks, but also provide countermeasures for the risks and test the effectiveness to enhance the security of smart building systems. Testing in a emulator reduces the risk of system crashes and data leakage that security testing in real building systems may face.

Our emulator facilitates collaboration across multiple disciplines in architecture, IoT, network communications, cybersecurity, and other areas for the development of smart building control systems. It can also be used as an educational and training tool to help students and those preparing to enter the field to better understand the complexity of smart building systems. Meanwhile, in the field of cyber security for smart buildings, researchers can customize our emulator to add multiple cyber attack scenarios for testing and simulate the proposed security

countermeasures against the countermeasure table. It provides an effective testing tool for security management and maintenance of intelligent buildings.

1.4 Organization

The remainder of this thesis is organized as follows:

- **Chapter 2: Related Work and Background**
  This chapter focuses on the research related to simulators for smart buildings and analyzes their limitations. Additionally, we review previous studies in the field of network security to provide theoretical support for this research. Relevant related theories used in this study is also introduced.

- **Chapter 3: Proposed System**
  This chapter provides a detailed description of the basic architecture and design background of the smart building control emulator developed in this study. We systematically explain the design principles and methods used in the development of the architecture. Furthermore, the chapter includes an in-depth analysis of each functional module.

- **Chapter 4: SBCSE Assessment**
  This chapter evaluates the proposed system, with a focus on the evaluation methods and approaches. It also includes a detailed analysis and comparison of the data collected from the system's logs to verify its effectiveness and performance.

- **Chapter 5: Attack Scenarios and Proposed Security Measures**
  This chapter will provide a detailed introduction to the security testing scenarios designed in the emulator environment, as well as the measures taken to ensure the security of the testing. This includes conducting security analysis of system components, designing potential attack scenarios, and formulating security measures tailored to the risks identified within the components.

- **Chapter 6: Experiment Results**
  This chapter will conduct experiments on the designed security scenarios and analyze the experimental results. We will mainly discuss MITM, DDoS, and BAC attacks. We will select corresponding security measures for each attack, apply these measures on the emulator, and then conduct attack tests again to verify the effectiveness of these security measures.

- **Chapter 7: Conclusion**
  This chapter summarizes the main conclusions of this study. It also identifies

the limitations of the research and proposes potential improvements and directions for future work, offering insights for further exploration.

Chapter 2

Related Work and Background

In this chapter we will first introduce the prior studies relevant to this study and explore the limitations of these studies. At the same time, we will outline the background knowledge related to this study. In Section 2.1, we will introduce existing simulators and research related to network security. In Section 2.2, we will present the background knowledge and associated theories of this study.

2.1 Related Work

2.1.1 Simulators

There have been a number of simulator-related projects proposed and widely used in education, infrastructure, industrial production and other fields. Compared with testing in real systems, simulators can conduct experiments and validations with lower cost, higher efficiency and greater flexibility. Researchers can use simulators to validate new algorithms and models in a controlled environment, and predict system performance to support further optimization and improvement.

With the development of smart buildings, simulators have become an indispensable tool in this field. However, existing research has mainly focused on smart home scenarios. These simulators are usually used to collect and analyze indoor environmental data, such as temperature, humidity, and light, to optimize the living environment. For instance, Open-SBS [5] is a cross-platform open-source smart building simulator that simulates complex indoor environments, collects and representative datasets, and provides the ability to save and share experimental models. However, the research of this simulator mainly focuses on data collection and analysis of the scenarios of smart homes, and the data collection from sensors is unidirectional, and does not delve into the simulation of the interaction between smart building operating systems and IoT devices.

Apart from the smart home domain, simulation studies on other aspects of smart buildings include: integrated simulation frameworks for smart grid systems, such as GridLAB-D [6] and HVAC simulations [7]. These simulators primarily focus on energy management and system performance optimization, aiming to simulate and optimize power and energy management systems. Additionally, IoT-related simulators, such as MobIoTSim [8], assist developers in examining the behavior of IoT systems and enabling more efficient development and evaluation of IoT applications.

As shown in Table 2.1, we have compared the functionality of these simulators with SBCSE. And the parts marked with an asterisk (*) mainly represent contributions from my lab collaborators. The comparison reveals that different simulators have their own features and advantages in terms of functionality and performance. Open-SBS is simulated based on a semantic rule engine, and the building control system here mainly targets the interaction between sensors in a smart home. The closest system to ours is MobIoTSim, but the control system here is mainly for IoT devices. SBCSE not only involves the simulation of IoT devices, but also focuses on the interaction of smart building control systems and robot elevators, and has been tested for safety. Specifically, SBCSE is able to simulate the interaction of various devices and sensors in a smart building control system, as well as the operation and scheduling of a robotic elevator. In terms of safety, our simulator tests and proposes countermeasures for a variety of safety scenarios.

Table 2.1: Simulator functionality comparison

Function	Simulator	Open-SBS	GridLAB-D	HVAC simulations	MobIoTSim	SBCSE
Environment & Resource Simulation	Physical Environment (Temperature, Humidity, Lighting, Wind Speed, etc.)	✓	✓	✓		
	Energy Control System (Electricity, Air Conditioning, Power Grid, etc.)	✓	✓	✓		
Behavior & Interaction Simulation	Human Behavior	✓	✓			
	Time Simulation	✓	✓	✓	✓	✓
	Network Communication Simulation/Emulation	✓			✓	✓
	Sensors / IoT Devices	✓	✓	✓	✓	✓
	Device Behavior Interaction	✓	✓	✓	✓	✓
	Building Control System	✓				✓
Technical & Operational Support	Protocol Support				✓	✓
	Multi-instance Support		✓		✓	✓
	User Interface Support	✓			✓	✓
	Data Analysis / Visualization	✓	✓	✓	✓	✓
	Security Testing					✓
	Security Measures					✓
	*Fuzz testing [9]					✓
	*Protocol Formal Verification [9]					✓

Although there are numerous research results, the current smart building simulators are still in the primary development stage, facing with many issues and challenges. For example, the existing simulators are deficient in integration with IoT devices and their operating systems, and cannot fully meet the complex needs

of smart building systems. Especially in terms of data interaction with smart building operating systems and IoT devices, real-time, and scalability. This study proposes an intelligent building control system simulator platform to address this aspect, which will provide more effective support for the design, verification, and optimization of smart building systems.

2.1.2 Security Frameworks

As a crucial aspect of smart city development, smart buildings offer enhanced comfort, security, and convenience to occupants. However, the proliferation of smart buildings has introduced new cybersecurity risks and challenges. The integration of numerous automated systems and IoT devices, while improving efficiency and bring convenience, also exposes more vulnerabilities, making smart buildings potential targets for cyberattacks. As the cyber threat to smart buildings is complex. Cyberattacks have been used to exploit smart building controls and breach corporate networks, cause critical building system failures [10]. Attacks against smart buildings are also increasing year after year globally.

To address these security challenges, researchers have proposed various cybersecurity frameworks and models. For instance, IoT security framework for smart cyber infrastructures [11] and an ontology-based cybersecurity framework for IoT [12] has been proposed. Despite these efforts, research on cybersecurity testing models specifically tailored for smart buildings remains relatively limited. The complexity and diversity of smart buildings necessitate a more comprehensive security strategy that encompasses not only the network layer but also the physical layer and the protection of data privacy. Therefore, further research and development of cybersecurity models suitable for smart buildings are crucial. This will not only enhance the security of smart buildings but also provide robust support for the security and stability of smart city infrastructure.

2.2 Background and Related Concepts

2.2.1 Smart Buildings

Smart buildings are modern structures that integrate Information and Communication Technology, enabling them to add intelligence to various facilities within the building, such as electrical equipment, air conditioning systems, meeting rooms, restrooms, smoking areas, and other architectural facilities, all of which can be equipped with the IoT and managed and coordinated centrally on a system [13]. Smart buildings consolidate data resources and application systems to provide people with a more intelligent and efficient lifestyle. In smart buildings, BACS (Build-

ing Automation and Control System) plays a central role. The Smart Building System Architecture Guideline [14] defines BACS as a system that integrates monitoring and control of electrical power, lighting, heating, air conditioning, drainage, sanitation, disaster prevention, and crime prevention equipment within a building. The purpose of such a system is to ensure the efficient and safe management and operation of equipment.

2.2.2 Simulation and Emulation

In this study, we primarily discuss the control systems for communication between devices and mobile operations within buildings. When discussing the control systems of smart buildings, we differentiate between simulation and emulation technologies. Simulation refers to the process of creating a model to imitate the behavior of a system or process, focusing primarily on the system's functions and outputs rather than the specific implementation of its internal structure. Emulation, is distinct in that it not only relies on models of the real world, but also replicates the system's state, behavior, and outcomes, emulating its actual operation and implementation [15].

In the system we designed, network communication and attack scenarios are achieved through emulation technology. We have emulated the communication interactions between devices through an actual network environment to test and verify the system's performance under real conditions. However, the movement of the robot and the elevator is done using simulation techniques. In this part, we focus more on the performance and output of the movement behavior rather than the details of its internal implementation. By combining simulation and emulation technologies, we can flexibly apply the technologies in different application scenarios to effectively test and optimize the control systems of smart buildings.

2.2.3 Protocol

In the field of communication for smart buildings, traditional communication protocols such as BACnet [16], which are used for communication between controllers and integrated gateways within the building, may have some shortcomings. For example, they may have limited device support, cause significant network load, or make system integration difficult.

Our system employs the MQTT protocol. MQTT is a protocol designed for many-to-many communications, transmitting messages among multiple devices through a central broker [17]. The message Publish/Subscribe process, as shown in Figure 2.1, is based on the publish/subscribe model and provides three levels of Quality of Service (QoS) to ensure the reliability of message delivery. Considering

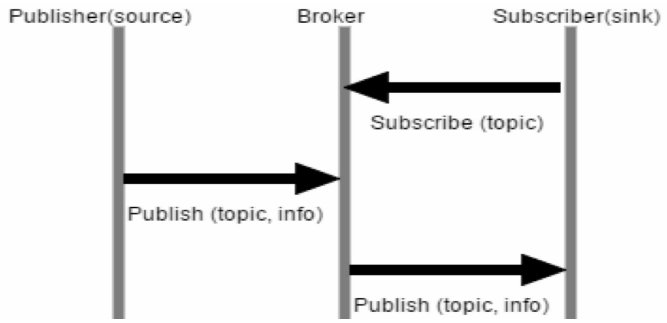

Figure 2.1: Publish/subscribe process utilized by MQTT [17].

devices with limited memory and processing capabilities, MQTT is highly suitable for Machine-to-Machine and IoT environments. It is considered one of the most effective protocols for achieving efficient connectivity and communication, particularly suitable for applications requiring low bandwidth and high reliability. Through the Broker, the MQTT protocol can achieve efficient communication between devices, maintaining stable message delivery even under unstable network conditions or when device resources are limited.

Chapter 3

Proposed System

This chapter provides detailed information about the proposed system. Section 3.1 explains the system's main components. Section 3.2 presents a brief introduction to the system's overall architecture. Section 3.3 describes the different modules within the system's basic framework and their respective functions. Finally, a comparison between SBCSE and the original building system is conducted in Section 3.4.

3.1 System Components

SBCSE was designed based on a smart building prototype from a certain construction company. Figure 3.1 shows the main components of the system: (i) Elevators (ELV); (ii) Building Operating System (BOS); (iii) Robot Platform (RPF); (iv) Robots (ROB); (v) MQTT-Broker.

BOS represents the building "operating system" that facilitates the collaboration among building equipment, IoT devices, and various applications. RPF is a robot control platform that enables remote robot control and collaborative work among multiple robots. MQTT-Broker acts as a middleware agent enabling communication between different devices and services via the MQTT protocol.

Specifically, in our system, BOS serves as the building's operating system, responsible for message forwarding and centralized management of equipment in the building. The currently connected device is the elevator. BOS handles the forwarding of messages exchanged between RPF and the elevator. The main function of RPF is to control and initiate task commands, acting as an external server to remotely control the robot. In our system, the communication protocol used is MQTT, which was introduced in Section 2.2. Message transmission and reception are achieved through subscription and publishing mechanisms.

The communication relationships among each components are shown in Figure 3.1. The black double-headed solid arrows represent the publish and subscribe

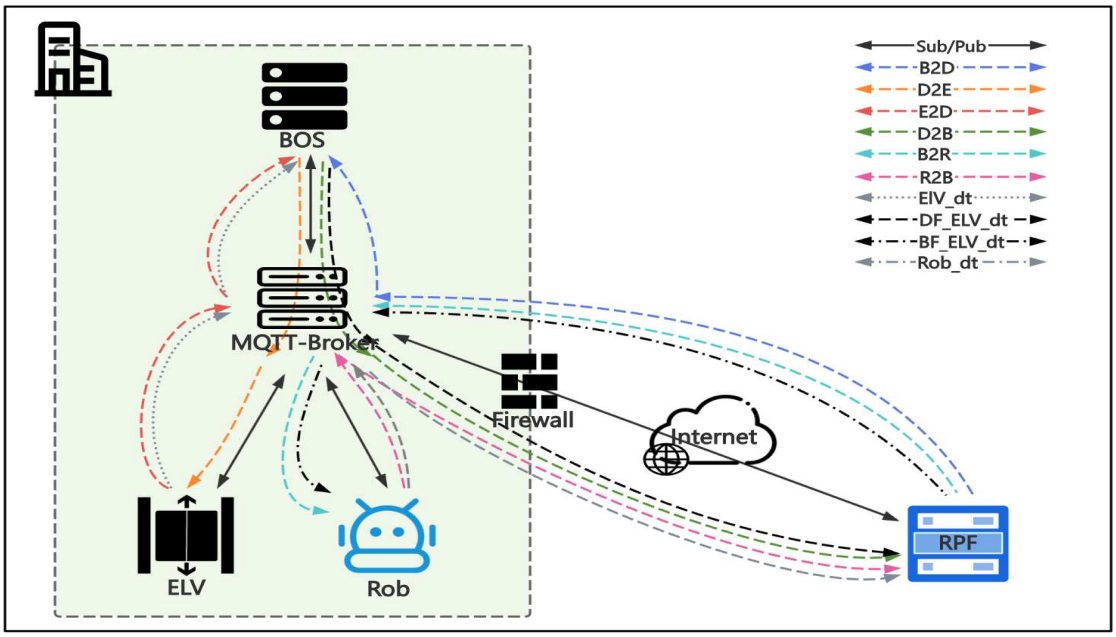

Figure 3.1: Component communication in the target smart building system.

connections established between each component and the Broker. The dashed lines indicate the communication relationships among the components and the flow of messages on various topics. Taking the message forwarded by BOS to ELV as an example, we can see the orange dotted line with "D2E" as the topic in the figure passes from BOS publish message to Broker, ELV passes subscribe the topic, Broker will forward this message to ELV, which completes one communication between BOS and ELV. The communication relationship for other messages is similar.

For a better view of how components subscribe to message topics, check out Figures 3.2 and 3.3. The upper half of Figure 3.2 illustrates the message topic subscriptions between RPF, BOS, and ELV. From top to bottom, this figure shows how messages are forwarded among different processes for the same command. The lower half of Figure 3.2 focuses on the message topic subscriptions between RPF and ROB.

In our system, there is also data related to the state of device. Specifically, the robot continuously sends its position and status data to RPF. Additionally, the elevator sends its status data to all components. Figure 3.3 displays the sequence of message forwarding processes. The first half of Figure 3.3 shows the forwarding process of the elevator's dt data, while the second half shows the forwarding process of the robot's dt data. By examining these figures, we can gain a clearer understanding of the message interactions between the components.

Meanwhile, the network topology of the components in our system is shown in

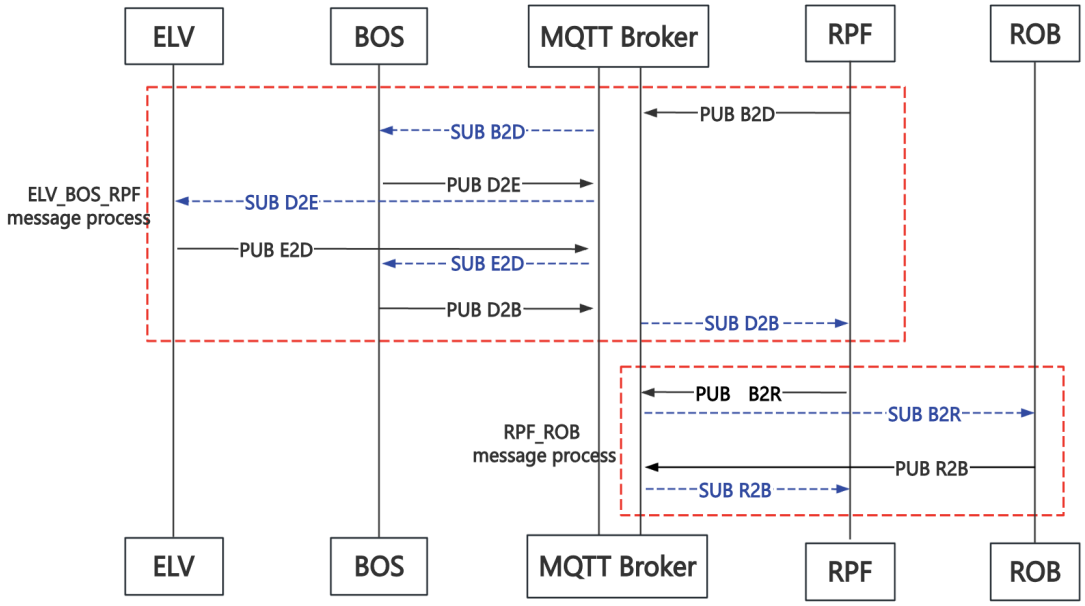

Figure 3.2: Command message subscriptions between components.

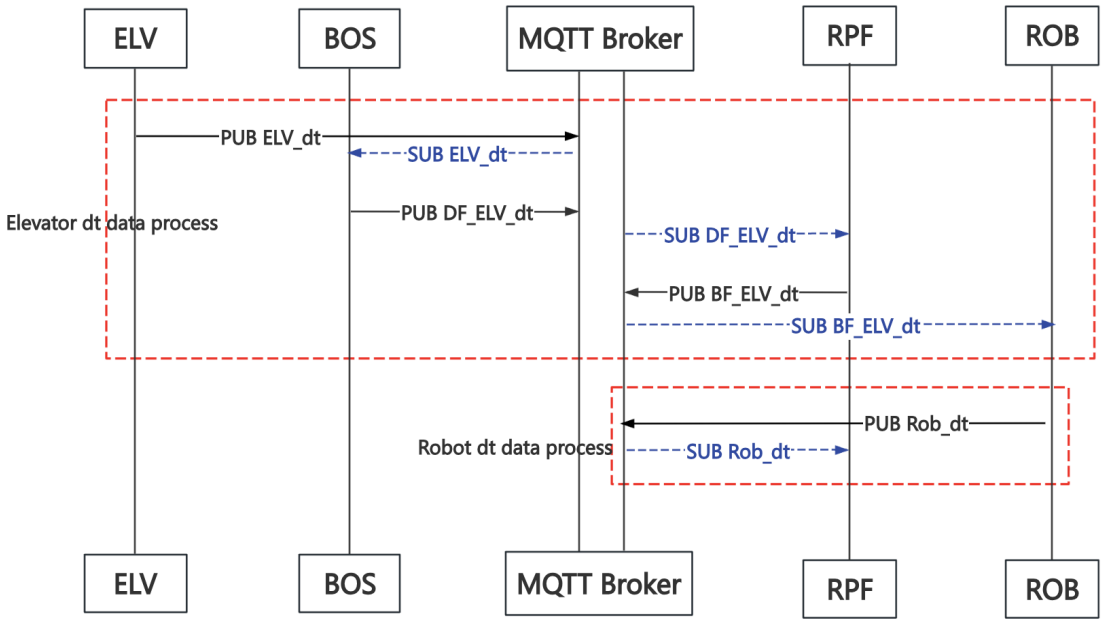

Figure 3.3: Data message subscriptions between components.

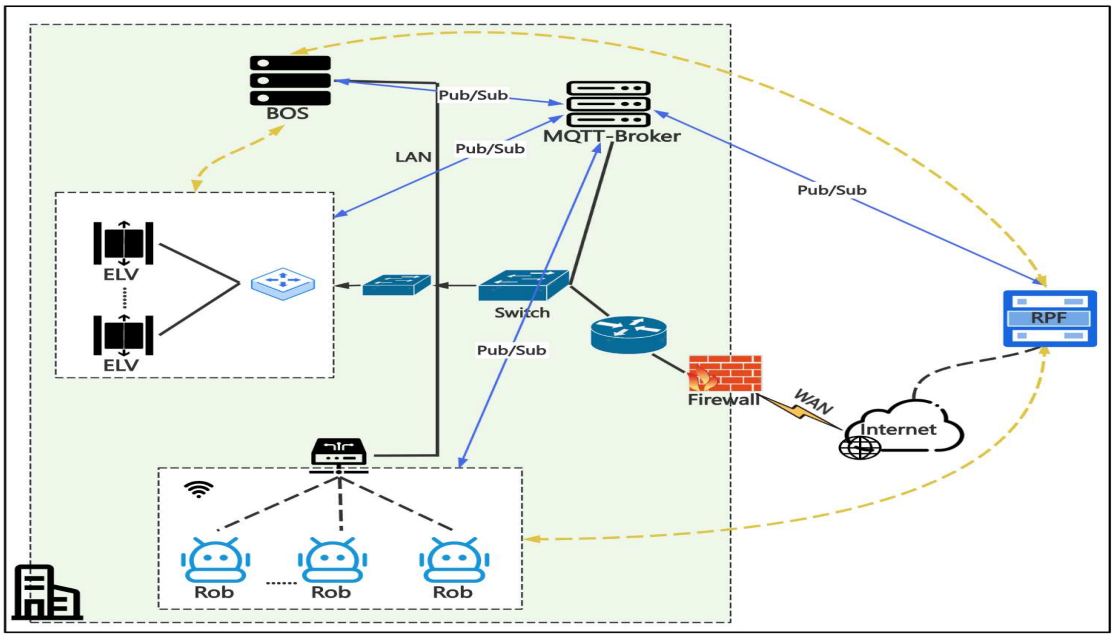

Figure 3.4: Network topology of the target system components.

Figure 3.4. The figure shows the distribution of each component in the system at the network level. The boundaries between in-building devices and out-of-building devices are clearly shown in the figure. the RPF server provides remote access and control of the ROB over the Internet. All components have a Publish/Subscribe relationship with the MQTT broker, which means that all messages are forwarded through the MQTT broker. The yellow dashed portion represents the communication relationship established between components. The RPF communicates with the BOS and ROB, and the RPF does not have a direct communication relationship with the ELV, and the message tasks need to be forwarded through the BOS. By gaining a comprehensive understanding of the functions of the components and the connection relationships between them. It is convenient for us to design for the subsequent test scenarios.

3.2 System Architecture

Based on an in-depth analysis of real building component communication logs and robot movement logs provided by the construction company, we have designed and developed an smart building control system emulation platform. The architecture of this platform is shown in Figure 3.5 and primarily consists of the following modules: a network communication module, a device motion module (the symbol

'*' is used to indicate this module was mainly developed by collaborators), an RPF control protocol module, and other supporting modules. These core modules provide the foundational simulation framework for each component. The simulator modules are responsible for simulating the components, while a central manager oversees the integration and management of all simulators.

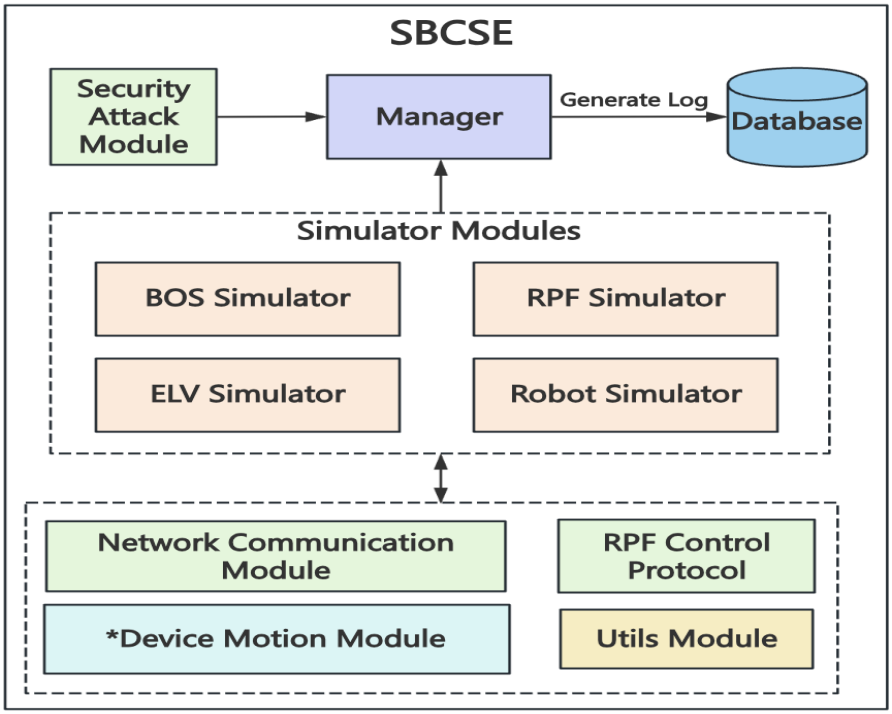

Figure 3.5: Overview of the SBCSE architecture.

After simulation ends, the generated log data is saved to a database. Additionally, we have integrated a cybersecurity attack simulation module to conduct security testing on the components. To enhance user convenience, we have also developed a user interface for the platform.

3.3 Fundamental Framework

This section will provide a detailed introduction to the Fundamental Framework of SBCSE. The modules of the emulator are mainly developed based on Python.

3.3.1 Network Communication Module

This module emulates the communication of real buildings. We use the MQTT protocol as the foundation and extend it to build our communication module. This

module is responsible for managing communication protocols, processing related data, and ensuring smooth communication throughout the system. Specifically, it includes components for defining and managing communication logic, data models, and structures. Additionally, the module handles messages between various components within the system. We have chosen the local Mosquitto broker to provide efficient and reliable message delivery services.

3.3.2 Device Motion Module

This module, primarily developed by collaborators, is used to define and configure the motion behaviors of robots and elevators. It supports the creation and management of instances for these devices. It includes defining the motion patterns and behaviors of robots and elevators, such as robot navigation routes and elevator movement between floors.

3.3.3 RPF Control Protocol

This module is responsible for controlling the communication protocol of RPF, which is designed based on the principles of automata theory. In our program, whenever a task instruction is issued, the completion of that task will follow a predetermined communication process.

Taking an example where RPF issues a command for the robot to go to the fifth floor for cleaning, the overall communication process of the system is as depicted in Figure 3.6. Initially, the system sends a command for the robot to wait in front of the elevator. Once the robot completes this action, RPF sends an "interlock" command to the elevator. The "interlock" command serves to switch the elevator to AGV mode, a dedicated mode that prevents external factors and human interference, ensuring the safe operation of door opening/closing and floor selection by the operating system. Through this command, the elevator can smoothly switch to AGV mode. Since our communication module is controlled by a state machine, the sequence of command execution is fixed, and the system can only proceed to the next command after the current command's task is completed. That is, only after the elevator completes the mode switch and returns an "interlock success" signal to RPF, will RPF send the next command "open". In this process, BOS is responsible for forwarding the communication data between the elevator and RPF.

In our system, the RPF control protocol's automaton tasks are executed in the following sequence: GoToElv, GoToElv success, Calling, Calling success, interlock, interlock success, call, call accept, call arrive, open, open success, GettingOn, GettingOn success, close, close success, go, go accept, go arrive, open, open success, GettingOff, GettingOff success, close, close success, interlock, interlock success, Schedule Work, Schedule Work success. The meanings of these commands are

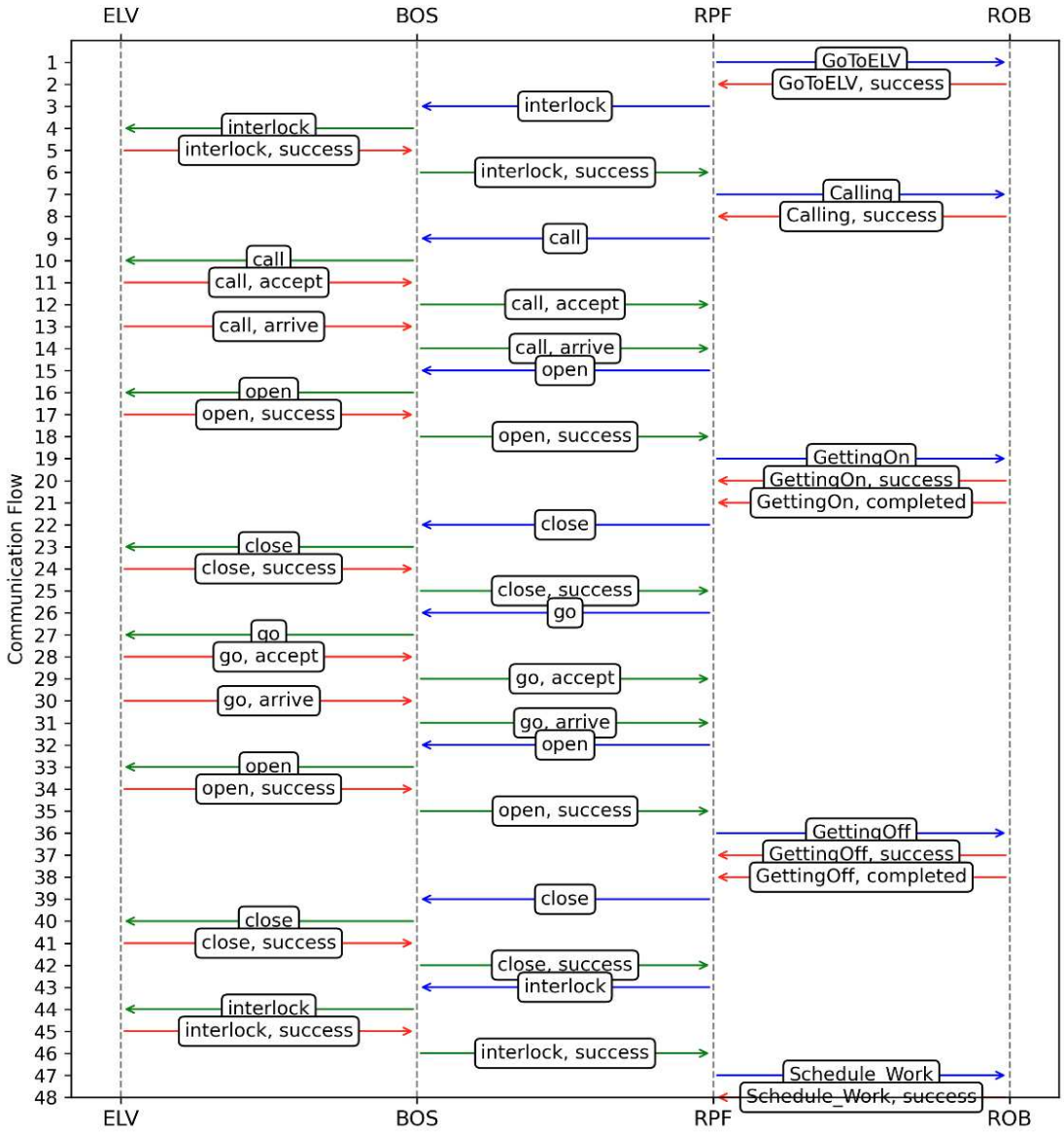

Figure 3.6: Communication process of command execution and response.

shown in Table 3.1. This sequence ensures that the interaction between the robot and the elevator is both efficient and safe.

To ensure the smooth execution of the entire process, our system also includes error detection and exception handling mechanisms. If any errors occur or if there is packet loss at any stage, the system will resend the command. When the command resending limit is reached, an error signal will be sent, and the current task will be halted. This design not only enhances the robustness of the system but also ensures the safety of robot and elevator operations. Through this finely controlled communication protocol, we can achieve precise command over the robot and elevator, thereby improving the efficiency and reliability of the entire system.

Table 3.1: RPF control protocol commands and their meaning

Command	**Meaning**
`\GoToElv`	Call the robot to wait in front of the elevator.
`\GoToElv success`	Robot successfully arrives.
`\interlock`	Enable or disable AGV mode.
`\interlock success`	Mode switch is successful.
`\Calling`	Send the target floor for the robot.
`\Calling success`	Robot accepts successfully.
`\call`	Call elevator.
`\call accept`	Elevator successfully receives the command.
`\call accept`	Elevator arrives at the floor.
`\open`	Elevator door open command.
`\open success`	Elevator door opens successfully.
`\GettingOn`	Command the robot to enter the elevator.
`\GettingOn success`	Robot successfully enters the elevator.
`\close`	Call close the elevator door.
`\close success`	Elevator door closes successfully.
`\go`	Command to go to the target floor.
`\go accept`	Elevator successfully receives the command.
`\go arrive`	Elevator arrives at the floor.
`\GettingOff`	Command the robot to exit the elevator.
`\GettingOff success`	Robot exits the elevator successfully.
`\Schedule Work`	Call the robot start cleaning task.
`\Schedule Work success`	Robot successfully accepts and begins work.

3.3.4 Simulator Module

The simulator module is a core component of the system, responsible for integrating the aforementioned modules to create individual simulators for the ROB, ELV, RPF and BOS. This simulator allows us to conduct separate operational simulations and tests for various components of the entire system without the need to deploy actual hardware.

By utilizing the simulator module, we can simulate the robot's behavior, including its movement, entering and exiting the elevator, and performing cleaning tasks. For the elevator simulation, we focus primarily on its floor-to-floor movement and the opening and closing of its doors. Additionally, the simulator can mimic the commands issued by RPF and the responses from the elevator, ensuring the smooth and correct flow of communication. This module enables us to test each component independently, allowing us to identify and resolve potential issues before actual deployment, thereby reducing risks.

3.3.5 Security Attack Module

The Security Attack Module is designed to assess the security of various components within the simulator by simulating a range of cyber-attack scenarios. We begin by analyzing the risks and impacts that the system's components would face under real-world conditions, and then design scenarios for several common types of attacks. These scenarios are subsequently tested within the simulator. We also propose countermeasures in response to the identified impacts. A detailed discussion of the attack emulation experiments will be presented in Chapter 5.

3.3.6 User Interface

As shown in Figure 3.7, to enhance user experience, we have developed an intuitive and user-friendly interface. This interface is built using Python and Kivy, offering cross-platform compatibility and high performance. Users can easily select the desired protocols, robots, elevators, tasks, and scenarios from the interface, and start the simulation by clicking the "Start" button. Once the simulation is complete, users can click the "Stop" button to view detailed runtime logs in the expandable area on the right, making it easy to access information at any time. The user interface features a modular design with clearly defined functional areas.

3.3.7 Other Modules

Other modules will not be introduced in detail, but they mainly include the storyboard and some common functions, as well as the definition of log formats and a

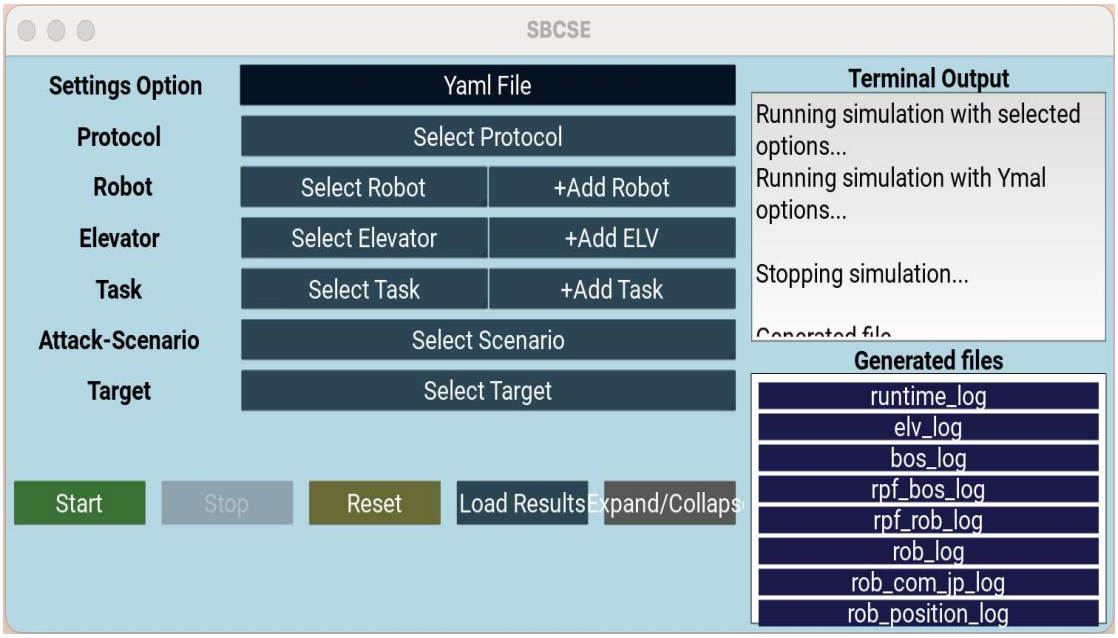

Figure 3.7: SBCSE user interface.

time simulation module.The time simulation module is responsible for controlling the time within the emulator uniformly. We have set a time acceleration factor, allowing the creation of accelerated time instances to more effectively control the simulation time and improve the efficiency of simulation testing.

3.4 Comparison of SBCSE and Real Building System

Although our system was designed and developed based on a smart building prototype from an architectural firm, we modified and adapted the communication protocols between some of the components during the design process. Specifically, we optimized the communication protocols between components to better meet the needs of our research objectives. For example, some components in the real building system use the PLC communication protocol, and the building company indicated that this component would be replaced with the MQTT protocol in the future. As a result, SBCSE then standardized on the MQTT protocol for communication. These modifications have resulted in some differences between our system and the original smart building prototype in terms of some features and performance.

In order to better analyze and discuss these differences, we used both Simu-

lation and Emulation. In our study, the different modules of the system take on their respective functions. SBCSE not only replicates the normal communication and command control between components, but also fixes the problems in the communication of the real building, so that the communication mechanism and performance of the simulator are closer to the real building environment.

Table 3.2: Feature Comparison: SBCSE vs. Real Building

Feature Comparison	**Details**	**SBCSE**	**Real Building**	**Type**
Network Communication Module	ELV & BOS Components	Interface: API, Protocol: MQTT, MQTTS	Real industrial protocols (PLC)	Emulation
	BOS & RPF Components	Interface: API, Protocol: MQTT, MQTTS	MQTT	Emulation
	RPF & ROB Components	Interface: API, Protocol: MQTT, MQTTS	MQTT	Emulation
Device Motion Module	Elevator	Simulation of elevator operation	Physical robot motion (sensor feedback + motion control)	Simulation
	Robot	Simulation of robot path planning and execution	Physical robot motion (sensor + motion control)	Simulation
Security Attack Module	Attack test	MITM, DDoS attack test	Real-world network attack scenarios	Emulation
	Security Measures	Supports security policy testing and optimization	Depends on physical protection and network security measures)	Emulation
Time Simulation	Accelerated Simulation	Adjustable, supports accelerated operation	Limited by physical network and device response speed	Emulation
User Interaction	Configuration File	Supports adjusting simulation parameters via configuration files (speed, robot, scenario selection)	-	-
	Code Interface	Supports dynamically adjusting simulation parameters	-	-
	User Interface (UI)	Kivy UI, supports task selection, log display	-	-
Other Features	Scalability	Easily extendable to add more devices	Limited by hardware cost and space constraints	-
	Cost	Computing resources, low cost	High cost for equipment purchase, maintenance, energyconsumption	-

In order to fully evaluate the performance of our system and to gain a deeper understanding of the similarities and differences between it and real buildings, we compared our system with real buildings. As shown in Table 3.2, we have compared and analyzed the simulation and emulation parts of the simulator with the corresponding parts of the real building system. Through this comparative analysis, we hope to clearly show the advantages and shortcomings of our system and provide a strong basis for further optimization and improvement. And it can help you better understand the functional implementation of different modules of SBCSE, the communication mechanism and how they work together to reproduce the behavior of a real building system.

Chapter 4

System Assessment

In this chapter, we will evaluate the developed system. The focus of the evaluation is to determine the effectiveness, accuracy, and reliability of the emulator in terms of emulation. In Section 4.1, we will discuss the methods used for the evaluation, outlining the specific approaches we have chosen to assess the emulator and providing the rationale for selecting these methods. Additionally, we will explain how these methods align with the objectives of our evaluation. In Section 4.2, we will present in detail the results obtained from the various tests and emulations conducted.

4.1 Evaluation Method

Regarding the evaluation method, we have chosen to analyze and visualize the log data. The reason for selecting log analysis is that our system is designed and developed based on logs provided by real buildings. Our goal is to verify whether the emulator can generate logs that are identical to those of real buildings. The evaluation steps are as follows:

- **Log Data Analysis**
  First, we analyze the log data generated by the emulator. This includes checking the structure, content, and format of the logs to ensure they match the logs provided by real buildings. If there are differences, we will analyze the reasons, propose corresponding countermeasures for the identified impacts, and modify the program to optimize the performance of the emulator, making it behave more like the real system.

- **Communication Flow Analysis**
  We analyze the communication flow to assess the overall communication process of the emulator and the communication between its components.

This includes confirming the subscription and publication of topics and the receipt and transmission of commands.

- **Data Visualization**
  After analysis, we visualize the communication data, which helps to intuitively understand the communication between components. Visualization allows us to more clearly see the differences between the emulator and the real system.

- **Comparative Analysis**
  Then, we conduct a comparative analysis to compare the communication flows between the actual system and the emulator. This step is crucial as it directly relates to the accuracy and reliability of the emulator. Our aim is to ensure that the communication process of the emulator is as close as possible to that of the actual system to verify its consistency. This step helps to improve the accuracy and reliability of the emulator and provides a emulation effect that is very close to the real system.

Through these steps, we can comprehensively evaluate the performance of the simulator and ensure that it can provide high-quality emulation results, laying a solid foundation for further development and practical application.

4.2 Assessment Results

This section will discuss the results obtained from the analysis and verification of the emulator's logs. All the test results presented in this thesis were performed on an Apple computer with the macOS 14.1.1 operating system, and a hardware configuration of an 8-core processor and 16 GB memory. The MQTT Broker agent version used in the communication process was Mosquitto 2.0.18.

4.2.1 Log Data Analysis

We first conducted a comparative analysis of the logs generated by SBCSE and the logs provided by the actual building system. Through this process, we confirmed that the structure, content, and format of the emulator logs match those of the actual building system logs. An example of the communication log format of SBCSE is shown in Figure 4.1. The communication logs are recorded in JSON format, with the left side of each record primarily recording the time, and the header of each message includes the relevant topic and direction of communication.

Compared to the actual building system, our system has made some adjustments. In the real system, the communication protocol between ELV and BOS

```
2024-11-05 20:20:54.433613 → E2D/cmd/toyosu/bld1/elevator/itest1/req was received:
2024-11-05 20:20:54.433613 {
2024-11-05 20:20:54.433613   sessionId: cc1d39f4-9e3f-49c0-9c66-544bbadce981
2024-11-05 20:20:54.433613   command: open
2024-11-05 20:20:54.433613   result: success
2024-11-05 20:20:54.433613 }

2024-11-05 20:20:54.450045 → cmd/toyosu/bld1/elevator/itest1/res was received:
2024-11-05 20:20:54.450045 {
2024-11-05 20:20:54.450045   sessionId: cc1d39f4-9e3f-49c0-9c66-544bbadce981
2024-11-05 20:20:54.450045   command: open
2024-11-05 20:20:54.450045   result: success
2024-11-05 20:20:54.450045 }
```

Figure 4.1: Example of the SBCSE log format.

uses PLC, and we were unable to obtain log data for this part. The construction company has indicated that they plan to replace the PLC protocol with MQTT in the future. Therefore, in our emulator, the communication between these two components also uses the MQTT protocol, and the data format has been unified to ensure consistency with other parts of the system.

4.2.2 Communication Flow Analysis and Data Visualization

In Section 3.3, we have already presented the overall communication flow of the system. Figure 3.6 showed the communication process of the system in executing commands and returning messages after the completion of commands. Where the vertical coordinates indicate the order of the communication commands, we can visualize the passing path of the commands among the components in the system as well as the sequential relationship among the messages.

Here, we analyze the communication flow of the system from another perspective. Figure 4.2 shows the overall communication flow of SBCSE after running with timestamps one the vertical axis. For ease of observation, we reset the timestamp to the seconds elapsed in the message relative to the previous message. We can more clearly observe the distribution of each component's communication command Message in the time dimension, which helps to analyze the timing of the communication, as well as can provide us with intuitive help when optimizing the communication performance of the system.

Regarding the steps of the analysis, we first extracted the main topics related to the commands, then filtered the data for each component, and visualized the

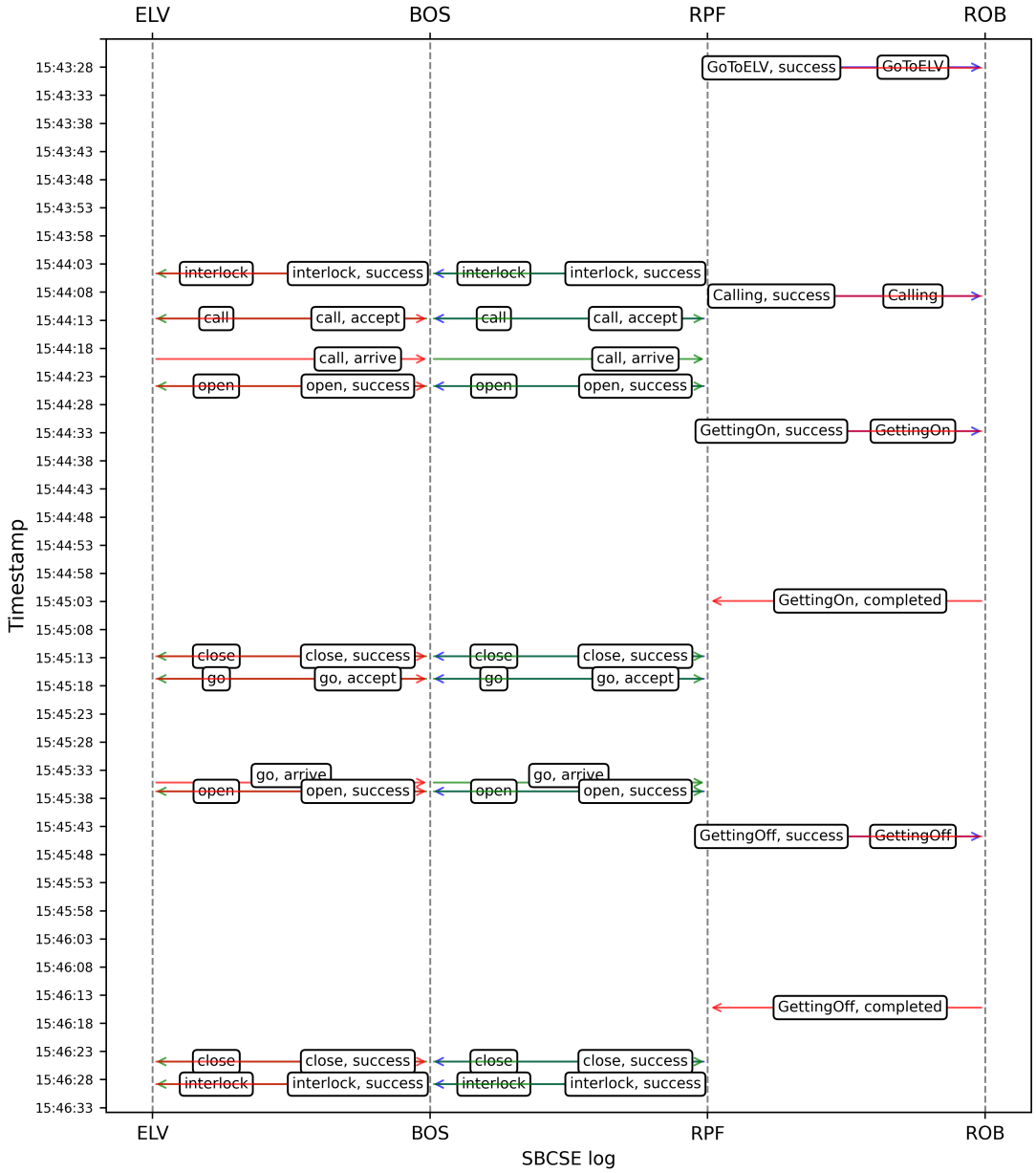

Figure 4.2: Entire communication flow according to SBCSE logs (Timestamp-Based).

communication process between the components.

Regarding the visualization of the communication data flow, we used a variety of libraries and tools provided by the Python, which greatly assisted us in performing an in-depth analysis of the communication log data. The powerful libraries that already exist in Python, such as Pandas, Matplotlib, and Seaborn for data manipulation and visualization, allowed us to more intuitively observe the inter-component communication between components and identify problems. With these tools, we were able to transform the communication log data into a graphical representation of the communication relationships between components, which allowed us to more accurately assess the communication performance of the simulator as well as help us to compare the simulator data with the real building data in a more comparable way. Through these analyses, we not only verified the accuracy of the SBCSE logs, but also enhanced our understanding of the system's communication flow, supporting further optimization and improvement of the system.

4.2.3 Comparative Analysis

In our comparative analysis, we focused on the consistency of communication. Figure 4.3 shows the comparison of the communication process between BOS and RPF in our emulator and the actual building system. From the figure, we can confirm that the communication process in our emulator matches that of the real system, ensuring the accuracy of our emulation. The main comparison here is between the content and the order of the communication commands.

In our evaluation, the accuracy calculation focuses on how well the emulator matches the command (cmd) data in the real building communication system. We defined the formula for calculating the communication accuracy of the SBCSE. Specifically, our evaluation metrics include two main aspects: the order of commands between the sender and receiver, and the consistency of the cmd message. There are cmd data and dt data in our system, here we are evaluating the cmd messages. The number of correctly matched messages is the number of messages in the emulator and in the actual system where the content and order of the messages related to cmd are exactly the same. The total number of messages is the total number of all cmd-related messages sent by the sender during the communication process.

In Figures 4.4 and 4.5 we compare the communication flow between SBCSE and the actual building system regarding the components BOS and RPF, using the timestamp as the vertical coordinate for the timing of the communication of the commands. As can be seen in those figures, the contents of the messages coincide, although there are some differences in the processing time. Since the communication protocols between some components in our system are different from the

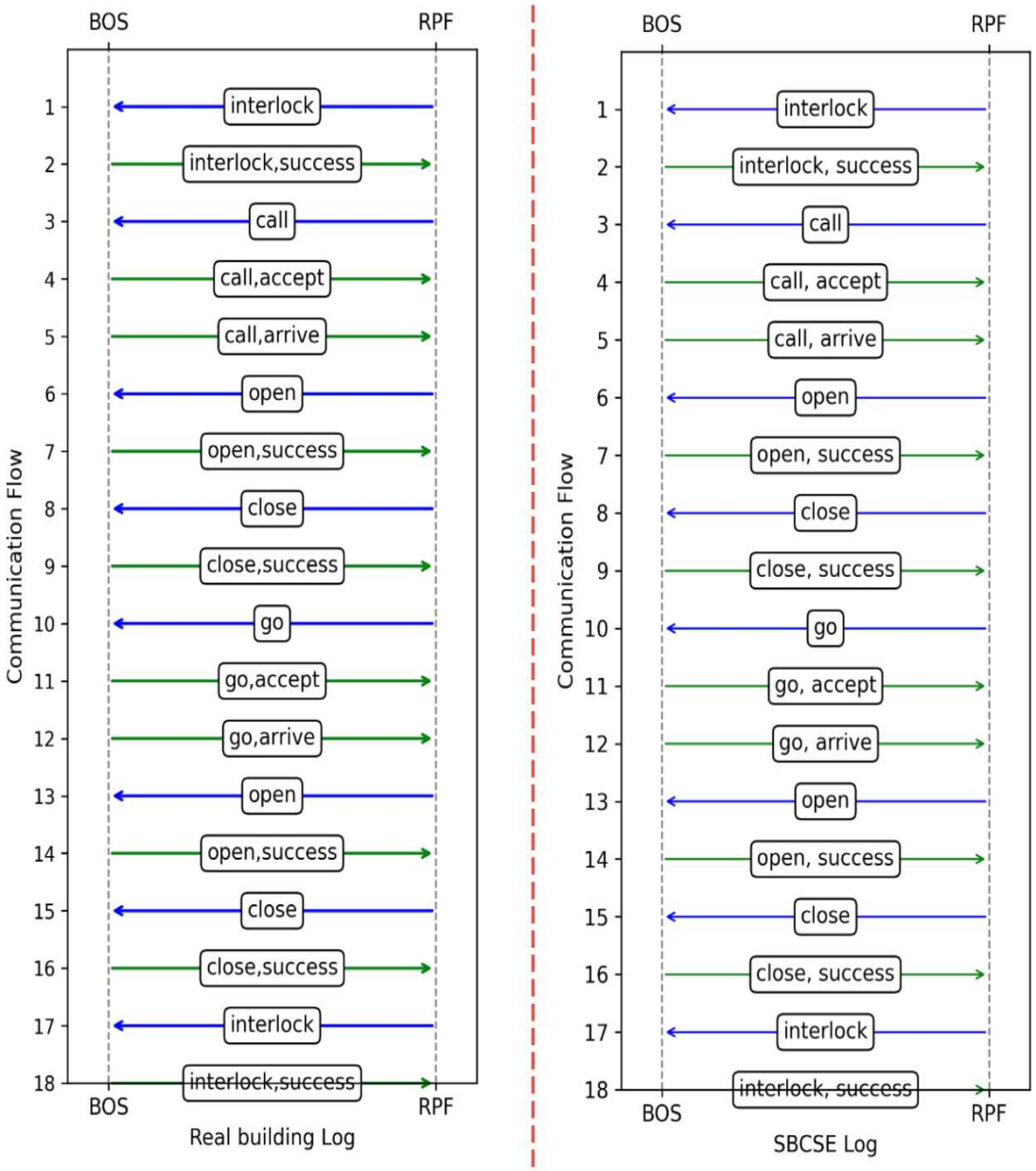

Figure 4.3: Communication flow in the real building versus the SBCSE log.

real system, such as the forwarding of MQTT protocol and the algorithm of the system's acknowledgment mechanism for the messages will lead to the difference in the message timing.

Given that network latency, processing time, and other factors in the actual system may lead to differences in the sending and receiving times of messages, and given that we extend some of the communications, we exclude the consistency of timestamps in the calculation of the accuracy rate and focus only on the order of communication messages. We define the system accuracy rate calculation formula as follows:

$$\text{Accuracy} = \frac{\text{Number of correctly matched messages}}{\text{Total number of messages}} \times 100\% \tag{4.1}$$

$$\text{Average match rate} = \frac{\sum_{i=1}^{n} \text{Accuracy}_i}{n} \tag{4.2}$$

We repeated the experiment 30 times using the emulator, thus ensuring the accuracy of our results. For the cmd data between BOS and RPF, the accuracy was calculated separately for each time according to the defined formula. Finally, the average value was calculated to give 100% accuracy for the communication flow between BOS and RPF.

Furthermore, due to the limited log data available for the robot, we expanded the communication design between the ROB and RPF based on the existing logs. Subsequently, we compared and visualized the communication results with the data provided by the actual building system. As shown in Figures 4.6 and 4.7, we achieved consistency in the communication between the ROB and RPF, proving the effectiveness of our design.

4.2.4 Summary and Discussion

We have validated the practicality of SBCSE via several experiments. The comparative results indicate that we have successfully replicated the communication processes of the real system, confirming the utility and accuracy of the emulator in real testing environments. These findings suggest that our emulator is reliable for testing in real environments. The responses to various inputs are also within normal standards, which is crucial for ensuring that the emulator can accurately predict the behavior of the actual system under different conditions.

By conducting this comparative analysis, we have not only confirmed the reliability of the emulator but also gained a deeper understanding of its performance in emulating real scenarios. This is particularly important for pre-testing system updates or changes in a controlled environment before implementing them in the actual building system. The ability to predict and analyze potential issues in a

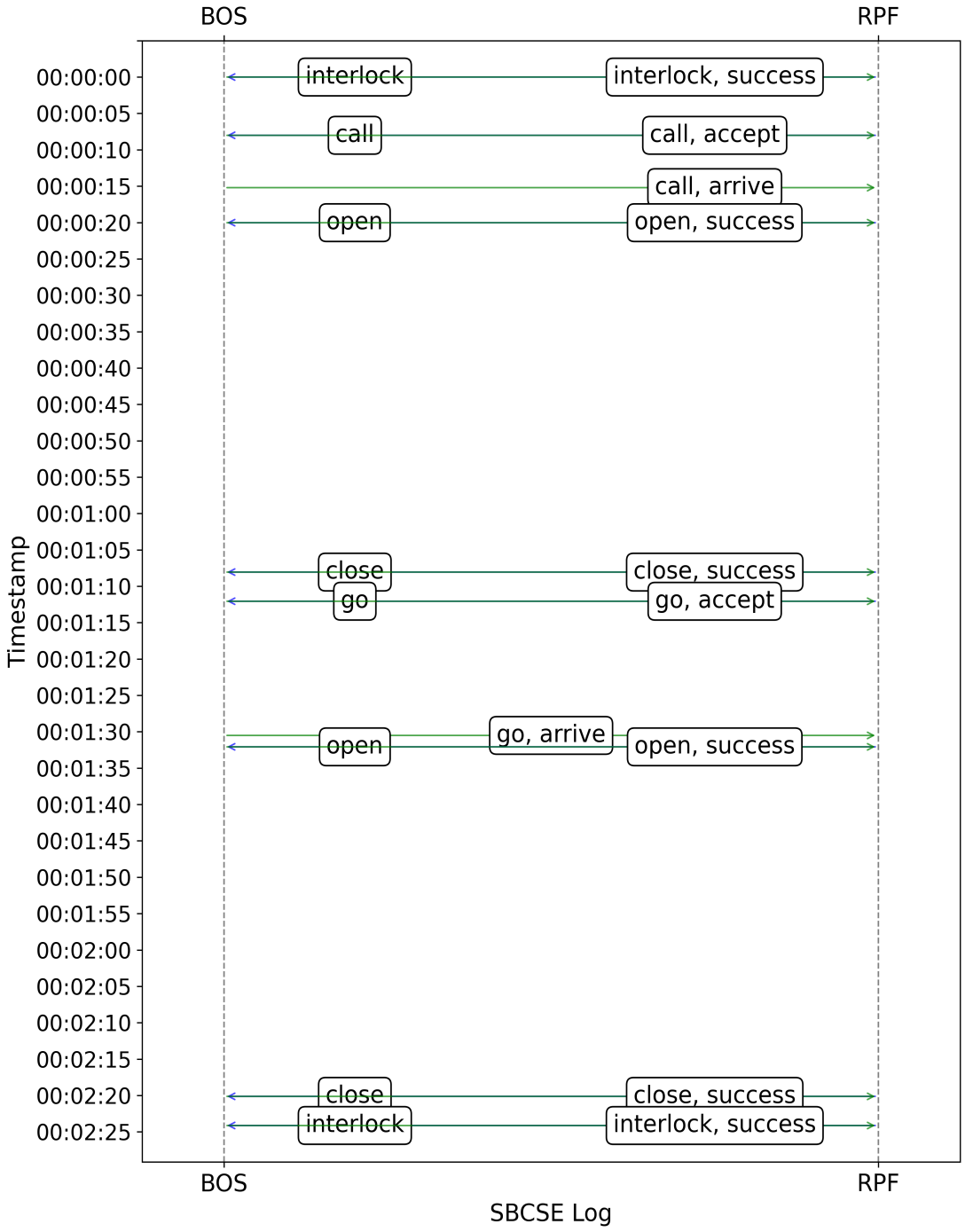

Figure 4.4: Communication flow log in SBCSE based on timestamp.

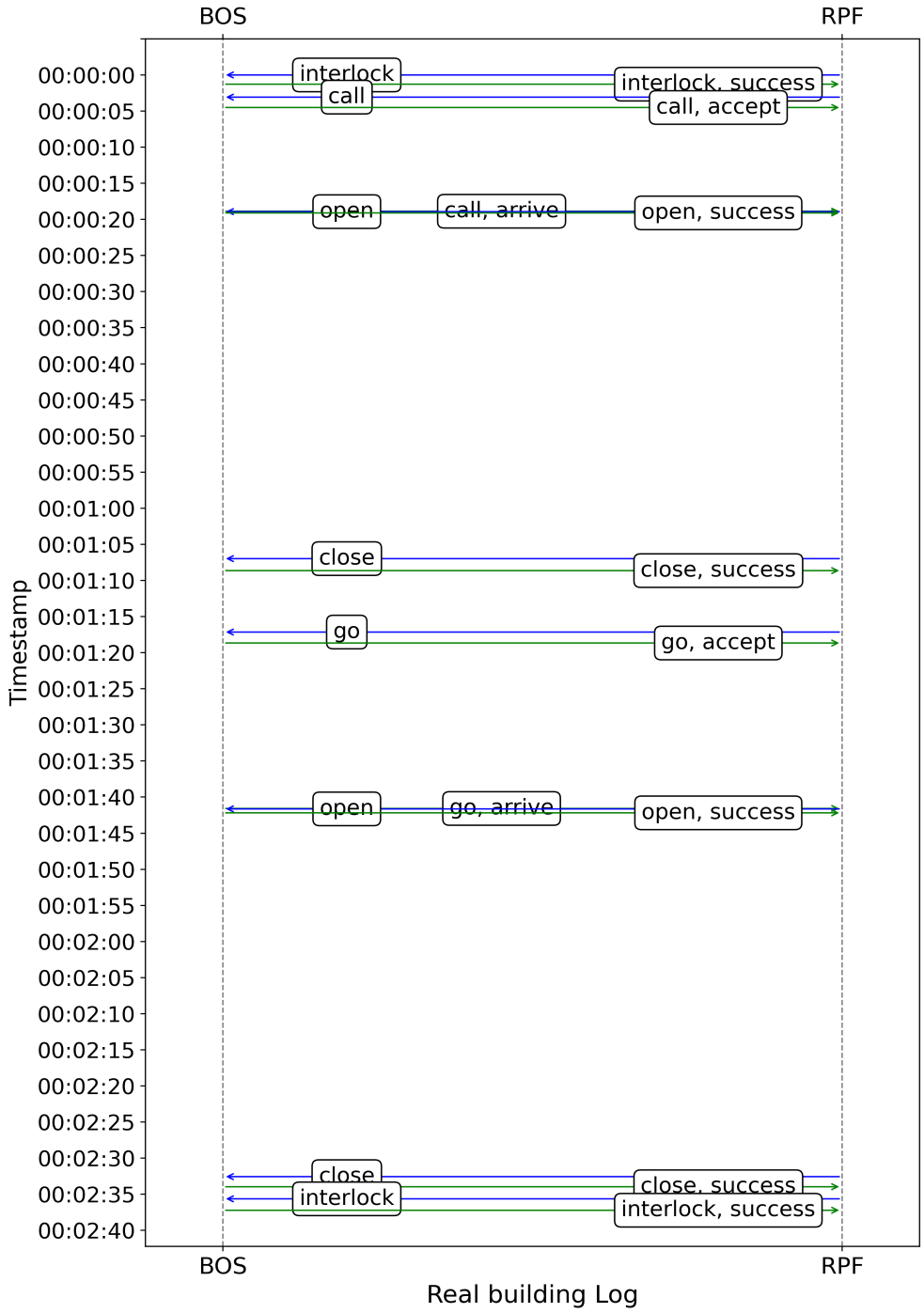

Figure 4.5: Communication flow log in the real building based on timestamp.

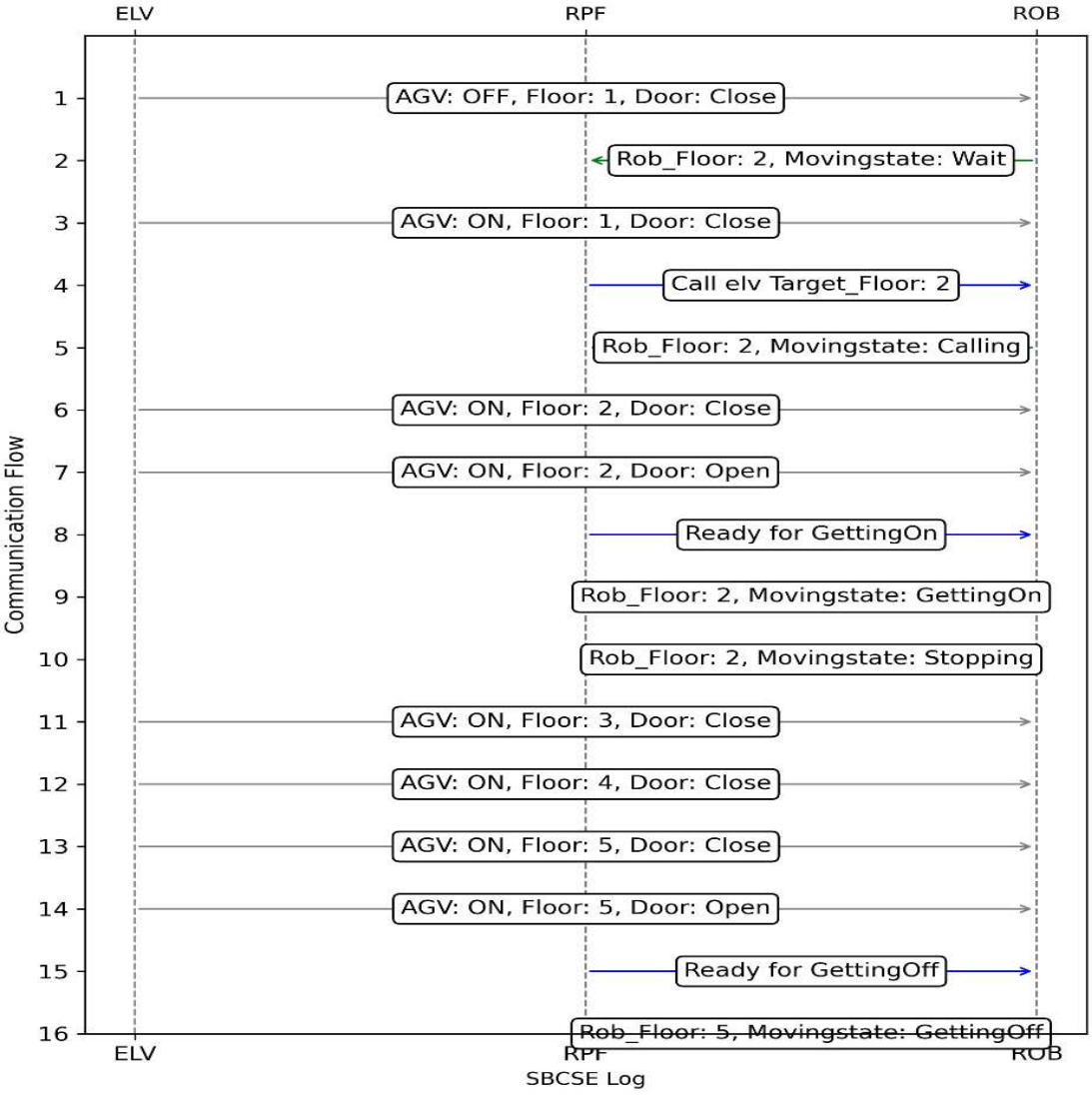

Figure 4.6: ROB-related communication flow in the SBCSE log.

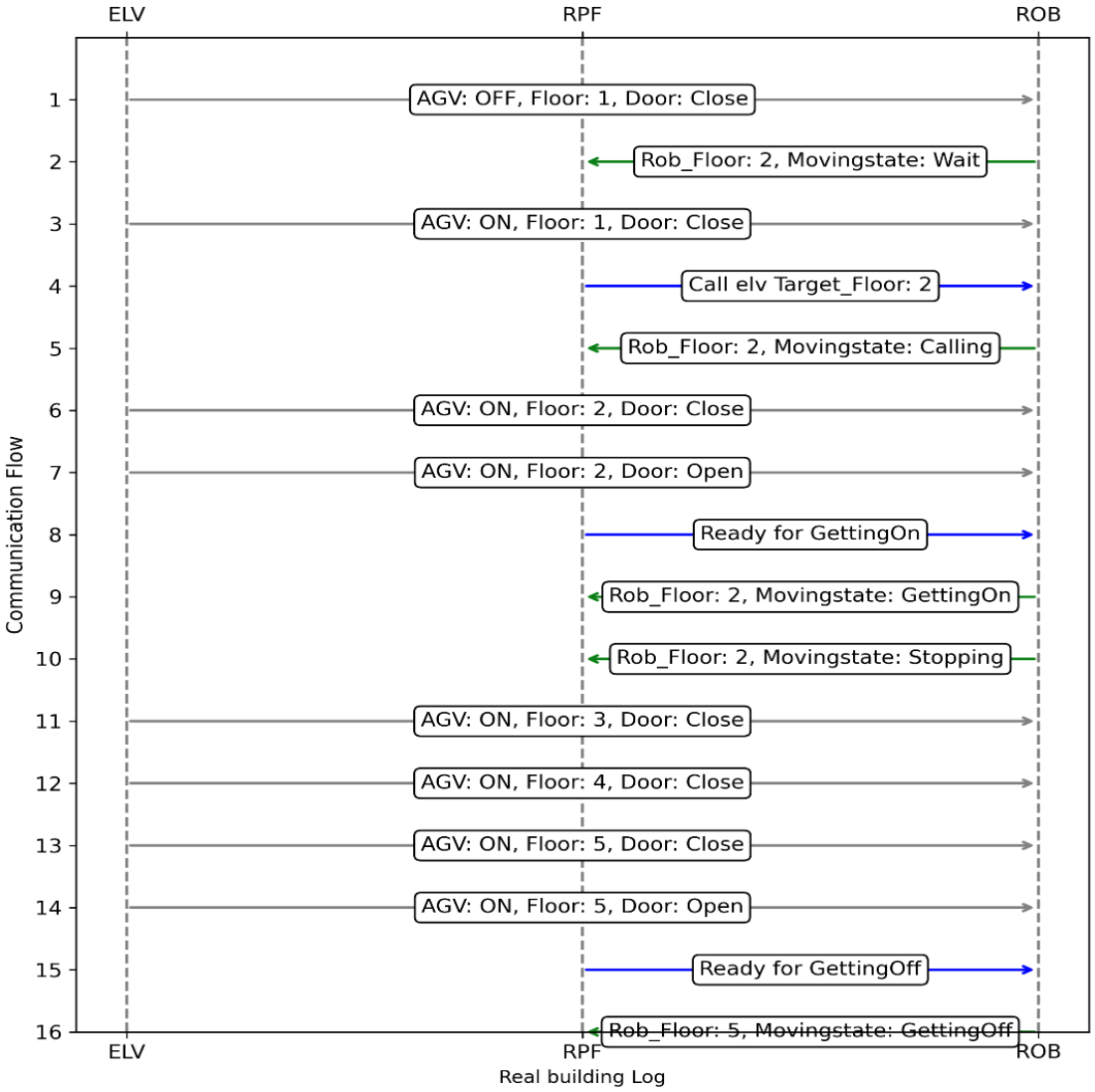

Figure 4.7: ROB-related communication flow in the real building log.

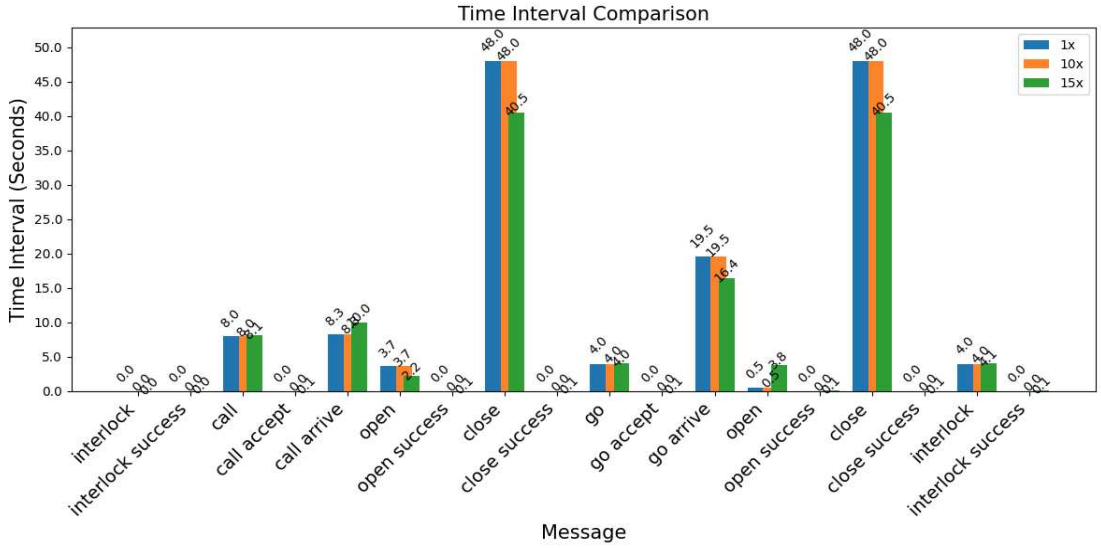

Figure 4.8: CMD message time interval comparison for various emulation speeds

emulated environment can save time and resources, reducing the risk of system failures or downtime in actual operations.

At the same time, the following can be said regarding the current shortcomings of the emulator. In terms of the emulator's accelerated performance, our system can emulate at a multiple speed, which can effectively save emulation time. We conducted several experiments on the emulation of different rates, and the experimental results are shown in Figure 4.8. From the figure, it can be observed that during the emulation process at a speed below 10 times, the response time and delay of each message are basically stable, and there is no packet loss. However, when the emulation speed exceeds 10 times, the reply time of messages begins to become unstable. In response to this issue, we will conduct in-depth discussions and improvements in our future research to further enhance the performance and stability of the emulator.

Chapter 5

Attack Scenarios and Proposed Security Measures

In this chapter, we will explore the experimental design for security testing on SBCSE and the organization of security measures. The designs of these security scenarios are all implemented in the Security Attack Module. In Section 5.1, we will detail the security analysis conducted on various components of the emulator, which forms the basis for our security testing. In Section 5.2, I will introduce the security attack scenarios designed for the high-level risks that our components may face. Finally, in Section 5.3, we will provide an overview of the relevant security measures.

5.1 Risk Analysis

Before implementing the attack scenarios, we first conduct a comprehensive risk analysis. The specific steps are as follows:

- **Step 1: Component Analysis**

  In this stage, we conduct analysis of the functionality of each component, as well as the data they store and process, and their roles within the overall architecture. We conduct a detailed review of each component to identify their security requirements and potential vulnerabilities.

  We have compiled Table 5.1, which serves as the foundation for our analysis. This chart helps us to better understand the structure of the system and the interactions between components, thereby providing a solid basis for subsequent risk assessment and security testing.

- **Step 2: Risk Categorization**

Table 5.1: System component functionality analysis

Component	Function	Stored or Processed Data
`MQTT Broker`	Relay messages, Manage device communication	Publish/subscribe message, Device status information
`BOS`	Data transfer, Control of devices in the building	Control commands, Elevator status data
`RPF`	Robot control and command transmission	Robot control commands, Task allocation, Commands to the elevator
`ROB`	Cleaning, Data transmission (robot dt)	Position data, Operation status data
`ELV`	Movement between floors, Data transmission (dt data)	Floor status, Operation status data

In this step, we refer to [18] to categorize the components using labels to facilitate subsequent risk classification and management.

The risk categories for the MQTT Broker mainly fall under the Server/ Protocol/ Information category, which covers risks at the server and protocol levels. Secondly, the risks for parts BOS and RPF are primarily concentrated in the areas of operating systems, software, and information security. Furthermore, the risk classification for the ROB involves the Internet of Things device information, intelligent robots, and the field layer where they are located. Lastly, the risk category for the ELV mainly encompasses the Internet of Things device information and the field layer where the elevator is situated.

- **Step 3: Attack Impact Analysis**

  In this step, we categorized and assessed the risks associated with each component to determine potential vulnerabilities.

  First, as shown in Table 5.2, we have referenced the standards and guidelines [19, 18, 20] to classify and organize the Threats and Risk Sources/Incidents that smart building systems are faced with. Then, in Table 5.3, we evaluated several components of the system, linking them to the tags and risk categories established in the previous steps. For each component, we compiled an Attack Surface for Components. The symbols used in this table are:

  1. "+" indicates a high likelihood of risks occurring in the system.
  2. "-" denotes components that are generally not prone to risks.

3. "±" represents cases where risks may occur but are minimal; such cases are excluded from this study.

- **Step 4: Threat Analysis and Scenario Design**

  In this stage, we analyze the potential impacts and associated threats to each component based on the risk categories established in Step 3. We have summarizes the impact scope, impact explanation, and examples of attack scenarios, and clearly marks the impact of different attack scenarios. Table 5.4 summarizes the security event impact analysis of ELV/BOS, Table 5.5 presents the security event impact analysis of RPF/ROB, and Table 5.6 summarizes the security event impact analysis of MQTT-Broker.

  Referring to [19], we identified five scenarios that have significant impacts on system components, namely:

  1. Attacks on information transmitted over the network (man-in-the-middle (MITM) attack)
  2. Distributed Denial-of-Service (DDoS) attacks using IoT botnets
  3. Attacks related to Broken Access Control (Unauthorized Access)
  4. Attacks involving malware
  5. Attacks based on human error and social engineering techniques

  In Tables 5.4, 5.5 and 5.6 we abbreviated the names of these attacks with labels, as follows: "MITM" stands for Man-in-the-middle attack, "DDoS" stands for Distributed Denial-of-Service, "BAC" stands for Broken Access Control, "MA" stands for malware, and "SEA" stands for social engineering attack. In the next step, we will design the corresponding attack scenarios and experimentally implement these scenarios.

Table 5.2: Threats and risk sources for smart building systems

Type	**Threat**	**Risk Sources/Incident**
Nefarious activity / Abuse	Denial of Service	DoS/DDoS attacks cause service disruptions and affect system availability.
	Malware	1. Malware infections occurring in part of the building system easily spread through the building's internal network. 2. Unauthorized devices are connected, and malware is introduced.
	Manipulation of hardware /software	Unauthorized modifications to hardware and software can lead to system failures or data breaches.
	Manipulation of Information / Unauthorized access	1. Broken Access Control 2. Unauthorized access or data tampering.
	Gathering system information from intruders	Easy access to log information allows intruders to examine the logs and launch attacks.
	Targeted attacks	1. Injection (SQL Injection), 2. Server-Side Request Forgery
	Abuse of personal data	Cryptographic failures can lead to personal data breaches.
	Brute force	Brute Force Attack can result in account compromise.
Eavesdrop /Interception /Hijacking	Man-in-the-Middle attack /Session hijacking	Executing unauthorized commands can cause improper behavior.
	IoT communication protocol hijacking	Attackers intercept or tamper with communication between IoT devices and send unauthorized commands.
	Network reconnaissance	Attackers collect information within the network and steal confidential or authentication data.
Physical attack	Vandalism and theft	Destroying or stealing physical equipment or devices can impact the system's availability and the confidentiality of data.
	Sabotage	Disrupting the normal operation of services or systems can cause chaos within the system.

TYPE	THREAT	Risk Sources/Incident
Fault /malfunction /outage	Failure or malfunction of a sensor / actuator	Causing disruption to the normal operation of the system.
	Failure or malfunction of a control system (PLC, RTU, DCS)	Failures or malfunctions in the control system can impact process control, threatening the overall safety and availability of the system.
	Software vulnerabilities exploitation	Security Misconfiguration
	Delayed countermeasures result from difficulty analyzing unauthorized intrusions	Logs are not properly collected, making it difficult to analyze the situation of intrusions or infections.
	Failure or disruption of service providers	The lack of proper backup data hinders recovery efforts in the event of damage to the system.
	Communication network outage	The communication network between systems is interrupted, making services and data unavailable.
	Power supply outage	The system stops, and services or data become unavailable.
	Loss support services (MES, ERP, CRM)	Delayed system management, monitoring, and troubleshooting can lead to service outages and data loss.
Company/ Personnel management	Building management company lacks adequate staff training	Delayed initial response to the attack can lead to the escalation of damage.
	Unintentional data or configuration changes in the system	Security and performance issues arise.
	Attacked by inside workers, etc.	Designated workers perform unauthorized operations on systems or devices beyond their authority.
	Improper use or care of equipment or systems	Emergency response procedures are not adequately established within the operational standards.

TYPE	THREAT	Risk Sources/Incident
Legal	Violation of rules and regulations / Breach of legislation / Abuse of personal data	Non-compliance with laws and regulations, unauthorized access to or misuse of personal data, may lead to legal issues and personal information leaks.
	Failure to meet contractual requirements	Violations of contractual terms can result in issues with service delivery or data management, leading to legal disputes or service interruptions.
Outage /Disaster	Natural disasters	Natural disasters (such as earthquakes or floods) can destroy systems and data, causing service disruptions and delays in recovery.
	Environmental disasters	Environmental disasters (such as fires or chemical spills) can result in system and data loss, causing service disruptions and environmental impact.
Secure SDLC, SDL / imit configuration	Insecure Design	Vulnerable and Outdated Components
	Authentication	Identification and Authentication Failures
	Damage caused by third parties	Attacks by third-party organizations or misconduct by service providers can compromise the system and data, leading to service disruptions.
	The log recording failed or is missing	Security Logging and Monitoring Failures

Table 5.3: Attack surface for SBCSE

TYPE	THREAT	Broker	BOS	RPF	ROB	ELV	Reference
Nefarious activity / Abuse	Denial of Service	+	+	+	±	±	[19] [18] [20]
	Malware	+	+	+	+	+	[19] [18]
	Manipulation of hardware /software	+	+	+	+	+	[19] [18]
	Manipulation of Information / Unauthorized access	+	+	+	+	+	[19] [18] [20]
	Gathering system information from intruders	-	+	+	+	+	[18]
	Targeted attacks	+	+	+	±	±	[19] [18] [20]
	Abuse of personal data	-	-	-	-	-	[19] [18] [20]
	Brute force	+	-	+	-	-	[19] [18]
Eavesdrop /Interception /Hijacking	Man-in-the-Middle attack /Session hijacking	+	+	+	±	±	[19] [18] [20]
	IoT communication protocol hijacking	+	+	+	±	±	[19] [18] [20]
	Network reconnaissance	+	+	+	+	+	[19] [18]
Fault /malfunction /outage	Failure or malfunction of a sensor / actuator	±	±	±	+	+	[19] [18]
	Failure or malfunction of a control system (PLC, RTU, DCS)	-	-	-	-	-	[19] [18]
	Software vulnerabilities exploitation	±	±	±	+	+	[19] [18] [20]
	Delayed countermeasures result from difficulty analyzing unauthorized intrusions	+	+	+	+	+	[18]
	Failure or disruption of service providers	±	±	+	+	+	[19] [18]
	Communication network outage	+	+	+	±	±	[19] [18]
	Power supply outage	±	±	±	+	+	[19] [18]
	Loss support services (MES, ERP, CRM)	-	-	-	-	-	[19] [18]

TYPE	THREAT	Broker	BOS	RPF	ROB	ELV	Reference
Physical attack	Vandalism and theft	±	±	±	+	+	[19] [18]
	Sabotage	±	±	±	+	+	[19] [18]
Company/ Personnel /management	Building management company lacks adequate staff training	-	-	-	-	-	[18]
	Unintentional data or configuration changes in the system	±	±	±	+	+	[19] [18]
	Attacked by inside workers, etc.	+	+	+	+	+	[18]
	Improper use or care of equipment or systems	±	+	+	+	+	[19] [18]
Legal	Violation of rules and regulations /Breach of legislation /Abuse of personal data	-	-	-	-	-	[19] [18]
	Failure to meet contractual requirements	-	-	-	-	-	[19] [18]
Outage/ Disaster	Natural disasters	+	+	+	+	+	[19] [18]
	Environmental disasters	+	+	+	+	+	[19] [18]
Secure SDLC SDL/ imit configuration	Insecure Design	+	+	+	±	±	[18] [20]
	Authentication	+	+	+	-	-	[20]
	Damage caused by third parties	±	-	+	+	+	[19] [18]
	The log recording failed or is missing	-	+	+	+	+	[20]

Table 5.4: Security incident impact analysis for ELV/BOS

CIA Triad	Scope of Impact	Impact Description	Attack
Confidentiality/ Integrity	Data tampering	Elevator status data is tampered, showing normal but with actual faults.	- MITM - BAC - MA - SEA
	Backdoor implantation	Attacker implants a backdoor, continuously monitoring and altering system data.	- BAC - MA - SEA
	System log tampering	Attacker manipulates system logs to cover their activities or mislead investigators, preventing detection of data breaches.	- BAC - MA - SEA
	System configuration errors	Malicious modification of system configuration leads to performance degradation or functionality failure.	- MA - SEA
	Erroneous Operation Guidelines	Attacker misguides maintenance personnel or admins into performing unsafe or incorrect actions.	- SEA
Availability	System Downtime and Misoperation	Attacker alters commands, causing the elevator to stop at non-designated floors.	- MITM - BAC - DDoS - MA - SEA
	Frequent elevator restarts and resource exhaustion	Elevator repeatedly opens/closes doors and restarts due to incorrect commands, eventually stopping.	- MITM - BAC - DDoS - MA - SEA
	Unmanned operation	Elevator starts abnormally when not in use, wasting energy and potentially creating safety risks.	- MITM - BAC - MA - SEA
	Emergency Response Failure	In specific cases, the elevator is ordered to shut down, preventing timely evacuation.	- MITM - BAC - DDoS - MA - SEA
Operation and management	Equipment Maintenance and Replacement	Frequent misoperations result in the need for frequent repairs or replacements.	- MITM - BAC - DDoS - MA - SEA
	Business Interruption and Delays	Elevator downtime causes business interruptions.	- MITM - BAC - DDoS - MA - SEA
Compliance	Non-compliance	System fails to meet safety standards, leading to regulatory investigations and penalties.	-

Table 5.5: Security incident impact analysis for RPF/ROB

CIA Triad	Scope of Impact	Impact Description	Attack
Confidentiality/ Integrity	Data tampering	The robot's status data is tampered with, appearing normal but actually malfunctioning.	- MITM - BAC - MA - SEA
	Backdoor implantation	The attacker implants a backdoor, continuously monitoring and tampering with system data.	- BAC - MA - SEA
	System log tampering	The attacker modifies system logs to cover their activities or mislead investigators.	- BAC - MA - SEA
	System configuration errors	The attacker alters system configurations, causing performance degradation or partial failure during high load.	- MA - SEA
	Erroneous Operation Guidelines	The attacker misleads maintenance staff or administrators into performing unsafe or incorrect actions.	- SEA
Availability	System Downtime and Misoperation	The robot receives tampered path instructions, which could lead to wrong delivery routes.	- MITM - BAC - DDoS - MA - SEA
	Physical collision and damage	Incorrect instructions could cause the robot to collide with people or objects, resulting in physical damage or injury.	- MITM - BAC - SEA
	Frequent restarts	The robot frequently restarts due to incorrect instructions, eventually ceasing operation.	- MITM - BAC - DDoS - MA - SEA
	Unmanned operation	The robot starts abnormally without supervision, wasting energy and potentially creating safety risks.	- MITM - BAC - MA - SEA
	Emergency Response Failure	In emergencies, the robot fails to stop or avoid hazards as expected, increasing risks.	- MITM - BAC - DDoS - MA - SEA
Operation and management	Equipment Maintenance and Replacement	Frequent misoperations result in the need for frequent repairs or replacements.	- MITM - BAC - DDoS - MA - SEA
	Business Interruption and Delays	Robot downtime causes business interruptions.	- MITM - BAC - DDoS - MA - SEA
Compliance	Non-compliance	System fails to meet safety standards, leading to regulatory investigations and penalties.	-

Table 5.6: Security incident impact analysis for MQTT-Broker

CIA Triad	Scope of Impact	Impact Description	Attack
Confidentiality/ Integrity	Data leakage	The attacker obtains sensitive business data through the attack, causing a risk of data leakage.	- BAC - MA - SEA
	Privilege leakage	The attacker exploits vulnerabilities to gain system administrative privileges, thereby taking full control of the system.	- BAC - MA - SEA
	Log leakage	The attacker acquires system operation logs to analyze the system's performance and potential security vulnerabilities.	- BAC - MA - SEA
	Backdoor implantation	The attacker implants a backdoor in the data platform, continuously monitoring and tampering with communication between the MQTT-Broker and RPF.	- BAC - MA - SEA
	Data tampering	The attacker alters communication from BEP to the MQTT-Broker, causing the robot to receive incorrect instructions.	- MITM - BAC - MA - SEA
	Security policy failure	The attacker disables the security policies in the data platform, allowing data to be transmitted without verification or authorization.	- BAC - SEA
Availability	Data loss or delay	The attacker causes data loss in the data platform's transmission, leading to incomplete business data.	- DDoS
	System stoppage /Incorrect operations	The attacker overloads the data platform's processing load, causing the system to crash.	- DDoS

5.2 Attack Scenarios

In this section, we will introduce the attack scenarios designed for our system components, including the basic steps of the attacks and a brief introduction to each attack. Our scenario design focuses primarily on attack mechanisms and attack steps without delving into the network topology level. Therefore, network devices such as routers and converters are omitted from the figure.

5.2.1 MITM Attack Scenario

This section presents the MITM attack scenarios designed for the emulator. Man-in-the-middle attack is a type of network attack where an attacker secretly intercepts and manipulates information between two communicating parties [21]. The entry point of the attacker is the access point used by the attacker to intercept and manipulate the communication data, this can be a router, server, device equipped

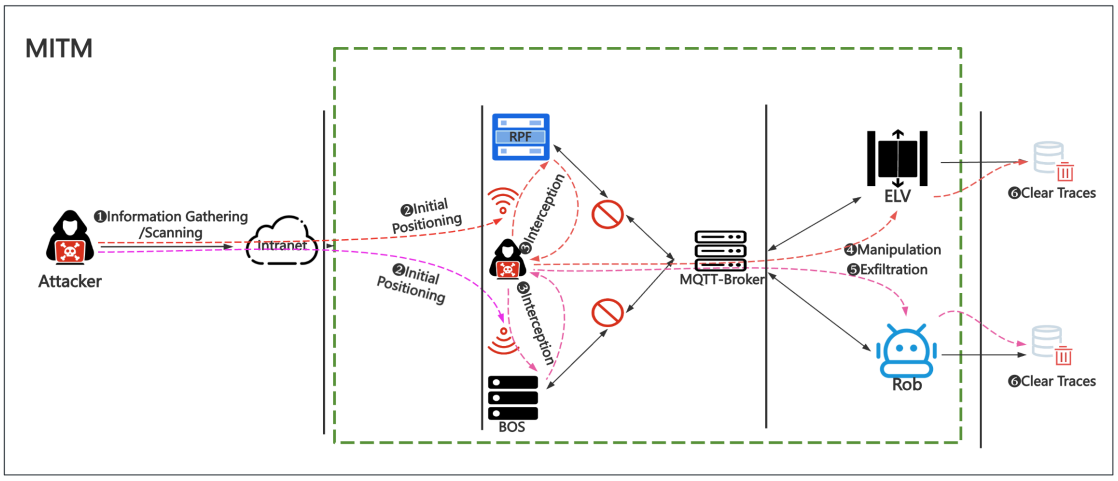

Figure 5.1: MITM attack scenario overview.

with a wireless card, etc. We have organized several common MITM attacks with reference to [22, 23, 24], and attackers generally use the following methods to implement the attacks:

1. Network Configuration Vulnerabilities and Other Rogue Access Points: Attackers can maliciously insert into communication paths by exploiting vulnerabilities in network configurations such as unencrypted Wi-Fi networks or cache vulnerabilities.

2. Network Spoofing: Attacks can be made by forging packets in network protocols so that the target device has a false perception of the attacker as a trusted device. Such as ARP Spoofing and DNS Spoofing, mDNS Spoofing and so on.

3. Exploit the vulnerability of the protocol: exploit the vulnerability of the communication protocol to attack. Such as brute force breaking WPA and WPA2, SSL/TLS degradation attacks.

4. Utilization of malware: Attackers can use phishing emails or malicious links to induce administrators to divulge sensitive information, or use malware to infect clients and gain access.

For SBCSE we have designed MITM attack scenarios. As shown in Figure 5.1, the components of our design that are susceptible to eavesdropping by attackers are BOS and RPF. The attack steps are shown below:

- Step 1: Information Gathering
  Collect information about the target systems (BOS/RPF) and the communication protocols used by the broker to identify potential vulnerabilities.
- Step 2: Initial Positioning
  Position the device controlled by the attacker within the communication path between the BOS/RPF and the MQTT broker.
- Step 3: Interception
  Intercept and capture the communication between the BOS/RPF and the broker.
- Step 4: Manipulation
  Manipulate the intercepted communication to alter the behavior of the elevator and robot.
- Step 5: Exfiltration
  Extract sensitive data from the intercepted communication.
- Step 6: Clearing Traces
  Remove all malicious software and clean system logs to hide the evidence of the attack.

5.2.2 DDoS Attack Scenario

This section will introduce a Distributed Denial of Service attack scenario designed for emulators. DDoS refers to the act where attackers send a large volume of requests or consume a significant amount of resources to a target system in a distributed manner, thereby rendering the system unable to provide services normally [25]. The attacker can utilize the public APIs exposed by the system to send a large number of malicious requests, causing the system to be unable to effectively handle the out-of-scope highly concurrent requests and eventually service interruption. We refer to [26, 27] to summarize the attacks. The common attack methods are as follows:

1. Traffic-based attacks: Attackers use to send malicious traffic requests to exhaust server resources to make the service stop. Such as UDP floods and other spoofed-packet floods and so on.
2. Application-layer attacks: Attackers can take advantage of fewer requests to paralyze an application. Attackers can target Apache, Windows vulnerabilities for low-speed and slow-speed attacks to crash the Web server. Application layer protocol weaknesses can be utilized to attack and exhaust

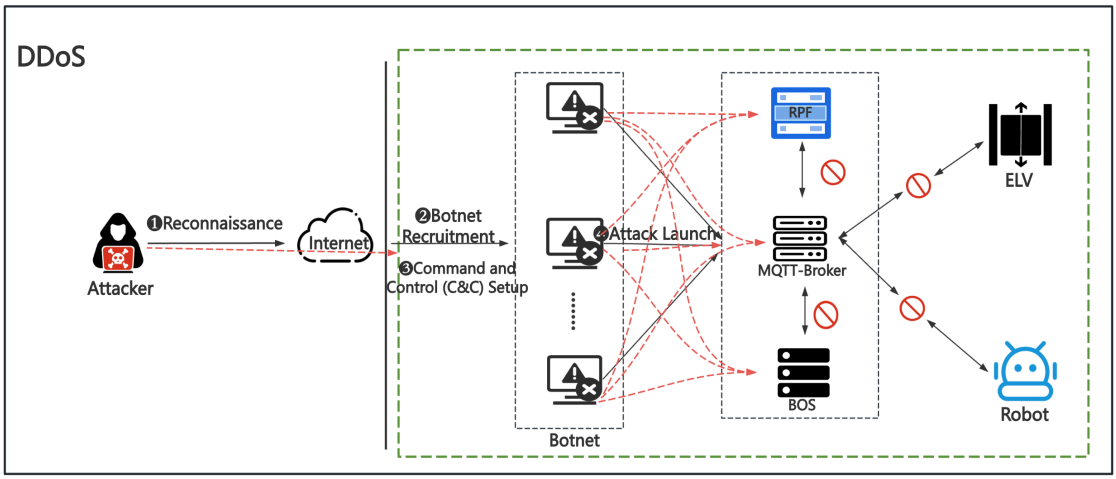

Figure 5.2: DDoS attack scenario overview.

server resources. Application layer attacks include: Session FloodingRequest Flooding Attack, Slow Request/Response Attack, etc.

3. Protocol-based attacks: Attacks that consume actual server resources or intermediate communication equipment resources, such as SYN floods, fragmented packet attacks, etc.

In our system, as shown in Figure 5.2, I have designed attack scenarios for our components. Attackers can create multiple Bot machines to send a large number of requests to the MQTT-Broker or BOS and RPF, thereby preventing the system from communicating normally and affecting the message exchange between components. Attackers typically need to go through the following steps to carry out an attack:

- Step 1: Reconnaissance
  Gather information about the target, including IP addresses, domain names, server types, and potential vulnerabilities.
- Step 2: Botnet Recruitment
  Compromise multiple devices to form a botnet, which can be used to generate traffic against the target.
- Step 3: Command and Control Setup
  Establish a command and control server to coordinate the botnet's activities.
- Step 4: Attack Launch
  Launch the DDoS attack to overwhelm the target server with traffic.

5.2.3 BAC Attack Scenario

This section will introduce BAC attack scenarios designed for emulator. Broken Access Control refers to the act that An attacker can bypass authorization and perform tasks as a legitimate user, allowing unauthorized users to access, modify, or delete data to which they are not entitled [28]. Attacker can use improper access mechanisms to gain illegal control over the system. The core issue of BAC is the failure to properly restrict user access permissions when the system is developed or configured.

The access point for attackers is typically the part of the target system where access control has not been correctly implemented or verified. Attackers can exploit misconfigurations, vulnerabilities, or unprotected API endpoints to bypass authentication for controlled access to the system. We refer to [28, 29] for a compilation of several common BAC attacks, which are typically exploited by attackers in the following ways: unauthorized access attacks. Specifically, Broken Access Control includes the following common types of attacks:

1. Exploiting endpoints: Attackers can exploit unprotected endpoints to bypass access controls, such as application-system interaction points, application-trusted services, APIs, and so on.

2. Privilege escalation: Attackers can gain access to ordinary users by changing URL parameters, weak passwords, etc., and then gain access to administrators or users with higher privileges horizontally or vertically to carry out malicious activities.

3. Exploiting vulnerabilities: By exploiting vulnerabilities such as Insecure Direct Object References (IDOR), attackers can guess the direct references of internal objects, brute force them, and bypass access controls.

In our system, as shown in Figure 5.3, we have designed attack scenarios for components. Among our components, BOS and RPF are vulnerable to unauthorized access by attackers. With unauthorized access to the system, the attacker can manipulate the system, which may cause some physical damage. An attacker usually needs to go through the following steps to carry out the attack:

- Step 1. Information Gathering
  Collect detailed information about the target systems BOS, RPF and MQTT-Broker to identify potential vulnerabilities and entry points.

- Step 2. Initial Access
  Gain initial access to MQTT-Broker.

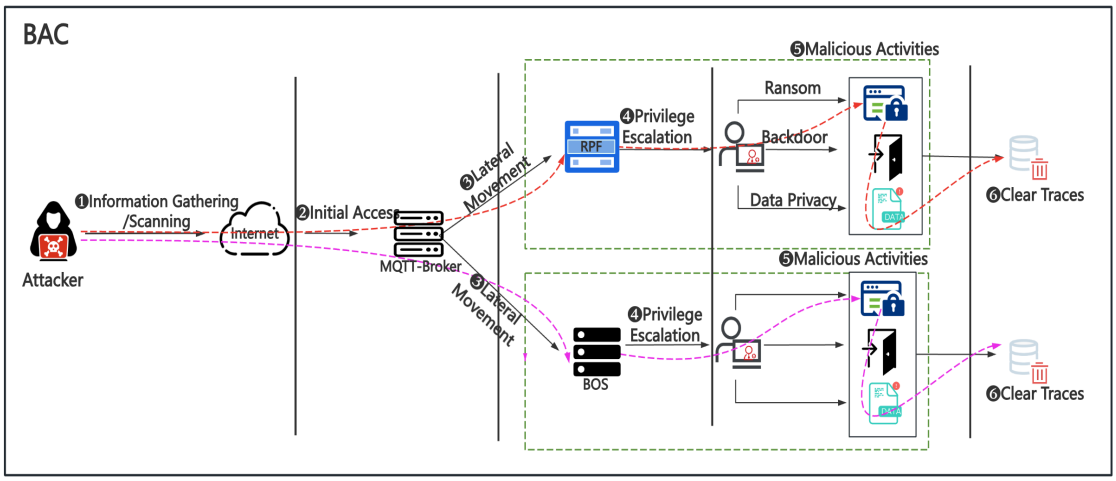

Figure 5.3: BAC attack scenario overview.

- Step 3. Lateral Movement
  Expand control within the network by moving laterally to BOS or RPF and infecting additional controllers.
- Step 4. Privilege Escalation Gain administrator-level access.
- Step 5. Malicious Activities Sends commands to elevator or robot execute Malicious Activity.
- Step 6. Clear Traces Removes all malicious software and cleans the system logs to hide the evidence of the attack.

5.2.4 Malware Attack Scenario

This section introduces malware attack scenarios designed for the simulator, focusing primarily on ransomware attacks. Ransomware is a type of malware that extorts money from its victims by encrypting the user's data or locking the user's system, sensitive data, or device and demanding a ransom to decrypt the data or unlock the system [30].

Ransomware includes both encrypted and non-encrypted ransomware, and the ultimate goal of the attackers is basically to obtain ransom. This is prone to cause great financial losses for the victim users or enterprises. We refer to [30, 31] to organize the common attack methods, the attacker usually through the following ways to attack:

1. Phishing emails: Attackers install ransomware on the target system by sending phishing emails, which usually come with malicious attachments or links

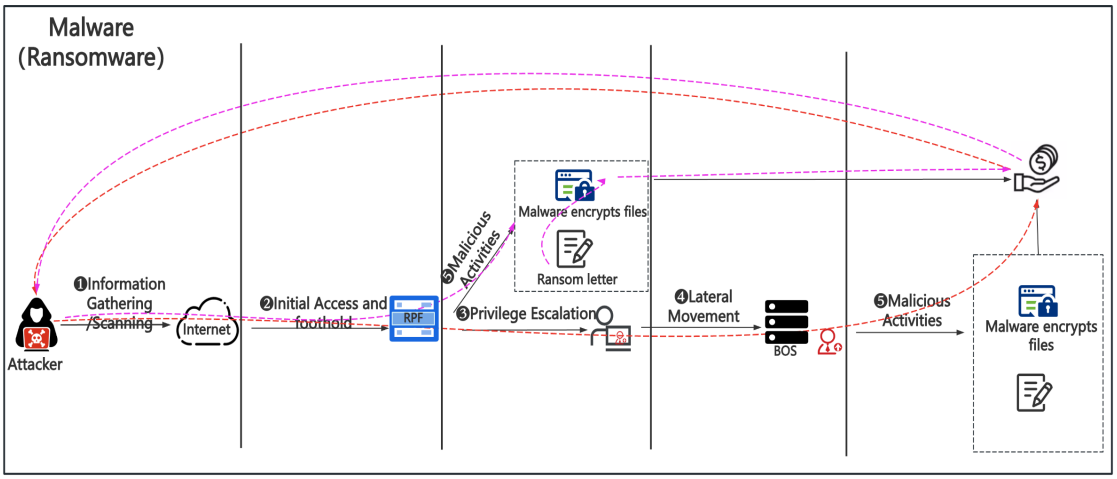

Figure 5.4: Malware attack scenario overview.

to trick victims into clicking on them.

2. Exploitation of vulnerabilities: Attackers exploit known or unknown software vulnerabilities to inject malicious code or implant malicious ransomware into the target system.

3. Malvertising: Attackers can spread ransomware on unsecured websites or ad networks, and the ransomware automatically downloads or executes in the background when users unknowingly visit these sites.

4. Bypassing Multi Factor Authentication: Attacker bypasses authentication by stealing and exploiting user credentials to compromise a user's system for malicious purposes.

In our system, as shown in Figure 5.4, I have designed a malware-based attack scenario, with ransomware as the core method. After gaining initial access, attackers can extend their control over the network by infecting other controllers to perform malicious activities, such as encrypting critical data or locking down system functions, forcing victims to restore their systems only after paying a ransom. Attackers typically need to go through the following steps to carry out an attack:

- Step 1. Information Gathering
  Collect information about the target systems and identify potential vulnerabilities.

- Step 2. Initial access and foothold
  After exploiting RPF and gaining access, then establish a foothold by using a remote-access tool.

- Step 3. Privilege Escalation
  Gain administrator-level access.
- Step 4. Lateral Movement
  Expand control within the network by moving laterally to BOS and infecting additional controllers.
- Step 5. Malicious Activities
  Execute Malicious Activity.

5.2.5 Social Engineering Attack Scenario

This section we will introduce an attack scenario designed for emulator based on human error and social engineering. Human error and social engineering refer to an attacker's ability to gain trust by exploiting human weaknesses, psychological manipulation, and other deceptive means to allow the target person to disclose sensitive information leading to security breaches or data leakage. Social engineering can bypass firewalls, intrusion detection, etc. to gain human trust and manipulate psychologically [32]. The core problem of this attack lies in the security issues caused by improper operation, misconfiguration or carelessness. People are often susceptible to psychological factors such as trust, curiosity, or fear, and inadvertently assist attackers in causing damage to systems. Common tactics used by attackers usually include the following:

1. Phishing attacks: Attackers can use phishing emails or malicious advertisements to lure people into clicking on malicious links or attachments, thereby stealing login credentials or other sensitive information.
2. Insider threat: An attacker can exploit the greed or dissatisfaction of an insider of the target of the attack to induce the insider to leak information.
3. Misconfiguration: An attacker can induce a user to misconfigure the system, resulting in a security breach.
4. Exploitation of trust vulnerabilities: attackers can impersonate technical or maintenance to gain the trust of the target of the attack to obtain internal privileges.

In our system, as shown in Figure 5.5, we designed a social engineering attack scenario. After gathering information about individuals and the company, the attacker gains access to an employee's PC through a carefully crafted scam. Subsequently, the attacker performs lateral movement to infect other controllers, expanding control within the network and executing malicious activities, such as taking control of the robots via control over RPF.

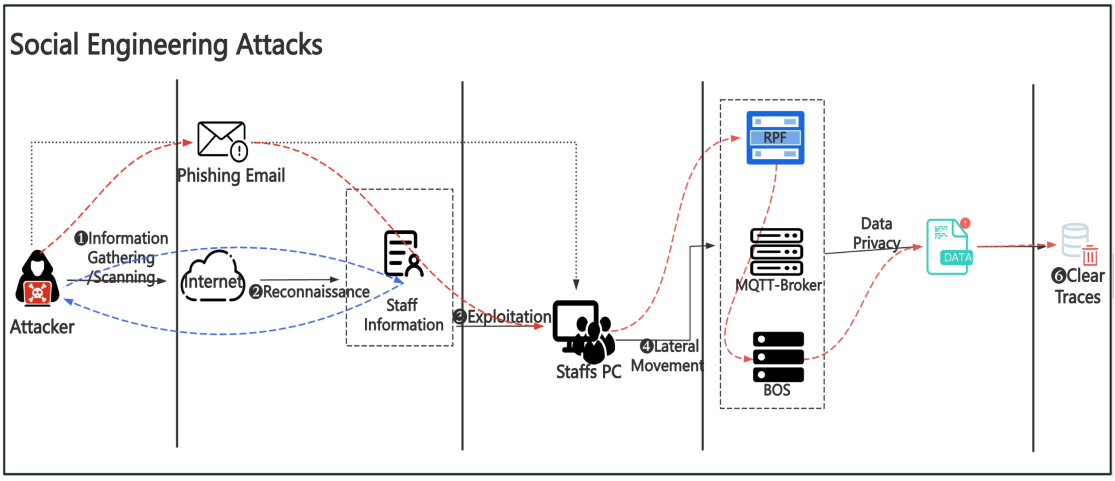

Figure 5.5: Social engineering attack scenario overview.

- Step 1. Information Gathering
  Collect information about the target individuals or organization to identify potential vulnerabilities.
- Step 2. Reconnaissance
  Gather information about personal,professional and company.
- Step 3. Exploitation
  Perform specific attacks to obtain sensitive information or access.
- Step 4. Lateral Movement
  Expand control within the network by moving laterally to RPF and infecting additional controllers.
- Step 5. Data Breach
  Steal sensitive data from organization.
- Step 6. Clear Traces
  Removes all malicious software and cleans the system logs to hide the evidence of the attack.

5.3 Security Measures

The smart building and Internet of Things (IoT) device interaction also brings more vulnerabilities to the smart building. Attackers tend to attack smart buildings through these potential threats, resulting in a significant risk to the system.

In order to effectively address these challenges, this section discusses a series of security measures. We refer to the literature [19, 18, 33, 34, 35] and focus specifically on the security of infrastructure and IoT devices in the smart building domain. We have classified and analyzed security measures, adopting a classification method similar to Table 5.6. In Table 5.7, we use the CIA (Confidentiality, Integrity, Availability) framework to categorize threats and and organize the corresponding security measures.

Regarding the use of the table, we need to refer to Tables 5.4, 5.5 and 5.6 where we can see what kind of impact the corresponding attack will face, and then we can find the security measure corresponding to this impact against Table 5.7. For example, if the system encounters a MITM attack, we can refer to Table 5.4 to find the corresponding Scope of Impact, one of which is Impact is Data tampering, and then we can find the same Data tampering label under Threats in the Countermeasures Table, and there are corresponding Security measures in the last column of the same row. One of the methods is Encryption of transfer channels, which is the countermeasure implemented in the MITM attack that we will discuss in later chapters. The query for the countermeasures for the other attacks and impacts is the same.

In the later sections, we will conduct experiments on attack testing and countermeasure validation in the simulator to ensure the effectiveness of these measures. By implementing these countermeasures, we aim to enhance the overall security profile of IoT devices in smart buildings and protect them from potential threats. These measures not only include protection at the technical level, but also involve aspects such as security management and personnel training. By synthesizing these measures, we can build a more secure, reliable and resilient smart building environment.

Table 5.7: Analysis of possible security measures for cybersecurity threats

Classification	**Threats**	**Details**	**Security measures**
Confidentiality / Integrity	Data leak/ Information leak	Attacks can obtain sensitive data, posing a risk of data leakage	1. Data encryption 2. Implement data access controls 3. Introduce data breach detection and response systems (DLP) For highly confidential data, implement encryption and key management to ensure only authorized users can access the information.
	Privilege leakage	The attacker gains system administrator privileges through a vulnerability and then takes complete control of the system.	1. Regular vulnerability scanning and patching 2. Multi-factor authentication (MFA) - Implement options like Apple Touch ID, security tokens 3. Enforce the principle of least privilege for access 4. Monitor and audit the use of privileges
	Log leak	Obtain system operation logs and analyze the system's operational status and potential security vulnerabilities.	1. Encrypt logs 2. Implement log access controls 3. Perform de-identification of sensitive log data 4. Conduct regular log audits - Adopt a Privileged Access Management (PAM) solution to manage elevated privileges administrator rights.
	Data tampering	Elevator's, robot's status data has been tampered with	1. Encryption of transfer channels - Define data exchange channels between devices and ensure system owners securely accept them. 2. Encryption of data in transit - Monitor production data both in storage and in transit to identify potential unauthorized modifications. 3. Use of digital signatures (non-repudiation) 4. Implementation of monitoring and detection systems - Establish a baseline and monitor for anomalies and compliance with that standard.
	Embedding a backdoor	Plants a backdoor, enabling continuous monitoring and tampering of system data.	1. Regular system scans 2. Detection and removal of backdoors - Implement software signing/ checksum management. 3. Strengthen access controls 4. Establish configuration baselines across business systems - Harden systems by removing unnecessary administrator accounts and closing unnecessary public-facing ports. - Verify integrity based on a root of trust, digital signatures, and embedded identifiers to ensure manufactured devices' integrity is assessed and verified. - To maintain system manageability, limit the number of protocols implemented within specific environments, and disable all unused default network services.

Classification	Threats	Details	Security measures
Confidentiality / Integrity	Tampering with system logs	The attacker tampers with system logs to hide their activities or mislead investigators, ensuring no trace of data leakage is discovered	1.Log monitoring and tampering detection - Protect data in use and during transfer. 2. Regular log audits - Protection of data at rest can be achieved through access control and authentication requirements. For critical data, the implementation of encryption algorithms is recommended.
	System configuration errors	Malicious changes to system settings result in degraded system performance or loss of functionality	1. Monitoring system configuration 2. Implementation of change management processes 3. Implementation of configuration management - Change default passwords and usernames during trial runs or initial use. - Include information such as IP addresses, physical locations, hosts, current firmware/OS versions, and used communication protocols. 4. Availability management processes 5. Backup settings and recovery plans - Ensure the continuity of system operations by creating a Business Continuity Plan (BCP) and Disaster Recovery Plan (DRP) to address security issues. 6. Regular review of access permissions - Periodically review access permissions and promptly remove access rights after employees transfer or leave the company
	Incorrect operational guidance	Misleading maintenance personnel to perform unsafe or incorrect operations	1. Education for maintenance personnel (security training and awareness enhancement) - Conduct cybersecurity training for employees. 2. Implementation of a double-check process 3. Provision of accurate operation manuals
	Expiration of security policies	Attacker disables data middleware security policies, allowing transmission without verification or authorization.	1. Implement security policy management. 2. Regularly review and update security policies. 3. Monitor security policy changes use automated tools. 4. Enforce multi-factor authentication and strong password policies.
Non-compliance	Equipment Maintenance and Replacement	Non-compliance with safety standards leads to investigation and sanctions by regulatory authorities.	1. Conduct regular audits. 2. Maintain safety standards. - Perform cybersecurity acceptance tests (e.g., FAT, SAT, pre-production penetration tests) during various validation activities and product lifecycle stages. 3. Ensure compliance with laws and regulations.

Classification	Threats	Details	Security measures
Availability	System stoppage and incorrect operations	-Alters commands, cause ELV to stop at an unintended floor. -Robot receiving tampered path instructions, may follow wrong route. -Performance bottleneck (network/server resource)	1. Strengthen access control. 2. Validate and authenticate commands. - Establish secure security associations between communication devices using proven encryption algorithms to ensure mutual authentication, integrity, and confidentiality. 3. Install emergency stop buttons.
	Physical collisions and damage	Incorrect instructions may cause the robot to collide with people or objects, leading to physical damage or injury.	1. Implement collision detection 2. Install safety sensors. 3. Develop and follow safe operating procedures. 4. Provide regular training for operators.
	Frequent restarts	-Due to incorrect instructions, the elevator repeatedly opens and closes doors, and restarts. -Incorrect instructions cause the robot to restart frequently, eventually stopping.	1. Validate instructions. - Establish secure security associations between communication devices using proven encryption algorithms to ensure mutual authentication, integrity, and confidentiality. 2. Limit the number of restarts. 3. Monitor the system.
	Operating without supervision	The ELV/ROB may start abnormally without supervision, wasting energy and potentially causing safety issues.	1. Detect unsupervised operation. 2. Implement energy management systems. 3. Add abnormal stop functions. 4. Monitor the system.
	Failure in emergency response	- In special situations, ELV may receive stop command, preventing personnel from evacuating properly. - In emergencies, ROB may fail to stop as expected, increasing risks due to evacuation failure.	1. Establish emergency response procedures. 2. Conduct regular evacuation drills. 3. Prepare detailed emergency manuals.
Operation / Management	Equipment repair and replacement	- Frequent misuse requires elevator repairs or replacement. - Frequent misuse necessitates frequent robot repairs or replacement.	1. Perform regular maintenance. 2. Monitor operation logs. 3. Implement preventive maintenance measures.
	Business Interruption and Delays	- Business interruption caused by elevator stoppage. - Business interruption caused by robot stoppage.	1. Develop an emergency response plan. 2. Implement system redundancy.

Chapter 6

Experiment Results

In this chapter, we will test the MITM and DDoS attack scenarios designed in Section 5.2 on the emulator and respond to these attack scenarios by implementing appropriate security measures. We will select suitable measures and test the attacks again after applying these countermeasures on the simulator to confirm the effectiveness of these countermeasures. Through this process, we aim to validate and optimize our security measures to ensure that they can protect IoT devices in smart buildings from damage when actual attacks occur. This results in a more secure and reliable smart building environment.

Since the BAC Attack Scenario, Malware Attack Scenario and Social Engineering Attack scenarios are closely related to user participation and personnel management, and simulation experiments of user behavior are difficult to be carried out effectively at the current stage, the current study has only conducted preliminary tests on BAC assuming that an administrator user exists. BAC was preliminarily tested, however, the experiments are not perfect enough. For the malware attack and social engineering attack scenarios, it is difficult to visualize the results due to the lack of personnel involvement, so this research does not involve the discussion of these two scenarios for the time being. We plan to extend the simulator for testing more attack scenarios in future research. For example, by introducing a personnel simulation module, richer simulation tests can be conducted to more comprehensively evaluate the security and response capabilities of the system.

6.1 MITM Attack Scenario

In this section, we will test the MITM attack scenarios designed in Chapter 5. As shown in Figure 6.1, in our system, the attacker's target for eavesdropping can be either BOS or RPF. Here, we will provide a more detailed explanation of

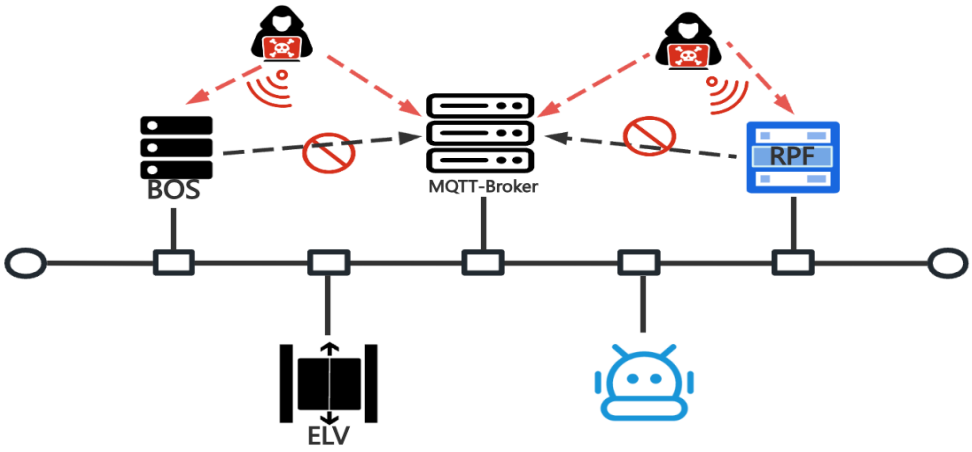

Figure 6.1: MITM intercept points

the attack scenarios. According to the designed scenarios, we have added MITM scenario testing to the security attack module in SBCSE. Considering the risks of network attack testing, we have omitted the earlier steps. In the emulator, we assume that the network has been successfully scanned and eavesdropped upon, and we conduct our experiments based on this assumption. That is to say, the part enclosed in green in Figure 5.1 is the main segment of our experiment.

In this section, we will introduce a MITM experiment targeting BOS, with an attack process that includes hijacking communication, intercepting data, and fabricating messages. As shown in Figure 6.2, we have emulated a scenario where RPF sends an "Open" command, which is supposed to be forwarded by BOS to the elevator to open the door.

6.1.1 Experiment 1 – MITM Attack Targeting BOS

In this scenario, the attacker intercepts and disrupts BOS's communication, preventing BOS from successfully forwarding the "Open" command to the elevator, resulting in the elevator door not opening. At the same time, the attacker fabricates a message and sends a false "Open success" response to RPF. This leads RPF's state machine to make an incorrect judgment and sends a "Getting On" command to the robot. As a result, with the elevator door still closed, the robot attempts to enter the elevator, which could cause a collision between the robot and the elevator, potentially leading to damage to physical assets.

Figure 6.3 presents the log analysis results after a successful implementation of the MITM attack. The logs indicate that the robot ultimately failed to enter the elevator. Based on observations of real robotic vacuum cleaners, we have

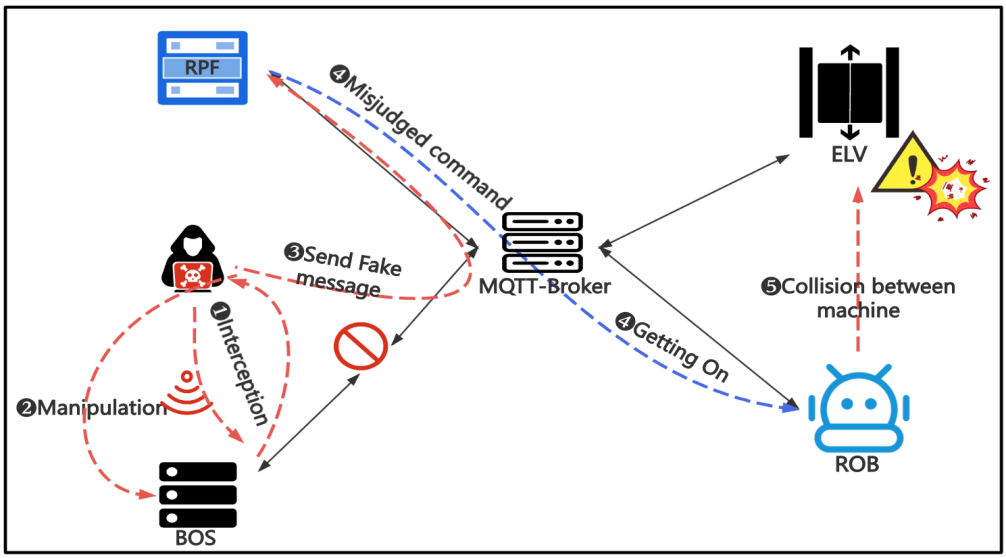

Figure 6.2: MITM attack targeting BOS.

considered that the robot is equipped with a camera to enable it to assess the status of the door and to report the failure to enter along with the reasons for the failure.

6.1.2 Experiment 2 – MITM Attack Targeting RPF

Similarly, we have designed a comparable MITM experiment targeting RPF. As depicted in Figure 6.4, we have emulated a scenario where the interception target is RPF. The attacker manipulates RPF, taking control to alter the internal code of the state machine, send unexpected commands, and provide feedback that appears to be normal. For instance, instead of forwarding the "Open" message to the ELV, the attacker causes RPF to send a "GettingOn" command to the robot. This emulates a similar scenario in which the elevator door is closed, prompting the robot to attempt entering the elevator.

Figure 6.5 displays the log analysis results following the successful execution of the MITM attack. The logs reveal that, consistent with the previous scenario, the robot ultimately fails to enter the elevator. From the results, it can be seen that the MITM attack experimental scenario has affected the normal communication of the system.

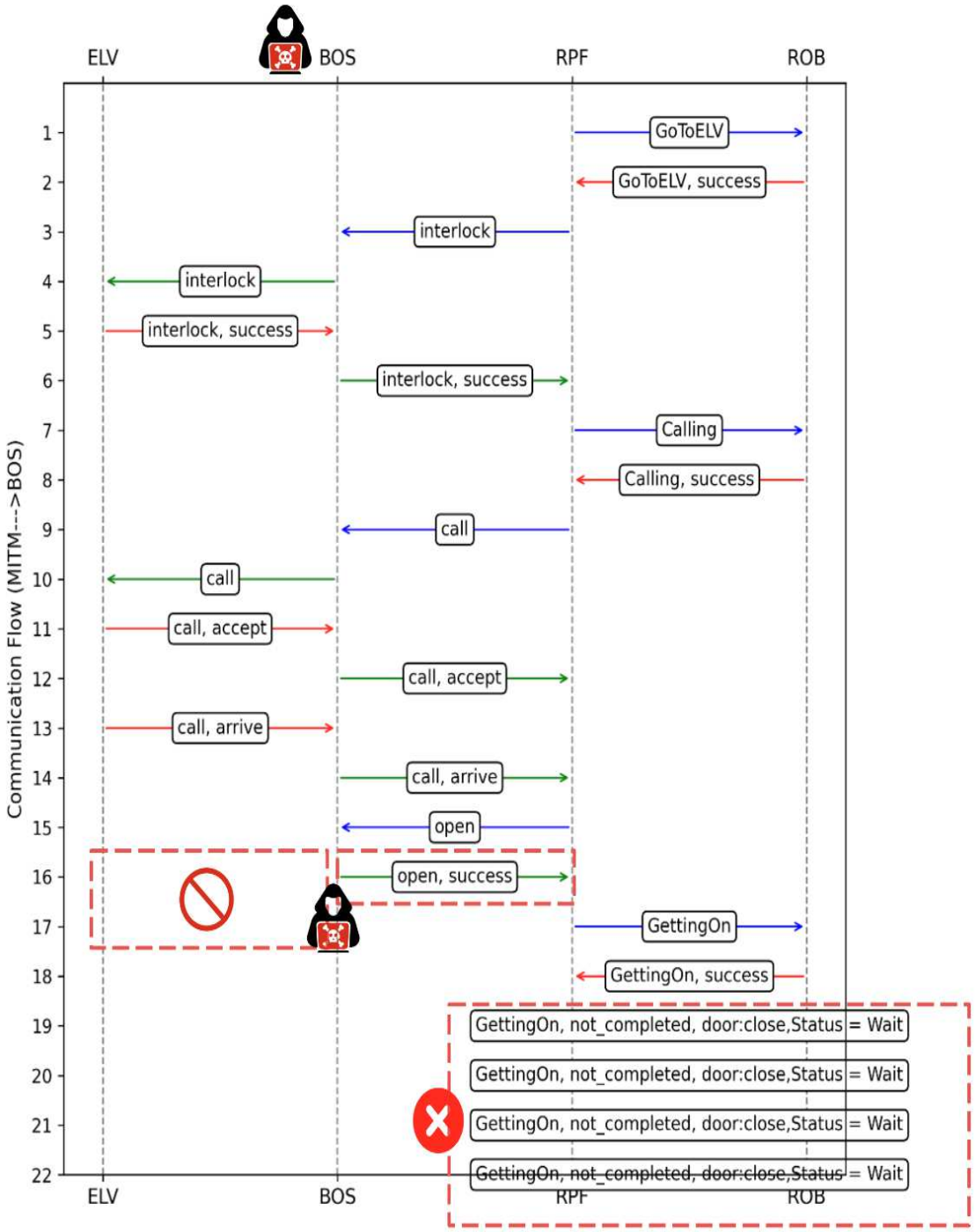

Figure 6.3: Experiment results for MITM attack targeting BOS.

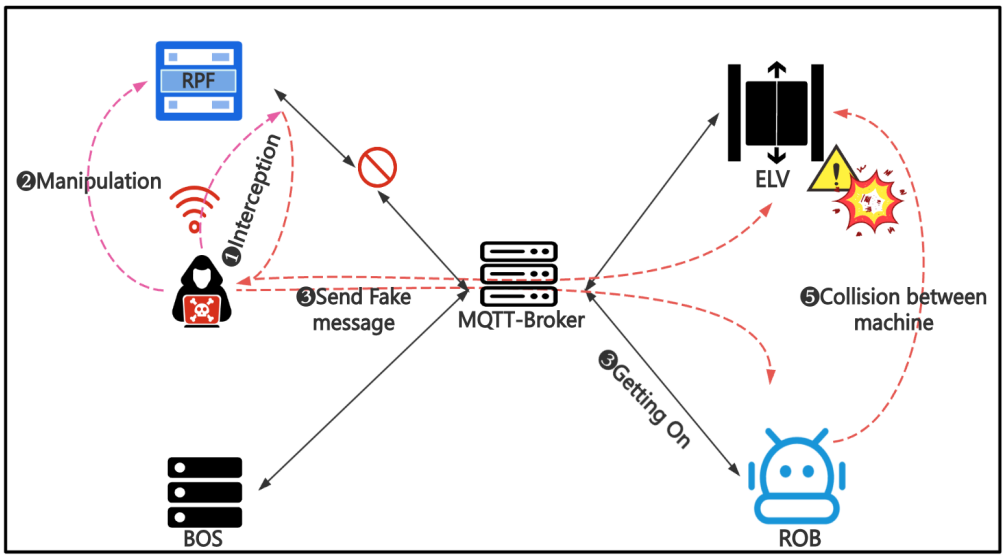

Figure 6.4: MITM attack targeting RPF.

6.1.3 Experiment 3 – MITM Attack Targeting BOS With Security Measures

Referring to the security measure Table 5.7, we understand that encryption of the communication forwarding channel is a key measure to prevent data tampering in MITM attacks. To this end, our system has integrated the MQTTS protocol, which is a secure communication protocol that combines the MQTT protocol with TLS encryption technology. The standardized name according to IANA is "secure-mqtt" [36]. To enable TLS encryption, we replace the default non-encrypted port 1883 with encrypted port 8883 to ensure secure data transmission. With the application of the MQTTS protocol, all transmitted data will be protected by TLS encryption, effectively preventing the possibility of attackers eavesdropping on or tampering with the data.

In the test, we specifically conducted a MITM attack targeting BOS using the MQTTS protocol, with the results shown in Figure 6.6. The test results clearly show that even under a emulated MITM attack environment, the communication tasks and command order are remains normal and unaffected. This result confirms that the MQTTS protocol can effectively prevent data tampering caused by MITM attacks in practical applications, ensuring the security and integrity of communications and data.

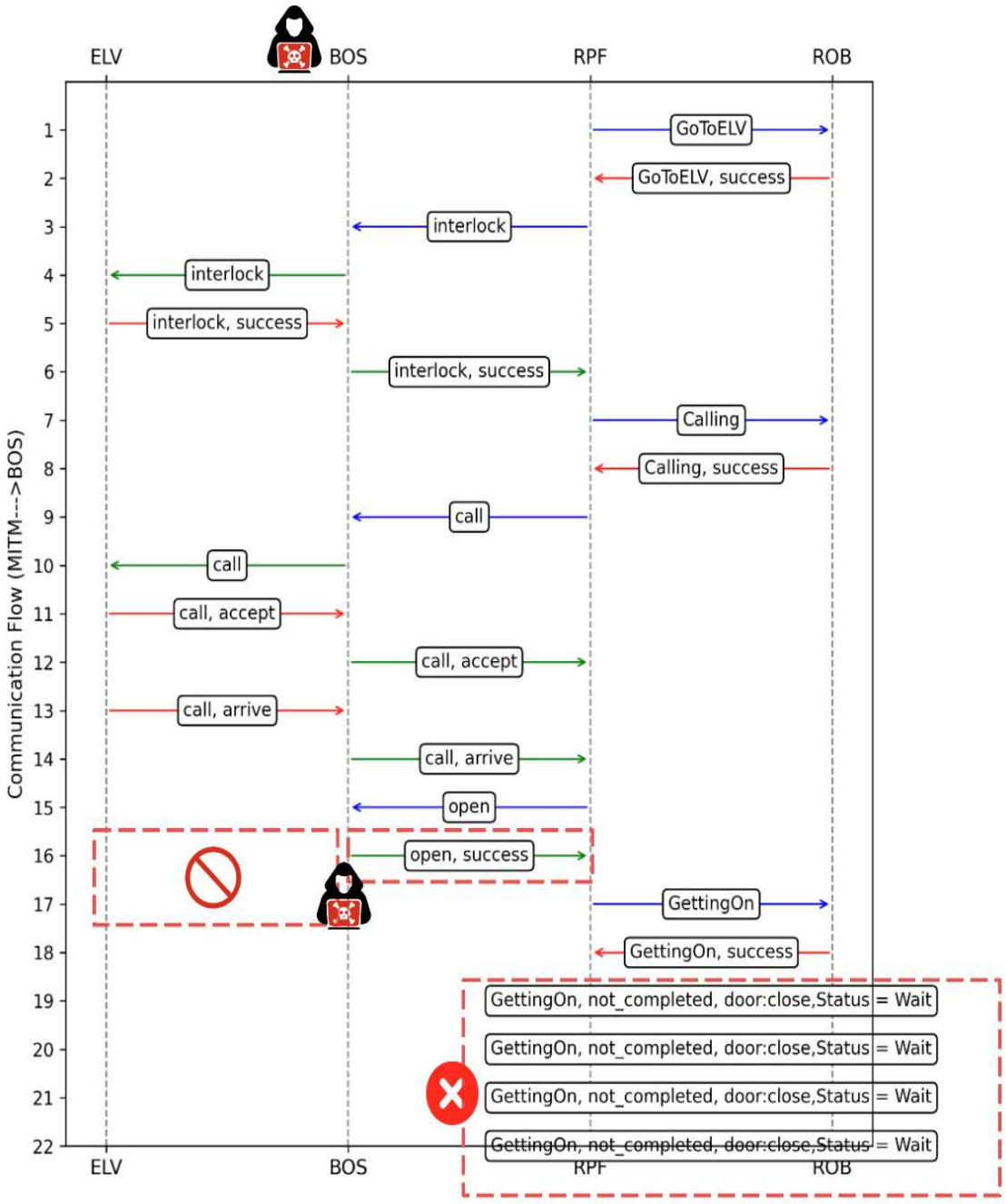

Figure 6.5: Experiment results for MITM attack targeting RPF.

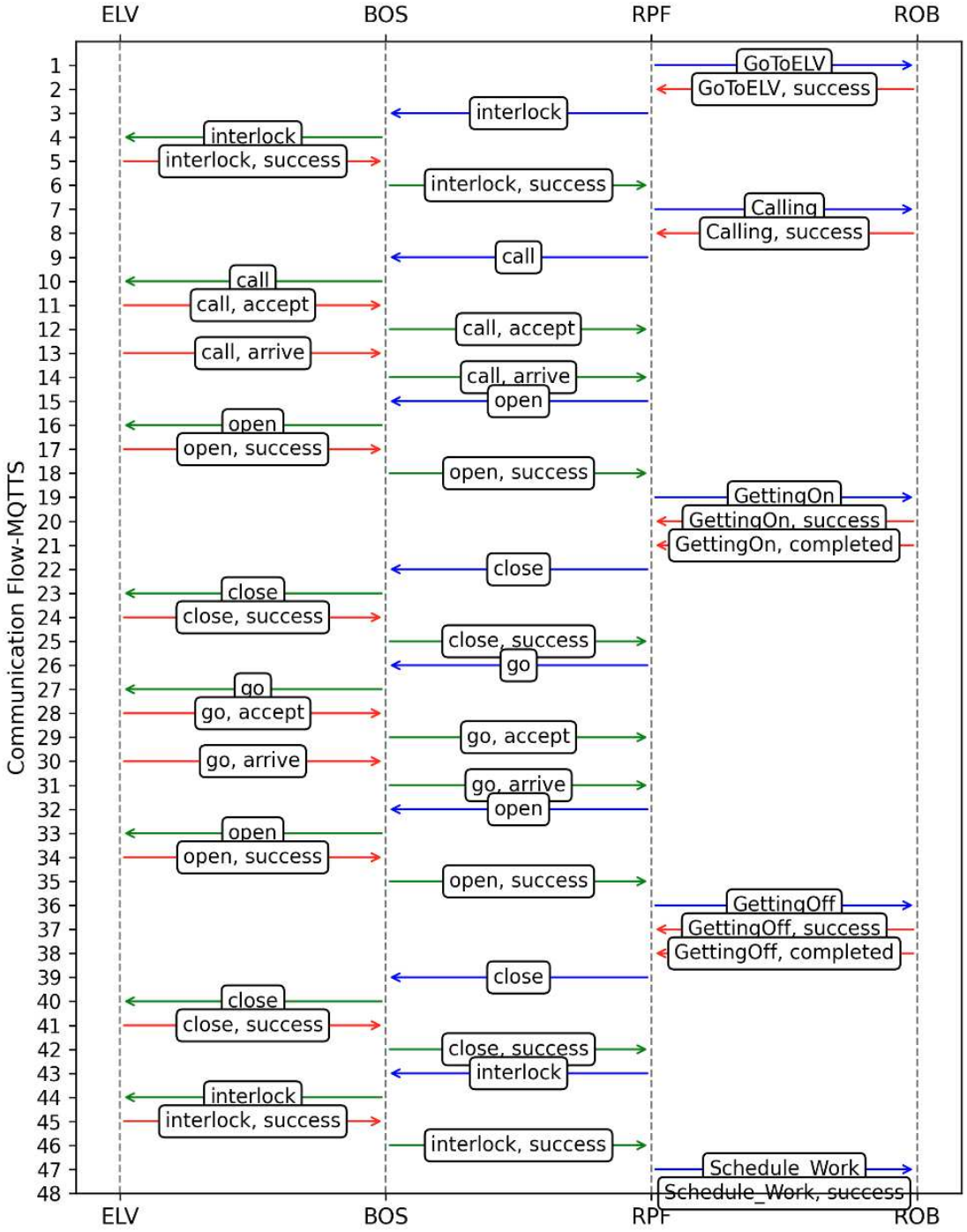

Figure 6.6: Experiment results for MITM attack targeting BOS when using MQTTS as security measure.

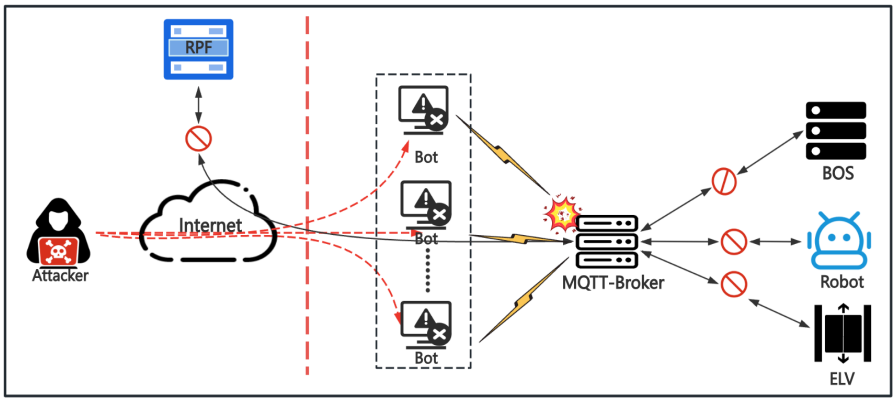

Figure 6.7: DDoS attack targeting MQTT-Broker.

6.2 DDoS Attack Scenario

In this section, we explore the testing of DDoS attack scenarios. As shown in Figure 6.7, we mainly use MQTT-Broker as the attack target for testing. The following is a detailed description of the attack scenarios. To test the DDoS attack, we have integrated two DDoS attack methods in the security attack module: publish flooding and connection flooding. we have chosen a local mosquitto for our Broker for testing. The following is a detailed discussion of these two attack scenarios.

6.2.1 Experiment 1 – DDoS Connection Flooding Attack

In this section, we will introduce an experiment targeting Broker with Connection Flooding. Connection Flood attacks are a type of denial-of-service attack that establishes a large number of connection requests, leading to service refusal. This attack exploits the server by initiating a large number of connections and not releasing them for a long time, thus occupying server resources, causing server efficiency to decrease or even exhausting resources, making it unable to respond to other clients' connection requests [37]. The attack level is at the application layer, where attackers emulate multiple clients initiating connection requests simultaneously, with the aim of exhausting the Broker's connection resources, causing service unavailability and affecting the communication of other components in SBCSE.

The attack process generally includes three steps: building a botnet, remote command and control, and launching an attack. When a large number of fake requests and malicious traffic hit the Broker at the same time, exceeding its processing capacity, it may cause the service to crash or become extremely slow. In order to observe and understand how the system reacts and behaves in the face

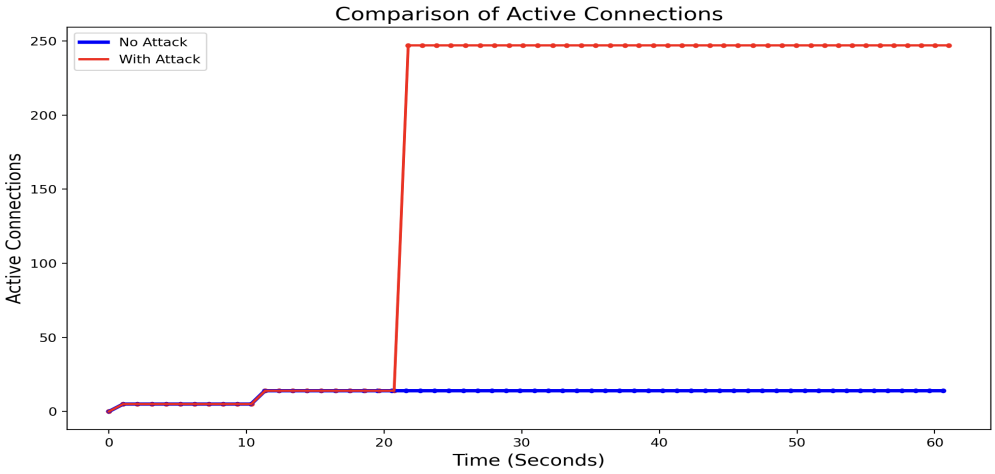

Figure 6.8: DDoS connection flooding experiment targeting MQTT-Broker: Comparison of active connections.

of a large number of simultaneous connection requests. We conduct these tests in SBCSE, where we created a large number of botnet client and Broker connections, and found that once the attack was launched, the server blocked due to the large number of connections occupying server resources, and the normal communication tasks of the components in the system could not be accomplished.

In order to better observe the experimental results, we add the ConnectionMonitor module, which can monitor the number of active connections on the broker side in real time and record them in the log. We test the no attack case and connection flooding attack case respectively, and visualize and analyze the log. From Figure 6.8, we can clearly see that in the case of no attack, the number of active connected clients detected from the broker side always stays below 15 and remains relatively stable after the system has been running for some time. Meanwhile, by observing the communication logs between components, we confirm that the communication between all components is in a normal state.

However, in the case of an attack, observing the part of the red line in the figure, at first the number of Active Connections is the same as in the case of "No attack", but as the attack starts, the number of active connections starts to rise significantly. Due to the large number of malicious connections established by the attacker with the broker, the broker was unable to process and respond to normal communication requests in a timely manner. This led to the failure of message forwarding between some components, and the system tasks could not be carried out normally.

6.2.2 Experiment 2 – DDoS Publish Message Flooding Attack

In this section, we describe the Publish flooding experiment targeting the MQTT Broker. MQTT Publish flood attack [38] is an attack against the MQTT protocol in which the attacker interferes with the MQTT Broker by sending a large number of Publish messages to the Broker with the goal of exhausting the Broker's resources or network bandwidth, thereby interfering with its normal service. In our experimental setup, we simulate this attack scenario by sending a large number of command packets to a customized attack topic. Here we set the number of simulated attack clients for the experimental settings to be 1000 and the frequency of data transmission to be 0.001. In this way, we are able to observe how the Broker behaves under high load conditions and how it affects normal communication.

In order to observe the experimental results, we have added the MQTTBroker-Monitor module, which can log the success rate of requests for messages received by the MQTT Broker, thereby facilitating our effective evaluation of the experiment. We conducted tests for both attack and non-attack scenarios and visualized the relevant logs. The formula for calculating the response rate of MQTT-Broker is as follows, where "Sent" represents the amount of data forwarded from the Broker end, and "Received" represents the amount of data published by the client and received by the Broker.

$$\text{Response success rate} = \frac{\text{Sent}}{\text{Received}} \times 100\% \tag{6.1}$$

Since we monitor some of the MQTT managed data, the amount of "Sent" data will be larger than the amount of "Received" data, resulting in a Response rate of more than 100%. In order to calculate more accurately, we will subtract the number of subscriptions to the management data. In addition, since our messages are QoS1, the MQTT Broker sends PUBACK or PUBREC acknowledgement messages to the publisher, and these acknowledgement messages may also increase the volume. At the same time, the case of response delay may also cause the number of packets sent to be more than the number of packets received. Therefore, we will combine the message logs of each component to observe the experimental results.

The comparison result of "Response success rate" is shown in Fig 6.9. In the case of no attack, the "Response success rate" of the Broker stays at 100%, which indicates that the Broker's processing of packets is normal. In the case of attack, the "Response success rate" decreases significantly and floats around 1.38%, which is close to 0. This is because the attacker sends a large number of packets to the Broker, exhausting its resources and interfering with the normal message sending and receiving process. We combined the message forwarding logs of the observation components and confirmed again that the normal communication was affected in

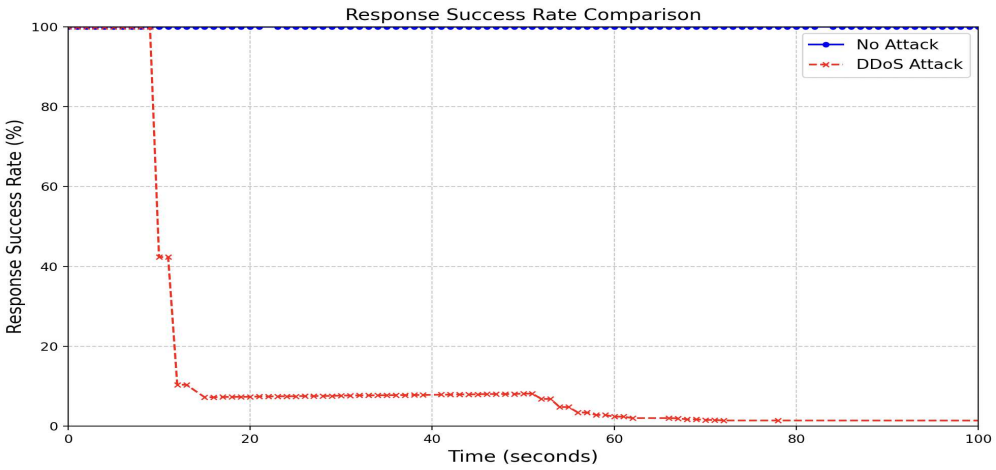

Figure 6.9: DDoS publish message flooding experiment: Comparison of MQTT-Broker response success rate.

the case of DDOS attacks.

6.2.3 Experiment 3 – DDoS Publish Message Flooding Attack With Security Measures

Referring to the countermeasures in Table 5.7, we understand that DDoS attacks bring serious impact on the availability of the system, which is mainly reflected in the performance bottleneck of network and server resources. In order to effectively defend against such attacks, we adopt the strategy of enhanced access control. Specifically, we implemented an allowlist setting in the connection with the MQTT Broker, so that only the users on the allowlist can establish a connection with the Broker and communicate normally. This means that users who are not on the list at the time of connection are filtered. In our system, only the individual components are set up to connect normally. This approach can effectively prevent illegal connections from external IPs, thereby reducing the potential risk of DDoS attacks. By setting up a allowlist, we are able to ensure that only authenticated and trusted clients are able to access the MQTT Broker, which not only improves the security of the system, but also reduces resource consumption due to unauthorized connections.

We once again conducted a DDoS Publish Flooding attack test on the Broker with allowlist access control enabled, and the test results of the message communication are shown in Figure 6.10. The test results clearly show that the communication process remains normal and is not affected in any way under the simulated DDoS attack environment. The response success rate of the MQTT Broker is

shown in Figure 6.11, from which we can see that it remains at 100% as in the case of no attack, which confirms the effectiveness of the security measures we have taken.

Additionally, there are other prevention methods that can further enhance the system's security. For example, a study proposed a detection method for DDoS flood attacks on SDN based on machine learning [39]. This method can effectively classify and detect attacks. In addition, reducing the server timeout value [40] can also reduce the intensity of the attack. We will test more countermeasures in future research.

6.3 BAC Attack Scenario

In this section, we will perform a preliminary test of the BAC scenario. As shown in Figure 5.3, the target of unauthorized access by an attacker in our system can be either BOS or RPF. Here, we will explain the attack scenarios in more detail. According to the designed scenarios, we add the test of BAC scenarios to the security attack module in our system. Considering the risk of network attack testing, as in the MITM Attack Test above, we omit the first few steps, and we assume that we have successfully scanned the network and initial access. And because the BAC Attack Scenario is related to the personnel, and the simulation of user behavior is not yet implemented in the current simulator, the current experiments are preliminary tests of BAC under the assumption that the administrator user exists, and the experiments are yet to be improved.

6.3.1 Experiment 1 – BAC Attack Targeting BOS

In this section, we introduce a BAC attack experiment targeting BOS. The attack process involves lateral movement to BOS, infecting the controller to expand control within the network, and obtaining administrator-level access to carry out malicious activities.

As shown in Figure 6.12, we emulated a scenario where the attacker uses default usernames and passwords to access and control BOS. It is assumed that the control elevator does not open the door when personnel and robots enter the elevator. When RPF sends an elevator "Open" command, the attacker manipulates BOS's system to forward a "close" command to the elevator instead. Once the elevator returns a "close success" response to BOS, BOS forwards a modified response indicating "open success" to RPF. This causes RPF's state machine to make an incorrect judgment and send a "Getting Off" command to the ROB. At this time, the elevator door is still closed and the robot is trying to exit the elevator, which may cause the robot to collide with the elevator. Or the control of the elevator

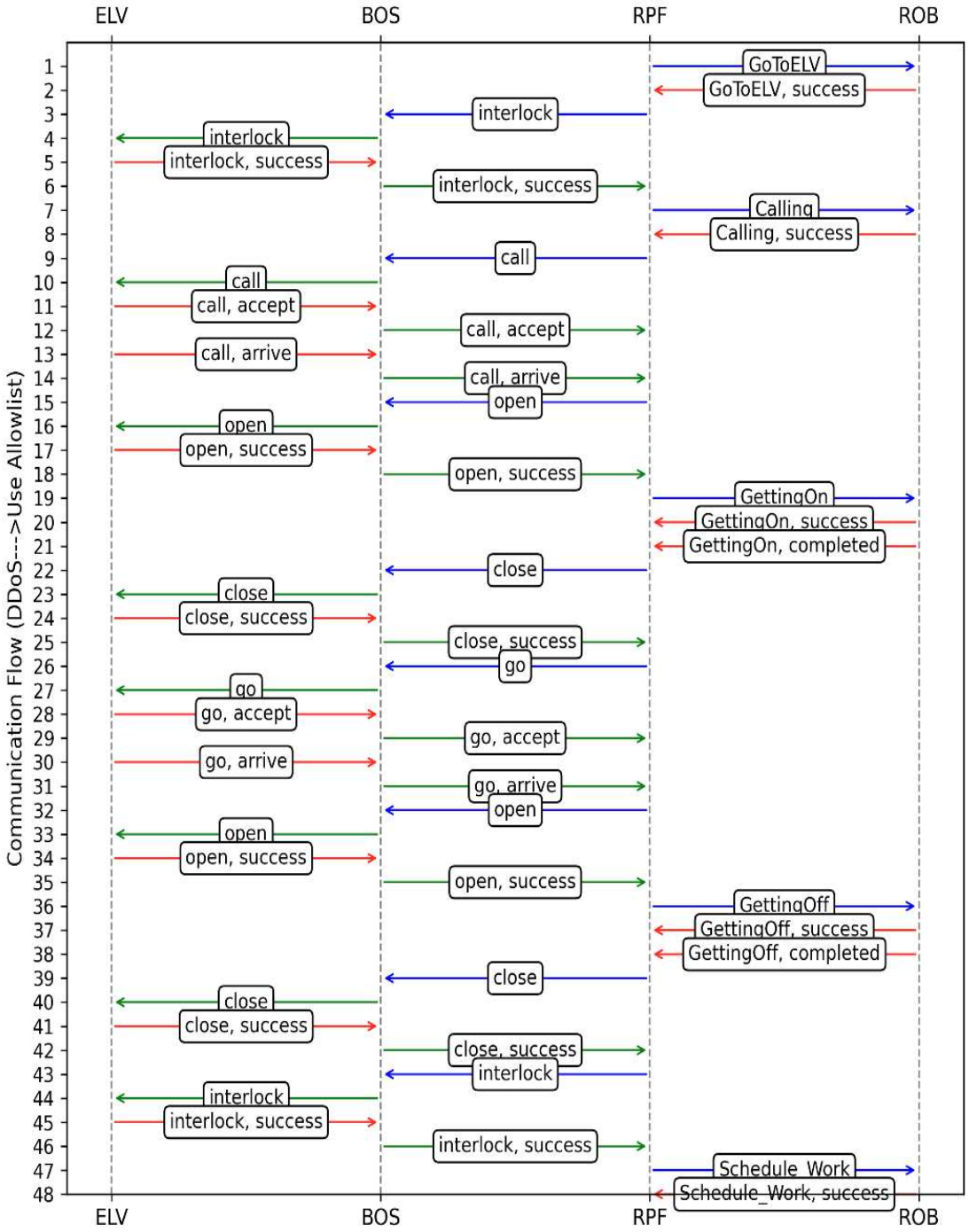

Figure 6.10: Experiment results for DDoS publish message flooding targeting MQTT-Broker when using allowlist as security measure.

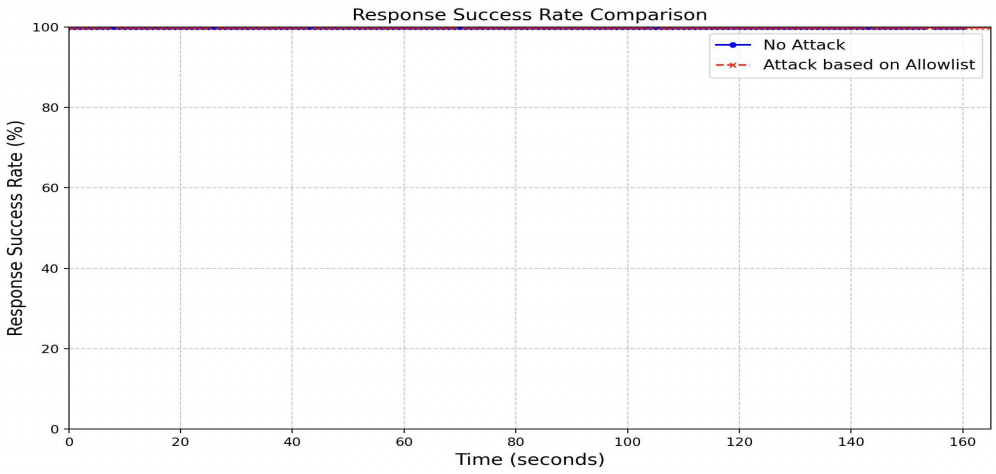

Figure 6.11: DDoS publish message flooding experiment: Comparison of MQTT-Broker response success rate when using allowlist as security measure.

does not open the door when the personnel is in the elevator, which causes panic and service interruption caused by the security problems.

Figure 6.13 shows the log analysis results after successfully implementing the BAC attack. From the log, we can see that in the case of BAC attack, the robot failed to exit the elevator, and the elevator door was always closed. The communication results are consistent with the scenario we designed.

6.3.2 Experiment 2 – BAC Attack Targeting RPF

Similarly, we designed a similar BAC attack experiment with RPF as the target. As shown in Figure 6.14, we emulated a scenario. Assuming that the target of the attacker's control is RPF, the attacker controls RPF to tamper with the internal code of the program, such as controlling RPF to send a "close" command to the elevator and then sending a "GettingOff" command to the robot. This could emulate the same effect as having the robot try to get out of the elevator when the doors are closed.

Figure 6.15 shows the results of log analysis after the successful implementation of the BAC attack. As can be seen from the logs, the same bot eventually fails to successfully exit the elevator. The communication results are consistent with the scenario we designed.

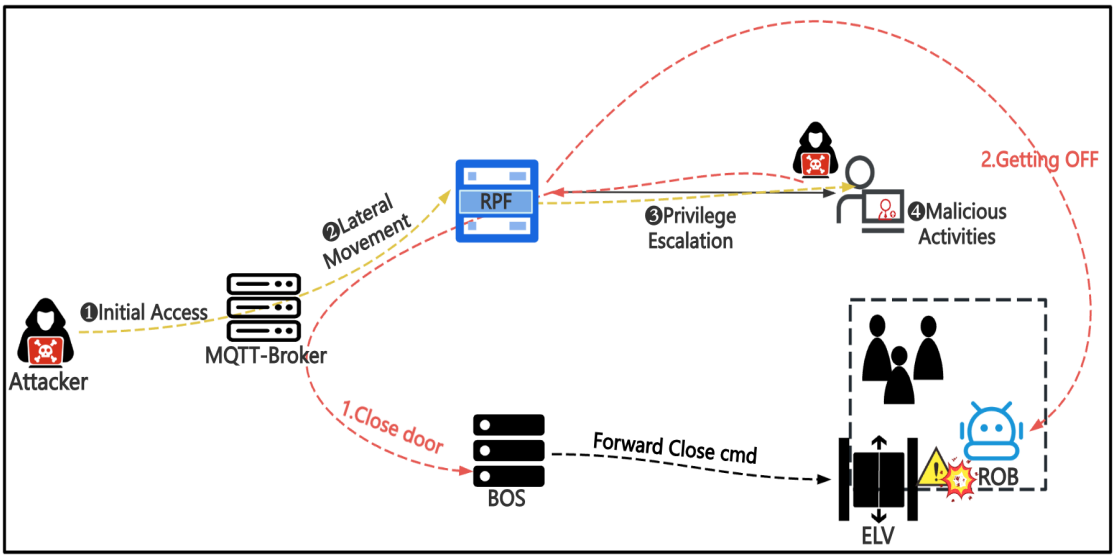

Figure 6.12: BAC attack targeting BOS.

6.3.3 Experiment 3 – BAC Attack Targeting BOS With Security Measures

Referring to the measures in Table 5.7, we understand that in order to prevent BAC attacks, the system must be designed and implemented with sufficient strength and rigor in access control. Of course using MQTTS is also one of the methods, and since the use of this method was discussed in the MITM attack above, here we discuss the other methods. For this reason, here we have adopted an encrypted authentication to prevent unauthorized access by attackers. Specifically, we added login authentication for the vulnerable components BOS and RPF. However, since an attacker can login to the system directly through the default weak password or other ways. So we use the bcrypt algorithm to encrypt the user account password and verify the registration. We use this method to strengthen in the storage and verification of passwords. Ensure that every request is validly authenticated and only authenticated users can access the system.

For the case of using encrypted authentication, we again conducted a BAC attack test targeting BOS. The test results are detailed in Figure 6.16. The test results clearly show that under the emulated BAC attack environment, the communication process remains normal and is not affected in any way. This result confirms the effectiveness of the countermeasures we have taken.

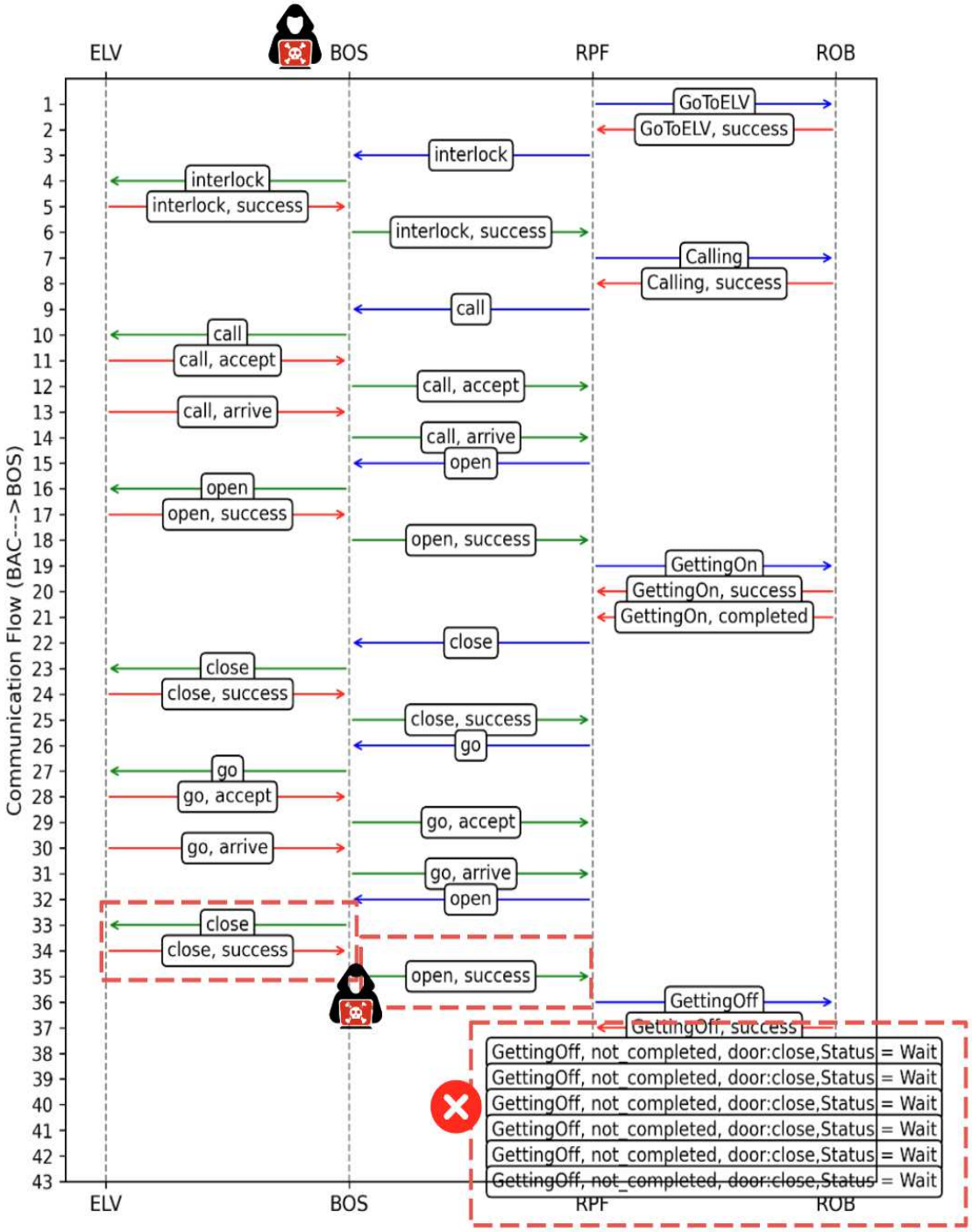

Figure 6.13: Experiment results for BAC attack targeting BOS.

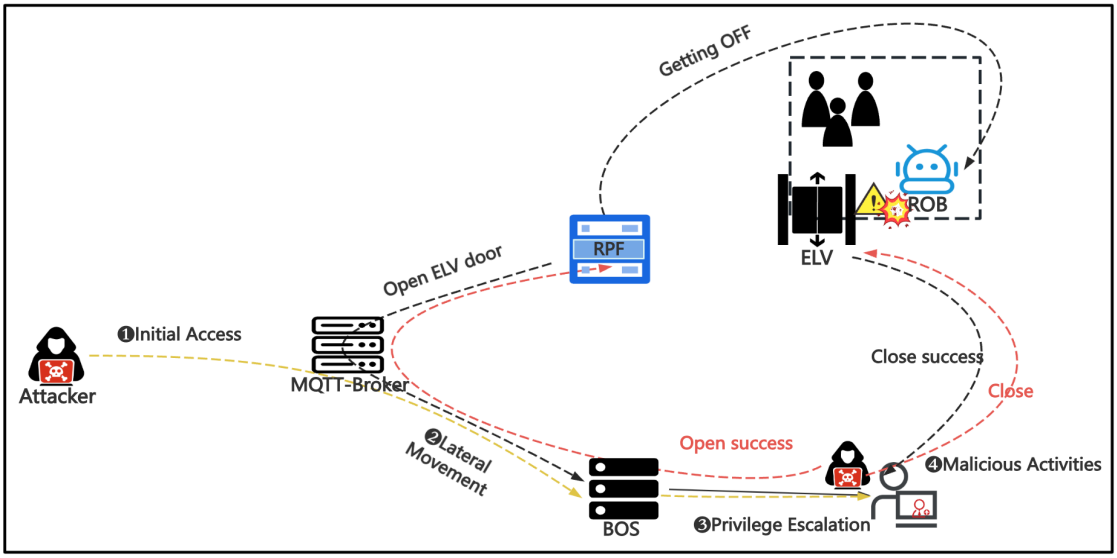

Figure 6.14: BAC attack targeting RPF.

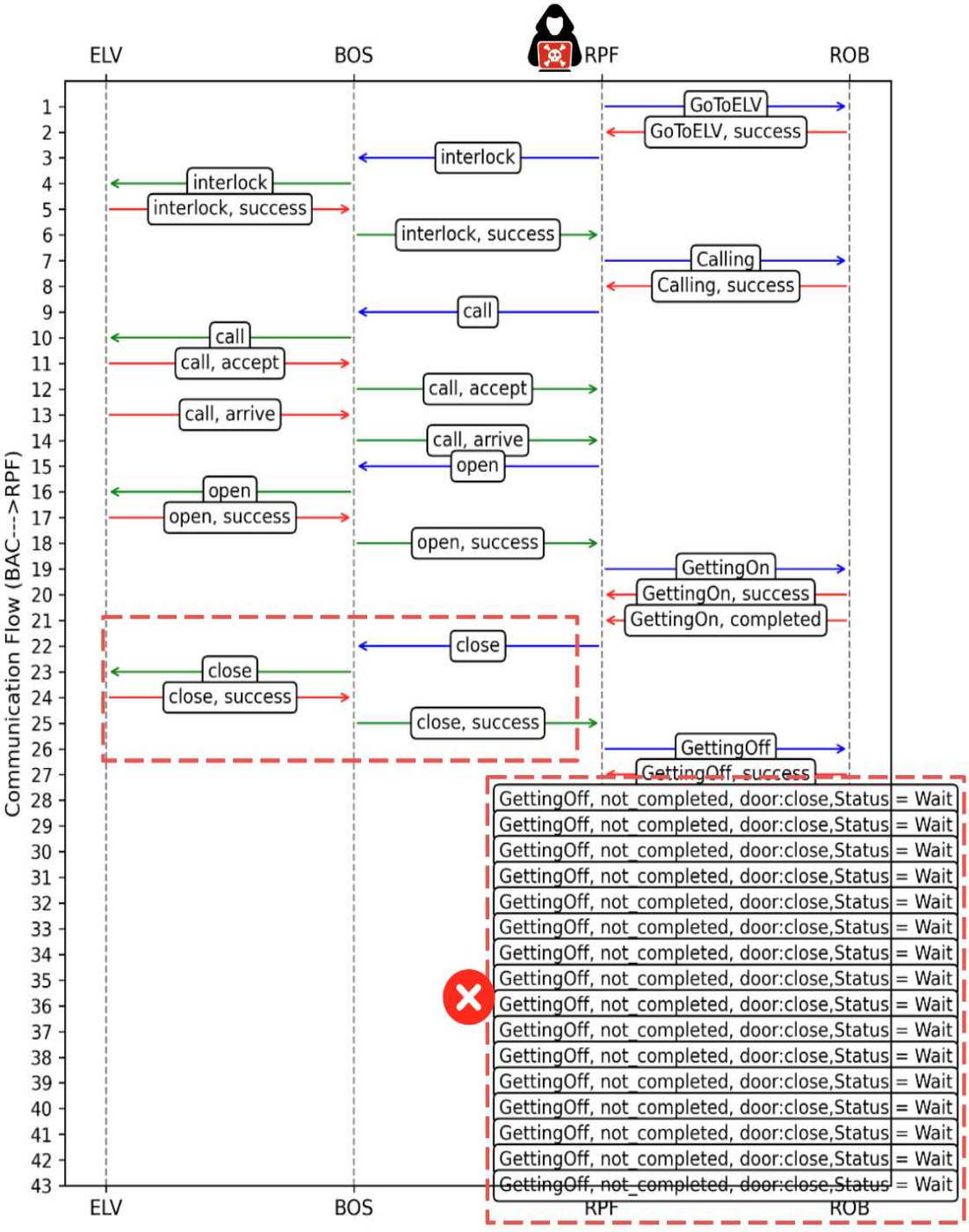

Figure 6.15: Experiment results for BAC attack targeting RPF.

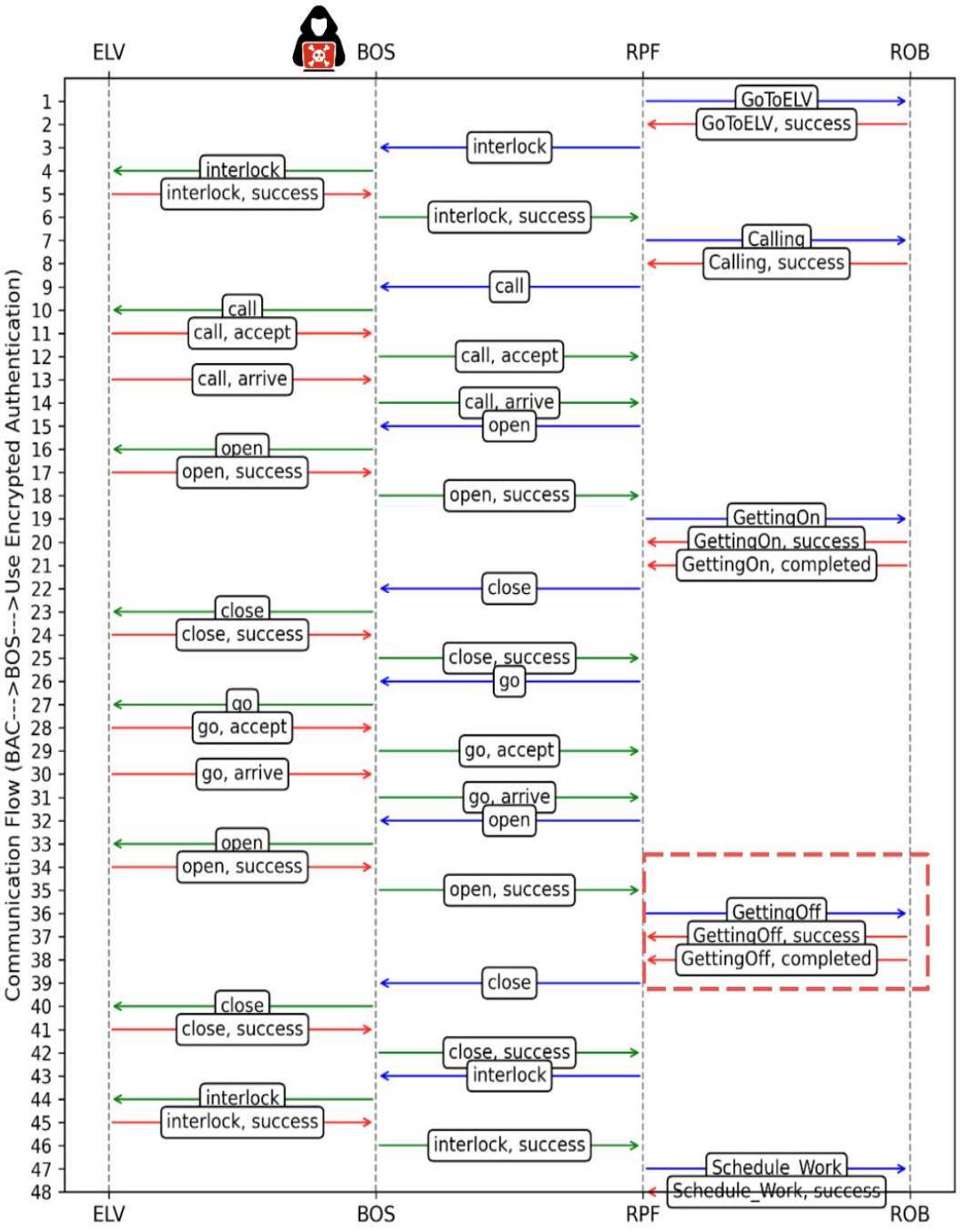

Figure 6.16: Experiment results for BAC attack targeting BOS when using encrypted authentication as security measure.

Chapter 7

Conclusion

7.1 Summary

To address the shortcomings of the current simulation platforms for smart building control systems, this study has successfully developed SBCSE, which provides a powerful tool for testing and evaluating the performance and security of smart building systems. In addition, SBCSE provides a user-friendly interface and a security module capable of emulating attack scenarios, so that various attack scenarios can be simulated and tested for potential risks, and corresponding security measures can be proposed. By developing a emulation platform that combines the simulation functions of smart building control systems with cybersecurity testing, we have not only helped to reduce testing costs, but also improved the security of smart building testing. In the context of increasing cybersecurity threats, simulators play an immeasurable role in ensuring the safety and reliability of smart building systems, and this study provides an effective testing platform to promote the further development and security of the smart building field.

Regarding the performance of the emulator, we verified the accuracy of the emulator by analyzing the logs of the SBCSE communication data and comparing them with the logs of the real building. Our analysis results show that the match rate between the SBCSE and real building data regarding the message command flow between BOS and RPF during the communication process reaches 100%, confirming that the emulator successfully reproduces the test environment of the real system. The emulator's response to various inputs was tested to be within normal standards, which is essential to ensure that the simulator can correctly predict the behavior of the real system under different conditions, guaranteeing the accuracy of the test environment. In addition, the speed of SBCSE in predicting and analyzing potential problems saves testing time compared to traditional testing methods, and it can be tuned for emulation speed. The results show that the

emulator's response time and latency for individual messages remain largely stable during experiments up to 10x speed, and no packet loss occurs.

In terms of cybersecurity in smart buildings, we have added a security attack module to SBCSE. Five attack scenarios were analyzed for the network attacks with high risk in the current system. In the emulator, we implemented three scenarios of attacks on MITM, BAC, and DDoS. Security measures are proposed for each of these attacks, and the effectiveness of the countermeasures is confirmed by implementing attack tests with the countermeasures applied. This module provides an effective method for intelligent buildings in network security testing.

Overall, the emulator reduces the risk of system failure or downtime that may result from testing in actual building systems. In this way, we can conduct in-depth testing and system optimization of smart building systems without increasing the risk of actual buildings, ensuring the safety and reliability of the technology before implementing it in actual building systems. SBCSE provides a more cost-effective solution for testing and evaluation of smart buildings.

NOTE: Generative AI technology was used for translation and grammar checking during the preparation of this thesis.

7.2 Future Work

In the future, we will focus on improving the performance of the emulator to address the limitations that currently exist. Specifically, we will enhance the error and exception handling mechanisms, especially in handling packet loss, timeouts, and exceptions. We will also consider the response latency of the real-time system, which is currently only guaranteed to be normal up to 10x speed, and we will follow up with improvements to increase fault tolerance at higher speeds.

Meanwhile, in order to cope with the evolving cybersecurity threats, we will extend the cybersecurity attack module of SBCSE. Although the current design already covers some attack scenarios, there are more security issues that need to be addressed in the field of intelligent buildings. Therefore, we will extend the system's functionality to support user-defined testing of security attack scenarios and plan to add more types of cyber attacks.

Finally, the scope of smart building control systems is very broad, and future research can simulate multiple robots, add other IoT devices and energy systems, and so on, to enrich the functionality of the simulator. In our future work we will continue to improve the simulator system and add more features while ensuring accuracy.

Publications

- Weng, X., Beuran, R. "Smart Building Control System Emulation Platform for Security Testing." 29th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2024). IEEE, 2024. p.220-223.

Bibliography

[1] ASHB. *2023 Intelligent Building Technology & Market Trends*. Report. Accessed: 2024-12-01. 2023. URL: https://www.ashb.com/wp-content/uploads/2024/02/HRI_ASHB_IBC_Survey-Executive-Summary_14-November-2023-Final-2.pdf.

[2] Wolfgang Kastner et al. "Communication systems for building automation and control". In: *Proceedings of the IEEE* 93.6 (2005), pp. 1178–1203.

[3] Shengwei Wang and Junlong Xie. "Integrating Building Management System and facilities management on the Internet". In: *Automation in construction* 11.6 (2002), pp. 707–715.

[4] Kaspersky. *Smart buildings threat landscape: 37.8% targeted by malicious attacks in H1 2019*. Online. Accessed: 2024-12-01. 2019. URL: https://me-en.kaspersky.com/about/press-releases/smart-buildings-threat-landscape.

[5] Houssem Eddine Degha et al. "Open-SBS: Smart Building Simulator". In: *2022 International Arab Conference on Information Technology (ACIT)*. IEEE. 2022, pp. 1–15.

[6] David P Chassin, Kevin Schneider, and Clint Gerkensmeyer. "GridLAB-D: An open-source power systems modeling and simulation environment". In: *2008 IEEE/PES Transmission and Distribution Conference and Exposition*. IEEE. 2008, pp. 1–5.

[7] Rongpeng Zhang and Tianzhen Hong. "Modeling of HVAC operational faults in building performance simulation". In: *Applied Energy* 202 (2017), pp. 178–188.

[8] Tamas Pflanzner et al. "MobIoTSim: Towards a mobile IoT device simulator". In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE. 2016, pp. 21–27.

[9] R. Beuran J. Wu. "Formal and Experimental Verification of Robot Control Protocols for Smart Buildings". In: *Symposium on Cryptography and Information Security (SCIS 2025)*. SCIS. 2025.

[10] Michael Mylrea and Sri Nikhil Gupta Gourisetti. “Cybersecurity and optimization in smart “autonomous” buildings”. In: *Autonomy and Artificial Intelligence: A Threat or Savior?* (2017), pp. 263–294.

[11] Jesus Pacheco and Salim Hariri. “IoT security framework for smart cyber infrastructures”. In: *2016 IEEE 1st International workshops on Foundations and Applications of self* systems (fas* w)*. IEEE. 2016, pp. 242–247.

[12] Bruno Augusti Mozzaquatro et al. “An ontology-based cybersecurity framework for the internet of things”. In: *Sensors* 18.9 (2018), p. 3053.

[13] Digital Transformation Channel. *What is a Smart Building? Explaining the Meaning and Definition from the Basics*. Online. Accessed: 2024-12-01. 2022. URL: https://www.cloud-for-all.com/dx/blog/what-is-smart-building#toc-0.

[14] Digital Architecture Design Center Information Processing Society of Japan. *Smart Building System Architecture Guideline*. Online. Accessed: 2024-12-01. 2023. URL: https://www.ipa.go.jp/digital/architecture/Individual-link/ps6vr70000016bq2-att/smartbuilding_system-architecture_guideline.pdf.

[15] Vinicius Fulber-Garcia. *Differences Between Simulation and Emulation*. Online. Accessed: 2024-12-01. 2024. URL: https://www.baeldung.com/cs/simulation-vs-emulation.

[16] Shu Tang et al. “BIM assisted Building Automation System information exchange using BACnet and IFC”. In: *Automation in Construction* 110 (2020), p. 103049.

[17] Muneer Bani Yassein et al. “Internet of Things: Survey and open issues of MQTT protocol”. In: *2017 international conference on engineering & MIS (ICEMIS)*. Ieee. 2017, pp. 1–6.

[18] Industrial Cyber Security Research Group (METI). *Cyber-Physical Security Measures Guideline for Building Systems*. Online. Accessed: 2024-12-01. 2023. URL: https://www.meti.go.jp/policy/netsecurity/wg1/bill_gideline_2.pdf.

[19] European Union Agency for Cybersecurity. *Good Practices for Security of Internet of Things in the context of Smart Manufacturing*. Online. Accessed: 2024-12-01. 2018. URL: https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot.

[20] OWASP. *OWASP Top 10 - 2021*. Online. Accessed: 2024-12-01. 2023. URL: https://owasp.org/Top10/.

[21] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. "A survey of man in the middle attacks". In: *IEEE communications surveys & tutorials* 18.3 (2016), pp. 2027–2051.

[22] Rapid7 Corporation. *Man in the Middle (MITM) Attacks*. Online. Accessed: 2024-12-01. URL: https://www.rapid7.com/fundamentals/man-in-the-middle-attacks/.

[23] Gregg Lindemulder and Matt Kosinski. *What is a Man-in-the-Middle Attack (MITM)?* Online. Accessed: 2024-12-01. URL: https://www.ibm.com/cn-zh/think/topics/man-in-the-middle.

[24] Manesh Thankappan, Helena Rifà-Pous, and Carles Garrigues. "Multi-Channel Man-in-the-Middle attacks against protected Wi-Fi networks: A state of the art review". In: *Expert Systems with Applications* 210 (2022), p. 118401.

[25] NEC Fielding Corporation. *What is a DDoS attack?* Online. Accessed: 2024-12-01. URL: https://www.fielding.co.jp/service/security/measures/column/column-1/#anc-01.

[26] Imperva Corporation. *DDoS Attacks*. Online. Accessed: 2024-12-01. URL: https://www.imperva.com/learn/ddos/ddos-attacks/.

[27] Ghafar A Jaafar, Shahidan M Abdullah, and Saifuladli Ismail. "Review of recent detection methods for HTTP DDoS attack". In: *Journal of Computer Networks and Communications* 2019.1 (2019), p. 1283472.

[28] Nedim Marić. *Broken Access Control: Attack Examples and 4 Defensive Measures*. Online. Accessed: 2024-12-01. 2023. URL: https://brightsec.com/blog/broken-access-control-attack-examples-and-4-defensive-measures/.

[29] HITACHI Corporation. *Preventing Broken Access Control Vulnerabilities*. Online. Accessed: 2024-12-01. URL: https://www.securebrain.co.jp/eng/blog/broken-access-control-vulnerabilities/.

[30] Matthew Kosinski. *What is ransomware?* Online. Accessed: 2024-12-01. 2024. URL: https://www.ibm.com/cn-zh/topics/ransomware.

[31] Jason Firch. *How Does Ransomware Spread?* Online. Accessed: 2024-12-01. 2024. URL: https://purplesec.us/learn/common-ways-ransomware-spreads/.

[32] Fatima Salahdine and Naima Kaabouch. "Social engineering attacks: A survey". In: *Future internet* 11.4 (2019), p. 89.

[33] European Union Agency for Cybersecurity. *Guidelines for Securing the Internet of Things*. Online. Accessed: 2024-12-01. 2020. URL: https://www.enisa.europa.eu/sites/default/files/publications/ENISA%20Report%20-%20Guidelines%20for%20Securing%20the%20Internet%20of%20Things.pdf.

[34] European Union Agency for Cybersecurity. *Threat Taxonomy A tool for structuring threat information*. Online. Accessed: 2024-12-01. 2016. URL: https://www.um.es/documents/2096502/4937674/Enisa.pdf/2374a6a9-3c9d-422c-b5ad-b047a2fb8568.

[35] NIST. *Draft NISTIR 8228: Considerations for Managing Internet of Things (IoT) Cybersecurity and Privacy Risks*. Online. Accessed: 2024-12-01. 2019. URL: https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8228-draft.pdf.

[36] HiveMQ Team. *TLS/SSL - MQTT Security Fundamentals*. Online. Accessed: 2025-1-24. 2024. URL: https://www.hivemq.com/blog/mqtt-security-fundamentals-tls-ssl/.

[37] Saman Taghavi Zargar, James Joshi, and David Tipper. "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks". In: *IEEE Communications Surveys & Tutorials* 15.4 (2013), pp. 2046–2069. DOI: 10.1109/SURV.2013.031413.00127.

[38] Ivan Vaccari et al. "MQTTset, a new dataset for machine learning techniques on MQTT". In: *Sensors* 20.22 (2020), p. 6578.

[39] Abimbola O. Sangodoyin et al. "Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning". In: *IEEE Access* 9 (2021), pp. 122495–122508. DOI: 10.1109/ACCESS.2021.3109490.

[40] Amit Praseed and P Santhi Thilagam. "DDoS attacks at the application layer: Challenges and research perspectives for safeguarding web applications". In: *IEEE Communications Surveys & Tutorials* 21.1 (2018), pp. 661–685.