Received 26 April 2023; | Revised 12 May 2023; | Accepted 19 June 2023

DOI: xxx/xxxx

ARTICLE TYPE

Understanding the Impact of IoT Security Patterns on CPU Usage and Energy Consumption on IoT Devices

Saeid Jamshidi* | Amin Nikanjam | Kawser Wazed Nafi | Foutse Khomh | Mohammad-Adnan Hamdaqa

[1] Ecole Polytechnique Montréal, Quebec, Canada

Correspondence
*Saeid Jamshidi, Email: jamshidi.saeid@polymtl.ca

Present Address
Ecole Polytechnique Montréal,Quebec, Canada

The Internet of Things (IoT) has given rise to numerous security issues that require effective solutions. IoT security patterns have been suggested as an effective approach to address recurrent security design issues. Although several IoT security patterns are proposed in the literature, it remains unclear how they impact the energy consumption and CPU usage of IoT-edge-based applications. We conducted an empirical study using three testbed IoT applications (i.e., smart home, smart city, and healthcare) to shed light on this issue. We evaluated the impact of six IoT security patterns, including Personal Zone Hub, Trusted Communication Partner, Outbound-Only Connection, Blacklist, Whitelist, and Secure Sensor Node, both in pairs and in combination (i.e., all patterns). Specifically, we conducted multiple penetration tests to first assess the pattern's effectiveness against attacks. Then, we conducted a comprehensive analysis of the energy consumption and CPU usage of the applications with/without the implemented security patterns, aiming to evaluate the potential impact of these patterns on energy efficiency and CPU usage. Our findings demonstrate a statistically significant increase in energy consumption and CPU usage. Based on these findings, we provide guidelines for IoT developers to follow when implementing IoT-edge-based applications.

KEYWORDS:
IoT Security Patterns, Security, CPU Usage, Energy Consumption

1 | INTRODUCTION

Internet of Things (IoT) refers to a network of interconnected physical objects embedded with software and hardware components. IoT has found wide-ranging applications in various domains, including commerce, healthcare, industry, and transportation [1]. IoT devices primarily collect, analyze, and communicate data with other connected devices. To facilitate this, edge devices such as sensors, gateways, routers, and edge servers are often used in IoT applications to collect and process data. The exchange of information among these devices occurs through both wired and wireless networks in IoT [2] which are vulnerable to various attacks. The interdependence of components within the IoT architecture, as depicted in Fig.1 , highlights the vulnerability of the entire system to a single security breach. Consequently, ensuring the safety and reliability of IoT-based applications has become crucial for their seamless operation in society.
On the other hand, IoT comprises low-powered devices such as the Raspberry Pi, which have limited access to different resources.

[0] **Abbreviations:** ANA, anti-nuclear antibodies; APC, antigen-presenting cells; IRF, interferon regulatory factor

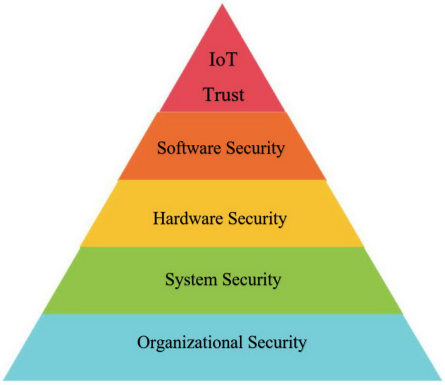

FIGURE 1 The figure illustrates the block diagram of the IoT pyramid.

Implementing security mechanisms on IoT devices typically requires additional processing, communication, and therefore computation resources. This can potentially impact the performance of IoT devices and networks[3]. Despite the performance overhead introduced by security mechanisms, they are indispensable for safeguarding IoT networks against cyber threats. Hence, striking a balance between an appropriate level of security and the physical limitations of the device, such as low computing power and restricted memory capacity in battery-powered devices, is crucial.

Researchers have proposed various security patterns to encapsulate solutions to recurrent problems in the security of IoT. These patterns consider the secure design of IoT-based applications, accounting for their heterogeneity and complexity[4], like the Personal Zone Hub, Trusted Communication Partner, Outbound-Only Connection, Blacklist, Whitelist[5], and Secure Sensor Node patterns[6]. However, despite the availability of these security solutions, no exploratory study has compared the energy consumption and CPU usage of IoT devices when set up with different security patterns. Therefore, finding a trade-off between the available IoT security solutions and IoT devices' energy consumption and CPU usage is crucial.

This study aims to investigate the impact of different security patterns on energy consumption and CPU usage of IoT devices and their inter-connected communication. To do so, we have implemented six distinct IoT security patterns: Personal Zone Hub, Trusted Communication Partner, Outbound-Only Connection, Blacklist, Whitelist, and Secure Sensor Node. Subsequently, we have chosen three IoT-edge-based applications, namely a smart home, a smart city, and a healthcare case, to evaluate the efficacy of these patterns against critical attacks. The attacks include Distributed Denial-of-Service (DDoS), Secure Socket Shell, Man-In-The-Middle (MITM), and Brute-force (BF) which are implemented by Kali Linux[7]. These attacks were selected to assess the effectiveness of the chosen patterns under attack and to evaluate the energy consumed and CPU usage. We used the PowerTOP[8] tool for our evaluations. Our results show that the examined security patterns have practical utility in protecting against attacks; however, the attacks may negatively impact application performance under certain conditions, such as DDoS attacks. The energy consumed and CPU usage in different scenarios exhibit a statistically significant difference in most cases under attacks. Our findings serve as a roadmap for practitioners in the selection of apt security solutions and patterns, specifically tailored to address the unique challenges of optimizing security while adhering to the inherent limitations of IoT devices.

To summarize, this paper makes the following contribution:

- We assessed the effectiveness of security patterns applied to IoT-edge-based applications against various attacks.
- We reported the trade-offs between security, energy consumption, and CPU usage in IoT devices and IoT-edge-based applications.

The remainder of this paper is organized as follows: Section 2 discusses related works to our research. Section 3 discusses the necessary background knowledge. In Section 4, we describe the experimental design, the Research Questions (RQs), and the metrics of the experiments. Section 5 explains our results and findings. Section 6 summarizes some guidelines for developers of IoT-edge-based applications based on our findings. Section 7 discusses threats to the validity of our study. Finally, Section 8 concludes the paper and outlines future work.

2 | RELATED WORKS

Research on finding effective security patterns and cryptography algorithms for IoT-based-edge applications and IoT frameworks is not new. Researchers are performing their studies in this direction for an extended period. Still, there remains a study gap in finding suitable security solutions for IoT devices, including the lack of practical investigation on the impact of security patterns on IoT devices and their communication, energy consumption, and CPU usage. Current research mainly focuses on searching for a secured IoT-edge devices interconnected network using different network-based distributed platforms such as fog computing, edge computing, or cloud and host IoT applications. This kind of architecture enables secure communication and storage of data among IoT devices and other interconnected components. This section will discuss the three topics largely aligned with the empirical study we performed.

2.1 | Secured IoT Architecture

Among various researchers in secured IoT architecture, Syed et al. presented a security pattern for IoT devices that uses the Fog Computing[9] in the back-end. Their pattern can assist users who wish to grasp the relationship among the IoT components. The authors of this work used a pattern-oriented software architecture that describes a pattern to provide an answer to a problem recurring in a particular context. Andreas et al.[10] also proposed an architectural pattern for Fog computing (Fogxy). By decoupling the heterogeneous IoT components from the centralized cloud components, Fogxy describes the connection and interplay of different components in IoT architecture. The cloud-thing continuum is created by introducing different layers of the system. In several domains, including manufacturing, autonomous vehicles, and smart grids, a decentralized deployment of clients enables real-time critical applications. The authors of this paper claimed that their proposed pattern is intended to represent a set of instructions for software engineers assigned the difficult task of creating applications related to fog computing. The properties of fog computing can be met by implementing Fogxy's distributed components which are organized in various hierarchies. The operation of applications, hence, using fog computing, will become less complicated. So, to put the authors' work in a nutshell, we can say, Fogxy directly works with components of Fog computing instead of only considering IoT components or IoT-edge devices. In another work, Fernandez et al.[11] describe a pattern that indicates each threat, such as DDoS, unauthorized access to the data in a thing, the wrong command to the actuator, etc., and its corresponding defenses against that threat. In addition to these, the authors of this paper also used different types of security patterns to ensure security in two architectures: cloud and fog. The authors of this paper used five varieties of secure patterns in two given architectures: secure clouds, secure fog, secure things, secure actuators, and secured sensors.

Researchers also checked whether blockchain architecture and its security mechanism could be used with IoT devices and made them secure. Pahl et al.[12] proposed a design pattern that consists of IoT-edge orchestration and a blockchain-based provenance mechanism. A state machine that describes the basic orchestration activities is formalized as an architecture pattern supported by a blockchain to enable trusted orchestration management. Fysarakis et al.[13] have discussed the development of IoT platform enablers and patterns for orchestrating smart objects in IoT applications with guaranteed security, privacy, dependability, and interoperability (SPDI). Accordingly, this can be achieved by using patterns defined as generic ways of integrating and orchestrating different types of smart objects and components guaranteeing specific SPDI properties, henceforth called SPDI patterns. Searching for a suitable secured IoT framework and interconnected network is not directly aligned with our present area of study and is out of the scope of this paper. In our paper, we implemented some of the cryptography algorithms. We secured interconnected networks for IoT devices proposed in different studies to explore the energy consumption and CPU usage of IoT-edge devices when placed in different secured IoT architectures. Thus, this paper's scope fully differs from the regular work of searching for a secured IoT architecture.

2.2 | IoT Security Patterns

Lee et al.[14] have applied five security design patterns while developing an IoT software system. The authors' goal in this work was to address the security problems of IoT applications and increase system security. Their security design patterns included a secure adapter, directory, logger, exception manager, and input validation. At the same time, the authors designed a secure directory, secure logger, and exception manager to address the issue of unsecured application data. They claimed that their secure adapter and input validation pattern handle the security issues of unprotected wireless communication and user information, respectively.

Later, Fernández et al.[15] implemented a secure distributed publish/
subscribe pattern. They claim this pattern is one of the most widely used patterns for IoT software design. They aimed to decouple event publishers and those interested in the events (i.e., subscribers) through IoT systems. Publication and subscription are securely handled. After that, Syed and Fernández[16] proposed to exploit the security flaws of IoT devices to make a denial of service (DoS) attack. The authors described how this attack is usually carried out and countermeasures that can be taken to mitigate it. In another work, Fernandez and Romero[17] showed how to misuse RF remote controllers to take control of the IoT devices using RF. In this paper, the authors described how attackers maintain control over the RF remote controllers and how to mitigate them. This pattern was developed as part of their work on building a Security Reference Architecture for Cargo Ports. This paper describes an attack on industrial RF remote controllers, widely used in cranes in container terminals, as a misuse pattern.

Cristian et al.[6] introduced a design pattern for secure sensor nodes to describe the problem and offer a reusable, adaptable, and extensible solution. In this pattern, they describe a secure sensor node as a cyber-physical system that obtains, stores, and transmits data securely from a physical environment to another node or system of information. Rafael et al.[18] demonstrated how the UAV-Aided Wireless Sensor Network (WSN) pattern could implement a mining dam monitoring system to warn against deadly collapses and possibly prevent them. Their report uses an unmanned aircraft vehicle as part of a WSN. Also, in WSN, sensor nodes that are located closer to the sink have a greater risk of becoming overloaded (as a result of receiving a significant amount of data packets) and are subject to the risk of having their battery drained faster than nodes those are located further away, which as a result reduces the network lifetime. To get around this issue, the UAV-aided WSN pattern simplifies the underlying architecture of a WSN using a UAV as a mobile sink node.

Security on IoT components is required to enhance users' privacy and protect sensitive information. Pape et al.,[19] demonstrated how applying privacy patterns to the IoT architecture can improve users' privacy. They mapped the privacy patterns onto IoT, fog computing, and cloud computing architectures. Fernandez et al.[4] surveyed and classified existing IoT security patterns to assess their scopes and quality and determine whether they are suitable for inclusion in a helpful catalog. A catalog must cover most of the common types of security systems to be useful. They concluded that the number of available patterns is insufficient for a workable catalog and that most patterns are either incomplete or employ descriptions that differ. They concluded that none of them had taken security concerns into account. They suggested making adjustments to the pattern-based approaches that have already been in use for distributed systems as a potential option.

To the best of our knowledge, no study presents comparative results or discussion about the impacts of different security patterns in terms of the level of security (which means the stability of different security patterns under different security attacks), energy consumption, and CPU usage in the context of the IoT.

2.3 | Energy Consumption in IoT devices

Researchers have been working on searching for energy-efficient and low-cost security solutions for IoT networks and frameworks for a long time[20]. Among them, Prakasam et al.[21] proposed an 8-bit-based Lightweight Cryptography algorithm for secured communication keeping in mind the low power resources of IoT devices. Although this approach is good for embedded IoT devices, for real-world IoT communication, 8-bit cryptography is not suitable at all. In another work, Yeh et al.[22] proposed an Elliptic Curve point multiplication-based low-energy inhaling cryptography algorithm for communication to secure its device from a single source attack. This algorithm might be suitable for intranet communication between the devices but not for the IoT applications open to the regular world of internet communication. As a consequence, it is required to see whether the available cryptography and security algorithms and security patterns used for regular Internet communication can be applied to IoT devices and frameworks and their Intranet and Internet communication.

A group of studies is performed to explore the adaptability of high-performance cryptography algorithms and security patterns in IoT devices and communication. Yazdinejad et al.[23] proposed an IoT architecture where they added a Software Defined Network (SDN) component with IoT devices along with regular blockchain-based security architecture to ensure security in an energy-efficient way. Very recently, Tekin et al.[24] performed an empirical study where they collected the energy consumption of Machine-Learning-based (ML) Intrusion detection models on IoT devices at the time of training and executing them. Their work was mainly offline-based analysis of the energy requirements of ML models. As far as we know, no previous research has been conducted on the topic covered by this paper, which focuses on examining the sustainability level of various Security Patterns in the face of different security breaching attacks. Additionally, it investigates the effects these patterns under attack have on CPU usage and energy consumption of IoT devices within an IoT network.

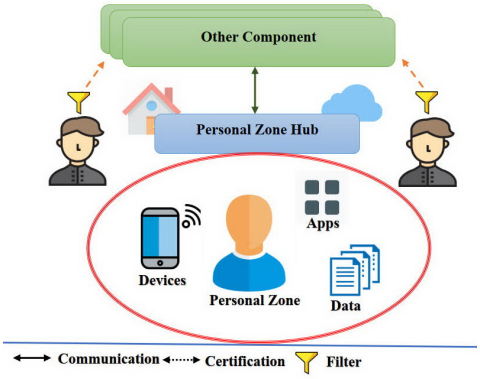

FIGURE 2 PZH pattern solution sketch[5].

3 | BACKGROUND

In this section, we introduce key concepts relevant to our study.

3.1 | Designing IoT applications

Developing and designing applications for IoT infrastructure can be challenging[25,26,27,28,29] for various reasons. Depleting the power of low-power IoT devices, even while they are operating at their minimum power levels. Nevertheless, power consumption control becomes increasingly intricate in IoT devices with advanced configurations. This is because, in high-performance IoT devices, the processor, display, and communication interfaces have varying power requirements. Furthermore, the utilization of the CPU is a crucial aspect that has a substantial effect on both the energy efficiency and the operational capacity of these devices. Hence, it is crucial to guarantee that various components of IoT devices, such as their CPUs, exhibit reduced power consumption and possess an extended battery lifespan. It is essential to guarantee that the apps running on them are optimized for both energy efficiency and reduced CPU usage at the same time.

Ensuring security during connectivity changes is essential in IoT, alongside power management and monitoring. Many IoT/M2M devices also offer a variety of internet connectivity choices. Any security error enables hackers to gain unauthorized access or assume control of the devices, perhaps resulting in connectivity issues[30]. Hence, it is imperative to discern both the potential risks and the energy consumption and CPU usage of each component within the complete IoT architecture. The range of threats in IoT systems is more varied than in any other IT system. Every IoT device integrated into an IoT infrastructure is seen as a component of cyber-physical systems and is susceptible to its own set of security risks. Therefore, it is imperative to take into account the safety and reliability features of IoT devices and their internal communications to safeguard the IoT infrastructure and its internal data transmission. Furthermore, IoT devices can be interconnected using a multitude of connecting protocols. To ensure the security of an IoT infrastructure, it is imperative to safeguard the connection protocols employed by these IoT devices. To address the security and privacy concerns in IoT networked devices, it is necessary to identify a security solution or pattern that effectively reduces energy consumption and CPU usage while maintaining efficiency[31,32].

3.2 | IoT patterns

This section explores the fundamental concept of the six security patterns and their significance in safeguarding the security of intra-connected and interconnected IoT devices. The implementation of the Stuttgart[5] patterns was deemed necessary as they furnish a comprehensive[4] set of IoT patterns. Furthermore, we integrated the SSN[6] pattern to facilitate secure data transmission between the server, user, and Raspberry Pi, along with its attached sensors.

i. Personal Zone Hub (PZH): The growing prevalence of IoT devices has presented users with a concomitant increase in the complexity of managing these devices' permissions, data sharing, and control across various gateways and cloud systems. To address these challenges, PHZs have emerged as a viable solution. PHZs enable users to effectively manage, control, and integrate all of their devices, services, applications, and workflows. Furthermore, PHZs offer the added advantage of creating a

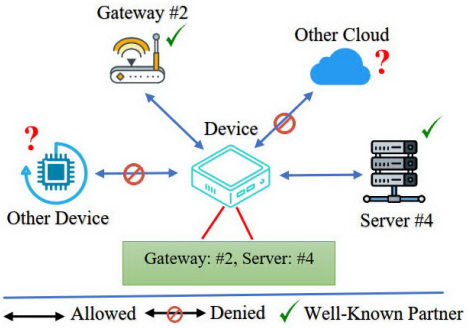

FIGURE 3 Trusted Communication Partner pattern solution sketch[5].

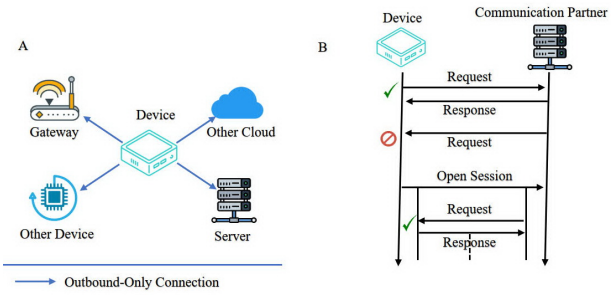

FIGURE 4 Outbound-Only Connection pattern solution sketch[5].

permanently addressable hub, thereby allowing users to selectively share access to different parts of the data and functionality encompassed by the PHZs. As a result, PHZs represent a promising solution to the challenges associated with managing and controlling IoT devices in a network.

ii. Trusted Communication Partner (TCP): In a dynamic environment, a device may have multiple communication options available. However, not all of these communication partners may be familiar or reliable. Some may even pose a security threat; attackers may exploit these communication media to gain unauthorized access to the device and its network. To mitigate such risks, it is recommended to configure the device with a single trusted communication partner or a list of dependable communication partners. This pattern enables not only secure incoming and outgoing communication with these trusted partners but also restricts other communication attempts and promptly alerts the designated person in charge for further investigation.

iii. Outbound-Only Connection (OOC): The proliferation of IoT devices has created a new avenue for malicious actors to gain access to individuals' personal and sensitive information. Cybercriminals often exploit vulnerabilities in these devices by sending unsolicited communication requests that trick the device into connecting to a compromised network. To mitigate such risks, many IoT devices are programmed only to allow authorized connections, thereby preventing unauthorized access attempts. This controlled connection OOC pattern allows the devices to initiate and control connections while not rejecting all incoming communication requests that lack proper authorization.

iv. Blacklist (BL): Privileges, whether they are freely given or explicitly granted, are susceptible to abuse. Establishing measures to prevent ongoing abuse and restrict potential future abuse is imperative. To achieve this, maintainers frequently create a BL containing records of individuals identified as abusive users or misused their privileges. By maintaining such a list, the maintainers can better manage the privileges and ensure that they are only granted to trustworthy individuals.

v. Whitelist (WL): The WL functions in contrast to the BL, which blocks known abusive communication partners. However, while the BL is never complete, the WL limits the number of possible attacks by providing an administrative interface to add communication partner identifiers. This interface enables the checking of privileges that the WL controls. When a communication partner on the WL requests a privilege, the system can proceed. Conversely, the system will automatically deny the request if the requester is not on the WL. Thus, the WL serves as an effective means of limiting access to known communication partners.

vi. Secure Sensor Node (SSN): A sensor node seamlessly performs data collection and transmission tasks in a sensor network. However, suppose the sensor node lacks security mechanisms. In that case, it may be vulnerable to various security issues, such

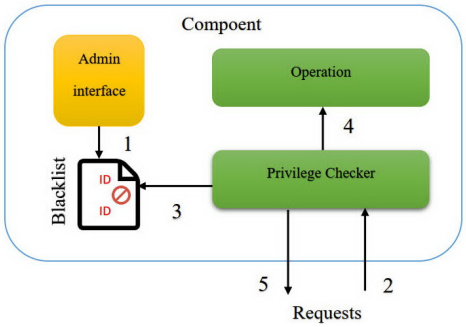

FIGURE 5 Blacklist pattern solution sketch[5].

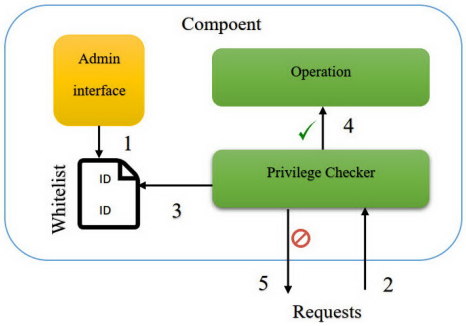

FIGURE 6 Whitelist pattern solution sketch[5].

as privacy, integrity, and confidentiality. This can compromise the data stored, data transmission, and the environment with which it interacts. The SSN pattern tells a sensor node how to securely send and store data in a WSN, regardless of its topology. A secure sensor node can encrypt data in communications and storage while keeping a record of events for auditing purposes. As the sensor node covers other aspects of the node without special security considerations, it has no mechanisms or design decisions. To deliver security features to a sensor node, it is necessary to apply additional components to encrypt data, log events, and perform authentication and authorization processes with other nodes of the WSN.

4 | STUDY DESIGN

This section outlines our methodology for evaluating the effects of specific IoT security patterns on IoT-edge-based applications. We first define our RQs and subsequently provide details on the experimental design and the metrics used to assess the impact of the patterns.

4.1 | Research Questions(RQs):

Our research aims to address the following RQs:

- **RQ1: To what extent does the utilization of IoT security patterns improve the security of IoT-edge-based applications?**
  We undertake an analysis of both in-pair and combined patterns (i.e., all patterns) to assess the security aspects of the applications deployed within the IoT network. The security is assessed using a penetration test, which involves launching simulated attacks against the applications. By adopting this approach, we aim to assess the effectiveness of the security patterns and identify any vulnerabilities that need to be addressed to enhance the security of IoT networks for particular implementation of applications.

- **RQ2: How does the utilization of security patterns in IoT edge-based applications impact energy consumption, CPU usage, and CPU load?** We conduct an empirical study on three IoT applications to investigate the impact of in-pair and the combined patterns on energy consumption, CPU usage, and CPU load. We measure these metrics with/without patterns and under the attacks.

4.2 | Experimental Design

In this subsection, we present the experimental setup of our study. We conduct an empirical investigation on three IoT-edge-based applications to assess six IoT security patterns: PZH, TCP, OOC, BL, WL, and SSN. To answer RQ1, we conducted a thorough investigation using multiple communication frameworks and IoT devices. A testbed was created, consisting of three Raspberry Pis and various sensors. Each device established an IoT-edge-based application configuration in conjunction with its sensor nodes. Our methodology involved implementing six distinct security patterns in the three applications and conducting a penetration test to evaluate their effectiveness in enhancing the security of the applications. We performed the penetration tests (i.e., launching attacks) using Kali Linux.
For RQ2, we measure energy consumption, CPU usage, and CPU load with/without patterns under the attacks simulated by the penetration tests. We created three IoT hubs with the sensors. To work with each sensor's data in each IoT hub, we developed individual applications and implemented related security patterns.

- **Smart Home:** The first experimental setup is called the smart home automation method using Raspberry Pi as the edge device[33]. It consists of a Motion sensor, an AC supply, a DHT22 sensor, a Module M-Q9, and an MCP3008 converter. By utilizing a smart home, we can exercise control over the internet-connected home equipment in our testbed. The application can measure the home's temperature and humidity and control the home's light and gas sensors. This application can also capture and identify intruders' presence. If the motion is detected, the edge will process, and the Raspberry Pi will send a notification to the user.

  In this paper, we targeted three IoT-edge-based applications:

- **Smart City:** The second experimental configuration involves the implementation of an advanced smart city[34] infrastructure that is purposefully tailored to facilitate real-time pollution monitoring. In this setup, the Raspberry Pi functions as an edge computing device, playing a vital role in processing and analyzing data at the local level. To achieve comprehensive air quality monitoring, a set of cutting-edge sensors is integrated into this smart city system. Specifically, the employed sensors comprise the DHT22, Module M-Q9, MCP3008 converter, pressure air sensor, and Module MQ135 sensor. These sensors collectively enable measuring and tracking various critical air quality parameters. The recorded environmental metrics encompass essential factors such as temperature, atmospheric pressure, altitude, humidity, carbon dioxide levels, carbon monoxide levels, methane concentration, and ammonium levels.

- **HealthCare:** The third experimental setup is an healthcare[35,36] application that incorporates ECG and heartbeat sensor monitoring system. The AD8382 ECG sensor to read patient's data, Arduino ESP8266 Wi-Fi module. Implementing the proposed ECG healthcare system enables the doctor to monitor the patient remotely using IoT. The data is then processed using a Raspberry Pi, and useful information is saved to the IoT edge. The system would primarily extract the biosignal ECG using an ECG and heartbeat sensor. Through continuous monitoring and graphical representation of the patient's information, doctors/nurses/relatives can remotely check the patient's condition.

Fig. 7 illustrates the topology of our configured testbed. All the data collection and interaction applications were written in Python and C++. The server side was developed using Flask[37] and Nginx[38], a lightweight web server. We used an SQLite database to store sensor data and the Google Chart API[39] to create visual representations of the sensor data and Plotly[40] for graphical sensor data analysis. Our application sends the data to the edge server. Also, only the authorized server maintainer can control the data, and the users can see or manipulate the data.

4.3 | Implementation of patterns

In this section, we delve into the practical aspects of how these patterns are implemented.
OOC: We implemented the OOC pattern via the Uncomplicated Firewall (UFW)[41]. The UFW tool aims to clarify the complexities associated with IP tables. The UFW was adjusted to effectively refuse all unsolicited incoming connections by executing

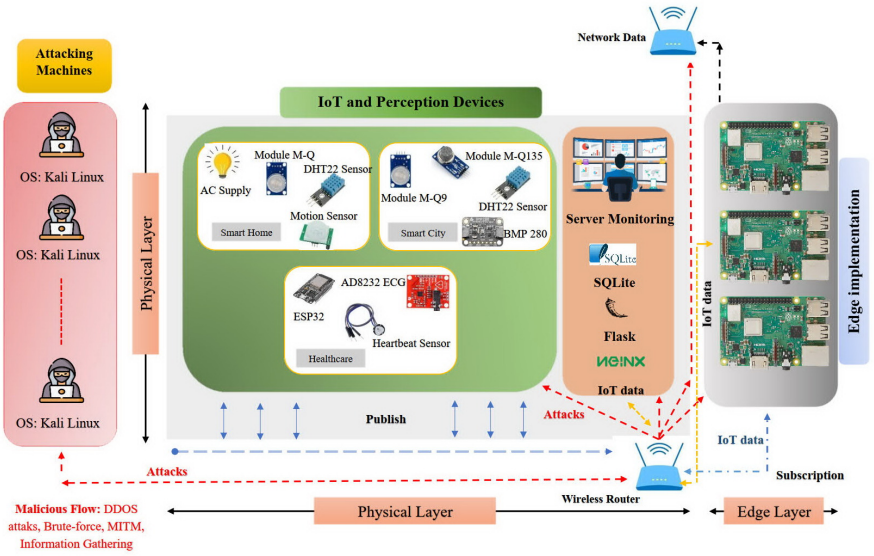

FIGURE 7 Testbed topology.

the command:

"sudo ufw default deny incoming"

This command serves as the basis of our defense strategy, guaranteeing that only connections started by the device in question are authorized. Simultaneously, we executed the following command to preserve the operational capabilities of the device, thereby enabling it to establish connections with trustworthy services for essential updates and data transfers:

"sudo ufw default to allow outgoing"

The implementation of a dual configuration that employs two complementary security measures to bolster its overall protection, while simultaneously maintaining the operational functionality of the network. The use of a dual configuration in this context serves to guarantee a resilient level of safeguarding, all the while maintaining the operating functionalities of the item. The UFW firewall enables us to selectively control incoming connections, by the strict demands of the OOC pattern, as exemplified by the following command:

"sudo ufw allow from [Server IP] to any port [Port Number] proto [Protocol]"

To optimize the configuration of the firewall, we can integrate precise rules. For instance, we can employ a directive to establish exact access restrictions, guaranteeing multi-tiered and resilient protection against potential risks.

PHZ: We implemented the PHZ by focusing on making a chosen whitelist. This whitelist is different because it includes devices that can be recognized by their Media Access Control (MAC) addresses. This makes sure that network communications are safe because the MAC addresses are a safe way to identify devices because they are unique. There are strict rules for controlling who can access our network equipment, which is made up of routers and switches. Only devices on the whitelist are allowed to connect to the network because of this strategy. One important part of our plan is to separate IP devices that are unknown. To do this, we considered several levels of protection. In the very strict network authentication process, the MAC addresses of devices trying to join the network are checked against a whitelist as part of a verification step. This method successfully finds and blocks devices that are not explicitly allowed, protecting our network from devices that are not supposed to be there.

TCP: The identification process of probable communication partners was initiated by employing Nmap[42], a comprehensive network scanning tool, rigorously. This process enabled a comprehensive record of authorized devices, including their MAC and IP addresses, to be securely documented. A Virtual Private Network (VPN) was implemented utilizing the OpenVPN protocol[43] to ensure the confidentiality and integrity of distant communications. This establishment of a VPN enables the creation of an encrypted channel, hence safeguarding the data exchange process and minimizing the potential vulnerability to interception.

BL: We implemented the BL management tool by constructing the Python and the Flask framework and merged it into the IoT network infrastructure. Using Python's libraries and the Flask framework, we were able to build a data-gathering module that can independently detect cases of network abuse. Python's implementation of SQLite proved capable of efficiently managing a blacklist dataset without jeopardizing its integrity[44]. Python scripts were used to establish connections with network devices to enforce the BL and put access rules into effect. The application made use of Flask's session management for encrypted web

TABLE 1 List of equipment for the testbed.

TestBed	Name	Description
Smart Home	Edge device	Raspberry Pi
	Node 1	DHT22 sensor
	Node 2	Module M-Q9
	Node 3	AC supply
	Node 4	MCP3008 converter
	Node 5	Motion sensor
	User	Connecting using Any device
	Router	Wireless Hub/ Connection Point
	Hacker	Intruder/ Cyber Attacker
Smart City	Edge device	Raspberry Pi
	Node 1	DHT22 sensor
	Node 2	Module M-Q9
	Node 3	BMP 280 (Air Pressure)
	Node 4	MCP3008 converter
	Node 5	Module MQ135 Sensor
	User	Connecting using Any device
	Router	Wireless Hub/ Connection Point
	Hacker	Intruder/ Cyber Attacker
Health Care	Edge device	Raspberry Pi
	Node 1	ESP32
	Node 2	AD8232 ECG Sensor
	Node 3	Heartbeat Sensor
	Node 4	DHT22 Sensor
	Node 5	Microcontroller ESP32 Module
	User	Connecting using Any device
	Router	Wireless Hub/ Connection Point
	Hacker	Intruder/ Cyber Attacker

communications and the Python cryptography tools to safeguard user data.

WL: To make our applications more secure, we have included a WL system built on top of the Flask framework[45]. At the heart of our WL system, there is a SQLite database. This database stores the API keys or user IDs of each user, together with the access privileges that the user has been granted. It also preserves a log of all modifications such as adding, removing, and updating user entries, so we will always know when they were made and who made them. Our WL performs a check against a predetermined list of authorized users at each attempted login. A user can proceed if they are on the approved list and have the necessary privileges. If they do not, they will not be able to enter.

SSN: We implemented the SNN by OpenSSL[46] to provide a comprehensive security architecture. Drawing from the work given in[47], we customized the current configuration to match the unique requirements of our testbed. To protect the sensor data during transmission, we set up a secured socket layer. This was crucial in establishing an encrypted connection between the sensor nodes and the database, protecting the privacy and security of the collected data. To ensure confidential exchanges, we made use of a dual encryption method, generating SSL keys (both public and private) with the help of RSA-2048[48]. Additionally, AES-256[49] was implemented to guarantee a high level of security for stored information. The data communication by the sensor nodes was made much more secure by employing a dual-layer encryption strategy.

TABLE 2 Patterns selected for experimentation.

Pattern	Abbreviation	Code
Personal Zone Hub	PZH	P_1
Trusted Communication Partner	TCP	P_2
Outbound-Only Connection	OOC	P_3
Blacklist	BL	P_4
Whitelist	WL	P_5
Secure Sensors	SSN	P_6
Without Pattern	-	P_0

4.4 | Metrics

We use the following metrics to capture the impact of the studied patterns on the edge systems.

CPU load refers to the amount of processing power that is currently being used by the CPU of a computer system. This measurement is typically expressed as a percentage of the CPU's total processing capacity, with a reading of 100% indicating that the CPU is fully occupied and has no remaining capacity. The CPU load can fluctuate depending on the tasks that the computer is performing, and it can be monitored using various system monitoring tools. When the CPU load is high, it can cause the computer to slow down or become unresponsive, especially if it persists for an extended period. Additionally, high CPU loads can result in increased power consumption and heat generation, which can be problematic for systems with limited cooling capabilities. It is crucial to keep a close watch on the CPU load to ensure that it remains within reasonable limits and does not negatively impact system performance or stability. Therefore, regular monitoring and analysis of the CPU load can help identify potential issues and take necessary steps to optimize system performance.

In the context of assessing the effects of attacks on applications, CPU load can be used to detect whether an application is under stress due to an attack. If an attacker is launching a DDoS attack, for example, the application's CPU load will likely increase significantly as it struggles to handle a large amount of traffic being directed at it. By monitoring the CPU load of an application, system administrators can detect and respond to attacks promptly, taking steps to mitigate the effects of the attack and prevent further damage. Monitoring CPU load can also help identify performance issues in an application that may be unrelated to an attack. By tracking CPU usage over time, administrators can identify trends and patterns that may indicate problems with the application's design or infrastructure, allowing them to make targeted improvements to improve overall performance and stability.

In this study, we employ PowerTOP[8] a widely used and effective tool for measuring and monitoring energy consumption, to quantify the energy consumption of specific processes or patterns.

4.5 | Analysis method for energy consumption and CPU usage

In this study, we employ the Mann-Whitney U test, also known as the Wilcoxon rank-sum test, as it was used in similar studies Khomh and Abtahizadeh[50]. It is used to test whether there is a significant difference in the distribution of a continuous variable between two independent groups. Since the findings indicate differences between several independent groups, we use the Mann-Whitney U test[51] to examine H_x^1, $H_{1,3}^2$, H_x^3, $H_{1,3}^4$, etc. Moreover, we used Cliff's δ effect size to determine the importance of the differences between metric values. Cliff's δ is a non-parametric effect-size measure that indicates the extent to which two sampling distributions overlap. All of our tests are performed at a 95% level of confidence (i.e., the p-value is less than 0.05). Because we conduct multiple tests of the null hypothesis, we use a Bonferroni correction[52] to deal with the issue of multiple comparisons, which involves dividing the threshold p-value by the number of tests.

4.6 | Designed attacks

In this section, we analyze the attacks we launched within our testbed environment to the efficiency and resilience of the security patterns employed.

- DDoS: DDoS attacks remain a significant danger in IoT networks due to the vast number of devices that could be compromised and utilized to launch such attacks[53]. IoT devices are frequently developed with insufficient processing capacity and security

TABLE 3 The name of patterns and method of implementing.

Pattern	Implementation
OOC	UFW firewall
PZH	Configure the network infrastructure, Implement isolation measures

measures, which leaves them open to exploitation by malicious actors. Many IoT devices operate in highly distributed environments, such as smart cities or industrial control systems. In these setups, a single susceptible device can become a point of entry for attackers, allowing them to access the wider network. Attackers frequently utilize DDoS to take advantage of weak IoT devices[54 55 56].

- SSH-MITM: This attack is still critical in an IoT network because it enables an attacker to secretly intercept and control the communication between IoT devices, applications, and servers without raising any suspicions. IoT devices generally communicate with one another over wireless networks, which, compared to wired networks, will not offer the same level of security citation required. An adversary may employ a variety of tactics to intercept the communication and attain unauthorized access to the data that these devices are sending. Many IoT devices are not built with robust security safeguards throughout the design process, making them especially susceptible to MITM attacks. This can be particularly concerning when IoT devices are used in essential infrastructure, such as healthcare or transportation, where a security breach can have serious repercussions[57 58].

- Brute-Force: Since the deployment of IoT applications has proliferated in recent years, connecting various devices to the internet is oftentimes without adequate security measures. Consequently, such devices are vulnerable to exploitation by malicious actors, who can launch brute-force attacks due to their worldwide accessibility. These attacks can be automated through the utilization of software tools, enabling cybercriminals to launch large-scale attacks with ease. This indicates that the hacker can target numerous devices simultaneously, increasing the likelihood that they will successfully acquire access to at least one of the targeted devices. After the hacker has gained access to an IoT device, they can use it as a gateway to access other IoT devices connected to the same network. This can provide them access to sensitive information or even give them control over the key infrastructure. Attacks using the brute-force pose a substantial risk to IoT networks. As such, they should be treated seriously by all parties involved, including manufacturers, developers, and end users[59 60].

5 | EXPERIMENTAL RESULTS

This section presents the results of our experiments, followed by a thorough analysis of the findings obtained through the utilization of the six selected patterns discussed in Section 3.2.

5.1 | OOC and PZH security patterns

5.1.1 | RQ1: The impact of OOC and PZH security patterns on security

To address RQ1, we conducted a penetration test on the applications available on our testbeds. The penetration test was executed using DDoS and MITM[61] attacks. Moreover, We selected these attacks for these patterns based on the concept of the OOC and PZH[62].

We have implemented the OOC pattern by installing the Uncomplicated Firewall (UFW) on devices. The UFW is the front end for the network filter firewall built on the Linux kernel. The UFW rules define which network traffic should be allowed or denied. Rules can be defined based on IP addresses, port numbers, and protocols of source and destination. In addition, as network traffic enters or exits the system, it is inspected by the network filter firewall. The firewall compares each packet against the defined UFW rules to determine whether it should be allowed or denied. If the packet matches a rule allowing traffic, it can pass through the firewall. If the packet matches a rule that denies the traffic, it is blocked, and a response is sent back to the source of the traffic.

Implementation:

The present study aims to implement the PZH pattern that involves identifying and isolating unidentified IP devices and allowing only recognized devices into the IoT network. To achieve this, we first identified the IP address range of the IoT network and then determined the range of IP addresses used in the network. This method enabled us to identify devices that were not

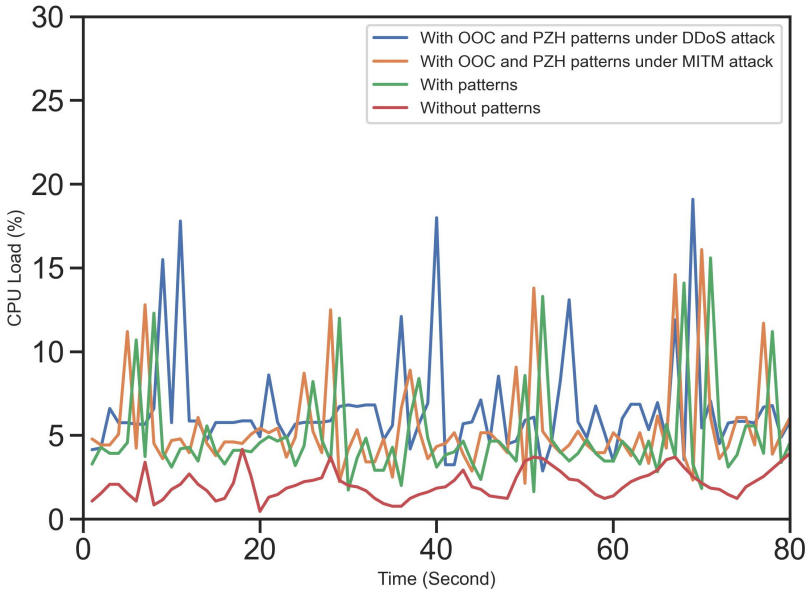

FIGURE 8 The CPU load for a smart home application with OOC, PZH patterns, without OOC, PZH patterns, and under DDoS and MITM attack.

recognized by the network.

Subsequently, we configured the IoT network infrastructure, including the router and switch, to permit only recognized devices to enter the network. To achieve this, we employed MAC address filtering as a security measure. Furthermore, we employed network monitoring tools such as Wireshark to detect devices that were not recognized by the network. These tools enabled us to isolate the devices that needed to be removed from the network.

It is worth noting that certain IoT devices may have dynamic IP addresses or utilize IP addresses that fall outside the recognized range of the IoT network. In such cases, alternative methods such as device fingerprinting or network traffic monitoring may be necessary to identify and isolate them. However, these approaches are beyond the scope of this research. The implementation of OOC and PZH patterns is summarized in Table 3 .

The findings of the experiment reveal that the tested applications exhibit resilience against DDoS attacks, albeit with a consequential increase in CPU load. While the applications experienced a slowdown in performance, they were able to withstand the attack due to the presence of OOC and PZH patterns. However, the experiment also highlights that the applications were vulnerable to MITM attacks. The MITM attack spoofed the address protocol by assuming the identity of the router to receive the target's packets, thereby bypassing the protective measures of the PZH pattern. The experiment's results indicate that the PZH pattern may not be an effective defense mechanism against MITM attacks, as this type of attack can forge the router's MAC address. Furthermore, the experiment demonstrates that the OOC and PZH patterns proved ineffective in IoT applications due to sensor data loss. Therefore, our findings suggest that alternative security measures be considered for IoT applications to address the vulnerabilities highlighted in the experiment.

Next, the CPU load of smart home, smart city, and healthcare applications was evaluated under normal conditions and during a DDoS attack. Results are shown in Fig. 8 , 9 and 10 those were obtained through simulation using a DDoS and MITM attack. The OOC and PZH patterns were identified in the simulation as affecting the applications' CPU load. Without the presence of these patterns, the CPU loads of the applications ranged between 0.5% and 2.75%. However, the introduction of OOC and PZH patterns resulted in a significant increase in CPU load, with values ranging from 1% to 16%. This represents an approximately six-fold increase in CPU load compared to normal conditions (that means in a normal running condition). During the attack, the CPU load with OOC and PZH patterns was found to be between 16.5% and 19%. This increase in CPU load caused the applications to respond slowly as the additional resources needed to serve these loads were being allocated. The presence of OOC and PZH patterns significantly increases the CPU load of a smart home, smart city, and healthcare applications, even without any attack. With the presence of any attack, the impact on CPU load is even greater.

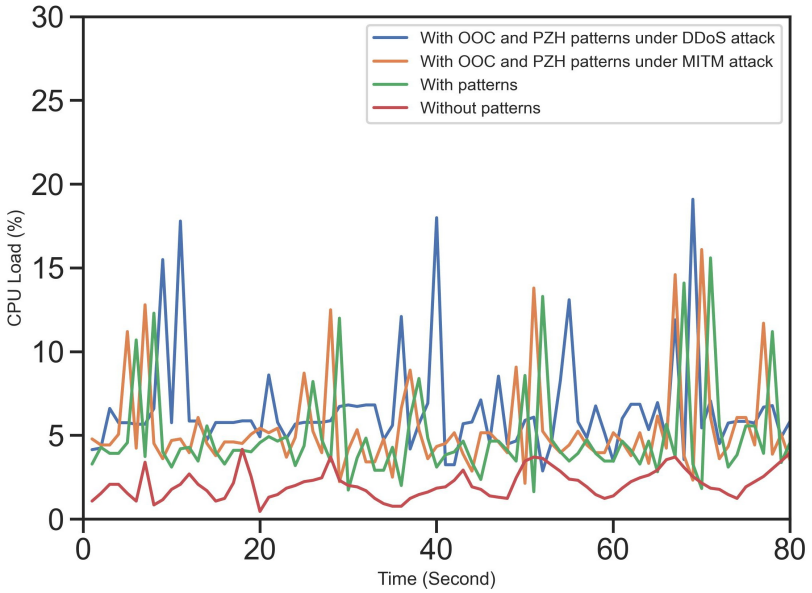

FIGURE 9 The CPU load for smart city applications with OOC, PZH patterns, without OOC, PZH patterns, and under DDoS and MITM attacks.

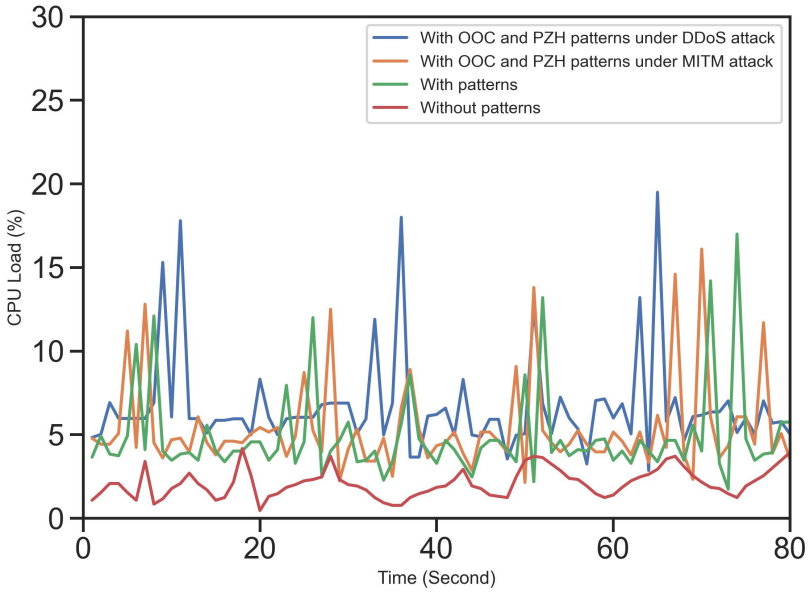

FIGURE 10 The CPU load for a healthcare application with OOC, PZH patterns, without OOC, PZH patterns, and under DDoS and MITM attacks.

5.1.2 | RQ2: Impact on energy consumption and CPU usage

The following set of null hypotheses are defined to answer RQ2: H^x, $x \in \{1....6\}$. Hypotheses are defined as follows where P_0 corresponds to the version of application that does not use patterns, $P_{1,3}$ indicates the applied patterns, namely PZH and OOC (see Table 2), $P_{1,3} - DDoS$ and $P_{1,3} - MITM$ represent the occurrence or presence of the DDoS and the MITM attacks in $P_{1,3}$, respectively:

$H^1_{1,3}$: There exists no difference in the average energy consumption between $P_{1,3}$ and P_0.

$H^2_{1,3}$: There is no difference in the average energy consumption between $P_{1,3}$ and $P_{1,3} - DDoS$.

TABLE 4 P-value of Mann–Whitney U test and Cliff's δ effect size for the average energy consumption under MITM and DDoS attack.

Version	Avg. Energy consumption					
	SmartHome		SmartCity		HealthCare	
	P-Value	Effect Size	P-Value	Effect Size	P-Value	Effect Size
P_0 vs. $P_{1,3}$	0.0001	0.1	0.0001	0.0	0.0001	0.0
$P_{1,3}$ vs. $P_{1,3} - MITM$	0.0226	0.493	0.3173	0.218	0.9840	0.112
$P_{1,3}$ vs. $P_{1,3} - DDoS$	0.0120	0.542	0.0238	0.428	0.0794	0.578

TABLE 5 P-value of Mann–Whitney U test and Cliff's δ effect size for the average CPU usage under MITM and DDoS attack.

Version	Avg. CPU Usage					
	SmartHome		SmartCity		HealthCare	
	P-Value	Effect Size	P-Value	Effect Size	P-Value	Effect Size
P_0 vs. $P_{1,3}$	0.0105	0.73	0.3320	0.213	0.0818	0.738
$P_{1,3}$ vs. $P_{1,3} - MITM$	0.0004	0.165	0.0009	0.667	0.0054	0.747
$P_{1,3}$ vs. $P_{1,3} - DDoS$	0.0001	1.0	0.0001	1.0	0.0001	1.0

$H^3_{1,3}$: The average amount of energy consumed by $P_{1,3}$ is not different from the energy consumed by $P_{1,3} - MITM$.
$H^4_{1,3}$: There is no difference between the average amount of CPU usage of $P_{1,3}$ and P_0.
$H^5_{1,3}$: There is no difference between the average CPU usage $P_{1,3}$ and the CPU usage observed for $P_{1,3} - DDoS$.
$H^6_{1,3}$: The average amount of CPU usage in $P_{1,3}$ is not different from $P_{1,3} - MITM$.

Table 4 reports the P-values of the Mann-Whitney U test and Cliff's δ effect size for energy consumption evaluation. The table specifically compares the average energy consumption observed during two distinct attack scenarios (MITM and DDoS), with/without security patterns.

According to the P-value in Table 4 , we reject $H^1_{1,3}$ for all applications. Based on the data, the evidence suggests that a statistically significant disparity exists in the average energy consumption levels across three distinct applications when comparing the utilization of OOC and PZH patterns without them.

According to the P-value for the MITM attack, we reject $H^3_{1,3}$ for the smart home application since the data analysis reveals a statistically significant difference in the energy consumption by $P_{1,3} - MITM$. However, we cannot reject $H^3_{1,3}$ for the smart city and healthcare cases. The data analysis reveals no statistically significant difference in the energy consumption by $P_{1,3} - MITM$.

According to the P-value for the DDoS attack, we reject $H^2_{1,3}$ for the smart home and smart city applications. The findings indicate a statistically significant difference in the energy consumption of by $P_{1,3} - DDoS$. However, we cannot reject $H^2_{1,3}$ for the healthcare applications, implying that there is no significant difference from the energy consumed by $P_{1,3} - DDoS$.

Fig. 11 , 12 , and 13 show the results obtained for all the implementations of the OOC and PZH patterns. The examination of energy consumption patterns reveals a discernible escalation in energy utilization. This effect becomes more pronounced when subjecting the applications (with patterns) are under DDoS and MITM attacks, further exacerbating energy consumption levels. Nonetheless, a statistically noteworthy disparity in energy consumption becomes evident when comparing the with/without patterns in all applications. Additionally, the disparity in energy consumption attains statistical significance when assessing the impact of an MITM attack on a smart home environment. Furthermore, statistical significance manifests in energy consumption discrepancies when the smart home and city confront DDoS attack scenarios.

Table 5 reports the P-values of the Mann-Whitney U test and the Cliff's δ effect size for CPU usage evaluation. The table specifically compares the average CPU usage levels observed during two distinct attack scenarios (MITM and DDoS), with/without security patterns.

According to the P-value in Table 5 , we reject $H^4_{1,3}$ for smart home application. Our analysis indicates a statistically significant difference in the average CPU usage when OOC and PZH patterns are applied in the smart home application. However, we cannot reject $H^4_{1,3}$ smart city and healthcare applications. There is no statistically significant difference between the average amount of CPU usage under different conditions with OOC and PZH patterns and without them in the two applications.

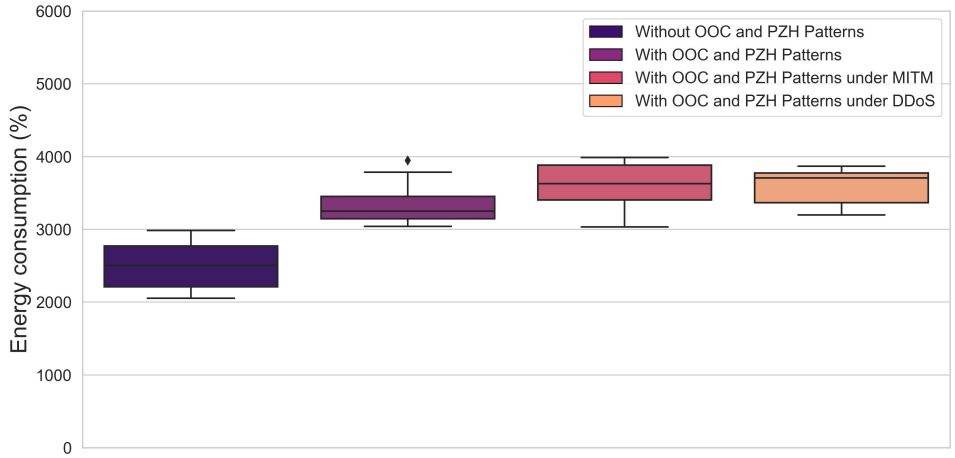

FIGURE 11 Energy consumption for a smart home application with OOC, PZH patterns, without OOC, PZH patterns, and under DDoS and MITM attacks.

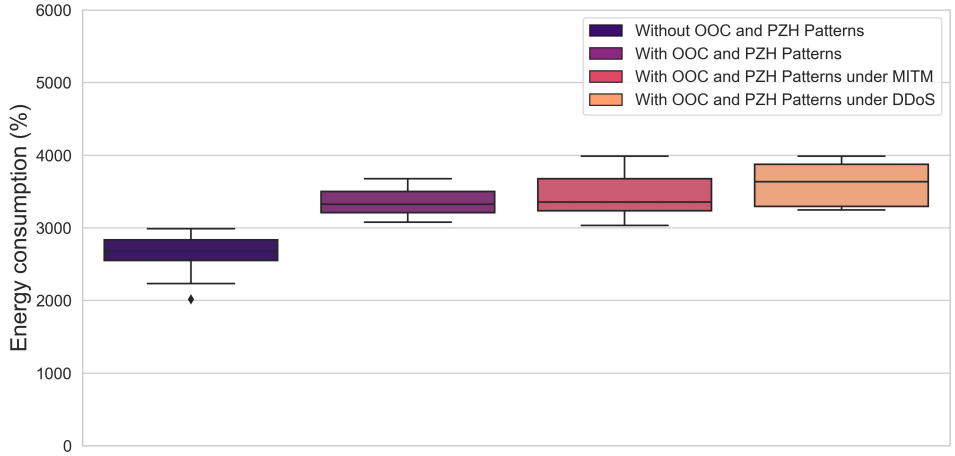

FIGURE 12 Energy consumption for a smart city application with OOC, PZH patterns, without OOC, PZH patterns, and under DDoS and MITM attacks.

For the MITM attack, we reject $H^6_{1,3}$ for all applications. Upon conducting the statistical tests, it is evident that there is a statistically significant difference in CPU usage by $P_{1,3} - MITM$.
In the case of the DDoS attack, we reject $H^6_{1,3}$ for all applications. After conducting the statistical tests, compelling evidence arises, indicating a statistically significant difference in CPU usage caused by $P_{1,3} - DDoS$.
Fig. 14 , 15 , and 16 summarize the results obtained for all the implementations of the OOC and PZH patterns. The examination of CPU usage during DDoS attacks reveals an increase when patterns are introduced. In addition, it is imperative to emphasize that a statistically significant variance in CPU usage is observed between scenarios for the smart home with and without patterns unless for the smart city and healthcare case. Moreover, a statistically significant variance in CPU usage is observed when applications are under DDoS and MITM attacks.

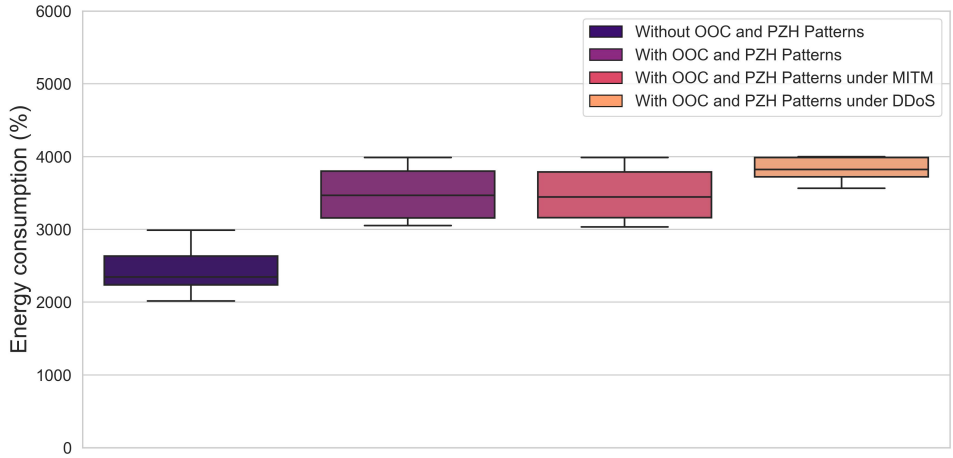

FIGURE 13 Energy consumption for a healthcare application with OOC, PZH patterns, without OOC, PZH patterns, and under DDoS and MITM attacks.

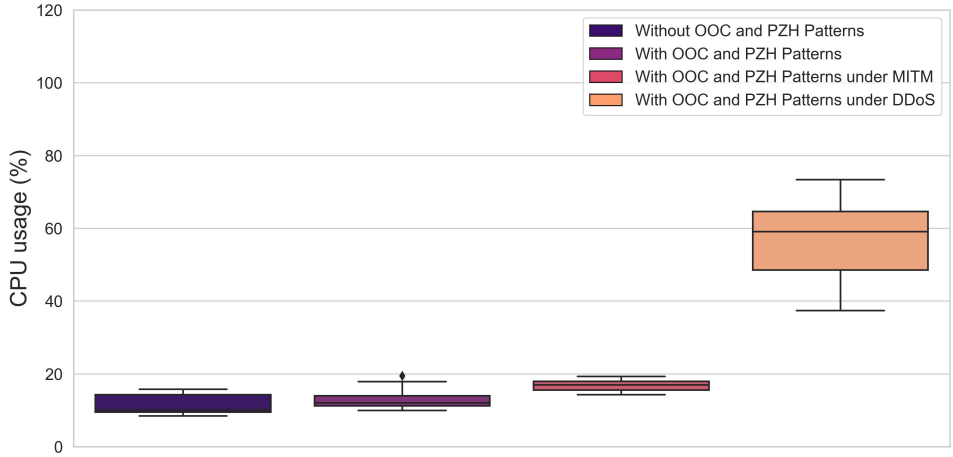

FIGURE 14 CPU usage for a smart home application with OOC, PZH patterns, without OOC, PZH patterns, and under DDoS and MITM attack.

Finding 1:

The PZH mechanism exhibits limitations in effectively countering MITM attacks. Additionally, the pattern's impact on energy consumption in all applications becomes evident in diverse situations, including those involving attacks and non-attack circumstances, except the smart city and healthcare under MITM attack and healthcare under DDoS attack. Furthermore, the patterns' impact on CPU usage in all applications becomes evident in diverse situations, including attacks and non-attack circumstances, except for the smart city and healthcare with/without patterns.

5.2 | WL, BL and TCP

5.2.1 | RQ1: The impact of WL, BL, and TCP security patterns on security

To address RQ1, three applications were developed with and without the utilization of WL and BL[62] patterns. The implementation involved the integration of an authentication system on both the Raspberry Pis and the server. The efficacy of the security

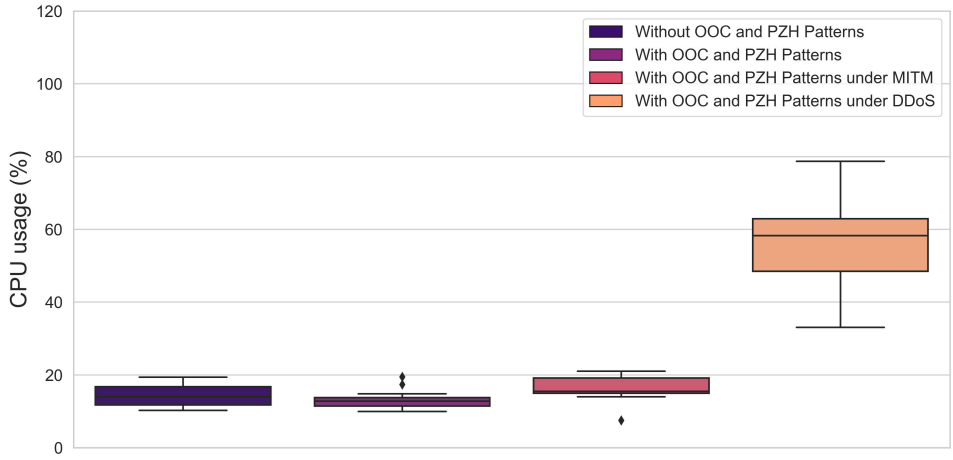

FIGURE 15 CPU usage for smart city applications with OOC, PZH patterns, without OOC, PZH patterns, and under DDoS and MITM attacks.

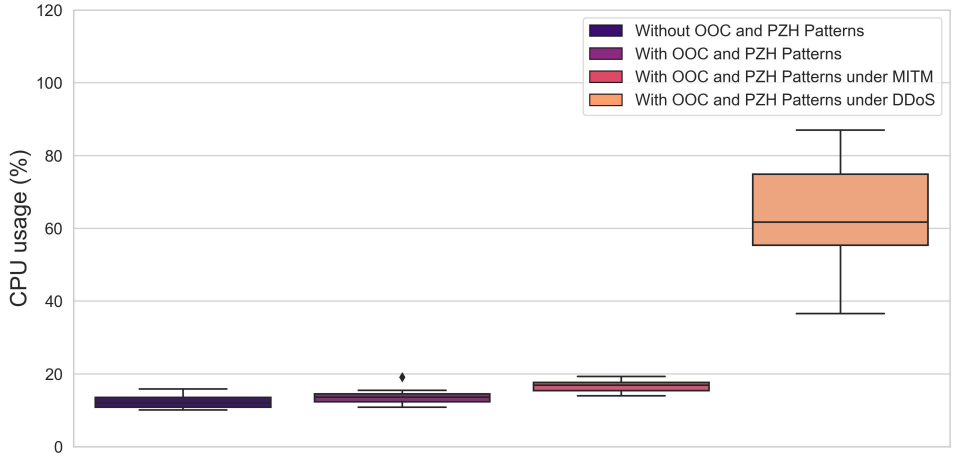

FIGURE 16 CPU usage for a healthcare application with OOC, PZH patterns, without OOC, PZH patterns, and under DDoS and MITM attacks.

TABLE 6 The name of patterns and method of implementation.

Pattern	Implementation
Whitelist	List of users, Device authentication
Blacklist	Forbidden List, Limit login attempts

measures was evaluated through the use of a penetration test as a brute-force[63] attack against the developed applications. This approach was adopted to evaluate the effectiveness of the WL and BL patterns, which informed the selection of this particular security testing methodology.

Implementation:

The present study outlines the implementation of SQLite database in IoT-based-edge applications to incorporate WL and BL. Specifically, a new attribute named access level was added to the database schema, enabling control and limitation of user accessibility within the system[64]. In cases where a user with WL privileges misuses their access rights, the administrator of the system

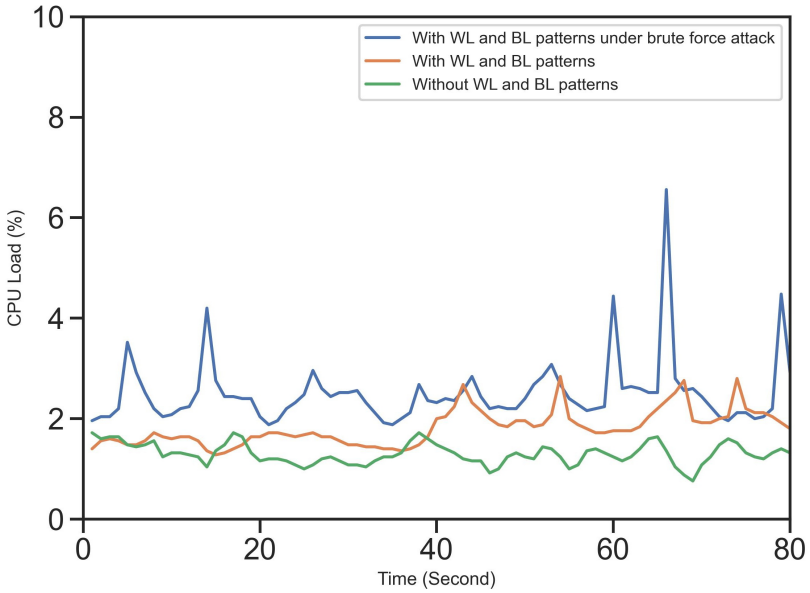

FIGURE 17 CPU load for a smart home application with WL, BL without WL, BL, and under BF attack.

can revoke their privileges and assign them to the BL. The administrator can also exercise control over the access of WL users. Additionally, high-performance computers were deployed to execute attacks that involved a large number of calculations per second, thereby enabling the testing of numerous combinations of usernames and passwords in the shortest possible time. Moreover, a limit was imposed on the number of login attempts to prevent unlimited password combinations by potential attackers. Table 6 illustrates the implementation of WL and BL patterns in the considered applications. The experimental results confirmed that stronger passwords, with a minimum length of 12 characters and a mix of uppercase and lowercase letters, numbers, and symbols, can enhance the security of the applications[47]. Overall, the incorporation of WL and BL functionalities has been found to enhance the security of IoT-based-edge applications.

Results:

Fig. 17 , 18 and 19 depict the three states of the smart home, smart city, and healthcare applications, respectively. The depicted CPU load highlights the potential impact of the WL and BL pattern under the brute-force attack on the normal state of the applications. Notably, the CPU load of the considered applications without WL and BL patterns ranges from 0.5% to 1.75%. The introduction of the WL and BL patterns increases the CPU load to a range of 1% to 2.5%. Under the brute-force attack, the CPU load increases to a range of 2% to 5.5%. Thus, the findings demonstrate the varying impact of WL and BL patterns on CPU load in different contexts.

5.2.2 | RQ2: Impact on energy consumption and CPU usage

The following set of null hypotheses are defined to answer RQ2: H^x, $x \in \{1....6\}$. Hypotheses are defined as follows where P_0 corresponds to the version of applications that do not use patterns, $P_{4,5}$ indicates the applied patterns, namely WL and BL (see Table 2), $P_{4,5} - BF$ indicates the occurrence or presence of brute-force attack in $P_{4,5}$, respectively:

$H^1_{4,5}$: There is no difference between the average energy consumption of $P_{4,5}$ and P_0.

$H^2_{4,5}$: The average amount energy consumption of $P_{4,5}$ is not different from $P_{4,5} - BF$.

$H^3_{4,5}$: There is no difference between the average amount of CPU usage of $P_{4,5}$ and P_0.

$H^4_{4,5}$: The average amount of CPU usage by $P_{4,5}$ is not different from the $P_{4,5} - BF$.

Table 7 reports the P-values of the Mann-Whitney U test and Cliff's δ effect size for energy consumption evaluation. The table specifically compares the average energy consumption levels observed during distinct attack scenarios (brute-force), with/without security patterns.

According to the P-value in Table 7 , we reject $H^1_{4,5}$; the analysis reveals a statistically significant difference in the average energy consumption when WL and BL patterns were applied in all applications.

For the brute-force attack, we reject $H^2_{4,5}$ for all applications. The results reveal that there is a statistically significant difference

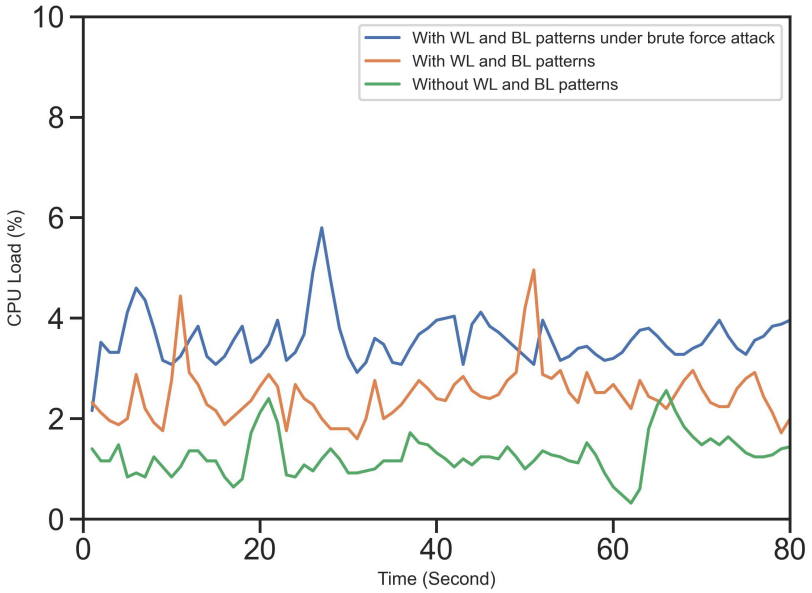

FIGURE 18 CPU load for a smart city application with WL, BL without WL, BL, and under BF attack.

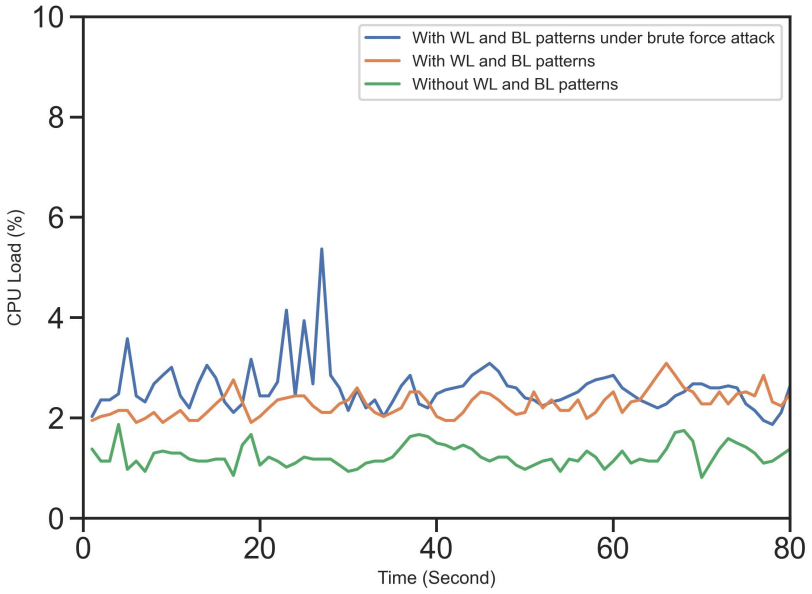

FIGURE 19 CPU load for a healthcare application with WL, BL without WL, BL, and under BF attack.

between the average amount of energy consumed when using WL and BL patterns for all examined applications under $P_{4,5} - BF$. Fig. 20 , 21 , and 22 illustrates the results obtained for all the implementations of the WL and BL patterns. The analysis of energy consumption during a brute-force attack indicates an escalation when patterns are introduced. Also, our results show a statistically significant difference in any of the cases under investigation.

Table 8 reports the P-values of the Mann-Whitney U test and the Cliff's δ effect size for CPU usage evaluation. The table specifically compares the average CPU usage observed during distinct attack scenarios (i.e., brute-force) with/without security patterns.

According to the P-value in Table 8 , we reject $H^3_{4,5}$ in smart home and smart city applications. The analysis unequivocally demonstrates a statistically significant difference in the average CPU usage across various cases, considering the presence of WL and BL patterns, compared to cases without WL and BL in smart home and smart city applications. Moreover, we cannot

TABLE 7 P-value of Mann–Whitney U test and Cliff's δ effect size for the average energy consumption under brute-force attack.

Version	Avg. Energy Consumption					
	SmartHome		SmartCity		HealthCare	
	P-Value	Effect Size	P-Value	Effect Size	P-Value	Effect Size
P_0 vs. $P_{4,5}$	0.0001	0.0	0.0001	0.0	0.0001	0.867
$P_{4,5}$ vs. $P_{4,5} - BF$	0.0008	0.819	0.0001	0.929	0.0018	0.809

TABLE 8 P-value of Mann–Whitney U test and Cliff's δ effect size for the average CPU usage under brute-force attack.

Version	Avg. CPU Usage					
	SmartHome		SmartCity		HealthCare	
	P-Value	Effect Size	P-Value	Effect Size	P-Value	Effect Size
P_0 vs. $P_{4,5}$	0.0028	0.782	0.0057	0.391	0.2996	0.471
$P_{4,5}$ vs. $P_{4,5} - BF$	0.0001	1.0	0.0001	1.0	0.0001	0.996

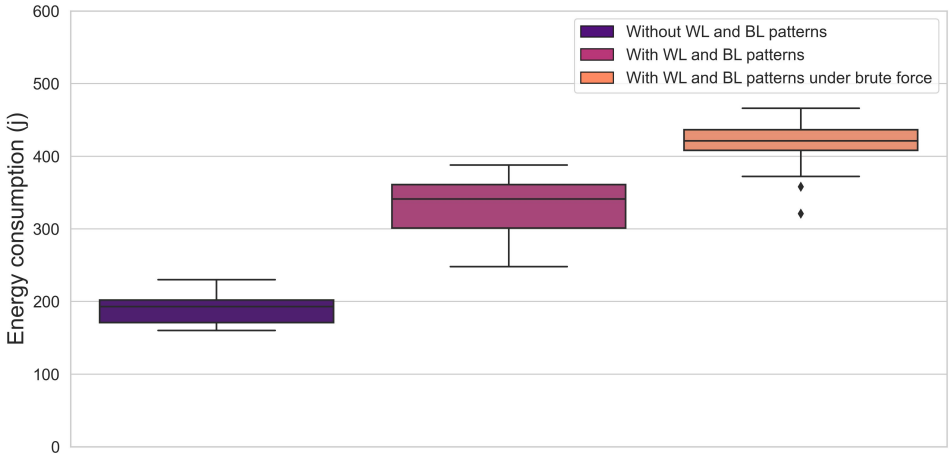

FIGURE 20 Energy consumption for a smart home application with WL, BL without WL, BL, and under BF attack.

reject $H^4_{4,5}$ for healthcare applications as there is no statistically significant difference in the CPU usage with/without WL, and BL patterns.

For the brute-force attack, we reject $H^3_{4,5}$ in all applications. The analysis unequivocally demonstrates a statistically significant difference in the average CPU usage across various cases when using $P_{4,5} - BF$.

Fig. 23 , 24 , and 25 show the results obtained for all the implementations of the WL and BL patterns. The analysis of CPU usage during a brute-force attack indicates an escalation when patterns are introduced. However, our analysis identifies a statistically significant difference in any of the cases under investigation, except healthcare with/without patterns.

Finding 2:

Our experimental findings suggest the efficacy of WL and BL patterns against brute-force attacks. Notably, incorporating patterns impacts CPU usage and energy consumption in active-attack scenarios and the absence of attacks. Also, results show a statistically significant difference in any of the cases under investigation.

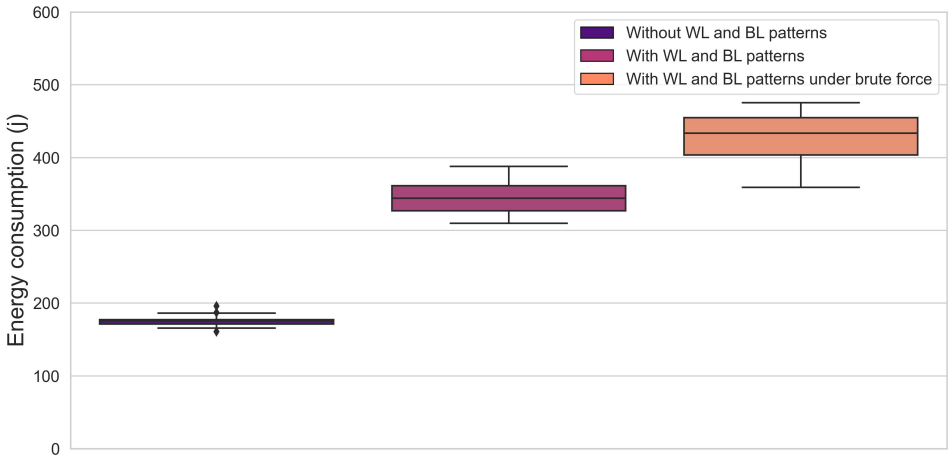

FIGURE 21 Energy consumption for a smart city application with WL, BL without WL, BL, and under BF attack.

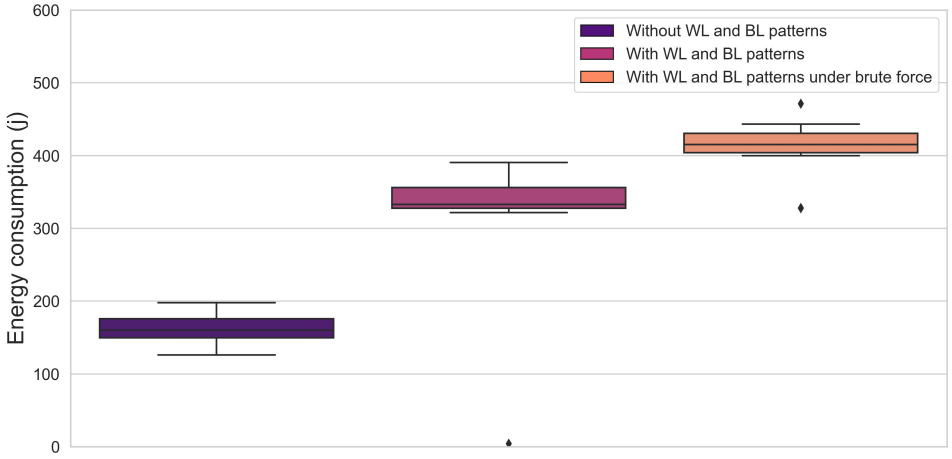

FIGURE 22 Energy consumption for a healthcare application with WL, BL without WL, BL, and under BF attack.

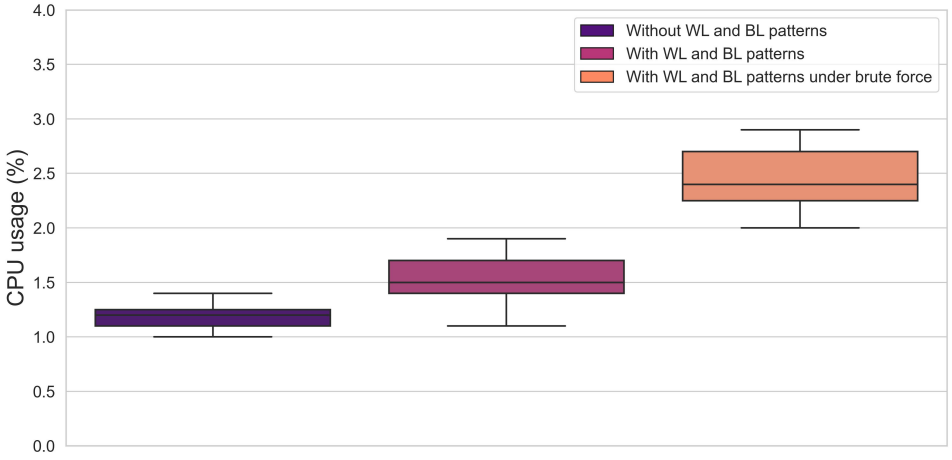

FIGURE 23 CPU usage for a smart home application with WL, BL without WL, BL, and under BF attack.

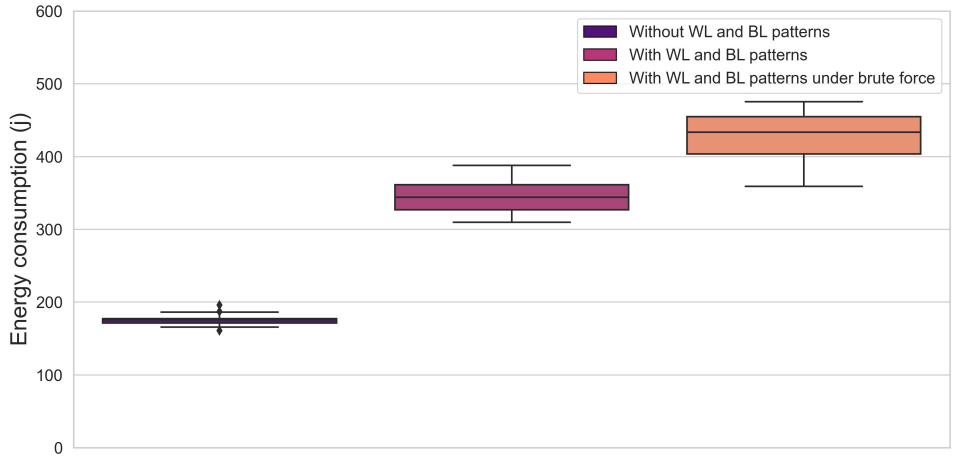

FIGURE 24 CPU usage for a smart city application with WL, BL without WL, BL, and under BF attack.

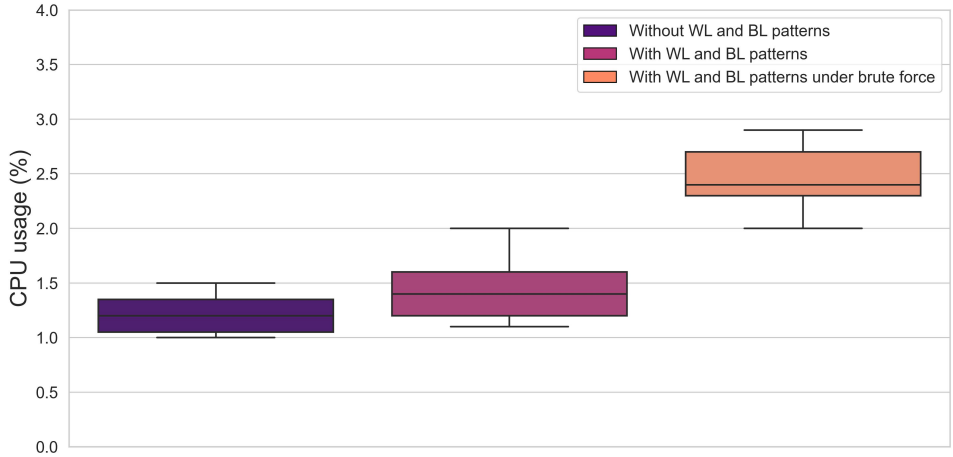

FIGURE 25 CPU usage for a healthcare application with WL, BL without WL, BL, and under BF attack.

5.3 | WL, BL and SSN

5.3.1 | RQ1: The impact of WL, BL, and SSN security patterns on security

This section encompasses an empirical evaluation aimed at assessing the efficacy of two distinct security patterns in the context of attack management. We aimed to answer RQ1 by implementing two applications with and without WL and BL patterns. In addition, we added an SSN pattern to ensure secure data transmission over an unsecured network. Brute-force and MITM attacks were implemented to assess the application's effectiveness. Table 9 provides a comprehensive overview of the implemented SSN pattern. However, the penetration test revealed that MITM was not able to log all entered data, including the user's password, which could not be returned to the legitimate user. The findings showed that the WL, BL, and SSN combination effectively managed brute-force and MITM attacks, which we will show in the following sections.

Implementation:

To implement the SSN and OpenSSL IoT security patterns in our testbed, we adapted the configuration discussed by[48]. Additionally, a secured socket was employed to facilitate the transfer of sensor data to the database. We used a combination of RSA-2048[49] and AES-256[65] to generate SSL keys (i.e., public and private keys for communication). During the implementation of the security patterns, measures were taken to ensure that the IoT hubs depicted in Fig. 1 were not directly accessible by external users. The implementation of these measures involved configuring the firewall at each of the Raspberry Pis' interfaces

TABLE 9 The name of patterns and method of implementation.

Pattern	Implementation
Secure Sensor Node	RSA-2048, AES-256

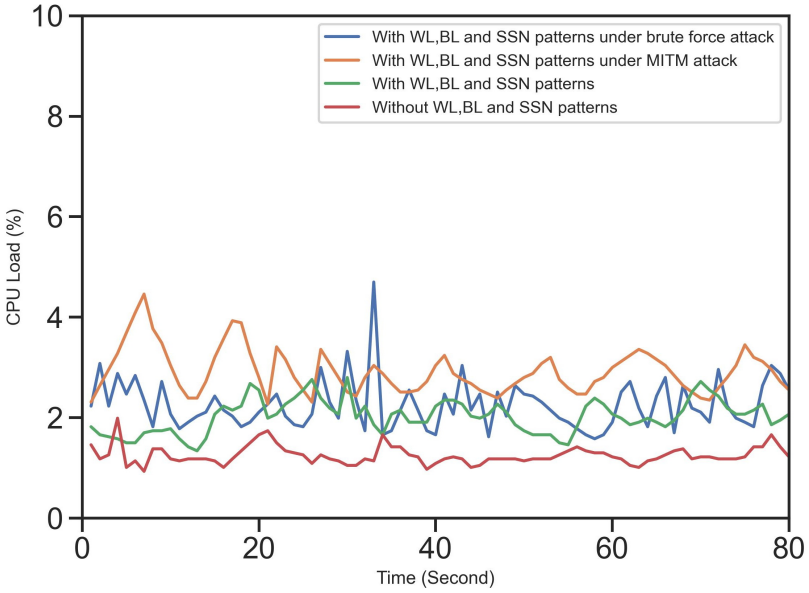

FIGURE 26 CPU load for smart home applications with WL, BL, SSN, without WL, BL, SSN, and under BF attack as well as MITM.

in such a way that only communication from the server IP address was permitted. Consequently, any user seeking to access the IoT hubs was required to go through the main server, which is responsible for managing the functions of the IoT hubs and storing the sensor data collected from different hubs. This approach guarantees a higher level of security, thereby reducing the risk of unauthorized access to IoT devices, which could potentially compromise sensitive data.

Results:

Fig. 26 , 27 and 28 , illustrate the impact of two security patterns, namely WL, BL, and SSN, on the performance of a smart home, smart city, and healthcare applications, respectively. Specifically, we investigate the CPU load of these applications in two different scenarios, i.e., normal operation under brute-force and MITM attacks. The findings indicate that the absence of the aforementioned security patterns results in a CPU load ranging from 0.5% to 2%. Conversely, the employment of these security patterns findings in a CPU load ranging from 2% to 3.5%. Notably, under the MITM attack, the CPU load increases from 2.5% to 4.5%.

5.3.2 | RQ2: Impact on energy consumption and CPU usage

The following set of null hypotheses are defined to answer RQ2: H^x, $x \in \{1....6\}$. Hypotheses are defined as follows where P_0 corresponds to the version of application that does not use patterns, $P_{4,5,6}$ indicates the applied patterns, namely WL, BL, and SSN (see Table 2), $P_{4,5,6} - BF$ and $P_{4,5,6} - MITM$ indicate the occurrence or presence of brute-force and MITM attacks in $P_{4,5,6}$, respectively:

$H^1_{4,5,6}$: There exists no difference between the average amount of energy consumed by $P_{4,5,6}$ and P_0.
$H^2_{4,5,6}$: There is no difference in the average energy consumption between $P_{4,5,6}$ and $P_{4,5,6} - MITM$.
$H^3_{4,5,6}$: The average amount of the CPU usage by $P_{4,5,6}$ is not different from $P_{4,5,6} - BF$.
$H^4_{4,5,6}$: There is no difference between the average amount of CPU usage by $P_{4,5,6}$ and P_0.
$H^5_{4,5,6}$: The average CPU usage of $P_{4,5,6}$ exhibits no difference when compared to $P_{4,5,6} - MITM$.
$H^6_{4,5,6}$: The average CPU usage in $P_{4,5,6}$ is not different from $P_{4,5,6} - BF$.

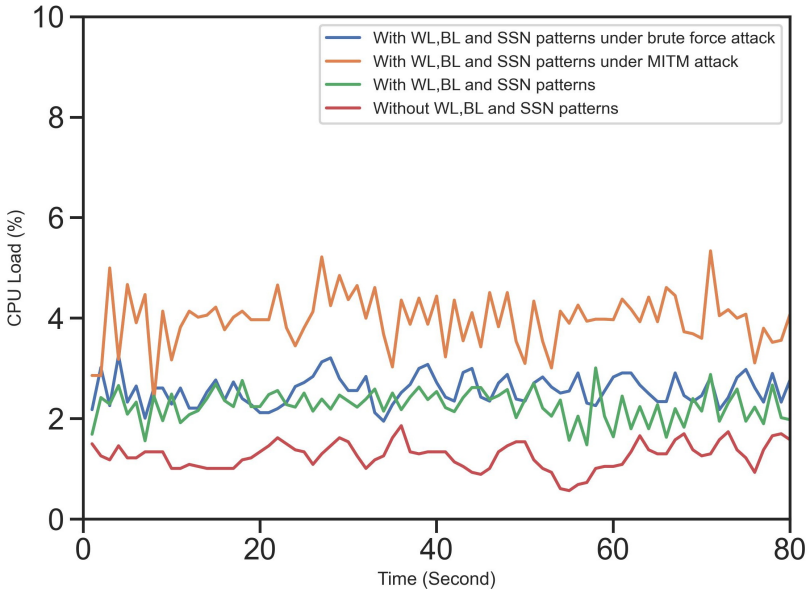

FIGURE 27 CPU load for smart city applications with WL, BL, SSN, without WL, BL, SSN, and under BF attack as well as MITM.

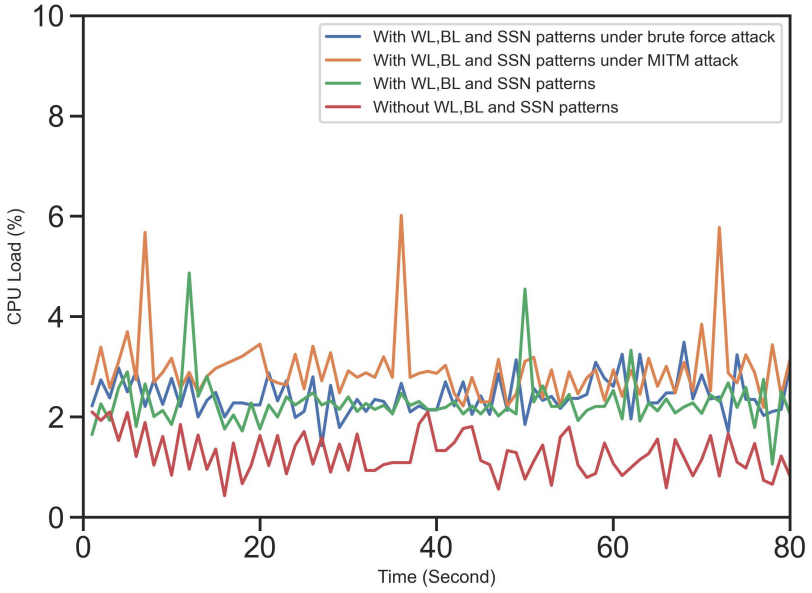

FIGURE 28 CPU load for healthcare applications with WL, BL, SSN, without WL, BL, SSN, and under BF attack as well as MITM.

Table 10 reports the P-values of the Mann-Whitney U test and Cliff's δ effect size for energy consumption evaluation. The table specifically compares the average energy consumption observed during two distinct attack scenarios (MITM, brute-force) with/without security patterns.

According to the P-value reported in Table 10 , we reject $H^1_{4,5,6}$ for all applications; based on the observed data, there exists a statistically significant disparity in the average energy consumption across various cases when considering the presence or absence of WL, BL, and SSN patterns in all applications.

For the MITM attack, We also reject $H^2_{4,5,6}$ for smart home and smart city applications as the data-driven analysis corroborates the belief that there is a statistically significant difference in the energy consumed by $P_{4,5,6} - MITM$. Moreover, we cannot

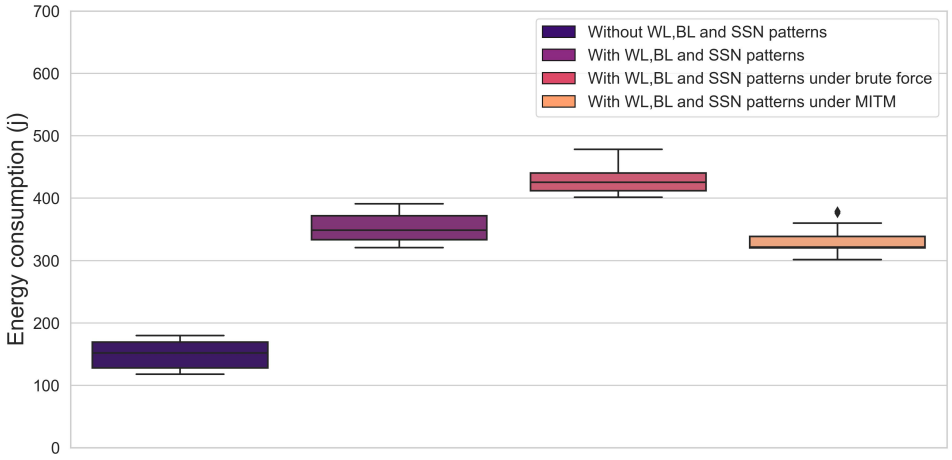

FIGURE 29 Energy consumption for smart home applications with WL, BL, SSN, without WL, BL, SSN, and under BF attack as well as MITM.

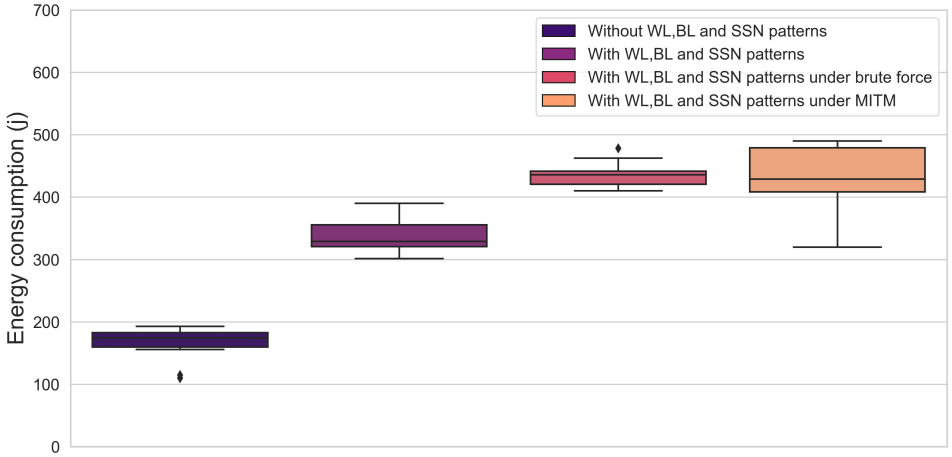

FIGURE 30 Energy consumption for smart city applications with WL, BL, SSN, without WL, BL, SSN, and under BF attack as well as MITM.

reject $H^2_{4,5,6}$ for the healthcare application. The analysis unequivocally demonstrates no statistically significant difference in the average CPU usage by using $P_{4,5,6} - MITM$.

For the brute-force attack, we reject $H^3_{4,5,6}$ in all applications. The data reveals a statistically significant difference in the energy consumed by $P_{4,5,6} - BF$. Fig. 29 , 30 , and 31 show the results obtained for all investigated scenarios for WL, BL, and SSN patterns. The analysis of energy consumption during brute-force and MITM attacks reveals a jump when patterns are introduced; however, there is a statistically significant variance in energy consumption. This consistent observation applies to three applications under investigation, irrespective of with and without patterns and under attack, except healthcare under MITM attack.

Table 11 reports the P-values of the Mann-Whitney U test and Cliff's δ effect size for CPU usage evaluation. The table specifically compares the average CPU usage observed during two distinct attack scenarios (MITM, brute-force) with/without security patterns.

According to the P-value we reported in Table 11 , we reject $H^4_{4,5,6}$ for all applications, as there is a statistically significant difference between the average amount of CPU usage with and without WL, BL, and SSN patterns in all cases.

In the case of the MITM attack, we reject $H^5_{4,5,6}$ for all applications. The analysis reveals a statistically significant disparity in

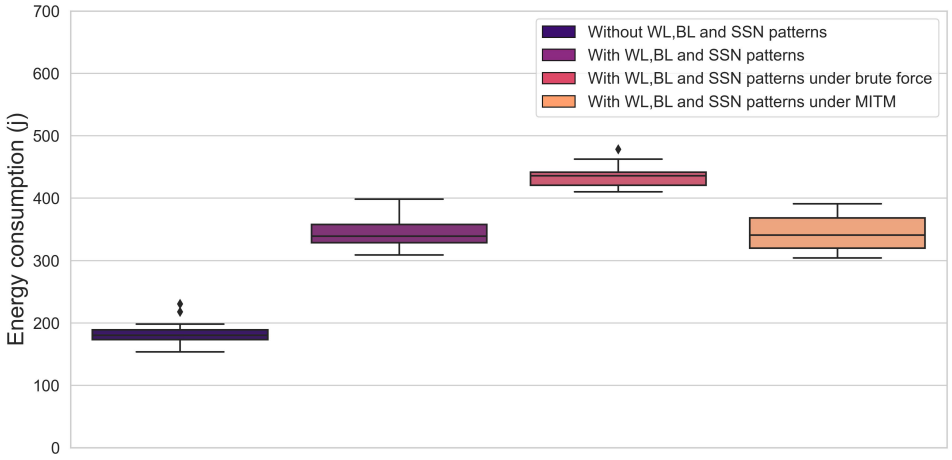

FIGURE 31 Energy consumption for healthcare applications with WL, BL, SSN, without WL, BL, SSN, and under BF attack as well as MITM.

TABLE 10 P-value of Mann–Whitney U test and Cliff's δ effect size for the average energy consumption under MITM and brute-force attacks.

Version	Avg. Energy Consumption					
	SmartHome		SmartCity		HealthCare	
	P-Value	Effect Size	P-Value	Effect Size	P-Value	Effect Size
P_0 vs. $P_{4,5,6}$	0.0001	1.0	0.0009	1.0	0.0005	1.0
$P_{4,5,6}$ vs. $P_{4,5,6} - MITM$	0.0002	1.0	0.0001	0.893	0.6599	0.098
$P_{4,5,6}$ vs. $P_{4,5,6} - BF$	0.0001	1.0	0.0001	1.0	0.0006	1.0

TABLE 11 P-value of Mann–Whitney U test and Cliff's δ effect size for the average CPU usage under MITM and brute-force attacks.

Version	Avg. CPU Usage					
	SmartHome		SmartCity		HealthCare	
	P-Value	Effect Size	P-Value	Effect Size	P-Value	Effect Size
P_0 vs. $P_{4,5,6}$	0.0008	1.0	0.0007	1.0	0.0001	1.0
$P_{4,5,6}$ vs. $P_{4,5,6} - MITM$	0.0001	1.0	0.0002	0.893	0.0007	0.098
$P_{4,5,6}$ vs. $P_{4,5,6} - BF$	0.0001	1.0	0.0003	1.0	0.0005	1.0

the CPU usage of $P_{4,5,6} - MITM$.
For the brute-force attack, we reject $H^6_{4,5,6}$ for all applications. The data does indicate a statistically significant difference in the CPU usage of $P_{4,5,6} - BF$.
Fig. 32 , 33 , and 34 illustrate the results we obtained for all implementations of WL, BL, and SSN patterns. The investigation of CPU usage during brute-force and MITM attacks yields an escalation when patterns are introduced. However, it is crucial to emphasize that there is a statistically significant variance in CPU usage and every consumption among scenarios under attack. This observation holds true for three applications under investigation, except healthcare under MITM attack in energy consumption.

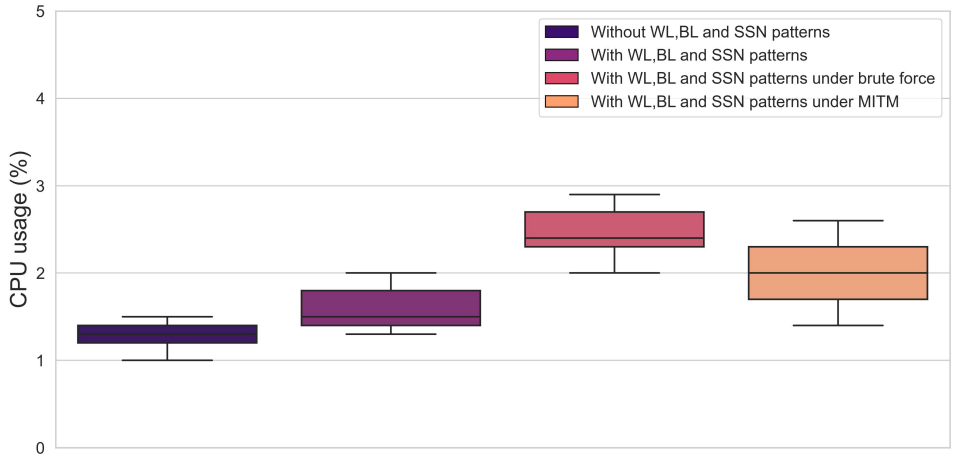

FIGURE 32 CPU usage for smart home applications with WL, BL, SSN, without WL, BL, SSN, and under BF attack as well as MITM.

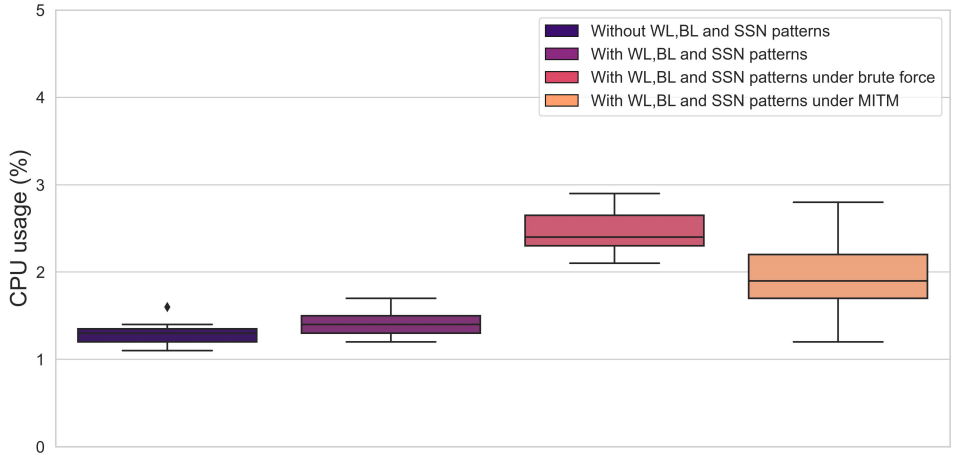

FIGURE 33 CPU usage for smart city applications with WL, BL, SSN, without WL, BL, SSN, and under BF attack as well as MITM.

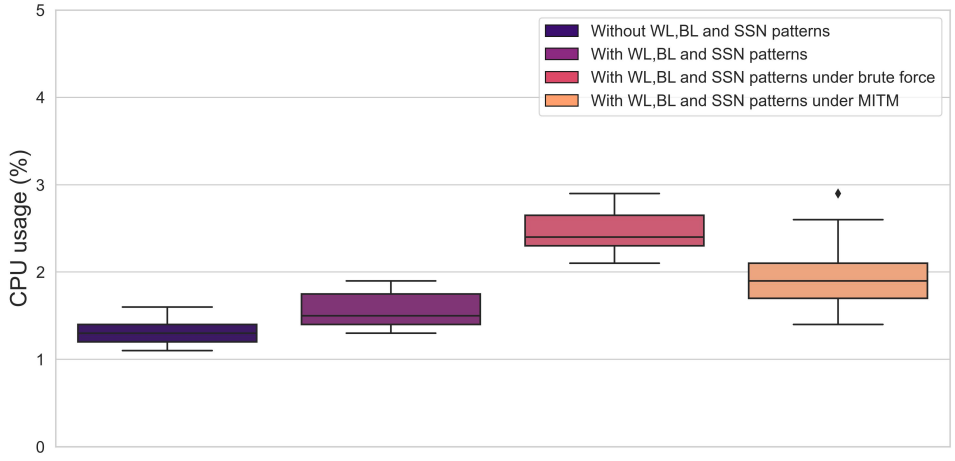

FIGURE 34 CPU usage for healthcare applications with WL, BL, SSN, without WL, BL, SSN, and under BF attack and MITM.

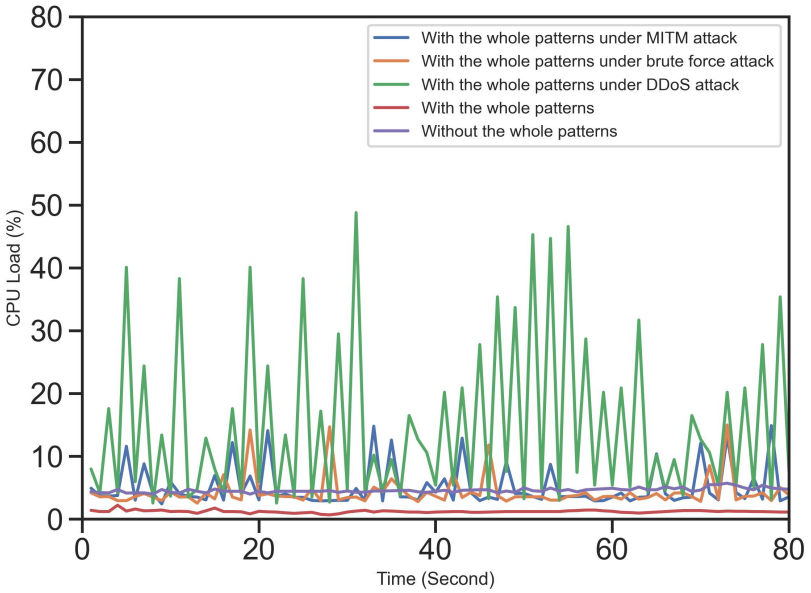

FIGURE 35 CPU load for smart home application with combination patterns, without combination patterns, and three attacks DDoS, MITM, and BF.

Finding 3:

The gathered experimental results show the efficacy of WL, BL, and SSN patterns in effectively mitigating BF and MITM attacks. Importantly, the incorporation of patterns during these attacks exhibits a discernible impact on CPU usage and energy consumption in all applications, with/without patterns and under attacks.

5.4 | Evaluating combination patterns

5.4.1 | RQ1: The impact of combination patterns on security

To address RQ1, we compared two applications, each with/without combined patterns. The aforementioned applications were subjected to a battery of attacks, including DDoS, MITM, and BF, which were carried out using Kali. These attacks were selected to ensure consistency with prior sections conducted in this area.

Results:

The findings of the study demonstrate that the CPU load of the examined systems exhibits a relatively low level in the absence of the combined patterns, measuring at a mere 2%. Nevertheless, the application of the comprehensive pattern leads to a substantial escalation in CPU load, reaching 5%. Moreover, the CPU load of applications under different types of attacks is further analyzed. Specifically, under the MITM and BF attacks, the CPU load of applications with the whole pattern increases to 15%. In contrast, the CPU load of applications under a DDoS attack with all patterns rises to a substantial 50%. These findings highlight the importance of considering the combined patterns in system design to ensure optimal performance and resilience against cyber threats.

The findings of our study indicate that the smart home, smart city, and healthcare are capable of managing various types of attacks. However, it is noteworthy that the application's performance is considerably hampered when it comes under a DDoS attack, causing a significant slowdown in its operations. The findings are presented in Fig. 35 , 36 , and 37 depicting the functioning of the smart home, city, and healthcare applications across three different situations. The analysis of the CPU load reveals that the entire patterns of usage, including normal state and attack scenarios, have a considerable impact on the application's performance.

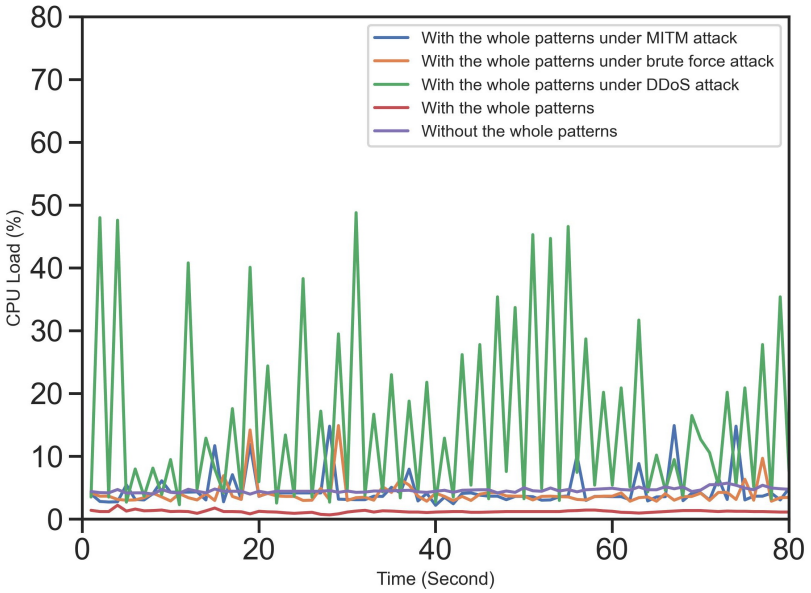

FIGURE 36 CPU load for a smart city application with combination patterns, without combination patterns, and three attacks DDoS, MITM, and BF.

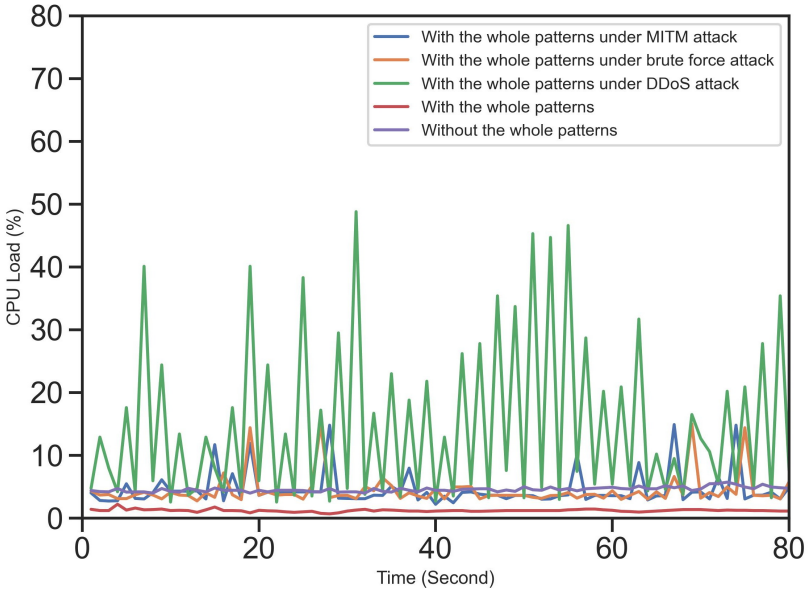

FIGURE 37 CPU load for a healthcare application with combination patterns, without combination patterns, and three attacks DDoS, MITM, and BF.

5.4.2 | RQ2: Impact on energy consumption and CPU usage

The following set of null hypotheses are defined to answer RQ2: H^x, $x \in \{1....8\}$. Hypotheses are defined as follows where P_0 corresponds to the version of the applications that do not use patterns, P_{all} indicates the scenario of applying all patterns, namely PZH, OOC, WL, BL, and SSN (see Table 2), $P_{all} - MITM$, $P_{all} - BF$ and $P_{all} - DDoS$ are indicate the occurrence or presence of MITM, brute-force, and DDoS attacks in P_{all}, respectively:

H^1_{all}: There is no difference between the average amount of energy consumed by P_{all} and P_0.

H^2_{all}: There is no difference between the average energy consumption of P_{all} and the $P_{all} - MITM$.

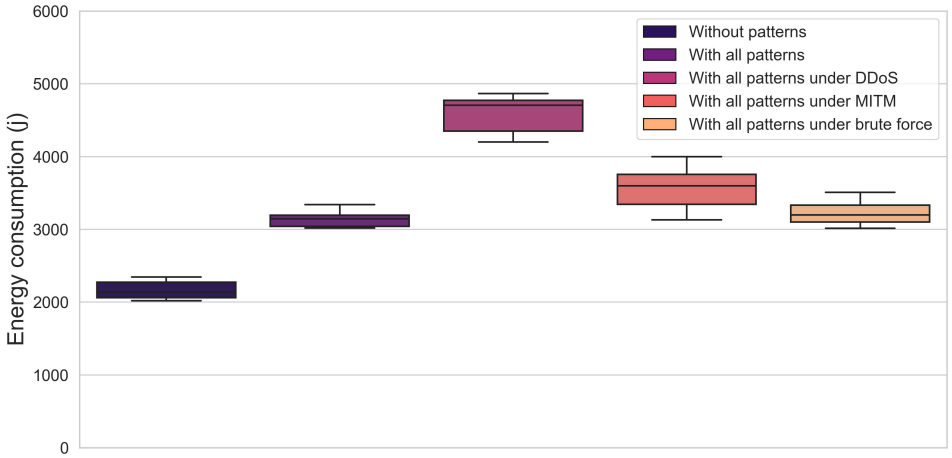

FIGURE 38 Energy consumption for smart home application with combination patterns, without combination patterns, and three attacks DDoS, MITM, and BF.

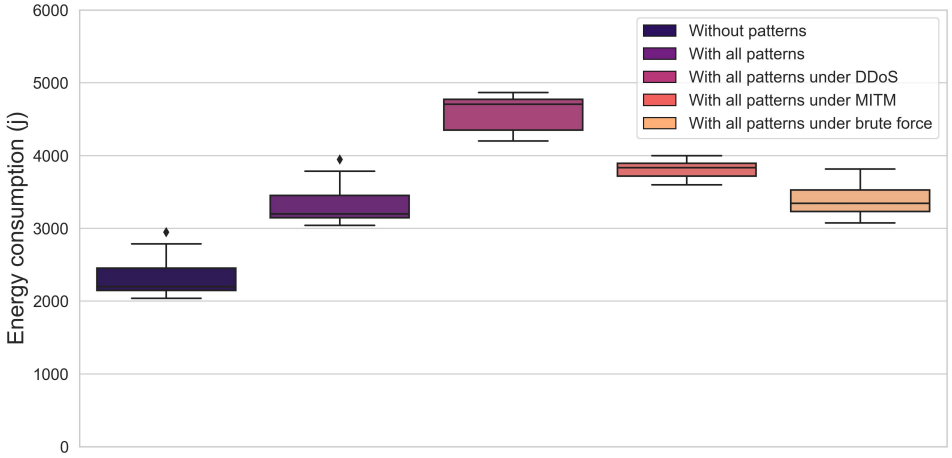

FIGURE 39 Energy consumption for a smart city application with combination patterns, without combination patterns, and three attacks DDoS, MITM, and BF.

H^3_{all}: There is no difference in the average energy consumed by P_{all} when compared to $P_{all} - BF$.
H^4_{all}: The average amount of energy consumed by P_{all} is not different from $P_{all} - DDoS$.
H^5_{all}: There is no difference between the average CPU usage by P_{all} and P_0.
H^6_{all}: There is no difference between the average CPU usage of P_{all} and $P_{all} - MITM$.
H^7_{all}: There is no difference in the average CPU usage by P_{all} when compared to $P_{all} - BF$.
H^8_{all}: The average amount of CPU usage in P_{all} is not different from $P_{all} - DDoS$.
Table 12 reports the P-values of the Mann-Whitney U test and Cliff's δ effect size for energy consumption evaluation. The table specifically compares the average energy consumption observed during three distinct attack scenarios (MITM, brute-force, and DDoS), with/without security patterns.
According to the P-value in Table 12 , we reject H^1_{all} in all applications; the analysis suggests that there is a trace of statistically significant differentiation in the average energy consumption in the domains of all applications.
According to the P-value for all applications under the MITM attack, we reject H^2_{all} for smart home and smart city applications. Upon statistical examination, a significant variation is found in the energy consumed by $P_{all} - MITM$ compared to P_{all}. However, we cannot reject H^6_{all} for healthcare application. The analysis indicates no statistically significant difference in energy

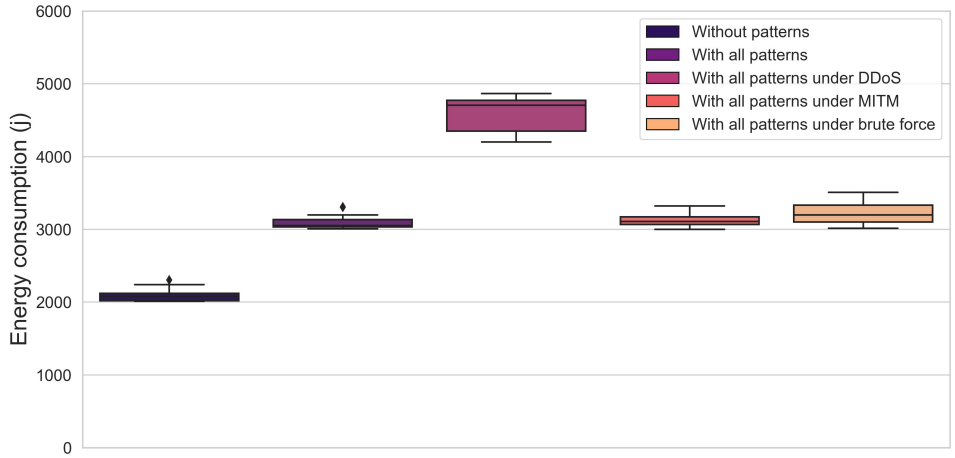

FIGURE 40 Energy consumption for a healthcare application with combination patterns, without combination patterns, and three attacks DDoS, MITM, and BF.

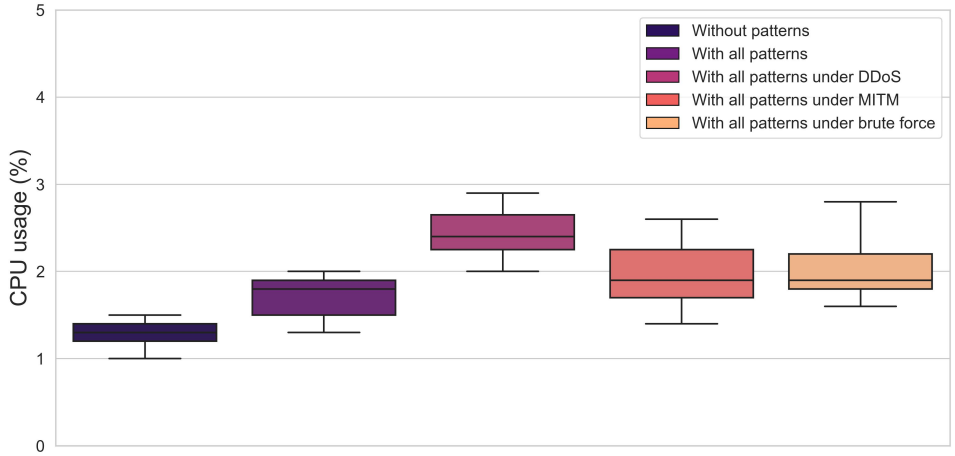

FIGURE 41 CPU usage for smart home application with combination patterns, without combination patterns, and three attacks DDoS, MITM, and BF.

consumed by $P_{all} - MITM$.
According to the P-value for all applications under the brute-force attack, we cannot reject H^3_{all} for all applications. The analysis uncovers no statistically significant effect on the energy consumed by $P_{all} - BF$.
According to the P-value for all applications under the DDoS attack, we reject H^4_{all} for all applications as a statistically significant difference is observed in the energy consumption of $P_{all} - DDoS$ for this application.
Fig. 38 , 39 , and 40 show the outcomes achieved across various implementations of the combined patterns. The examination of energy consumption during DDoS, brute-force, and MITM attacks reveals an increase when all patterns are introduced. Nevertheless, it is imperative to underscore that a statistically significant variance in energy consumption exists between scenarios with/without all patterns. Furthermore, there is a statistically significant variance in energy consumption under brute-force attacks in the smart home and in smart home and healthcare during MITM attack as well as all patterns during DDoS attack.

Table 13 reports the P-values of the Mann-Whitney U test and the Cliff's δ effect size for CPU usage evaluation. The table specifically compares the average CPU usage levels observed during three distinct attack scenarios (MITM, brute-force, and DDoS) with/without security patterns.
According to the P-value in Table 13 , we reject H^5_{all} for all applications; as there is a statistically significant difference between

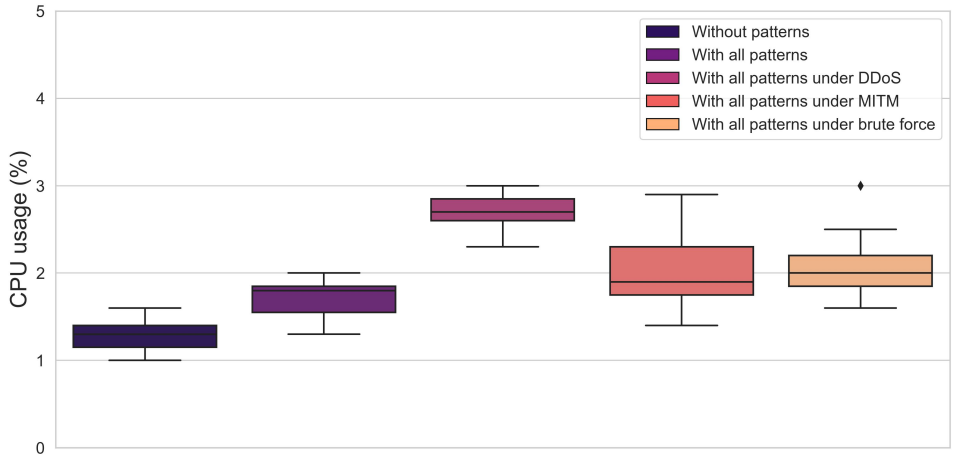

FIGURE 42 CPU usage for a smart city application with combination patterns, without combination patterns, and three attacks DDoS, MITM, and BF.

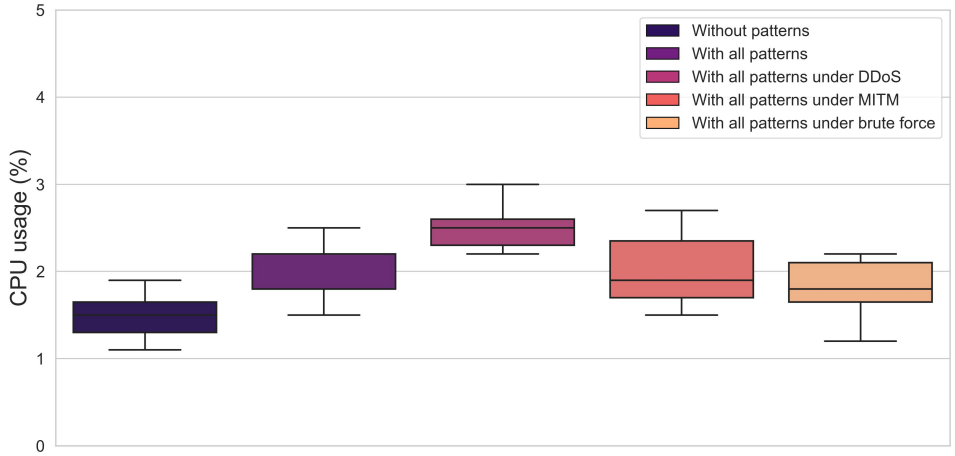

FIGURE 43 CPU usage for a healthcare application with combination patterns, without combination patterns, and three attacks DDoS, MITM, and BF.

the average amount of CPU usage under different conditions with/without a combination of patterns in three applications.
According to the P-value for all applications under the MITM attack, we cannot reject H^6_{all} for all applications. The analysis indicates no statistically significant difference in CPU usage by $P_{all} - MITM$.
According to the P-value for all applications under the brute-force attack, we cannot reject H^7_{all} for smart home and healthcare applications. The data reveals no significant disparity in CPU usage by $P_{all} - BF$. Moreover, we also reject H^7_{all} for smart city application.The data reveals a significant disparity in CPU usage by $P_{all} - BF$.
For the case of the DDoS attack, we reject H^8_{all} for the smart city and healthcare applications. A careful analysis leads us to affirm that statistically significant variation is found in the CPU usage by $P_{all} - DDoS$. However, we cannot reject H^8_{all} for the smart home application as a statistically significant variation is not found in the CPU usage by $P_{all} - DDoS$.
Fig. 41 , 42 , and 43 illustrate the results obtained for all implementations of the combined patterns. The analysis of CPU usage during DDoS, brute-force, and MITM attacks demonstrates an increase when all patterns are introduced. However, it is essential to emphasize that there is a statistically significant variance in CPU usage between scenarios with and without all patterns. Moreover, there is no statistically significant variance in CPU usage during the brute-force attack in the smart city and during the DDoS attack in the smart city and the healthcare.

TABLE 12 P-value of Mann–Whitney U test and Cliff's δ effect size for the average energy consumption under MITM, BF, and DDoS attacks.

Version	Avg. Energy Consumption					
	SmartHome		SmartCity		HealthCare	
	P-Value	Effect Size	P-Value	Effect Size	P-Value	Effect Size
P_0 vs. P_{all}	0.0001	1.0	0.0001	1.0	0.0001	1.0
P_{all} vs. $P_{all} - MITM$	0.0001	0.893	0.0001	1.0	0.3953	1.0
P_{all} vs. $P_{all} - BF$	0.1096	0.347	0.3173	0.210	0.1428	0.529
P_{all} vs. $P_{all} - DDoS$	0.0001	1.0	0.0001	0.1	0.0001	1.0

TABLE 13 P-value of Mann–Whitney U test and Cliff's δ effect size for the average CPU usage under MITM, BF, and DDoS.

Version	Avg. CPU Usage					
	SmartHome		SmartCity		HealthCare	
	P-Value	Effect Size	P-Value	Effect Size	P-Value	Effect Size
P_0 vs. P_{all}	0.0012	0.813	0.0001	0.813	0.0002	0.804
P_{all} vs. $P_{all} - MITM$	0.8014	0.369	0.3400	0.458	0.1770	0.031
P_{all} vs. $P_{all} - BF$	0.3400	0.458	0.0114	0.547	0.9044	0.293
P_{all} vs. $P_{all} - DDoS$	0.3400	0.982	0.0001	1.0	0.0002	0.804

Finding 4:

Our results indicate the efficacy of all patterns (when combined) in effectively mitigating BF, MITM, and DDoS attacks. Importantly, we observe a discernible impact on CPU usage and energy consumption, both in the attack scenarios and with and without all patterns with statistically significant differences.

6 | DISCUSSION

The intersection of security, energy consumption, and CPU usage poses a multifaceted trade-off for developers and practitioners striving to select an appropriate IoT security pattern. Thus, prudent caution is essential when choosing patterns for IoT applications.

An optimal IoT system should have the ability to proactively predict risky situations and intelligently use different security patterns based on the context to keep the system safe energy and CPU usage within limits. Unforeseen circumstances, such as abrupt mode shifts or environmental uncertainties, might lead to indifferent behavior in a selected security pattern. Hence, the system's adaptability is crucial, necessitating the integration of experiential learning from such occurrences to continually refine the decision-making processes for enhanced efficiency. Activating all security patterns over the application's lifespan impacts performance, energy consumption, and CPU usage, so a more judicious approach involves tailoring security patterns to specific contextual scenarios. Furthermore, the challenge of identifying the most appropriate security pattern can be analogously likened to the renowned "Robot in the Grid World" problem[66]. In this conceptualization, the software architecture assumes the role of a navigating robot within a grid. The overarching objective involves determining an optimal grid position that maximizes system efficiency concerning security, CPU, and energy thresholds. To illustrate, consider a scenario featuring three distinct security patterns aimed at countering DDoS attacks, each characterized by unique advantages and drawbacks, including resource consumption implications. Consequently, the IoT application's dynamic selection of an apt security pattern becomes imperative, as it strategically responds to prevailing conditions – encompassing factors like IoT application resource availability – to avert CPU efficiency and energy consumption losses.

Our research underscores the importance of prudent pattern selection in IoT applications due to the intricate interplay between

security, energy consumption, and CPU usage. The experimental findings provide crucial insights for developers and practitioners to make informed decisions when implementing IoT security patterns. For comprehensive details, we refer the reader to Table 14 , which outlines the effects of the six IoT security patterns on security, energy efficiency, and CPU usage.

Security: The deployment of applications at the edge of IoT has given rise to several security concerns and risks, primarily due to the lack of established security protocols, insufficient authentication and authorization mechanisms, and data privacy and confidentiality issues. The nature of IoT-edge-based applications necessitates communication between multiple devices, and with the cloud, a high probability of security breaches exists. The limited computational power and memory of many devices at the edge of the IoT pose a significant challenge in deploying robust security measures, thereby rendering them vulnerable to attacks such as DDoS. Despite the availability of IoT security patterns, which aim to secure applications and resources and can handle most critical attacks in IoT-edge-based applications, novel security patterns with advanced effectiveness are necessary.

Energy consumption and CPU usage: The IoT ecosystem includes edge devices that are typically small, compact and possess limited resources, such as batteries and processors. These limitations, in turn, lead to a restricted capacity for energy and computing power delivery. In addition, such devices are designed to be cost-effective and accessible, which often necessitates lower-end hardware components, further limiting their energy consumption and computational capabilities. It becomes crucial to consider and optimize the trade-offs between functionality and cost while designing IoT edge devices, taking into account the available resources.

Our experiments indicate that the employing of security patterns has an impact on energy consumption and CPU usage. Hence, IoT developer architects should refer to the guidelines provided in Table 15 , which includes the six patterns studied on IoT security, to choose suitable patterns during application development.

7 | THREATS TO VALIDITY

Empirical research inevitably encounters issues related to the validity of findings. In light of this, the present section seeks to identify and discuss possible threats to our research's validity, per the recommendations of[67].

7.1 | Internal

Our empirical study is vulnerable to various internal threats. We specifically examine one implementation of the analyzed security patterns in three IoT applications. The particularity of this could impact the applicability of our findings, since diverse implementations may produce various results in terms of security, energy consumption, and CPU usage. We detailed our implementation in Section 4. Another threat is the precision of our measurement of CPU usage and the efficacy of security patterns stand out. The real-time behavior of IoT components, such as the Raspberry Pis, often diverges from their theoretical models, which makes it challenging to ensure accuracy in performance measurement. To minimize the potential influence of network or hardware perturbations and environmental interference on our tracing, we conducted each experiment fifteen times and computed average values. This approach allowed us to mitigate potential biases and enhance the sustainability of our results. The security algorithms we selected for our study can also suffer from some internal threats like configuration and accuracy. For different attack scenarios, the security algorithms and patterns may behave differently in terms of attack detection and prevention accuracy. To minimize these threats, we adapted the standard configuration of the algorithms used in different academic and industry publications. At the same time, we adapted the attacks with the standard configurations used in different research. Thus, our present study is bias-free for any of the internal threats related to encryption algorithms and security breaching attacks.

7.2 | External

In the selection of subject systems and analysis methods, it is crucial to address potential threats to the validity and reliability of research findings. In this study, we have mitigated such threats by choosing three IoT-edge-based applications that are widely adopted and referenced in the related literature. However, it should be noted that the findings of our analysis may be influenced by various factors such as hardware, sensors, and environmental conditions, including power fluctuations. In addition, we have also implemented commonly used IoT security patterns and cryptography algorithms to enhance the credibility of our results. Additionally, to ensure the transparency and reproducibility of our study, we have provided detailed information about the

TABLE 14 Impact of security patterns on security, energy efficiency, and CPU usage.

Context Problem	Pattern	Security	Energy	CPU usage
Security, No Open Ports, Firewalls, Low Energy	OOC	Improved	increased	increased
Centralized Access Control, User Control, Trust	PZH	Not effective	increased	increased
Explicit Allowance, Flexibility, Trust, Simplicity, Completeness	WL	Improved	increased	increased
Flexibility, Explicit Blocking, Simplicity, Outdated Entries	BL	Improved	increased	increased
Secure Communication, Access Control	SSN	Improved	increased	increased

* The TCP pattern is not mentioned because this pattern is generally used along with WL and BL patterns, and we did not analyze the impact of an isolated implementation of this pattern.

TABLE 15 Guideline for selecting the six patterns.

Applications most important non-functional requirement	OOC	PZH	WL	BL	SSN
Security	😊	☹️	😊	😊	😊
Energy efficiency	😐	😐	😐	😐	😐
CPU usage	😐	😐	😐	😐	😐

experimental setup and made our testbed and results publicly available for further research[64]. By adopting these measures, we have attempted to provide robust validation and increase the inability to reject our findings among practitioners and researchers.

8 | CONCLUSIONS

This research aims to investigate the impact of six IoT security patterns, namely Personal Zone Hub, Trusted Communication Partner, Outbound-Only Connection, Blacklist, Whitelist, and Secure Sensor Node, on energy consumption, CPU usage, and CPU load and provide guidance to developers in selecting and using them. We implemented three IoT-edge-based applications, i.e., smart home, smart city, and healthcare. We applied the patterns in isolation and also combined them in all applications. Each application with/without patterns and under different attacks was examined. The results indicate that while employing these patterns improves security, usually, there is a jump in energy efficiency and CPU usage of the IoT-edge-based application when examined, which is statistically significant in many cases. However, the scalability and response time of these patterns is still unknown. Additionally, our study highlights the need for more comprehensive security patterns covering all IoT security aspects.

8.1 | Bibliography

References

1. Laghari AA, Wu K, Laghari RA, Ali M, Khan AA. A review and state of art of Internet of Things (IoT). *Archives of Computational Methods in Engineering* 2021: 1–19.
2. Mohanta BK, Jena D, Satapathy U, Patnaik S. Survey on IoT security: Challenges and solution using machine learning, artificial intelligence, and blockchain technology. *Internet of Things* 2020; 11: 100227.
3. Schiller E, Aidoo A, Fuhrer J, Stahl J, Ziörjen M, Stiller B. Landscape of IoT security. *Computer Science Review* 2022; 44: 100467.
4. Fernandez EB, Washizaki H, Yoshioka N, Okubo T. The design of secure IoT applications using patterns: State of the art and directions for research. *Internet of Things* 2021; 15: 100408.
5. Reinfurt L, Breitenbücher U, Falkenthal M, Fremantle P, Leymann F. Internet of Things security patterns. In: ; 2017: 20.
6. Orellana C, Fernandez EB, Astudillo H. A pattern for a Secure Sensor Node. In: ; 2020.
7. Allen L, Heriyanto T, Ali S. *Kali Linux, Assuring security by penetration testing*. Packt Publishing Ltd . 2014.
8. Ven v. dA. Powertop. https://github.com/fenrus75/powertop; .
9. Syed MH, Fernandez EB, Ilyas M. A pattern for Fog computing. In: ; 2016: 1–10.
10. Seitz A, Thiele F, Bruegge B. Fogxy: An architectural pattern for Fog computing. In: ; 2018: 1–8.
11. Fernandez EB. A pattern for a secure cloud-based IoT architecture. In: ; 2020.
12. Pahl C, El Ioini N, Helmer S, Lee B. An architecture pattern for trusted orchestration in IoT edge clouds. In: IEEE. ; 2018: 63–70.

13. Fysarakis K, George S, Petroulakis N, Soultatos O, Bröring A, Marktscheffel T. Architectural patterns for secure IoT orchestrations. In: IEEE. ; 2019: 1–6.

14. Lee WT, Law PJ. A case study in applying security design patterns for IoT software system. In: IEEE. ; 2017: 1162–1165.

15. Fernandez EB, Yoshioka N, Washizaki H. Secure distributed publish/subscribe (P/S) pattern for IoT. *Procs. of AsianPLoP* 2020.

16. Syed MH, Fernandez EB, Moreno J. A misuse Pattern for DDoS in the IoT. In: ; 2018: 1–5.

17. Fernandez EB, Romero VM. A security reference architecture for cargo ports. *Internet of Things and Cyber-Physical Systems* 2022; 2: 120–137.

18. Papa R, Fernandez EB, Cardei M. A pattern for a UAV-aided wireless sensor network. In: ; 2019: 1–9.

19. Pape S, Rannenberg K. Applying privacy patterns to the Internet of Things(IoT) architecture. *Mobile Networks and Applications* 2019; 24(3): 925–933.

20. Mahamat M, Jaber G, Bouabdallah A. Achieving efficient energy-aware security in IoT networks: a survey of recent solutions and research challenges. *Wireless Networks* 2023; 29(2): 787–808.

21. Prakasam P, Madheswaran M, Sujith K, Sayeed MS. An enhanced energy efficient lightweight cryptography method for various IoT devices. *ICT Express* 2021; 7(4): 487–492.

22. Yeh LY, Chen PJ, Pai CC, Liu TT. An energy-efficient dual-field elliptic curve cryptography processor for Internet of Things applications. *IEEE Transactions on Circuits and Systems II: Express Briefs* 2020; 67(9): 1614–1618.

23. Yazdinejad A, Parizi RM, Dehghantanha A, Zhang Q, Choo KKR. An energy-efficient SDN controller architecture for IoT networks with blockchain-based security. *IEEE Transactions on Services Computing* 2020; 13(4): 625–638.

24. Tekin N, Acar A, Aris A, Uluagac AS, Gungor VC. Energy consumption of on-device machine learning models for IoT intrusion detection. *Internet of Things* 2023; 21: 100670.

25. Harris K. 7 challenges in IoT and how to overcome them. https://www.hologram.io/blog/challenges-in-iot/; 2021.

26. D'mello A. 5 challenges still facing the Internet of Things (IoT). https://www.iot-now.com/2020/06/03/103228-5-challenges-still-facing-the-internet-of-things/; 2020.

27. Bhoyar P, Sahare P, Dhok SB, Deshmukh RB. Communication technologies and security challenges for the Internet of Things(IoT): A comprehensive review. *AEU-International Journal of Electronics and Communications* 2019; 99: 81–99.

28. Kimani K, Oduol V, Langat K. Cyber security challenges for IoT-based smart grid networks. *International Journal of Critical Infrastructure Protection* 2019; 25: 36–49.

29. Tawalbeh L, Muheidat F, Tawalbeh M, Quwaider M. IoT Privacy and security: Challenges and solutions. *Applied Sciences* 2020; 10(12): 4102.

30. Hameed S, Khan FI, Hameed B. Understanding security requirements and challenges in Internet of Things (IoT): A review. *Journal of Computer Networks and Communications* 2019; 2019.

31. Zikria YB, Kim SW, Hahm O, Afzal MK, Aalsalem MY. Internet of Things (IoT) operating systems management: Opportunities, challenges, and solution. *Sensors* 2019; 19(8): 1793.

32. D'mello A. Top 4 Challenges in IoT Data Collection and Management. https://www.firstpoint-mg.com/blog/top-4-challenges-in-IoT-data-collection-and-management/; 2021.

33. Patchava V, Kandala HB, Babu PR. A smart home automation technique with Raspberry pi using IoT. In: IEEE. ; 2015: 1–4.

34. Toma C, Alexandru A, Popa M, Zamfiroiu A. IoT solution for smart cities' pollution monitoring and the security challenges. *Sensors* 2019; 19(15): 3401.

35. Rahman A, Rahman T, Ghani NH, Hossain S, Uddin J. IoT-based patient monitoring system using ECG sensor. In: IEEE. ; 2019: 378–382.

36. Hasan D, Ismaeel A. Designing ECG monitoring healthcare system based on Internet of Things(IoT) blink application. *Journal of applied science and technology trends* 2020; 1(3): 106–111.

37. Flask . https://palletsprojects.com/p/flask/; 2023.

38. Nginx . NGINX: Advanced Load Balancer, Web Server, & Reverse Proxy. https://www.nginx.com/; 2023.

39. Chart G. https://developers.google.com/chart/; 2023.

40. Plotly . https://plotly.com/python/; 2023.

41. Guchu MW. *A Representational state transfer web tool for firewall service management and monitoring in a Local Area Network*. PhD thesis. Strathmore University, ; 2020.

42. Lyon GF. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure . 2009.

43. Coonjah I, Catherine PC, Soyjaudah K. Experimental performance comparison between TCP vs UDP tunnel using OpenVPN. In: IEEE. ; 2015: 1–5.

44. Rastogi V, Shao R, Chen Y, Pan X, Zou S, Riley RD. Are these Ads Safe: Detecting Hidden Attacks through the Mobile App-Web Interfaces.. In: ; 2016.

45. Mufid MR, Basofi A, Al Rasyid MUH, Rochimansyah IF, others . Design an mvc model using python for flask framework development. In: IEEE. ; 2019: 214–219.

46. Yarom Y, Genkin D, Heninger N. CacheBleed: a timing attack on OpenSSL constant-time RSA. *Journal of Cryptographic Engineering* 2017; 7: 99–112.

47. Sreesailam VB, Pentakota DG, Pappala T, Kopanati S, Siripurapu CP. A Novel Methodology Proposed To Produce A Secure Password. *Journal of Pharmaceutical Negative Results* 2022: 5142–5150.

48. Takahashi A, Tibouchi M. Degenerate fault attacks on elliptic curve parameters in OpenSSL. In: IEEE. ; 2019: 371–386.

49. Mezher AE. Enhanced RSA cryptosystem based on a multiplicity of public and private keys. *International Journal of Electrical and Computer Engineering* 2018; 8(5): 3949.

50. Khomh F, Abtahizadeh SA. Understanding the impact of cloud patterns on performance and energy consumption. *Journal of Systems and Software* 2018; 141: 151–170.

51. Sheskin DJ. *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC . 2003.

52. Dmitrienko A, Molenberghs G, Chuang-Stein C, Offen W. Analysis of Clinical Trials Using SAS: A Practical Guide. SAS Institute, 2005. *http://www.google.ca/books*.

53. Doshi K, Yilmaz Y, Uludag S. Timely detection and mitigation of stealthy DDoS attacks via IoT networks. *IEEE Transactions on Dependable and Secure Computing* 2021; 18(5): 2164–2176.

54. Vishwakarma R, Jain AK. A survey of DDoS attacking techniques and defence mechanisms in the IoT network. *Telecommunication systems* 2020; 73(1): 3–25.

55. Kumari P, Jain AK. A Comprehensive Study of DDoS Attacks over IoT Networks and Their Countermeasures. *Computers & Security* 2023: 103096.

56. Kumar P, Bagga H, Netam BS, Uduthalapally V. Sad-IoT: Security analysis of DDoS attacks in IoT networks. *Wireless Personal Communications* 2022; 122(1): 87–108.

57. Salem O, Alsubhi K, Shaafi A, Gheryani M, Mehaoua A, Boutaba R. Man-in-the-Middle attack mitigation in the Internet of Medical Things. *IEEE Transactions on Industrial Informatics* 2021; 18(3): 2053–2062.

58. Thomas J, Cherian S, Chandran S, Pavithran V. Man in the middle attack mitigation in LoRaWAN. In: IEEE. ; 2020: 353–358.

59. Saputro ED, Purwanto Y, Ruriawan MF. Medium interaction honeypot infrastructure on the Internet of Things(IoT). In: IEEE. ; 2021: 98–102.

60. Stiawan D, Idris M, Malik RF, et al. Investigating Brute force attack patterns in IoT network. *Journal of Electrical and Computer Engineering* 2019; 2019.

61. Salvati M. SSH-MITM. https://github.com/byt3bl33d3r/MITMf; 2018.

62. Reinfurt L, Breitenbücher U, Falkenthal M, Leymann F, Riegg A. Internet of Things(IoT) patterns. In: ; 2016: 1–21.

63. SecLists . Brute-force. https://github.com/danielmiessler/SecLists/tree/master/Passwords; 2022.

64. Jamshidi S. Data of experiments. https://github.com/saeidjam/DataSecurityPattern; 2023.

65. Abdullah AM, others . Advanced encryption standard (AES) algorithm to encrypt and decrypt data. *Cryptography and Network Security* 2017; 16: 1–11.

66. Tokic M, Ammar HB. Teaching reinforcement learning using a physical robot. In: ; 2012.

67. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. *Experimentation in software engineering*. Springer Science & Business Media . 2012.

How to cite this article: Williams K., B. Hoskins, R. Lee, G. Masato, and T. Woollings (2016), A regime analysis of Atlantic winter jet variability applied to evaluate HadGEM3-GC2, *Q.J.R. Meteorol. Soc.*, *2017;00:1–6*.