Monitoring Vulnerabilities in IoT Environments

Ana Vidal
Robust Software
MSc. Cibersecurity
Aveiro, Portugal
rvidal@ua.pt

Fábio Santos
Robust Software
MSc. Cibersecurity
Aveiro, Portugal
falexandre@ua.pt

Rodrigo Marques
Robust Software
MSc. Cibersecurity
Aveiro, Portugal
rjmm@ua.pt

Simão Andrade
Software Robusto
MSc. Cibersecurity
Viseu, Portugal
simao.andrade@ua.pt

***Abstract*— This report addresses the design of an innovative system for monitoring and analysing information from IoT (Internet of Things) devices. This system aims to address crucial reliability, robustness, and security concerns in the context of IoT.**

Our project distinguishes itself from conventional IoT applications by offering a comprehensive solution that aggregates data from devices of different brands, all communicating via the same protocols, if all companies in this industry have a standard protocol and not several protocols. What sets it apart is its prioritized emphasis on security and preserving the privacy of end users. While other IoT applications often neglect security issues, our system will identify and mitigate possible vulnerabilities, guaranteeing a reliable and secure experience for users.

By analysing the data coming from these devices in detail, the system will provide clear information to users, indicating what information is being extracted from each device and identifying potential security threats. With this focus on security and transparency, we offer a robust solution that fills a critical gap in the IoT market, prioritizing reliability and user privacy.

Keywords—security, confidentiality, quality

I. Introduction

The Internet of Things (IoT) universe has witnessed exponential growth in recent decades, with almost 16 billion IoT devices in operation, ranging from smart home devices to critical industrial systems. However, this dizzying expansion is not without significant challenges. This report addresses the pressing need to solve problems inherent in this scenario, such as the lack of adequate security measures on the part of manufacturers, the proliferation of vulnerable devices and the technological illiteracy common among the average users.

One of the most pressing problems is the lack of robust security measures implemented by IoT device manufacturers. Many of these devices arrive on the market without proper security safeguards, making them easy targets for malicious intrusions. These vulnerabilities can result in the disclosure of users' private information and the use of these devices as part of botnets to carry out denial-of-service (DoS) attacks, jeopardizing the stability of digital infrastructure.

It is imperative to address these challenges to ensure a secure and reliable environment for the growing network of IoT devices. In this report, we will present an innovative solution that focuses on several crucial aspects, including secure development, solution quality, confidentiality, integrity, non-repudiation, accountability, and authenticity. This solution aims not only to mitigate existing vulnerabilities, but also to promote a culture of security across the entire IoT ecosystem, providing end users with a safer and smoother experience. By tackling these challenges head on, we are committed to shaping a future where IoT is synonymous with reliability, robustness, and security.

II. Secure Life Cycle Description

This section presents the secure development life cycle that will be applied to our solution to ensure that the security requirements are met.

A. Education and Awareness - Provide Training (Ana Vidal)

Security is everyone's job. It is important that everyone in the company, regardless of their position, has the basics to apply appropriate security knowledge on the tasks they are entitled to perform within the project. The effective training provides the team ways to know how to build security around a software, so this software meets the quality and security standards, addressing business needs and delivering use value. Although not everyone is a security expert, it is necessary to understand the attacker's perspective and what are his goals to take actions that prevent those attacks.

This training could be provided by speeches, workshops or tests that focus on concepts related to the security scope like:

- Non-functional Requirements: privacy, security, safety, recoverability, etc.

- Good Practices: Use of cryptography, use of authentication, prevent SQL Injections, etc.

- Vulnerabilities and Risk Analysis: Top 25 CWE (Common Weakness Enumeration), input sanitizing and validation, etc.

Security and safety are two of the most important aspects of a Software, especially on a company that provides services of IoT devices. The damage that the lack of security could provide to our systems is outrageous, since lots of important data is stored on it, like information about closed/opened doors or windows or video from a security camera installed on a home.

No company likes to have its data exposed, much less in a world where the media disseminates information to billions of people, hence the importance to reinforce the learning of skills related to security. In the business context, situations like this can make the costumers lose their trust in a brand or collaborators want to distance themselves from our company. Therefore, it is important to provide this type of training to everyone in the team, whether they are developers, engineers, or product managers.

B. Project Inception - Define and Use Cryptography standards (Simão Andrade)

The use of cryptography is one of the key parameters in the implementation of safe and secure software. This makes it possible to manipulate the data stored or sent, to protect it from possible attacks such as man-in-the-middle or data leaks. With an increasing number of mobile devices and cloud computing, it is important to ensure that all data is secure, as it may contain sensitive information.

No homeowner with IoT devices would want to have their privacy invaded by an attacker. Therefore, by accessing the software that underpins a Smart Home, an attacker can control the washing machine, open, or close a garage door or even view cameras in real time. It is important to use encryption to protect all user data, in compliance with the General Data Protection Regulation (GRPD).

To protect data, it is important to use an algorithm that has never been broken and has few collisions. A poor choice of algorithm can lead to catastrophic damage, thus the importance of developing clear encryption standards. To properly implement data encryption in software, it is recommended that you use industry-approved libraries and always being aware of possible CVEs (Common Vulnerabilities and Exposures) that these libraries may have. That said, the use of RSA will be the most appropriate algorithm for encrypting the data sent by the application between the IoT devices and the disk.

C. Analysis and Requirements – Define Security Requirements (Fábio Santos)

Security requirements in software projects are the rules and guidelines that ensure your software is protected from potential threats and vulnerabilities. To define those set of rules, it’s imperative that we firstly identify the objectives and scope of the application. This involves determining the primary goal of the application, which is to monitor vulnerabilities in IoT environments. Additionally, it's important to specify the scope of the application, which includes defining the types of IoT devices and systems that will be supported.

Once the objectives and scope are established, the next step is to identify potential threats and vulnerabilities that the application must address. These threats and vulnerabilities can encompass a wide range of issues, such as network attacks, exploitation of IoT device vulnerabilities, data leakage, and more.

Authentication and authorization requirements must also be clearly defined. This involves establishing robust authentication mechanisms for users and devices, as well as implementing authorization policies to control access to the application and the data it monitors.

Data encryption is a crucial component of security, requiring that data be encrypted both in transit and at rest to safeguard information during transmission and storage.

Intrusion and anomaly detection systems shall be implemented to identify suspicious behaviour within the application and on monitored devices.

To ensure ongoing security, it's essential to enable secure updates for the application, which allows for the correction of known vulnerabilities. Continuous monitoring features shall be developed to track the integrity of the application and the IoT devices it monitors.

Measures to protect the application against Distributed Denial of Service (DDoS) attacks shall be implemented to prevent interruptions in its operation.

An incident response plan is necessary to address security events, such as the detection of critical vulnerabilities or intrusions.

Detailed logs of all application activities, including logins, user actions, and security events, shall be maintained for audit and record-keeping purposes.

Regular security tests, including penetration tests and vulnerability assessments, shall be planned to proactively identify and address security weaknesses.

Comprehensive documentation for administrators and users on security practices is essential, and training shall be conducted to ensure that staff members understand and adhere to security procedures.

Compliance with relevant data security and privacy regulations, such as GDPR, Health Insurance Portability and Accountability Act (HIPAA), or other regional regulations, is imperative.

Periodic risk assessments shall be performed to identify new threats and ensure the ongoing security of the application.

Finally, it's crucial to maintain a mindset of continuous improvement, being prepared to update the application and its security requirements as threats evolve and new technologies emerge.

D. Architetural and Detailed Design – Establish Design Requiremtens (Rodrigo Marques)

Design requirements in software projects serve as the blueprint for the entire system, ensuring security and functionality to it, this shall derive from the requirements previously mentioned.

One of the critical aspects in this phase is the development of a user-friendly interface. The user interface is the gateway through which individuals interact with smart appliances, and it needs to be intuitive and easy to navigate. This user-

friendly design allows users to effortlessly control and monitor their smart appliances, with clear and simple navigation to access different appliances and their respective functions.

Device compatibility is another key consideration, especially in an era where various devices and platforms are prevalent. Ensuring that the application has a responsive design and is compatible with a wide range of devices, particularly for mobile platforms like iOS and Android, is imperative. This compatibility ensures that users can access and manage their smart appliances regardless of the device they choose.

Security is of paramount importance in this phase. Robust authentication and authorization mechanisms are established to protect user accounts and the system. Strong authentication methods, including email and password, biometrics, and two-factor authentication, are implemented to fortify the security of the application.

The ability for users to control their appliances remotely is a crucial feature. Real-time control from anywhere provides convenience and flexibility to users, allowing them to turn appliances on/off and adjust settings, even when they are not at home.

Moreover, the emphasis on energy efficiency is integrated into the design. Automatic schedules are created to optimize energy consumption, reducing not only costs but also environmental impact.

To enhance the user experience and expand functionality, third-party integration is seamlessly woven into the design. This integration allows for the addition of more appliances and new features, making the application versatile and attractive to manufacturers.

Scalability is also addressed during this phase. The app is designed to handle a growing number of devices and users reliably, with minimal downtime. This is essential in ensuring that the application can accommodate the expanding user base and the ever-increasing array of smart appliances.

E. Implementation and Design - Perform penetration testing (Ana Vidal)

This section shall ensure that the security requirements and architecture stated in points C and D are properly implemented and completed. This can be done by performing a PenTest.

- Perform Dynamic Analysis Security Testing (DAST).
    - Run-time verification of compiled or packaged software to check functionality that is only apparent when all components are integrated and running.
    - Use of a suite of pre-built attacks and malformed strings that can detect memory corruption, user privilege issues, injection attacks, and other critical security problems.
    - May employ fuzzing, an automated technique of inputting known invalid and unexpected test cases at an application, often in large volume.
    - Like SAST, can be run by the developer and/or integrated into the build and deployment pipeline as a check-in gate.
- Another resource for structuring penetration tests is the OWASP Top 10 Most Critical Web Application Security Risks.
- Penetration testing is black box testing of a running system to simulate the actions of an attacker.
- To be performed by skilled security professionals, internal or external to the organization, opportunistically simulating the actions of a hacker.
- The objective is to uncover any form of vulnerability.
    - From small implementation bugs to major design flaws resulting from coding errors, system configuration faults, design flaws or other operational deployment weaknesses.

F. Release, Deployment and Support – Establish a Standard Incident Response Process (Simão Andrade)

A key component of this process is the preparation of an Incident Response Plan. The IRP is a comprehensive document that addresses how the organization will deal with security incidents, from identification to resolution and communication with stakeholders.

1. Emergency Contacts: This shall define who will be contacted in the event of a security emergency. This could include members of the internal security team, regulatory authorities, cybersecurity vendors or external consultants.
2. Vulnerability Mitigation Protocol: The plan shall establish clear procedures for dealing with vulnerabilities and threats. This may involve imposing corrections or patches on affected systems.
3. Response and Communication with Customers: It shall describe how the organization will respond to affected customers in the event of a data breach or privacy-related incidents. Transparent communication is essential to maintain customer trust.
4. Rapid Implementation of Solutions: The plan shall include a strategy for implementing security solutions quickly. This can include applying security updates or deploying new protection measures.
5. Management of Inherited Code: If the organization uses third-party code or inherits code from other internal parties, the plan shall address how these components will be monitored and updated for known vulnerabilities.
6. Third Party Code: For code provided by third parties, the plan shall stipulate how the organization will ensure that this code is secure and free of vulnerabilities. This can include security checks and contractual requirements for suppliers.
7. Testing the Plan: A critical part of the process is to test the IRP before it is needed. This involves simulating security incidents to ensure that the team is prepared to respond effectively.

III. Security Requirements

This section outlines our secure development life cycle, ensuring adherence to stringent security requirements. By integrating best practices and advanced security measures, our goal is to fortify the solution against potential threats and vulnerabilities, reflecting our commitment to delivering a product that surpasses industry standards in information security.

TABLE I
Requirements Catalogue

This table provides a structured overview of the essential criteria that shape our solution. Each entry in this catalogue represents a pivotal aspect of our project, contributing to its overall success.

Confidentiality	*REQ-01*	*All data sent by the program shall be protected by encryption protocols such as TLS or SSL.*
	REQ-02	*The program shall also store confidential data in secure locations, with access restricted to authorized personnel only.*
	REQ-03	*The databases shall be protected against intentional or unintentional changes of data by a non-privileged user.*
Integrity	*REQ-04*	*The program shall ensure that the most sensitive data is regularly backed up*
	REQ-05	*The program shall require the user to authenticate to confirm any process involving log manipulation (e.g., removing) using RSA digital signatures.*
	REQ-06	*The program shall be able to verify the integrity of monitored IoT devices, ensuring that they have not been compromised or subjected to unauthorized modifications or replacements by using RSA digital signatures.*
Authentication	*REQ-07*	*The program shall allow the user, in addition to system authentication, to perform two-factor authentication (2FA) or multi-factor authentication (MFA).*
	REQ-08	*The program shall allow the user to use One Time Password (OTP) authentication applications to validate an entry.*
	REQ-09	*The program shall give an interval of at least three seconds between each login attempt to prevent brute force attacks.*
	REQ-10	*The program shall stop authenticating the user after five unsuccessful attempts. After that it shows a warning at the fifth and locks the user at the sixth try, for three minutes.*
Availability	*REQ-11*	*The program shall be able to collect data and logs of the IoT devices, without affect its storage efficiency.*
	REQ-12	*The program shall be highly available to ensure that users can access it whenever necessary.*
	REQ-13	*The program shall be designed to recover from failures to ensure that it continues to function as it did before the error. As such, the program is expected to work 99.9% of the time (8.76 hours of down-time at the end of a year).*
	REQ-14	*The program shall recover the data using its backup, within ten minutes.*
	REQ-15	*The program supports no more than thousand devices per user.*
General Requirements	*REQ-16*	*The user shall be notified if the program is accessed by an unrecognized device, adding a new layer of security and user awareness.*

IV. Security Test Plan

A. Penetration Testing

Our security testing strategy is based on a thorough and proactive approach. We use several comprehensive testing methods, both manual and automated, to identify and mitigate possible application vulnerabilities. We give priority to detecting the most common threats.

We take a meticulous approach to conducting penetration tests, using tools that are well-known in the cybersecurity field.

Nmap: We use *nmap* to carry out a detailed mapping of the network and ports, allowing an exhaustive analysis of the infrastructure and services running, identifying potential entry points.

OWASP ZAP: We used OWASP-ZAP to carry out analysis and identify vulnerabilities in the dependencies of the API used to run our application.

Wireshark: This software plays a key role in security, once again enabling detailed analysis of the network traffic. It is essential for identifying security gaps that both previous ones cannot analyse in such detail, namely the anomalous behaviour of data flows.

B. Types of Attacks (Vulnerabilities)

Analysing the types of attacks makes it possible to identify gaps and weak points in the application, providing valuable insights for strengthening security and mitigating potential security risks.

Attack	***Vulnerability***	***Sensitive Part***
TLS/SSL Encryption Bypass	*Attempting to bypass or disable encryption protocols to transmit unencrypted data.*	*Transmission layer and data transfer functions where encryption protocols are implemented.*
Unauthorized Access to Secure Locations	*Exploiting weaknesses to gain unauthorized access to confidential data storage areas.*	*Secure storage locations and access control mechanisms.*
Database Threads	*SQL Injection attacks targeting databases or attempts to compromise them externally or internally.*	*Database management system, SQL queries, and data access points.*
Authentications Vulnerabilities	*Brute force attacks or exploiting weak authentication mechanisms.*	*User authentications modules, login processes and password management.*

C. Fuzzy Testing

We use OWASP-ZAP for fuzz testing, thus analyzing the outputs for certain inputs, we will then identify and correct possible vulnerabilities and strengthen the application's security.

Input	*Output*
Special Chars (SQL Injection)	*If the application does not have protection against SQL injection, the application could interpret the characters as part of an SQL query, leading to unauthorized access.*
	Example: "fabio.santos';--" in log in form
Long Inputs	*If the application is not prepared to deal with large data inputs, this can result in poor performance, leading to a failure, buffer overflow or even interruption of a service.*
	Example:" aaaaaaaaaaaaaaaaaaa…" on any input value.
Scripts (Cross-Site Scripting-XSS)	*If the application has no defences against XSS attacks, allowing malicious code to be executed on the user's browser.*
	Example: https://example-api.com/user/profile?name=<script>alert ('XSS Attack!');</script>
Manipulating the URL or Form parameters	*If the application does not properly validate entries, this can lead to unauthorized redirections, improper access, or manipulation of functionalities.*
	Example: https://example-iot-platform.com/change-device-access *Manipulated Form Fields:* *- DeviceID: XYZ123* *- DeviceType: unauthorized_access_point* *- FirmwareVersion: 1.0.0* *- Manufacturer: RogueTech* *- AdminUsername: admin* *- AdminPassword: malicIousP@ssw0rd*
Escape Chars Format (Command Injection)	*The application can interpret the characters as application commands, allowing unauthorized execution of these commands.*
	Example: {{os.system("whoami")}}

Authentication Manipulation (Brute Force)	*If the application does not have limitations or blocks for repeated login attempts, it could compromise user confidentiality.*
	Example: *Attempted Username: user123* *Attempted Passwords (from Dictionary): [password123, secret@123, 123456, userpass]*
OSINT based Fuzzing	*Use OSINT tools to discover information about certain users on other platforms (like social media).*
	Example: "layla1307" if a person has a dog called Layla and was born on 07/13.
Unexpected character sequence	*The application may fail input validation, resulting in a generic error message, or unexpected behavior.*
	Example: *https://example-api.com/data/query?search=< > & ; /*
Authentication Manipulation (Token)	*The application can accept forge tokens allowing unauthorized to an attacker.*
	Example: *eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiAiMTIzNDU2Nzg5MCIsICJyb2xlIjogInVzZXIifQ.XXXX (a base64 token, where the first part represents the header, second part the payload and lastly the signature)*
Boolean-based Blind SQL Injection with Automated Fuzzing for Password Discovery	*Using SQL Injection on the password receiving a Boolean value. It is possible to use wildcards ("%a%") in order to understand if a character is in a password. Doing this for every character we can restrict the number of characters in the possible alphabet using a script that makes an automatic fuzzing. After knowing every character in the password, it is possible to start fuzzing character by character until the whole password is discovered.*
	Example: "admin123' UNION SELECT 1,2,3 from users where username like 'a%" on the login form that returns a true or false if the query (originally log in) was successful.

D. Test Specifications For Each Security Requirement

This table serves as a comprehensive guide delineating the intricate details of our security measures. Within this catalogue, every entry encapsulates a set of test specifications, each playing a crucial role in validating and fortifying our security requirements.

TST-01	*Send a set of payloads through all communication and message channels to get a response from the program, capture it with Wireshark and verify if the data was transmitted securely with de encryption protocol (TLS/SSL).*	*REQ-01*
TST-02	*Attempt to access to device logs data with a non-privileged user.*	*REQ-02*
TST-03	*Test to change, delete, and create data in a database with a non-root user.*	*REQ-03*
TST-04	*Test and weekly evaluate the backups integrity and the data retrieved.*	*REQ-04*
TST-05	*Try to perform a manipulation to a log and check if the signature corresponds to the user that performed that action.*	*REQ-05*
TST-06	*Try to perform a modification on the device's firmware and software, and check if the signature corresponds to that action.*	*REQ-06*
TST-07	*Add to a giving user, the option to enable two-factor authentication and verify if the account receives the OTP (One Time Password) via email or SMS.*	*REQ-07*
TST-08	*Test if the user receives the OTP (One Time Password) when he tries to change its password and when he tries to perform any change on the list of connected devices.*	*REQ-08*
TST-09	*Try to login consecutively in less than 3 seconds and observe that it is not possible and try to login consecutively 3 seconds apart and the system should already allow it.*	*REQ-09*
TST-10	*Test logging in using wrong passwords. After the fifth attempt, check for the warning that confirms the account has been locked in a specific amount of time, and make a 6th attempt to check its validity.*	*REQ-10*
TST-11	*Add a device to the program, create a ten thousand of logs and evaluate its behaviour. Perform the same test with a thousand devices at the same time and compare the response time performance on both tests. The expected result should be that both evaluations don't differ more than 20%.*	*REQ-11*
TST-12	*Create a bot that fills the program with requests to test the availability during a heavy use of the program. To measure it, check if the failures percentage don't surpass the 5%.*	*REQ-12*

TST-13	*Monitor the program during a week and test and evaluate the uptime and downtime on normal usage and when errors are being forced. Preferably the test should be run in a heavy use of the program and document the program response time and system usage daily.*	*REQ-13*
TST-14	*Try to delete video footage, logs, and configurations from a set of devices from the database, request a data recover and test the time it takes to the program to recover that same data from the backup.*	*REQ-14*
TST-15	*Test if a user can add more than a thousand devices. The expected result should be that the 1001st device gets denied access.*	*REQ-15*
TST-16	*Login with a new device and verify if the user gets a notification about the event in the current device and via email/SMS.*	*REQ-16*

V. Hazards analysis

The rapid proliferation of Internet of Things (IoT) devices has ushered in a new era of interconnected technologies, encompassing everything from smart homes to critical industrial systems. However, this exponential growth is accompanied by a series of critical challenges that demand immediate attention. This chapter delves into the hazards inherent in the current IoT scenario, highlighting the urgency to address issues such as inadequate security measures from manufacturers, the prevalence of vulnerable devices, and the widespread technological illiteracy among end-users.

One major problem is that many IoT devices don't have strong security protections from the companies that make them. A lot of these devices are sold without basic security features, making it easy for bad actors to hack into them. This lack of protection puts users' privacy at risk and makes the devices vulnerable to being used in harmful activities, like attacks that can disrupt digital systems (known as denial-of-service attacks). This poses a big threat to the stability of our digital networks.

In response to these pressing concerns, this report endeavours to present an innovative solution that addresses critical aspects such as secure development, solution quality, confidentiality, integrity, non-repudiation, accountability, and authenticity. The focus is not only on mitigating existing vulnerabilities, but also on fostering a culture of security throughout the entire IoT ecosystem. The goal is to provide end-users with an enhanced level of safety and a seamless experience within the IoT landscape.

As we embark on this exploration of hazards within the IoT realm, each subsequent section will delve into specific challenges and propose practical solutions. From identifying and blocking malicious devices to addressing firmware vulnerabilities and preventing denial-of-service attacks, this report aims to pave the way for a future where IoT is synonymous with reliability, robustness, and security.

With this in mind, we have separated 5 possible hazards that we consider the most important:

1. *Malicious/Counterfeited Device* - Ensure the application identifies and blocks malicious or counterfeit devices from being added or utilized within the system.
2. *Firmware Vulnerability* - Detect and address vulnerabilities in device firmware upon addition to the system to prevent potential exploits.
3. *Denial of Service (DoS) Prevention* - Implement measures to prevent or mitigate DoS attacks that could disrupt the system's functionality or availability.
4. *Insecure Communication* - Ensure secure communication channels between devices and the system to prevent interception or manipulation of transmitted data.
5. *Elevation of Privileges* - Implement controls to prevent unauthorized access or privilege escalation within the system.

VI. Conclusion

In the rapidly evolving landscape of the Internet of Things (IoT), where smart devices seamlessly connect to enhance our lives, the need for robust, reliable, and secure systems becomes paramount. This comprehensive report has delved into the challenges and hazards inherent in the current IoT scenario and proposed an innovative solution designed to address these critical issues.

Our secure development life cycle, as outlined in this report, encompasses crucial stages from education and awareness to project inception, analysis, and requirements, architectural design, implementation, and deployment. Each phase is meticulously designed to address specific security concerns, ensuring a robust and resilient solution.

The emphasis on education and awareness reflects our commitment to instilling a culture of security within the team, recognizing that security is everyone's responsibility. The inclusion of cryptography standards, security requirements, and secure design principles further fortifies our solution against potential vulnerabilities.

The penetration testing and security test plan detailed in this report exemplify our proactive approach to identifying and mitigating security threats. Through a combination of manual and automated testing methods, we have endeavoured to simulate real-world scenarios, ensuring the effectiveness of our security measures.

The security requirements outlined in our report, covering confidentiality, integrity, authentication, and availability, provide a structured framework for the development and evaluation of our solution. These requirements, coupled with thorough testing specifications, aim to create a secure environment for users and protect against potential threats.

The hazards analysed, ranging from the threat of malicious and counterfeit devices to firmware vulnerabilities, denial-of-service attacks, insecure communication channels, and the risk of unauthorized access, underscore the vulnerabilities that exist within the expansive IoT ecosystem. Recognizing these challenges, our project stands out by offering a holistic approach, aggregating data from devices of different brands while prioritizing security and user privacy.

As we conclude this report, our commitment to shaping a future where IoT is synonymous with reliability, robustness, and security remains unwavering. By addressing hazards head-on and offering a comprehensive solution, we aspire to contribute to an IoT ecosystem that not only meets the demands of today but also ensures a secure and seamless experience for users in the years to come. In doing so, we envision a future where the potential of IoT is fully realized, accompanied by a heightened level of trust and confidence in the security of connected devices.