A robust approach for modern digital logistics in supermarket chains

Vítor Santos
MSc. Cibersecurity
DETI, University of Aveiro
Aveiro, Portugal
vitor.mtsantos@ua.pt, NMEC 107186

João Luís
MSc. Cibersecurity
DETI, University of Aveiro
Aveiro, Portugal
jnluis@ua.pt, NMEC 107403

Lis Araújo
MSc. Cibersecurity
DETI, University of Aveiro
Aveiro, Portugal
lisaraujo@ua.pt, NMEC 123987

Ricardo Quintaneiro
MSc. Cibersecurity
DETI, University of Aveiro
Aveiro, Portugal
ricardoquintaneiro@ua.pt, NMEC 110056

***Abstract*—This paper presents a modern approach to managing and securing logistics digitally within modern supermarket chains. Our solution addresses the limitations of traditional centralized architectures by incorporating a decentralized framework that enhances both operational reliability and security. Each of the features presented is designed to minimize downtime and protect against both cyber and operational threats. To ensure data confidentiality, integrity and authenticity, the system integrates secure protocols and comprehensive monitoring tools, and proactively identifies and mitigates vulnerabilities by leveraging a secure development lifecycle, ensuring compliance with industry standards and resilience against cyber attacks. The proposed solution not only optimizes supply chain logistics but also enhances scalability and risk management for supermarket chains. This work aims to set a new standard in digital logistics, combining robust and secure software with operational efficiency to address the complex and evolving needs of modern retail supply chains.**

***Index Terms*—reliability, robustness, security, logistics, retail**

I. Introduction

When it comes to supermarket chains, having efficient and secure logistics is critical for the success of their retail operation as it allows maintaining continuity of operation, meet consumer demand and finally maximize profits. Supermarket chains often rely on sophisticated logistical software to manage these events. A logistical system that is well-functioning ensures the seamless movement of products from suppliers to warehouses, and eventually to stores, optimizes inventory management and minimizes waste due to spoilage or overstocking.

However, traditional systems often face significant challenges and suffer from reliability issues such as system crashes, inaccurate inventory tracking, from poor scalability and security risks, among others. This can lead to disruptions of the supply chain and to significant financial losses, not only from commercial operations but also from hurting the image of the brand and the perception of its customers.

This paper introduces a robust, decentralized logistics management solution tailored for supermarket chains, designed to address the aforementioned critical challenges of reliability, scalability and security. Our system architecture emphasizes redundancy which means scalability through distributed computing, minimizing single points of failure and ensuring high availability even under heavy traffic or system stress.

Our motivation behind our proposal is simple. Traditional systems in supermarket chains are typically centralized, which poses several risks and challenges:

- Single points of failure can bring down the entire network, resulting in widespread operational downtime. This could manifest as inventory mismatches, delayed order processing or even system crashes during peak usage times, such as holidays or promotional sales.
- Security vulnerabilities in these systems expose them to potential major cyberattacks, including data breaches, tampering with orders or even complete system sabotage.
- Existing systems often struggle with scalability, unable to handle the increased load as supermarket chains grow, adding more store locations and warehouses to the network.

In addition, logistical software must accurately track and manage perishable goods to prevent waste and ensure legal compliance. Many current systems fail to track expiration dates accurately, leading to spoilage, customer dissatisfaction and potential legal ramifications for selling expired goods. Furthermore, the lack of real-time error detection and handling exacerbates inventory management issues, with miscounted or misplaced items going unnoticed until they create larger problems.

Our decentralized supermarket logistics system addresses the limitations of traditional centralized architectures by introducing several key features designed to improve operational reliability and security such as automated backup and recovery,

dynamic load balancing, performance monitoring and error detection, end-to-end encryption for data transfers and policies to enforce security audits. These features coupled with real-time inventory monitoring and tracking of goods ensure that stock levels are accurately maintained and that perishable items are managed efficiently throughout the supply chain.

To achieve this it incorporates a secure development life-cycle. Each component of the system undergoes rigorous testing, including vulnerability assessments and penetration testing, meaning that potential attack vectors are identified and mitigated before they are integrated into the system. Additionally, role-based access control and state of the art encryption are integrated to protect sensitive data such as order details, inventory levels and supplier contracts from unauthorized access, ensuring a secure and robust operational environment.

The solution will also adhere to data protection standards such as GDPR for consumer privacy. Integrity is maintained by using hash-based validation, ensuring that no tampering occurs during communications. Accountability and non-repudiation are enforced through digital signatures and logging mechanisms that provide a verifiable audit trail, ensuring that all transactions and system interactions can be traced and verified if needed, supporting both regulatory compliance and operational transparency.

By implementing these advanced features in such a way, our proposed system provides a resilient and secure platform for managing supermarket logistics, reducing operational risks and ensuring continuous product availability.

II. Secure Life Cycle Description

This section outlines the secure development life cycle that will be implemented in our solution to ensure compliance with security requirements.

A. Education and Awareness - Provide Training

Security is a shared responsibility across the entire organization. It is crucial that all employees, regardless of their role, possess a foundational understanding of how to apply appropriate security measures within the scope of their work. Effective training equips the team with the knowledge needed to incorporate security into software development, ensuring that the product meets both quality and security standards while addressing business requirements and delivering value to users. While not everyone may be a security specialist, it is essential to grasp the attacker's perspective and objectives to take proactive steps that prevent potential threats.

This training should be done in both the technical and conceptual aspects, through workshops and presentations, focusing on different security scope concepts such as risk assessment, use of patterns and good practices (e.g. OWASP Cheat Sheet Series [1]), and also non-functional requirements. Also, all developers, testers, and program managers must complete at least one security training class each year.

In our system in particular, reliability and availability are very important aspects since a proactive approach to these concepts allows us to reduce the risk of system failures or computer attacks, especially during the holiday season or operational peaks, where the economic impact on a company of this kind would be greatest.

B. Project Inception - Use approved tools

Establishing a list of approved tools is not only a secure practice, but also required for ISO/IEC 27001 compliance, an international standard for information security management.

Tools will be selected for the approved tool list using the following criteria:

1) The tool's author must have a positive reputation in security.
2) The tool must not possess any known high or critical severity vulnerabilities, as classified by the CVSS v4.0 [2].
3) The tool must be currently supported by its author.
4) The tool must have a compatible license with developed software.
5) The tool undergoes regular security testing.

Ensuring tools are valid for approval will involve using research and news resources like Snyk [3], NVD [4], the CVE program [5], CSO online [6], Dark Reading [7], repository host sites and other tool author resources to verify regular security checking and compliance.

C. Analysis and Requirements - Perform threat modeling

Before identifying and assessing the potential threats and risks on their own, business functionality and structure must be minimally outlined.

1) Actors: During the stakeholder interview process, actors will be outlined, not just limited to human system users, but also external company systems that should be integrated. Sales system and identity provision systems are some of the expected external components.

The product owner should take special note of logistical roles in the company, as well as existing processes and systems.

2) Use scenarios: Use scenarios should describe the observed and expected workflows actors will perform with the system. Besides interviewing company managers, the product owner should also interview workforce employees in the company's operational logistics level to not only understand workflow details, but also concerns and problems with current systems and approaches.

Use scenarios will be documented as sequences of steps and illustrated with flowcharts.

3) Use cases: Use scenarios are analyzed to derive use cases. Some scenarios will be complex, reflecting the many variables contributing and influencing modern logistic chains, requiring some scenarios to be decomposed and bundled as alternate flows to formulate use cases.

Use cases will be described and cataloged using the format displayed on Table I.

Use case relationships will also be illustrated using UML Use Case diagrams [8] for a concise overview of functionality.

TABLE I
USE CASE FORMAT

Use case name	
Use case number	
System	
Stakeholders/actors	
Use case goal	
Primary actor	
Preconditions	
Basic flow	
Alternative flows	

4) Assets: Assets are defined as an object of value in a system that needs protection. Assets will be identified and prioritized with stakeholder interviews and research on strategic assets in the supermarket industry. The highest priority assets are expected to be mission-critical and the most sensitive to disruptions and attacks.

These high-level assets will drive threat prioritization as their compromise would cascade into a near-total communications breakdown on the logistics chain.

5) Architecture overview: In the context of threat modeling, a technical in-depth low-level architecture report is not necessary as threats on this level are implementation dependent and detected during development. This is why the architecture overview will be modeled using a variant of Level 2 DFD (Data Flow Diagram) with some explicit architecture elements (databases, caches, message brokers, applications, etc...) [9]. Level 2 DFDs provide adequate balance between simplicity and detail, exposing enough information to understand attack surfaces and security properties.

6) Threat Identification: In practice, threat identification is done via a meeting with all team members. Microsoft SDL recommends this meeting to last 2 hours: the first hour to reach a common understanding of system functionality and use scenarios, the second hour to identify threats, mitigations and risks [9].

Based on OWASP recommendations, the STRIDE methodology will be used to enumerate a range of basic potential threats [10].

Advanced threat identification requires offensive thinking, i.e. thinking like an attacker. There's no single definitive methodology for this [9], so all team members are invited to an exercise in imagining scenarios where security assumptions are challenged and security controls fail.

At this stage, threat identification will require technical knowledge of common system weaknesses, vulnerabilities and attacks.

D. Architectural and Detailed Design - Establish design requirements

Thinking about design and architecture in a secure way from the start of the system's development makes it possible to reduce the risk of future problems, which would be more expensive to solve than if they were thought about and mitigated at an early stage. Following this ideas, we can use Saltzer and Schroeder's [11] design principles to guide us in this process.

Bearing in mind that part of the system's target users are people with no technical knowledge, presenting a user-friendly interface with good usability, i.e. above 68 on the System Usability Scale (SUS) [12] and easy access to real-time data and analytics should be an aspect to consider at this stage.

It is also essential to implement an access policy, so that only those who need it are allowed to access or modify the data (also known as least privilege principle). An Role-Based Access Control (RBAC) model can be used for this, granting separation of privileges, in which the training programs mentioned previously help to understand the risks associated with each role.

A system of this kind must have an elastic infrastructure, i.e. the ability to scale both horizontally and vertically, since the integration of new shops, warehouses or suppliers must not affect the system's overall performance. Both this infrastructure and authentication should apply the open design principle, i.e. avoiding obscurity and applying transparent security protocols.

The system should enforce complete mediation by validating each access request dynamically. Instead of assuming that prior access authorization suffices, the system checks permissions each time a user accesses or modifies data, ensuring that any change in user privileges or security policies is applied immediately.

In architectural terms, to achieve a decentralized architecture, a cloud solution can be used which, as well as offering more redundancy, leverages flexibility and scalability. By distributing resources across multiple regions or availability zones within the cloud, the system avoids relying on a single point of failure, which also aligns with the principle of least common mechanism by minimizing shared dependencies among different parts of the system.

Both the architecture and other decisions should be well documented from the start of the project, so that both future participants and stakeholders understand all the decisions made up to a certain point.

E. Implementation and Testing - Perform dynamic analysis security (DAST)

The implementation phase of the supermarket logistics system will focus on adhering to the secure development practices outlined in the earlier stages. All code will be developed within approved IDEs, applying secure coding practices. The system will be containerized using Docker and orchestrated with Kubernetes to ensure consistent and scalable deployments across different environments. Secure development practices will be enforced through Git repositories hosted on GitHub or GitLab, with Jenkins automating the continuous integration/continuous deployment (CI/CD) pipelines. During the CI/CD process, security tools like OWASP ZAP and Nikto will be utilized to detect vulnerabilities in the web applications.

As the system is implemented, comprehensive DAST scans will be performed as part of the integration and system testing stages. Integration testing will validate the interactions

between modules, ensuring that the system functions cohesively, while DAST will assess the system's behavior under various conditions, allowing us to identify vulnerabilities that might arise due to external interactions. Performance tests will simulate high-traffic scenarios to evaluate the system's ability to handle operational peaks, while DAST will continue to detect potential security flaws in real time as the system processes high volumes of requests.

To ensure comprehensive security and monitoring, Security Information and Event Management (SIEM) tools like Wazuh will be integrated. Wazuh will facilitate real-time monitoring of security events and ensure that any anomalies or potential threats are quickly identified and addressed. Additionally, Prometheus will be employed for system performance monitoring and alerting, enabling the team to detect performance bottlenecks or failures during operational peaks.

Following implementation, DAST results will be reviewed and prioritized for mitigation, ensuring that all critical vulnerabilities are addressed before the next phase. Finally, a formal User Acceptance Testing (UAT) phase will be conducted, where actual end-users, such as local inventory managers and administrators, interact with the system to ensure it meets their needs, functions as expected and delivers a user-friendly experience.

F. Release, Deployment, and Support - Establish a Standard Incident Response process Process

The team will develop and maintain a comprehensive Incident Response Plan (IRP) that includes the roles, responsibilities, and communication protocols for all team members involved in incident response. Regular training sessions and exercises will be conducted to familiarize the incident response team with simulated attack scenarios, ensuring prompt and effective responses. Additionally, monitoring tools such as Wazuh Security Information and Event Management (SIEM) and Prometheus will be configured to provide real-time alerts for any unusual activity, enabling early detection of potential security threats.

In the event of an incident, detection and analysis will be conducted to assess the nature and scope of the incident. Alerts from Wazuh and Prometheus will be reviewed to determine the type, severity and potential impact of the event. An assigned incident responder will document all relevant details, including timestamps, affected systems and preliminary analysis of the incident's origin. Log files, network traffic data and attack-defense trees will be examined to identify patterns and determine the most effective response approach.

Once the incident is confirmed, containment measures are implemented to isolate the affected systems and prevent the incident from spreading. The containment strategy may involve short-term containment (such as isolating affected nodes from the network) and long-term containment (such as deploying patches or reconfiguring network parameters). Access controls will be reviewed and adjusted as necessary, ensuring that only authorized personnel can access or modify affected systems. Any temporary fixes applied during containment will be documented and tested to avoid unintended consequences on overall system operations.

After containment, the next process is eradication and that involves identifying and eliminating the root cause of the incident, such as removing malware, closing vulnerable ports or applying necessary patches. During recovery, systems will be carefully restored to normal operation in a controlled manner. This may involve restoring data from backups, revalidating the integrity of affected components and gradually reintegrating them into production. Continuous monitoring will be maintained to confirm that the incident is fully resolved and that no residual vulnerabilities remain.

Following successful containment and recovery, a post-incident review will be conducted to document and analyze the incident response. This includes reviewing all actions taken, assessing the effectiveness of the response and identifying areas for improvement. The insights gained will be incorporated into the Incident Response Plan and may lead to updated security measures, enhanced monitoring rules or new training programs. Lessons learned will also be shared with relevant stakeholders to reinforce a proactive security culture across the organization.

III. Security Requirements

This section describes our secure development life cycle, designed to meet strict security requirements. By incorporating industry best practices and robust security measures, we aim to strengthen the solution against potential risks and vulnerabilities, underscoring our commitment to delivering a product that exceeds information security standards.

A small glossary of particular terms that may raise doubts is also provided to keep requirements small and non-redundant:

1) **Incremental snapshot**: a point-in-time database backup that only records changes to the database since the last incremental snapshot. The first snapshot is a full copy of the database [13].
2) **Backup**: Refers to a database backup, which consists of a full database copy, independent from the source database for restoration [14].

IV. Security Test Plan

In order to aid in validating (please check correctness in using this term vs using the term "verification" in this context) security requirements and ensure general system security, as specified by ISO/IEC 25002:2024 [18], within reasonable business expectations and available resources, we propose a security test plan.

Given the particular context of this project, a few business and system factors must be highlighted and taken into consideration:

1) **Large scale** - At the supermarket chain's current national scale, the proposed system is expected to be relatively large in scale as well, requiring sparse distribution over geographic regions and a high number of network processes for redundancy. For instance, failures in consistent and secure deployment of new infrastructure components

TABLE II
SECURITY REQUIREMENTS CATALOGUE

This table outlines the key criteria guiding our solution, with each entry highlighting a crucial element of the project that plays a significant role in its success.

Confidentiality	REQ-01	User authentication passphrases shall be stored in a hashed format using the Password-based Key Derivation Function 2 with the Secure Hashing Algorithm 3 (SHA-3) standard.
	REQ-02	User authentication passphrase salts shall be obtained using a cryptographically-secure pseudo-random number generator that relies on the operating system built-in functionality.
	REQ-03	All communications between system components, including client-server interactions and data exchanges with external systems, shall be encrypted using version 1.3 of Transport Layer Security.
	REQ-04	The system shall only allow the following TLS ciphersuites: 1) TLS_AES_128_CCM_8_SHA256 2) TLS_AES_128_CCM_SHA256 3) TLS_AES_128_GCM_SHA256 4) TLS_AES_256_GCM_SHA384 5) TLS_CHACHA20_POLY1305_SHA256
	REQ-05	The role-based access control model shall be enforced as the main access-control mechanism.
Integrity	REQ-06	All organization and user data shall be fully backed-up once per month and incremental snapshots (sourced from the last backup) once per week, during the early morning (starting at 0:00 up until 2 hours before morning opening time) of Tuesday.
	REQ-07	All organization and user data back-ups shall be tested once immediately after creation and then routinely tested once per week.
	REQ-08	The system shall use transaction mechanisms that ensure atomicity. All steps in a transaction are completed successfully or none are applied at all.
Availability	REQ-09	The system shall be structured to handle failures effectively, maintaining consistent functionality even after an error occurs. As such, the system availability shall be greater than 0.999 (8.76 hours of down-time per year).
	REQ-10	The system shall be subject to bi-weekly maintenance periods on Mondays, starting at 0:00 and ending 2 hours before typical morning opening hours.
	REQ-11	Maintenance periods shall be partial in nature, i.e. do not encompass the entire system at any given time, covering at least 10% and at most 25% of system infrastructure.
	REQ-12	The system shall continue to operate with no deviation from expected behavior, even during maintenance periods.
	REQ-13	Load balancers shall distribute incoming network traffic efficiently: automatic detection of server health and routing traffic away from any failed or underperforming instances.
	REQ-14	The system shall recover the data using its backup, within seven hours.
	REQ-15	The system must have automated scaling capabilities such has virtual servers that can be dynamically allocated based on real-time traffic and load.
Authentication	REQ-16	The system shall enforce the use of an OTP (one-time password) mechanism on top of the existing password-based mechanism for all users, in order to accomplish multi-factor authentication.
	REQ-17	The system shall enforce password resets for all accounts found to be holding a blacklisted password, as revealed by the Have I Been Pwned (HIBP) API [15].
Organizational	REQ-18	All incoming source code changes shall be approved during a peer-review process (e.g. Pull Request Review for the Git version control system) by the Product Owner(s) before they are effective.
	REQ-19	All incoming source code changes shall pass all tests under an automated Continuous Integration/Continuous Delivery pipeline before they are effective.
	REQ-20	All employees (users) shall undergo a monthly two-hour training session on cybersecurity best practices and social engineering attack awareness.
Accountability	REQ-21	All user authentication operations pertaining to existing users (i.e "log-in" actions, as opposed to account creation), including successful authentication attempts, shall be logged as entries containing the inserted username, Unix timestamp at the moment of logging and IPv4 address of the authentication request sender.
	REQ-22	Authentication request logs shall be aggregated in a separate database system that shall only be accessed (read & write) by server administrator level entities.
Compliance	REQ-23	Security audits for GDPR [16] and ISO/IEC 27001 [17] compliance shall be conducted at least once a year.

(e.g. infrastructure authentication misconfiguration) are statistically likelier to occur.

2) **Ease-of-access** - System end-points will be relatively exposed to physical use by nefarious actors, mainly point-of-sale terminals, which, while not an internal logistics system component, still interface with the inventory management system and may represent an attack vector.
3) **Employee training** - Current company employees are not trained to recognize or handle social manipulation attacks, as company recruiters have dismissed this skill to be secondary at best. As such, we expect employees to be somewhat susceptible to external/common phishing attacks and highly susceptible to internal phishing attacks.

A. Assets

A pragmatic and economically viable security test plan must focus on mission-critical assets. To assist in visualizing these assets, a data flow diagram was conceived:

General Testing	REQ-24	Vulnerability assessments shall be conducted at least once a year.
	REQ-25	Penetration testing shall be conducted at least twice per year.
	REQ-26	Business continuity plans (BCPs) shall be tested and revised at least once a year.
	REQ-27	Disaster recovery plans (DRPs) shall be tested and revised at least twice per year.
User-Specific	REQ-28	User-created passwords shall be no less than 8 characters in length, allowing the usage of all printable characters, as specified by UTF-32, and whitespaces.

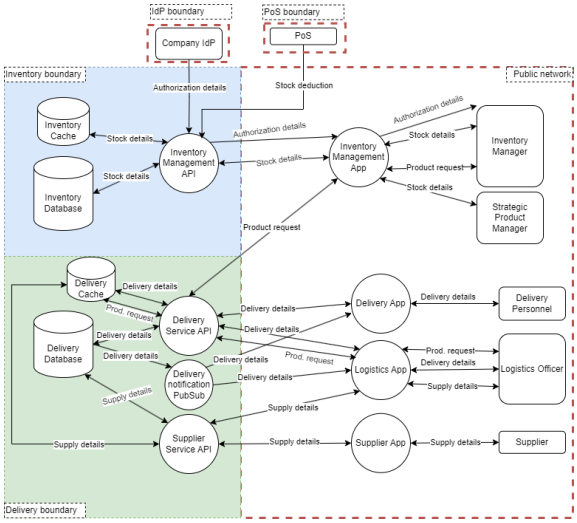

Fig. 1. Data flow overview

It should be noted that the company's IdP (identity provider) and point-of-sale systems are external to the logistics systems and are simply being integrated for consistency and flexibility throughout the enterprise.

Ordered by importance, the mission critical assets the security test plan must primarily focus on are:

1) **Delivery coordination system** - A total failure to receive a delivery request in the first place will quickly lead to widespread product shortages and massive profit loss. Delays, while relatively less impactful, are just as likely to cause profit loss, especially in high-demand periods.
2) **Inventory records** - Without a notion of inventory quantity and product catalog, there is simply little way to reliably and accurately determine what is needed and, therefore, what deliveries to request.
3) **Company IdP integration** - The IdP serves as the main authentication and authorization mechanism for all employees using the system. Without this, role-based access restrictions are neutralized and ineffective, allowing any actor to perform any operation in the system. Trying to implement an independent logistics IdP would simply originate further implementation complexity and be more prone to inconsistencies and errors.
4) **Point-of-sale integration** - Failure to properly integrate the PoS system will result in synchronization problems with sales and inventory records never reflecting decreases in product quantity, in the worst case scenario.

B. Penetration Testing

1) Strategy: A "gray box" approach will be used to perform penetration testing, providing some underlying system and business information to testers about the selected scope, but not all information there is as it could easily detract penetration testing from being more realistic to expected attack scenarios. Some scope components will be provided with more information due to the higher risk their exploitation may pose.

2) System & Business Scope: Penetration testers will be limited to the following system and business scope, defining supporting resources and target components:

1) **Employee information** - Testers will have access to a set of employee email addresses and a simple company role name for social engineering. These email addresses will consist mostly of delivery personnel, inventory managers and logistics officers. A single fake email address using the company's own email domain will also be made available.
2) **Architecture overview and endpoints** - Testers will be provided with a basic diagram and description of the system's overall architecture and high-level processes. Details about low-level processes or code are not briefed. Testers will also be briefed on all endpoints and where they can be accessed.
3) **Cloud infrastructure configuration** - Part of the logistics system will be deployed using cloud infrastructure services on datacenters for scalability. Testers may exploit this type of configuration, as it can be conducive to errors from lack of experience in the complex cloud ecosystem. Vulnerabilities in this aspect have also been reported to be catastrophic [19].
4) **Web application** - Testers will try to attack the delivery web application with the full attack scope, while the inventory management web application should not be tested for DoS attacks and data leakage. Web applications generally present a wider variety and quantity of vulnerabilities [20]. We deemed the delivery web app to be a higher risk asset as its frontend is expected to be exposed to a greater diversity of environments, i.e. networks and platforms, including those potentially compromised.
5) **Networks** - Testers will be given access to the guest and business networks of supermarkets and delivery centers.

3) Types of Attacks (Vulnerabilities): Understanding the various attack vectors and associated vulnerabilities is crucial for identifying potential security flaws within the system. This analysis helps in recognizing sensitive areas that may be exposed to risks, enabling the development of targeted strategies to enhance the application's overall security posture and safeguard critical assets against potential exploitation.

TABLE III
ATTACK TYPES AND ASSOCIATED VULNERABILITIES CATALOG

The table below provides a detailed analysis of five major attack types, outlining the associated vulnerabilities and the sensitive system components that could be exploited. This catalog serves as a reference for penetration testers, helping to define the attack scope, identify critical risk areas, and guide mitigation strategies to enhance the overall security of the application.

Attack type	Description	Affected components
Broken Authentication	Exploiting weak authentication mechanisms or using brute force attacks to gain unauthorized access to user or administrative accounts	User authentication modules, login processes, and password management systems.
Denial of Service (DoS)	Overloading system resources to render the application or network infrastructure unavailable to legitimate users.	Application servers, database systems, and network interfaces.
SQL Injection	Exploiting improperly sanitized SQL queries to execute malicious commands, access unauthorized data, or alter database content.	Database management systems, input validation mechanisms, and data access layers.
Privilege Escalation	Exploiting weaknesses in access control mechanisms to gain unauthorized administrative or elevated privileges.	Access control systems, role-based access management, and user session handling.
Data Leakage	Inadequate protection or exposure of sensitive data, such as customer information or payment details, through insecure APIs or storage.	Payment processing systems, customer data storage, third-party integrations, API endpoints and data transfer mechanisms.

4) Tooling: Penetration testing will be performed using a set of tools selected for their reputation, varied features, robustness and adequacy for the range of attacks permitted:

1) **nmap** - Network enumeration and system fingerprinting; well-known, highly supported, configurable and extensible to testers' needs. [21].
2) **nikto** - Web server scanner; reputable, useful for its vulnerability assessment capabilities in network server contexts, automatic attacks and support for anti-IDS mechanisms [22].
3) **OWASP ZAP** - Web application scanning tool; highly configurable, noteworthy in its assessment of web pages, request interception and forgery [23].
4) **Burp Suite Professional** - Same purpose as OWASP ZAP. Provides a majority of OWASP ZAP's features and more. Compared to ZAP, more control is provided over existing capabilities [24].
5) **Metasploit** - A highly reputable and popular penetration testing framework with wide support for exploiting most software vulnerabilities. Useful for fine-tuning and programming attacks to the tester's specific needs and conditions, which may not be automatically covered by the other tools on this list [25].

C. Fuzzy Testing

Fuzzing or fuzzy testing is a concept where the application is tested by being fed with unexpected, malformed or random inputs to uncover vulnerabilities and possible points of failure. The main objective is to guarantee that the system resists security vulnerabilities such as injection attacks, file validation vulnerabilities and denial-of-service exploits.

This type of testing will be integrated directly into the CI/CD pipeline. It automates fuzz tests through the CI/CD process so that every commit and/or merge will undergo comprehensive security testing for vulnerabilities before deployment. This guarantees that there is a security check at every step in the development chain, from creating new features to maintenance changes.

Automation tools, such as OWASP ZAP [23], ffuf [26] and other custom fuzzing scripts, will be employed to generate a wide range of test cases applicable to the system's architecture. The results of these tests will be analyzed and any identified vulnerabilities will trigger alerts, blocking the deployment of insecure builds.

Fuzzy testing within the CI/CD pipeline will target critical areas of the system, including:

- **Authentication Services**: To verify the robustness of JWT-based authentication and prevent bypass attacks.
- **File Upload Features**: To detect improper file validation that may lead to malicious code execution.
- **Database Interactions**: To discover vulnerabilities in SQL and NoSQL query handling.
- **API Endpoints**: To ensure resilience against command injections, buffer overflows and directory traversal attacks.

Input	Output
SQL Injection	If the system applications that have relational databases implemented lack SQL injection protection, they might interpret certain characters as part of an SQL query, potentially allowing unauthorized access, data manipulation and service disruption.
	Example: "jnluis ';– " in a login form
NoSQL Injection	When user input is not properly sanitized, NoSQL databases are susceptible to injection attacks. This is particularly critical given the use of queries in NoSQL systems that might directly parse user inputs, as is the case in our document based MongoDB databases.
	Example: "db.users.find({"username": username, "role": "manager"})"
Scripts (Cross-Site Scripting-XSS)	If the application has no defenses against XSS attacks, allowing malicious code to be executed on the user's browser.
	Example: https://example.com/profile?description=<script>alert ('This page is vulnerable to XSS attacks');</script>

File Upload	Improper file validation can allow attackers to upload malicious files if they have access to parts of the system where store managers or administrators upload product images or related documents, leading to arbitrary code execution or information leakage.
	Example: Uploading a .png file containing malicious JavaScript: malicious.png with content <script>alert('Compromised!');<script>
Buffer Overflow	If the application does not handle input size correctly this may allow attackers to overflow a buffer, potentially causing a crash or arbitrary code execution. This is applicable, for example, to the inventory management system's data entry fields such as those used for updating product descriptions or supplier details.
	Example: Providing an overly long string: "A" * 5000
Directory Traversal	Attackers might access unauthorized files on the server if the application is not properly sanitizing file paths. This test can target the API endpoints or file-upload functionality used for storing and retrieving documents or other files in the system.
	Example: ../../../../etc/passwd
Command Injection	Unsanitized input in sensitive areas may allow storage and possible execution of arbitrary commands on the server. It's relevant for backend APIs that handle store-wide administrative commands, such as bulk updates to product pricing or inventory records.
	Example: {{os.sytem(; rm -rf /)}}
Denial of Service (DoS)	Inputs that are not sanitized in the checkout and inventory APIs may cause excessive resource consumption, leading to service unavailability.
	Example: { "payload": "A" * 10**6 }
JWT Authentication Bypass	This test targets the JWT-based authentication mechanism used for user sessions within the logistics system. If the system does not validate JWTs properly, attackers may forge or manipulate tokens to gain unauthorized
	Example: Changing "role": "user" to "role": "admin"
XML External Entity (XXE) Injection	If the system parses XML inputs in an insecure way, then the attackers could exploit this to read sensitive server files, perform server-side request forgery (SSRF) or execute other malicious actions. This can impact the data import functionality, such as uploading supplier details or product catalogs in XML format.
	Example: <!DOCTYPE data [<!ENTITY file SYSTEM "file:///etc/passwd">]><data>&file;</data>

D. Test Specifications For Each Security Requirement

TABLE IV

TST-01	Register a new user with a known password in the application front-end interface and verify the hash stored in the databases is formatted using PBKDF2 with SHA-3, salt length of 32 (bytes) and 100,000 iterations in compliance with the format established in RFC 8018. Then, using the stored parameters from the PBKDF2 hash (such as salt and iterations), recreate the hash in a test environment with the known password and confirm that the generated hash matches the one stored in the database.	REQ-01
TST-02	Register multiple new users with the same password through the application front-end interface. Verify in the database that each password is hashed with a unique salt and that the salt values are being generated using the operating system's cryptographically-secure pseudo-random number generator, by monitoring system API calls to its CSPRNG.	REQ-02
TST-03	Connect to the server, using different TLS versions and ciphersuites, verifying that only the correct version and selected ciphersuites are allowed.	REQ-03 REQ-04
TST-04	Create multiple users with different roles. Attempt actions restricted to specific roles, such as accessing administrative settings with a non-admin account. Verify that users can only perform actions permitted by their assigned roles and that unauthorized attempts are blocked and logged.	REQ-05
TST-05	Verify the backup schedule by checking the system logs for full monthly backups and weekly incremental snapshots. Confirm that all backups occur within the specified time frame (0:00 to 2 hours before opening time) on Tuesdays.	REQ-06
TST-06	After each backup or snapshot, perform a routine test and restore it to a test environment to verify data accuracy, and guarantee they are complete and functional.	REQ-07
TST-07	Execute a fully successful transaction and verify that all intended changes are accurately applied. Simulate a partial failure by interrupting specific operations within a transaction and confirm that no intermediate changes persist in the database. Simulate a system crash during transaction execution and ensure that the database state is rolled back to its pre-transaction state. Perform concurrent transactions with overlapping operations and validate that data consistency and isolation are maintained during simultaneous access.	REQ-08
TST-08	Simulate high request volumes, i.e. twice the amount of traffic observed during peak activity, for 12 hours and verify the system's total downtime is no more than 0.01% of the testing period, i.e. 7 minutes and 12 seconds.	REQ-09
TST-09	Verify automated maintenance protocols start at 0:00 and finish 2 hours before the company-wide morning opening hour on Mondays.	REQ-10
TST-10	Execute automated maintenance protocols on an entirely outdated test system, monitor and verify the portion of updated system components to be between 10% and 25%.	REQ-11
TST-11	Execute automated maintenance protocols on a sample system and perform all end-to-end tests on the sample system; verify no end-to-end tests failed.	REQ-12

TST-12	Crash or overburden worker servers with redundant internal processes; monitor all network traffic and verify no load balancer traffic is redirected to crashed or underperforming instances after the load balancer identifies such instances.	REQ-13
TST-13	Initiate a data recovery process and measure the time required to recover various data types, including structured data (e.g., relational databases), unstructured data (e.g., files and media), and semi-structured data (e.g., JSON, XML) from the backup, ensuring that the recovery time is proportional to the size of the data and does not exceed 7 hours for full recovery. Specifically, for a dataset of 1 TB, the recovery time should not exceed 3 hours. Smaller datasets should have shorter recovery times, with 500 GB expected to recover in approximately 1 hour and a half, and 100 GB in 18 minutes. Simulate partial data recovery scenarios for each data type and confirm that recovery times are consistent with data size, demonstrating proportional correlation. Verify the integrity of both partially and fully recovered data by performing consistency checks against the original dataset. For structured data, compare checksums, row counts, and foreign key relationships to ensure data consistency. For unstructured data, compare file hashes (using SHA256) and confirm the proper restoration of files or media in their entirety. For semi-structured data, validate the structure and content by checking for schema conformity and data accuracy through automated validation tools.	REQ-14
TST-14	Simulate high traffic and load on the system using Apache JMeter, a load testing tool. Verify that virtual servers are automatically allocated to handle the increased demand and that system performance remains stable. Reduce the load and confirm that unused servers are deallocated to optimize resource usage.	REQ-15
TST-15	Attempt to login with a valid password but an invalid OTP and confirm that access is denied. Then log in with a valid password and a valid OTP to ensure access is granted. Verify that each OTP expires after use or a set period.	REQ-16
TST-16	Create a list of false users with a mix of valid passwords, previously breached passwords (from the HIBP API), and passwords found in common dictionary attack wordlists. Run the password blacklist check through all false users and verify that each blacklisted password is correctly identified as such. Ensure the system triggers a password reset and notifies the user with an appropriate message when a blacklisted password is detected. Additionally, verify that the system correctly falls back to the locally stored blacklist when the HIBP API is unavailable, and that the system applies the same blacklist check for passwords in the local list.	REQ-17
TST-17	Submit a source code change through a pull request and attempt to merge it without approval. Verify that the system blocks the merge until a Product Owner approves the change. Then, have a Product Owner review and approve the pull request to confirm that the merge is allowed only after approval.	REQ-18
TST-18	Submit a source code change with intentional errors for test failures through a pull request. Verify that the CI/CD pipeline detects the failures and blocks the changes from being merged even if approved. Then, submit a corrected version that passes all tests and confirm that the pipeline allows the changes to proceed. Check the pipeline logs to ensure all tests were executed and results recorded.	REQ-19
TST-19	Verify the attendance records for the monthly training sessions to ensure all employees participated. Check the training content to confirm it includes up to date best practices and social engineering awareness. Conduct a follow-up survey or quiz to assess employee understanding and retention of the training material.	REQ-20
TST-20	Perform a valid login attempt and check the system logs to confirm that an entry is recorded containing the correct username, Unix timestamp, and IPv4 address. Repeat with invalid login attempts to verify that they are also logged with the same details.	REQ-21
TST-21	Attempt to access the authentication request logs with non-administrator credentials and verify that access is denied. Use administrator credentials to access the logs and confirm that read and write operations are permitted. Make sure that the logs are stored in a separate database system and that access permissions are correctly configured.	REQ-22
TST-22	Verify that a security audit and a vulnerability assessment are conducted at least once per year by checking the audit and assessment schedule and reports. Confirm that the audit covers GDPR [16] and ISO/IEC 27001 [17] compliance requirements and that the assessment covers all system components, including network, application and infrastructure. Review the findings to ensure they address all relevant areas and verify that any identified issues are tracked, documented and resolved within the timeframe set.	REQ-23 REQ-24
TST-23	Conduct a review of the penetration testing records to ensure that tests are performed at least twice a year. Confirm that the testing scope includes critical assets such as user authentication systems, APIs and databases. Check for documentation of any discovered vulnerabilities and verify that remediation actions are taken, then test by simulating real-world attack scenarios.	REQ-25
TST-24	Verify that the business continuity plan (BCP) has been tested at least once in the past year and that the disaster recovery plans (DRPs) have been tested at least twice in the past year by reviewing the testing schedule and documentation. Confirm that the test scenarios include critical system failure, data recovery, and operational continuity during a disaster, simulating scenarios such as data loss, system outages, and critical infrastructure failure. Make sure the necessary revisions to the plans were made based on the test outcomes.	REQ-26 REQ-27
TST-25	Test the password creation functionality by attempting to register or update a user password that is shorter than 8 characters to ensure the system rejects it. Then, test with a password exactly 8 characters long and a variety of printable characters from UTF-32, including whitespaces, to verify that the system accepts it.	REQ-28

V. Hazard Analysis

This section focuses on the main findings of the hazard analysis that can be fully consulted in the hazard log spreadsheet file. In our analysis, the hazards found already have at least two safety measures which mitigate their potential impact, so the consequences of these are relatively controlled.

The following list details those potential hazards that could impact our logistics system. Each hazard highlights a specific scenario and its implications for the security and safety of the system.

- **Expired Perishable Goods**: Costumer might buy and consume a perishable good that has passed its expiration date due to a failure to track and rotate inventory items.
- **Supermarket Data Breach**: Unauthorized access to sensitive data, such as sales records, deliveries or supplier details.
- **Elevation of Privileges**: Unauthorized escalation of user privileges within system, in a PoS terminal.
- **Denial of Delivery Notification Service**: Delivery field personnel are unable to view their newly assigned deliveries.
- **System Crash**: The logistics system experiences a complete or partial crash, making it inaccessible to users.
- **Unauthorized Deletion of Critical Data**: Malicious actors or unauthorized users delete critical operational or customer data from the system.
- **Unauthorized Access to API Endpoints**: Unauthorized users or malicious actors gain access to API endpoints and perform actions such as reading, modifying, or deleting sensitive data or initiating unauthorized operations.
- **Suspicious Login Attempts**: Repeated unsuccessful login attempts to user accounts, potentially indicating low-level malicious activity or testing of weak passwords.

From the above list, the first, the sixth and the seventh hazards are the most dangerous in our perspective, being in the yellow zone of the risk matrix, i.e. undesirable.

VI. Conclusion

This project presents a practical solution to address critical challenges of reliability, robustness, and security in the management of digital logistics for supermarket chains. Using a decentralized architecture and a secure software development lifecycle, the system is designed to handle disruptions and cyber threats effectively. With features like end-to-end encryption, role-based access control, and compliance with international standards, it ensures data protection and smooth operations. The approach also includes thorough testing strategies, such as penetration and fuzzy testing, to uncover and fix vulnerabilities during development.

The results of this work highlight the importance of integrating technical innovation with proactive risk management to meet the evolving demands of the retail industry. The proposed solution improves logistical efficiency, scalability, and resilience, offering a modern approach to meet the demands of the retail industry. By prioritizing security and reliability, this work lays a foundation for organizations to enhance their logistics systems sustainably and securely.

References

[1] J. Manico, J. Maćkowski, K. W. Wall, and S. Z. Heigh, "Owasp cheat sheet series," accessed 25-October-2024. [Online]. Available: https://owasp.org/www-project-cheat-sheets/

[2] "Cvss v4.0 specification document," June 2024, accessed 24-October-2024. [Online]. Available: https://www.first.org/cvss/v4.0/specification-document

[3] "Snyk software supply chain security," accessed 24-October-2024. [Online]. Available: https://snyk.io/pt-BR/solutions/software-supply-chain-security/

[4] "National vulnerability database," accessed 24-October-2024. [Online]. Available: https://nvd.nist.gov/

[5] "Cve® program website homepage," accessed 24-October-2024. [Online]. Available: https://www.cve.org/

[6] "Cso online homepage," accessed 24-October-2024. [Online]. Available: https://www.csoonline.com/

[7] "Dark reading homepage," accessed 24-October-2024. [Online]. Available: https://www.darkreading.com/

[8] K. Fakhroutdinov, "Uml use case diagrams," Jan 2014, accessed 25-October-2024. [Online]. Available: https://www.uml-diagrams.org/use-case-diagrams.html

[9] "Perform secure design review and threat modeling," accessed 25-October-2024. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl/practices/secure-by-design

[10] L. Conklin, V. Drake, S. Strittmatter, Z. Braiterman, and A. Shostack, "Owasp threat modeling process," accessed 25-October-2024. [Online]. Available: https://owasp.org/www-community/Threat_Modeling_Process

[11] M. Mardjan, "Saltzer and schroeder's design principles," Jun 2024. [Online]. Available: https://nocomplexity.com/documents/securityarchitecture/architecture/saltzer_designprinciples.html

[12] N. Thomas, "How to use the system usability scale (sus) to evaluate the usability of your website," 2019, [Online; accessed 23-October-2024]. [Online]. Available: https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/

[13] Microsoft, "Incremental snapshots in azure virtual machines," https://learn.microsoft.com/en-us/azure/virtual-machines/disks-incremental-snapshots?tabs=azure-cli, 2024.

[14] P. Jayaram, "Understanding database snapshots vs database backups in sql server," https://www.sqlshack.com/understanding-database-snapshots-vs-database-backups-in-sql-server/, 2018.

[15] T. Hunt. (2024) Pwned passwords api (v3). Online documentation. [Online]. Available: https://haveibeenpwned.com/API/v3#PwnedPasswords

[16] "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)," Official Journal of the European Union, L119, 1-88, 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[17] *ISO/IEC 27001:2022: Information security, cybersecurity and privacy protection — Information security management systems — Requirements*, Std., 2022. [Online]. Available: https://www.iso.org/standard/27001

[18] *ISO/IEC 25002:2024: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality models overview and usage*, Std., 2024. [Online]. Available: https://www.iso.org/standard/78175.html

[19] SecurityWeek. (2024) Cloud misconfigurations expose 110,000 domains to extortion in widespread campaign. [Online]. Available: https://www.securityweek.com/cloud-misconfigurations-expose-110000-domains-to-extortion-in-widespread-campaign/

[20] OWASP Foundation, "Owasp top ten project," 2024. [Online]. Available: https://owasp.org/www-project-top-ten/

[21] G. Lyon. (2024) Nmap: The network mapper. [Online]. Available: https://nmap.org/

[22] C. Sullo and D. Lodge. (2024) Nikto2: Open source web server scanner. [Online]. Available: https://cirt.net/nikto2

[23] ZAP, "Fuzzing," accessed 25-November-2024. [Online]. Available: https://www.zaproxy.org/docs/desktop/addons/fuzzer/

[24] P. Ltd. (2024) Burp suite: Web vulnerability scanner and penetration testing toolkit. [Online]. Available: https://portswigger.net/burp
[25] R. LLC. (2024) Metasploit framework: Penetration testing software. Accessed: 2024-12-13. [Online]. Available: https://www.metasploit.com/
[26] ffuf, "Fuff fuzzer." [Online]. Available: https://github.com/ffuf/ffuf
[27] Microsoft, "Security training," [Online; accessed 22-October-2024]. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl/practices/security-training
[28] Microsoft, May 2012, [Online; accessed 22-October-2024]. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/cc307407(v=msdn.10)
[29] WKRC, "Walmart charged wrong prices for days due to internal system failure, 1600 stores affected," accessed 25-October-2024. [Online]. Available: https://www.abc3340.com/news/nation-world/walmart-charged-wrong-prices-internal-system-failure-1600-stores-affected-cincinnati-consumer-alerts-hundreds-of-stores-affected-price-point-scanning-certain-items-largest-retailer-fix-issue-documents-acknowledge-breakdown-computer-net-information-detail
[30] S. Moss, "It outage at uk supermarket sainsbury's cancels "vast majority" of online orders," Mar 2024, accessed 25-October-2024. [Online]. Available: https://www.datacenterdynamics.com/en/news/it-outage-at-uk-supermarket-sainsburys-cancels-vast-majority-of-online-orders/
[31] M. Boyle, "A $50 million glitch? target takes a hit from register outage," May 2024, accessed 25-October-2024. [Online]. Available: https://www.datacenterknowledge.com/outages/a-50-million-glitch-target-takes-a-hit-from-register-outage