Arrow Project

Final Report

Author:
David Méndez Giménez, Jesus Daniel Azuaje, Noa Paguera Contelles,
Lucas Mira, Eva Alisson Roda Rivas, Gabriel Iniesta,
Alejandro Martínez Torre

Advisors:
Josep Pegueroles Valles
Beatriz Otero Calviño

June 12, 2025

Revision History and Approval Record

Revision	Date	Purpose
0	14/05/2025	Document creation
1	31/06/2025	Started documentation
2	06/06/2025	Group review
3	12/06/2025	Google Docs to Overleaf

Document Distribution List

Name	Email
David Mendez Gimenez	david.mendez.gimenez@estudiantat.upc.edu
Jesus Daniel Azuaje	jesus.daniel.azuaje@estudiantat.upc.edu
Noa Paguera Contelles	noa.paguera@estudiantat.upc.edu
Lucas Mira	lucas.mira@estudiantat.upc.edu
Eva Alisson Roda Rivas	eva.alisson.roda@estudiantat.upc.edu
Gabriel Iniesta	gabriel.iniesta@estudiantat.upc.edu
Alejandro Martínez Torre	alejandro.martinez.torre@estudiantat.upc.edu
Beatriz Otero Calviño	botero@ac.upc.edu
Josep Pegueroles Valles	josep.pegueroles@upc.edu

Written by		**Reviewer and approved by**	
David Mendez, Jesus Daniel Azuaje, Noa Paguera, Lucas Mira, Gabriel Iniesta, Alejandro Martínez, Eva Roda **Date 12/06/2025**		David Mendez, Jesus Daniel Azuaje, Noa Paguera, Lucas Mira, Gabriel Iniesta, Alejandro Martínez, Eva Roda **Date 12/06/2025**	
Name	Eva Roda	**Name**	David Mendez
Position	Document Responsible	**Position**	Project Leader

Contents

1 Document Scope **6**

2 Project Summary **7**

3 Time Plan Updated **8**

4 System Design documentation **10**
4.1 **Functional Overview** . . . 10
4.2 **Detailed design by Work Package** . . . 11
4.2.1 WP1 - Traffic Capture . . . 11
4.2.2 WP2 – Preprocessing and Analysis . . . 11
4.2.3 WP3 – Classification . . . 12
4.2.4 WP4 – Network Environment . . . 12
4.2.5 WP5 – Encryption and Authentication . . . 13
4.2.6 WP6 – User Interface . . . 13
4.2.7 WP7 – Response and Mitigation . . . 14
4.2.8 WP8 – Post-Incident Analysis . . . 14

5 System Implementation documentation **16**
5.1 Final system architecture and final schematics . . . 16
5.2 Encryption and Authentication . . . 17
5.3 Network Environment . . . 18
5.4 Packet Capture . . . 19
5.5 Classification . . . 19
5.6 Mitigation . . . 20
5.7 Reports & User interface . . . 21

6 System Characterization **24**

7 Costs **26**

8 Business Model Canvas **28**

9 IP Considerations **33**
9.1 Abridged patent search results . . . 33
9.2 Invention disclosure document . . . 35

10 Sustainability Analysis **37**
10.1 Functional Value . . . 37
10.2 Materials . . . 37
10.3 Production . . . 38

10.4 Supplies & Outsourcing . . . 38
10.5 Distribution . . . 38
10.6 Use Phase . . . 38
10.7 End-of-Life . . . 39
10.8 Environmental Impacts . . . 39
10.9 Environmental Benefits . . . 39
10.10Critical reflection . . . 40

11 Ethical Implications **41**

12 Conclusions **42**

13 Reflection Document **43**

14 Annex **44**
14.1 Packet Capture code . . . 44
14.2 Classification code . . . 46
14.3 Mitigation code . . . 51
14.4 Encryption and Authentication code . . . 52
14.5 Reports: User Interface code . . . 57
14.6 Reports: Post-Incident Analysis code . . . 61
14.7 Cyber Resilience Act Table . . . 72

List of Figures

1 Time Plan calendar 8
2 WP Descriptions 9
3 General functional overview 10
4 Classification diagram 12
5 Final diagram 16
6 Final diagram - Work Packages overview 16
7 Direct and Indirect costs diagram 27
8 Persona Canvas: Lucy 28
9 Persona Canvas: the Hospital 28
10 Value Proposition Canvas 31
11 Business Model Canvas 32
12 Alert transmission - Patent 33
13 Enpoint detection and reponse with ML - Patent 34
14 IDRS system with Machine Learning drawing 36

List of Tables

1 Components List 26
2 Components Amortization Table 26
3 Salary breakdown by task and role 26
4 Annual facilities, equipment and license costs 27
5 Direct and indirect costs 27
6 Environmental Impacts 39
7 Critical reflection 40
8 Table of CRA requirements with brief descriptions 73

1. Document Scope

The goal of this document is to present to the reader the work done by the group project **ARROW**. It provides a comprehensive overview of the project's scope, objectives, deliverables, and outcomes, with a focus on the implementation of a cybersecurity interface aligned with the requirements of the **Cyber Resilience Act (CRA)** (see Annex 14.7).

This document is distinct from technical documentation or development logs, as it focuses on the overall project management, including planning, execution, team coordination, and quality assurance. It also reflects on the challenges faced, lessons learned, and future recommendations.

By offering both a strategic and operational perspective, this document serves as a final project report intended for academic evaluation, stakeholder review, and documentation of the methodology and results achieved throughout the lifecycle of the project.

2. Project Summary

This project has developed a **cybersecurity management and monitoring interface** that provides real-time threat detection, incident response, and compliance reporting. The system is designed to enhance the security of critical infrastructures by integrating **artificial intelligence for threat detection, encryption mechanisms for data protection, and compliance tools aligned with industry regulations.**

Scope Inclusions

The project will include the following key components:

1. **Backend Development:** A secure and scalable infrastructure for processing and storing security-related data.
2. **Threat Detection and Response:** AI-driven threat analysis and automated alerts.
3. **Incident Management and Reporting:** Tools for forensic analysis, attack visualization, and regulatory compliance reporting (e.g., GDPR).
4. **Data Encryption and Protection:** Implementation of AES-256 and RSA-2048 encryption for data confidentiality and integrity.
5. **Web-Based Interface:** A user-friendly dashboard for monitoring, managing security policies, and responding to threats in real-time.

Scope Exclusions

To maintain focus and feasibility, the project will **not** include:

- Development of a proprietary machine learning model from scratch (pre-existing models will be adapted).
- Hardware development or manufacturing (the system will integrate with existing infrastructure).
- Direct end-user support (the system will be designed for enterprise IT and security teams).
- Physical security solutions (e.g., biometric authentication or hardware firewalls).

By clearly defining what is and isn't included in the project, this scope ensures alignment between stakeholders and the project team while preventing scope creep. The Project Manager will oversee adherence to these boundaries and address any scope adjustments through formal change management processes.

3. Time Plan Updated

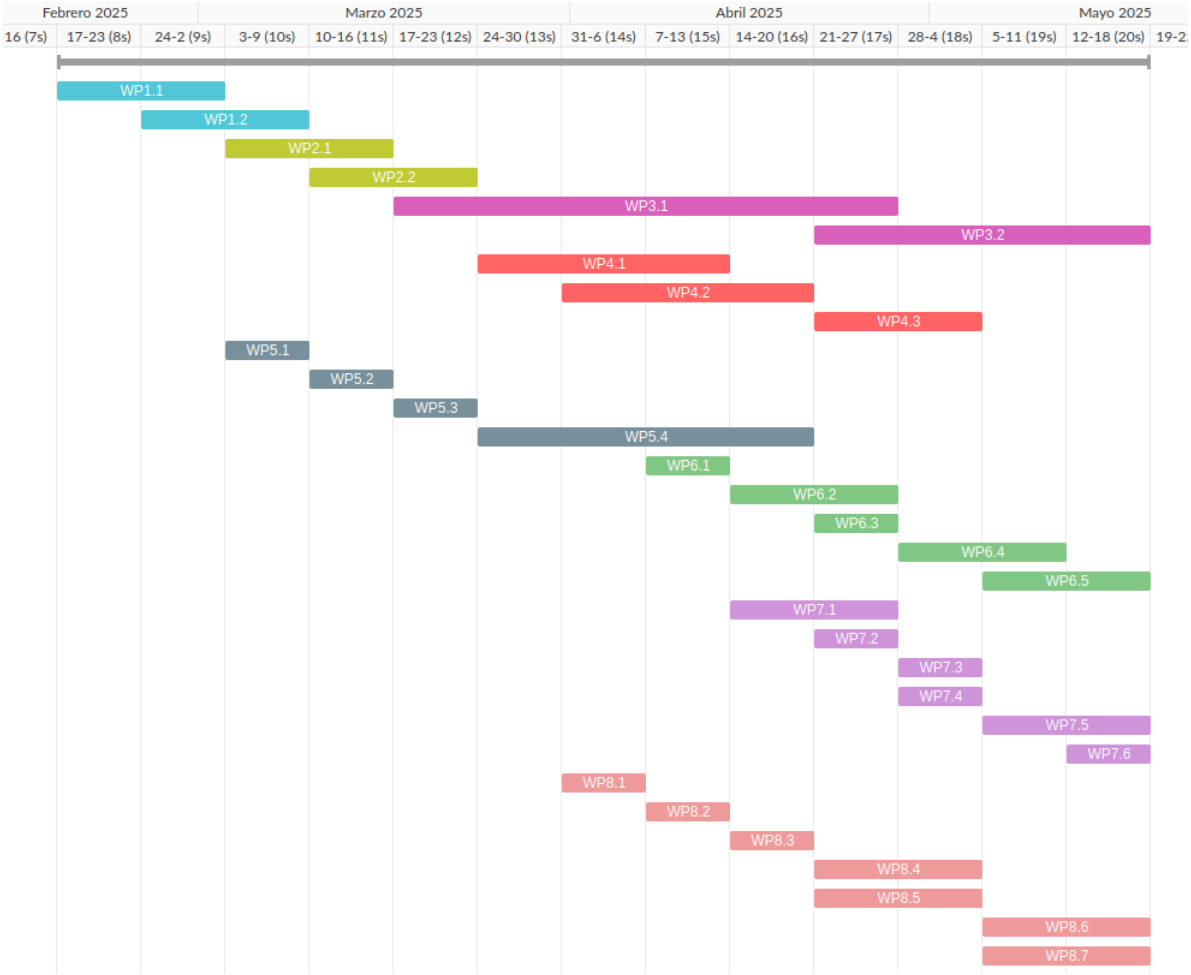

Figure 1: Time Plan calendar

The Machine Learning work package was more time-consuming than originally foreseen due to the complexity of integrating the classification model.
During the course of development, we encountered unforeseen technical issues concerning the preparation of datasets and challenges in grasping key Machine Learning concepts and their application, which required extra time for proper implementation. Additional time was also spent to ensure the model could be deployed effectively on the Raspberry Pi and that its outputs were correct and reliable. The timeline for this work package was therefore adjusted to meet these technical requirements without compromising quality.

WP1. Traffic capture
WP1.1: Dummy capture: try to see one frame captured from the desired interface.
WP1.2: Evaluating management of length of capture blocks related to the live capture, conversion rate and dataset input.
WP2. Preprocessing and Analysis
WP2.1: Test on terminal of .pcap to .csv conversion using tshark '-r' option to read file.
WP2.2: Python implementation using 'subprocess' library.
WP3: Classification
WP3.1: Initial Design of the Machine Learning System.
WP3.2: Model completion.
WP4. Network Environment
WP4.1: Configure the Raspberry Pi to act as an IoT device within a network with active traffic.
WP4.2: Set up an access point to create a private Wi-Fi network for controlled cyber-attack simulations.
WP4.3: Perform security tests to evaluate the effectiveness of the detection system.
WP5. Encryption and Authentication
WP5.1: Implementation of AES-256 and RSA-2048 on the Raspberry Pi.
WP5.2: Integration of X.509 certificates and authentication via digital signatures.
WP5.3: Application of encryption to network communication (TCP/IP).
WP5.4: Security testing using Kali Linux to assess encryption strength and authentication robustness.
WP6. User Interface
WP6.1: Investigate the necessary tools for implementation.
WP6.2: Creation of the basic interface.
WP6.3: Enhance the basic interface with the option to integrate graphs.
WP6.4: Integrate with the user's real-time results.
WP6.5: Implement the WP8 results for report generation.
WP7. Response and Mitigation
WP7.1: Investigate and select the appropriate security tools to automate response.
WP7.2: Develop a real-time detection system for malicious behaviors.
WP7.3: Configure and program automatic containment measures for identified attacks (e.g., IP blocking, endpoint isolation).
WP7.4: Design and implement an automated recovery system to restore affected systems to a secure state.
WP7.5: Integrate the response and mitigation system with existing monitoring platforms.
WP7.6: Conduct tests with attack scenarios to assess the effectiveness of the automated response.
WP8. Post-Incident Analysis
WP8.1: Analyze the available data (e.g., pre-generated logs).
WP8.2: Define the key metrics and events to extract for analysis.
WP8.3: Begin structuring the automatic report generation.
WP8.4: Develop the extraction of patterns (event correlation, incident timeline).
WP8.5: Define the temporal event graph.
WP8.6: Integrate with logs from other teams and test automatic report generation.
WP8.7: Integrate with the user interface.

Figure 2: WP Descriptions

4. System Design documentation

4.1. Functional Overview

From the outset, our aim was to develop an IDRS that:

- Continuously captures IoT network traffic on a Raspberry Pi (**WP1 & WP4**).
- Converts captured PCAP files to CSV format and preprocesses features (**WP2**).
- Applies a machine-learning model in real time to classify packets as benign or malicious (**WP3**).
- Encrypts all data flows end-to-end with AES-256 and authenticates using RSA-2048/X.509 (**WP5**).
- Automatically contains malicious traffic by blocking or isolating affected devices (**WP7**).
- Presents live results via a web-based interface (**WP6**).
- Generates a fully formatted, automated report at the click of a button (**WP8**).

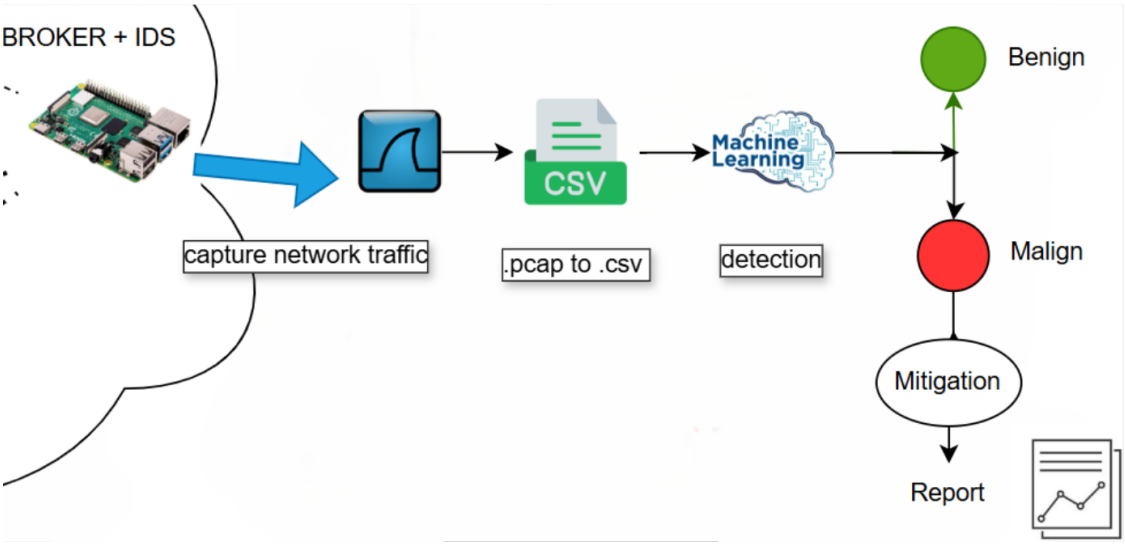

Figure 3: General functional overview

By integrating these eight components, our design was intended to satisfy the key pillars of the Cyber Resilience Act (see the CRA Table in Annex 14.7).

4.2. Detailed design by Work Package

4.2.1. WP1 - Traffic Capture

Objective: The aim of WP1 is to capture live network traffic from connected IoT medical devices using lightweight hardware. This traffic serves as the basis for intrusion detection and is the first critical step in the data processing pipeline.

Tools and design: We used Python along with the pyshark library to implement real-time packet sniffing. The Raspberry Pi served as the host platform, ensuring realistic deployment within constrained environments.Captured traffic is stored in .pcap format and later converted into .csv files for analysis (handled in WP2).

Initial data fields captured:

- Inter-arrival time
- Source and destination IP and port
- Protocol used (TCP, UDP, etc.)
- TCP flags

These fields were selected to preserve relevant context for both attack signature analysis and anomaly detection.

4.2.2. WP2 – Preprocessing and Analysis

Objective: The goal of WP2 was to transform raw .pcap data into structured .csv datasets and clean the data for use in machine learning.

Steps taken:[noitemsep]

1. Use of pyshark and pandas to convert .pcap into .csv.
2. Removal of unnecessary or null features.
3. Encoding of categorical features (e.g., protocol type) and normalization of numerical data.
4. Generation of balanced datasets for supervised learning using both benign and malicious traffic samples.

Comparison with industry practice: While traditional systems might rely on static feature extraction, our pipeline is fully automated and adapts to new fields or formats, increasing flexibility.

4.2.3. WP3 – Classification

The primary objective of this work package is to develop and establish a machine learning-classification system that will be able to examine network traffic data in the .csv files generated by the previous work package. The system should provide malicious and benign traffic identification with high accuracy, and accomplish the overall objective of enhancing cybersecurity through automation of threat detection.

To enable practical deployment, particularly on Internet of Medical Things (IoMT) devices, the solution must be lightweight, scalable, and reliable and be able to execute on devices with low power processing.

The final product of this work package is an operational classification system that classifies network traffic in real time, enabling the detection of potential attacks and the triggering of appropriate security responses.

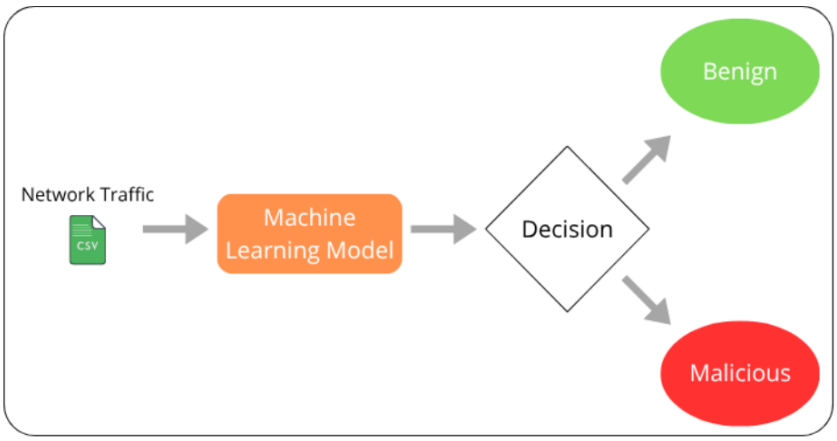

Figure 4: Classification diagram

4.2.4. WP4 – Network Environment

Objective: The goal of WP4 is to design a network environment that simulates a real hospital room with IoT medical devices connected in a private network. The environment must allow monitoring, traffic capture, and intrusion detection.

Steps taken: Defined the network topology with medical devices (Arduino + ECG), a private hospital network, and a Raspberry Pi running the IDS.

- Simulated the network using MQTT protocol for device communication.
- Configured the Raspberry Pi to capture mirrored traffic, store it as .pcap, and analyze it in real time (WP1-WP3 flow).
- Designed the environment to support response actions such as blocking malicious IP addresses (linked to WP7).

- Ensured the architecture reflects realistic hospital conditions, supporting future testing of encryption (WP5) and reporting (WP6).

Comparison with industry practice:
In typical hospital networks, segmentation and traffic monitoring are often not implemented for IoT medical devices. Our design enforces traffic control and real-time monitoring, increasing network resilience.

4.2.5. WP5 – Encryption and Authentication

Objective: Design and implement secure communication between a client and a server using AES-256 for data confidentiality and RSA-2048 for secure key exchange. The system will include a proxy component to simulate real-world intermediate networks.

Main tasks:

- WP5.1: Define cryptographic architecture (AES + RSA)
- WP5.2: Generate key pairs and exchange mechanisms
- WP5.3: Implement secure handshake protocol
- WP5.4: Develop socket-based client-server architecture
- WP5.5: Integrate transparent proxy and validate that data remains encrypted
- WP5.6: Test and document the security of the communication

Resources:

- Tester
- Raspberry Pi (optional)
- Python, OpenSSL libraries

Deliverables:

- Working prototype of secure client-server communication
- Technical documentation of cryptographic implementation
- Proxy test report confirming end-to-end encryption

4.2.6. WP6 – User Interface

Objective: Design and develop a lightweight, secure, and intuitive web interface that allows authorized users to interact with the intrusion detection and response system (IDRS). The interface will support real-time log monitoring, report generation, and alert visualization, with a focus on usability and deployment in low-resource healthcare environments. It will also integrate results from other work packages (notably WP8) to enhance reporting capabilities.

Main tasks:

- WP6.1: Investigate the necessary tools for implementation
- WP6.2: Create the basic interface
- WP6.3: Enhance the basic interface with the option to integrate graphs
- WP6.4: Integrate with the user's real-time results
- WP6.5: Implement the WP8 results for report generation

Resources:

- Python 3.10+
- Streamlit
- sounddevice, soundfile, matplotlib or plotly, reportlab or fpdf2

Deliverables:

- Fully functional and tested web-based user interface
- Secure login and session management system
- Real-time log viewer with IP filtering and audio alerts
- Post-incident reporting module integrated with WP8 outputs

4.2.7. WP7 – Response and Mitigation

Objective: To automate the response to detected threats in real time, reducing human intervention and mitigating impact on hospital operations.

Implementation details:

- A script monitors the folder of classified malicious files.
- Reads each .csv, extracts source IPs.
- Matches against a whitelist.txt to avoid false positives.
- Applies iptables rules to block IPs not on the whitelist.
- Logs all blocked IPs for auditing.

Comparison with similar systems: Compared to commercial systems which may require human approval for blocking, our system executes autonomously and is script-driven, offering faster reaction time.

4.2.8. WP8 – Post-Incident Analysis

Objective: Automated system for post-incident analysis: visualizing the timeline, performing root cause analysis, and tracking detection and mitigation.

Main tasks:

- WP8.1: Analyze the available data (e.g., pregenerated logs).
- WP8.2: Define the key metrics and events to extract for analysis.
- WP8.3: Begin structuring the automatic report generation.
- WP8.4: Develop the extraction of patterns (event correlation, incident timeline).
- WP8.5: Define the temporal event graph.
- WP8.6: Integrate with logs from other teams and test automatic report generation.
- WP8.7: Integrate with the user interface.

Resources:

- Python 3.13 with pandas, matplotlib, seaborn
- MiKTeX program, it will generate PDF using the pdflatex command

Deliverables:

- Report structure
- Report example (pdf)
- UI image integration

5. System Implementation documentation

The final IDRS prototype integrates five core functional blocks—Packet Capture, Classification, Mitigation, Encryption & Authentication, and Reporting—corresponding to the Work Packages outlined in Section 4.

5.1. Final system architecture and final schematics

In our final architecture, all traffic—benign or malicious—is secured with AES-256 encryption and authenticated via RSA-2048/X.509 (WP5). The final system architecture is shown in the following diagram.

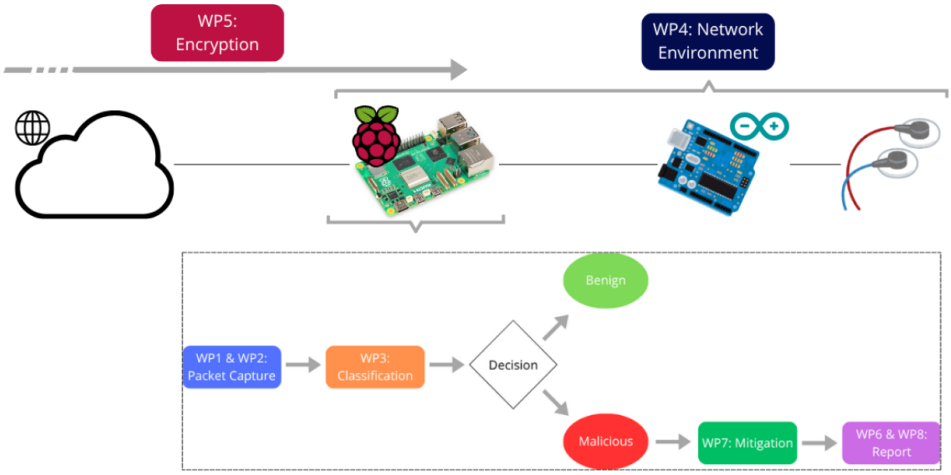

Figure 5: Final diagram

We can identify five distinct blocks, each serving a specific function. The breakdown of these blocks is illustrated below:

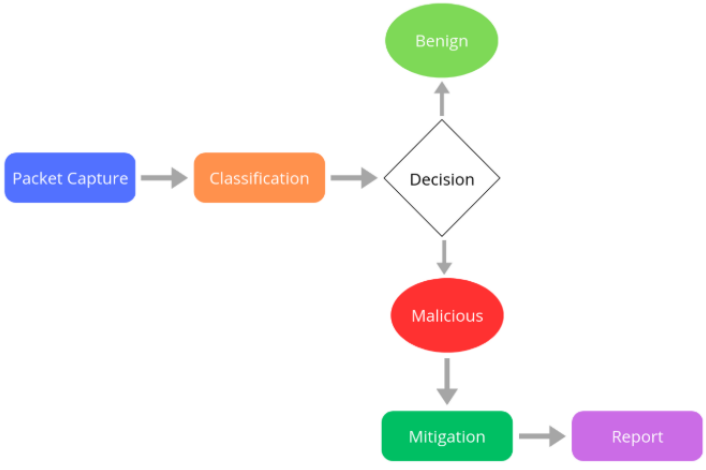

Figure 6: Final diagram - Work Packages overview

5.2. Encryption and Authentication

The implemented system establishes a secure communication channel between a client and a server using a hybrid encryption approach, combining **RSA-2048** for key exchange and **AES-256** for symmetric encryption of data. The architecture includes a **TCP proxy**, positioned between the client and server, to simulate a realistic network scenario in which data traverses intermediate nodes that may intercept or manipulate traffic.

→ Key Generation and Exchange

The server generates a 2048-bit RSA key pair (public and private). The public key is either pre-distributed or requested by the client during the initial phase. The client, upon establishing a connection, generates a random 256-bit AES key and a random initialization vector (IV). These values are encrypted using the server's RSA public key and sent to the server. The server decrypts them using its private key, thus securely retrieving the shared AES key and IV without exposing them during transmission.

→ Secure Data Transmission

Once the shared AES parameters are established, both client and server use **AES-256 in CBC mode** for all subsequent message encryption and decryption. All messages exchanged are encrypted at the source and decrypted at the destination, ensuring that only the intended recipient can access the original data.

→ Proxy Integration

A TCP proxy server is implemented to relay all traffic between the client and the server. It sits in the communication path and forwards all incoming and outgoing messages, serving as an intermediary layer that reflects real-world network conditions, such as potential inspection or modification attempts during transit

→ Security Measures

The RSA encryption uses **PKCS1_OAEP padding** to prevent padding oracle attacks. Random IVs are generated for each session to enhance security and avoid patterns in ciphertexts. Additionally, logging mechanisms are implemented to record connection attempts and encryption activity for debugging and future auditing purposes

→ Technology Stack

The entire system is developed in **Python**, using the `socket` library for network communication and the `cryptography` library for RSA and AES operations. The solution is platform-independent and tested on Linux environments, including scenarios with Raspberry Pi devices acting as client or server nodes.

5.3. Network Environment

The network environment was designed to emulate a hospital room setup where medical IoT devices continuously monitor patient vital signs and transmit the data through a private and secure local network. This emulation allows us to test both the communication between devices and the behavior of our intrusion detection system under realistic conditions.

To begin with, a private Wi-Fi network was created using an access point, which acts as the backbone for all communications in the simulated room. All components — both clients and server — are connected through this isolated network to ensure a controlled and secure environment.

We then configured an Arduino Uno equipped with an ECG (electrocardiogram) sensor. This hardware represents a typical medical monitoring device used in hospitals. The Arduino is programmed to measure ECG signals and send the readings using an MQTT client library over the private network. The ECG readings are published to a specific MQTT topic at regular intervals.

The Raspberry Pi 4 plays a dual role in the system architecture:

- First, it is configured as an MQTT broker using Mosquitto, an open-source MQTT broker. It receives data from the Arduino (MQTT client) by subscribing to the appropriate topic.
- Second, the Raspberry Pi runs the Intrusion Detection System (IDS), which monitors all traffic on the network interface (`wlan0`) and processes it through the full pipeline: packet capture, preprocessing, classification, and response.

All devices are connected to the same local network and communicate using the MQTT protocol, which is widely adopted in IoT environments for its low overhead and efficiency. MQTT's publish/subscribe architecture fits well in our case, where the sensors publish data and the broker (Raspberry Pi) handles subscriptions and message delivery.

To simulate real-world threats, we introduced malicious traffic using tools available in Kali Linux, which acts as the attacker node inside the same Wi-Fi network. This allows for realistic testing of our IDS and mitigation mechanisms against threats like Denial-of-Service (DoS) and flooding attacks.

The network architecture is intentionally simple and representative:

- Clients: Arduino Uno with ECG sensor (MQTT publisher)
- Broker + IDS: Raspberry Pi (MQTT Mosquitto broker, sniffer, classifier, mitigator)
- Attacker: Kali Linux device sending crafted packets

This configuration replicates the communication between medical IoT devices and a central server or gateway, while also allowing the Raspberry Pi to inspect and analyze traffic for anomalies or known attack patterns. It serves as a faithful model of how such technologies would interact in an actual hospital setting and enables testing of response workflows in a safe, controlled, and repeatable manner.

5.4. Packet Capture

The packet capture subsystem consists of Python scripts running on a Raspberry Pi that continuously sniff traffic from the `wlan0` interface.
Captured data is saved in `.pcap` format and then transformed into `.csv` for use in later stages.
The script uses `pyshark` for capture and `pandas` for formatting. This design ensures high compatibility and low overhead.

Diagram and Tools:

- Interface: `wlan0`
- Output format: `.csv`
- Technologies: `pyshark`, `pandas`
- Host: Raspberry Pi 4

This modular capture system simplifies integration with various ML models or detection rules and enables replay or inspection.

5.5. Classification

To implement the classification system, we employed the Random Forest machine learning algorithm due to its strong robustness to noise and usual incomplete data characteristic of real network traffic. It also has low resource consumption and it works well without the need for complex hyperparameter optimization, which was an ideal choice given the extent and limitation of our project.

We used Python as our development language since it is used most widely in data science and machine learning due to its simplicity, rich ecosystem, and large community support base. We used tools such as Pandas and NumPy to manipulate and preprocess the datasets so that useful features could be extracted. For actually building the classification model, we relied on scikit-learn because of its complete set of ML tools, good integration, and thorough Random Forest implementation.

The building of the classification system began with the generation of the training datasets. We built them by combining benign network traffic with the malicious traffic that was generated through the use of the Kali Linux operating system and its native penetration

testing tools. All of the samples in the dataset had been labeled appropriately as either benign or malicious in order to facilitate supervised learning.

Having preprocessed the dataset, we applied the Random Forest model to learn each category of traffic patterns. We applied the trained model on a platform of Raspberry Pi, where it classifies real-time network traffic. In case a dataset is marked as malicious, it is kept in a folder for further analysis and correlation with security incidents. Similarly, benign traffic is retained independently to facilitate logging, assessment, and future model refinement if needed.

5.6. Mitigation

Mitigation in this system is conceived as an automated, lightweight, and modular component capable of reacting to threats in real time. Upon classifying traffic as malicious, the system triggers a chain of actions designed to isolate the threat while maintaining operational stability in the network.

Mechanism Overview: Once a file is classified as malicious and saved into the corresponding directory, a continuously running Python daemon detects the new entry. This daemon then parses the file for unique identifiers—particularly source IP addresses. These are checked against a maintained whitelist to prevent the blocking of internal or trusted addresses.

Blocking Protocol: Untrusted IPs are blocked at the OS level using iptables, a proven Linux utility for managing network packet filtering and NAT. Each rule is logged, timestamped, and stored for later inspection. The blocked IP list is persistent across reboots and can be manually overridden if needed.

System Efficiency and Load Considerations: Unlike heavier response systems that may incorporate agent-based quarantine or require human-in-the-loop validation, our approach is fully script-based. This makes it viable for resource-constrained environments like Raspberry Pi, where performance must be balanced against reliability.

Security and Robustness: To enhance security, the mitigation component includes checks for spoofed IPs using TTL analysis and optionally integrates with an external threat intelligence feed to validate the origin and threat level of an IP before blocking it. These extensions make the system adaptable to both known and unknown attack vectors.

Comparison with Related Approaches: Traditional NIDS/NIPS frameworks such as Snort or Suricata often rely on signature-based detection followed by human approval or static rules. Our model acts faster by combining machine learning inference with direct mitigation, minimizing response time. While commercial systems offer GUI dashboards

and SIEM integrations, they often require heavier infrastructure.

The mitigation process, as designed, is practical, efficient, and scalable. It closes the feedback loop between detection and response without requiring external dependencies. This level of autonomy is especially valuable in high-stakes environments like healthcare, where immediate isolation of threats can prevent cascading failures or privacy violations.

5.7. Reports & User interface

To support efficient and secure monitoring of the device performance and detection , a custom web-based interface was developed using the Streamlit framework. This user interface serves as a front end for interacting with a simplified Intrusion Detection System (IDS), allowing authorized users to view logs, filter security events, and generate post-incident reports. The design prioritizes usability, clarity, and accessibility, especially for healthcare institutions with limited technical resources.
The interface consists of three main sections, accessible through a sidebar navigation menu:

1. Home
2. Log Viewer
3. Post-Incident Report

These views are complemented by a secure login system and alert mechanisms to provide a secure experience.

→ Login Page:
The first screen presented to the user is the Login page, which acts as the system's first layer of protection. It ensures that only authorized personnel can access the log data and post-incident reporting features. The login form includes:

- A username field.
- A password field (with masking for security).
- A Login button that authenticates the user.

If incorrect credentials are entered, the system displays an error message. Once authenticated users gain access to the application dashboard.

→ Home View:
The Home page serves as the initial screen after login. While currently minimalist in content, it:

- Confirms successful login.

- Provides a general introduction to the purpose of the application: real-time security monitoring for medical devices.
- Serves as a potential location for future dashboard components (e.g., statistics, device summaries, attack trends).

This page represents a clean and neutral starting point, guiding the user towards the core functionalities.

→ Log Viewer

The Log Viewer is the central feature of the interface, designed for interactive analysis of system log entries. It allows users to explore the activity recorded by the IDS and react promptly to suspicious behavior. The view includes:

- Filter inputs for source and destination IP addresses, enabling precise targeting of logs related to specific devices or endpoints.
- A dropdown menu to choose how many log entries to display, from 5 to 1000 or all available logs.
- Color-coded log entries to improve readability:
  - Red for critical errors.
  - Dark green for warnings.
  - Black for neutral entries.

These visual distinctions help analysts quickly detect serious issues without reading every line.

A unique feature is the automated audio alert system, triggered when certain keywords (like "error" or "warning") appear in the logs. A siren sound is played asynchronously for 45 seconds. The interface displays a warning while the alarm is active, and users may manually stop the sound using the "Stop Alarm" button.

Additional controls include:

- "Refresh Logs", which reloads log data.
- Real-time display of how many logs match the user's filter criteria.

Log entries are parsed using regular expressions and read from a text file (`warnings.txt`), allowing for flexible adaptation to various data sources, including simulated alerts or real system logs.

→ Post-Incident Report

This section enables downloadable Post-Incident Reports based on user-defined filters, fulfilling important compliance, forensic, and documentation purposes. The workflow is as follows:

- The user enters a source IP and/or destination IP to filter relevant incidents; They

can also narrow results by specifying a time range.
- The system invokes `procesamiento_datos_rango()` (see Annex 14.6.) where the system guarantees that only the relevant time-slice of data is included in the timeline, root-cause analysis, and final PDF
- If results are found, a "Download Post-Incident Report" button becomes available, triggering `generate_post_incident_report()` (see Annex 14.6.) to export a pregenerated PDF.

This functionality supports:

- Incident response teams documenting security breaches.
- Auditing processes required by legal and medical standards.
- Root cause analysis in post-mortem evaluations.

It transforms raw data into a format suitable for sharing with internal security teams or external authorities.

→ Conclusion

This user interface provides a practical and intuitive solution for managing and analyzing security events in medical environments. Its modular design—featuring a login system, real-time log viewer, and post-incident reporting tool—delivers essential capabilities for network monitoring and incident response. Furthermore, its minimal hardware requirements, streamlined controls, and accessible layout make it highly suitable for deployment in real-world healthcare scenarios, where simplicity and reliability are crucial.

6. System Characterization

The network architecture was designed to simulate a hospital room where a patient is connected to IoT medical devices (such as an ECG sensor). The architecture included:

- Several clients (Arduino with ECG sensor + additional sensors) connected to a WiFi network.
- Communication between clients was implemented using the MQTT protocol, where the Raspberry Pi acted as both MQTT broker and Intrusion Detection System (IDS).
- The Raspberry Pi captured all network traffic using a mirrored interface and processed it in real time.
- This setup reproduced a realistic Medical IoT scenario, with private network traffic simulating communication between patient-monitoring devices and a hospital server or control system.

The Raspberry Pi architecture enabled simultaneous traffic capture, ML-based classification, and response actions, providing a fully autonomous security pipeline.

Measurement graphs and results. Final performance
Starting with the Traffic Capture, it was demonstrated that it is feasible to deploy an effective and persistent traffic capture system in resource-constrained environments based on available hardware like the Raspberry Pi. Utilizing the pyshark library, the system can deduce high-fidelity .pcap streams of data and convert them into .csv format for analysis. The extracted fields of inter-arrival time, protocol, and IP headers were maintained with enough context to allow successful classification.

The classification efficacy was confirmed through its execution within a live monitoring system. The decision tree model worked consistently in detecting cases of attacks (such as Denial of Service, detected by abnormal inter-arrival times and source IP patterns) and forwarding them to the designated folder for future analysis and reporting. The model proved to be suitable for real-time use in resource-constrained settings, where quick classification is critical.

Regarding the encryption system, it can effectively establish secure communication between a client and a server through a hybrid system using RSA-2048 and AES-256. The handshake and message exchange were successfully tested, ensuring that data in transit was fully encrypted. Validation with Wireshark confirmed that no unencrypted private information was visible in the captured traffic.

In terms of mitigation, the system demonstrated the ability to react automatically to detected threats by applying iptables rules to block malicious IPs. The response time between classification and mitigation was very low, allowing rapid containment of attacks. The use of a whitelist, logging of blocked IPs, and script-driven automation provides a lightweight but effective real-time response capability, comparable to more advanced security systems but at a fraction of the complexity and cost.

Finally, the post-incident report module (based on `procesamiento_datos_rango()` and `generate_post_incident_report()`) merges preprocessed CSV logs, filters by timestamp or IP address, and generates a concise PDF report. The report includes an attack timeline, root-cause correlations, and aggregated log excerpts. This final step completes our end-to-end pipeline: capturing packets, classifying in real time, mitigating threats, and generating forensic reports that support compliance and audit requirements.

7. Costs

Components List with Approximate Costs (annual)
Includes Python libraries, encryption libraries and certificate fees.

Hardware	€/unit	#	Total (€)	Software	Total (€)
ArduinoUNO	40,00 €	2	80,00 €	Python Libraries	0,00 €
ArduinoNano	28,00 €	1	28,00 €	Encryption Libraries	0,00 €
Raspberry Pi 4	40,00 €	2	80,00 €	Encryption Certs	150,00 €
ECG sensor	18,00 €	1	18,00 €		
Access Point	100,00 €	1	100,00 €		
Laptop (amortized)	128,00 €	7	896,00 €		
		TOTAL	**1.202,00 €**		**150,00 €**

Table 1: Components List

Materials List	€/unit	Residual value (20%)	Anual amortization		Total
Laptop	800,00 €	160,00 €	128,00 €	7	896,00 €
				TOTAL	**896,00 €**

Table 2: Components Amortization Table

Design and Prototyping Labour Costs (annual) Labour costs are calculated as (number of people x hours per person x hourly salary x month)

- Salaries are based on gross wages (including 30% social security).
- "All staff" assumes mixed-role support at an average salary.

Task	Roles	#	€ / hour	Hours	Gross Wage	Social Security (30%)	Net Salary	Total Monthly	Total
Requirement analysis and IU interface	**Project Leader**	1	13,00 €	160	2.080,00 €	624,00 €	1.456,00 €	1.456,00 €	17.472,00 €
Encryption data	Engineer	1	12,00 €	160	1.920,00 €	576,00 €	1.344,00 €	1.344,00 €	16.128,00 €
Network Configuration and Traffic Capture	Engineer	2	12,00 €	160	1.920,00 €	576,00 €	1.344,00 €	2.688,00 €	32.256,00 €
ML for Detection and Mitigation	Engineer	2	12,00 €	160	1.920,00 €	576,00 €	1.344,00 €	2.688,00 €	32.256,00 €
Automated Report and documentation	Engineer	1	12,00 €	160	1.920,00 €	576,00 €	1.344,00 €	1.344,00 €	16.128,00 €
	TOTAL	**7**							**114.240,00 €**

Table 3: Salary breakdown by task and role

Facilities, equipment and license costs (annual)

Category	Description	Estimated cost
Infrastructure	Office/lab space rental	9.600,00€
Communications	Internet, mobile lines	1.200,00€
Maintenance and Support	Basic technical contingencies (hardware faults, spare items, etc.)	1.000,00€
Administration and Management	Project compliance, basic accounting	1.500,00€
Cloud and Storage (GitHub)	GitHub (free plan)	0,00€
	TOTAL	**13.300,00€**

Table 4: Annual facilities, equipment and license costs

Summary (annual costs)

Category	Cost
Employees	114.240,00 €
Hardware	1.202,00 €
Software	150,00 €
DIRECT COST (annually)	**115.592,00 €**
Infrastructure	9.600,00€
Communications	1.200,00€
Maintenance and Support	1.000,00€
Administration and Management	1.500,00€
Cloud and Storage (GitHub)	0,00€
INDIRECT COST (annually)	**13.300,00€**
TOTAL (annually)	**128.892,00 €**

Table 5: Direct and indirect costs

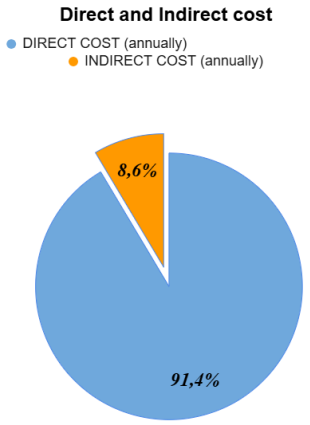

Figure 7: Direct and Indirect costs diagram

The annual budget is overwhelmingly driven by direct labor costs — employees account for roughly 91% of total spend — while hardware and software add only a small fraction. Indirect expenses (infrastructure, communications, maintenance, and administration) represent just under 9% of the overall budget. In other words, R&D effort (personnel) is the primary cost center, and keeping overhead low helps ensure most resources go directly toward development.

8. Business Model Canvas

Although hospital IT teams purchase and manage our solution, its value is measured in the patient experience. Take Lucy: she needs her ECG monitor to "just work", give clear status updates, and keep her data private.

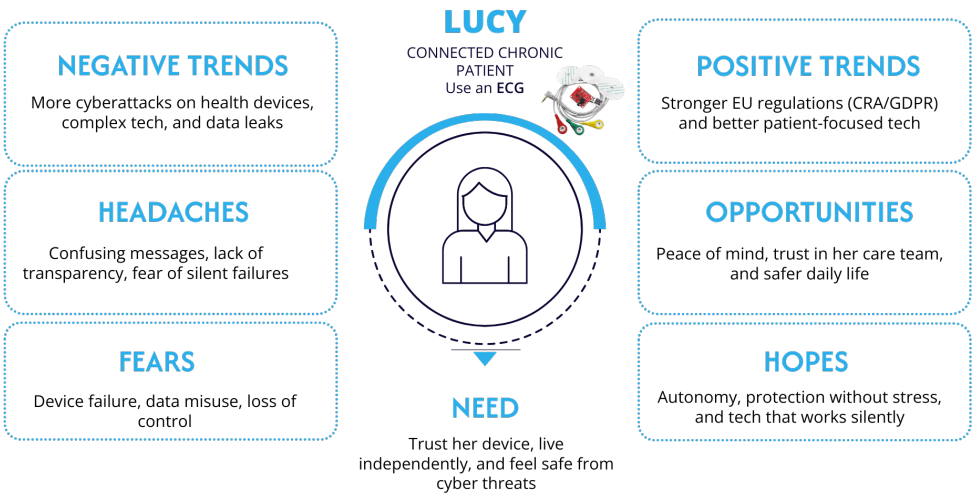

Figure 8: Persona Canvas: Lucy

However, IT's priorities are different: seamless implementation, predictable scaling, and hassle-free compliance.

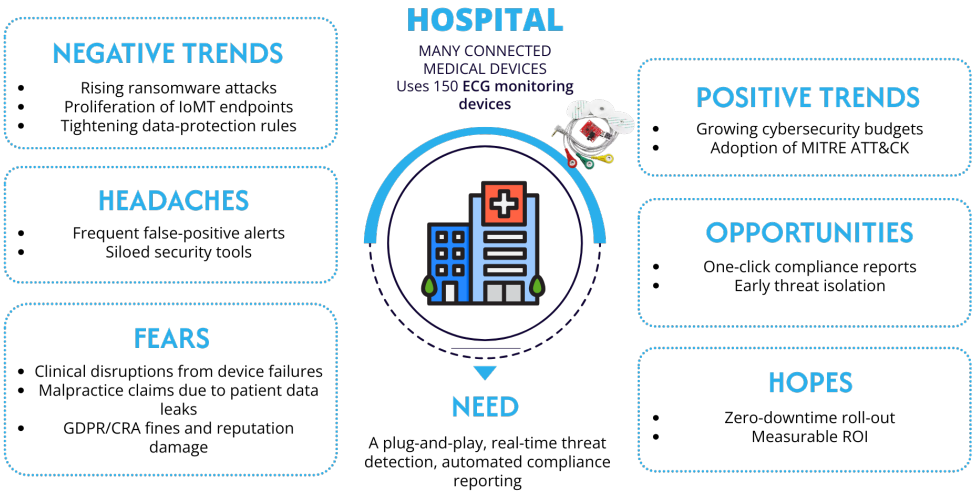

Figure 9: Persona Canvas: the Hospital

What do we offer? We offer a plug-and-play, hospital-grade IDRS with AES-256 encryption, real-time threat detection and fully automated, audit-ready CRA/GDPR reporting.

This combination delivers:

- **Patients** experience uninterrupted, reliable monitoring with guaranteed data confidentiality
- **Hospitals** achieve compliance at the click of a button, maintain continuous device uptime, and reduce legal and financial risk.

By aligning IT requirements with patient expectations, our IDRS ensures a secure compliant care environment that keeps both operations and patients' peace of mind intact.

Value Proposition Canvas

To show exactly how our IDRS features map to the hospital's core needs, pains and gains, we use a Value Proposition Canvas.

The Value Proposition Canvas (VPC) is a strategic tool used to design, visualize, and evaluate how a product or service creates value for a specific customer segment.
The VPC is divided into two main components:

- The **Customer Profile**, which explores what the customer wants to achieve (Jobs), what challenges they face (Pains), and the benefits they seek (Gains).
- The **Value Map**, which outlines the products and services offered, how they relieve customer pains (Pain Relievers), and how they create gains (Gain Creators).

By systematically comparing both sides, the canvas helps identify whether the proposed value proposition truly meets the real needs of the target users.
Now, let us dive into our Value Proposition Canvas.

Customer Profile

Customer Jobs

In the healthcare context, IT and cybersecurity managers have three primary responsibilities:

- Protect medical devices from cyber threats: Connected devices (such as infusion pumps, heart monitors, etc.) are vulnerable to cyberattacks. Preventing intrusions that could compromise their functionality or integrity is essential.

- Ensure the continuous operation of healthcare services: Any detection and response system must not disrupt clinical operations. System availability is critical, as any interruption can directly affect patient care.
- Comply with current legal and regulatory frameworks: Legislation such as the Cyber Resilience Act (CRA) and regulations on data privacy (e.g., GDPR) require proactive security measures, incident tracking, and demonstrable compliance.

Pains

The costumers face the following risks or also known as pains:

- Exposure of medical devices to cyberattacks: Many hospitals lack dedicated IDS/IPS systems, leaving critical devices vulnerable.
- High cost and complexity of security tools: Existing solutions often demand advanced resources (licenses, hardware, skilled staff), limiting their accessibility—especially in underfunded healthcare facilities.
- Compatibility issues with legacy systems: Hospitals frequently rely on outdated systems that are difficult to integrate with modern security technologies.
- False positives generated by AI: AI-based detection tools may trigger unnecessary alerts, overwhelming security teams and hindering their ability to respond to real threats effectively.

Gains

These are the positive outcomes the customer expects to achieve:

- Guaranteeing data privacy and protection: Securing medical records and patient data is essential for maintaining trust and meeting legal obligations.
- Compliance with the legal framework: The solution must support audits, documentation, and traceability of incidents in accordance with European and national cybersecurity regulations.
- Support for audit processes and compliance reporting: Customers seek tools that simplify regulatory documentation and audit preparation.

Value Proposition

Products and Services

- IDRS software for medical devices: A specialized Intrusion Detection and Response System designed to operate in clinical environments with existing devices.

- Compliance with CRA and related legal frameworks: The system includes mechanisms to ensure adherence to cybersecurity, data protection, and resilience standards.
- Open-source solution: Promotes transparency, adaptability, cost reduction (no license fees), and third-party auditability.

Pain Relievers

Our value proposition addresses customer pains through:

- Non-disruptive integration with medical operations: The system can be deployed without interrupting healthcare workflows or disabling connected devices.
- Operates on low-cost hardware: No need for major infrastructure upgrades, facilitating adoption in resource-constrained environments.
- AI-powered threat detection and mitigation: Automates monitoring and response, reducing reliance on human intervention while increasing efficiency.

Gain Creators

Our solution increases customer value by:

- Quick installation and updates: Saves time and accelerates deployment, even in hospitals with limited technical staff.
- Real-time incident alerts and reports: Enhances visibility into network and device status, enabling prompt response to threats.
- Training and support for staff: Accompanied by a training program to ensure proper use and maximize the effectiveness of the system.

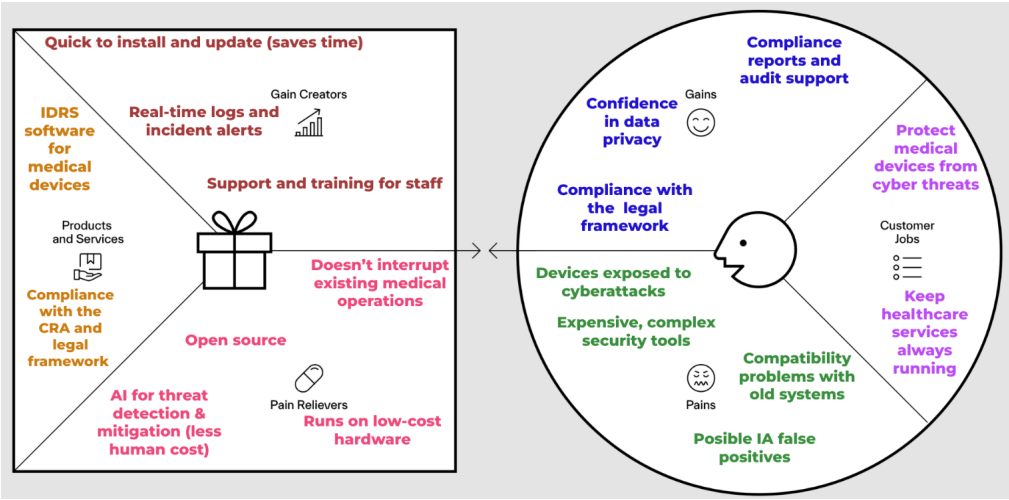

Figure 10: Value Proposition Canvas

Business Model Canvas

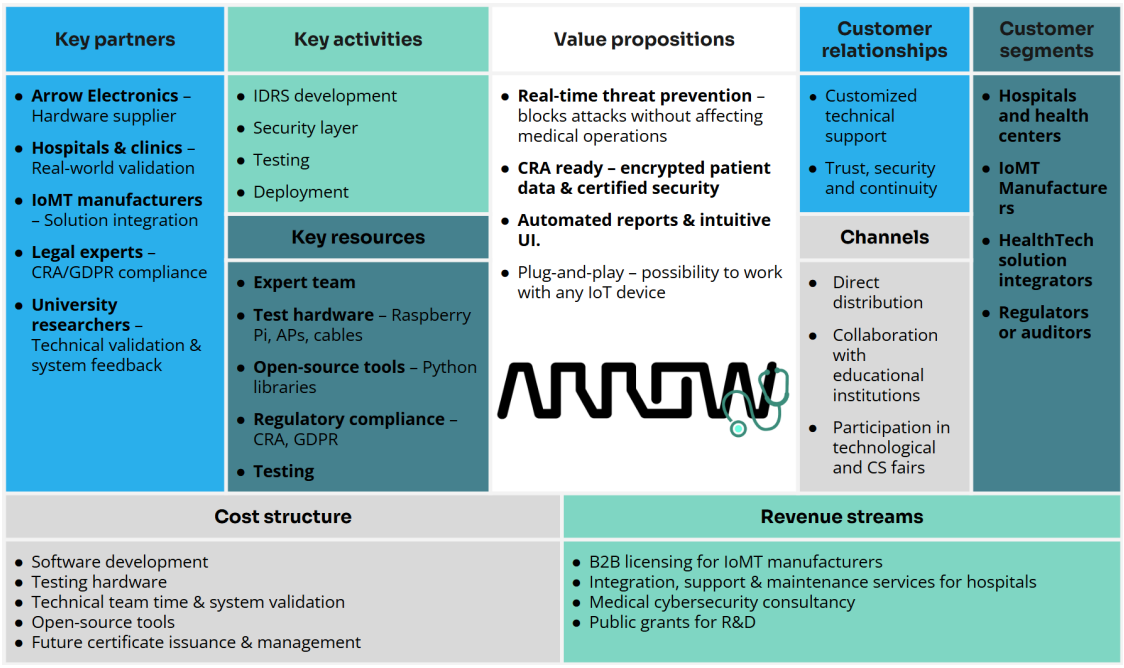

Figure 11: Business Model Canvas

Drawing on the needs and gains we identified in the patient and hospital persona canvases and the tailored offerings in our value proposition canvas, this business model canvas maps out the strategic partnerships, core activities, key resources, cost structure and revenue streams required to bring our encrypted, real-time IDRS to market at scale.

9. IP Considerations

Given the extensive patent landscape in IoT security, it is likely that elements of our design have precedents. Accordingly, we conducted a targeted search to identify three relevant patents, which informed our Invention Disclosure Document and underscore how our approach extends beyond existing work.

9.1. Abridged patent search results

We reviewed existing patents to find those most similar to our IDRS which combines AES-256 data encryption, real-time intrusion detection and response using machine learning and IU interface for reports. Three patents emerged as closest prior:

1. **Alert transmission device for policy based intrusion detection and response and method therefor**

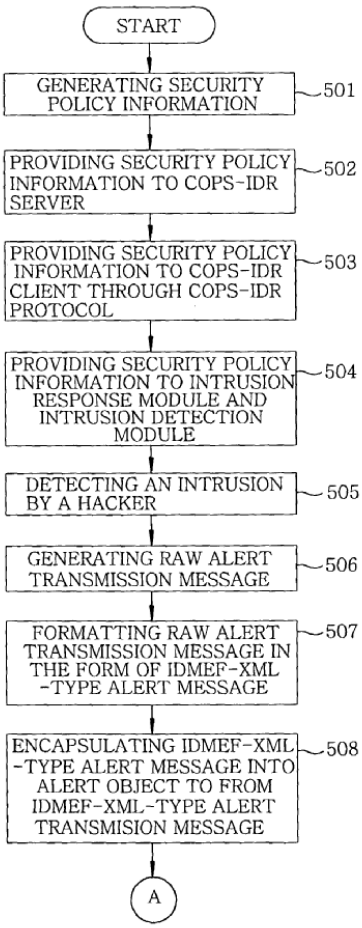

Figure 12: Alert transmission - Patent

- **Patent Number:** KR1020040038034, US20040088583
- **Priority date:** 31.10.2002, 30.05.2003
- **Assignee:** Electronics and Telecommunications Research Institute
- **URL:** https://patentscope.wipo.int/search/es/detail.jsf?docId=US40421170
- **Abstract:** A policy-based alert transmission system comprises a Central Policy Server (CPS) and an Intrusion Detection & Response System (IDRS). The CPS uses a policy management tool to generate and store security policies, then sends them via COPS-IDR to the IDRS. When the IDRS's intrusion detection module identifies a threat, its response module acts and an IDMEF-XML alert is built and sent back to the CPS. The CPS parses these alerts for storage in a database or display in an alert viewer.

2. **Endpoint detection and response utilizing machine learning**

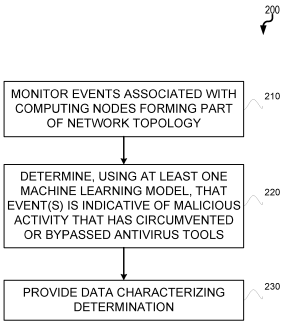

Figure 13: Enpoint detection and reponse with ML - Patent

- **Patent Number:** US 201762445172 P, US 201815862067
- **Priority date:** 11.01.2017, 04.01.2018
- **Assignee:** Cylance Inc
- **URL:** https://patents.google.com/patent/EP3568792B1/en?q=(EDR+machine+learning)&oq=EDR+with+machine+learning
- **Abstract:** This system continuously monitors events on multiple network nodes—even those protected by antivirus—and uses machine learning models to detect suspicious activity that evades traditional AV tools. When a potential threat is identified, it generates and provides detailed alert data (e.g., on-screen, stored, or transmitted). The system can pinpoint the malicious source node, automatically isolate it, and initiate remediation using reinforcement-learning strategies. Various ML techniques (supervised and unsupervised) adapt over time to recognize new attack patterns. Overall, it enables swift threat detection, isolation, and response to minimize data loss and maintain network security.

3. **Adaptive IOT systems using machine learning for enhanced data security**

** There is no main drawing attached to the patent **

- **Patent Number:** IN202511021932
- **Priority date:** 11.03.2025
- **Assignee:** Dr. Saurabh Sharma, Dr Praveen Kumar Mannepalli, Mr. Chitranjan Prasad Sah, Dr Parul Dutta
- **URL:** https://patentscope.wipo.int/search/es/detail.jsf?docId=IN453983640&_cid=P20-MAABUB-51054-1

- **Abstract:** A decentralized IoT security framework uses machine learning to detect and mitigate threats in real time across heterogeneous device networks. Blockchain ensures immutable logs, smart-contract–driven responses, and decentralized identity management. Federated learning lets devices train and share threat models without revealing raw data. The system adapts security policies dynamically and isolates compromised nodes automatically. Scalable and privacy-aware, it secures diverse IoT ecosystems against evolving attacks.

Comparing the three closest patents shows a shared focus on automated detection and response in different professional fields, but each misses key components of our design. None combines inlines encryption with even partial CRA compliance.

- **Alert transmission device** focuses implementation policy distribution and alert messaging via COPS-IDR without leveraging any ML classifier. It does not encrypt traffic on the wire, or generate compliance reports.
- **Endpoint detection and response using ML** focuses installing ML packs on endpoints for anomaly detection and isolation, but lacks network-level encryption and an integrated CRA reporting.
- **Adaptive IoT security with ML** focuses on collaborative machine learning across devices and blockchain-based tamper-proof with smart-contract-driven isolation, but it doesn't encrypt live packets or provide reports.

9.2. Invention disclosure document

Title: Penetration testing in Medical IoT devices for Cybersecurity Resilience Act compliance

Short description: We disclose a web-based cybersecurity platform that combines real-time, machine learning threat detection with inline AES-256/RSA-2048 encryption and fully automated compliance reporting. The system continuously ingests telemetry from critical infrastructure endpoints, applies pre-trained machine-learning models to detect anomalies, automatically isolates compromised assets, and generates audit-ready forensic and regulatory reports through a single dashboard. Administrators can manage policies, view live attack visualizations, and review encrypted logs without manual log aggregation or offline analysis.

In contrast to the three patents identified earlier, our solution integrates **real-time AI detection and response, end-to-end AES-256/RSA-2048 encryption, automated compliance reporting, and a scalable, web-based management console.**

Main claim: A method for securing and managing cybersecurity within critical-infrastructure

environments, comprising the steps of:

1. **Secure data ingestion and encryption:** Collecting security telemetry (network logs, endpoint events) from multiple devices and immediately encrypting sensitive data using AES-256 for data at rest and RSA-2048 for key exchange.

2. **Real-time AI-driven threat detection:** Applying pre-trained supervised and unsupervised machine-learning models to decrypted telemetry in real time to identify malicious behavior or anomalies.

3. **Automated incident response:** Upon detection of a threat, automatically isolating the compromised asset (e.g., quarantining network segment or disabling user credentials) without human intervention.

4. **Forensic logging and reporting:** consolidating encrypted logs, detection metadata, and response actions into a unified data store and generating a fully formatted, audit-ready compliance report (e.g., GDPR) with a single user action.

5. **Web-Based:** Presenting a user-friendly dashboard that allows administrators to monitor live threat metrics, adjust security policies, trigger manual overrides, and review historical forensic data in a secure, browser-based interface.

Main drawing:

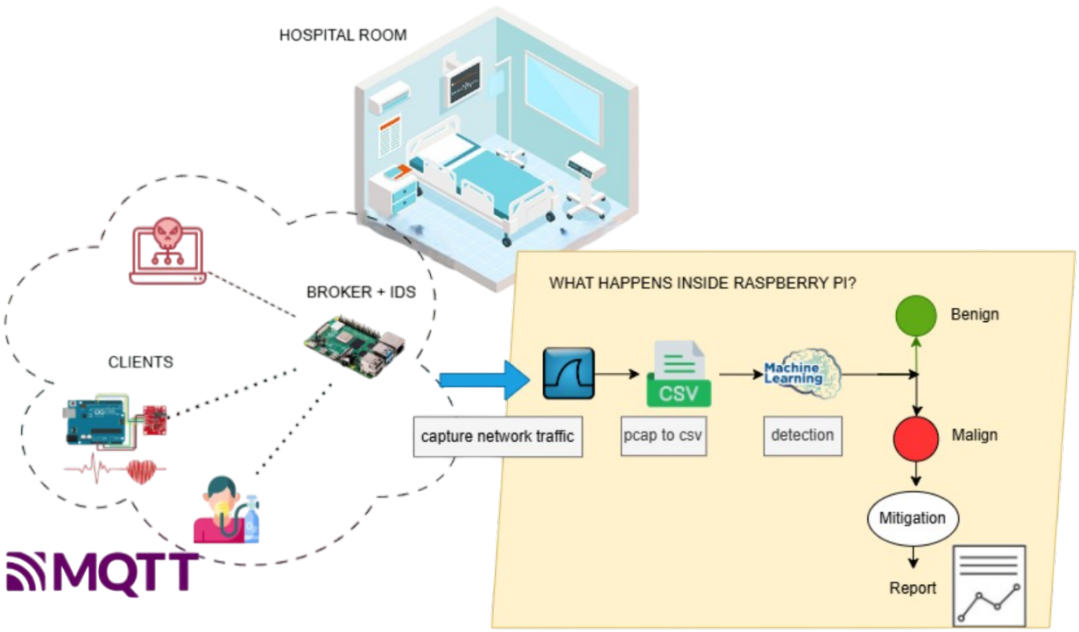

Figure 14: IDRS system with Machine Learning drawing

10. Sustainability Analysis

Introduction & Methodology

To critically assess the sustainability of our PAE project, we have followed the Environmental Business Model Canvas (EBMC) method and the guidance provided during the sustainability seminar. Our goal was not only to evaluate the environmental footprint of our solution, but also to consider its economic and social impacts throughout the entire product lifecycle, from development to end-of-life.
While the carbon footprint of our solution is relatively low, the true value of this report lies in the depth of our analysis, including both positive contributions and areas for improvement.

10.1. Functional Value

Purpose: Develop a lightweight IDRS system for use in connected medical devices (e.g., insulin pumps, heart monitors), with a lifecycle of ∼1–3 years.
Key Features: Real-time log monitoring, local report generation, GDPR and CRA compliance.
Energy Efficiency: Designed to run on Raspberry Pi or similar low-power hardware.
Sustainability Goal: Minimize unnecessary hardware consumption and enable modular, updatable cybersecurity solutions for healthcare.

- Benefit: Extends life of existing devices by protecting them from cyberattacks.
- Challenge: Cybersecurity systems may become obsolete quickly due to evolving threats, reducing long-term sustainability.

10.2. Materials

Components used:
Raspberry Pi
Standard access points, cables, PCs (typically reused)
Open-source software tools (Python, ML libraries, encryption)

- Advantage: Low material input; avoids custom or proprietary hardware.
- Limitation: The Raspberry Pi, while low-consumption, still has a production footprint (~ 0.3 kg CO_2e).

10.3. Production

Development:
100 hours of programming and system integration on laptops (∼5 kWh)
ML model training and encryption design (∼15–20 kWh)

- Reflection: Local production avoids shipping-related emissions but still consumes energy for computing and testing.
- Total estimated impact: ~ 6.2 kg CO_2e.

10.4. Supplies & Outsourcing

Resources used:
Electricity (testing, updates)
Internet and cloud storage (low volume)
Public libraries and academic tools

- Positive: Low reliance on external services; open access tools reduce long-term economic and environmental cost.
- Risk: If scaled to large deployments, cloud and energy costs could rise.
- Estimated impact: ~ 2 kg CO_2e

10.5. Distribution

Digital delivery only:
Scripts shared via GitHub
System images distributed online

- Advantage: No shipping, packaging, or warehousing needed.
- Improvement: Optimization of codebase could reduce update size, further cutting network energy use.
- Estimated impact: ~ 0.5 kg CO_2e

10.6. Use Phase

Assumptions:
10W continuous usage over 3 years = 262.8 kWh
Based on EU average (0.2kg CO_2/kWh): ~ 52.5 kg CO_2e

- Main concern: The use phase is the most carbon-intensive stage.
- Mitigation: Compared to enterprise-grade IDS appliances, this is a fraction of the energy footprint.

10.7. End-of-Life

Approach:
Devices remain reusable for other purposes (education, testing).
No hazardous components or complex disposal.

- Benefit: Avoids disposal and reduces demand for new devices.
- Offset credited: $-5\text{kg}\,CO_2e$ (based on avoided emissions from replacing one new device).

10.8. Environmental Impacts

Lifecycle Stage	**Estimated CO_2e (kg)**
Materials	0.3
Production	6.2
Supplies & Testing	2.0
Distribution	0.5
Use Phase (3 years)	52.5
End-of-Life Offset	–5.0
Total	$66 - 70\ \text{kg}\,CO_2e$

Table 6: Environmental Impacts

10.9. Environmental Benefits

Low-carbon alternative to commercial IDS appliances
Modular and updatable, extending system life
No external infrastructure needed (e.g., cooling, large servers)
Supports sustainability in healthcare digitalization

10.10. Critical reflection

Positive Consequences	Negative Consequences
Low energy use and material input	Short device lifespans may require frequent updates
Promotes reuse of existing infrastructure	Use of electricity in 24/7 mode (standby emissions)
Encourages open-source and low-cost tools	Difficult to recycle embedded devices if not repurposed
Digital-only distribution avoids shipping waste	Integration into legacy systems may increase debugging and energy use

Table 7: Critical reflection

We recognize that true sustainability involves continuous adaptation. Our project sets a baseline for low-impact cybersecurity in healthcare, but future iterations must address device longevity, efficient updates, and waste management.

11. Ethical Implications

As our IDRS solution relies on machine-learning-based detection and response in IoMT environments, it must balance innovation with patient safety in hospitals.

Firstly, the most pressing ethical concerns arise from automated decision-making and data bias. In clinical settings, a false positive might interrupt a life-saving device, while a false negative could allow a threat to go undetected. Likewise, training models on non-representative data risks unfair outcomes for minority patients or unusual device configurations. Moreover, even anonymized hospital traffic demands rigorous consent and protection, and tiered pricing models must never leave smaller facilities under-protected.

Secondly, operational misuse introduces its own dangers. Rolling out the system without real-world validation can destabilize critical networks, while lapsed updates or expired certificates reopen old vulnerabilities. Inadequate key management or lax access controls only amplifies the risk of exposing sensitive health data, and tools designed for analysis can be co-opted for illicit surveillance.

In the event of an incident—be it a data breach, system failure or patient impact—a clear communication protocol is paramount. Operators and device manufacturers would be notified at once, followed by any patients whose care continuity or identifiable information is at risk. Regulatory bodies under GDPR and the Cyber Resilience Act would receive formal reports within legal timeframes, and all partners would have access to training materials and simulation exercises to reinforce preparedness.

To prevent recurrence, a suite of corrective measures must be woven into every release. Security-by-design principles dictate sandbox testing and independent code-and-model audits. Fail-safe fallbacks ensure that, even if detection fails, critical functions remain online. Detailed access logs establish accountability, while periodic certification for hospital IT staff keeps key-management and incident-response skills sharp. These actions aim to not only recover from issues, but to reduce their likelihood and impact from the outset.

Finally, handling personal data under GDPR requires more than compliance checkboxes—it calls for conviction. All network traffic is encrypted end-to-end, certificates are rigorously managed, and data are minimized or pseudonymized whenever possible. Explicit, informed consent guides any testing, and a full Data Protection Impact Assessment precedes production deployment. Also, we firmly believe that cybersecurity in medical environments must not be a luxury: hospitals save lives just as IoMT devices can, and safeguarding patient dignity cannot be treated as a variable in a cost-benefit analysis. Therefore, our system is designed to be both ethically responsible and technically robust from the ground up — with security, transparency, and equity at its core.

12. Conclusions

To provide a comprehensive view, our project is divided into 3 main sections: technical, business and sustainability. While the overall system fulfills the requirements set by the CRA regulation, we think that we left much room for improvement. Specifically regarding the technical implementation, the AI implementation in the project represented an entirely new area for us we did not have enough time to get the most out of it. Due to our lack of experience, we stuck with a simple "malicious/benign" classification of the network traffic. Later in the project we realized that, in order to fully exploit the AI capabilities, we could have implemented a full classification that allowed us to detect not only what traffic was malicious but also the kind of attack that it represented. This would have enabled the construction of a more elaborate Threat Mitigation and Post-Incident Report.

Regarding the business part, our solution turns cybersecurity into a strategic asset for hospitals and IoMT manufacturers - where compliance meets patient care. By offering plug-and-play protection with automated reporting on affordable hardware, we've created a model that's as operationally practical as it is technically robust. Ultimately, this isn't just security software; it's peace of mind packaged for healthcare's unique needs, delivering value to IT teams and patients alike.

Finally, our environmental impact analysis yielded encouraging results. Most notably, we've demonstrated that effective cybersecurity can be both lightweight and environmentally friendly - secure by design and sustainable by choice, though we recognize opportunities for further improvement in this area as well.

13. Reflection Document

The development of our project not only addressed a significant technical challenge but also provided valuable opportunities to reflect on the lessons learned, areas for improvement, and team performance.

Through the technical implementation of our project, we gained valuable insights about the implementation of Machine Learning models into real-world applications. We also developed a foundational understanding of the environments in which Internet of Medical Things (IoMT) devices operate and the network architecture surrounding it. Integration was a particularly critical aspect, offering us hands-on experience in setting up and executing projects in practical, working environments.

Beyond the technical elements, we learned key lessons in project management and team dynamics. These can be summarized as follows:

- Planning and prioritization: establishing fixed deadlines and adhering to them more rigorously would have led to a more efficient workflow.
- Interdisciplinary collaboration: combining diverse areas of expertise enriched both our solutions and our learning experience.
- Flexibility: greater adaptability would have helped us respond more effectively to unexpected setbacks, allowing us to adjust our approach without compromising progress.

Regarding the involvement of our lecturers, their guidance was generally effective and appreciated. However, there are areas where improvements could enhance the process. While general technical guidance was provided, more in-depth technical sessions could have accelerated the development of the Intrusion Detection and Response System (IDRS). Additionally, increasing the frequency of review sessions with advisors could have optimized the time management of the project.

As a team, we believe our overall performance was solid. Nevertheless, for future projects, we would aim to optimize certain aspects of the work process. While meetings were generally effective, clearer task delegation and more consistent progress monitoring would have helped avoid delays. More rigorous technical documentation would have been key during the final integration phase.

14. Annex

14.1. Packet Capture code

```python
import os
import pyshark as ps
import pandas as pd
from collections import defaultdict

HEADER = '\033[95m'
ENDC = '\033[0m'
BOLD = '\033[1m'

INTERFACE = 'wlp2s0'
OUTPUT_FOLDER = 'ficheros_csv'
DOC_NAME = 'csv_benigno_'
FORMAT = '.csv'
PARSE_IND = 19 # 18 caracters + 1 _ +
START_CELL = 2
END_CELL = 101
flags = {
        "FIN": 0,
        "SYN": 0,
        "RST": 0,
        "PSH": 0,
        "ACK": 0,
        "ECE": 0,
        "CWR": 0,
}

def calculate_iat(packet, prev_time):
    if prev_time is not None:
        iat = (packet.sniff_time - prev_time).total_seconds() * 1000  # Convertir a ms
        iat = round(iat, 2)
    else:
        iat = 0 #Por que iat = None si puede hacerla 0
    return iat, packet.sniff_time

def get_header(p):
    total_hlength = 0

    if hasattr(p, 'eth'):
       total_hlength += 14

    if hasattr(p, 'ip'):
        total_hlength += int(p.ip.hdr_len)

    if hasattr(p, 'icmp'):
       total_hlength += int(p.icmp.hdr_len)

    if hasattr(p, 'tcp'):
       total_hlength += int(p.tcp.hdr_len)

    if hasattr(p, 'udp'):
       total_hlength += int(p.udp.hdr_len)

```

```
    return total_hlength

def get_flags(p):
    if hasattr(p, 'tcp'):
        for flag in flags.keys():
            field_name = flag.lower() + "_flag_number"  # ejemplo: syn_flag_number
            flags[flag] = int(getattr(p.tcp, field_name, 0))
    return flags

def get_port(p):
    dport = sport = None
    if hasattr(p, 'tcp'):
        dport = p.tcp.dstport
        sport = p.tcp.srcport
    elif hasattr(p, 'udp'):
        dport = p.udp.dstport
        sport = p.udp.srcport
    else:   #Agregamos 0 para que tenga sentido en el ML
        dport = 0
        sport = 0
    return dport, sport

def extract_title(timestamp):
    return str(timestamp).replace("-","").replace("
    ↪  ","_").replace(":","").replace(".","")[:PARSE_IND]

def get_title(time_start, time_end):
    return "I"+extract_title(time_start)+"-"+"F"+extract_title(time_end)

try:
    i = 0
    while(True):
        packet_count_dict = defaultdict(int)
        byte_count_dict = defaultdict(int)
        prev_time = None
        label = 0
        data = []
        cap = ps.LiveCapture(interface = INTERFACE)
        try:
            for p in cap.sniff_continuously(100):
                iat, prev_time = calculate_iat(p, prev_time)
                src_ip, dst_ip = (p.ip.src, p.ip.dst) if 'IP' in p else (0, 0) #Agregamos 0 para
                ↪  que tenga sentido en el ML
                dport, sport = get_port(p)
                proto = p.highest_layer
                flags = get_flags(p)
                src_tos = getattr(p, 'ip', 0)
                src_tos = getattr(src_tos, 'tos', 0) if src_tos else 0
                key = (src_ip, dst_ip, sport)
                packet_count_dict[key] += 1
                byte_count_dict[key] += int(p.length)
                paket_cnt = sum(packet_count_dict.values())
                bytes_cnt = sum(byte_count_dict.values())
                data.append({"timestamp":prev_time,
                             "src_ip":src_ip,
                             "dst_ip":dst_ip,
                             "sport":sport,
                             "dport":dport,
```

```
                            "protocol":proto,
                            **flags,
                            "src_tos":src_tos,
                            "iat(ms)":iat,
                            "packet_cnt":paket_cnt,
                            "bytes_cnt":bytes_cnt,
                            "label":label})
        except KeyboardInterrupt:
            print(BOLD + HEADER + "\nFinalizando 'pcap_to_csv' de manera segura por interrupcion
            ↪  de teclado" + ENDC)
            cap.close()
            break
        except Exception as e:
            print(f"Error durante la captura: {e}")
            cap.close()
            continue

        df = pd.DataFrame(data)

        os.makedirs(OUTPUT_FOLDER, exist_ok=True)
        output_path = os.path.join(OUTPUT_FOLDER, get_title(df.iloc[START_CELL]['timestamp'],
        ↪  df.iloc[50]['timestamp']) + '.csv')
        df.to_csv(output_path, index=False)

        i +=1
        if(i == 100):
            break

except KeyboardInterrupt:
    print(BOLD + HEADER + "\nFinalizando 'pcap_to_csv' de manera segura por interrupcion de
    ↪  teclado" + ENDC)
```

14.2. Classification code

train.py :

```
import pandas as pd
import numpy as np
import os
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    classification_report,
    ConfusionMatrixDisplay,
    confusion_matrix,
    precision_score,
    recall_score,
    f1_score
)
import joblib
import matplotlib.pyplot as plt

INPUT_FOLDER = 'Combined_datasets'
MODEL_FILE = 'primera_prueba.plk' # random_forest_model.pkl
ENCODER_FILE = 'encoders_primera_prueba.plk' # encoders.pkl
```

```
CATEGORICAL_COLS = ['protocol', 'src_ip', 'dst_ip']

## Configuración del Search de Hyperparametros
parametros = {
    'n_estimators': [100, 200, 300, 400],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'max_features': ['sqrt', 'log2']
}


def load_csvs_from_folder(folder_path):

    csv_files = [file for file in os.listdir(folder_path) if file.endswith('.csv')]

    if not csv_files:
        raise ValueError(f"No CSV files found in {folder_path}")

    dfs = []
    for file in csv_files:
        full_path = os.path.join(folder_path, file)
        try:
            df = pd.read_csv(full_path)
            dfs.append(df)
            print(f"Loaded: {file} ({df.shape[0]} rows)")
        except Exception as e:
            print(f"Error loading {file}: {e}")

    combined_df = pd.concat(dfs, ignore_index=True)
    print(f"\n Combined DataFrame shape: {combined_df.shape}")

    return combined_df

# Cargamos datasets
df = load_csvs_from_folder(INPUT_FOLDER)
df = df.drop(columns=['timestamp'])

encoders = {}
df_encoded = df.copy()

for col in CATEGORICAL_COLS:
    encoder = LabelEncoder()
    df_encoded[col] = encoder.fit_transform(df[col])
    encoders[col] = encoder

# Vista general del dataset codificado
print("\n Info de Dataset:")
df_encoded.info()

# División de datos entre Train/Test
X = df_encoded.iloc[:,:-1] #Todas las columnas menos label
Y = df_encoded.iloc[:,-1] #La columna label que es el objetivo a predecir
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.1, random_state=123)
↪  #Dividmos el dataset

# Entrenamos el modelo utilizando RandomizedSearchCV
# '--> Nos permite encontrar los mejores parametron para un modelo de ML
#      retornando el mejor modelo basado en una metrica como el f1_score
```

```
search = RandomizedSearchCV(RandomForestClassifier(random_state=123, oob_score=True),
                            param_distributions=parametros,
                            scoring='f1',
                            n_iter=10,
                            cv=3)

search.fit(x_train, y_train)
forest = search.best_estimator_

print("Mejores parametros:", search.best_params_)

# Evaluación con OOB (Out Of Bag)
oob_probs = forest.oob_decision_function_
oob_preds = np.argmax(oob_probs, axis=1) # uso de numpy

precision = precision_score(y_train, oob_preds)
recall = recall_score(y_train, oob_preds)
f1 = f1_score(y_train, oob_preds)

# Guardamos el modelo y los encoders
joblib.dump(forest, MODEL_FILE)
joblib.dump(encoders, ENCODER_FILE)

# Hacemos test
print("\n Test:")
preds = forest.predict(x_test)
print("\n\nEVALUACION CON MUESTRAS DEL TEST SET")
print(classification_report(y_test, preds))

# Confusion Matrix
ConfusionMatrixDisplay(confusion_matrix(y_test, preds)).plot()
plt.title("Confusion Matrix on Test Set")
plt.tight_layout()
plt.show()

filtered_df = x_test[preds == 1]

# Presentación de métricas OOB
print("\n\nEVALUACION CON MUESTRAS OOB")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"Filtered predictions: {filtered_df.shape[0]} positive predictions")
```

random_forest_classification.py :

```
import os
import time
import pandas as pd
import joblib
import numpy as np
import shutil

# Para presentación
HEADER = '\033[95m'
OKBLUE = '\033[94m'
```

```
OKCYAN = '\033[96m'
OKGREEN = '\033[92m'
WARNING = '\033[93m'
FAIL = '\033[91m'
ENDC = '\033[0m'
BOLD = '\033[1m'
UNDERLINE = '\033[4m'

# Configuración
INPUT_FOLDER = 'ficheros_csv'
ATTACK_FOLDER = 'Carpeta_ataque'
BENIGN_FOLDER = 'Carpeta_benigno'
PROCESSED_FOLDER = 'procesados'
MODEL_FILE = 'primera_prueba.plk' # random_forest_model.pkl
ENCODERS_FILE = 'encoders_primera_prueba.plk' # encoders.pkl
CATEGORICAL_COLS = ['timestamp','protocol', 'src_ip', 'dst_ip']
BUFFER_SIZE = 5


# Cargar modelo y encoders
forest = joblib.load(MODEL_FILE)
encoders = joblib.load(ENCODERS_FILE)

# Función de codificación segura
def encode(series, encoder):
    return encoder.transform([
        x if x in encoder.classes_ else encoder.classes_[0]
        for x in series
    ])

def decode(df_to_decode):
    for col in CATEGORICAL_COLS:
        if col in df_to_decode.columns:
            df_to_decode[col] = encoders[col].inverse_transform(df_to_decode[col])
    return df_to_decode

def preprocess(df):

    for col in CATEGORICAL_COLS:
        if col in df.columns:
            df[col] = encode(df[col], encoders[col])

    return df

def save(df, path_csv, preds):
    df['label'] = preds
    has_attack = int((preds == 1).any())

    output_dir = ATTACK_FOLDER if has_attack else BENIGN_FOLDER
    output_filename = os.path.join(output_dir, os.path.basename(path_csv))
    df.to_csv(output_filename, index=False)

    if has_attack:
        print(WARNING + f"Ataque detectado en {path_csv}. Guardado en {output_filename}" + ENDC)
    else:
        print(OKGREEN + f"Sin ataques detectados en {path_csv}" + ENDC)
    return has_attack

```

```
69  def move_processed(df, path_csv):
70      # mover a 'procesados' opcional
71      try:
72          processed_path = os.path.join(PROCESSED_FOLDER, os.path.basename(path_csv))
73          df.to_csv(path_csv, index=False)
74          shutil.move(path_csv, processed_path)
75          print(BOLD + f"Archivo movido a {processed_path}" + ENDC)
76      except Exception as e:
77          print(BOLD + f"No se pudo mover {path_csv}: {e}" + ENDC)
78
79  # Función principal de procesamiento
80  def process_file(path_csv):
81      print(f"\nProcesando archivo: {path_csv}")
82
83      df_original = pd.read_csv(path_csv)
84      df_model = df_original.copy()
85      df_model = preprocess(df_model) # Se encarga de encoder y preprocesado
86
87      #Predicción
88      preds = forest.predict(df_model)
89      df_original['label'] = preds
90      df_model = decode(df_model)
91      save(df_model, path_csv, preds) #Guarda en maligno o benigno dependiendo de la clasificación
92      move_processed(df_original, path_csv) #Mueve los .csvs procesados a otra carpeta
93
94  def erase_oldest_csv(csv_list, csv_folder):
95      all_paths = [csv_folder + "/{0}".format(x) for x in csv_list]
96      oldest_csv = min(all_paths, key=os.path.getctime)
97      os.remove(oldest_csv)
98
99
100 # Bucle de escucha de archivos
101
102 # Crear carpetas si no existen
103 if not os.path.exists(ATTACK_FOLDER):
104     os.makedirs(ATTACK_FOLDER)
105
106 if not os.path.exists(BENIGN_FOLDER):
107     os.makedirs(BENIGN_FOLDER)
108
109 if not os.path.exists(PROCESSED_FOLDER):
110     os.makedirs(PROCESSED_FOLDER)
111
112 print(UNDERLINE + "\nEsperando archivos nuevos..." + ENDC)
113
114
115 try:
116     while True:
117         csv_input_list = os.listdir(INPUT_FOLDER)
118         processed_csv_list = os.listdir(PROCESSED_FOLDER)
119         attack_csv_list = os.listdir(ATTACK_FOLDER)
120         benign_csv_list = os.listdir(BENIGN_FOLDER)
121
122         if csv_input_list:
123             for csv_file in csv_input_list:
124                 csv_file_path = os.path.join(INPUT_FOLDER, csv_file)
125                 try:
126                     process_file(csv_file_path)
```

```
                except Exception as e:
                    print(f"Error procesando {csv_file_path}: {e}")
        else:
            print(OKBLUE + "No hay archivos nuevos. Esperando..." + ENDC)

        if (len(processed_csv_list)) >= BUFFER_SIZE:
            erase_oldest_csv(processed_csv_list,PROCESSED_FOLDER)

        if (len(attack_csv_list)) >= BUFFER_SIZE:
            erase_oldest_csv(attack_csv_list,ATTACK_FOLDER)

        if (len(benign_csv_list)) >= BUFFER_SIZE:
            erase_oldest_csv(benign_csv_list,BENIGN_FOLDER)

        time.sleep(5)

except KeyboardInterrupt:
    print(BOLD + HEADER + "\nFinalizando 'random_forest_classification' de manera segura por interrupcion de teclado" + ENDC)

```

14.3. Mitigation code

```
import os
import time
import signal
import subprocess
import pandas as pd

CARPETA = "ficheros_malignos"
CARPETA_LOGS = "bloqueados"
WHITELIST = "whitelist.txt"
LOG_BLOQUEADOS = os.path.join(CARPETA_LOGS, "bloqueados.log")

ya_bloqueadas = set()

os.makedirs(CARPETA_LOGS, exist_ok=True)

def cargar_whitelist(ruta):
    if not os.path.exists(ruta):
        return set()
    with open(ruta, 'r') as f:
        return set(line.strip() for line in f if line.strip())

def cargar_log_bloqueados(ruta):
    if not os.path.exists(ruta):
        return set()
    with open(ruta, 'r') as f:
        return set(line.strip() for line in f if line.strip())

def bloquear_ip(ip):
    print(f"Bloqueando IP de origen: {ip}")
    subprocess.run(["sudo", "iptables", "-A", "INPUT", "-s", ip, "-j", "DROP"])
    with open(LOG_BLOQUEADOS, 'a') as log:
        log.write(ip + "\n")
    ya_bloqueadas.add(ip)
```

```
34
35  def procesar_csv(path, whitelist):
36      try:
37          df = pd.read_csv(path)
38          ips_malas = df[df["label"] == 1]["src_ip"].unique()
39          for ip in ips_malas:
40              ip = ip.strip()
41              if ip not in whitelist and ip not in ya_bloqueadas:
42                  bloquear_ip(ip)
43
44      except Exception as e:
45          print(f"Error al procesar {path}: {e}")
46
47  def restaurar_iptables(signum=None, frame=None):
48      print("\nRestaurando reglas iptables...")
49      subprocess.run(["sudo", "iptables", "-F"])
50      print("Reglas iptables restauradas. Saliendo.")
51      exit(0)
52
53  def main():
54      print("Monitoreando carpeta en busca de ataques...")
55
56
57      whitelist = cargar_whitelist(WHITELIST)
58      global ya_bloqueadas
59      ya_bloqueadas = cargar_log_bloqueados(LOG_BLOQUEADOS)
60      ya_procesados = set()
61
62      signal.signal(signal.SIGINT, restaurar_iptables)
63      signal.signal(signal.SIGTERM, restaurar_iptables)
64
65      while True:
66          archivos = [f for f in os.listdir(CARPETA) if f.endswith(".csv")]
67          nuevos = [f for f in archivos if f not in ya_procesados]
68          for archivo in nuevos:
69              ruta = os.path.join(CARPETA, archivo)
70              print(f"Detectado archivo: {archivo}")
71              procesar_csv(ruta, whitelist)
72              ya_procesados.add(archivo)
73          time.sleep(5)
74
75  if __name__ == "__main__":
76      main()
77
```

14.4. Encryption and Authentication code

Proxy Client code :

```
1  import socket
2  import threading
3  import os
4  from cryptography.hazmat.primitives.ciphers.aead import AESGCM
5  from cryptography.hazmat.primitives import hashes
6  from cryptography.hazmat.primitives.asymmetric import padding
7  from cryptography.hazmat.primitives import serialization
```

```
8  from cryptography import x509
9
10 # Cargar certificado público
11 with open('certificate.pem', 'rb') as f:
12     certificate = f.read()
13     public_key = x509.load_pem_x509_certificate(certificate).public_key()
14
15 # Configuración del proxy
16 PROXY_IP = '0.0.0.0'
17 PROXY_PORT = 8080
18 DESTINATION_IP = '127.0.0.1'  # Cambiado para conectar al servidor en la misma máquina, IP
   ↪  específica para cada conexion
19 DESTINATION_PORT = 12345
20
21
22 # Función para manejar la comunicación con cada cliente
23 def handle_client(client_socket):
24     try:
25         # Recibir datos del cliente
26         data = client_socket.recv(4096)
27         if not data:
28             client_socket.close()
29             return
30
31         print("Datos recibidos desde curl o cliente externo, generando clave AES...")
32
33         # Generar clave AES-256 y cifrar los datos
34         key_AES = AESGCM.generate_key(bit_length=256)
35         AESgcm = AESGCM(key_AES)
36         nonce = os.urandom(16)
37         mensaje_cifrado = AESgcm.encrypt(nonce, data, None)
38
39         # Cifrar la clave AES con la clave pública RSA
40         clave_AES_cifrada = public_key.encrypt(
41             key_AES,
42             padding.OAEP(
43                 mgf=padding.MGF1(algorithm=hashes.SHA256()),
44                 algorithm=hashes.SHA256(),
45                 label=None
46             )
47         )
48
49         print("Intentando conectar al servidor en 127.0.0.1:12345...")
50
51         # Crear socket hacia el servidor real
52         with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
53             try:
54                 server_socket.connect((DESTINATION_IP, DESTINATION_PORT))
55                 print("Conexión establecida con el servidor.")
56
57
58
59                 server_socket.sendall(clave_AES_cifrada + b'||' + nonce + b'||' +
                   ↪  mensaje_cifrado)
60
61                 # Recibir la respuesta del servidor
62                 response = server_socket.recv(4096)
63                 print("Respuesta recibida desde el servidor, reenviando al cliente.")
```

```
64
65                  # Reenviar la respuesta al cliente original
66                  client_socket.sendall(response)
67
68              except Exception as e:
69                  print(f"Error al conectar con el servidor: {e}")
70
71      except Exception as e:
72          print(f"Error al procesar la solicitud del cliente: {e}")
73      finally:
74          client_socket.close()
75
76
77  # Iniciar el proxy
78  proxy_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
79  proxy_socket.bind((PROXY_IP, PROXY_PORT))
80  proxy_socket.listen(5)
81
82  print(f'Proxy activo en {PROXY_IP}:{PROXY_PORT}')
83
84  try:
85      while True:
86          client_socket, addr = proxy_socket.accept()
87          print(f'Conexión entrante desde {addr}')
88          client_handler = threading.Thread(target=handle_client, args=(client_socket,))
89          client_handler.start()
90  except KeyboardInterrupt:
91      print("Proxy cerrado")
92  finally:
93      proxy_socket.close()
94
```

Server code :

```
1   #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4   import socket
5   import threading
6   import os
7   from cryptography.hazmat.primitives.ciphers.aead import AESGCM
8   from cryptography.hazmat.primitives import hashes
9   from cryptography.hazmat.primitives.asymmetric import padding
10  from cryptography.hazmat.primitives import serialization
11
12  # Cargar clave privada
13  with open('key.pem', 'rb') as f:
14      private_key = serialization.load_pem_private_key(
15          f.read(),
16          password=b'passphrase'  # Aquí se especifica la contraseña
17      )
18
19  # Configuración del servidor
20  SERVER_IP = '0.0.0.0'
21  SERVER_PORT = 12345
22
23
```

```python
def handle_client(client_socket):
    try:
        # Recibir datos del cliente
        data = client_socket.recv(4096)
        if not data:
            client_socket.close()
            return

        # Intentar separar datos cifrados (se espera b'||' en medio)
        if b'||' in data:
            try:
                clave_AES_cifrada, nonce, mensaje_cifrado = data.split(b'||')

                # Descifrar la clave AES con RSA
                key_AES = private_key.decrypt(
                    clave_AES_cifrada,
                    padding.OAEP(
                        mgf=padding.MGF1(algorithm=hashes.SHA256()),
                        algorithm=hashes.SHA256(),
                        label=None
                    )
                )

                # Descifrar mensaje con AES-256
                AESgcm = AESGCM(key_AES)
                mensaje_descifrado = AESgcm.decrypt(nonce, mensaje_cifrado, None)
                print(f'Mensaje descifrado recibido:
                ↪  {mensaje_descifrado.decode(errors="ignore")}')

                #Responder con un mensaje cifrado

                respuesta = b'Respuesta segura desde el servidor'
                nonce_respuesta = os.urandom(16)
                respuesta_cifrada = AESgcm.encrypt(nonce_respuesta, respuesta, None)

                client_socket.sendall(nonce_respuesta + b'||' + respuesta_cifrada)
            except Exception as e:
                print(f'Error al procesar la solicitud cifrada: {e}')
        else:
            # Si no está cifrado, responder con HTTP estándar
            print("Datos recibidos en texto plano:\n" + data.decode(errors='ignore'))

            http_response = b"HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\n\r\nServidor activo
            ↪  y funcionando correctamente."
            client_socket.sendall(http_response)

    except Exception as e:
        print(f'Error al procesar la solicitud: {e}')
    finally:
        client_socket.close()


# Iniciar servidor
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((SERVER_IP, SERVER_PORT))
server_socket.listen(5)

print(f'Servidor activo en {SERVER_IP}:{SERVER_PORT}')
```

```
80
81  try:
82      while True:
83          client_socket, addr = server_socket.accept()
84          print(f'Conexión entrante desde {addr}')
85          client_handler = threading.Thread(target=handle_client, args=(client_socket,))
86          client_handler.start()
87  except KeyboardInterrupt:
88      print("Servidor cerrado")
89  finally:
90      server_socket.close()
91
```

Key code :

```
1   #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3   """
4
5
6   @author: alejandro
7   """
8
9   from cryptography.hazmat.primitives.ciphers.aead import AESGCM
10  from cryptography.hazmat.primitives import hashes
11  from cryptography.hazmat.primitives.asymmetric import padding
12  from cryptography.hazmat.primitives import serialization
13  from cryptography import x509
14  import os
15
16  class KeyAes:
17      def __init__(self, public_key_path=None, private_key_path=None):
18          self.public_key = None
19          self.private_key = None
20          if public_key_path:
21              self.load_public_key(public_key_path)
22          if private_key_path:
23              self.load_private_key(private_key_path)
24
25      def load_public_key(self, path):
26          with open(path, 'rb') as f:
27              cert = x509.load_pem_x509_certificate(f.read())
28              self.public_key = cert.public_key()
29
30      def load_private_key(self, path):
31          with open(path, 'rb') as f:
32              self.private_key = serialization.load_pem_private_key(f.read(), password=None)
33
34      def generate_aes_key(self):
35          return AESGCM.generate_key(bit_length=256)
36
37      def encrypt_message(self, message, key):
38          aesgcm = AESGCM(key)
39          nonce = os.urandom(16)
40          ciphertext = aesgcm.encrypt(nonce, message, None)
41          return nonce, ciphertext
42
```

```
    def decrypt_message(self, nonce, ciphertext, key):
        aesgcm = AESGCM(key)
        return aesgcm.decrypt(nonce, ciphertext, None)

    def encrypt_key(self, key):
        if not self.public_key:
            raise ValueError("Clave pública no cargada")
        return self.public_key.encrypt(
            key,
            padding.OAEP(
                mgf=padding.MGF1(algorithm=hashes.SHA256()),
                algorithm=hashes.SHA256(),
                label=None
            )
        )

    def decrypt_key(self, encrypted_key):
        if not self.private_key:
            raise ValueError("Clave privada no cargada")
        return self.private_key.decrypt(
            encrypted_key,
            padding.OAEP(
                mgf=padding.MGF1(algorithm=hashes.SHA256()),
                algorithm=hashes.SHA256(),
                label=None
            )
        )
```

14.5. Reports: User Interface code

```
import streamlit as st
import re
import sounddevice as sd
import soundfile as sf
import threading
import time


# ==================== CONFIGURACIÓN ====================
LOG_COLORS = {
   "warning": "#006400",  # Verde oscuro
   "error": "#FF0000",    # Rojo fuerte
   "default": "#000000"   # Negro
}


BACKGROUND_COLOR = "#F0F0F0"  # Gris muy claro
BORDER_COLOR = "#C0C0C0"      # Gris medio


# ==================== FUNCIONES PRINCIPALES ====================
def main():
   st.set_page_config(page_title="Security Dashboard", layout="wide")

   # Inicializar variables de sesión
   session_defaults = {
```

```
        'src_ip_filter': "",
        'dst_ip_filter': "",
        'log_count': 10,
        'audio_playing': False,
        'logged_in': False,
        'current_view': "Home"
    }

    for key, value in session_defaults.items():
        if key not in st.session_state:
            st.session_state[key] = value

    if not st.session_state.logged_in:
        login()
    else:
        dashboard()


def login():
    st.title(" Login")
    username = st.text_input("Username")
    password = st.text_input("Password", type="password")

    if st.button("Login"):
        if username == "1234" and password == "1234":
            st.session_state.logged_in = True
            st.rerun()
        else:
            st.error("Invalid username or password")


# ==================== FUNCIONES DE AUDIO ====================
def play_audio_with_timer():
    try:
        if not st.session_state.audio_playing:
            st.session_state.audio_playing = True
            audio_data, sample_rate = sf.read("sirena.mp3", dtype='float32')

            def audio_player():
                sd.play(audio_data, sample_rate)
                sd.wait()

            audio_thread = threading.Thread(target=audio_player)
            audio_thread.start()

            def stop_audio():
                time.sleep(45)
                sd.stop()
                st.session_state.audio_playing = False
                if st.session_state.current_view == 'View Logs':
                    st.rerun()

            threading.Thread(target=stop_audio).start()

    except Exception as e:
        st.error(f"Error al reproducir audio: {e}")
        st.session_state.audio_playing = False

```

```
85
86  # ==================== FUNCIONES DE LOGS ====================
87  def parse_log_line(log_line):
88      ip_pattern = r'(?:\d+:)?(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})'
89      matches = re.findall(ip_pattern, log_line)
90
91      if len(matches) >= 2:
92          return {'src_ip': matches[0], 'dst_ip': matches[1], 'full_log': log_line}
93      elif len(matches) == 1:
94          return {'src_ip': matches[0], 'dst_ip': None, 'full_log': log_line}
95      return {'src_ip': None, 'dst_ip': None, 'full_log': log_line}
96
97
98  def filter_logs(logs, src_ip_filter="", dst_ip_filter=""):
99      filtered = []
100     for log in logs:
101         if not log.strip():
102             continue
103         parsed = parse_log_line(log)
104         src_match = (not src_ip_filter) or (parsed['src_ip'] and src_ip_filter in
        ↪  parsed['src_ip'])
105         dst_match = (not dst_ip_filter) or (parsed['dst_ip'] and dst_ip_filter in
        ↪  parsed['dst_ip'])
106         if src_match and dst_match:
107             filtered.append(parsed['full_log'])
108     return filtered if filtered else []
109
110
111 def read_last_n_logs(filename="warnings.txt", n=None):
112     try:
113         with open(filename, "r") as file:
114             logs = [line.strip() for line in file.readlines() if line.strip()]
115         return logs[-n:] if n and n < len(logs) else logs
116     except FileNotFoundError:
117         st.error(f"Log file {filename} not found. Using default logs.")
118         return [
119             "02:36:30 || 37138:192.168.1.1 -> 54123:10.0.0.2 || Device: Insuline_pump-01 ||
            ↪  Warning: High number of pings",
120             "03:15:45 || 192.168.1.2 -> 10.0.0.3 || Device: electro-02 || Error: Unauthorized
            ↪  access attempt",
121             "04:22:10 || 45123:192.168.1.3 -> 10.0.0.2 || Device: electro-03 || Warning:
            ↪  Unauthorized access attempt",
122             "05:50:20 || 192.168.1.1 -> 10.0.0.4 || Device: Insulina_pump-04 || Error: Bad packets
            ↪  detected",
123             "06:12:55 || 192.168.1.5 -> 10.0.0.4 || Device: Insulina_pump-04 || Warning: CPU usage
            ↪  above 90%",
124         ]
125
126
127 def display_logs(logs):
128     colored_logs = []
129     for log in logs:
130         log_lower = log.lower()
131         if "error" in log_lower:
132             color = LOG_COLORS["error"]
133         elif "warning" in log_lower:
134             color = LOG_COLORS["warning"]
135         else:
```

```
            color = LOG_COLORS["default"]
        colored_logs.append(f'<span style="color: {color}; font-weight: 500;">{log}</span>')
    html_content = "<br>".join(colored_logs)
    st.markdown(f"""
    <div style="font-family: 'Courier New', monospace; white-space: pre; overflow-x: auto;
    ↪  overflow-y: auto; padding: 12px; background-color: {BACKGROUND_COLOR}; border-radius: 8px;
    ↪  border: 1px solid {BORDER_COLOR}; max-height: 65vh; width: 100%; font-size: 0.92em;
    ↪  line-height: 1.4;">
    {html_content}
    </div>
    """, unsafe_allow_html=True)
    st.markdown(f"""
    <style>
    div[data-testid="stMarkdownContainer"] div::-webkit-scrollbar {{ height: 8px; width: 8px; }}
    div[data-testid="stMarkdownContainer"] div::-webkit-scrollbar-track {{ background: #E0E0E0; }}
     div[data-testid="stMarkdownContainer"] div::-webkit-scrollbar-thumb {{ background: #A0A0A0;
     ↪  border-radius: 4px; }}
    </style>
    """, unsafe_allow_html=True)


# ==================== INTERFAZ PRINCIPAL ====================
def dashboard():
    st.sidebar.title("Navigation")
    st.session_state.current_view = st.sidebar.selectbox("Select View", ["Home", "View Logs",
    ↪  "Post-Incident Report"], key="nav_select")
    st.sidebar.button("Logout", on_click=logout)


    if st.session_state.current_view == "Home":
        st.title(" Security Dashboard")
        st.write("Welcome to the security monitoring interface.")


    elif st.session_state.current_view == "View Logs":
        st.title(" Log Viewer")
        if not st.session_state.audio_playing:
            threading.Thread(target=play_audio_with_timer).start()
        if st.session_state.audio_playing:
            st.warning(" Alarma activa (se apagará automáticamente en 45 segundos)")
        col1, col2 = st.columns(2)
        with col1:
            st.session_state.src_ip_filter = st.text_input("Filter by Source IP",
            ↪  value=st.session_state.src_ip_filter, placeholder="Ej: 192.168")
        with col2:
            st.session_state.dst_ip_filter = st.text_input("Filter by Destination IP",
            ↪  value=st.session_state.dst_ip_filter, placeholder="Ej: 10.0.0")
        st.session_state.log_count = st.selectbox("Number of logs to display", options=[5, 10, 50,
        ↪  100, 1000, "All"], index=[5, 10, 50, 100, 1000,
        ↪  "All"].index(st.session_state.log_count if st.session_state.log_count in [5, 10, 50,
        ↪  100, 1000] else "All"))
        all_logs = read_last_n_logs()
        filtered_logs = filter_logs(all_logs, src_ip_filter=st.session_state.src_ip_filter,
        ↪  dst_ip_filter=st.session_state.dst_ip_filter)
        if st.session_state.log_count != "All":
            filtered_logs = filtered_logs[-int(st.session_state.log_count):]
        display_logs(filtered_logs)
        st.info(f"Showing {len(filtered_logs)} of {len(all_logs)} log entries")
```

```
        col1, col2 = st.columns(2)
        with col1:
            if st.button(" Refresh Logs"):
                st.rerun()
        with col2:
            if st.button(" Stop Alarm"):
                sd.stop()
                st.session_state.audio_playing = False
                st.rerun()


    elif st.session_state.current_view == "Post-Incident Report":
        st.title(" Post-Incident Report")


        col1, col2 = st.columns(2)
        with col1:
            src_ip_post = st.text_input("Source IP", placeholder="e.g. 192.168")
        with col2:
            dst_ip_post = st.text_input("Destination IP", placeholder="e.g. 10.0.0")


        all_logs = read_last_n_logs()
        filtered_logs = filter_logs(all_logs, src_ip_filter=src_ip_post,
        ↪  dst_ip_filter=dst_ip_post)


        if filtered_logs:
            st.success(f"{len(filtered_logs)} matching entries found. You can now download the
            ↪  report.")
            with open("post_incident_report.pdf", "rb") as file:
                st.download_button(label="Download Post-Incident Report", data=file,
                ↪  file_name="post_incident_report.pdf", mime="application/pdf")
        else:
            st.warning("No matching logs found. Please adjust your filters.")


def logout():
    sd.stop()
    st.session_state.logged_in = False
    st.session_state.src_ip_filter = ""
    st.session_state.dst_ip_filter = ""
    st.session_state.log_count = 10
    st.session_state.audio_playing = False
    st.rerun()


# ==================== EJECUCIÓN ====================
if __name__ == "__main__":
    main()

```

14.6. Reports: Post-Incident Analysis code

data_processor_post_incident.py :

```
import glob
import os
import subprocess
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from datetime import datetime, timedelta

# ---------------------- Configuraciones iniciales ----------------------
COLUMNS_FILTER = [
    "timestamp", "src_ip", "dst_ip", "sport", "dport", "protocol",
    "FIN", "SYN", "RST", "PSH", "ACK", "ECE", "CWR",
    "src_tos", "iat(ms)", "packet_cnt", "bytes_cnt", "label"
]

BOLD_RED = '\033[1;31m'
BOLD_GREEN = '\033[1;32m'
RESET = '\033[0m'

config_graficos = [
    {   "args": {"x": "timestamp", "y": "bytes_cnt", "tipo": "lineplot", "hue": "label",
    ↪  "output": "1-graph_line_timestamp-bytes_cnt.png"},
        "caption": "Temporal distribution of bytes count by label",
        "categoria": "Temporal Analysis"
    },
    {   "args": {"x": "timestamp", "y": "packet_cnt", "tipo": "lineplot", "hue": "label",
    ↪  "output": "2-graph_line_timestamp-packet_cnt.png"},
        "caption": "Temporal distribution of packet count by label",
        "categoria": "Temporal Analysis"
    },
    {   "args": {"x": "timestamp", "y": "label", "tipo": "lineplot", "output":
    ↪  "3-graph_line_timestamp-anomalias.png"},
        "caption": "Temporal distribution of detected anomalies",
        "categoria": "Temporal Analysis"
    },
    {   "args": {"x": "src_ip", "y": "sport", "titulo": "HeatMap Source", "tipo": "heatmap",
    ↪  "output": "4-graph_heatmap_source.png", "top_n": 100},
        "caption": "Heatmap of source IPs and their corresponding source ports (top 100)",
        "categoria": "IPs \\& Ports"
    },
    {   "args": {"x": "dst_ip", "y": "dport", "titulo": "HeatMap Destination", "tipo": "heatmap",
    ↪  "output": "5-graph_heatmap_destination.png", "top_n": 100},
        "caption": "Heatmap of source IPs and their corresponding destination ports (top 100)",
        "categoria": "IPs \\& Ports"
    },
    {   "args": {"x": "protocol", "y": "iat(ms)", "titulo": "Interarrival time per protocol",
    ↪  "hue": "label", "tipo": "barplot", "output": "6-graph_duracion_protocolos.png"},
        "caption": "Duration of network traffic by protocol and label",
        "categoria": "Protocols"
    },
    {   "args": {"x": "protocol", "titulo": "Protocol distribution", "tipo": "barplot", "output":
    ↪  "7-graph_protocolos-anomalias.png"},
        "caption": "Protocol distribution comparing normal and anomalous traffic",
        "categoria": "Protocols"
    },
```

```
    {   "args": {"x": "Flag", "titulo": "Flags Histogram", "tipo": "hist", "hue": "label",
    ↪  "output": "8-graph_histograma_flags.png"},
        "caption": "Distribution of TCP flags in normal and anomalous traffic",
        "categoria": "Flags Analysis",
        "preprocesar": lambda data: data.melt(
            id_vars = ["label"],
            value_vars = ["FIN", "SYN", "RST", "PSH", "ACK", "ECE", "CWR"],
            var_name="Flag",
            value_name="Valor"
        )
    },
    {   "args": {"x": "bytes_cnt", "titulo": "Bytes tx Histogram", "tipo": "hist", "hue":
    ↪  "label", "output": "9-graph_histograma_bytes.png"},
        "caption": "Distribution of transmitted bytes in network traffic",
        "categoria": "Traffic histograms"
    },
    {   "args": {"x": "packet_cnt", "titulo": "Packet tx Histogram", "tipo": "hist", "hue":
    ↪  "label", "output": "10-graph_histograma_packet.png"},
        "caption": "Distribution of transmitted packets in network traffic",
        "categoria": "Traffic histograms"
    },
    {   "args": {"x": "iat(ms)", "titulo": "Interarrival time Histogram", "tipo": "hist", "hue":
    ↪  "label", "output": "11-graph_histograma_iat.png"},
        "caption": "Distribution of inter-arrival times between network packets",
        "categoria": "Traffic histograms"
    }
]

# ---------------------- Funciones privadas ----------------------
def graficar(df, x, y=None, titulo="", tipo="lineplot", hue=None, output="grafico.png",
                    figsize=(10, 5), xlabel=None, ylabel=None, rot_xticks=45, top_n=None,
                    ↪  **kwargs):

    plt.figure(figsize=figsize)

    if hue in df.columns and set(df[hue].dropna().unique()).issubset({0, 1}):
        df[hue] = df[hue].map({0: "Benign", 1: "Anomaly"})
        if 'palette' not in kwargs:
            kwargs['palette'] = {"Benign": "green", "Anomaly": "red"}

    if top_n and y:
        df = df.sort_values(by=y, ascending=False).head(top_n)

    try:
        if tipo == "lineplot":
            sns.lineplot(data=df, x=x, y=y, hue=hue, **kwargs)
        elif tipo == "barplot":
            sns.barplot(data=df, x=x, y=y, hue=hue, estimator=sum, **kwargs)
        elif tipo == "barh":
            df.set_index(x)[y].plot(kind="barh", hue=hue, **kwargs)
        elif tipo == "scatter":
            sns.scatterplot(data=df, x=x, y=y, hue=hue, **kwargs)
        elif tipo == "hist":
            sns.histplot(data=df, x=x, hue=hue, bins=kwargs.get("bins", 50),
            ↪  kde=kwargs.get("kde", False), **kwargs)
        elif tipo == "pie":
            df.set_index(x).plot.pie(y=y if y else "count", autopct='%1.1f%%', legend=False)
        elif tipo == "boxplot":
```

```
            sns.boxplot(data=df, x=x, y=y)
        elif tipo== "heatmap":
            top_x = df[x].value_counts().head(top_n).index if top_n else df[x].unique()
            top_y = df[y].value_counts().head(top_n).index if top_n else df[y].unique()
            df_filtrado = df[df[x].isin(top_x) & df[y].isin(top_y)]
            tabla = df_filtrado.groupby([x, y]).size().unstack(fill_value=0)
            sns.heatmap(tabla, cmap="YlOrRd", annot=True, fmt="d")
        else:
            raise ValueError(f"Tipo de gráfico '{tipo}' no soportado.")

        plt.title(titulo)
        if xlabel: plt.xlabel(xlabel)
        if ylabel: plt.ylabel(ylabel)
        if rot_xticks: plt.xticks(rotation=rot_xticks)
        plt.tight_layout()
        plt.savefig(output)
        plt.close()
        return True
    except Exception as e:
        print(f"Error generating {output}: {str(e)}")
        plt.close()
        return False

def procesamiento_datos_rango(data_dir="./procesados", start_date_str=None, end_date_str=None):
    # Example: "25/04/2025 14:11:00"
    start_time = datetime.strptime(start_date_str, "%d/%m/%Y %H:%M:%S")
    end_time = datetime.strptime(end_date_str, "%d/%m/%Y %H:%M:%S")

    all_dfs = []

    for file in os.listdir(data_dir):
        if not file.endswith(".csv") or '-' not in file:
            continue

        try:
            p1, p2 = file.split('-')
            f_start = datetime.strptime(p1[1:16], "%Y%m%d_%H%M%S")
            f_end = datetime.strptime(p2[1:16], "%Y%m%d_%H%M%S")

            if f_end < start_time or f_start > end_time:
                continue

            df = pd.read_csv(os.path.join(data_dir, file))
            df["timestamp"] = pd.to_datetime(df["timestamp"], errors="coerce")
            df = df[(df["timestamp"] >= start_time) & (df["timestamp"] <= end_time)]
            if not df.empty:
                all_dfs.append(df)

        except Exception as e:
            print(f"Error en {file}: {e}")

    if not all_dfs:
        print("No se encontraron archivos que cumplan con los criterios de búsqueda.")
        return None

    data = pd.concat(all_dfs, ignore_index=True)
    data = data.sort_values("timestamp")
    data = data[[col for col in COLUMNS_FILTER if col in data.columns]]
```

```
160
161     return data
162
163 def generar_warnings_file(data, output_dir="./warnings_logs", max_lines=1_000_000):
164     warnings = [
165         f"{row['timestamp'].strftime('%d/%m/%Y %H:%M:%S')}\t||"
166         f"Source: {'0.0.0.0' if row['src_ip'] == '0' else row['src_ip']}:{row['sport']}\t||"
167         f"Destination: {'0.0.0.0' if row['src_ip'] == '0' else row['dst_ip']}:{row['dport']}\t||"
168         f"Protocol: {row['protocol']}\t||"
169         f"\t WARNING [!]\n"
170         for _, row in data.iterrows() if row["label"] == 1
171     ]
172
173     os.makedirs(output_dir, exist_ok=True)
174
175     if not warnings:
176         print("{BOLD_RED}No hay warnings para guardar.{RESET}")
177         return
178
179     total = len(warnings)
180     for i in range(0, total, max_lines):
181         chunk = warnings[i:i+max_lines]
182
183         # Extraer fecha de la primera línea del chunk
184         if chunk:
185             primera_linea = chunk[0]
186             try:
187                 fecha_str = primera_linea.split('\t||')[0]
188                 fecha_dt = datetime.strptime(fecha_str, "%d/%m/%Y %H:%M:%S")
189                 nombre_fecha = fecha_dt.strftime("%Y%m%d_%H%M%S")
190             except Exception as e:
191                 print(f"{BOLD_RED}Error al parsear fecha en línea: {primera_linea}\n↪  {e}{RESET}")
192                 nombre_fecha = f"chunk_{i}"
193         else:
194             nombre_fecha = f"chunk_{i}"
195
196         # Crear nombre de archivo
197         filename = os.path.join(output_dir, f"warnings_{nombre_fecha}.csv")
198
199         # Guardar
200         with open(filename, "w", encoding="utf-8") as f:
201             f.writelines(chunk)
202
203         print(f"{BOLD_GREEN}saved: {filename} ({len(chunk)} warnings){RESET}")
204
205 def generar_graficos(data):
206      # Borrar todas las imagenes creadas anteriormente de la carpeta ./graficos
207     [os.remove(f) for f in glob.glob("./graficos/*.png")]
208
209     graficos_generados = []
210     for config in config_graficos:
211         args = config["args"].copy()  # Copia para no modificar el original
212         output_path = os.path.join("graficos", args["output"])
213         output_path = output_path.replace("\\", "/")
214         args["output"] = output_path
215
216         df = config["preprocesar"](data) if "preprocesar" in config else data
```

```
217
218         # Verificar que los datos no estén vacíos
219         if df.empty:
220             print(f"{BOLD_RED}Datos vacíos para {args['output']}, se omite.{RESET}")
221             continue
222
223         # Generar el gráfico
224         graficar(df, **args)
225
226         # Guardar metadatos para LaTeX
227         graficos_generados.append({
228             "ruta": output_path,
229             "caption": config["caption"],
230             "categoria": config["categoria"]
231         })
232
233     return graficos_generados
234
235 def generar_reporte_pdf(graficos_dir="./graficos", graficos=None, start_date=None,
    ↪  final_date=None, primera_anomalia=None):
236     #Constantes y variables locales
237     start_date = datetime.strptime(start_date, "%d/%m/%Y %H:%M:%S")
238     final_date = datetime.strptime(final_date, "%d/%m/%Y %H:%M:%S")
239     current_category = None
240     latex_code = ""
241
242     # Contenido
243     contenido = fr"""
244     \section{{Introduction}}
245     This report presents a post-incident analysis between \textbf{{{start_date}}} to
    ↪  \textbf{{{final_date}}}.\\ \\"""
246
247     if pd.notnull(primera_anomalia):
248         contenido += fr"""The first anomaly was at \textbf{{{primera_anomalia}}}\\ \\
249                           The following visualizations summarize the main findings of the
                          ↪  analysis. The interpretation of these charts should be carried out
                          ↪  by the responsible technical team.\\"""
250     else:
251         contenido += fr"""No anomalies were detected in the indicated range.\\ \\"""
252
253     # Añadiendo graficos
254     for g in graficos:
255         # Verificar que exista la imagen
256         if not os.path.exists(g["ruta"]):
257             print(f"{BOLD_RED}Imagen no encontrada: {g['ruta']}{RESET}")
258             continue
259
260         # Agregar título de categoría (si cambia)
261         if g["categoria"] != current_category:
262             latex_code += f"\n\\section{{{g['categoria']}}}\n\n"
263             current_category = g["categoria"]
264
265         # Añadir figura
266         latex_code += f"""\\begin{{figure}}[H]
267         \\centering
268         \\includegraphics[width=0.85\\textwidth]{{{g['ruta']}}}
269         \\caption{{{g['caption']}}}
270         \\end{{figure}}\n\n"""
```

```
271
272     latex_code += f"""\\end{{document}}"""
273
274     # Abrimos el archivo original
275     with open("template_report.tex", "r", encoding="utf-8") as f:
276         lineas = f.readlines()
277
278     # Buscamos la línea clave
279     indice = next(i for i, linea in enumerate(lineas) if "% CONTENIDO PRINCIPAL" in linea)
280
281     # Reescribimos el archivo a partir de esa línea
282     with open("post-incident_report.tex", "w", encoding="utf-8") as f:
283         f.writelines(lineas[:indice+1])  # conservamos hasta la línea clave
284         f.write("\n" + contenido)  # insertamos contenido nuevo
285         f.write("\n" + latex_code)
286
287     # Compilar PDF
288     try:
289         subprocess.run(["pdflatex", "post-incident_report.tex"], check=True)
290         print(f"{BOLD_GREEN}PDF post-incident_report.pdf generated successfully.{RESET}")
291     except subprocess.CalledProcessError:
292         print(f"{BOLD_RED} Error compiling the PDF report.{RESET}")
293
294 # ---------------------- Funciones públicas ----------------------
295
296 def generar_reporte_post_incidente(start_str, end_str):
297     if not os.path.exists(os.path.dirname("./post-incident_files")):
298         os.makedirs("./post-incident_files", exist_ok=True)
299
300     data = procesamiento_datos_rango("./procesados", start_str, end_str)
301     if data is None or data.empty:
302         return f"{BOLD_RED} No data has been found. {RESET}"
303
304     generar_warnings_file(data,"./warnings_logs")
305     os.chdir("post-incident_files")      # Nos movemos de directorio para guardar los graficos
306     generar_reporte_pdf("./graficos", generar_graficos(data),
307                         "25/04/2025 14:11:00", "25/04/2025 14:13:00",
308                         data[data["label"] == "Anomaly"]["timestamp"].iloc[0] if not
                            ↪  data[data["label"] == "Anomaly"].empty else None)
309     os.chdir("..")
310     return f"{BOLD_GREEN}PDF report generated successfully. {RESET}"
311
312 def generar_warnings_realtime():
313     now = datetime.now()
314     two_minutes_ago = now - timedelta(minutes=2)
315     data = procesamiento_datos_rango("./procesados", two_minutes_ago.strftime("%d/%m/%Y
        ↪  %H:%M:%S"), now.strftime("%d/%m/%Y %H:%M:%S"))
316
317     if data is None or data.empty or data[data["label"] == 1].empty:
318         return f"{BOLD_GREEN}ALL OKAY :) THERE IS NO WARNING!{RESET}"
319
320     generar_warnings_file(data,"./warnings_logs")
321     return "[!] New warnings have been generated."
322
323 # ---------------------- Código main - ejemplo ----------------------
324
325 modo = input("¿PDF report or Warnings files? (report or warning) ").strip().lower()
326 if modo == "report":
```

```
    start_str = "06/05/2025 01:09:00"   # input("Start time (dd/mm/yyyy HH:MM:SS): ")
    end_str = "06/05/2025 01:11:00"     # input("End time   (dd/mm/yyyy HH:MM:SS): ")
    print(generar_reporte_post_incidente(start_str, end_str))
elif modo == "warning":
    print(generar_warnings_realtime())
else:
    print("Mode unrecognized.")

```

post-incident_files/template_report.tex :

```
%\documentclass[a4paper,12pt, titlepage, twoside]{article}
\documentclass[12pt]{article}
%Cedit a tots els estudiants futurs
\title{Arrow: Post-Incident Analysis Report}
\author{XXX,YYY}
\date{February 2025}

% Language and font encodings
\usepackage[english]{babel} %spanish, english, ...
%\usepackage{lipsum}
\usepackage[utf8]{inputenc}

\usepackage{amsfonts}

\usepackage{parskip}
\setlength{\parskip}{4mm}
\setlength{\footskip}{60pt}
\setlength{\headheight}{15pt}

%% Sets page size and margins
\usepackage[a4paper,top=3cm, bottom=3.2cm, inner=3.2cm, outer=3cm, footnotesep=1cm,
↪  heightrounded]{geometry}

\usepackage{titlesec}

\titleformat{\section}[hang]{\raggedright\LARGE\bfseries}{\thesection.}{0.5em}{}
\titleformat{\subsection}[hang]{\raggedright\Large\bfseries}{\thesubsection.}{0.5em}{}
\titleformat{\subsubsection}[hang]{\raggedright\large\bfseries}{\thesubsubsection.}{0.5em}{}

\usepackage{setspace}
\setstretch{1.2}
%\onehalfspacing % Establece un espacio de 1.5 líneas entre líneas

%\fontsize{12pt}{15pt}\selectfont % Establece un tamaño de letra de 12pt y un interlineado de
↪  15pt
\usepackage{ragged2e}
\usepackage{parskip}

\justifying % Justifica el texto en todo el documento

%% Useful packages

\usepackage{float}
\usepackage{verbatim}
\usepackage{amsmath}
```

```
\usepackage{systeme}
\usepackage{graphicx}
\usepackage{multirow}
\usepackage{caption}
\usepackage{subcaption}
\usepackage{wrapfig}
\usepackage[colorinlistoftodos]{todonotes}
\usepackage[colorlinks=true, allcolors=blue]{hyperref}
\usepackage{titlesec}
\usepackage{fancyhdr}
\usepackage[firstpage]{draftwatermark}
\usepackage{transparent}
\usepackage{textcomp} %podem escriure ``o'' amb la comanda \textdegree també € amb \texteuro
\usepackage[gen]{eurosym} %tb official en lloc de ``gen''
\usepackage[section]{placeins} %Evita que les figures saltin de secció
\usepackage{fancyref}
\usepackage{array}
\usepackage{longtable}
\usepackage{mathtools}
\usepackage{commath}
\usepackage{scrextend}%Indentar un bloc sencer
\usepackage{dtk-logos}
\usepackage{amssymb}
\usepackage{amsmath}
\usepackage{afterpage}
\usepackage{tikz}
\usetikzlibrary{calc, positioning, shapes, backgrounds}
\usepackage[numbers,sort&compress]{natbib}
\usepackage{booktabs}    % Necesario para lineas profesionales
\usepackage{lscape}      % Permite rotar la sección deseada al formato apaisado
\usepackage{changepage}
%Import the bibliography file
\renewcommand\labelitemi{\tiny$\bullet$}
\renewcommand{\arraystretch}{1.15}

\NewDocumentCommand{\myrule}{O{1pt} O{2pt} O{black}}{%
    \par\nobreak % don't break a page here
    \kern\the\prevdepth % don't take into account the depth of the preceding line
    \kern#2 % space before the rule
    {\color{#3}\hrule height #1 width\hsize} % the rule
    \kern#2 % space after the rule
    \nointerlineskip % no additional space after the rule
}
\usepackage[section]{placeins}
\hypersetup{
  linkcolor=black
}
%%Comentaris
\begin{comment}
Aquí podem posar tots els comentaris que calgui sense anar posant %
REFERENCIAT
\label{marker}, \ref{marker} and \pageref{marker} \footnote{footnote text}

\begin{addmargin}[esq]{dta}
El text d'aquí incrementa els seus marges segons ``esq'' i ``dta''
\end{addmargin}
\end{comment}

```

```
%%Comença pàgina nova per cada Secció SI hi ha alguna cosa escrita a la anterior

%%\newcommand{\sectionbreak}{\cleardoublepage}


%%Caps de pagina i peus (E/O (even/odd), L/C/R (left/center/right) y H/F (header/footer))
\fancyhf{}
\fancyhead[ER]{Report}
\fancyhead[OL]{Arrow: Post-Incident Analysis}
\fancyhf[ELH, ORH]{page \thepage}
%\fancyfoot[C]{}
\fancyfoot[EL, OR]{\includegraphics[scale=0.4]{UPC_logo.png}}
\fancyfoot[C]{
    \begin{tikzpicture}[remember picture, overlay]
        \fill[upclight] (current page.south west) rectangle ++(\paperwidth,1.326cm);
        \node[anchor=center] at (0.5\paperwidth,0.75cm) {};
    \end{tikzpicture}
}
\pagestyle{fancy}
\raggedbottom

\SetWatermarkText{\hspace{9mm}\transparent{0.15}\includegraphics[scale=3]{UPC_logo.PNG}}
\SetWatermarkAngle{0}
\SetWatermarkLightness{1}

% Portada mejorada
\usepackage{tikz}
\newcommand{\titlewatermark}{
    \begin{tikzpicture}[remember picture, overlay]
        \node[opacity=0.15, anchor=north west, xshift=2cm, yshift=-4cm] at (current page.north
        ↪  west) {};
    \end{tikzpicture}
}

%\setmainfont{Times New Roman}

\begin{document}

\definecolor{upcblue}{RGB}{0,92,138}     % Azul UPC oficial
\definecolor{arrow}{RGB}{0,0,0}          % Negro Arrow
\definecolor{upclight}{RGB}{200,230,255} % Complementario UPC

\begin{titlepage}

    % Cabecera
    \begin{tikzpicture}[remember picture, overlay]
        \fill[upcblue] (current page.north west) rectangle ++(\paperwidth,-1.5cm);
        \fill[arrow] (current page.north east) rectangle ++(-5cm,-1.5cm);
    \end{tikzpicture}

    {\centering
        \vspace{5mm}
        {\Large \textbf{Arrow Project}}\\
        {\centering
            \includegraphics[scale=0.3]{arrow_logo.png}\\
            % {\Large Universitat Politècnica \\ De Catalunya}\\
            \vspace{3mm}
            \par
```

```
        }
        \vspace{30mm}
        \myrule[2pt][7pt]
        \Huge \textbf{Post-Incident Analysis Report}\\
        \vspace{2mm}
        \myrule[2pt][7pt]
        \vspace{10mm}
    }
    \vspace{15mm}
    \centering
    \begin{minipage}{0.8\textwidth}
        \centering
        \large
        \textbf{Authors:}\\
        David Méndez Giménez, Jesus Daniel Azuaje,\\
        Noa Paguera Contelles, Lucas Mira,\\
        Eva Alisson Roda Rivas, Gabriel Iniesta,\\
        Alejandro Martínez Torre\\

        \vspace{1cm}

        \textbf{Advisors:}\\
        Josep Pegueroles Valles\\
        Beatriz Otero Calviño\\

        \vspace{10mm}
        \large\today\\ %Si e lloc del dia de la darrera edició es vol una data fixa, elimineu
        ↪  \today i poseu la data
    \end{minipage}

    \vfill
    % Pie de portada
    \begin{tikzpicture}[remember picture, overlay]
        \fill[upclight] (current page.south west) rectangle ++(\paperwidth,1.5cm);
        \node[anchor=center] at (0.5\paperwidth,1.5cm) {\large\color{upcblue}};
    \end{tikzpicture}

  \end{titlepage}

\clearpage

\pagenumbering{arabic}

\justifying
\tableofcontents
\newpage
\listoffigures
\newpage

% CONTENIDO PRINCIPAL

\end{document}
```

14.7. Cyber Resilience Act Table

	Description	IDS /IPS	IDRS	SIEM	EDR	Automatic System Updates	Encryption	Notes:
a	Be released without any known exploitable vulnerabilities.	X*	X*	X*	X*			*Prior verification is a prerequisite for this.
b	Be released with the default configuration unless a specific agreement has been reached with the customer.							The default settings must be set at the factory.
c	Provide automatic security updates, with the option to postpone them.	X	X	X	X*	X		*Involves software update.
d	Ensure protection against and report unauthorized access through appropriate control mechanisms, including authentication.	X*	X*	X*	X*			*Reporting of possible unauthorized access is not included in their tasks.
e, m	Ensure the confidentiality of personal data through measures such as data encryption and secure data deletion.						X	
f	Ensure the integrity of data protection, commands, programs and configurations by protecting them from unauthorized manipulation and requiring a report on any corruption.	X*	X*	X*	X*		X	*They are responsible for alerting and reporting, but not for protecting the integrity.
g	Process only information that is strictly necessary, appropriate, and relevant for the specific purpose of the product.			X				
h	Protect basic functions, before and after an attack, to provide solutions.		X		X			
i	Mitigate any adverse consequences that may be exerted on the remaining network devices.		X	X*	X			*SIEM is a monitoring system that cannot act on threats.

j	Be designed, developed, and produced to limit attack surfaces.						Undertake measures to minimize attack surfaces.
k	Ensure a secure design and mechanisms that reduce vulnerability exploitation.		X		X*	X	*Involves software update.
l	Log and monitor key activities.	X	X	X			

Table 8: Table of CRA requirements with brief descriptions

Bibliography

[1] Devo. (2024, November 8). *Threat Hunting Guide: Incident Response Life cycle - Devo.com*. Devo.com. Retrieved from https://www.devo.com/threat-hunting-guide/incident-response-life-cycle/

[2] Empresa, S. I. (2022, September 15). *¿Qué es un sistema EDR, características y funciones*. Banco Santander Impulsa Empresas. Retrieved from https://www.impulsa-empresa.es/que-es-sistema-edr/

[3] European Union. (2024, November 20). *REGLAMENTO (UE) 2024/2847 DEL PARLAMENTO EUROPEO y DEL CONSEJO*. Diario Oficial De La Unión Europea. Retrieved from https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=OJ:L_202402847#tit_1

[4] European Union. (2024, November 20). *Regulation (EU) 2024/2847 of the European Parliament and of the Council*. Official Journal of the European Union. Retrieved from https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L_202402847#tit_1

[5] Gyongyoși, L. (2025, January 13). *The complete guide: How to create an Endpoint Detection and Response (EDR) Strategy*. Heimdal Security Blog. Retrieved from https://heimdalsecurity.com/blog/edr-strategy-guide/

[6] IBM. (2024a, July 15). *¿Qué es un IDS?* IBM. Retrieved from https://www.ibm.com/es-es/topics/intrusion-detection-system

[7] IBM. (2024b, July 16). *¿Qué es un sistema de prevención de intrusiones (IPS)?* IBM. Retrieved from https://www.ibm.com/es-es/topics/intrusion-prevention-system

[8] IBM. (2024a, July 19). *¿Qué es la detección y respuesta de endpoints (EDR)?* IBM. Retrieved from https://www.ibm.com/es-es/topics/edr

[9] IBM. (2024b, November 8). *¿Qué es la Gestión de Eventos e Información de Seguridad (SIEM)?* IBM. Retrieved from https://www.ibm.com/es-es/topics/siem

[10] IOMT Dataset 2024 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (n.d.). Retrieved from https://www.unb.ca/cic/datasets/iomt-dataset-2024.html

[11] *¿Qué es SIEM? | Seguridad de Microsoft*. (n.d.). Retrieved from https://www.microsoft.com/es-es/security/business/security-101/what-is-siem#:

~:text=Preguntas%20m%C3%A1s%20frecuentes-,Definici%C3%B3n%20de%20SIEM,a%20las%20operaciones%20del%20negocio

[12] *¿Qué es un sistema de prevención de intrusiones?* (n.d.). Palo Alto Networks. Retrieved from https://www.paloaltonetworks.lat/cyberpedia/what-is-an-intrusion-prevention-system-ips

[13] ScienceDirect. (n.d.). *Intrusion Response System.* ScienceDirect. Retrieved from https://www.sciencedirect.com/topics/computer-science/intrusion-response-system#chapters-articles

[14] *The EU Cyber Resilience Act - what you need to know.* (n.d.). A&O Shearman. Retrieved from https://www.aoshearman.com/en/insights/ao-shearman-on-tech/the-eu-cyber-resilience-act-what-you-need-to-know

[15] *What is SIEM?* (n.d.). Palo Alto Networks. Retrieved from https://www.paloaltonetworks.com/cyberpedia/what-is-siem

[16] ManualDeLATEX *Listas y enumeraciones en LaTe* Manualdelatex.com. Retrieved from: https://manualdelatex.com/tutoriales/listas-y-enumeraciones