D5.4

HEIR Integrated framework final version

Project number	883275
Project acronym	HEIR
Project title	A secure Healthcare Environment for Informatics Resilience
Start date of the project	September 1st, 2020
Duration	36 months
Programme	H2020-SU-DS-2019
Deliverable type	Demonstrator
Deliverable reference no.	D5.4
Workpackage	WP5
Due date	02-2023-M30
Actual submission date	02/05/2023
Deliverable lead	ITML
Editors	George Tsakirakis, Panagiotis Rodosthenous, Nikos Dimakopoulos (ITML)
Contributors	Dragoș Gavriluț, Bogdan Prelipcean (BD), Iulia Ilie (SIE), Andreas Alexopoulos, Leonidas Kalipollitis, Chronis Ballas, Miltiadis Kokkonidis (AEGIS), Michalis Smyrlis, Andreas Zacharakis, Iordanis Xanthopoulos (STS), Eliot Salant (IBM), Marwan Darwish Khabbaz (TUD)
Reviewers	Dimitrios Karamitros (WEL), Iulia Ilie (SIE)
Dissemination level	PU
Revision	FINAL 1.0
Keywords	Cybersecurity, Privacy, Integrated solution, Security, Information security technologies, Secure data flow

Abstract

This deliverable describes all integration activities and outcomes that have been performed during the 2nd half of the project. It provides an abstract presentation of the HEIR modules (final versions) that have been integrated for the final version of the framework including the final conceptual specification of the HEIR framework architecture and deployment. The HEIR modules are grouped into three architecture layers, namely, the Technology Facilitators layer, the Core Framework layer and the Visualization layer. Also, it depicts the communication between HEIR modules based on the current use case scenarios/playbooks elaborated in WP6 and the results of the technical testing that was conducted.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883275

1 Executive Summary

The HEIR Integrated Framework aims at providing a set of innovative ICT modules integrated to support (i) Real-time threat-hunting services, facilitated by advanced machine learning technologies; (ii) Policy-driven data sharing facilitated by the HEIR privacy-aware framework; (iii) Innovative Benchmarking based on the calculation of the Risk Assessment of Medical Applications (RAMA) score and (iv) The delivery of an Observatory for the security assessment of all participating healthcare entities and their assets.

This document is an accompanying report to the Demonstrator of the Final Integrated Framework of the HEIR project. It includes the final full set of the required functionalities covering all four pilots of the project by incorporating feedback received during the previous evaluation periods while encapsulating all the achievements, updated modules and overall technical work performed in work packages 3 and 4. This final version of the HEIR framework acts as the testbed for the HEIR stakeholders to experience HEIR capabilities and assess the concepts and knowledge conveyed by the project.

The HEIR Integrated Framework, far from being a simple container for the individual modules, is a coherent solution, where several different modules reside and seamlessly collaborate. The final version of the HEIR framework is a step further from the initial version concept (D5.3) for the HEIR end users. It encapsulates the underlying technologies and gives a clear and easy-to-use graphical interface, exposing every available feature so far.

The main architectural and deployment aspects of the HEIR framework have not changed significantly since the initial version but are described in the current document for completeness. It should be noted that the benefit of the final architecture is that any additional functionality can be wrapped into a separate component and added to the framework, provided that it abides by the basic communication standards exposed by the HEIR framework architecture. The scope of this practice is to keep the HEIR framework evolving and enable future extensions to develop functionalities, which will maximize the potential for further exploitation and adoption of the framework beyond the end of the project.

Table of Contents

List of Figures

List of Tables

2 Introduction

2.1 Scope and Objectives

This deliverable is the third deliverable of T5.2 (Continuous integration towards the realisation of HEIR framework) which emphasizes on the realization of the final complete version of the HEIR integrated framework. Generally, the scope of WP5 is an end-to-end integrated cybersecurity framework for healthcare systems with the objective to (i) design and develop the HEIR secure data fusion and management infrastructure, (ii) implement the integrated HEIR framework that realises the envisioned HEIR technology convergence.

For the final complete integrated version and as an updated version of the 1st integrated version (D5.3), the various HEIR modules developed were integrated into a unified solution which is deployed in pilot partners' infrastructure and in the HEIR Observatory infrastructure provided by ITML. The objective of the document is to showcase the final integrated version of the HEIR framework that enables detection of cyber threats, unauthorised access to the network, and analysis of relative information through the visualization tools that were deployed. In this way, end-to-end security and privacy can be ensured in a healthcare environment.

2.2 Document Structure

This document reports on the activities and effort placed in the integration of the various technologies and tools towards delivering the final version of the HEIR Integrated framework. Following the HEIR approach, the integration effort is guided from the Agile Software Development methodology, aiming to progress the development work in parallel teams and regularly integrating their output, based on a well-defined design. The scope of this document is to act as an accompanied report to the final demonstration version of the HEIR integrated framework and, as such, it is structured as follows:

Section 3: Presents the final System Architecture which includes the architecture diagram, the deployment diagram, the HEIR modules (the facilitators, the core framework modules and the HEIR visualization modules) and the sequence diagrams that are based on the current use cases scenarios/playbooks implemented under WP6.

Section 4: Documents the integration methodology and specifications including the technical testing results.

Section 5: Provides the final conclusions regarding the HEIR integrated framework.

2.3 Relation to other Tasks and Work Packages

This deliverable is linked with the technical WPs: WP2 (The HEIR facilitators), WP3 (HEIR client and aggregator) and WP4 (HEIR Observatory). Moreover, there is a close relation with WP6 (HEIR real-life demonstration and validation) which defines and crystallizes the use case scenarios/playbooks, the demonstration and validation of the HEIR solution.

3 System Architecture

3.1 Architecture Diagram – Logical view

The presentation of the HEIR Integrated framework architecture follows a logical path from the use cases supported by the solution to the functional conceptualization of the needed modules, and ultimately to the software modules and tools that will be used for the implementation of these functionalities.

Using the technology convergence – architecture revision and tools specifications described in D1.3, the actions and sequences that implement the processes necessary for serving the current use case scenarios/playbooks (the final ones will be described in D6.2), were defined and explained in section 3.4 of the current deliverable. These system use cases reveal the degree of complexity and needs for modularity, communication and orchestration of the various modules that will be integrated within the HEIR integrated framework. The architectural design aims to cover all these intricacies, while maintaining the openness of the system, ensuring that it will be scalable and easily modifiable. Furthermore, all the design and implementation decisions are grounded in established technology and industry standards. We consider a design as successful when it covers the following aspects (i) Usability, (ii) Performance, (iii) Security, (iv) Maintainability, (v) Adaptability and (vi) Portability. The HEIR modules are responsible for a specific set of functionalities. By convention, the different layers interact with each other in a top-down manner. The implemented architectural design addresses all the aspects that we are targeting. Together with other software architectures and standards that are followed, it helps us to apply the best standards in all the design aspects we are focusing on. More specifically:

Usability: The fact that we are following the defined architecture isolates the presentation components (1st and 2nd layer of visualizations), giving the possibility to focus on good User Experience design (UX). Thus, a web designer or a usability expert can work separately on the User Interface unaffected by the backend system developers. These experts can focus on the User Interface to maximize the quality of user experience. Internally we followed a Model View Controller (MVC) design pattern for building a web application, starting from a plain, well-defined user interface which consumes the services provided by the backend. The use of modern web technologies for advanced visualisations (e.g., Hicharts.js[1], plotly[2] or other visualisation library) and interactive, responsive dashboards (e.g., React.js or Angular.js) offered the best set of front-end features to provide a clean and fully functional interface. Finally, an agile methodology was followed for developing the HEIR framework, based on rapid prototyping and frequent iterations. As a result, there were more frequent assessments conducted in close proximity to the end-user of the solution, leading to improved outcomes in meeting the usability requirements.

Performance: For tackling performance issues, we relied on two factors – caching and distribution. The backend implementation of the framework architecture logic is modular and communicates through Apache Kafka, an open-source distributed event streaming platform for high-performance data pipelines. Module services offer parallelization in calls to the backend and can be deployed independently in a distributed way, following a Software Oriented Architecture. Load balancing and caching mechanisms are available and can be applied, if needed.

[1] https://www.highcharts.com/
[2] https://plot.ly/

Security: The security features will be applied system-wise, covering the whole architecture. An Access Control and Identity Management system ensures that only users with appropriate permissions will be able to access the data relying in the solution's data.

Maintainability: To ensure maintainability, we are adhering to an architecture that is focused on standards and independent of any specific technology. For the communication between the various developer teams, we enabled communication channels via emailing lists and regular telcos which were utilized on the second half of the project.

Scalability: The system's ability to handle requests will increase proportionally with the volume of requests. The distributed, service-oriented nature of the backend allows for easy scalability of the architecture. Scalability in terms of data is also required for HEIR mainly because of potential big volumes of data that will be analysed and visualised. Due to the nature of the monitored data (e.g., medical devices), scalability might be restricted due to the limits imposed by the original data source, i.e., limitation in API calls of a source. Such kind of limitations were examined during the project.

Reliability: To build a reliable system, a certain number of characteristics must be considered. These characteristics include maturity, availability, fault tolerance and recoverability as described in the software quality model of the BS ISO/IEC 25010:2011[3] standard. Some of the modules of the HEIR integrated framework are based on existing solutions that have been used in the past and will only require some adaptations to serve the needs of the users, therefore they are mature enough to be part of a reliable system. Furthermore, the modules that are created for the purposes of HEIR are also based on widely used technologies or pre-existing tools which can be easily supported by the owners or the community in the case of open-source solutions.

The modules of the framework are designed in a way that tries to help the end-users avoid mistakes and misuse of the offered functionalities. Nevertheless, errors are always a possibility, so each component incorporates an internal error handling mechanism to be tolerant of misconfiguration or malicious input. Furthermore, the loosely coupled architecture of the framework avoids points of single failure and provides the ability to have a working production framework even if one of the modules temporarily fails to perform adequately. For example, temporary failure of a module would result into its output being unavailable but not in bringing the whole framework to a halt. The following architecture diagram of the present state of components' interconnections is shown in Figure 1.

[3] https://www.iso.org/standard/35733.html

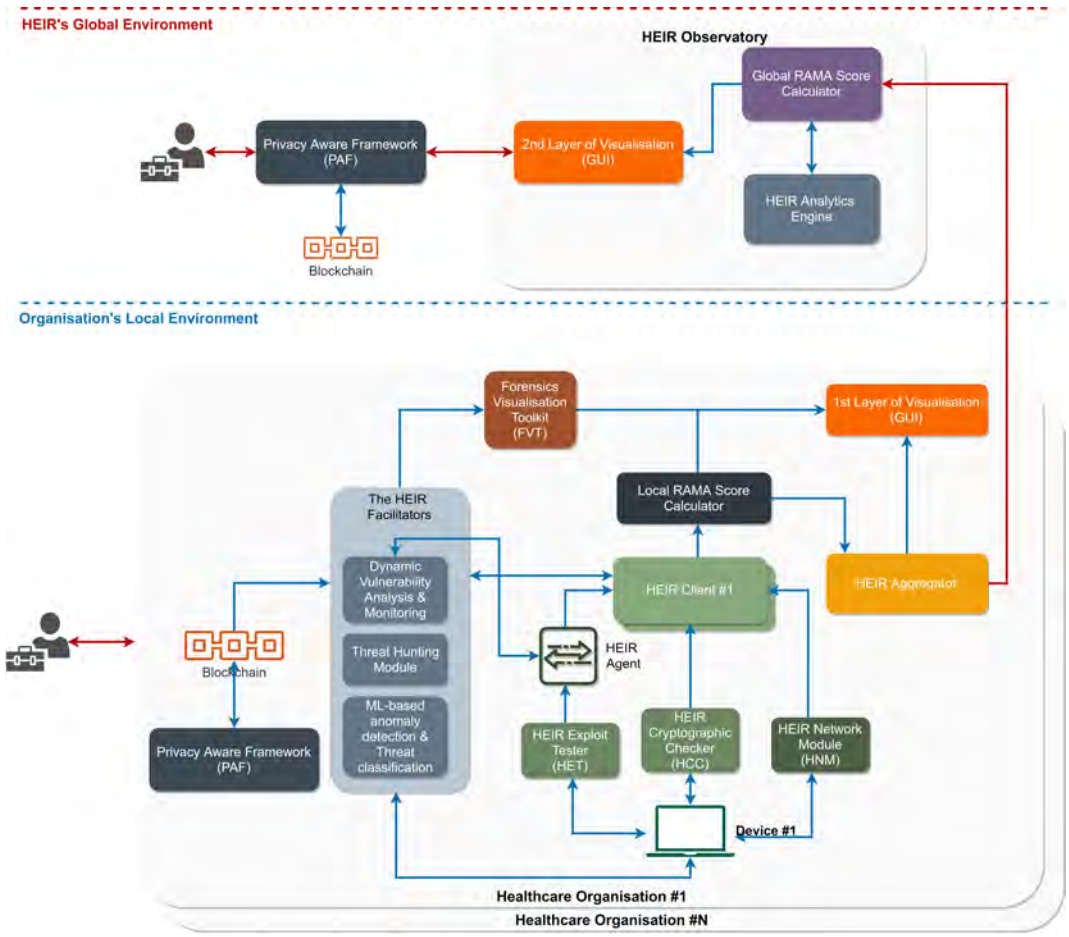

Figure 1: Architecture diagram

In all pilot use-cases data is collected by the HEIR agents installed in servers/workstations/devices and web traffic intercepted by the HEIR Client network module. Wherever pertinent data can be obtained, for instance Hospital Information System (HIS) data, the ML-based Anomaly Detection & Threat Classification component is employed.

Data deriving from the SIEM agents are portrayed on the Forensics Visualization Toolkit (FVT) which serves as a GUI for the threat hunting module for use by the hospital IT administrators. Furthermore, this threat hunting module data, in conjunction with data generated by the HEIR agents relayed via Dynamic Vulnerability Analysis & Monitoring, are fed to the HEIR Client processing system. Subsequently, the data is utilized to determine the RAMA score of an individual department. If the Pilot healthcare environment contains more departments, the individual RAMA scores per department are aggregated by the HEIR aggregator component so that a Hospital RAMA score is calculated.

The above RAMA security score and metadata is communicated to the HEIR Observatory database and is used by the HEIR Analytics Engine to produce the Global RAMA score, which is also communicated back to the hospital (1st layer of visualization). Finally, the RAMA scores and metadata are portrayed in the Observatory graphical interface (2nd layer of visualization).

3.2 Deployment Diagram

The following deployment diagrams provide information about the framework deployment topology. The modularity of the HEIR integrated framework enables the use of tailored deployments of the framework per pilot. So, while all four pilots share the fundamental set of solution functionalities, different combinations of the HEIR individual modules are employed

per pilot. So, in cases where relevant data is obtainable (such as the Pagni and the Croydon pilot sites), the ML component is utilized, while in NSE where patient related data is available the PAF component is employed. The deployment diagram of a local HEIR client (i.e., Pagni pilot site) is depicted in Figure 2.

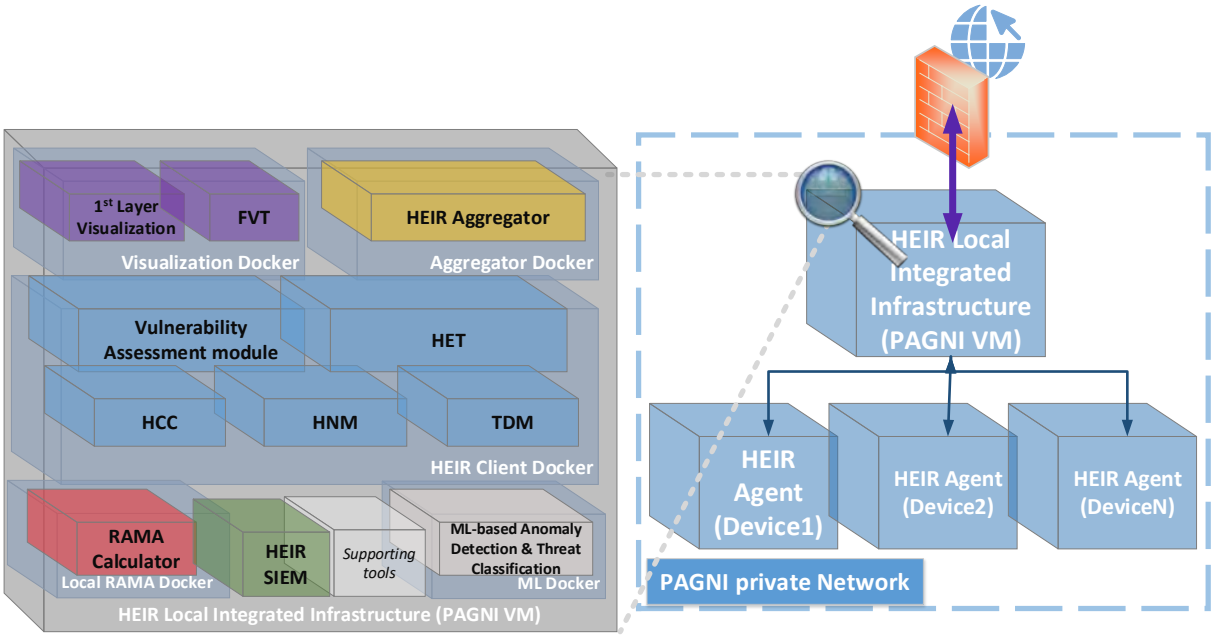

Figure 2: Local deployment diagram

The global integrated deployment diagram of the HEIR eco-system containing multiple HEIR clients is shown in Figure 3.

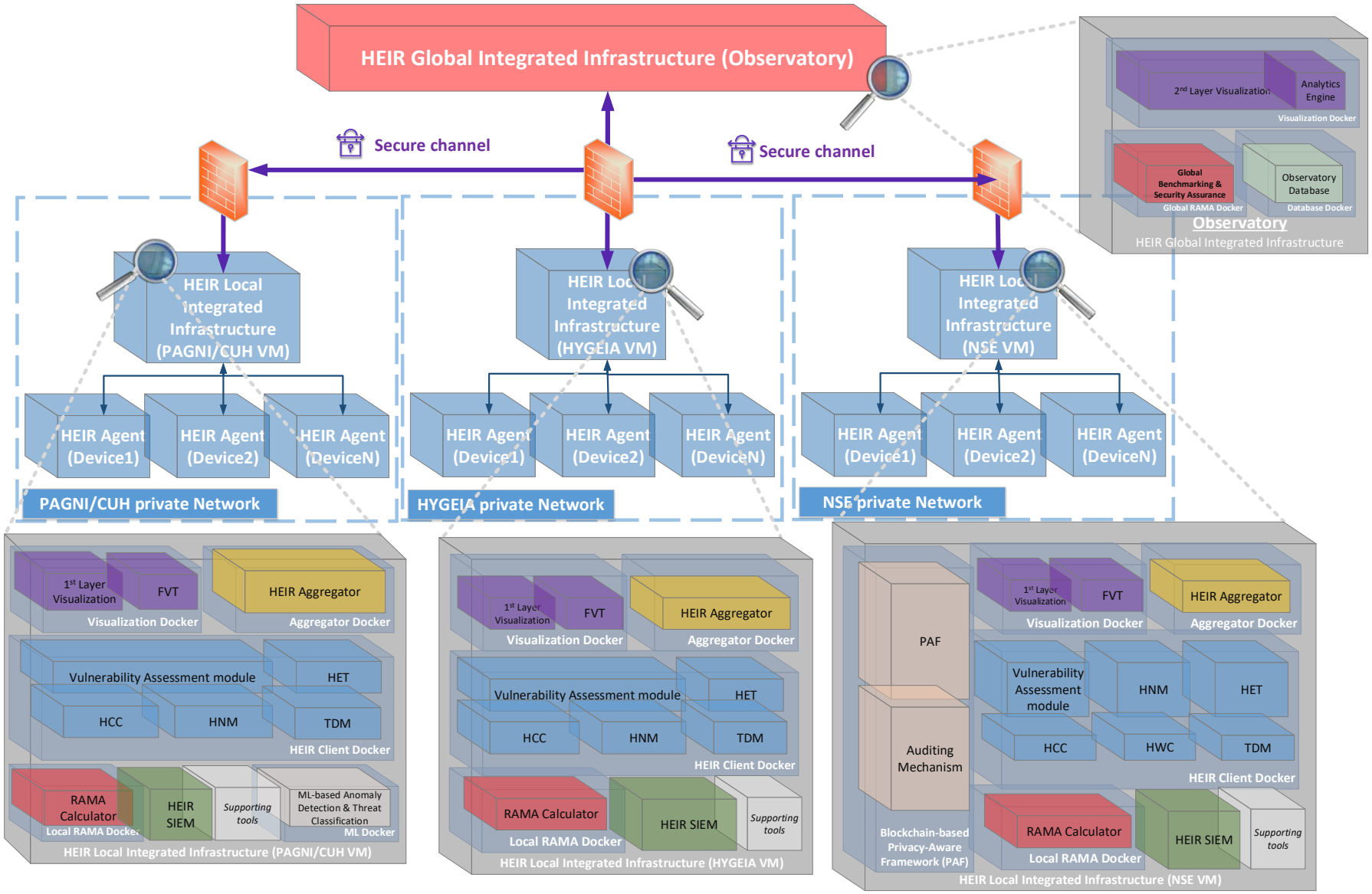

Figure 3: Global integrated diagram

3.3 The HEIR Modules

The subsections that follow provide a concise way of depicting the various modules that have been developed and integrated in the context of the HEIR project. The particulars given for each module are beneficial for acquiring a comprehensive overview and making it simple to grasp the technologies involved in all the end-use cases. More detail regarding the individual modules hereby described can be found in the related deliverables D2.3 (The HEIR facilitators package: Final complete version), D3.3 (The HEIR 1st layer of services package, final version) and D4.3 (The HEIR 2nd layer of services package, final version).

3.3.1 HEIR Facilitators

3.3.1.1 Forensics Visualization Toolkit (FVT)

The FVT provides users with a timeline-based representation of the security events for each department and the corresponding connected devices of the hospital that are captured by the sub-modules of the HEIR's environment (e.g., SIEM, RAMA Calculator, ML). It is accessed through the 1st layer of visualizations and represents the logged events in a more detailed way, including advanced visualization widgets (time-wise synchronized), filtering capabilities and configurable views. Authorized users (by the HEIR access-controlled system, see Sect. 4.4) who belong to the hospital staff and have access to the HEIR Client GUI (HCG) can further investigate any of the connected HEIR Clients of the hospital through the FVT.

Module's interfaces		
Input		
Name	Type	Short Description
Department's local RAMA score & Metadata (RAMA Calculator)	JSON	Selected department's RAMA score & metadata. Metadata refers to the output of the HEIR Client's modules. (Vuln. Assessment, HCC, HNM, HET, security status information and more.) Source: RAMA Score Calculator.
Devices' logged events (SIEM)	JSON	SIEM's reported events for the connected devices of the department. Source: HEIR SIEM.
Anomaly Detection Module's results (ML)	JSON	Processed events' description and anomaly probability score for the selected department. Source: Anomaly Detection Module.

Table 1: HEIR FVT interfaces

3.3.1.1.1 HEIR SIEM

The HEIR SIEM component supplies various security related data from all endpoints to the HEIR Interactive Forensics Module, AEGIS' FVT, as well as BD's HEIR client. This event data is collected from lightweight agents installed on the endpoints (i.e., workstations, servers etc.). The SIEM solution used in HEIR is built around ITML's SAAS product, named Security Infusion[4], which incorporates a plethora of open-source software tools in a unified framework.

It is based on the Wazuh[5] open-source solution which provides a multitude of security related services that continuously monitor an IT infrastructure. All data is collected by lightweight

[4] https://security-infusion.com/
[5] https://wazuh.com/

agents which run on the monitored systems, collecting events, and forwarding them to the Wazuh Manager, where data is aggregated, analyzed, indexed, and stored. This ensures that the resources needed at the client level are kept to a minimum since the security intelligence and data analysis is solely performed at the server level. Wazuh clients run on many different platforms, including Windows, Linux, Mac OS X, AIX, Solaris and HP-UX. The events reported by the Wazuh agents are the outcome of a wide range of tasks such as:

- Inventory of running processes and installed applications
- Log and events data collection
- File and registry keys integrity monitoring
- Monitoring of open ports and network configuration
- Configuration assessment and policy monitoring

These events are received by the Wazuh server and processed through a toolset of decoders and rules, using threat intelligence to look for well-known IOCs (Indicators Of Compromise). As a result of this analysis, all events are appointed a severity level enabling the administrators to focus on the crucial issues that need to be addressed. This is further delivered via customized alerts that are sent to an Elastic Stack[6] which also provides a powerful interface for data visualization and analysis via its integration with Kibana[7]. In addition to logs and events deriving from the operating system, Wazuh can collect and integrate logs deriving from network devices such as routers, firewalls etc. either by monitoring the log files themselves or via forwarding log messages in through Rsyslog[8]. This can potentially facilitate the collection of logs from medical devices that will need to be monitored within the hospital use-case environments. The Wazuh event flow management is depicted in Figure 4.

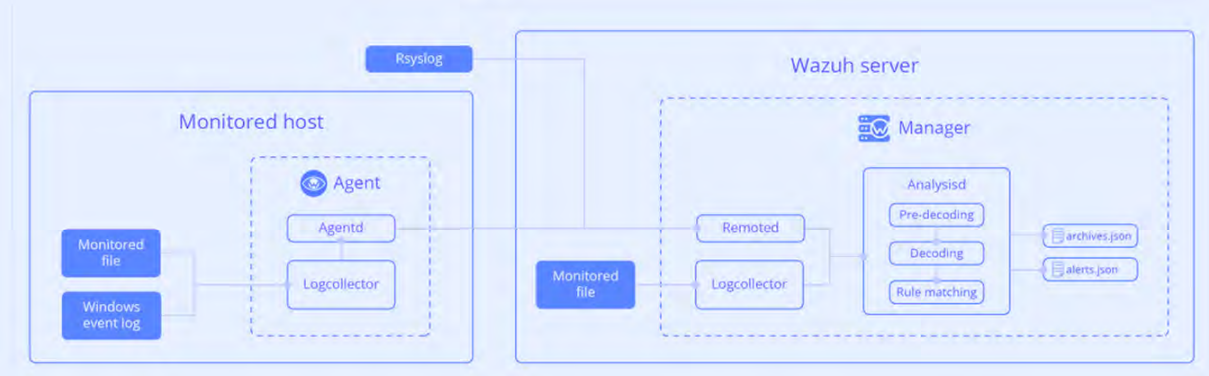

Figure 4: Wazuh event flow management

Furthermore, Wazuh offers a powerful RESTful API that allows the interaction of the Wazuh manager with web browsers, command line tools like cURL[9], or any scripts or programs that can make web requests. This, combined with the RESTful APIs provided by ElasticSearch, will greatly aid the seamless accumulation of the HEIR SIEM security metrics in the integrated HEIR Client and its role in the RAMA score calculation.

It should be noted that the verbose functionality and abilities of Wazuh are further complemented by the parallel use of the deployed Security Infusion agent enabling the accumulation of extra metrics like Resource Allocation (CPU/Memory/Disk) analysis of

[6] https://www.elastic.co/elastic-stack/
[7] https://www.elastic.co/kibana/
[8] https://www.rsyslog.com/
[9] https://curl.se/

computer processes and open file handles etc. This provides an even more verbose real time monitoring of the computer endpoints, and additionally provides valuable data to be used post event for forensic purposes.

In the 1st iteration of the SIEM component all the security related events were solely used in feeding the FVT dashboard, enabling a security administrator to recognize potential security incidents. To this end, by collecting data originating from the Security Infusion agent such as the endpoint's resources utilization, it provides the ability of performing a thorough forensic analysis by correlating the reported events with additional computer metrics (i.e., Disk & Memory utilization etc.). However, the real-time automatic reporting of high severity detected event was not yet implemented. In the 2nd iteration of the SIEM component, an extra Elastic Connector component was deployed which extracts all high severity events from the Wazuh data stored in the Elastic Search and sends them to a Kafka topic which is monitored by the HEIR client, adding one more source of security related events that are used to calculate the RAMA score.

Module's interfaces		
Input		
Name	Type	Short Description
Windows Logs	Application, Security, System	Windows logs of application and system messages, including errors, information messages, and warnings for troubleshooting all kinds of different Windows issues, security or other.
3rd party Logs	Date-based logs, Syslog Logs, IIS Logs tec.	Wazuh agents are able to accumulate 3rd party logs from computers and network devices and parse them based on their format (i.e., Web Server Logs, Syslog Logs from Network Devices etc.)
Output		
Name	Type	Description
Wazuh API REST[10]	RESTful API	Allows for interaction with the Wazuh manager from a web browser, command line tool like cURL or any script or program that can make web requests.
ElasticSearch	E.S Indices	ES indices are relational databases with individual mapping which defines multiple types.

Table 2: HEIR SIEM interfaces

3.3.1.2 HEIR Machine Learning (ML)-based Anomaly Detection & Threat Classification module

This module provides efficient event and threat data classification based on specific rules related to cyber security requirements and cyber-threats level of criticality and novel machine-learning (ML) models. In particular, adaptations of existing ML models utilized in anomaly detection and/or threat classification are incorporated, which match the requirements of the health systems. The ML module takes the input from HEIR IoT (Logs) and processes the records to differentiate between anomalies and non-anomalies. After that, the ML component

[10] https://documentation.wazuh.com/current/user-manual/api/reference.html

processes the results in a detailed report. The result is visualized in the FVT toolkit to represent the results tangibly. The selected model/algorithm based on supervised/ unsupervised learning algorithms depends on the use case (PAGNI, CROYDON). Each use case has its own model which was built to suit the data structure and classify the anomalies in a good order. Both use cases are explained below in detail:

3.3.1.2.1 The PAGNI Use Case

The approach followed for the PAGNI use case is based on a voting mechanism; we use six cutting-edge algorithms, collect the results, and then aggregate them. We determine the severity of the event or probable anomaly based on the vote results. For the implementation the following algorithms have been taken into consideration **KNN (K-Nearest Neighbours)**, **SUOD (Single-class Unsupervised Outlier Detection)**, **PCA (Principal Component Analysis)**, **LOF (Local Outlier Factor)**, **COPOD (Cluster-based Outlier Probability Density)**, and **HBOS (Histogram-based Outlier Score)** and in order to proceed with the voting engine of the machine-learning component, several steps have be taken beforehand to clean, format, and initialize the data correctly. Below are in brief the steps followed:

a) **Data Preprocessing:** The provided dataset from 01.01.2016 to 08.31.2017 contains **2,497,457 records**; The different features for actions related to the hospital are shown in the below table:

Column Name	Description
Hospital	indicates the hospital where the action took place.
Role	indicates the person who performed the action
Action	indicates the action that was performed
VPN	indicates whether or not the action was performed using a VPN
Department	indicates in which department the specific role is located
Date Time	indicates the date and time the action was performed

The values from role and action had to be pre-processed and transferred to integer values without any logic because they were encoded as strings. Furthermore, the date time column was divided into three shifts to cover 24 hours in the hospital **(First shift at 07:00–17:00; Second shift at 17:00–23:59, and Third shift at 00:00–06:59)**.

b) **Model Building:** All the unlabeled data from the hospital, role, action, VPN, and datetime columns were input to all algorithms (KNN, SUOD, PCA, LOF, COPOD and HBOS).\

c) **Data Outcome:** After training all six algorithms with the archived data, the voting system can capture and classify the anomalies according to a probability counter-set for each record in the provided dataset. The maximum number of votes was six, whereas the minimum one was four votes. The table below explains the outcome details of the PAGNI archived dataset. The total percentage of anomalies captured out of total events, including all votes, was ~**1,6%.**

Votes	Anomalies
1	40
2	10.040
3	25010
4	3231
5	14
6	6

3.3.1.2.2 The CROYDON Use Case

The approach followed for the CROYDON use case was to use supervised learning. The training data used to train the model covers the regular/irregular patterns (**50 regular patterns / 90 irregular patterns from different series of time**) from the other samples of CTG data of Team3 fetal monitoring devices. The algorithm used is LSTM (Long Short-Term Memory), a Recurrent Neural Network (RNN) commonly used for analyzing time series data and solving tasks related to sequential information, such as speech recognition, language translation, and stock price prediction. In this case, it can classify signals captured from Team3 devices and detect anomalies. LSTM networks are trained in the data sequence and use memory cells to capture long-term dependencies in the input data, making them well-suited for time series analysis. Furthermore, various evaluation techniques were implemented to obtain the highest resolution. The features that were used are CTGDataId, CTGId, OffsetSN, Version, BucketIndex, BucketOffsetSN, IsStartOfBucket, IsEndOfBucket, IsEndOfTrace, FHR, TOCO, MHR, Status, LastModified. Below is the context of some essential features that were used to train the model:

- **FHR (Fetal heart rate):** A regular fetal heart rate typically ranges from 110 to 160 beats per minute (bpm).
- **TOCO (tocodynamometry):** is a measure of uterine contractions. A regular TOCO reading would be consistent with the normal uterine contractions described above.
- **MHR (maternal heart rate):** the mother's heart rate. A normal MHR typically ranges from 60 to 100 bpm.
- **Last Modified:** This column indicates the date and time the action was performed.

Model Objective

Multivariate Time Series Classification: An anomaly might be due to sudden changes in various signals. For example, a sudden change in fetal heart rate (e.g., a rapid increase or decrease) could be an anomaly. The table below represents both the input used for PAGNI/CUH and the machine learning output to post the results in a tangible way using the FVT toolkit.

Module interfaces		
Input		
Name	Type	Short Description
HEIR LOG	SQL, CSV (provided by the partners)	SQL log was provided, and it was changed into CSV format in order to be processed within the ML component. The PAGNI file contains (id, user, hospital, department, role, action, caseID, vpn, datetime). The CUH file contains (id, CTGId, FHR1, FHR2, FHR3, TOCO, MHR, Status, LastModified).
Output		
Name	Type	Description
ML output for PAGNI	JSON (provided by machine learning component)	JSON file was generated as an outcome from ML component to be used in AEGIS side by creating UI tile like SIEM. The file contains (id, user, hospital, department, role, action, caseID, VPN, datetime and Anomaly) + ML score in the bottom of the JSON format. For Instance (from one use case):

		{"id": "100000", **"user": "1188",** **"hospital": "2",** **"department": "102",** **"role": "Doctor",** **"action": "Nursing instructions",** **"caseID": "2051641978",** **"vpn": "0",** **"datetime": "2016-01-27 22:12:25",** **"Anomaly": "NO"}**
ML output for CROYDON	JSON (provided by machine learning component)	JSON file was generated as an outcome from ML component to be used in AEGIS side by creating UI tile like SIEM. The file contains (id, CTGId, FHR1, FHR2, FHR3, TOCO, MHR, Status, LastModified and Anomaly) + ML score in the bottom of the JSON format. For Instance (from one use case): {"Anomaly": "NO", “Signal”:[{ "id": "100000", "CTGId": "1188", "FHR1": "698", "FHR2": "699", "FHR3": "699", "TOCO": "21", "MHR": "388", "Status": "0", "LastModified": "2016-01-27 22:12:25"}, { "id": "100001", "CTGId": "1189", "FHR1": "698", "FHR2": "698", "FHR3": "698", "TOCO": "20", "MHR": "387", "Status": "0", "LastModified": "2016-01-27 22:12:25"}]}

Table 3: HEIR ML-based Anomaly Detection interfaces

3.3.1.3 Blockchain-based Privacy-Aware Framework (PAF)

The goal of the PAF is to provide a secure path to a data source, where data access is controlled by a set of policies typically provided by an organization’s Governance Officer. The PAF is built on top of the Open Source Fybrik framework (https://fybrik.io/), which in turn is built on

top of leading Open Source technologies such as Kubernetes and Istio for service mesh implementation, and Open Policy Agent (https://www.openpolicyagent.org/). In the second half of the project, the PAF focused on expanding its functionality to cover a new set of use cases which arose from discussions with the use case partners.

The "Consent" use case tackles the issue of how patient consent to share their records can be incorporated into the PAF mechanism. This use case demonstrates how the FHIR Consent resource within the hospital's FHIR server can be used to hold consent conditions on an individual patient basis. It has been created a scenario where a patient can grant sharing of their Observation records for research within a given time window. Whereas without the PAF, any researcher with access to the FHIR server can access all Observation records, the policy rules in PAF were defined to return to the HEIR Fybrik module a new data redaction/transformation action called "Consent" if the accessor is a Researcher. The HEIR Fybrik module was extended to accommodate this new action, which checks the consent permission associated with each Observation record to be returned. Records without an associated patient's consent, or which fall outside of the consent time window (i.e., the Observation was performed outside of the time window) have all PII information redacted before being returned. Records which have the correct consent permissions and timestamp are returned unredacted.

A second major use case implemented in the second half arose from discussions with a clinician from the Norwegian Diabetes Registry team. There is a large number of healthcare registries distributed across Norway, however each registry is individually managed. This means that a clinician or researcher who wants to cross-correlate data between registries (for example, the Diabetes registry with the Prescription registry) needs to fill out a large number of request forms and wait to obtain access approval – a process that typically takes upwards of a year.

HEIR tackled this by prototyping a solution where the PAF transparently federates the distributed registry data stores and issues queries for data across all the linked data stores. PAF then applies its policy-driven data transformation/redaction actions across the returned federated data set. Our prototype was directed at two main actors – Patients and Researchers. A patient can obtain all of their records from all of the linked registries, essentially combining these formerly siloed data stores into an integrated electronic healthcare record. Researchers can request records from across this federated linkage, however PAF policy causes PII information in the returned records to first be anonymized by use of a hashing algorithm. As in our other use cases, users identify, and role is embedded in a cryptographic token (JSON Web Token) which must accompany every request for data.

3.3.1.3.1 Blockchain-based HEIR Auditing Mechanism

The goal of the HEIR Auditing mechanism is to provide tamper-resilience and, thus, practical immutability for critical, data access related logs that are generated through the HEIR PAF. The HEIR Auditing mechanism is built on top of the Hyperledger Fabric framework, an established and production-ready blockchain framework, whose main business functions are performed via smart contracts. For the purposes of the intermediate version of the HEIR framework, the Auditing mechanism consists of a simple Fabric network of one ordered node and one peer node (where the chain code and the ledger data reside), along with a front-facing client application that acts as a single point of entry to the network modules and the smart contracts deployed in them. A dedicated smart contract that performs all management aspects for audit logs is implemented with the Fabric Java SDK and converted to chaincode.

This Spring Boot-based client application is taking advantage of the Fabric Gateway SDK. The Fabric Gateway SDK allows a client application to perform and expose not only internal to the network functions (such as Fabric user management) but also the services exposed by the

deployed smart contracts. In addition, the client application is equipped with a Kafka consumer, that allows for direct integration with the PAF. The Kafka consumer listens on a specific Kafka topic for incoming audit logs, processes and stores them in the ledger. Various queries/filtering operations are exposed by the smart contract itself and can be accessed via the REST API. Examples include queries based on userID, outcome, intent, executed and the time range. A description of the Auditing mechanism interfaces is presented below.

Module's interfaces		
Input		
Name	Type	Short Description

Kafka Consumer	JSON	{
		"Timestamp": "", timestamp when access request was received
		"Requester": "", username or SUB field of the passed JWT
		"Query":"", Query in the REST request
		"ClientIP": "", origin IP of the request
		"assetID":"", Corresponds to requested dataset,
		"policyDecision": "", corresponds to the policy decision that authorised or blocked the data access request
		"intent": "", Declared at Fybrik invocation
		"Outcome": "","AUTHORIZED" or "UNAUTHORIZED"
		}

Output		
Name	Type	Description
REST API	JSON	Exposes the complete functionality supported by the deployed smart contract with respect to filtering and querying operations on the ledger data.

Table 4: HEIR Blockchain-based Auditing Mechanism interfaces

Originally based on a Docker-based deployment for local testing and integration purposes, the Auditing mechanism has now been reconfigured to be deployed in a Kubernetes [11] environment, tightly integrated with the PAF.

[11] https://kubernetes.io/

3.3.2 HEIR Core Framework

3.3.2.1 HEIR Client's Processing system (HEIR Client)

This is the component that integrated facilitator modules.

Module's interfaces		
Input		
Name	Type	Short Description
HEIRClient	JSON	Configuration file with hospitalId and kafkabroker.
Output		
Name	Type	Description
HEIRClient	JSON	Aggregate output for HET, VA, HCC, HNM.

Table 5: Novel HEIR Client interfaces

3.3.2.1.1 Network module (HNM)

Monitors the network for critical issues, attacks, malware and information leaks.

Module's interfaces		
Input		
Name	Type	Short Description
HNM	JSON	Configuration file with network selection and ethernet device along with client ID, hospital ID, Scan ID, Kafka broker address.
Output		
Name	Type	Description
HNM	JSON	Contains information, alerts, detection along with meta information.

Table 6: HEIR HNM interfaces

3.3.2.1.2 HEIR Exploit Tester (HET)

Establishes access to the system by bypassing security restrictions - normally runs after a vulnerability analysis is completed.

Module's interfaces		
Input		
Name	Type	Short Description
HNM	N/A	From HEIR Agent
Output		
Name	Type	Description
HNM	JSON	Exploit surfaces exposed and misconfigurations.

Table 7:HEIR HET interfaces

3.3.2.1.3 HEIR Cryptographic Checker (HCC)

Estimates the attack surfaces regarding security protocols. It lists the active cryptographic protocols.

Module's interfaces		
Input		
Name	Type	Short Description
HNM	JSON	Configuration file with analyzing targets along clientId, hospitalId, scan_id, kafka_broker address.
Output		
Name	Type	Description
HNM	JSON	Active cryptographic protocols and if there are vulnerable implementation (SSLScan output).

Table 8: HEIR HET interfaces

3.3.2.1.4 HEIR Threat detection module (TDM)

The Threat Detection Module (TDM) is a module integrated technically in the HEIR Agent but provides information to the HEIR Client in the same way and with the same meaning as the HNM. The module is able to scan local files and/or processes from an endpoint machine and to detect malicious content when executed. It provides another layer of information about malicious activity that influences further the RAMA Score and alerts.

Module's interfaces		
Input		
Name	Type	Short Description
TDM	JSON	Configuration file with analyzing targets along clientId, hospitalId, scan_id, kafka_broker address.
Output		
Name	Type	Description
TDM	JSON	Information about the scanned object and along with information regarding the detection type (alert).

Table 9: HEIR TDM interfaces

3.3.2.1.5 HEIR Vulnerability Assessment module

This module reports intelligent real-time security, privacy and data protection warnings.

Module's interfaces		
Input		
Name	Type	Short Description
VA	N/A	Information about installed software on the HEIR client device.
Output		
Name	Type	Description
VA	JSON	CVEs list for the installed applications.

Table 10: HEIR VA interfaces

3.3.2.2 Local RAMA Score Calculator

The Local RAMA Score calculator enables healthcare practitioners - and especially security experts – to identify the risk of their organization by using a number of tools coming from the HEIR Client. The RAMA score acts as a benchmark for the IT security of a hospital or healthcare facility. It is responsible for estimating the attack surface and resilience of the medical devices by incorporating several critical issues in a live manner. To calculate the score, the RAMA Score Calculator receives aggregated input from several HEIR components, through the HEIR Client. These components are:

- the HEIR Network Module (HNM)
- the HEIR Exploit Tester (HET)
- the HEIR Cryptographic Checker (HCC),
- the Vulnerability Assessment module,
- the Security Information and Event Management (SIEM) module, and
- the Threat Detection Module (TDM)

Further information for the Local RAMA score and the calculator is available in WP3's deliverables, i.e., D3.1 – The HEIR 1st layer of services package for the MVP ", "D3.2 - The HEIR 1st layer of services package: 1st complete version", and "D3.3 - The HEIR 1st layer of services package, final version". The table below includes the sample input from:

Module's interfaces		
Input		
Name	Type	Short Description
HEIR Client Input	JSON	The Local RAMA Calculator receives input from the HEIR Client. The latter incorporates scores from: • the HEIR Network Module (HNM) • the HEIR Exploit Tester (HET) • the HEIR Cryptographic Checker (HCC) and • the Vulnerability Assessment module • the SIEM module • The TDM module The full set of the expected JSON is available in D3.3.

Output		
Name	Type	Description
Local RAMA Output	JSON	The Local RAMA Calculator provides the RAMA Score (base and temporal score) and the corresponding metadata. The full set of the outputted JSON is available in D3.3.

Table 11: HEIR Local RAMA calculator interfaces

3.3.2.3 HEIR Aggregator

The HEIR Aggregator is a component of the HEIR framework initially designed for health institutions (HI) with multiple independent departments. Currently, due to its connector role between the HEIR Clients and the RAMA Calculator on the one hand and the HEIR 1st layer of services GUI on the other, the Aggregator is deployed to all HI where the HEIR framework is deployed. The Aggregator compiles statistical information on possible events or vulnerabilities discovered by the HEIR clients for the independent departments. An aggregated local RAMA score is also computed after having been provided with multiple local RAMA scores by the HEIR clients deployed in the individual departments. Once there are multiple HEIR clients independently writing to the Elasticsearch storage from one institution, the HEIR Aggregator is capable of compiling both the RAMA scores and the statistical information on HEIR client status. The HEIR Aggregator is triggered based on a user-defined schedule (e.g., every 5 minutes), reads the most recent outputs from the HEIR clients from the Elasticsearch storage, computes the aggregates, and writes the aggregated values for RAMA and event statistics to the Elasticsearch storage, where they can be accessed by the HEIR GUI. The Aggregator also sends its anonymized output to the HEIR Observatory database via a Kafka broker.

Module's interfaces		
Input		
Name	Type	Short Description
Local RAMA score and Metadata	JSON	Reads the JSON structure from the local VM ES "rama-heir-gui" index inserted by the KAFKA broker.
Output		
Name	Type	Description
Aggregated Rama and Metadata	JSON	Inserts JSON to the "aggregator-to-gui" ES index.

Table 12: HEIR Aggregator interfaces

3.3.2.4 HEIR Observatory

The HEIR Observatory is responsible for collecting, analyzing and presenting the results of all the deployed HEIR Clients inside the hospitals in order to provide anonymized global insights on the level of security in healthcare environments. The Observatory database stores all this information which will be analyzed by the HEIR Analytics Engine in order to produce statistics, historical analysis and trends as well as recommendations and best practices. For the final version of the Observatory, a new screen was developed, where an authorized policy maker of a hospital or someone with similar data clearance, can observe every active policy that exists in the HEIR system (PAF). Those policies are written in REGO language and the hospital's information is anonymized. The available results are presented in the 2nd layer of visualization.

3.3.2.4.1 HEIR global benchmarks (Global RAMA Score Calculator & The Security and Privacy assurance platform)

The HEIR global benchmarks are composed of two components, the Global RAMA Score Calculator and the Security and Privacy assurance platform. The former enables healthcare stakeholders to identify common issues for different healthcare sectors. It receives input from the HEIR aggregator and acts as an aggregator of all the local scores of a healthcare facility providing a unified score.

Module's interfaces		
Input		
Name	Type	Short Description
HEIR Aggregator	JSON	The Global RAMA Calculator receives input from the HEIR Aggregator.
Output		
Name	Type	Description
Global RAMA Output	JSON	The Global RAMA Calculator provides its output to the 2nd Layer of visualization of the Observatory.

Table 13: HEIR Global RAMA Calculator interfaces

As for the latter, the Security & Privacy Assurance Platform (SPAP) is responsible for monitoring, testing, and assessing the security (& privacy, if needed) posture of the protected organisation(s) and their assets, in a real-time, continuous manner. Several built-in security assessments addressing the Confidentiality – Integrity – Availability (CIA) principles (via custom metrics that can be tailored with respect to the platform's components) can be utilized, leveraging an evidence-based approach, to provide security assurance assessments with certifiable results. A high-level view of SPAP's internal architecture is provided in Figure 1; five primary modules can be identified:

- Cyber System Asset Loader: This component responsible for maintaining the cyber system's asset model for the target organization. This model includes the assets of the organization, security properties for these assets, threats that may violate these properties, relations between assets in the model, and the security controls that protect the assets and is based on the Assurance Model.
- Vulnerability Analyzer: The Vulnerability Analyzer is responsible to identify known vulnerabilities of assets defined within an organisation's asset model. This component automatically constructs the Common Platform Enumeration (CPE, a structured naming scheme for information technology systems, software, and packages) per asset and then retrieves the relevant Common Vulnerabilities and Exposures (CVE, a reference method for defining unique, common identifiers for publicly known information-security vulnerabilities and exposures) entries, by searching in a local copy of the National Vulnerability Database (NVD a U.S. government repository of standards-based vulnerability management data, maintained by the National Institute of Standards and Technology, NIST). This copy is continuously updated by utilising an in-house component that fetches the latest known CVEs from NVD's JSON files.
- **Dynamic Tester:** The component responsible for initiating dynamic testing assessments (i.e., penetration testing) to the target organization. The dynamic tester can also identify assets that are not included in the used asset model. The module consists of two components: (a) the dynamic tester or manager and (b) the dynamic testing tool (e.g., any external testing component; see below).
- **EVEREST**: A runtime monitoring engine, built-in Java, that offers an API for establishing the monitoring rules to be checked and forwards the runtime events from the application's

monitored properties, finally obtaining the monitoring results. This module is composed of two submodules: (a) the monitoring database and (b) the monitor (EVEREST). Regarding the latter, EVEREST can reason on rules expressed in Event Calculus and Drools rules. Through these, EVEREST can provide a model-based monitoring approach, allowing the users to tailor the monitoring rules to the organization's specific needs.

- **Event Captors**: An Event Captor is a tool that, based on a specification set by EVEREST, aggregates log and event information from the targeted infrastructure, and encapsulates in a specific format that can be consumed by the EVEREST model. Logs and events are mostly collected through Elasticsearch based on lightweight shippers (namely Beats) that forwards and centralizes log data. The needed Event Captors are initiated through EVEREST.

For HEIR, the Security and Privacy Assurance Platform's Dynamic Tester component will be utilised to ensure the security of the HEIR solution, as deployed in the pilots. For privacy reasons, the results of these assessments will remain confidential.

3.3.2.4.2 Observatory Database

The HEIR Database stores the data sent by the deployed HEIR Clients. This data includes RAMA scores and other generated outputs that are relevant for the parameters of the experimentation protocol.

3.3.2.4.3 Analytics Engine

The Analytics Engine is responsible for collecting and analyzing the data from the Observatory DB in order to provide global insights, statistics, and recommendations. Moreover, the Analytics Engine supports the historical analysis, advanced queries mechanisms, and user interaction capabilities, based on the role and the requirements of the end-user.

Module's interfaces		
Input		
Name	Type	Short Description
Local Aggregator's Metadata	JSON	The aggregated metadata per Hospital. (Aggregation of the connected departments' relevant output). Metadata refers to the output of the HEIR Client's modules. (Vuln. Assessment, HCC, HNM, HET, security status information and more.) Source: HEIR Aggregators.
Output		
Name	Type	Description
Statistical Data	JSON	Statistical data from the aggregated metadata of the connected Hospitals are fed to the 2nd layer of Visualization.

Table 14: HEIR Analytics Engine interfaces

3.3.3 HEIR Visualization

3.3.3.1 1st Layer of Visualizations

The HEIR Client GUI (HCG) includes interactive visualizations of information generated by the 1st level services running inside a hospital environment. This information is only available via authentication performed by an Angular custom-made mechanism to authorized users belonging to the hospital staff since it contains security-related information of the infrastructure. Moreover, the HCG fetches information from the HEIR Observatory to be used as a 'comparison' of the local aggregated RAMA score and the global one, thus providing users

with an idea of how their hospital stands with regards to other infrastructures. For the final version of the platform, a new screen was developed where authorized users of the hospital can investigate events regarding the users' data requests that were performed and collected via the PAF. Those events are recorded and accessible via the blockchain mechanism of HEIR. Based on the user's role, the displayed data may be redacted or not. (e.g., full set of data are available for authorized auditors).

Module's interfaces		
Input		
Name	Type	Short Description
Aggregated RAMA + Metadata	JSON	Aggregated RAMA score & metadata for the Hospital (aggregation of the connected departments' relevant output). Metadata refers to the output of the HEIR Client's modules. (Vuln. Assessment, HCC, HNM, HET, security status information and more.) Source: HEIR Aggregator.
Data requests logs	JSON	Logs about the data requests that have been made through the Privacy Aware Framework of HEIR. The data are retrieved via REST APIs from the Blockchain component of HEIR.

Table 15: HEIR 1st Layer of Visualization interfaces

3.3.3.2 2nd Layer of Visualizations

The 2nd layer of Visualizations is a web application that includes all the elements and methods to present information gathered by the HEIR Observatory. Basic recommendations are available through the visualization dashboard. Users accessing the HEIR Observatory will have read-only access to the anonymized data. During the 2nd half of the project, a new screen was developed where the enabled policies across all connected to HEIR ecosystem hospitals, are displayed.

Module's interfaces		
Input		
Name	Type	Short Description
Global RAMA + Metadata	JSON	Global RAMA score, details about HEIR's ecosystem (e.g., number of connected hospitals etc.) & metadata. Metadata are produced from the Aggregators' output of each Hospital. Source: HEIR Global RAMA Score Calculator.
Data policies (REGO)	JSON	Active data policies of hospitals that are enabled in Privacy Aware Framework of HEIR. The policies are retrieved via REST APIs and the displayed information are anonymized as per hospitals' identification information.

Table 16: HEIR 1st Layer of Visualization interfaces

3.4 Modules' Communication

In this section, the communication among the HEIR modules is presented. The following diagrams are on implementation level, and they are based on pilots' current use case scenarios/playbooks. The final version of the playbooks will be presented in D6.2.

3.4.1 Vulnerability management for outdated software & "infected" device

Figure 5 depicts the modules' flow on providing insights related to outdated software to the System Administrator of the pilots through the 1st layer of visualizations.

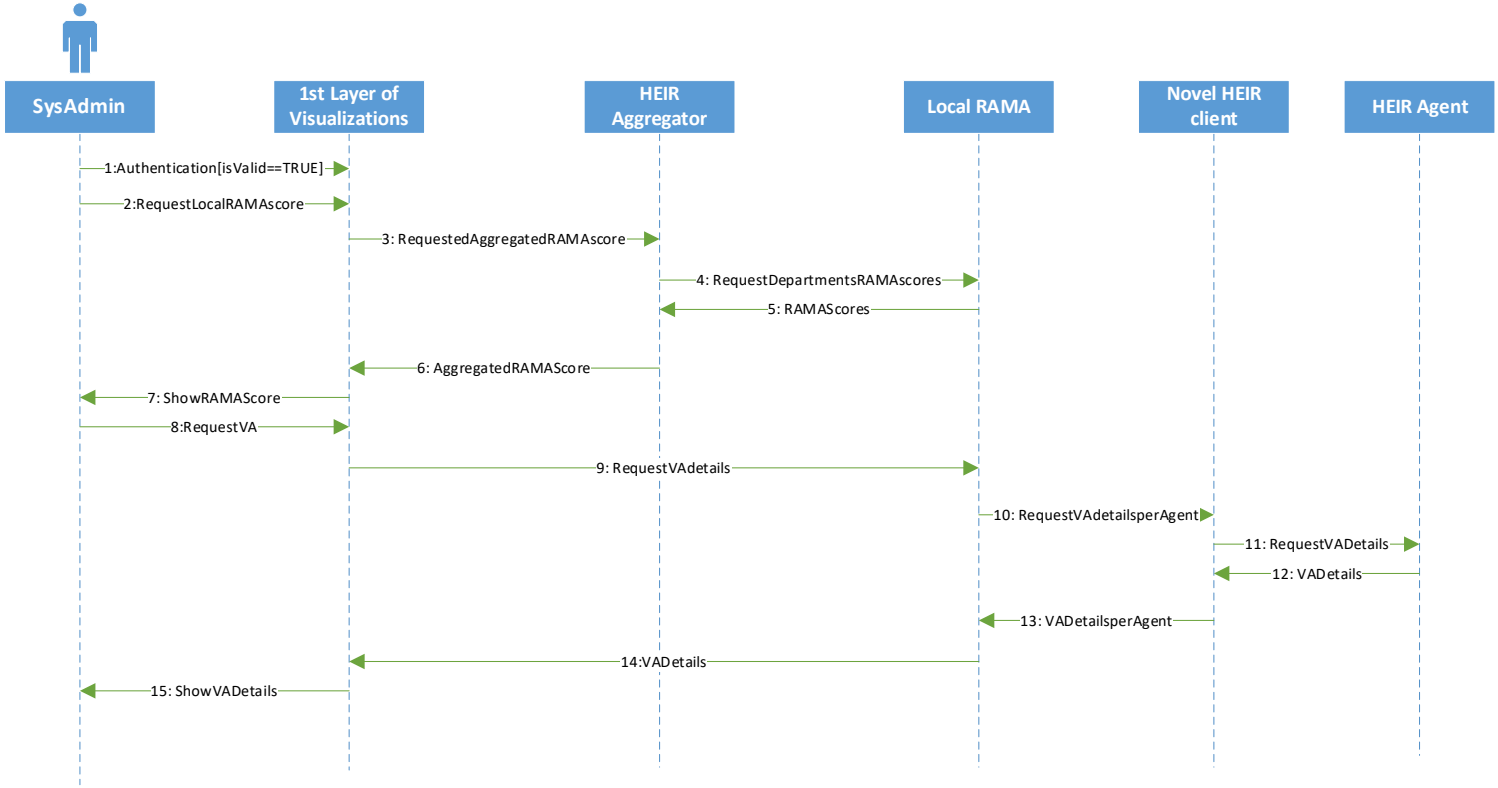

Figure 5: Data flow/sequence diagram (PAGNI, HYGEIA and CUH use case scenarios)

3.4.2 Anomaly detection by monitoring and reporting logs

Figure 6 depicts the modules' flow on providing insights related to suspicious behaviour from the log files to pilots' System Administrator through the 1st layer of visualizations.

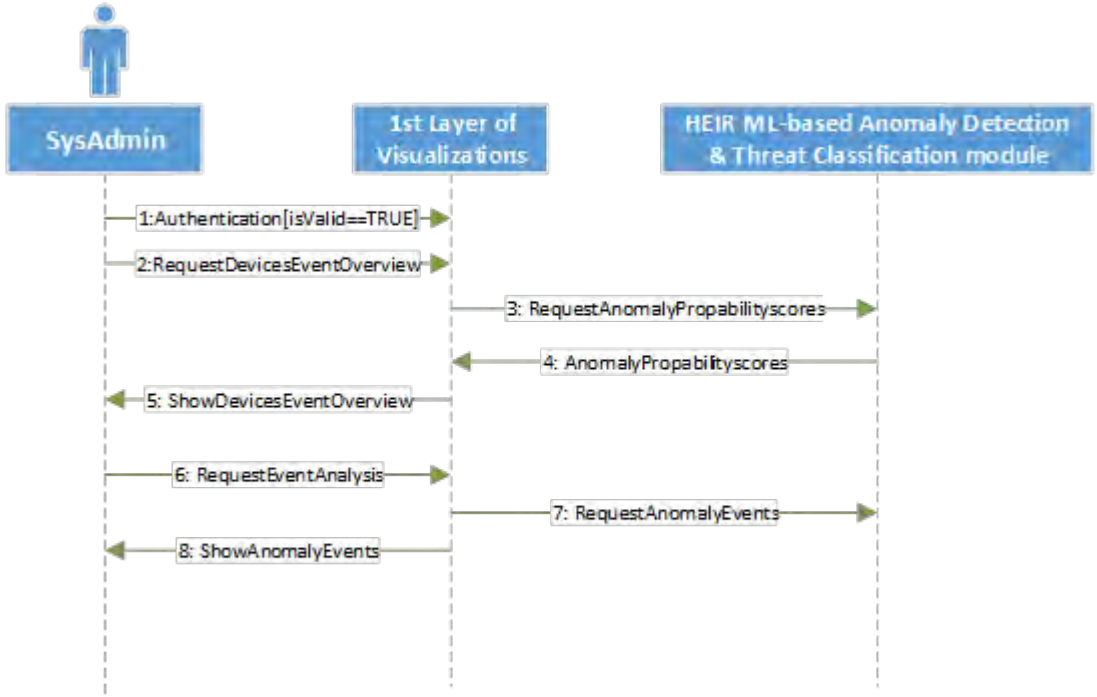

Figure 6: Data flow/sequence diagram on the Anomaly detection functionality

3.4.3 NSE - PAF

The following figures (Figure 7 & Figure 8) indicate the sequence diagrams for the two scenarios, the "push" model (automatic data redaction) where diabetes Observations are pulled out of a FHIR server and pushed into an S3 store, and a "pull" model ("Policy-based access to HL7 FHIR data for its ad-hoc analysis").

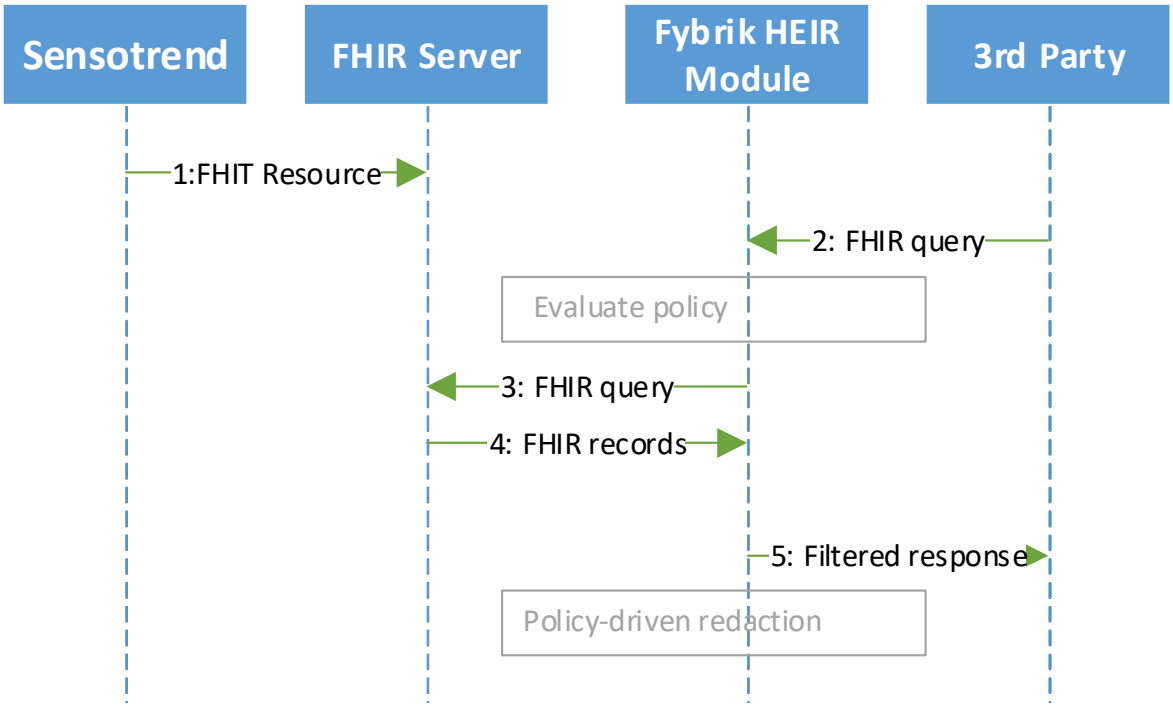

Figure 7: Data flow/sequence diagram for the "pull" model (NSE use case scenario)

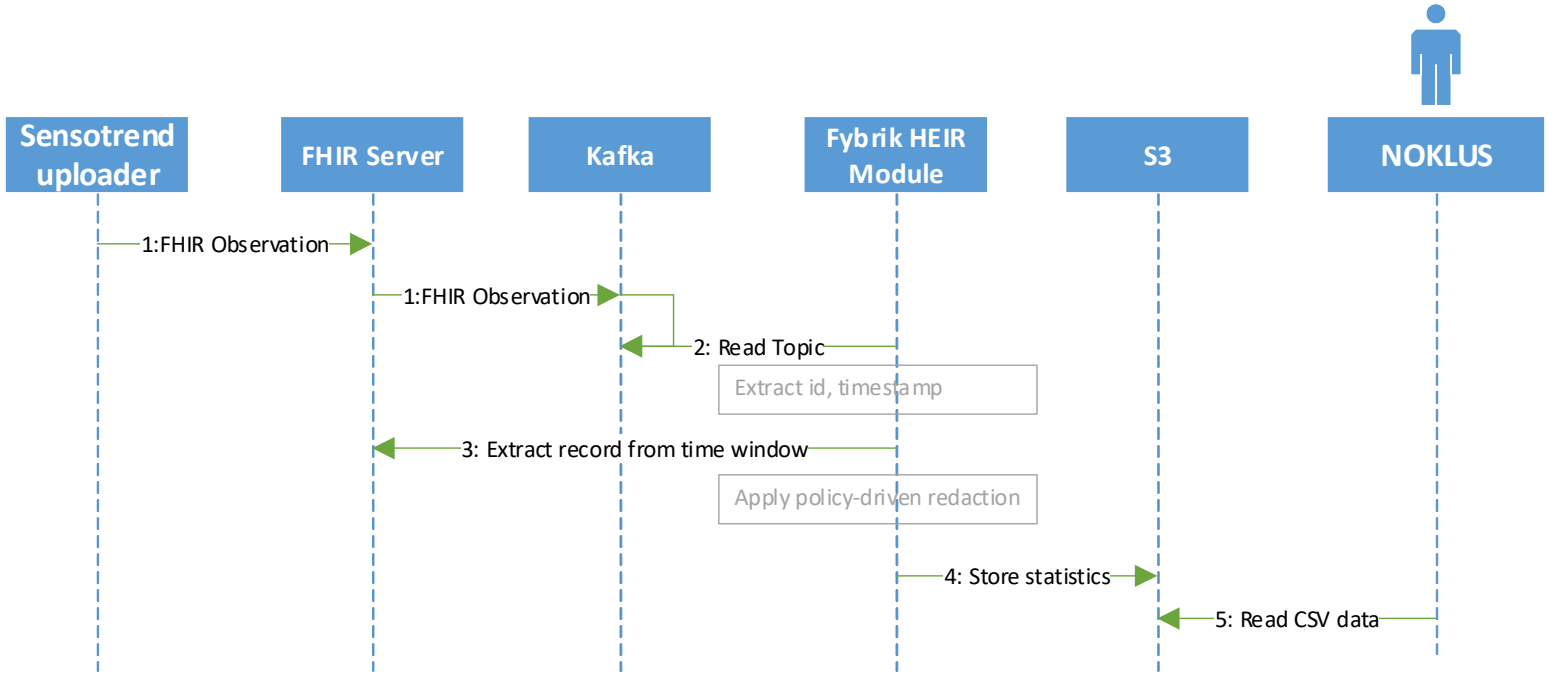

Figure 8: Data flow/sequence diagram for the "push" model (NSE use case scenario)

3.4.4 Observatory

Figure 9 indicates the module's flow on providing insights through the Global RAMA score and statistical analysis regarding the cybersecurity status of the healthcare domain. The analysis includes indications about the top vulnerabilities and the main issues faced by the healthcare institutions that are monitored by the HEIR platform as well as basic recommendations and mitigation actions.

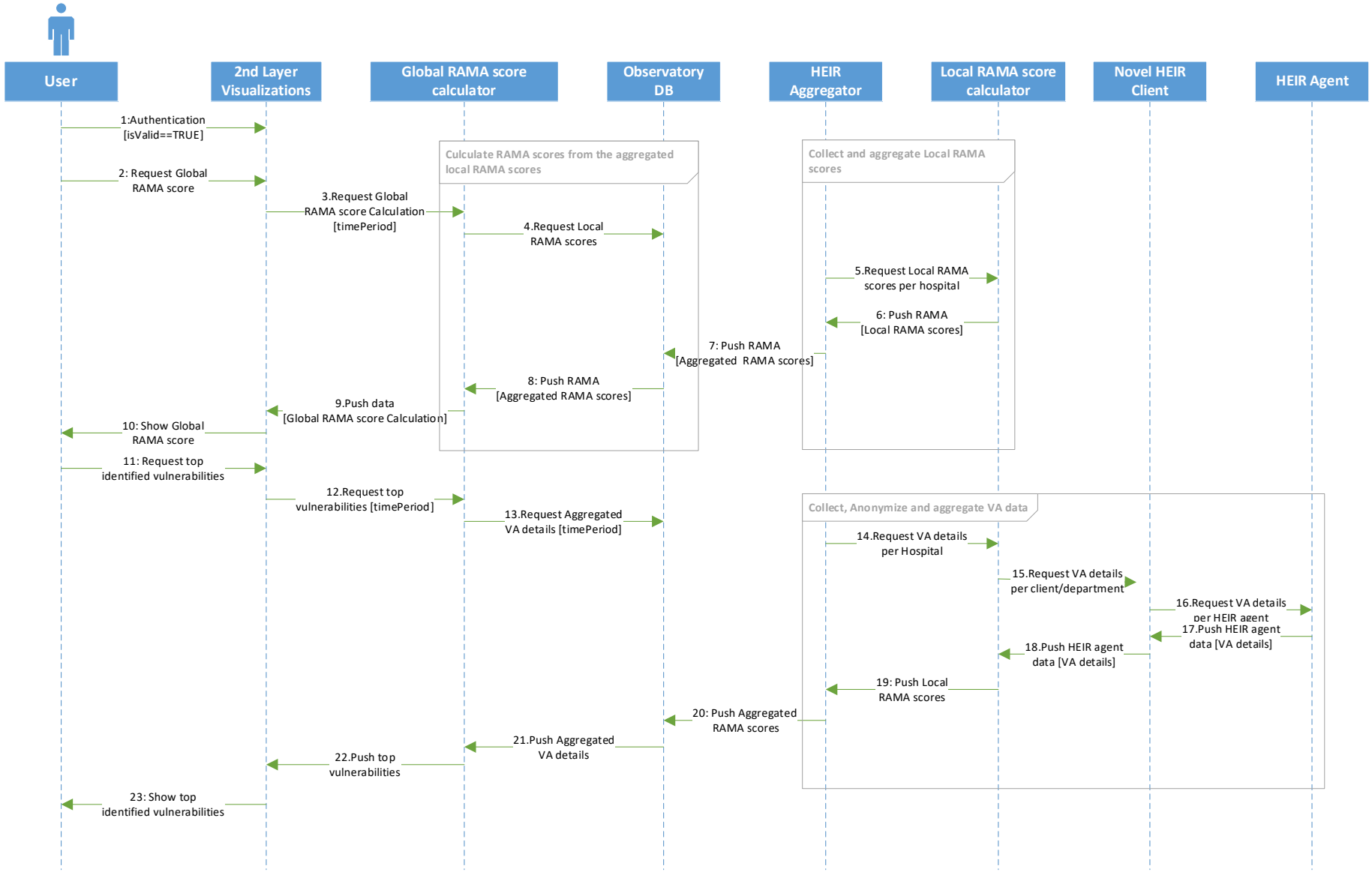

Figure 9: Data flow/sequence diagram on the Observatory use case scenario/playbook

4 Integration

For the integration purposes of the project, we followed the Agile Software Development Practices with frequent integration cycles, rapid prototyping and close collaboration between self-organising, cross-functional teams. A defined set of principles guided the integration process of the components together with the use of specific tools that fostered this process. Based on Agile practices, we applied Continuous Integration techniques to perform automated deployment of the provided modules.

4.1 Methodology

The methodology, processes and software toolset that were employed in the development and deployment of the HEIR platform and components were thoroughly presented in the previous iteration of the present document (D5.3). Overall, several of these described tools were extensively exploited (e.g., the deployment of all the updated versions of the HEIR components - docker images) under Kubernetes.

4.2 System & Network Security Specifications

To protect the services and data of the HEIR framework from unauthorized access, several different security measures are enforced to all nodes of the HEIR. These measures are summarized in the list below:

- All communication between the publicly available APIs of the platform is done through a secure protocol (HTTPS).
- Users are also defined on an Angular custom-made mechanism level, where different users have read-only privileges on the various data of the HEIR dataset.
- At the network level, a firewall employs rules that allow the traffic flow only through specific ports and domains to the services and databases of the system.
- Remote or physical access to the servers is provided only to authorized personnel.

In the following subsections are provided details on the HEIR infrastructure specifications.

4.2.1 System Specifications

In the context of the HEIR project, ITML set up a server machine in the pilots' premises for the development and collaboration purposes of the project's partners. The technical characteristics and specifications of the server machine are more than adequate for the relevant needs of the project. Since the Continuous integration towards the realization of HEIR framework (T5.2) has started in M8 of the project, the necessary hardware/software infrastructure and environment set up is taking place towards the end of the project. The technical specifications of the HEIR VM deployed in the pilot environments, based on the initial pilots' environment are presented in the table below.

OS	Ubuntu Server 20.04.3 LTS
CPU	4 X vCpu
RAM	16GB
Disk	100GB disk

Table 17: HEIR VM Requirements

Regarding the SIEM agents, the minimum and recommended specifications are listed in the table below.

	Minimum	Recommended
OS	Windows 7	Windows 10 or higher
Requirements	32bit/64bit	32bit/64bit
CPU	Intel i3	Intel i5
Memory	3G	4G

Table 18: Windows Agent Requirements

	Minimum	Recommended
OS	Centos 7 or Ubuntu 18.04	Centos 7 or Ubuntu 18.04
Platform	32bit/64bit	32bit/64bit
Software requirements	JRE 1.8	JRE 1.8
CPU	Intel i3	Intel i5
Memory	3G	4G

Table 19: Linux Agent Requirements

Note: The HEIR Agent provided by BD currently supports only Windows OS.

4.2.2 Network Specifications

All healthcare pilot environments are accessed via a virtual private network (VPN) that was either already in place or was setup for the needs of the HEIR project. The Pagni pilot already had an OpenVPN[12] server installed, HYGEIA had already deployed FortiVPN[13] whereas for the Croydon and NSE environments a new OpenVPN server was deployed for the needs of the project. Regarding the internal network requirements for every pilot, all deployed agents/end points must be able to communicate with the main Kafka message broker. The HEIR framework does not require general internet access apart from:

1. Access to connect to the Heir Observatory OpenVPN, and Kafka broker.
2. For specific components that need updates (i.e., to acquire a list of vulnerable software) specific internet addresses are requested to be whitelisted and allowed.

4.3 HEIR Kubernetes core deployments

For the facilitation of HEIR modules' integration, the below Kubernetes core deployments have been incorporated:

- ES Connector: The ES-connector is a Spring Boot[14] application consuming logs from the Kafka cluster and saves them to Elasticsearch offering multitenancy on the logs on the index level (e.g., the logs of group 1 are saved in index agents_1-YYYY.MM.dd). It is deployed by plain Kubernetes artifacts in a single VM but can be scaled if more VMs are added.

[12] https://openvpn.net
[13] https://www.fortinet.com
[14] https://spring.io/projects/spring-boot

- ES Connector 2: It is a Spring Boot application which extracts all high severity events from the Wazuh data stored in the Elastic Search and sends them to a Kafka topic which is monitored by the HEIR client, adding one more source of security related events that are used to calculate the RAMA score.
- Elasticsearch is deployed and operates using Elastic's team instructions.
- Kibana[15] is deployed to provide visualizations on the data reaching Elasticsearch.
- A Kafka cluster is deployed using custom Kubernetes artifacts and a private Certificate Authority to issue certificates. The Kafka operations are managed using a custom Spring Boot application which wraps often used actions in a convenient REST API. Moreover, Kafka mirroring (MirrorMaker[16]) is used to maintain a replica of the Observatory Kafka cluster in all pilots, making available computed metrics from all the HEIR eco-system. (e.g., Global Rama Score)

4.4 Access Control and Identity Management System

The HEIR integrated framework includes mechanisms for implementing an Access Control and Identity Management System (ACIMS), which can be supported by Keycloak[17]. Keycloak offers robust user management and security features, including password policies, user reminder settings, and complete login security procedures. Its web interface allows administrators to assign permissions for a variety of actions and manage roles, which enables them to establish an access control policy tailored to the project's requirements.

Although Keycloak has not been enabled in the current implementation of the HEIR framework, it has been designed to be easily incorporated, if needed. This will allow the HEIR Access Control and Identity Management System to take advantage of Keycloak's administrative capabilities, enabling administrators and moderators to manage roles and permissions through an intuitive online user interface.

However, deploying an access control and identity management system entails certain security risks, which the HEIR framework has identified and mitigated using a strategy presented below.

4.5 Security Risks and Mitigation

Risk	*Likelihood*	*Impact*	*Description*
Insecure access control	Low	High	A common requirement of most multi-user information systems is to provide a mechanism for access control. Access control comprises identification, authentication and authorization. By providing insecure access control mechanisms in HEIR, stakeholders might be able to access information of other users in the system. Furthermore, an attacker might get access to the HEIR integrated framework, which would enable him to use data that are not publicly available or misconfigure system settings.
Identity theft	Medium	High	Identity theft is about an attacker who pretends to be someone else. This is a serious risk, especially in an environment like HEIR framework which stores

[15] https://www.elastic.co/what-is/kibana
[16] https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=27846330
[17] https://www.keycloak.org/

			sensitive data. An attacker gaining access to the HEIR integrated framework as an existing user would have access to the user's profile and services.
Data Leakage	Medium	Medium	Data leakage refers to unauthorized third parties gaining access to personal or business-related data. Depending on the feedback and the granularity of the data, an attacker might have access to a large number of personal/sensitive data records. The HEIR framework will maintain sensitive patients' data as well as private analysed sets of data which can be used for advanced insights. Leakage of such data would expose sensitive medical information that could hurt system's credibility.

Mitigation: Security pattern-based access control

Security patterns are a well-established domain within the IT-security field. Security patterns describe well-proven security solutions for common IT-security problems. They are written by security experts in their respective domains. To implement access control in HEIR, a combination of security patterns is required. **Figure 10** depicts a system of security patterns to implement the HEIR integrated framework access control mechanism. By implementing the "Single Access Point" pattern, only one point of access needs to be secured. The "Checkpoint" pattern provides the framework for implementing the required authentication and authorization patterns and its enforcement. Relying on a security pattern approach, the insecure access control risk can be mitigated as only authorized users have access to the HEIR integrated framework. Moreover, a secure access control mechanism also indirectly mitigates the risk for identity theft as only the authorized users have access to services and protected data. Furthermore, it prevents data leakage, as all data stored in the HEIR integrated framework is only available to authorized users.

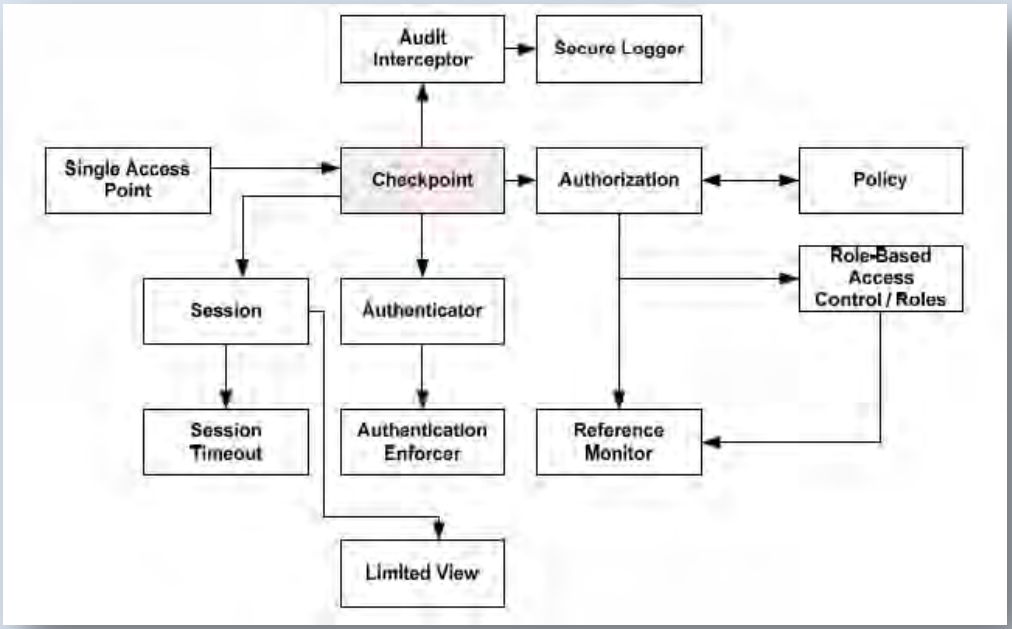

Figure 10: System of Security Patterns realizing Access Control

The HEIR security features are applied system-wise, covering all the architecture. The HEIR ACIMS will ensure that only users with appropriate permissions will be able to access the data relying in the framework's data storage. Moreover, the external APIs (Service Interfaces) exposed by the framework can be secured by means of encryption over HTTP via SSL, which is the standard protocol for security over the Internet.

Table 20: Identified security risks & mitigation.

4.5.1 Integration & Development Risks and Mitigations

The risks and respective mitigation methods/techniques that can be identified in the context of the HEIR integration process are as follows:

RISK	Mitigation
Wrong understanding of the modules to be developed	Close communication amongst partners (Technical). Delivery of Clear and representative reports for requirements, architecture etc.
Lack of Collaborative framework	Usage of Collaborative tools and Technologies accessible for all partners. (Discord, SVN, etc.)
Inconsistency of selected technologies	Clear definition of the technologies to be used in the related documentation. Usage and exchange of internal documentation for the technical partners.
Inadequacy of the infrastructure	Strong Assessment of the selected technologies and infrastructure. Low cost doesn't mean gain for the project.
Delays of Module Delivery	All the partners need to respect the deadlines. WP coordinators and Task leaders must prevent delays with early notifications and alerts.
Integrated System of high complexity	Non-technological partners and partners with expertise on like HEIR framework systems need to assess system mock-ups and workflows prior to the development phase.
Low level of Usability	End-Users and Non-Technological partners must get involved in the system assessment and provide feedback in a constructive way. Creation of an easy to use and collaborative evaluation framework.
Insufficiency of HEIR integrated framework related to user needs	End-Users and hospitals' representatives' involvement from requirements gathering phase.
HEIR Outcomes of poor quality	Adaptation of a methodology for software validation (e.g., 1012-2012 - IEEE Standard for System and Software Verification and Validation).

Table 21: Identified Integration & Development risks and mitigations.

4.6 Testing methodology & Results

In this section we provide an overview of the results of the system validation of the platform in the quality characteristics selected based on the ISO/IEC 25010:2011 as presented in D5.3. In general, software validation is the process of developing a "level of confidence" that the framework meets all requirements, functionalities, and user expectations as set out during the design process. It is a critical tool used to assure the quality of its component and the overall system. It allows for improving/refining the software development output.

4.6.1 Functional suitability

This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. The functional completeness of the HEIR Integrated framework refers to satisfying the user and functional requirements as introduced in the first half of the project documented in D1.1-2. The testing process followed a number of test scenarios/playbooks per pilot use case (are included in D6.2), which were iteratively repeated. These tests were successfully accomplished from the HEIR Integrated Framework and, in all iterations, they provided the same results. The test scenarios relate to the HEIR requirements and for each of them the result of the observation testing process will be reported in D6.2. Also, for the purposes of this quality characteristic in every HEIR module (code project), several unit and integration tests were performed to guarantee that the functional needs of the project are met, based on the user needs. The target set for the project was that at least 50% of the code would be covered by unit tests and this target percentage was achieved by all HEIR modules.

4.6.2 Performance Efficiency

This characteristic represents the performance's relation with the number of resources used under stated conditions. To evaluate the overall performance and scalability of the HEIR Integrated Framework involved in the user interfaces, we have examined the overall response times of these services in defined use case scenarios/playbooks. To implement the performance tests, we used the services running on the pilots' common servers of HEIR which has the following minimum specifications:

- Ubuntu Server 20.04 LTS operating system
- CPU 16 Vcores
- HDD 100GB
- 16 GB RAM

To test the performance of the user interfaces and identify potential weaknesses, the popular open-source tool JMeter[18] was used. JMeter is designed to load test functional behaviour and measure performance of web applications. Test Plans were created to test the performance of the most demanding GUIs that are handled by the HEIR visualization modules. Even that the envisioned number of HEIR visualization modules end-users is due to the limited number of the targeted stakeholders (for the 1st Layer of visualization: the IT personnel of a hospital and for the 2nd Layer of visualization: Researchers), the results show that the services can handle a larger number of concurrent users (over 80) without significant delays in response times (average: 8372 msec) or errors (0,3 %). The average times get higher as the concurrent users and number of requests increases but in an acceptable rate which doesn't result in loss of responsiveness. These results prove that the internal services of the framework can handle an

[18] https://jmeter.apache.org/

increasing number of users accessing the visualizations without any change of the current implementation.

4.6.3 Compatibility - Interoperability

This is the degree to which a product, system or component can exchange information with other products, systems, or components, and/or perform its required functions, while sharing the same hardware or software environment. All the modules in the HEIR project use well established communication protocols to exchange information. The two ways the communication is performed is through JSON REST web services and Kafka messages with the help of the ElasticSearch. Both ways guarantee that all the modules of the framework can exchange information and use the information that has been exchanged. The fact that most of the modules are dockerized, guarantees that the 2 or more modules can coexist in the same common environment, sharing the resources of the host server, without issues, since the docker images run in isolation and any dependency of one component, cannot affect another component.

4.6.4 Usability

Usability refers to the interaction between the target users and the platform interface. Testing the user interface of the platform will verify the correct communication of the users with the underlying software. Testing the user environment will furthermore ensure that the interface elements of the framework operate as they should and that they agree with the technical and functional requirements. Testing usability involves a step-to-step navigation among all the pages, fields and functionalities of the framework. These exhaustive tests reveal the perception of the framework's functionalities by the users and their understanding of the current status of the framework. Therefore, the evaluation of the platform's usability will not be analysed in this deliverable since there is a separate work package that deals with this process. WP6 sets the metrics and includes a series of evaluation reports which will provide extensive feedback on the HEIR integrated framework usability performance.

4.6.5 Security

This is the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. As mentioned in Section 4.2 of the current document, several measures were taken to assure that the communication between the components and the data of the system are secured. Also, penetration testing has been performed. Penetration testing, also known as pen testing, is a crucial step in ensuring the security of any project, especially those in the healthcare industry. In the context of HEIR, penetration testing involves evaluating the system's security by attempting to exploit vulnerabilities in the system's software, hardware, and network infrastructure, through the use of the Dynamic Testing module of the Security and Privacy Assurance Platform. To perform penetration testing in the pilots of HEIR, the following steps were taken:

- *Identify the scope*: The first step is to identify the scope of the penetration testing. This includes defining the objectives of the test, the assets to be tested, the testing methodology, and the tools to be used.
- *Gather information*: The next step is to gather information about the system's architecture, operating systems, applications, and network topology. This information can be obtained through documentation, interviews with stakeholders, and automated tools such as port scanners.
- *Vulnerability assessment*: The third step is to conduct a vulnerability assessment to identify potential vulnerabilities in the system. This involves using automated tools such

as vulnerability scanners to identify known vulnerabilities in the system's software and hardware.

- *Exploitation testing*: Once potential vulnerabilities have been identified, the next step is to attempt to exploit them. This involves using a combination of manual and automated techniques to exploit vulnerabilities and gain access to sensitive data or systems.
- *Reporting*: After the penetration testing is complete, a detailed report should be prepared that outlines the findings, including any vulnerabilities that were identified and the steps taken to exploit them. The report should also include recommendations for improving the system's security.
- *Remediation*: Finally, any vulnerabilities that were identified during the penetration testing should be remediated as soon as possible. This may involve patching software, reconfiguring network devices, or implementing new security controls.

Note: For privacy results, the results of these activities were only shared with the pilots.

4.6.6 Maintainability

This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. As mentioned, the architecture is modular, the components of the system are loosely coupled, thus internal changes in one module will not affect the operations of another. Changes however in the inputs or outputs of a module, may affect the operations of the integrated system. Moreover, having a proper documentation and software packaging, as well as complete installation instructions (will be reported at D5.5) increases the maintainability of the code and allows for the re-use of the existing modules and their re-use in future enhancements of the framework's functionality.

4.6.7 Adaptability

The implementation technologies of the HEIR Integrated framework ensure the performance of the modules and tools and provide high capability of adaptation to various installation and production environments. The HEIR modules are developed in different development environments which have been set up in the pilots' infrastructure and are accessible to all the technical partners. Upon finishing the first cycle of development, the modules were ported to the production environments which are different server machines without any significant problems during the migration process. The HEIR visualizations' functionalities have been tested with all the modern browsers to ensure adaptability in terms of client-side technology used to access the various modules.

4.6.8 Portability

Portability is the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. All HEIR modules of the main nodes are dockerized and pushed into the project's private registry. All the supporting tools of these modules are also dockerized and a docker compose file states the docker containers and the dependencies between them and can easily be configured so that it can be used in other servers. The databases and the message bus can easily back up through supporting tools and be restored in another environment easily through the same tools. Any file needed can also be pulled into a new environment through the project's private repositories. So, the platform can be replicated in a relatively easy manner.

5 Conclusion

This deliverable is the accompanying report of the updated and final version of the HEIR Integrated framework because of the work performed in the context of T5.2. This final version includes the complete set of functionalities covering the needs of the pilot use cases and enabling the HEIR stakeholders to test and evaluate the concepts and knowledge conveyed by the project. The described framework integrates the final versions of the individual modules and provides a unified environment through which the HEIR end users will perform their actions and provide feedback for further improvements.

The document offered a comprehensive outline of the integrated platform, covering various aspects such as the system's architecture, the HEIR modules integrated within it, the communication between these modules, and how they support use case scenarios and playbooks. Additionally, the document also provided details on the methodology employed, system and network specifications, security risks and measures taken to mitigate them, as well as the results of technical testing.

In comparison to the previous version of the deliverable, the HEIR platform has undergone significant technological enhancements and upgrades. These improvements include the integration of newly developed components, such as the Threat Detection module, as well as the inclusion of additional metrics for calculating the RAMA score, including SIEM Events and Machine Learning analysis of medical and patient data.

The next steps include the preparation of the final evaluation round and the final evaluation of its platform functionalities that is going to be reported in D6.2 which is due M36. The framework will be maintained in a fully functional state throughout the project's duration to provide a stable and efficient system for end-users, and appropriate measures are taken to ensure that this state is maintained beyond the project's completion.

6 ANNEX I – Components APIs

6.1 Blockchain-based HEIR Auditing Mechanism API

The current exposed endpoints are described below.

Description	GetAllLogs - Returns the list of the stored logs
Method	GET
Endpoint	<domain>/queryAllLogs
Data Source	Fabric ledger
Parameters	N/A
Description	**queryExists - Returns a boolean value indicating the existence of specific log**
Method	GET
Endpoint	<domain>/queryExists
Data Source	Fabric ledger
Parameters	Timestamp
Description	**queryLog - Returns a specific log based on timestamp**
Method	GET
Endpoint	<domain>/queryLog
Data Source	Fabric ledger
Parameters	Timestamp
Description	**getLogsByRange - Returns the list of the stored logs for the desired time range**
Method	GET
Endpoint	<domain>/getLogsByRange
Data Source	Fabric ledger
Parameters	startDate, endDate (yyyy-mm-dd, yyyy-mm-dd)
Description	QueryLogsByID - Returns the list of the stored logs for specific userID
Method	GET
Endpoint	<domain>/queryLogsById
Data Source	Fabric ledger
Parameters	userID (the identifier of the user that generated the access log)
Description	QueryLogsByIntent - Returns the list of the stored logs for specific intent
Method	GET
Endpoint	<domain>/queryLogsByIntent
Data Source	Fabric ledger
Parameters	Intent (analysis, research, visualization etc)
Description	QueryLogsByOutcome - Returns the list of the stored logs for specific query

Method	GET
Endpoint	<domain>/queryLogsByQuery
Data Source	Fabric ledger
Parameters	Query (observation etc)
Description	QueryLogsByQuery - Returns the list of the stored logs for specific outcome
Method	GET
Endpoint	<domain>/queryLogsByOutcome
Data Source	Fabric ledger
Parameters	Outcome (AUTHORIZED or UNAUTHORIZED)
Description	getLogsByRangeAndID - Returns the list of the stored logs for the desired time range and for specific userID
Method	GET
Endpoint	<domain>/getLogsByRangeAndID
Data Source	Fabric ledger
Parameters	startDate, endDate, userID
Description	getLogsByRangeAndOutcome - Returns the list of the stored logs for the desired time range and for specific outcome
Method	GET
Endpoint	<domain>/ queryLogsByRangeAndOutcome
Data Source	Fabric ledger
Parameters	startDate, endDate, Outcome
Description	getLogsByRangeAndIntent - Returns the list of the stored logs for the desired time range and for specific intent
Method	GET
Endpoint	<domain>/ queryLogsByRangeAndIntent
Data Source	Fabric ledger
Parameters	startDate, endDate, Intent
Description	QueryLogsByIDAndOutcome - Returns the list of the stored logs for specific userID and outcome
Method	GET
Endpoint	<domain>/ queryLogsByIdAndOutcome
Data Source	Fabric ledger
Parameters	userID, Outcome
Description	QueryLogsByIDAndIntent - Returns the list of the stored logs for specific userID and intent
Method	GET
Endpoint	<domain>/ queryLogsByIdAndIntent
Data Source	Fabric ledger
Parameters	userID, Intent

6.2 Local RAMA Score Kafka

Description	HEIR Client to Local RAMA
Method	-
Endpoint	topics = HeirClientToRama, topicId = sphynx-consumer
Data Source	HEIR Client
Parameters	N/A
Description	**Local RAMA to HEIR Aggregator**
Method	-
Endpoint	topic = "RamaToHeirGUI"
Data Source	Local RAMA
Parameters	-

6.3 Global RAMA Score Kafka

Description	HEIR Aggregator to Global RAMA
Method	-
Endpoint	topics = sie-aggr, topicId = sphynx
Data Source	HEIR Aggregator
Parameters	N/A
Description	**Global RAMA to Observatory**
Method	-
Endpoint	topic = "GlobalToObservatory"
Data Source	Local RAMA
Parameters	-

6.4 PAGNI Realtime Data for ML Component

Description	PAGNI API
Method	-GET
Endpoint	/logs2ml.php
Data Source	PAGNI
Parameters	N/A
Description	**PAGNI logs to machine learning component**
Method	-
Endpoint	/logs2ml.php
Data Source	PAGNI
Parameters	-