



## D5.3

### HEIR Integrated framework intermediate version

Project number	883275
Project acronym	HEIR
Project title	A secure Healthcare Environment for Informatics Resilience
Start date of the project	September 1 <sup>st</sup> , 2020
Duration	36 months
Programme	H2020-SU-DS-2019

Deliverable type	Demonstrator
Deliverable reference no.	D5.3
Workpackage	WP5
Due date	02-2022-M18
Actual submission date	21/03/2022

Deliverable lead	ITML
Editors	George Tsakirakis, Panagiotis Rodosthenous, Nikos Dimakopoulos (ITML)
Contributors	Dragoş Gavriluţ, Bogdan Prelipcean (BD), Iulia Ilie (SIE), Andreas Alexopoulos, Leonidas Kalipollitis (AEGIS) Michael Smyrlis (STS), Eliot Salant (IBM), Marwan Khabbaz (TUD)
Reviewers	Ovidiu Costel MIHAILA (BD), Eliot Salant (IBM)
Dissemination level	PU
Revision	1.0
Keywords	Cybersecurity, Privacy, Integrated solution

#### Abstract

This deliverable provides a conceptual specification of the HEIR framework architecture and deployment. It presents the current status of the implementation of the HEIR modules 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 elaborated in WP6. It presents the integration methodology and specifications, the identified risks and mitigations as well as introduces the integrated solution testing methodology. The results of this deliverable will serve as guidelines for the technical WPs (WP2, WP3, WP4, WP5) and for the HEIR integrated framework final version

#### Disclaimer

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author’s view – the European Commission is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability.



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

## 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.

In this context, this document describes the initial integrated prototype of the HEIR project. This version is the first integrated working version of the framework and 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 initial version is a step closer to the MVP concept (D5.2) for the HEIR end users. It encapsulates most of the underlying technologies and gives a clear and easy to use graphical interface, exposing every available feature so far.

The benefit of the current 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 arising functionalities, which will maximize the potential for further exploitation and adoption of the framework beyond the first half of the project.

## Table of Contents

<b>EXECUTIVE SUMMARY .....</b>	<b>2</b>
<b>1 INTRODUCTION.....</b>	<b>5</b>
1.1 SCOPE AND OBJECTIVES.....	5
1.2 DOCUMENT STRUCTURE .....	5
1.3 RELATION TO OTHER TASKS AND WORK PACKAGES .....	5
<b>2 SYSTEM ARCHITECTURE.....</b>	<b>6</b>
2.1 ARCHITECTURE DIAGRAM .....	6
2.2 DEPLOYMENT DIAGRAM .....	10
2.3 THE HEIR MODULES .....	12
2.4 MODULES' COMMUNICATION .....	23
<b>3 INTEGRATION.....</b>	<b>29</b>
3.1 METHODOLOGY .....	29
3.2 SYSTEM & NETWORK SPECIFICATIONS .....	33
3.3 ACCESS CONTROL AND IDENTITY MANAGEMENT SYSTEM.....	35
3.4 SECURITY RISKS AND MITIGATION .....	35
3.5 INTEGRATION & DEVELOPMENT RISKS AND MITIGATIONS .....	37
3.6 TESTING METHODOLOGY .....	37
<b>4 TIME PLAN .....</b>	<b>40</b>
<b>5 CONCLUSION AND FUTURE STEPS .....</b>	<b>41</b>
<b>6 ANNEX I – COMPONENTS APIS .....</b>	<b>42</b>
6.1 BLOCKCHAIN-BASED HEIR AUDITING MECHANISM API .....	42
<b>7 ANNEX II – USE CASE SCENARIOS/PLAYBOOKS .....</b>	<b>44</b>
7.1 PAGNI USE CASE SCENARIO/PLAYBOOK.....	44
7.2 NSE USE CASE SCENARIO/PLAYBOOK .....	44
7.3 HYGEIA USE CASE SCENARIO/PLAYBOOK .....	46
7.4 CUH USE CASE SCENARIO/PLAYBOOK .....	47
7.5 OBSERVATORY USE CASE SCENARIO/PLAYBOOK.....	48

## List of Figures

FIGURE 1 ARCHITECTURE DIAGRAM .....	9
FIGURE 2 LOCAL DEPLOYMENT DIAGRAM .....	10
FIGURE 3 GLOBAL INTEGRATED DIAGRAM .....	11
FIGURE 4 WAZUH EVENT FLOW MANAGEMENT .....	13
FIGURE 5 ALGORITHM STRUCTURE .....	15
FIGURE 6: DATA FLOW/SEQUENCE DIAGRAM (PAGNI, HYGEIA AND CUH USE CASE SCENARIOS) .....	25
FIGURE 7: DATA FLOW/SEQUENCE DIAGRAM FOR THE “PULL” MODEL (NSE USE CASE SCENARIO) .....	26
FIGURE 8: DATA FLOW/SEQUENCE DIAGRAM FOR THE “PUSH” MODEL (NSE USE CASE SCENARIO) .....	27
FIGURE 9: DATA FLOW/SEQUENCE DIAGRAM ON THE OBSERVATORY USE CASE SCENARIO/PLAYBOOK .....	28
FIGURE 10 CONTINUOUS INTEGRATION WORKFLOW.....	31
FIGURE 11 CONTINUOUS INTEGRATION WORKFLOW (2) .....	32
FIGURE 12 SYSTEM OF SECURITY PATTERNS REALIZING ACCESS CONTROL .....	36
FIGURE 13 THE ISO/IEC 25010:2011 SYSTEM/SOFTWARE QUALITY MODEL CHARACTERISTICS.....	39

## List of Tables

TABLE 1: HEIR VA INTERFACES .....	12
TABLE 2: HEIR FVT INTERFACES .....	12
TABLE 3: HEIR SIEM INTERFACES .....	14
TABLE 4: HEIR ML-BASED ANOMALY DETECTION INTERFACES.....	16
TABLE 5: HEIR PAF INTERFACES .....	17
TABLE 6: HEIR BLOCKCHAIN-BASED AUDITING MECHANISM INTERFACES .....	18
TABLE 7: NOVEL HEIR CLIENT INTERFACES.....	19
TABLE 8: HEIR HNM INTERFACES .....	19
TABLE 9: HEIR HET INTERFACES .....	19
TABLE 10: HEIR HET INTERFACES .....	19
TABLE 11: HEIR LOCAL RAMA CALCULATOR INTERFACES.....	20
TABLE 12: HEIR AGGREGATOR INTERFACES .....	21
TABLE 13: HEIR GLOBAL RAMA CALCULATOR INTERFACES.....	21
TABLE 14: HEIR ANALYTICS ENGINE INTERFACES.....	22
TABLE 15: HEIR 1ST LAYER OF VISUALIZATION INTERFACES .....	22
TABLE 16: HEIR 1ST LAYER OF VISUALIZATION INTERFACES .....	23
TABLE 17: HEIR VM REQUIREMENTS .....	34
TABLE 18: WINDOWS AGENT REQUIREMENTS .....	34
TABLE 19: LINUX AGENT REQUIREMENTS .....	34
TABLE 20: IDENTIFIED SECURITY RISKS & MITIGATION .....	36
TABLE 21: IDENTIFIED INTEGRATION & DEVELOPMENT RISKS AND MITIGATIONS .....	37
TABLE 22: INTEGRATION TIME PLAN .....	40

# 1 Introduction

## 1.1 Scope and Objectives

This deliverable is part of WP5 and specifically the second deliverable of T5.2 (Continuous integration towards the realisation of HEIR framework), emphasising the realization of the 1<sup>st</sup> 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 1<sup>st</sup> complete integrated version and as an upgraded version of the MVP (D5.2), the various HEIR modules developed by the technical partners were integrated into a unified solution which is deployed in pilot partners' infrastructure and in the HEIR Observatory infrastructure provided by ITML. The objective is to showcase a working 1<sup>st</sup> integrated version of the framework with successful detection of cyber threats, unauthorised access to the network, and analysis of this information through the visualization tools that were deployed. In this way, security and privacy in an end-to-end healthcare environment can be ensured.

## 1.2 Document Structure

This document reports on the initial activities and effort placed in the integration of the various technologies and tools towards delivering a functional 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 current demonstration version of the HEIR integrated framework and, as such, it is structured as follows:

- Section 3: Presents the 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 introduces the information sequence diagrams of the implemented the use cases scenarios/playbooks.
- Section 4: The integration methodology and specifications
- Section 5: The integration time plan
- Section 6: Finally, conclusion and future steps of the HEIR integrated framework.

## 1.3 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). Additionally, there is a close relation with WP6 (HEIR real-life demonstration and validation) which defines and crystallizes the use case scenarios, the demonstration and validation of the solution.

## **2 System Architecture**

### ***2.1 Architecture Diagram***

The presentation of the HEIR solution architecture follows a logical path, from the use cases supported by the solution to the functional conceptualization of the needed components, 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 Use Case initial scenarios (

ANNEX II – Use case scenarios/playbooks), were defined and explained in section 0 of the current document. These System Use Cases reveal the degree of complexity and needs for modularity, communication and orchestration of the various components that will be integrated within the HEIR solution. The architectural design aims to cover all these intricacies, while maintaining the openness of the system and 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 all the following aspects:

- Usability
- Performance
- Security
- Maintainability
- Scalability
- Reliability

The HEIR components are responsible for a specific set of functionalities. By convention, the 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 (1<sup>st</sup> and 2<sup>nd</sup> 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 are going to follow a Model View Controller design pattern for building a web application, starting from a plain, well-defined user interface that consumes the services provided by the backend. The use of modern web technologies for advanced visualisations (e.g. Highcharts.js<sup>1</sup>, plotly<sup>2</sup> or other visualisation library) and interactive, responsive dashboards (e.g. React.js or Angular.js) will offer the best set of Front-end features to provide a clean and fully functional interface. Finally, we are following an agile methodology for developing the HEIR solution, based on rapid prototyping and frequent iterations. This enables more frequent evaluations close to the end-user of the solution and better result in terms of meeting the usability requirements.

**Performance:** For tackling performance issues, we are going to rely on two factors – caching and distribution. The solution architecture logic is implemented modular in the backend 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. If needed, load balancing and caching will be applied.

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

**Maintainability:** For addressing maintainability, we are following standards oriented and technology independent architecture. For communication between the various developer teams,

---

<sup>1</sup> <https://www.highcharts.com/>

<sup>2</sup> <https://plot.ly/>

we have set in place Redmine<sup>3</sup> issue tracking system which will be used on the second half of the project during and after the evaluation period. This way all issues can be traceable, and the history of the development process can be examined. Section 5 provides a detailed description of the integration methodology to be used.

**Scalability:** The system will scale up based on the volume of the requests. The architecture is free to scale up easily due to the service oriented distributed nature of the backend. 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 will be examined in the course of the project and results and possible actions will be proposed.

**Reliability:** In order 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 standard. Some of the modules of the HEIR solution are based on existing solutions that have been used in the past and will only require some adaptation in order 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-sourced solutions.

The components of the solution 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 in order to be tolerant of misconfiguration or malicious input. Furthermore, the loosely coupled architecture of the solution avoids points of single failure and provides the ability to have a working production solution 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 solution to a halt.

The following architecture diagram of the present state of components' interconnections is shown in **Erreur ! Source du renvoi introuvable.** Figure 1.

---

<sup>3</sup> <http://www.redmine.org/projects/redmine/wiki>



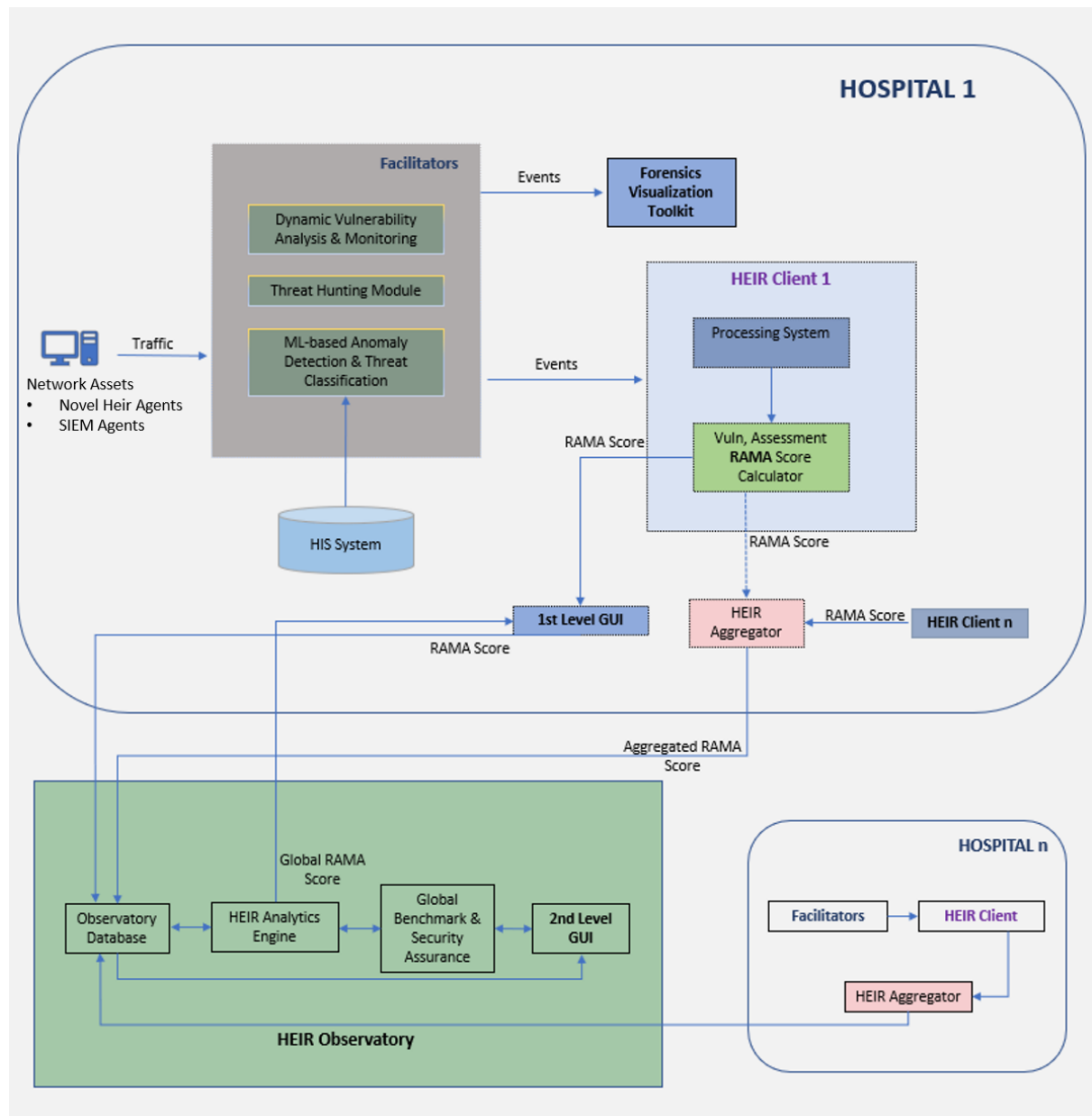


Figure 1 Architecture diagram

In pilot use-cases (PAGNI, HYGEIA, CUH), the data is collected by:

- HEIR agents installed in servers/workstations
- Web Traffic intercepted by the HEIR Client network module
- Data from the Hospital Information System (HIS) that explicitly feeds the ML-based Anomaly Detection & Threat Classification component

Data deriving from the SIEM agents are presently explicitly portrayed on the Forensics Visualization Toolkit serving as a GUI for the threat hunting module for use by the hospital IT administrators. Data originated from the HEIR Novel agents are relayed via Dynamic Vulnerability Analysis & Monitoring to the HEIR Client processing system where it is used to calculate the RAMA score for a single 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.

All the above RAMA security score and metadata is communicated to the HEIR Observatory database and is used by the HEIR Analytics Engine in conjunction with the Global Benchmark & Security Assurance component, to produce the Global RAMA score, which is also communicated back to the hospital (1<sup>st</sup> Level GUI). Finally, the RAMA scores and metadata is portrayed in the Observatory graphical interface (2<sup>nd</sup> level GUI)

## 2.2 Deployment Diagram

The following deployment diagrams provides information about the solution deployment topology. The deployment diagram of a local HEIR client (i.e., Pagni pilot) is depicted in Figure 2.

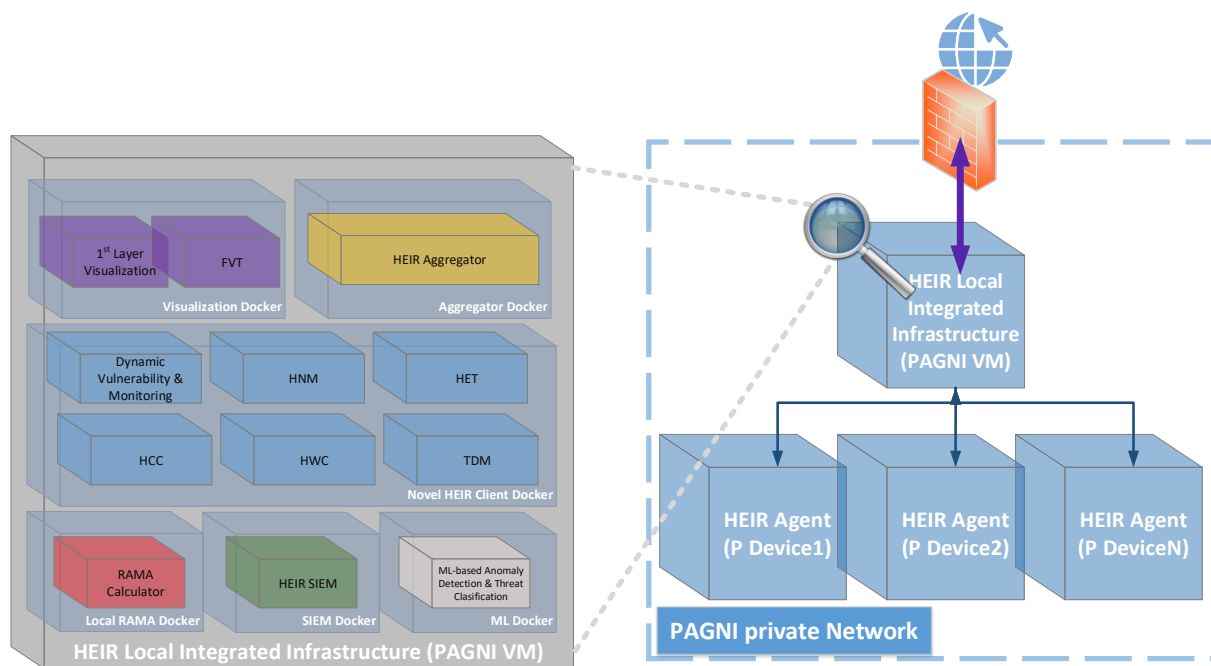


Figure 2 Local deployment diagram

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

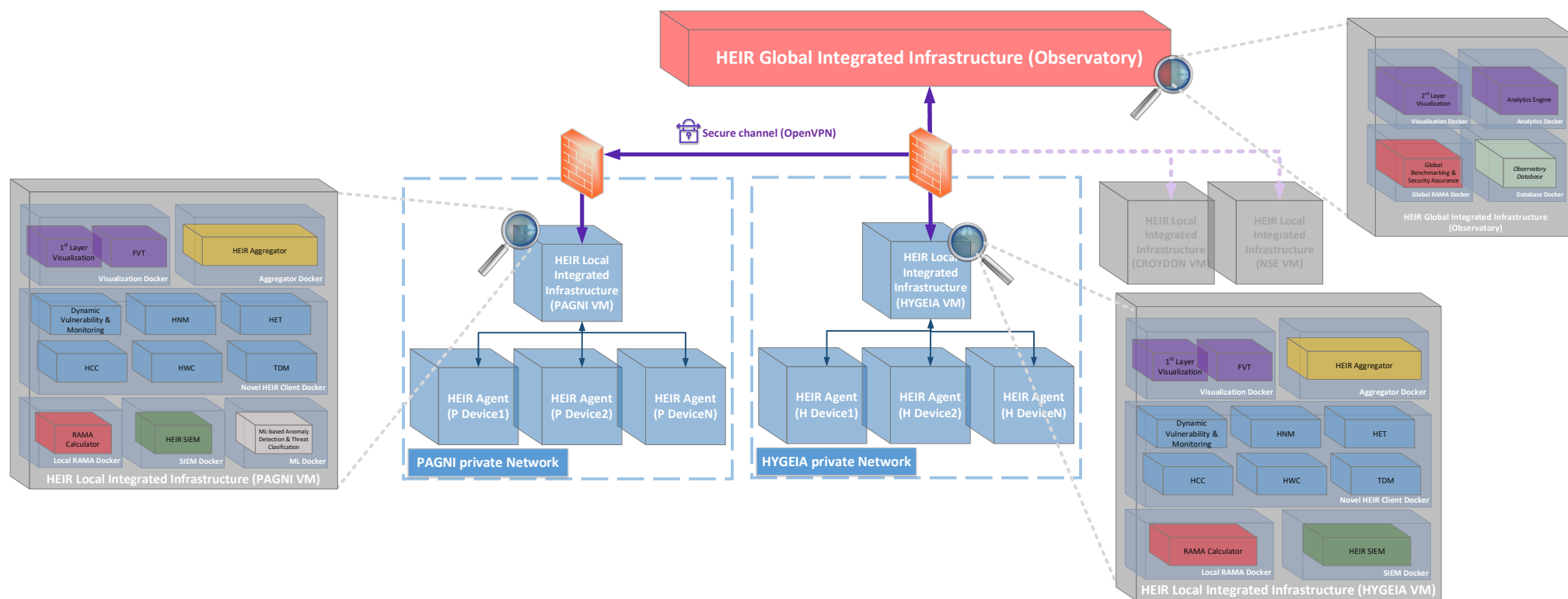


Figure 3 Global integrated diagram

## Disclaimer

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author’s view – the European Commission is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability.



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

## 2.3 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 specific details that are provided for each module are useful in terms of obtaining an overview and understanding easily which technologies are involved in all use cases. More detail regarding the individual modules hereby described can be found in the related deliverables D2.2, D3.2 and D4.2.

### 2.3.1 HEIR Facilitators

#### 2.3.1.1 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	From HEIR Agent
Output		
Name	Type	Description
VA	JSON	CVEs list for the installed applications

Table 1: HEIR VA interfaces

#### 2.3.1.2 Forensics Visualization Toolkit (FVT)

The FVT provides users with a timeline-based representation of the security events for each department 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 1<sup>st</sup> layer of visualizations and is meant to represent the logged events in a more detailed way. Authorized users 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 + Metadata	JSON	Selected department's RAMA score & metadata. Metadata refer 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 2: HEIR FVT interfaces

##### 2.3.1.2.1 HEIR SIEM

#### Disclaimer

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author's view – the European Commission is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability.



The HEIR SIEM component supplies various security related data from all endpoints to the HEIR Interactive Forensics Module and it is planned to support the Vulnerability Assessment Module providing another source of security information that can be evaluated and exploited together with the rest of the available components and data sources.

It is based on the Wazuh<sup>4</sup> open-source solution which provides a multitude of security related services that continuously monitor an IT infrastructure. All data is collected by lightweight 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 is 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<sup>5</sup> which also provides a powerful interface for data visualization and analysis via its integration with Kibana.<sup>6</sup>

In addition to logs and events deriving from the operating system, Wazuh is able to 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<sup>7</sup>. 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**

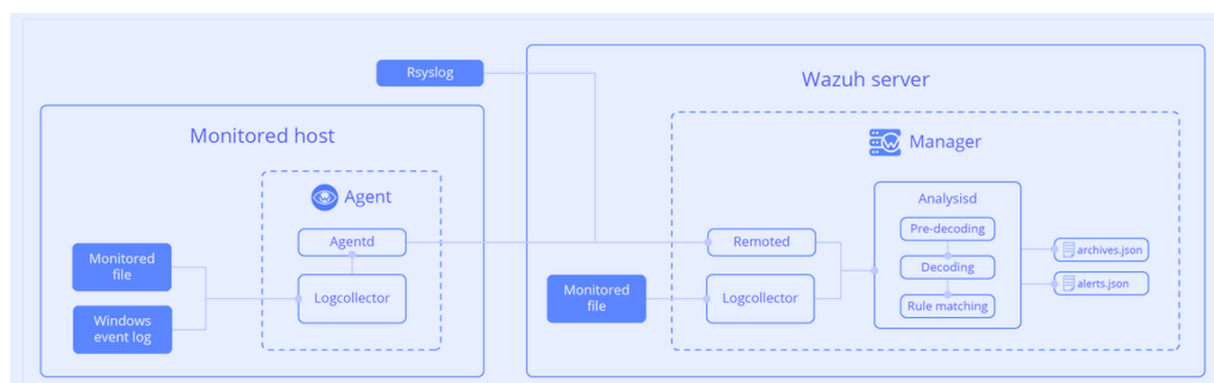


Figure 4 Wazuh event flow management

<sup>4</sup> <https://wazuh.com/>

<sup>5</sup> <https://www.elastic.co/elastic-stack/>

<sup>6</sup> <https://www.elastic.co/kibana/>

<sup>7</sup> <https://www.rsyslog.com/>

Furthermore, Wazuh offers a powerful RESTful API that allows the interaction of the Wazuh manager with web browsers, command line tools like cURL <sup>8</sup>, or any scripts or programs that can make web requests. This, combined with the RESTful APIs provided by ElasticSearch, will greatly aid to 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 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.

For the initial stage of the MVP the HEIR SIEM main role is providing all necessary information via Elastic to be depicted in the Forensics Visualization Toolkit (FVT).

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
3 <sup>rd</sup> party Logs	Date based logs, Syslog Logs, IIS Logs tec.	Wazuh agents are able to accumulate 3 <sup>rd</sup> 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 <sup>9</sup>	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 3: HEIR SIEM interfaces

### 2.3.1.3 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-threat level of criticality, 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 machine learning module takes the input from HEIR IoT (Logs) and process the records in a way to differentiate the anomalies and non-anomalies. After that, the ML component process the results in a detailed report. The result is visualized in FVT toolkit to represent the results in a tangible way.

<sup>8</sup> <https://curl.se/>

<sup>9</sup> <https://documentation.wazuh.com/current/user-manual/api/reference.html>

The selected model/algorithm is the Random Forest algorithm which is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. Erreur ! Source du renvoi introuvable. illustrates the structure of Random Forest algorithm.

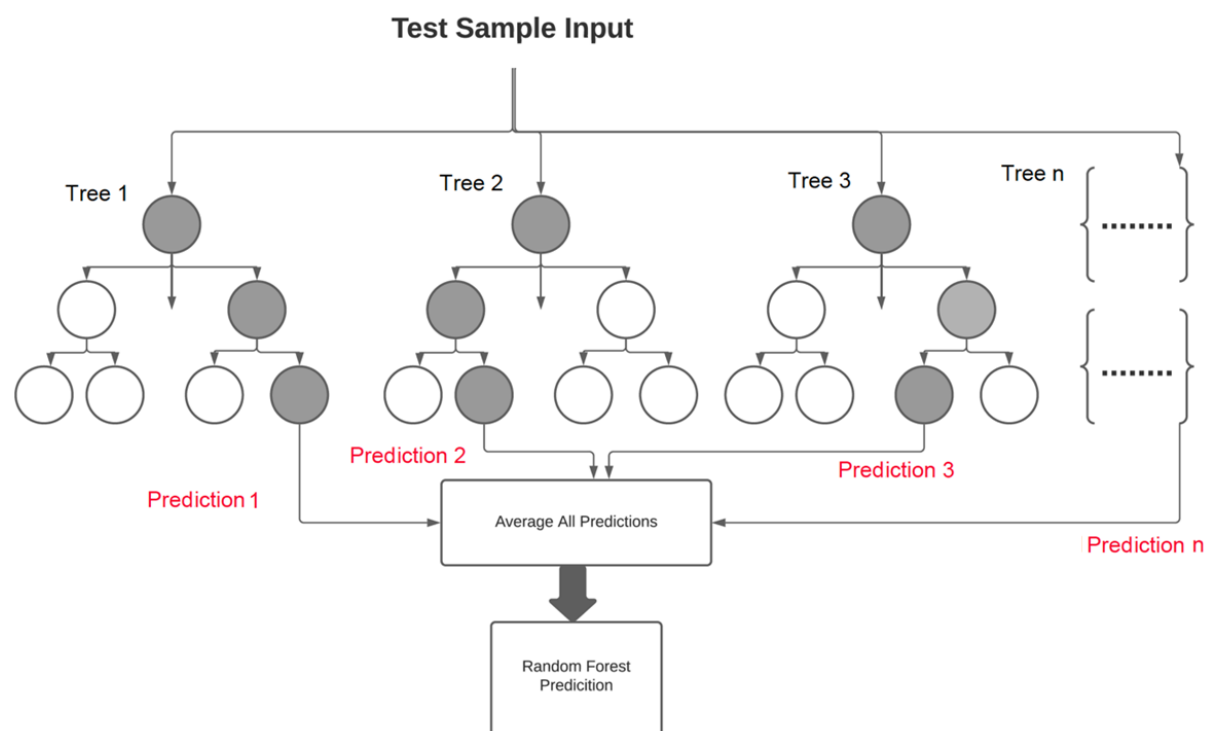


Figure 5 Algorithm Structure

Module's 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 file contains (id, user, hospital, department, role, action, caseID, vpn, datetime)
Output		
Name	Type	Description
ML output	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, Anomaly, Z-score) + ML score in the bottom of the JSON format. For Instance (from one use Case):

		<pre>{   "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",   "Z_score": "0.6494445760764975"} </pre>
--	--	--

Table 4: HEIR ML-based Anomaly Detection interfaces

#### 2.3.1.4 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/>).

Advancing beyond the MVP, its capabilities have been expanded, including the development of a HEIR Fybrik module, to provide policy-driven control of data accessed from a FHIR (<https://www.hl7.org/fhir/>) server, token authentication for data requests, as well as logging data transaction requests to Kafka.

The creation of a data pipeline through Fybrik invocation is initiated by a specified requester. All FHIR requests for data must include a JSON Web Token (JWT) in the REST header which encodes the requester. The PAF will reject any received FHIR request which encodes a different requester than the one that invoked the data pipeline.

The developed HEIR Fybrik module utilizes decisions from the Fybrik Policy Manager to provide fine-grained access both to individual FHIR resources, as well as redact individual fields within specified FHIR resources.

All requests to access data are logged as JSON records to a dedicated topic on Kafka for consumption by the HEIR blockchain framework. An example of a logged access request can be seen in:

```
{
  "Timestamp": "2022-02-06 15:41:43",
  "Requester": "EliotSalant",
  "Query": "Observation",
  "ClientIP": "127.0.0.1",
  "assetID": "sql-fhir/observation-json",
  "intent": "research",
  "Outcome": "UNAUTHORIZED"
}
```

*Note: The “Outcome” field will show whether or not the requester is allowed access to the requested FHIR resource.*



Module's interfaces		
Input		
Name	Type	Short Description
None	N/A	
Output		
Name	Type	Description
Kafka logging	JSON	{"Timestamp": "", timestamp when access request was received "Requester": "", from encode field in the passed JWT "Query": "", Send in the REST request "ClientIP": "", "assetID": "", Corresponds to requested dataset "intent": "", Declared at Fybrik invocation "Outcome": ""} "AUTHORIZED" or "UNAUTHORIZED"

Table 5: HEIR PAF interfaces

#### 2.3.1.4.1 Blockchain-based HEIR Auditing Mechanism

The goal of the HEIR Auditing mechanism is to provide enables tamper-resilience and, thus, practical immutability for critical, data access related logs that are generated through the HEIR Privacy Aware Framework. 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 simple a 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 the direct integration with the PAF. The Kafka consumer listens on a specific Kafka topic for incoming audit logs (described in detail in Section 2.3.1.4), 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	<pre>{   "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" }</pre>
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 6: 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<sup>10</sup> environment, tightly integrated with the Privacy Aware Framework.

## 2.3.2 HEIR Core Framework

### 2.3.2.1 HEIR Client's Processing system (Novel 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.

<sup>10</sup> <https://kubernetes.io/>

Table 7: Novel HEIR Client interfaces

## 2.3.2.1.1 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 8: HEIR HNM interfaces

## 2.3.2.1.2 HEIR Exploit Tester (HET)

Establish 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 9: HEIR HET interfaces

## 2.3.2.1.3 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 10: HEIR HET interfaces

### 2.3.2.2 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) and,
- the Vulnerability Assessment module.

The RAMA Score represents the security level of a specific sector clinic by aggregating the respective results.

Module's interfaces		
Input		
Name	Type	Short Description
HEIR Client Input	JSON	<p>The Local RAMA Calculator receives input from the HEIR Client. The latter incorporates scores from:</p> <ul style="list-style-type: none"> <li>• the HEIR Network Module (HNM)</li> <li>• the HEIR Exploit Tester (HET)</li> <li>• the HEIR Cryptographic Checker (HCC) and</li> <li>• the Vulnerability Assessment module.</li> </ul>
Output		
Name	Type	Description
Local RAMA Output	JSON	The Local RAMA Calculator provides the RAMA Score (base and temporal score) and the corresponding metadata.

Table 11: HEIR Local RAMA calculator interfaces

### 2.3.2.3 HEIR Aggregator

The HEIR Aggregator is a component of the HEIR framework designed for health institutions with multiple independent departments. 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 on 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. hourly), read the most recent outputs from the HEIR clients from the Elasticsearch storage, compute the aggregates, and write the aggregated values for RAMA and event statistics to the Elasticsearch

storage, where they can be accessed by the HEIR GUI. In the following iterations of the MVP, the Aggregator will also send its output to the HEIR Observatory database.

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

#### 2.3.2.4 HEIR Observatory

The HEIR Observatory is responsible to collect, analyse and present the results of all the deployed HEIR Clients inside the hospitals in order to provide global insights on the level of security in healthcare environments. The Observatory database will store 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. The available results are presented in the 2<sup>nd</sup> layer of visualization.

##### 2.3.2.4.1 HEIR global benchmarks and Security Assurance (Global RAMA Score Calculator)

The HEIR global benchmarks and Security Assurance (Global RAMA Score Calculator) 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 2 <sup>nd</sup> Layer of visualization of the Observatory.

Table 13: HEIR Global RAMA Calculator interfaces

##### 2.3.2.4.2 Observatory Database

The HEIR Database handles the data being 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. At the time of this writing, ElasticSearch suffices for database usage pending unidentified data needs that may arise which would potentially mandate the need of an alternate database type.

### 2.3.2.4.3 Analytics Engine

The Analytics Engine is responsible to collect and analyze 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 refer 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 2 <sup>nd</sup> layer of Visualization.

Table 14: HEIR Analytics Engine interfaces

## 2.3.3 HEIR Visualization

### 2.3.3.1 1<sup>st</sup> Layer of Visualizations

The HEIR Client GUI (HCG) includes visualizations of information generated by the 1<sup>st</sup> level services running inside a hospital environment. This information is only available via authentication performed by KeyCloak<sup>11</sup> 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.

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 refer to the output of the HEIR Client's modules. (Vuln. Assessment, HCC, HNM, HET, security status information and more.) Source: HEIR Aggregator.

Table 15: HEIR 1st Layer of Visualization interfaces

<sup>11</sup> <https://www.keycloak.org/>

### 2.3.3.2 2<sup>nd</sup> Layer of Visualizations

The 2<sup>nd</sup> 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.

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.

Table 16: HEIR 1st Layer of Visualization interfaces

## 2.4 Modules' Communication

In this section, the communication among the previous modules is presented. The following diagrams are on implementation level, and they are based on the use case scenarios elaborated in WP6 (

ANNEX II – Use case scenarios/playbooks).



### 2.4.1 Vulnerability management for outdated software & “infected” device

The following figure (Figure 6) indicates the module’s flow on providing insights related to outdated software to the System Administrator of PAGNI, HYGEIA CUH pilots through the 1<sup>st</sup> layer of visualizations.

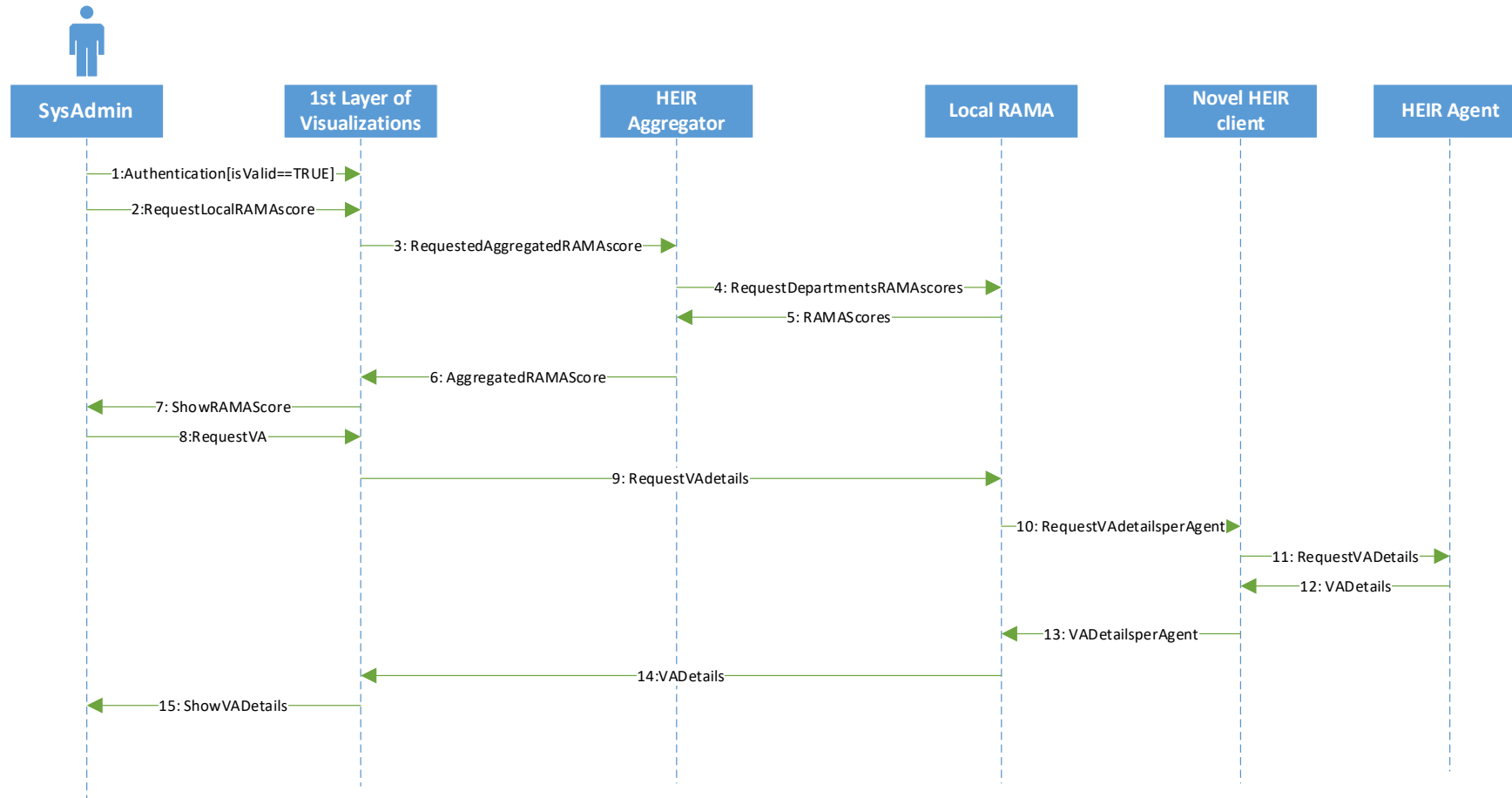


Figure 6: Data flow/sequence diagram (PAGNI, HYGEIA and CUH use case scenarios)

## 2.4.2 NSE

The following figures (Figure 7 & Figure 8) indicates 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”).

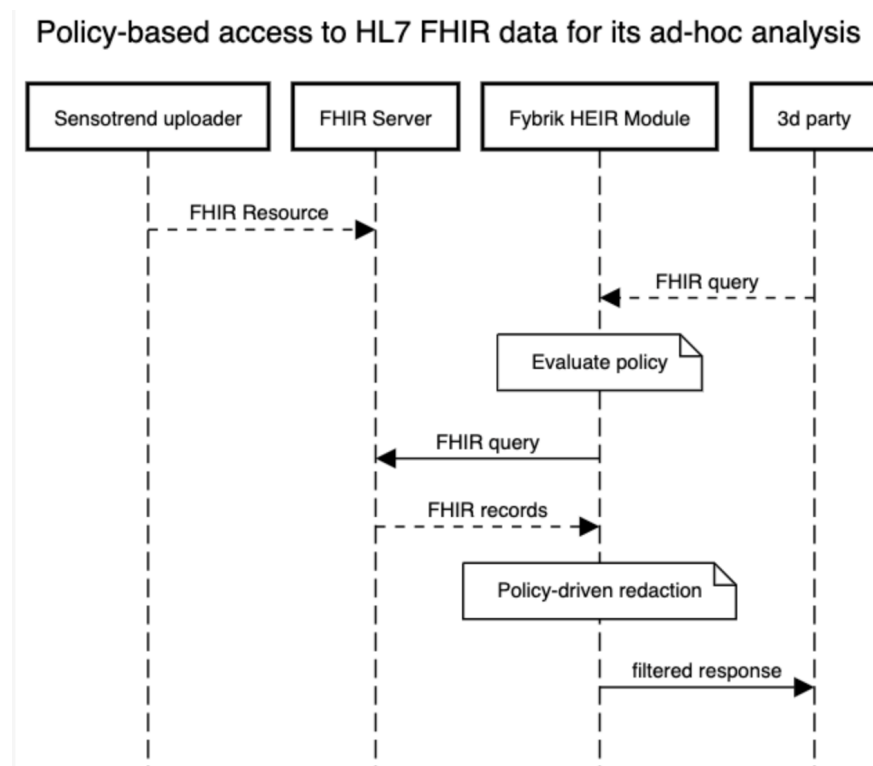


Figure 7: Data flow/sequence diagram for the “pull” model (NSE use case scenario)

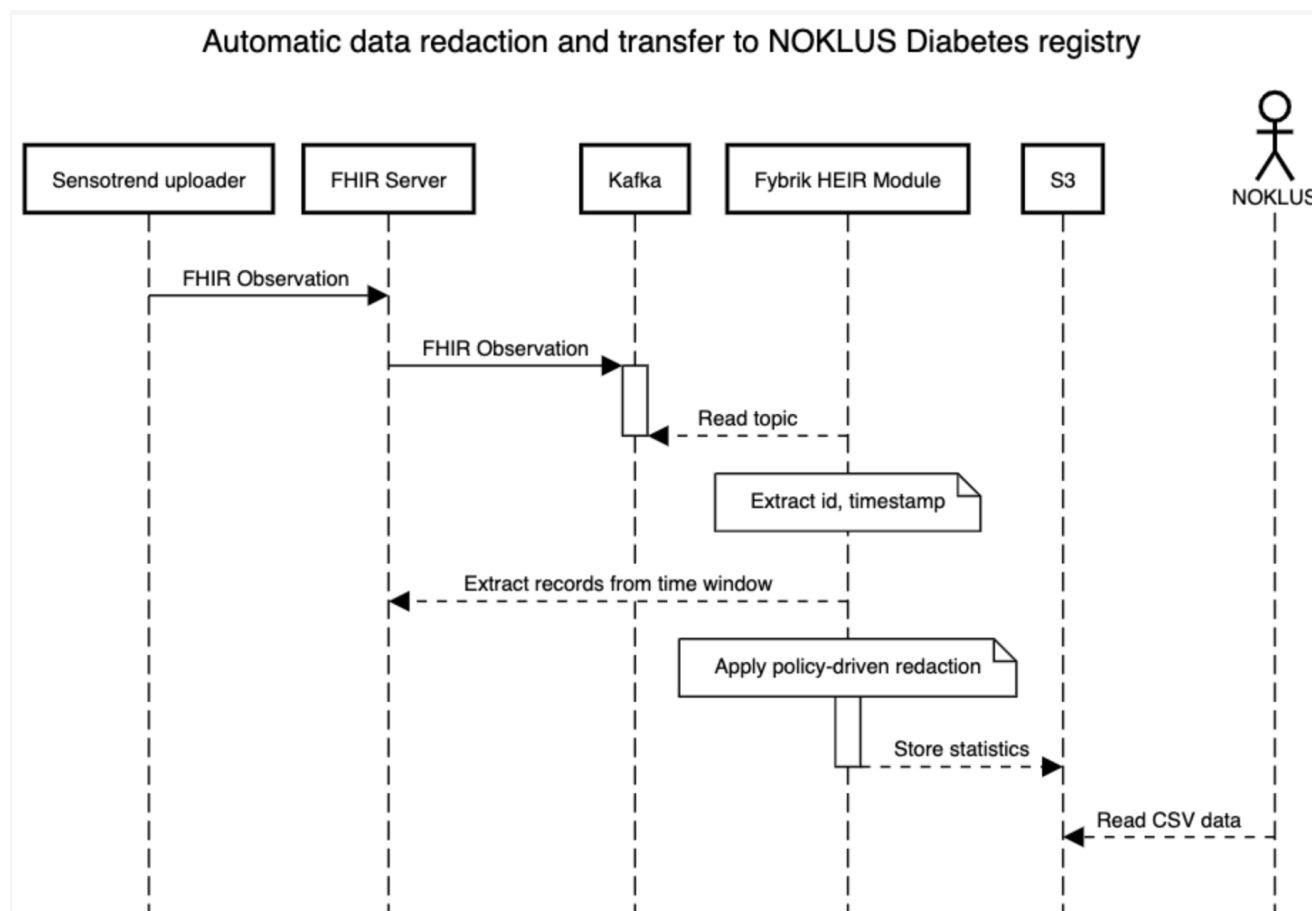


Figure 8: Data flow/sequence diagram for the “push” model (NSE use case scenario)

### 2.4.3 Observatory

The following figure (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.

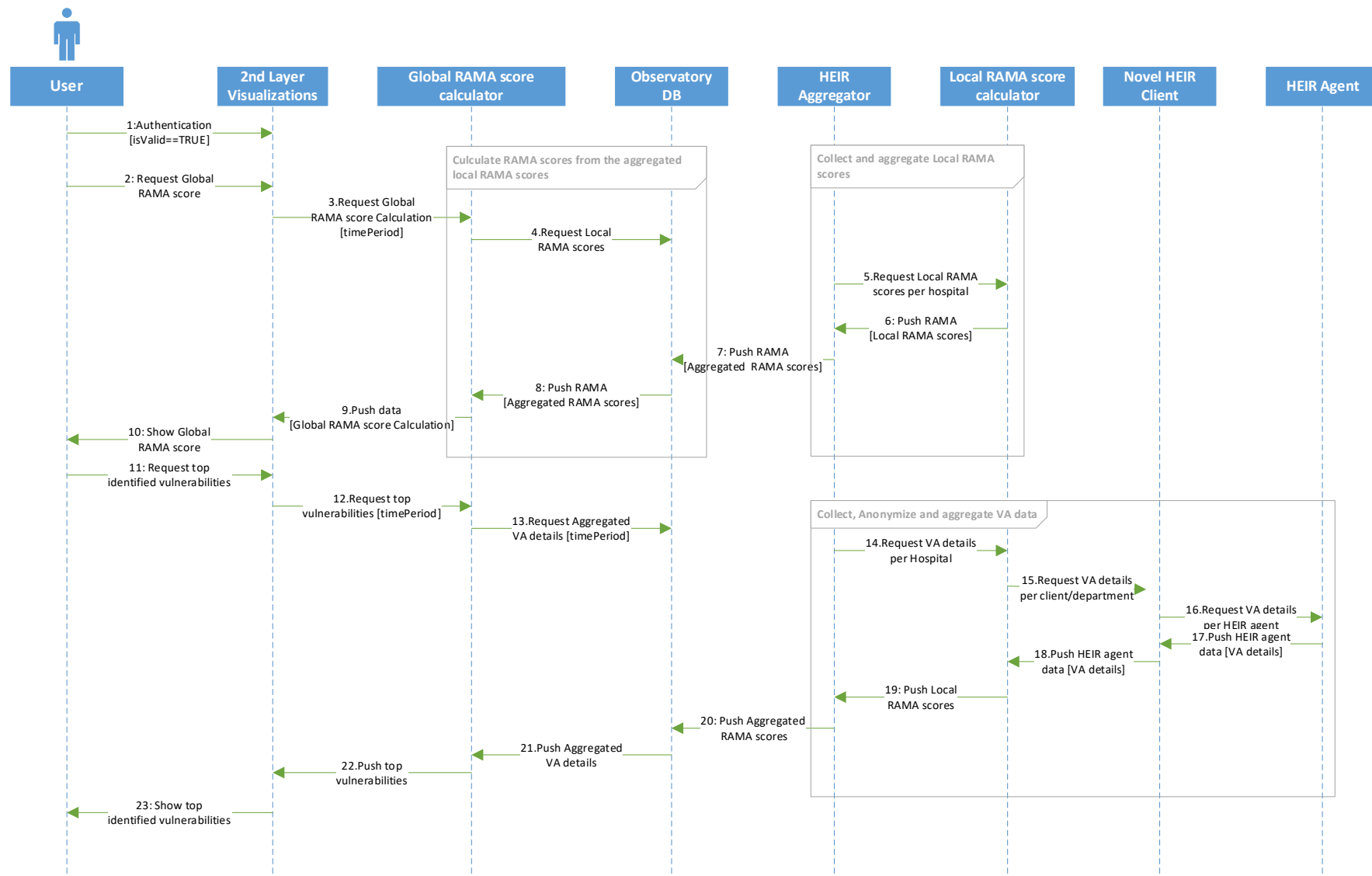


Figure 9: Data flow/sequence diagram on the Observatory use case scenario/playbook

### 3 Integration

For the integration purposes of the project, we are following 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 guides the integration process of the components together with the use of specific tools that will foster this process. The integration time plan includes the next steps towards the integration of the various components into the HEIR Architecture.

#### 3.1 Methodology

Agile software development (ASD) is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organising, cross-functional teams, addressing the development efforts performed in the various stage of a project.

There are many specific ASD methods. Most of them promote development, teamwork, collaboration, and process adaptability throughout the life cycle of the project. ASD goes beyond traditional software development processes (such as Waterfall) and exploits an evolutionary method that is an iterative and incremental approach to software development and integration. Thus, the requirements and design phases are iteratively met with the development phase to incrementally produce system software releases, which can be assessed over the suitability, the maturity and the immediate business value. On top of them, ASD foresees an intense testing phase, in which the unit testing is achieved from the developer's perspective and the acceptance testing is conducted from the customer's perspective. Thus, the major difference with respect to the iterative approaches of plan-based methodologies (like RUP and Spiral) is the fact that requirements and testing are part of the actual development iterative process, and the target stakeholders can be progressively involved in the development process aiming to deliver high quality software.

“The best architectures, requirements, and designs emerge from self-organising teams”. ASD manifests that “working software is the primary measure of progress”. Thus, agile methodology approaches the requirements of a software environment, iteratively by frequently delivering working software prototypes. These prototypes enable the project development and business teams to work together and maximise the quality of the produced output. ASD can be suitable for the development strategy of an ICT solution, mainly because:

A parallel process between the development of the planned software solution and the verification of the requirements can be followed, leading to business-oriented people to actively participate in the specification of use cases and the evaluation of the system developments and provide valuable feedback in an iterative way:

- The work on individual and independent development fields is split among small groups comprising the separate development teams.
- As a ready to the market solution is envisaged rapidly, the solution can benefit from frequent releases to align the work done among the individual teams.
- The producing releases can be exchanged among senior technical teams and business-oriented groups to evaluate the effectiveness of the solution in real business situations.

Extreme Programming (XP)<sup>12</sup> and SCRUM<sup>13</sup> are the most popular agile-based software development methodologies. Such methodologies can decrease the time required to produce releases and engage the customer in the development process, maximising the possibilities for a high-quality output. On the other hand, the effectiveness of these methodologies is tightly coupled to the proper communication between business end users and engineering teams, while the lack of specific documentation might put the final acceptance at risk.

### 3.1.1 Continuous Integration Process

Continuous Integration process fits perfectly with the Agile Methodology. Continuous integration (CI) comprises practices such as often builds and additional checks so as to prevent bugs. In order to enable automatic daily builds, CI software gathers the whole source code in one place (with different revisions), automate the build process and testing, and provides the latest working executable to anyone involved in the project. The CI model comprises a set of activities for the process implementation: building the system, running tests, deployment activities, and finally reporting test and deployment results.

The practice of CI assumes a high degree of tests, which are automated into the software: a facility that can be seen as “self-testing code”, often using a testing framework, such as Jenkins<sup>14</sup>.

Ideally each automated build performs the following steps:

- Compilation and creation of the build package
- Creation of a report on code metrics (such as package, dependency analysis and cyclomatic complexity)
- Creation of a report on how much source code follows declared Coding Standard
- Creation of a report on possible bugs by a static code analysis execution
- Automated deployment in testing environment
- Execution of automated test cases and creation of test report
- Publishing the build artifacts
- Notification to stakeholders and involved developers about build outcome by email and on central build dashboard

CI is a software development practice where:

- Members of a team integrate their work frequently.
- Each integration is automatically compiled and built
- New artifacts are automatically deployed in testing environment
- Integration and Unit Testing is performed automatically
- Automated notifications to developers are sent
- Creation of reports about software quality code

In the context of the HEIR project, we are aiming to incorporate as many of the core CI/CD features and practices, namely the migration of the components docker images under Kubernetes and the creation of a central repository so to aid the seamless deployment of the

---

<sup>12</sup> <http://www.extremeprogramming.org/>

<sup>13</sup> <https://www.scrum.org/>

<sup>14</sup> <https://www.jenkins.io/>

HEIR components' updates to all pilots. However, it is not foreseen that a complete CI/CD process will be feasible to be applied till the end of the project.

The workflow is also depicted in **Figure 10**.

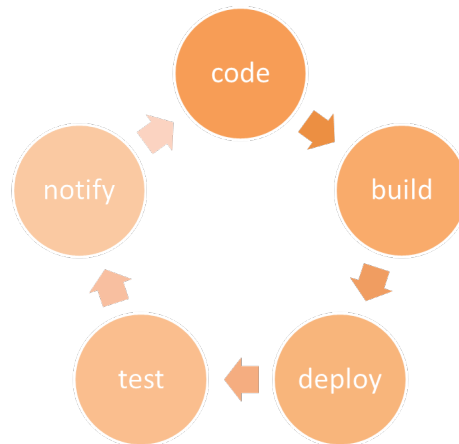


Figure 10 Continuous Integration Workflow

### 3.1.2 Off-the-self tools for development infrastructure

#### Jenkins CI:

For the continuous integration workflow of the integration mechanisms for the HEIR integrated framework, there are two possible solutions that could be followed, either Hudson<sup>15</sup> or Jenkins Continuous Integration server. Both systems:

- are open-source tools to perform Continuous Integration as described in the previous paragraph
- monitor a SCM (Source Control System) and if changes occur to start and monitor a build system (for example Apache Ant or Maven).
- will monitor the whole process and provide reporting functionality and notification functionality to report success or errors.

Both tools control the builds of the project through a graphical interface accessible by the administrator of the integration process. Thus, the integrator can easily change the maven goals to be executed as well as the execution mode and time of the scheduled maven jobs.

**Apache Maven:** Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. It is important that all integration modules are compatible with Apache Maven rules, sharing the same .pom file. In this way, the integration will be easily controlled through Apache Maven and moreover, the project will be independent of an IDE; as a maven project can be opened in Eclipse<sup>16</sup>, NetBeans<sup>17</sup>, IDEA<sup>18</sup> etc.

**Redmine<sup>19</sup>:** An enhanced wiki and issue tracking system for software development projects.

<sup>15</sup> <http://eclipse.org/hudson/>

<sup>16</sup> <https://eclipse.org/>

<sup>17</sup> <https://netbeans.org/>

<sup>18</sup> <https://www.jetbrains.com/idea/>

<sup>19</sup> <http://www.redmine.org>

- A standard issue creation process must be followed by all the developers
  - Tickets assigned directly to feature owner (status=new) - very important to add type: feature/bug/support and a due date
  - The owner has to accept the ticket (status=in progress) or redirect it to other developer.
  - If developer fixes issue the ticket shall be redirected to the original submitter with status=resolved and description of the fix.
  - The original submitter is responsible to test and close the ticket.

**Git<sup>20</sup>:** A software versioning and a revision control system distributed under a free license.

**Sonar<sup>21</sup>:** Sonar is an open-source software quality platform. Sonar uses various static code analysis tools such as Checkstyle<sup>22</sup>, PMD<sup>23</sup>, FindBugs<sup>24</sup>, Clover<sup>25</sup> to extract software metrics, which then can be used to improve software quality.

In a more descriptive way, the workflow of the procedure is depicted in **Figure 11**

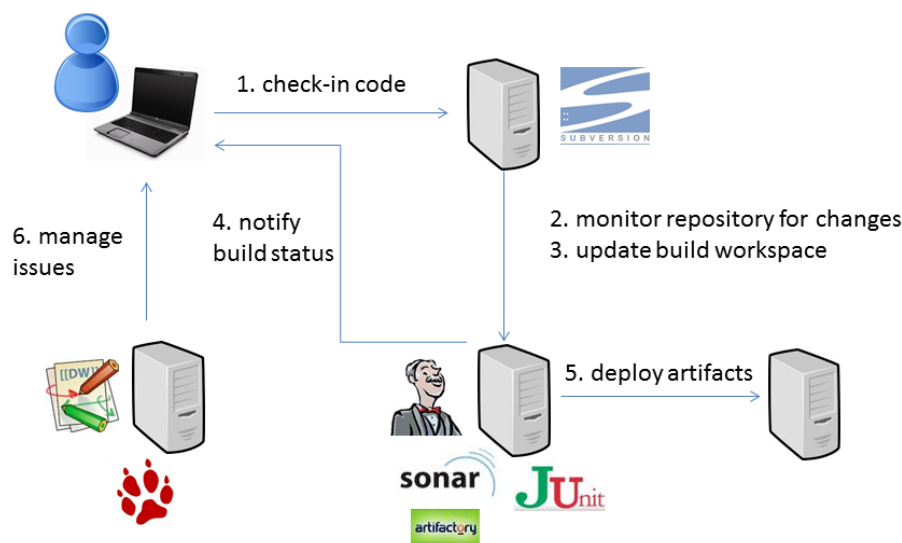


Figure 11 Continuous Integration Workflow (2)

The development of some components might also follow a remote deployment setup and the provision of the tools as fully functional *Docker* containers.

**Docker** is an open platform for developers and System Administrators to build, ship, and run distributed applications. Consisting of the Docker Engine, a portable, lightweight runtime and packaging tool, and the Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments.

<sup>20</sup> <https://git-scm.com/>

<sup>21</sup> <http://www.sonarqube.org>

<sup>22</sup> <http://docs.sonarqube.org/display/PLUG/Checkstyle+Plugin>

<sup>23</sup> <http://docs.sonarqube.org/display/PLUG/PMD+Plugin>

<sup>24</sup> <http://docs.sonarqube.org/display/PLUG/FindBugs+Plugin>

<sup>25</sup> <http://docs.sonarqube.org/display/PLUG/Clover+Plugin>



### 3.1.3 HEIR Management/Communication tools

Redmine will be used in the second half of the project for issue tracking during the development process where solution evaluation will start. It is actually a free and open source, web-based project management and issue tracking tool. It has been written using the Ruby on Rails framework, and it is cross-platform and cross-database. Some of the main features of Redmine<sup>26</sup> are:

- Multiple projects support
- Flexible role-based access control
- Flexible issue tracking system
- Gantt chart and calendar
- News, documents & files management
- Feeds & email notifications
- Per project wiki
- Per project forums
- Time tracking
- Custom fields for issues, time-entries, projects and users
- SCM integration (SVN, CVS, Git, Mercurial, Bazaar and Darcs)
- Issue creation via email
- Multiple LDAP authentication support
- User self-registration support
- Multilanguage support
- Multiple databases support

The Redmine platform will be used for issuing tickets in the process of software development. The issuer will have to specify the type of issue: feature/bug/support. In addition to type, the issuer must assign a due date for the ticket to be resolved (status = new). The developer who will undertake to develop/fix/resolve the issue, must first accept the ticket (status = in progress). If the developer accomplishes the task, he/she must redirect the ticket to the original submitter (status = resolved) and provide description of the fix.

Discord<sup>27</sup> is used as a common communication tool for technical instant communication among the project partners. This is a tool that has widespread usage and can be arranged within our Consortium to be used in conjunction with emails for the better and faster communication and coordination regarding technical issues, software development and deployment.

## 3.2 System & Network Specifications

### 3.2.1 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 realisation of HEIR framework (T5.2) has been started in Month 8 of the project, the necessary hardware/software infrastructure and environment set up is taking place towards the end of the project.

---

<sup>26</sup> <http://www.redmine.org/projects/redmine/wiki>

<sup>27</sup> <https://discord.com/>

The technical specifications of the HEIR VM deployed in the pilot environments, based on the initial pilots' environment are presented in the table below.

<b>OS</b>	Ubuntu Server 20.04.3 LTS
<b>CPU</b>	4 X vCpu
<b>RAM</b>	16GB
<b>Disk</b>	100GB disk

Table 17: HEIR VM Requirements

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

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

Table 18: Windows Agent Requirements

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

Table 19: Linux Agent Requirements

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

### 3.2.2 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<sup>28</sup> server installed, HYGEIA had already deployed FortiVPN<sup>29</sup> whereas for the Croydon environment 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 have to be able to communicate with the main Kafka message broker.

The HEIR platform 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

<sup>28</sup> <https://openvpn.net>

<sup>29</sup> <https://www.fortinet.com>

### 3.3 Access Control and Identity Management System

In order to implement an Access Control and Identity Management System (ACIMS), the HEIR integrated framework mainly relies on the mechanisms provided by Keycloak<sup>30</sup>.

Keycloak ships with robust user management and security features including password policies, user reminder settings, and complete log-in security procedures. A full set of permissions for an extended set of actions is provided via a web interface to the administrators of a Keycloak. This feature together with the role management facilities, allows administrators to set up an access control policy according to the needs of the project.

The HEIR Access Control and Identity Management System are heavily relying on these security features of Keycloak. The software components will be fully integrated with Keycloak in order to take advantage of the administrative capabilities offered by the Keycloak and allow administrators or moderators to manage roles and permissions in an easy and effective manner via an online user interface.

There are a number of risks when deploying an access control and identity management system. To this end, we have identified the security risks and the mitigation strategy of HEIR as presented below.

### 3.4 Security Risks and Mitigation

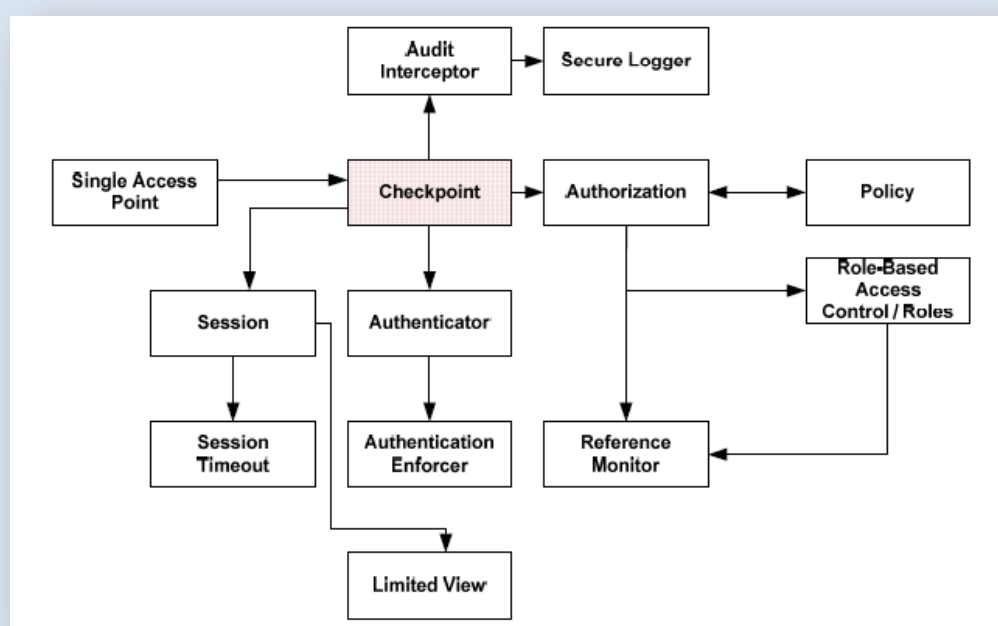
<i>Risk</i>	<i>Likelihood</i>	<i>Impact</i>	<i>Description</i>
<b>Insecure access control</b>	Low	High	A common requirement of most multi-user information systems is to provide a mechanism for access control. Access control comprises identification, authentication and authorization. By providing insecure 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
<b>Identity theft</b>	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 which stores 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.
<b>Data Leakage</b>	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 m-RESIST Platform will maintain sensitive patients' data as

<sup>30</sup> <https://www.keycloak.org/>

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 12** depicts a system of security patterns to implement the HEIR integrated framework access control mechanism. By implementing the “Single Access Point” pattern, only a 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 12 System of Security Patterns realizing Access Control*

The HEIR security features are applied system-wise, covering all the architecture. The HEIR ACIMS ensures that only users with appropriate permissions will be able to access the data relying in the platform'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*

### 3.5 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
<b>Wrong understanding of the modules to be developed</b>	Close communication amongst partners (Technical). Delivery of Clear and representative reports for requirements, architecture etc.
<b>Lack of Collaborative framework</b>	Usage of Collaborative tools and Technologies accessible for all partners. (Discord, SVN, etc.)
<b>Inconsistency of selected technologies</b>	Clear definition of the technologies to be used in the related documentation. Usage and exchange of internal documentation for the technical partners.
<b>Inadequacy of the infrastructure</b>	Strong Assessment of the selected technologies and infrastructure. Low cost doesn't mean gain for the project.
<b>Delays of Module Delivery</b>	All the partners need to respect the deadlines. WP coordinators and Task leaders have to prevent delays with early notifications and alerts.
<b>Integrated System of high complexity</b>	Non-technological partners and partners with expertise on similar to HEIR framework systems need to assess system mockups and workflows prior to the development phase.
<b>Low level of Usability</b>	End-Users and Non-Technological partners have to get involved in the system assessment and provide feedback in a constructive way. Creation of an easy to use and collaborative evaluation framework.
<b>Insufficiency of HEIR integrated framework related to user needs</b>	End-Users and hospitals' representatives' involvement from requirements gathering phase.
<b>HEIR Outcomes of poor quality</b>	Adaptation of a methodology for software validation (eg. 1012-2012 - IEEE Standard for System and Software Verification and Validation).

Table 21: Identified Integration & Development risks and mitigations

### 3.6 Testing methodology

The HEIR integrated framework will be tested to assess the maturity of the technical implementation and the alignment to the user requirements from a technical perspective. The technical assessment of the HEIR integrated framework is supported by monitoring the technical parameters of the system performance and aims to determine how closely the integrated framework meets the technical requirements and the functional specifications.

HEIR development tasks are tested according to established standards on software assurance process. This process aims to assess the efficiency of the system functionalities and provide evidence that the integrated framework is fully functional and available for release through a

software assurance process. In principle, software assurance can be realised by evaluating both the software itself (the product) and how it has been developed (the process). Both aspects are important for a system targeting to support scalable functionalities.

However, in prototypes such as the HEIR integrated framework, the software assurance of the process becomes quite difficult. This is mostly due to the constant changes in both the design and the requirements of the software during the project cycle itself, as a result of the on-going work in the other work packages and the living lab approach used for the user requirements gathering and the user engagement process. Thus, the software assurance process means testing the outcome of the framework is preferred, considering this as an integrated framework consisting of individual components which are integrated using the architecture principles that are analysed in this document.

Software validation is the “confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled”. Since software is usually part of a larger hardware system, the validation of software typically includes evidence that all software requirements have been implemented correctly and completely.

In general, software validation is the process of developing a “level of confidence” that the system 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 end product.

Software validation is realised through quality models. In the past, different quality models have been proposed, each of which addresses different quality attributes that allow evaluating the developed software. Some of the most well-known are:

- McCall's model of software quality (GE Model, 1977), which incorporates 11 criteria encompassing product operation, product revision, and product transition.
- Boehm's spiral model (1978) based on a wider range of characteristics, which incorporates 19 criteria. The criteria in both, this and the GE model, are not independent as they interact with each other and often cause conflicts.
- ISO 9126-1 incorporates six quality goals, each goal having a large number of attributes. These six goals are then further split into sub-characteristics, which represent measurable attributes (custom defined for each software product).

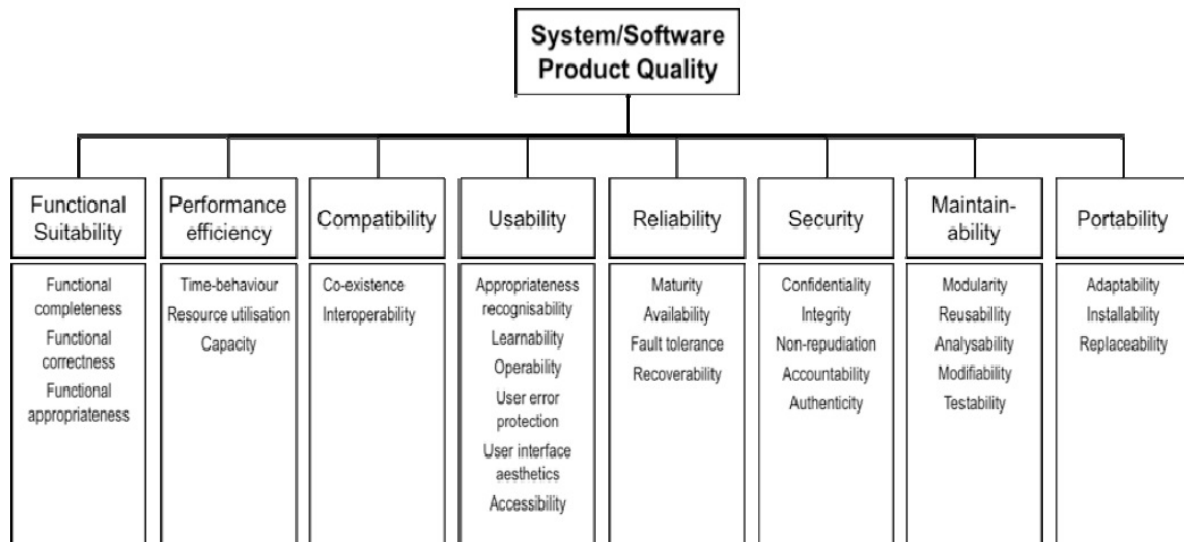
Recently the BS ISO/IEC 25010:2011 standard about system and software quality models has replaced ISO 9126-1. Applying any of the above models is not a straightforward process. There are no automated means for testing software against each of the characteristics defined by each model. For each model, the final attributes must be matched against measurable metrics and thresholds for evaluating the results must be set. It is then possible to measure the results of the tests performed (either quantitative or qualitative/observed).

For the HEIR case, we have adopted the ISO/IEC 25010:2011 standard, which is the most widespread reference model, and it includes the common software quality characteristics that are supported by the other models. This standard defines two quality models providing a consistent terminology for specifying, measuring and evaluating system and software product quality:

- Quality in use model, which is composed of five characteristics that relate to the outcome of interaction with the system and characterises the impact that the product can have on the stakeholders.

- Product quality model, which is composed of eight characteristics that relate to static properties of software and dynamic properties of the computer system.

For our case, the product quality model is adopted. The eight characteristics are further divided into sub-characteristics, as shown in **Figure 13** below:



*Figure 13 The ISO/IEC 25010:2011 system/software quality model characteristics*

For each of the sub-characteristics, a metric/measurable attribute will be defined, along with thresholds. These metrics and thresholds are customised for each software product, which in our case is the HEIR integrated framework (consisting of individual components). By evaluating these metrics, we will be able to assess the overall quality of our platform and the percentage to which we were able to meet the user requirements (reflected to system specifications and functionalities) defined during the design phase of the project.



## 4 Time plan

The integration process of the HEIR framework consists of 3 iterations. During the first iteration the technical requirements for all the project's modules were delivered. After the delivery of the technical requirements, the process continued with the development of the first versions of the HEIR modules. In the meantime, the integration of the HEIR modules into one integrated framework started and delivered the MVP. This iteration started on month 8 and ended on month 12. ITML and the pilot partners have set up the hardware/software infrastructure.

Accordingly, during the second iteration of the HEIR integration process the updates of the HEIR modules' specifications and the second versions of the HEIR modules will be delivered by mid-month 19 (officially M18). In parallel, the integration of the second versions of the HEIR modules into one integrated solution started and delivered the HEIR integrated framework intermediate version. This iteration starts on month 13 and ends on month 18. After the second iteration, a preliminary testing results from the pilot use cases will be considered for the development of the final release of the HEIR solution.

The same procedure will apply for the third iteration. Regarding a concise view of the HEIR integrated, the following table shows its versions and the months at the end of which they should be delivered.

HEIR Integrated System	Status	Date
MVP	Delivered	M12
HEIR integrated framework intermediate version	Delivered	M19
HEIR integrated framework final version	To be delivered	M30

Table 22: Integration time plan



## 5 Conclusion and Future Steps

This deliverable is the accompanying report of the HEIR Integrated framework intermediate version. This version includes an almost complete set of functionalities covering the needs of the pilot use cases and enabling the HEIR stakeholders to test and evaluate at a great extent the concepts and knowledge conveyed by the project. The integration of new submodules of the Novel HEIR Client, as well as the updates of existing ones was a straightforward process and proved the flexibility and extensibility of the framework. The architecture allows for easy adaption of new capabilities and functionalities developed in the context of technical work packages and made it possible to quickly present the latest updates through a continuous delivery of new releases. In the second half of the project the remaining modules of the HEIR Client, Threat Detection Module (TDM) and HEIR Whitelist Checker (HWC) will be added to the HEIR infrastructure and functionalities. Similarly, NSE pilot will be integrated in the HEIR infrastructure, and PAF will be incorporated in all applicable HEIR use cases.

The next steps include continuous updating of the HEIR framework solution according to the feedback that will be received during the evaluation period, as well as the ongoing work in the technical WPs. Internal intermediate releases are planned to continue during the second half of the project so as to facilitate the project time plan and the needs of the other WPs in view of the final release which is officially planned by M30 of the project.

## 6 ANNEX I – Components APIs

### 6.1 Blockchain-based HEIR Auditing Mechanism API

The current exposed endpoints are described below.

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

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

## 7 ANNEX II – Use case scenarios/playbooks

### 7.1 PAGNI use case scenario/playbook

<b>Use Case Id</b>	SysAdmin(PAGNI):01
<b>Use Case Name</b>	Outdated software
<b>Use Case Summary</b>	Outdated software risks poisoning the systems of the hospital and wreaking havoc on the organization's security. HEIR security mechanisms can detect outdated software to the servers and workstations that belong to the hospital and inform the IT department for the issues and the actions needed.
<b>Actors</b>	IT administrator of the hospital
<b>Preconditions</b>	None
<b>Trigger</b>	The administrator opens the HEIR web application 1 <sup>st</sup> Layer GUI as part of her/his daily work routine.
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1. The system administrator logs in to the HEIR local dashboard (1<sup>st</sup> Layer GUI).</li> <li>2. Checks the local RAMA score and identifies that the score has been reduced.</li> <li>3. Having a closer look at the 1<sup>st</sup> Layer GUI of HEIR, the Vulnerability Assessment module shows that one of the connected clients (a workstation at a clinic) has malicious findings.</li> <li>4. Opens the vulnerabilities details for the specific client and the system informs her/him that a client has an outdated version of the Mozilla Firefox browser.</li> <li>5. The administrator goes to the specific workstation or connects via remote desktop and updates the software (Mozilla).</li> <li>6. Opens again the 1st Layer GUI of HEIR and identifies that there are no vulnerabilities any more for the specific client and that the local RAMA score has been increased.</li> </ol>
<b>Alternate Flows</b>	None
<b>Postconditions</b>	None

### 7.2 NSE use case scenario/playbook

#### 7.2.1 The Patient's view

A patient in a home setting plugs the CGM-communicating device into the laptop. The screen shows the Sensotrend uploader – a patient uses her credentials to log in, shows a selection of devices that can be used with the Uploader and picks the one she owns. Then she triggers the upload of all data options.

Once the upload is done, the patient opens a web browser (this already happens in a screen recording mode) and logs into the Sensotrend dashboard. The data is visible in the Dashboard with updated statistics.

#### 7.2.2 The Developer's view.

A Developer walkthrough using the Cloud UI that shows individual servers, how they are set up etc. Developer's console (terminal) window presents:

- Running containers in the Kubernetes cluster, i.e., Sensotrend, Fybrik (Privacy-Aware framework) and HL7 FHIR server;

- (Optional) Event if turning off one of the servers that are backing up the Kubernetes cluster for (fictional) security reasons (for example, the server, for some reason, has skipped the patching window and became vulnerable), the whole solution remains functional because the other server is still operational.
- The solution has become vulnerable due to a newly discovered security vulnerability, but we quickly deploy security fixes due to a sustainable architecture design (continuous integration, docker images, Kubernetes)

### 7.2.3 PAF view (or Policy maker view)

SGV (sensor glucose values) Observation records in the HL7 FHIR server that correspond to the data previously visible in the Dashboard. A set of policies is defined in the Open Policy Agent (OPA), which is a part of PAF, where the policies are typically defined by a Governance Officer.

The demo consists of two different scenarios:

- Automatic data redaction (calculation of statistical summary) from data residing in HL7 FHIR server and transfer of this data to NOKLUS Diabetes registry
- Policy-based access to HL7 FHIR data for its ad-hoc analysis in a Jupyter Notebook

### 7.2.4 The NOKLUS view

In this scenario, we aim to transmit the summary statistics to NOKLUS. While many patients and clinicians need a visual understanding of data, some actors need only a posteriori data analysis on data. For NOKLUS, the primary needs are to have summary statistics about GCM data every 14 days.

The FHIR server writes to a Kafka queue where the PAF reads from and the policy affects redacted data. We log in to the blob storage [2] and explore the blob content and identify newly redacted data (CSV files). The user downloads the CSV file, that has just been computed based on newly processed data and further verifies the content inside of the file (MEAN, STD, and TIR values).

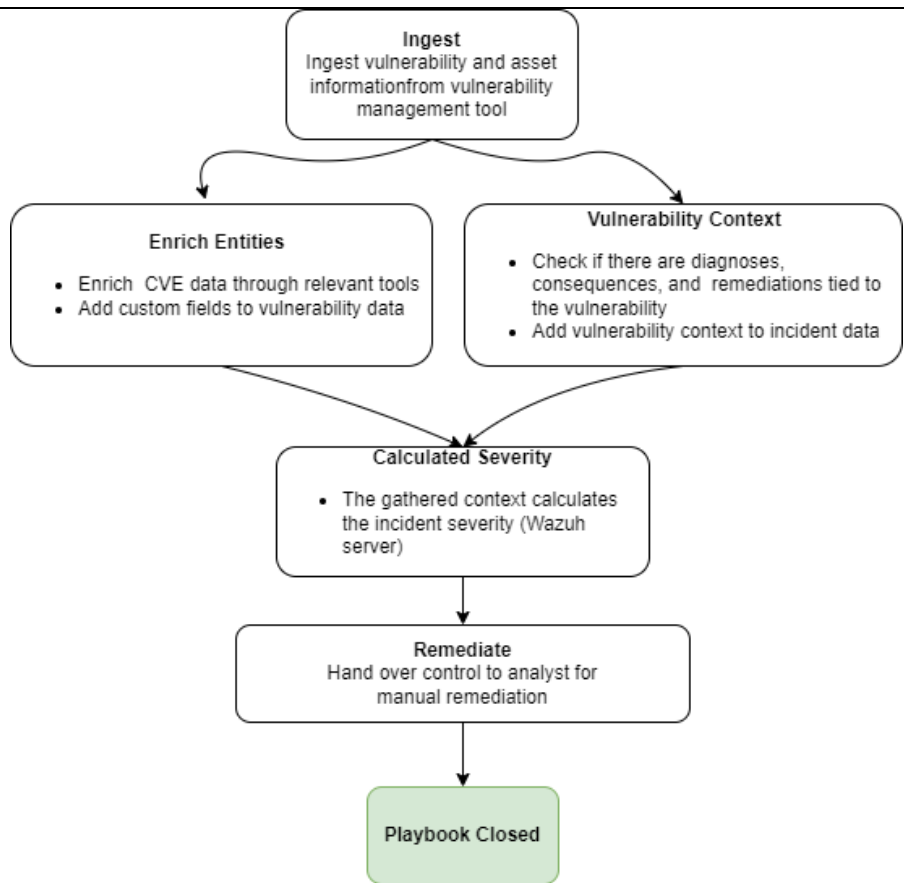
The user switches to the Windows VM that runs DIPS Communicator. The user identifies the service that takes care of downloading CSV files and placing them into a location where a DIPS Communicator picks them up. After that, the user shows the DIPS Communicator GUI and messages that have been sent to NOKLUS. In the end, the user checks NOKLUS Diabetes Registry and the stored data.

### 7.2.5 The Data scientist's view.

More generally, data scientists can also use the information transmitted to NOKLUS in summary statistics. If patients and clinicians need a broader view of the source data, data scientists need a data frame to perform posterior, ad-hoc data analysis.

The user opens the Jupyter notebook and fetch data from HL7 FHIR into the Vaex dataframe, which facilitates the data analysis. The PAF kicks in and applies the policy when the user is about to fetch data into Jupyter notebook.

### 7.3 Hygeia use case scenario/playbook

<b>Use Case Id</b>	SysAdminPHR(HYGEIA):001
<b>Use Case Name</b>	Vulnerability Management PHR backend Infrastructure Risks
<b>Use Case Summary</b>	<p>Vulnerability management is a strategically important process that covers both proactive and reactive aspects of security operations. Since vulnerability management encompasses all computing assets, security teams often grapple unsuccessfully with correlating data across environments, spending too much time unifying context and not enough time remediating the vulnerability.</p> <p>The HEIR integrated framework through HEIR vulnerability analysis module informs the Administrator for new emerging risks relating to operating system configurations</p>
<b>Actors</b>	PHR Backend System Administrator
<b>Preconditions</b>	None
<b>Trigger</b>	Outdated OS or urgent security patching to be applied
<b>Basic Flow</b>	 <pre> graph TD     Ingest[Ingest Ingest vulnerability and asset information from vulnerability management tool] --&gt; Enrich[Enrich Entities • Enrich CVE data through relevant tools • Add custom fields to vulnerability data]     Ingest --&gt; Context[Vulnerability Context • Check if there are diagnoses, consequences, and remediations tied to the vulnerability • Add vulnerability context to incident data]     Enrich --&gt; Severity[Calculated Severity • The gathered context calculates the incident severity (Wazuh server)]     Context --&gt; Severity     Severity --&gt; Remediate[Remediate Hand over control to analyst for manual remediation]     Remediate --&gt; Closed[Playbook Closed]   </pre>
	<p><b>Ingestion:</b></p> <p>The playbook ingests asset and vulnerability information from a vulnerability management tool.</p> <p><b>Enrich Entities:</b></p>

	<p>The playbook enriches endpoint and CVE data through relevant tools. It also adds custom fields to the incident if the newly gathered data requires them.</p> <p><b>Vulnerability Context:</b></p> <p>The playbook queries the vulnerability management tool for any diagnoses, consequences, and remediations tied to the vulnerability. If any vulnerability context is found, it's added to the incident data.</p> <p><b>Calculate Severity:</b></p> <p>Based on the gathered context, the HEIR calculates the severity of the incident.</p> <p><b>Remediate:</b></p> <p>The playbook now hands over control to the security analyst for manual investigation and remediation of the vulnerability.</p>
--	--

#### 7.4 CUH Use Case scenario/playbook

This use case aims to demonstrate the clinical application, Team 3 CTG device, in use, filmed and recorded as a podcast. The various components of HEIR would be sliced within the video, before ending with completion of the filming, namely the birth of an infant.

The various stages of the film demonstration of HEIR components are depicted below.

<b>Use Case Id</b>	CUH.01
<b>Use Case Name</b>	Medical application / device data interrogation.
<b>Use Case Summary</b>	Insights through the Anomaly Detection module based on a medical device (Huntleigh Team 3 CTG device).
<b>Actors</b>	Clinicians and End-Users of the medical application / device.
<b>Pre-conditions</b>	<ol style="list-style-type: none"> <li>1. Deployment of HEIR components to the (segmented) HEIR platform, which is a replica of a live clinical – SonicAid Centrale Labour/ Maternity - application.</li> <li>2. Calculation of Local RAMA score before any threat work.</li> </ol>
<b>Trigger</b>	The Team 3 device would be in use, RAMA score calculations would be seen at the user interface. Triggering change in RAMA score would be engineered by experimentation. This would also include triggering of true and false positive as well as true and false negative threat detection modules utilising working and non-working (infected) devices
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1. The user connects the medical device to the application;</li> <li>2. The medical device shows on the application UI;</li> <li>3. The user associates the device to a patient name, within the application UI;</li> <li>4. The user interacts with the medical device (ie. Initiates a scan)</li> <li>5. Data displays on the application GUI;</li> <li>6. Local RAMA Score calculator shows normal behaviour;</li> <li>7. Observatory shows normal behaviour</li> <li>8. The User initiates an abnormal activity on the medical device OR the medical device is 'infected' (via the USB port) to simulate a threat attack</li> <li>9. Application GUI shows an abnormal activity eg.no scan output;</li> <li>10. Local RAMA shows different score;</li> </ol>

	11. Observatory shows different (alert) output.
<b>Alternate Flows</b>	There are no alternative flows
<b>Post-conditions</b>	There are no post-conditions.

## 7.5 *Observatory use case scenario/playbook*

<b>Use Case Id</b>	Obsrv.01
<b>Use Case Name</b>	A short string indicating the meaning of the use case
<b>Use Case Summary</b>	<p>Insights through the Global RAMA score and statistical analysis regarding the cybersecurity status of the healthcare domain.</p> <p>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 (CVE).</p>
<b>Actors</b>	Security Analysts and Hospital Managers
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• Deployment of HEIR clients to at least two (2) hospitals.</li> <li>• Calculation of Local Rama score</li> </ul>
<b>Trigger</b>	The user accesses the Observatory dashboard (i.e., the 2 <sup>nd</sup> layer of visualizations)
<b>Basic Flow</b>	<ol style="list-style-type: none"> <li>1. The user logs in to the Observatory dashboard</li> <li>2. The user navigates to the available visualizations</li> <li>3. The user interacts with the available visualizations (e.g. timeline of RAMA score values)</li> <li>4. The user clicks one of the top vulnerabilities identified by the HEIR solution</li> <li>5. The user is redirected to the CVE webpage of the specific vulnerability</li> </ol>
<b>Alternate Flows</b>	There are no alternative flows
<b>Postconditions</b>	There are no postconditions.