Ref. Ares(2019)7921555 - 27/12/2019

Horizon 2020 Programme

H2020-SU-TDS-02-2018
Trusted digital solutions and Cybersecurity in Health and Care

ProTego

DATA-PROTECTION TOOLKIT REDUCING RISKS IN HOSPITALS AND CARE CENTERS

Project Nº 826284

ProTego

D7.2 *Report on the results of preliminary experiments and demonstrations*

Responsible:	GFI INFORMÁTICA
Contributors:	All partners with the leadership of Gfi.
Document Reference:	D7.2. Report on the results of preliminary experiments and demonstrations
Dissemination Level:	Public
Version:	Version 1.0
Date:	27/12/2019

Executive summary

This document describes the trial tests performed for all the technical components for Risk Assessment (WP4) and Risk Mitigation (WP5) in an isolated way, before starting to work on the integration of all the tools into a homogeneous platform.

The first objective for these tests is to confirm that the components can be deployed on environments other than those where they were developed and that they work properly wherever they are deployed. The second objective is to test the basic functionality that has been implemented so far.

Additionally, this document describes the initial plans for the first phase of the integration of all components, outlining the guidelines that will be followed towards accomplishing this.

Contributors Table

DOCUMENT SECTION	AUTHOR(S)	REVIEWER(S)
I. Introduction	Luis Carrascal	ALL
II. Infrastructures and Lab Environments	Luis Carrascal	ALL
III. Description of Demonstrations	Konstantinos Kouvaris, Luis Carrascal, Eliot Salant, Dave Singelee, Henrique Cesar Carvalho de Resende, Carlos Cilleruelo	ALL
IV. Results	Konstantinos Kouvaris, Luis Carrascal, Eliot Salant, Dave Singelee, Henrique Cesar Carvalho de Resende, Carlos Cilleruelo	ALL
V. Conclusion	Philip Usher, Luis Carrascal	ALL

Table of Contents

I. INTRODUCTION 10

I.1. PURPOSE AND ORGANIZATION OF THE DOCUMENT 10
I.2. SCOPE AND AUDIENCE 10

II. INFRASTRUCTURES AND LAB ENVIRONMENTS 11

II.1. DEFINITION OF THE ENVIRONMENTS 11
II.2. PRE-PRODUCTION ENVIRONMENT 11

III. DESCRIPTION OF DEMONSTRATIONS 13

III.1. SSM 13
III.2. SIEM 15
III.2.1. Elastic Stack 16
III.2.2. Wazuh 19
III.2.3. OpenVAS 20
III.2.4. Monitored Machines 20
III.3. DATA GATEWAY 21
III.3.1. Overview of the demo 21
III.3.2. Architectural components 22
III.4. ACCESS CONTROL FRAMEWORK 24
III.4.1. Architectural components 24
III.4.2. Key Management System 25
III.4.3. Access control 26
III.5. NETWORK SLICING 27
III.5.1. Performance Isolation 27
III.5.2. Privacy Isolation 29
III.6. CONTINUOUS AUTHENTICATION 30

IV. RESULTS 34

IV.1. SSM 34
IV.2. SIEM 39
IV.3. DATA GATEWAY 47
IV.4. ACCESS CONTROL FRAMEWORK 48
IV.5. NETWORK SLICING 49
IV.5.1. Performance Isolation 49
IV.5.2. Privacy Isolation 50
IV.6. CONTINUOUS AUTHENTICATION 51

V. CONCLUSION 54

VI. REFERENCES AND INTERNET LINKS 55

VI.1. INTERNET LINKS 55

Table of Figures

Figure II.2-1 PRE-Production Environment 12
Figure III.1-1 The high-level architecture of the Secure System Modeller. 13
Figure III.1-2 Main screen of Secure System Modeller 14
Figure III.1-3 E-learning course on risk analysis - SSM tool. 15
Figure III.2-1 Elastic Stack Architecture 16
Figure III.2-2 Elasticsearch Cluster Deployed 17
Figure III.2-3 Current Logstash Pipeline 18
Figure III.2-4 Future Logstash Evolution 18
Figure III.2-5 Future Kibana Cluster 19
Figure III.2-6 Future Wazuh cluster 20
Figure III.2-7 Alzheimer VM Architecture 21
Figure III.3-1 The data ingest path 22
Figure III.3-2 Analysing stored data 23
Figure III.3-3 Extending the FHIR Observation resource for meal portions 23
Figure III.3-4 Use FHIR client code to create an Observation resource 24
Figure III.4-1 High-level view on the data access architecture 25
Figure III.4-2 Access control components within ProTego 26
Figure III.5-1 Access Point and Processing Unit 28
Figure III.5-2 ProTego Application Scenario Simulation 29
Figure III.5-3 Privacy Isolation experimental setup 29
Figure III.5-4 Configuration of the three Open vSwitch bridges 30
Figure III.6-1 ProTego CA app 31
Figure III.6-2 ProTego CA permissions and service 32
Figure III.6-3 ProTego CA app user recognition 33
Figure IV.1-1 Snapshot of the initial risk model developed for the validation scenario of Marina Salud 34
Figure IV.1-2 Snapshot of the initial risk model developed for the validation scenario of San Raffaele 35
Figure IV.1-3 System model of the IT environment of Marina Salud 35
Figure IV.1-4 System model of the IT environment of San Raffaele 36
Figure IV.1-5 List of misbehaviours and threats for the validation scenario of MS where no security measures are considered. 37
Figure IV.1-6 List of misbehaviours and threats for the validation scenario of OSR where no security measures are considered. 38
Figure IV.1-7 List of misbehaviours and threats for the validation scenario of OSR where basic security measures are considered 39
Figure IV.2-1 Log events stored as Elasticsearch documents 39
Figure IV.2-2 Events generated for a Windows log in 40
Figure IV.2-3 Packets captured for 24 hours on Alzheimer virtual machine 40
Figure IV.2-4 IPv6 packets 41
Figure IV.2-5 Alzheimer virtual machine network settings 42
Figure IV.2-6 Host network settings 42
Figure IV.2-7 Why IPv6 packets are seen 43
Figure IV.2-8 Example of Packetbeat tuning 44
Figure IV.2-9 Nginx reverse proxy access log 44
Figure IV.2-10 Bind DNS service log 45
Figure IV.2-11 Example Dashboard 46
Figure IV.2-12 Example Report 46
Figure IV.2-13 Vulnerability scan XML report 47
Figure IV.3-1 Reported meal portions stored in Parquet files 47
Figure IV.3-2 Querying stored FHIR data 48
Figure IV.5-1 Performance isolation experiment with downlink UDP traffic through wireless link 49

Figure IV.5-2 Performance isolation experiment with downlink TCP traffic through wireless link 50
Figure IV.5-3 VXLAN tunnel throughput per VXLAN Network Identifier (VNI) 51
Figure IV.6-1 Three different user metrics obtained from keyboard usage 52
Figure IV.6-2 ProTego CA metrics 53

List of Tables

Table 1 Performance isolation experimental setup.....28

Table of Acronyms and Definitions

Acronym	Definition
AP	Access Point
API	Application Programming Interface
ARP	Address Resolution Protocol
CA	Continuous Authentication
COPS	Common Open Policy Service
CVE-ID	Common Vulnerabilities and Exposures Identification
DEK	Data Encryption Key
DNS	Domain Name System
EC	European Commission
EDR	Endpoint Detection and Response
FHIR	Fast Healthcare Interoperability Resources
HL7	Health Level 7
HTTP	Hyper Text Transfer Protocol
IAM	Identity and Access Management
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPSEC	Internet Protocol Security
IPv6	Internet Protocol Version 6
ISO	International Standardization Organization
JDBC	Java Data Base Connectivity
KEK	Key Encryption Key
KMS	Key Management System
KVM	Kernel-based Virtual Machine
LXC	LinuX Containers
Mx	Month x (of the project)
NAT	Network Address Translation

NUC	Next Unit of Computing
OvS	Open Virtual Switch
PC	Personal Computer
RAN	Radio Area Network
REST	REpresentational State Transfer
SIEM	Security Information and Event Management
SQL	Structured Query Language
SSM	Secure System Modeller
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNI	Virtual Network Identifier
VPN	Virtual Private Network
VXLAN	Virtual eXtensible Local Area Network
WP	Work Package
WTP	Wireless Termination Point
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

I. INTRODUCTION

I.1. Purpose and organization of the document

This is the second deliverable (D7.2) released by ProTego WP7 and provides an overview of the ongoing trials and the planning for upcoming trials. It consists of the results of the activities performed in this work package up to M12.

The deliverable consists of three major parts: the first section describes the Infrastructures and Lab Environments used for the trials. The second section describes the trials themselves that have been performed so far. The third section describes the most relevant results of the trials described in the previous section.

I.2. Scope and audience

This deliverable is considered as a public document to the ProTego project. It provides a description of the status quo of ongoing trials and the planning for the next trial phase.

This deliverable is primarily intended for EC Officers and project-appointed reviewers. Furthermore, this document is relevant for project members involved in the scenario modelling (mainly WP2).

II. INFRASTRUCTURES AND LAB ENVIRONMENTS

II.1. Definition of the environments

Throughout the course of the project, several environments will be used for the software components. The naming convention for the environments is the following:

- **Development**: this is the environment that each partner has on their own premises and is used to develop and perform the first tests of the different components.
- **PRE-Production**: this is an environment provided by GFI that all partners can use as a common testing platform where components can be tested in an isolated way or integrated.
- **Production**: these are the environments that hospitals, both San Raffaele and Marina Salud, will provide as testbeds for the scenarios defined. The objective for this environment is to deploy components as Docker containers, using Kubernetes as orchestration system, and the Rancher management platform.

As of the release of this document, not all software components are prepared to work as Docker containers. Therefore, the PRE-Production environment is used to accomplish this task. There are two exceptions:

- **SSM**: the offline risk assessment tool provided by IT Innovation is offered as a service and therefore will not be deployed at any environment other than those supplied by them.
- **Network slicing**: this component involves the use of specific hardware. Therefore, so far, it has only been tried at IMEC premises. Once the hospitals acquire the necessary equipment it can be tried in other environments as well.

II.2. PRE-Production environment

This environment is hosted on two physical servers located at one of the Data Centres in France [1] managed by OVH [2], an Internet Services Provider.

To access the environment, it is mandatory to connect to a SoftEther VPN, that is hosted on those servers. Other services are available, such as a GitLab. There is a central authentication system that is used by all services and servers, whether physical or virtual.

For testing purposes there is a virtual machine called partners, where all components can be tried either in an isolated way or integrated in future phases of the demonstration work.

The following is a diagram that represents the environment:

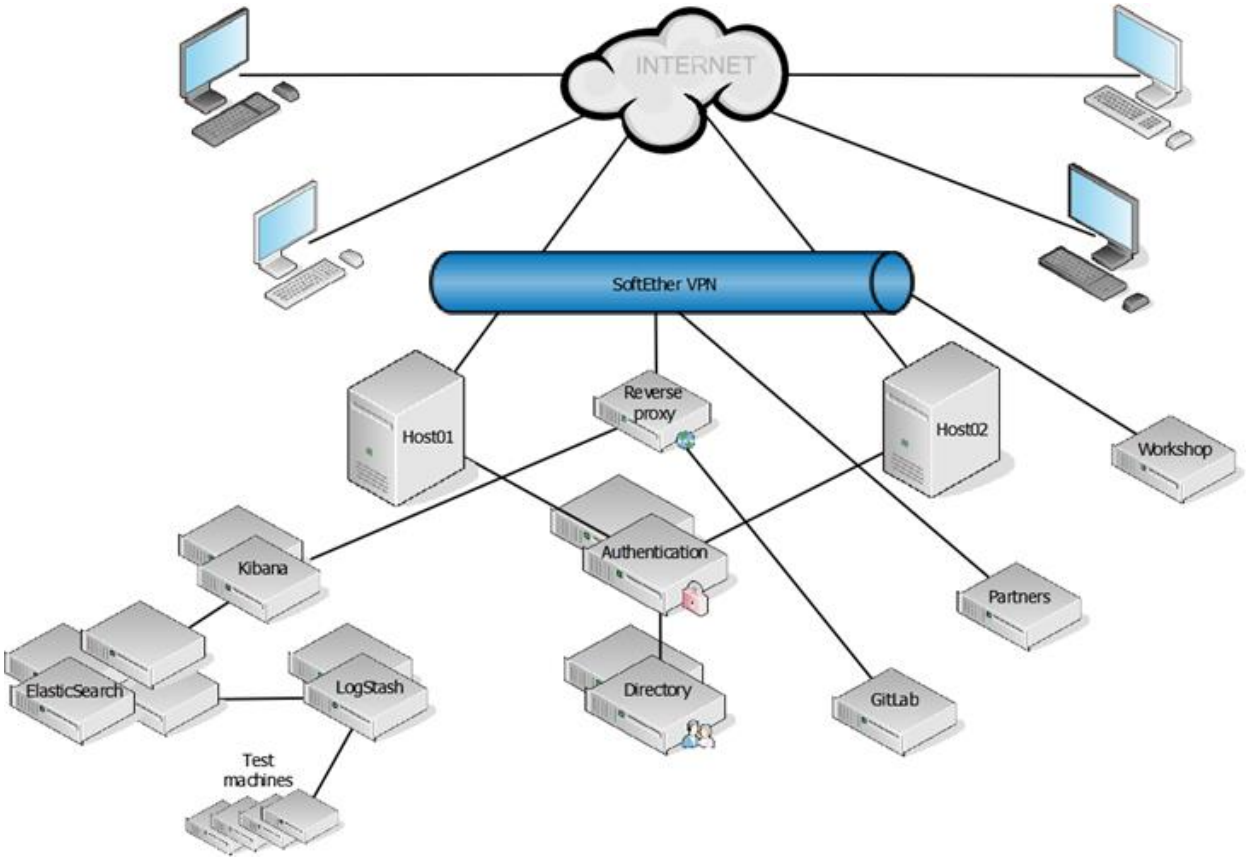

Figure II.2-1 PRE-Production Environment

III. DESCRIPTION OF DEMONSTRATIONS

III.1. SSM

The Secure System Modeller (SSM) is a graphical tool for performing cybersecurity risk assessment analysis at design time. This allows a system designer or analyst to identify security requirements for their system using the procedures defined by ISO 27005. The SSM tool models the assets of the IT environment along with their relations following a graph-based approach. A cybersecurity knowledge base of model inference and threat identification rules is used to generate a catalogue of potential threats and the corresponding risk levels.

Figure III.1-1 gives an overview of the architecture of SSM. As shown, the UI of SSM is a web-based user interface where the services are accessed through a REST API. These API requests are then passed to the underlying model querier and model validator. The users of SSM are presented with a catalogue of assets and update methods that allow them to edit their system model to better match their live system and the impact threats can have to their business. The system models are passed to the persistent layering by being stored in separate graphs. This allows the user to easily import and export the models.

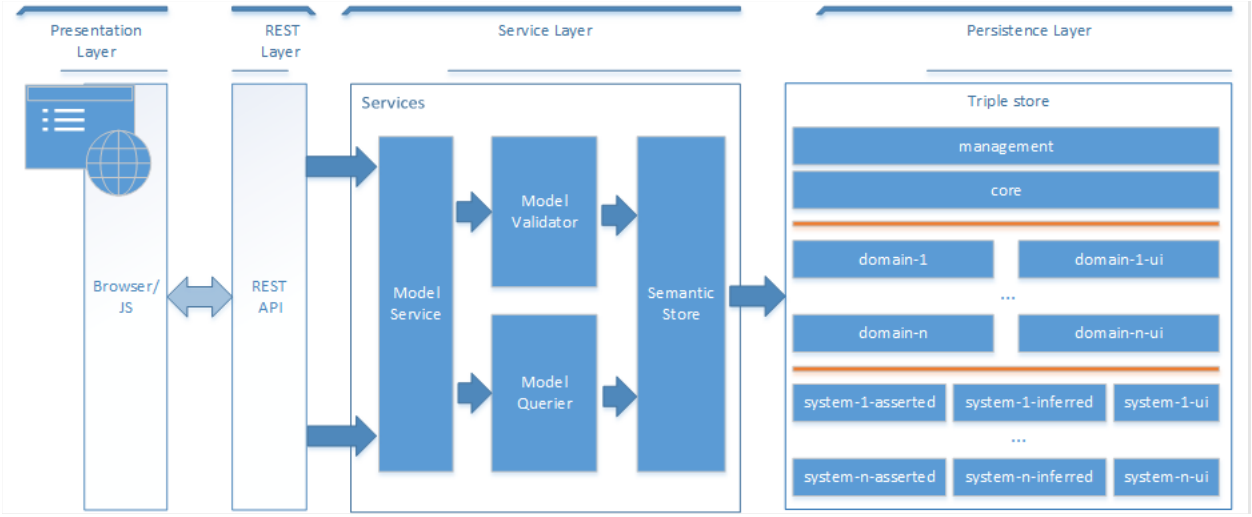

Figure III.1-1 The high-level architecture of the Secure System Modeller.

Our aim here is to provide and equip the end-users from Ospedale San Raffaele SRL (OSR) and Marina Salud (MS) hospitals with the right tool for and enable them perform ISO 27005 cybersecurity risk analysis. The objective of this demonstration is twofold; to show the availability and accessibility of the SSM tool by OSR and MS, and the use of SSM for cybersecurity risk assessment at design time.

In this demonstration scenario, the SSM service for the ProTego project was setup to be accessible through a web-browser (preferably Chrome) at the following address: https://protego.it-innovation.soton.ac.uk/system-modeller/ (see Figure III.1-2). The server is hosted by an SSM VM.

Initially, the users must create an account with SSM. Once they are registered and have their account activated, they can start designing their IT system using SSM and analyse their risk levels. The domain model that is currently deployed by SSM includes all the latest ProTego specific extensions covering the modelling of systems with Internet of Things (IoT) devices (for more information, see D4.1). The ProTego-specific domain model supports modelling system's stakeholders, such as clinicians and patients, that interact directly or indirectly with the system.

Performing an ISO27005 risk assessment analysis procedure using SSM typically involves the following five steps:

1. identify and add assets of the system to the model by giving them meaningful labels;
2. insert links between assets of the system using the relationship types as provided by the domain model deployed;
3. validate the model for consistency (during that step implicitly defined assets and relations are inferred);
4. estimate and set the levels of trustworthiness for each asset and the impact level misbehaviours can have to the business (i.e., very low to very high);
5. calculate risk levels to determine which threats pose the highest risk.

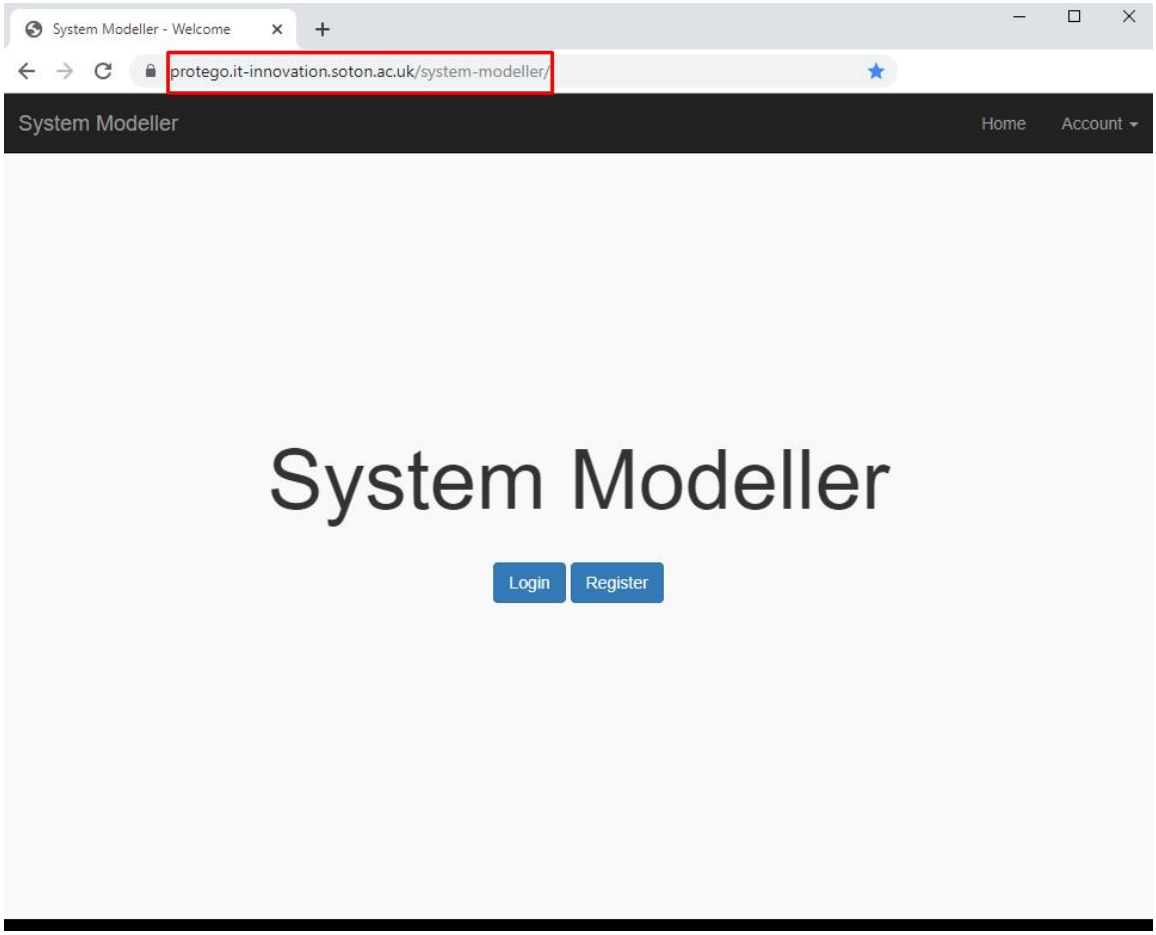

Figure III.1-2 Main screen of Secure System Modeller

To support the validation partners with using the SSM tool for ISO 27005 analysis, we carried out a series of tutorials on SSM risk analysis via web conferencing. For the purpose of the tutorial, an exemplary model of the IT environment of an online shop has been used to serve as a testbed for risk analysis. The idea for the online tutorial sessions was for the end-users to better understand how to use SSM to build and analyse their own validation scenario models based on their IT infrastructure and familiarise themselves with the steps of risk analysis (as they are outlined above). Material from the online tutorial has been upload onto the ProTego SharePoint, Documents > Work Packages > WP4 > SSM Tutorial 2019-10. Following the tutorial on SSM, we offered them access the access to the e-learning platform designed for SSM to further explore and learn from at their own pace (see Figure III.1-3). The online course can be found here: https://learnssm.it-innovation.soton.ac.uk/.



Figure III.1-3 E-learning course on risk analysis - SSM tool.

III.2. SIEM

The SIEM is the run-time tool for security monitoring that allows log management, long-term storage and analysis of log data, real-time monitoring and notification of events, and other additional functionality to enhance the detection of illegitimate use of the monitored Information Systems.

To build the SIEM three major Open Source components are used:

- The Elastic Stack [3]: a supporting platform used as central repository of information, provides the means to ingest data and transform it as desired, as well as a very flexible interface to query data, and offers many integration possibilities. It is becoming the de facto standard to store security information. For future integration purposes two other Open Source projects are supported:
    - Apache Kafka [4]: a robust data bus. It will be used to integrate the SIEM with other toolkit components.
    - Apache Spark [5]: an Analytics Engine for Big Data. It will be used to perform machine learning techniques.
- Wazuh [6]: a fork of the OSSEC project [7] that provides an engine that supports the definition of Decoders and Rules. In addition, it provides other features that can be used such an inventory of the monitored machines, a set of predefined decoders and rules, and more. The information generated by Wazuh will be stored on Elasticsearch.
- OpenVAS [8]: an active vulnerability scanner. Information about the discovered vulnerabilities will be stored on Elasticsearch and passed to the SSM to recalculate the risks.

III.2.1. Elastic Stack

The objective for the SIEM is to make it a security analytics platform that can be used for machine learning purposes. To accomplish that the Elastic Stack is used as the supporting platform. The main components of the Elastic Stack are represented in the following diagram:

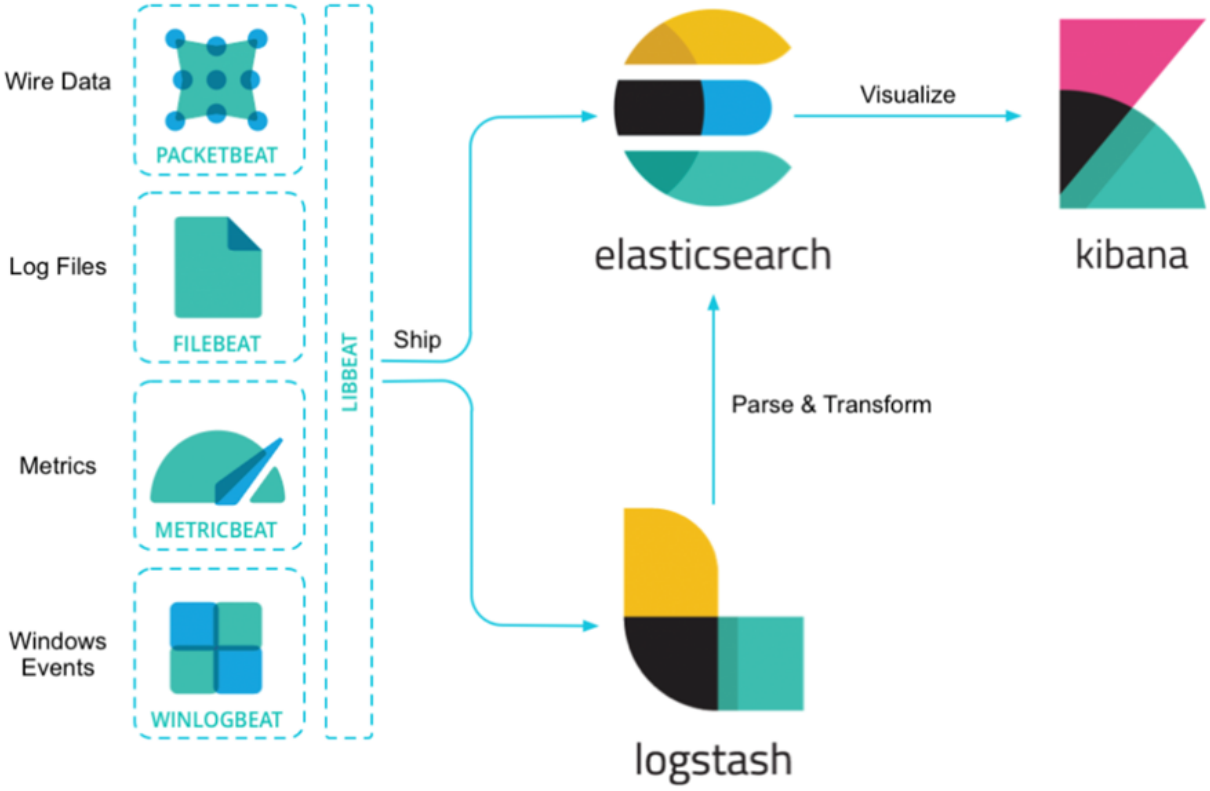

Figure III.2-1 Elastic Stack Architecture

For Elasticsearch, the deployment that has been performed on the PRE-Production environment, resembles what is necessary on a production system as a bare minimum. All the nodes have been deployed using Docker containers:

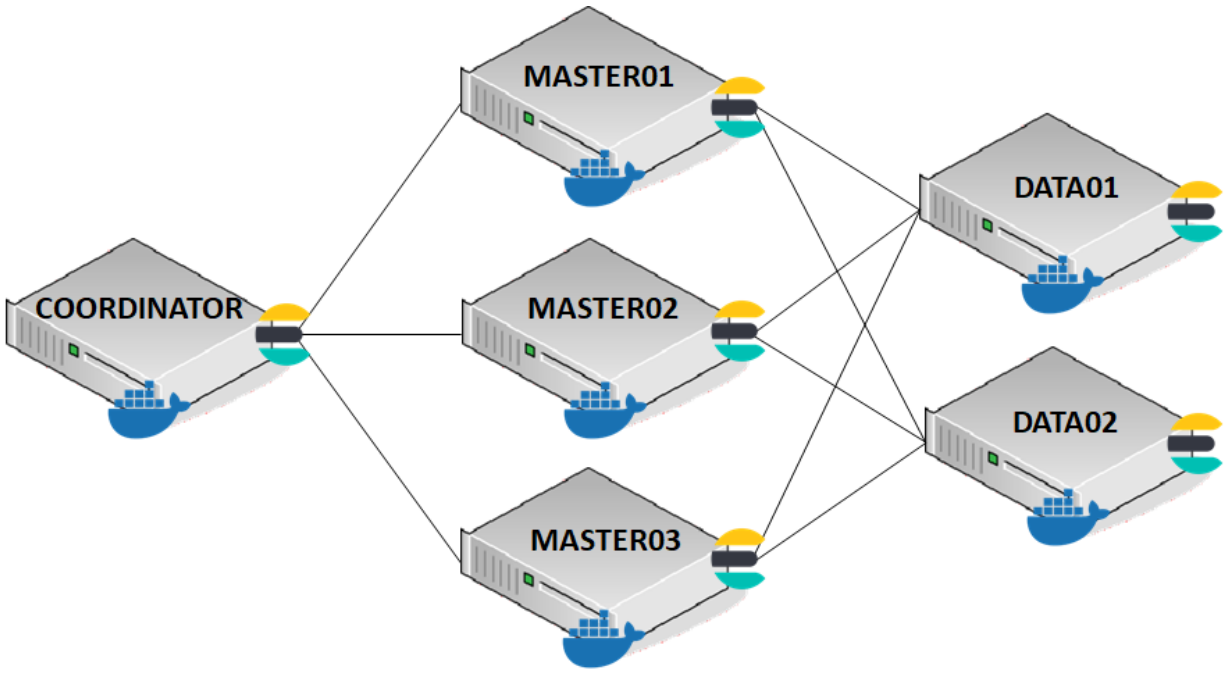

Figure III.2-2 Elasticsearch Cluster Deployed

Initially just one Logstash node has been deployed, also as Docker container, with a single pipeline to process all data:



Figure III.2-3 Current Logstash Pipeline

For future tests there will be different pipelines and different nodes to balance the load as the data ingested increases:

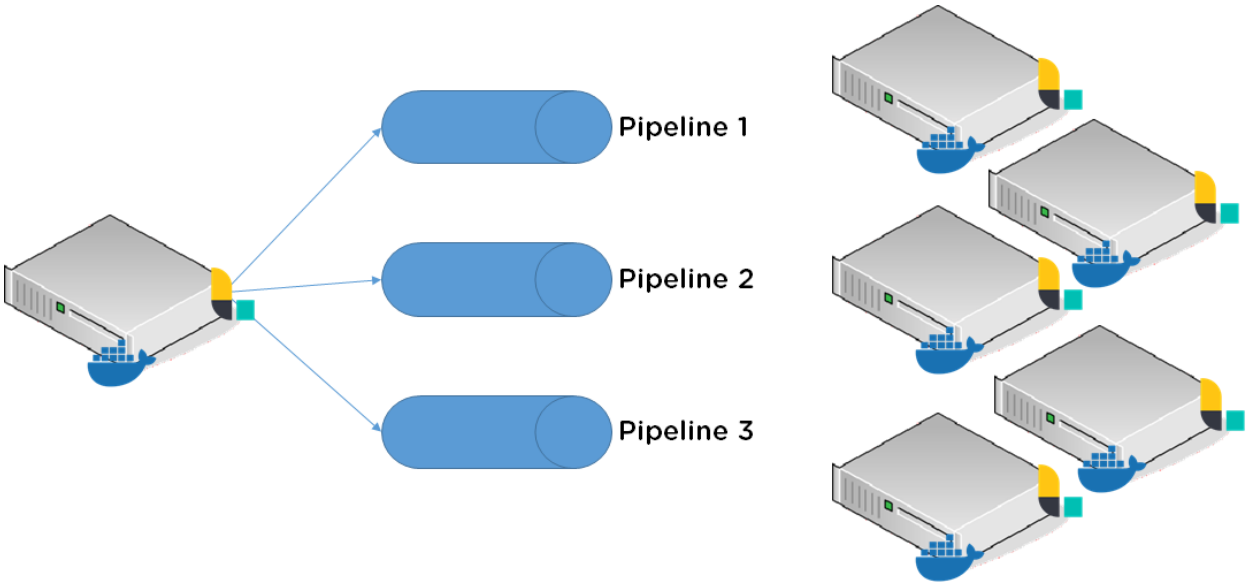

Figure III.2-4 Future Logstash Evolution

In the same manner, for Kibana also just one node has been deployed as a Docker container. Even though it is unlikely that more are necessary, if in the future the load raises to levels that require more processing capacity they can be easily balanced:

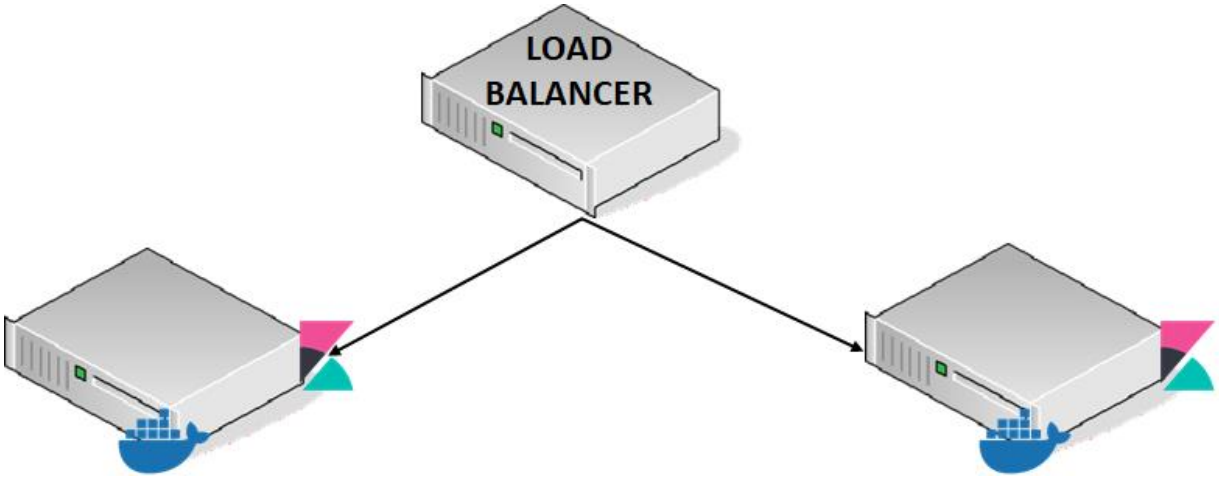

Figure III.2-5 Future Kibana Cluster

Finally, the beats to ingest data have been deployed on the monitored machines. Several beats have been tested:

- Filebeat: to get events from log files.
- Winlogbeat: to get events from Windows logs.
- Packetbeat: to get packet from network traffic.

III.2.2. Wazuh

The main component for Wazuh is called Wazuh Server or Wazuh Manager. It has been deployed as a single Docker container. Nevertheless, Wazuh can be deployed as a cluster to scale horizontally when there are hundreds, or thousands of machines monitored.

There are two types of cluster nodes:

- Master: just one node can have this role and controls the whole cluster.
- Worker: provide the Wazuh Manager functionality. The master node can be used as a worker.

To distribute the workload across cluster nodes, it is necessary to use a load balancer, as shown in the following diagram:

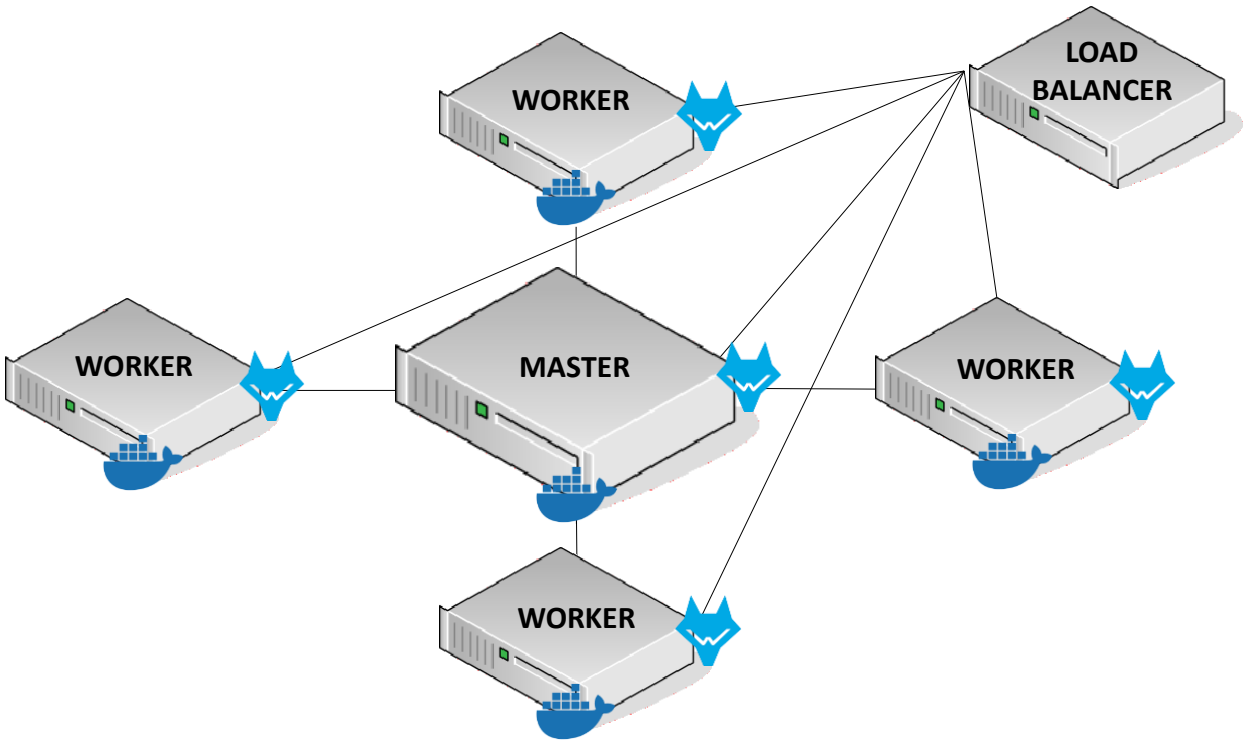

Figure III.2-6 Future Wazuh cluster

The Wazuh agent has been installed on the monitored machines.

III.2.3. OpenVAS

Although Wazuh has certain functionality related to vulnerability awareness, at the time this document is written, it is still very limited, it is just available for a few Linux distributions (Debian, Red Hat, and Ubuntu), and important systems such as Windows are not included.

Therefore, the OpenVAS project is being analysed to be integrated in the final architecture. Nevertheless, just basic trials have been performed with this component. Manual scans have been launched against the monitored machines and the results analysed but features for the integration such as running as Docker containers, automation, APIs, etc, have not been studied yet.

Because of the very nature and functionality of this component, it has been deployed on a separate virtual machine until it was thoroughly understood how it could affect the rest of the toolkit.

III.2.4. Monitored Machines

To have a variety of data sources for the SIEM, several machines are monitored:

- The physical hosts of the PRE-Production environment: These are called host01 and host02. Both hosts have Proxmox VE as a virtualization platform, which is built on Debian Linux and provides the following main services:
    - KVM virtualization
    - LXC containers
    - Docker containers
    - DNS service with Bind.

  The monitored logs are *syslog* and *auth* using Filebeat:
    - Syslog is a standard logging facility. It collects messages of various programs and services including the kernel.

- Auth is the log that collects information about authorizations performed by the system, for operations such as login, ssh, su, sudo, shutdown, etc. It includes both successful and failed attempts.

- The Partners virtual machine: This is the server that is used to deploy and test components and future integration trials. The system is Ubuntu 18.04 LTS server. The monitored logs are *syslog* and *auth* using Filebeat.
- The Alzheimer virtual machine: To resemble an Information System, similar to those available in a hospital, that could be monitored by the SIEM in the PRE-Production environment, GFI Health Business Unit has supplied a virtual machine cloned from a Production environment:

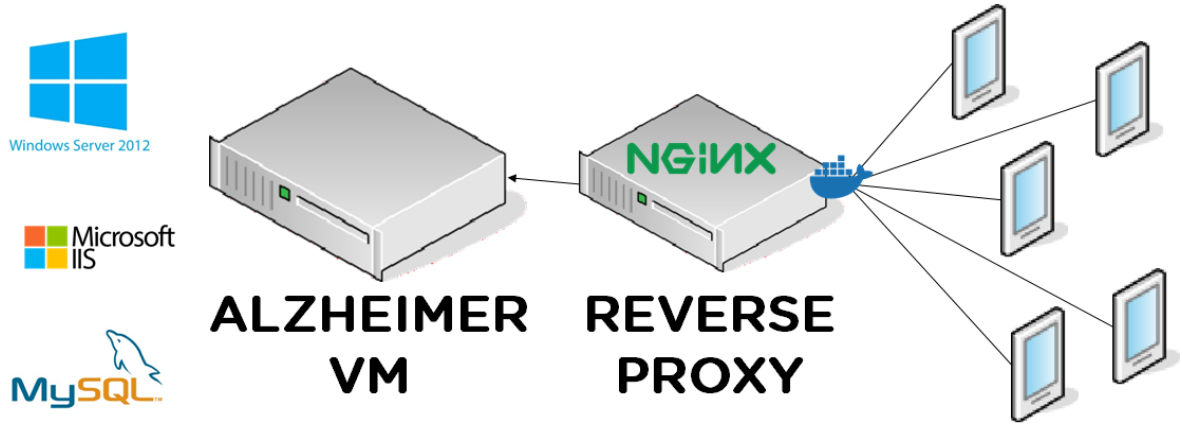

Figure III.2-7 Alzheimer VM Architecture

This virtual machine uses Windows Server, Internet Information Server, and MySQL. A reverse proxy with Nginx in a Docker container has been placed in front of the service offered by Internet Information Server and is published on the Internet. A few client devices access this service. Both Winlogbeat and Packetbeat are used to monitor this service, as well as the access log of the Nginx reverse proxy.

III.3. DATA GATEWAY

HL7 FHIR is an evolving standard describing data formats and healthcare-related resources, along with an application programming interface for exchanging electronic health records. FHIR is driven by healthcare IT leaders such as Cerner, and widely gaining traction with pledged support from leading technology providers (Amazon, Google, Microsoft, Salesforce, IBM, and Oracle) jointly issuing a statement of collaboration for healthcare interoperability (with a focus on cloud and AI). (Bresnick, 2018). The FHIR standard has been selected as the main format for data exchange and storage in ProTego project.

The Data Gateway is the IBM ProTego component that receives and stores the FHIR data. It also provides access to the stored FHIR data - via fetching individual records, or via running analytic queries on that data, using SQL.

IBM developed a Year One demo which not only demonstrates and tests the implementation of the Data Gateway but can also be used as a reference implementation by the use case partners (active in WP2). In this report, we demonstrate the Data Gateway using the concept of OSR's Food Coach recommendation system as an inspiration.

III.3.1. Overview of the demo

This demo is a proof-of-concept for how Big Data from the world of healthcare can be securely stored and accessed, harnessing the power of Spark analytics. We show how we can extend the

functionality of a commercial FHIR server to stay compatible with the standardized FHIR messaging format.

To illustrate an end-to-end use of the Data Gateway, we have put together a simplified version of what is envisioned to be an actual healthcare application that the project will be producing, namely Food Coach. The goal of the Food Coach application is to allow participants to record their daily food intake, and then have this information centrally and securely stored. Dieticians will then be able to access the stored information for a participant and perform queries and analytics on the data in order to provide personalized, dietary recommendations.

In this demo, we generate simulated meal portions for a patient over a period of time and use a FHIR client to securely transmit this information to a FHIR server enriched by the ProTego Data Gateway. The Data Gateway will store this information as Parquet files, and we have created a basic SQL interface to the Gateway which illustrates how the stored information could then be accessed and analysed by a dietician.

III.3.2. Architectural components

The Data Gateway is built around IBM's Open Source implementation of a FHIR server, freely available from: https://ibm.github.io/FHIR, and IBM's extension to Apache Parquet standard to support modular encryption in the file format. IBM's FHIR server provides a method to implement the interception and processing of FHIR messages sent to the FHIR server, which we use to redirect store requests of FHIR objects from a JDBC-based database backend, to a Parquet backend. A conceptual view of the data ingest path is shown in Figure III.3-1.

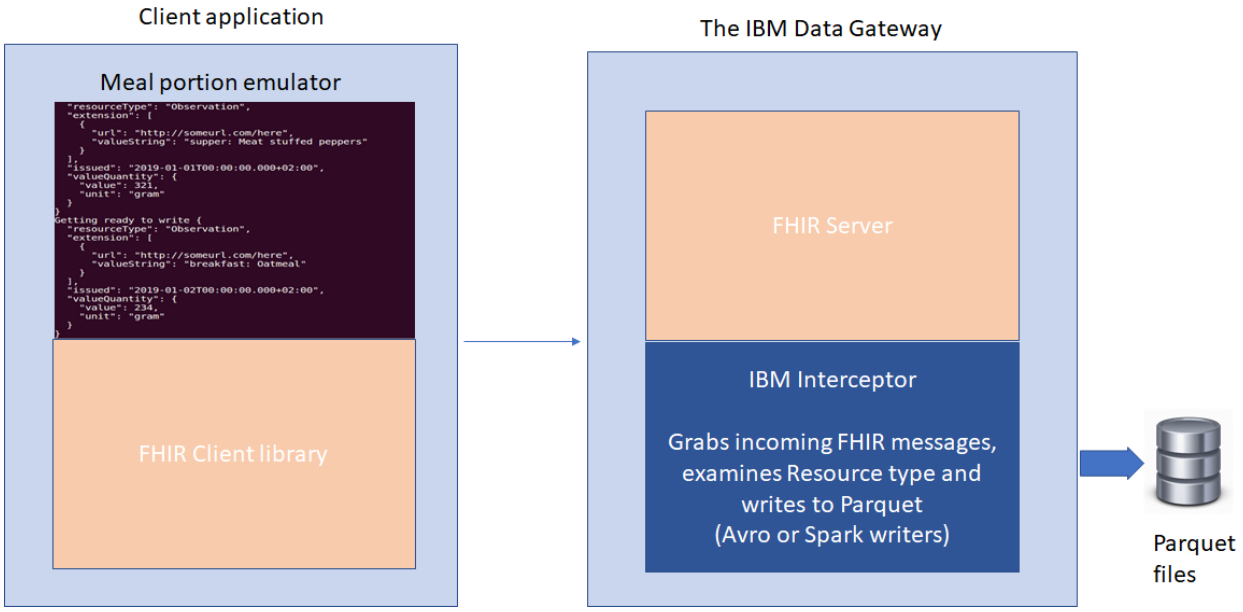

Figure III.3-1 The data ingest path

SQL queries for analytics of the stored data use a REST endpoint which directly invokes Spark SQL on the stored data files, without having to go through the FHIR server, as shown in Figure III.3-2.

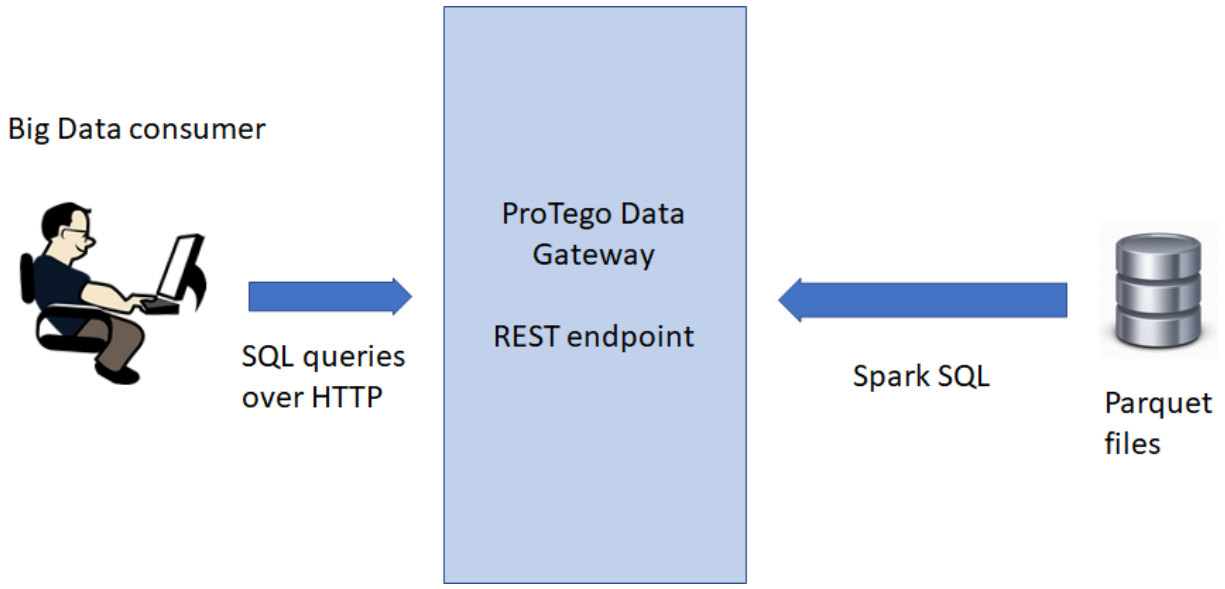

Figure III.3-2 Analysing stored data

Meal portions are represented in FHIR format by extending the standard Observation resource. As shown in Figure III.3-3, the FHIR *extension* attribute is used to describe what was consumed (e.g. “hamburger”), and for which meal (e.g. “lunch”). The standard Observation *valueQuantity* is used to show how much was consumed (e.g. 223 grams).

```
{
"resourceType": "Observation",
"id": "23",
"subject": "009",
"extension": [
      {
      "url": "http://http://protego.eu/mealPortion",
      "valueString" : "lunch: Hamburger"
      }
],
"valueQuantity":{
      "value":223,
      "unit": "gram"
      },
"issued":"2019-01-03T12:00+02:00"
}
```

Figure III.3-3 Extending the FHIR Observation resource for meal portions

Simulations of the consumed portions can be generated by a direct REST POST command. However, to more easily generate large amounts of data, simulating consumption over an extended period of time, IBM created a meal portion generator which uses Java-client code to send the resources directly to the FHIR server (Figure III.3-4).

```
Observation o = Observation.builder()
                    .status(ObservationStatus.FINAL)
                    .id(Id.builder().value(observationID).build()) // identifier for observation
                    .subject(Reference.builder().identifier(Identifier.builder().value(string(patID)).build()).build())
                    .code(CodeableConcept.builder()
                    .coding(Coding.builder()
                                .system(Uri.of("http://snomed.info/sct"))
                                .code(Code.of("226379006"))
                                .display(string("ProTego demo"))
                                .build()).build())
                    .value(Quantity.builder()
                                .value(Decimal.of(meal.generateAmountEaten()))
                                .system(Uri.of("http://unitsofmeasure.org"))
                                .unit(string("gram"))
                    .build())
                    .extension(Extension.builder()
                               .url("http://http://protego.eu/mealPortion")
                               .value(string(meal.generateMeal(mealLoop)))
                    .build())
                    .issued(Instant.builder().value(meal.generalTimeEaten(dayLoop, monthLoop,"UTC+2")).build())
                    .build();
```

Figure III.3-4 Use FHIR client code to create an Observation resource

III.4. ACCESS CONTROL FRAMEWORK

One of the central components in the ProTego architecture is the Data Gateway. Its main functionality is to store and retrieve FHIR data, and to provide access to the FHIR data which is stored externally and encrypted via Parquet Modular Encryption. Due to this encryption, one is forced to use the Data Gateway to store or retrieve FHIR data. Bypassing the Data Gateway to fetch the external data would mean that this data would not get decrypted and therefore would be unusable. However, this architectural design has two consequences:

- Data encryption requires the use of cryptographic keys. These keys need to be stored securely or be computed on the fly (via a secure key derivation function).
- Multiple users can set up a secure connection, via dedicated applications such as Food Coach, to the Data Gateway to store or retrieve FHIR data. However, each of these users should not necessarily get access to all FHIR data, there are typically *security policies* in place. For example, one could enforce that patients only get to retrieve their own data, or that physicians would only get access to the data of their own patients. Therefore, additional functionality should be in place to decide when the Data Gateway should give access to the fetched FHIR data (or likewise store the data externally).

Both these items are tackled in the access control framework developed in ProTego. The discussion above also shows that both functionalities – key management and data access control – are tightly coupled to the Data Gateway and the Parquet Modular Encryption. In other words, in the context of ProTego, both key management and data access control do not have any added value when used in isolation, they only make sense when used in combination with the actual data storage and retrieval (i.e. the Data Gateway). Therefore, most of the work has focused on the access control architecture and how it should be integrated with the Data Gateway.

III.4.1. Architectural components

A simplified and high-level view of the access control architecture is shown in Figure III- 15-. Note that this figure omits many components in the ProTego architecture and only highlights the components that are relevant for the discussion below. To further simplify the discussion, we will focus only on data retrieval. However, note that storing data works similarly.

Before the application can start interacting with the Data Gateway, one first needs to know which user is actually using the application (i.e. which user wants to retrieve data). This functionality is handled by the *Identity and Access Management (IAM)* system. This component manages all the

users in the system, and knows the different users and their roles. This party acts as a trusted authority and issues signed *authorization credentials*. These are basically assertions on the users identity and its role(s). After the user is successfully authenticated, it will receive an authorization token which it can hand over the Data Gateway.

The Data Gateway will forward this authorization credential to the *access control component*. This component will decide whether the Data Gateway should grant access to the requested data or not. This decision will be based on the actual data request (i.e. which data one wants to access), the content of the authorization credential and on the relevant security policies in place.

To retrieve the data that is stored externally, the Data Gateway needs to decrypt the data first. Therefore, it will contact the *key management system (KMS)* to obtain the required decryption key. The main functionality of the KMS is to securely store all encryption keys externally, i.e. these keys are not stored directly in the Data Gateway itself.

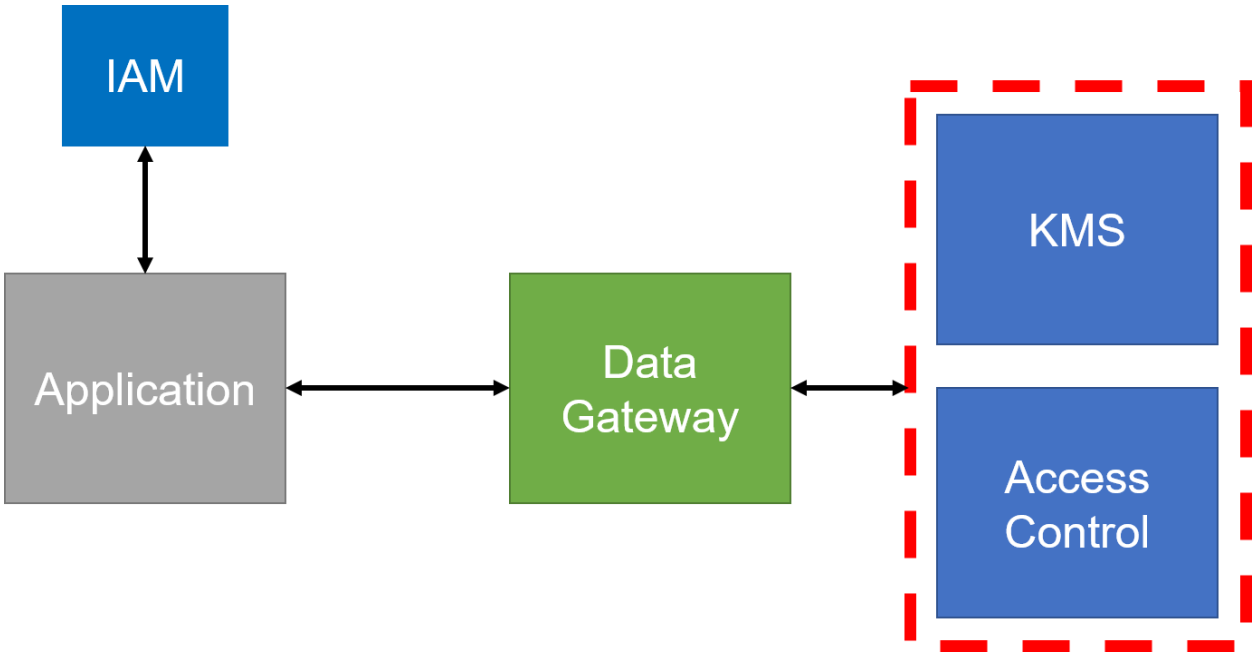

Figure III.4-1 High-level view on the data access architecture

These three components combined – IAM, KMS and access control – constitute the access control framework. However, the work of ProTego mainly focuses only on the design and development of the KMS and the access control components, as depicted in Figure III- 15 -with the dotted red line. ProTego will not focus on the development of an IAM component. However, to realize the integrated architecture, some IAM functionality will be needed. Therefore, in the integrated ProTego demo that will be implemented in M18, commercial off-the-shelf components will be used for the IAM building block, for example Keycloack (https://www.keycloak.org). However, the selection of this component is out of the scope of this deliverable.

III.4.2. Key Management System

In the first trial of the ProTego integrated demo, to be implemented by M18, an open source component will be used to manage the storage and retrieval of the encryption keys. We opted to use Hashicorp Vault (https://www.vaultproject.io/) to protect the application secrets (i.e. the encryption keys used by the Data Gateway). We assume that a mutual TLS session has been set up between the Data Gateway and Vault, such that both components can communicate securely.

The main functionality that Vault offers, is "encryption as a service". This is done via envelope encryption. The basic idea is the following. To encrypt FHIR data, the Data Gateway will generate a unique ‘data encryption key’ (DEK). This key is then sent to Vault over the secure TLS channel. Vault will encrypt this DEK with its internal ‘key encryption key’ (KEK). The result is sent back over

the secure TLS channel to the Data Gateway, which then stores this secure envelope (containing the encapsulated DEK) together with the encrypted FHIR data. If the Data Gateway later wants to retrieve encrypted data, a similar procedure is used. It will retrieve the secure envelope and send it to Vault. Upon receiving this encapsulated DEK, Vault will use its internal KEK to decrypt the envelope and retrieve the DEK. Next, it will send the DEK securely over the TLS channel back to the Data Gateway.

For the first phase of the ProTego integrated demo, we only need this key management functionality (i.e. secure storage of keys), which is offered by Vault. Therefore, for this particular component, the work within ProTego is immediately targeted towards the integration into the overall architecture by M18, and not on the development/use of the individual KMS component (i.e. an isolated YearOne demo).

However, it is good to point out that in the second half of the project, the key management functionality will be further enhanced and improved, for example by deriving keys from the attributes of the user. At that stage in the project, we will need to evaluate whether the use of Vault would still make sense, or has to be (partially) replaced by other components that will have to be developed within ProTego. However, since this discussion relates to future research and development plans in ProTego, it is out of the scope of this deliverable.

III.4.3. Access control

The access control building block will decide whether access requests to data should be granted or not. However, this is a complex process, and is typically realized by combining different functionalities. To increase the flexibility and adaptability of the system, each of these access control functionalities need to be realized by different functions (i.e. building blocks), as is shown in Figure III.4-2

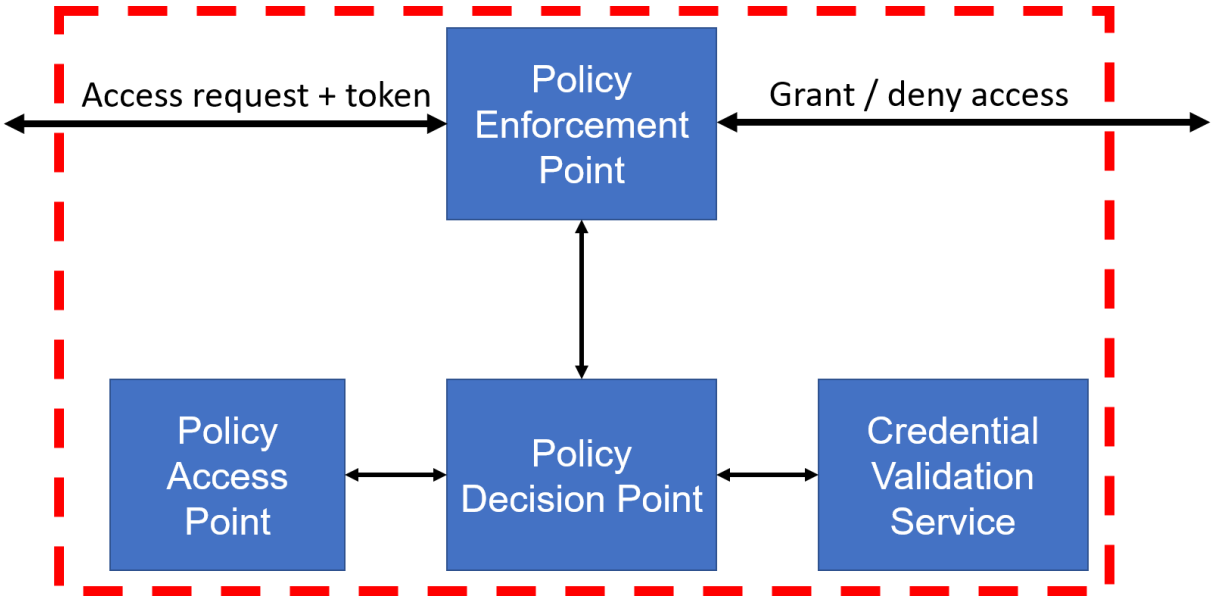

Figure III.4-2 Access control components within ProTego

The system works as follows. The *policy enforcement point* receives the data access request information and the authorization token that was issued by the IAM. It will forward this to the *policy decision point*, which will retrieve and evaluate the relevant security permissions. The security permissions are specified in XACML, a widely used standard within the domain of access control. These security permissions are stored, created and managed in the *policy access point*. However, to evaluate permissions and decide whether access should be granted, the authorization token first needs to be validated. This is done by the *credential validation service*. The credential validation service needs to contain a list of trusted credential issuers. In our case, this will be the Identity and Access Management (IAM) system in place. After the authorization token is validated

and the security permissions are retrieved, the policy decision point can now answer the following question “given the security permissions, and this subject with the following valid role(s), does it have the right to perform this operation (retrieve/store data)”. The answer to this question, which is basically a Boolean Yes or No, is then forwarded back to the policy enforcement point, which then enforces this decision.

The advantage of this distributed access control architecture is that each component has a clear distinct function, and can be managed independently of the other components. For example, when new security permissions need to be used, only the policy access point needs to be updated (i.e. the security permissions that are stored in this component).

Although the distributed access control functionality makes most sense in combination with the actual data retrieval and storage (i.e. the Data Gateway), a stand-alone YearOne demo has been developed. Within this demo, server software has been developed that simulates the integrated distributed access control architecture encompassing the components shown in Figure III.4-2, as well as simplified version of the IAM itself. This server software can be deployed on a Linux machine running Ubuntu or Debian. The client can then log in to this server using its browser. Depending on the type of credentials provided to the server, different functionality can be shown:

- **Administrator credentials:** manage the identities and roles of the users, specify or change security permissions, etc.
- **User credentials:** simulate a patient or clinician that requests data access

III.5. NETWORK SLICING

Network slicing in the ProTego context is the logical isolation network flow per application to provide an extra security layer for ProTego applications and prioritize the traffic in the radio link. With these characteristics, hospital applications, such as critical patient health data, can be stored in the hospital server in a secure and guaranteed manner. In order to provide performance isolation in the radio link and the extra security layer in the client-server communication, it is necessary to integrate two different technologies. For performance isolation, the WiFi slicing software called 5G-Empower is required to prioritize the ProTego application data. 5G-Empower can prioritize downlink network traffic to specific applications, guaranteeing different performance among the applications accessing the same access point. For the extra security layer, the usage of VxLAN was proposed, which can create a virtual network and apply different VLAN ids to application flows which will help to isolate the network traffic. Moreover, VxLAN supports IPSEC, which will add an extra cryptographic layer.

The preliminary experiments for network slicing have as its objective to validate the behaviour and configuration of the performance and privacy isolation solutions so that it can be further integrated into the ProTego environment. For the 5G-EmPower, we assessed the behaviour of the software with UDP and TCP protocols to check how much of the performance the ProTego applications will avail from the performance isolation given the protocol utilization. Furthermore, the privacy isolation using VxLAN is evaluated in terms of deployment time of a slice and performance degradation.

III.5.1. Performance Isolation

For the performance isolation, we deployed 5G-EmPOWER components in a local testbed environment. As can be seen in **Error! Reference source not found.** and Figure III.5-1, the local testbed is composed of a PC Engines APU2D4, a D-Link DGS-1005D 5-Port Gigabit Switch, and Intel Baby Canyon NUC7i5BNK. For the client role, we utilized a MacBook. iPerf clients were used in the MacBook connected to the iPerf servers located in the processing host to generate the traffic in the wireless link.

Table 1 Performance isolation experimental setup

Processing host	Hardware	Intel Baby Canyon NUC7i5BNK
	Operating System	Ubuntu 19.04
Access Point	Hardware	PC Engines APU2D4
	Operating System	Lede 17.01
Client	Hardware	MacBook
	Operating System	OS X Sierra

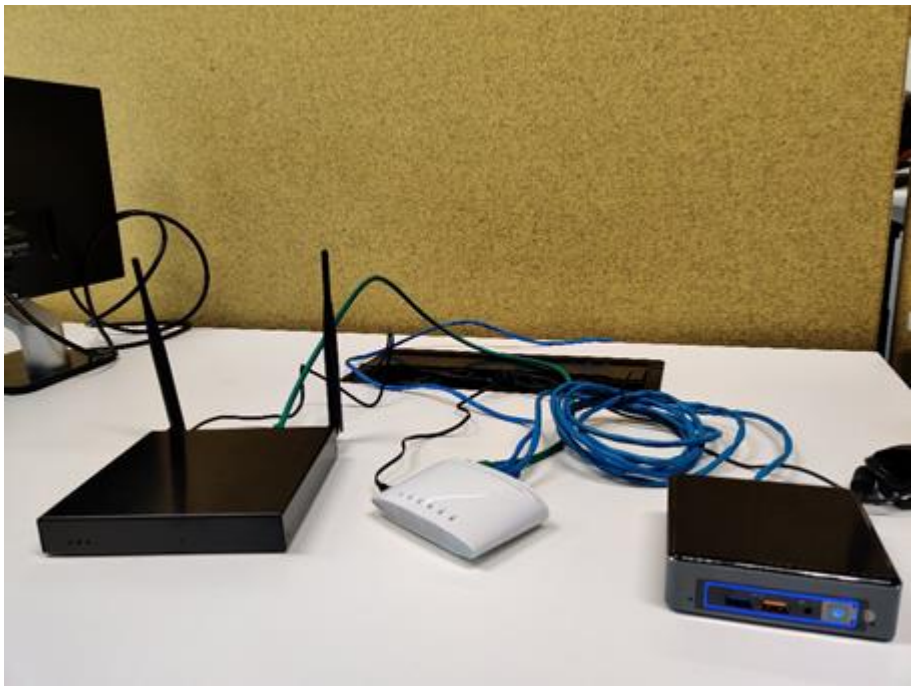

Figure III.5-1 Access Point and Processing Unit

The 5G-EmPOWER components are threefold: EmPOWER controller, Backhaul controller, and the Wireless Termination Point (WTP). The EmPOWER controller is the component responsible for the management of the Radio Access Network (RAN) by deploying the necessary configuration in the WTP, which will prioritize the network traffic. The Backhaul controller manages the Software-Defined Networking switches, which identify and tag the network packets that need to be prioritized. These tagged network packets will be identified by a packet processor responsible for prioritizing the wireless network traffic. Both controllers are deployed in the Intel NUC wrapped up in Docker containers and accessible by the access points. The WTPs are the WiFi access points where the clients connect to, and the network bandwidth is monitored and managed. This experiment was set up by deploying an iPerf3 server in the Intel NUC and iPerf3 client in the MacBook, simulating a ProTego application scenario.

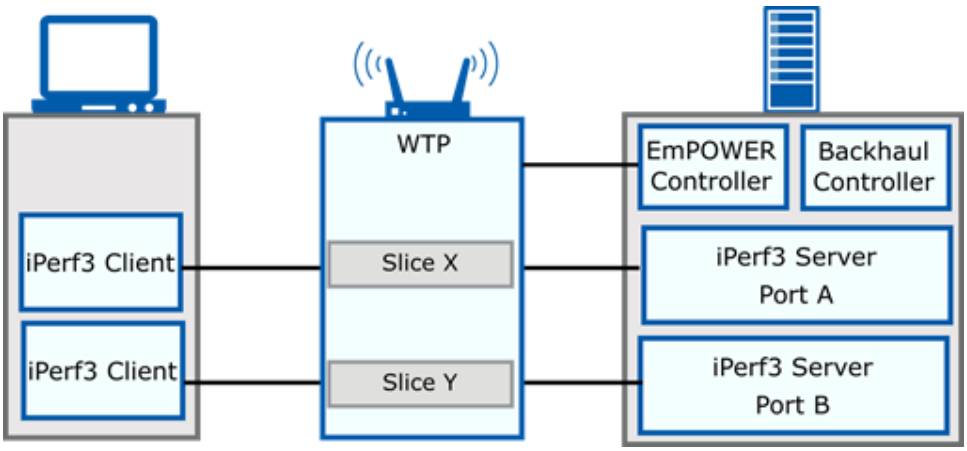

Figure III.5-2 ProTego Application Scenario Simulation

III.5.2. Privacy Isolation

For the privacy isolation, the experiment was set up in two virtual machines, as can be seen in Figure III.5-3. The virtual machines were connected using a NAT network with the host computer. Our objective was to install and set up a virtual OpenFlow switch with a default interface for communication between the two VMs and an alternative interface that provides connectivity between the VMs as well but encrypts all the traffic passing through the network interface. Therefore, with the virtual OpenFlow switch, we can redirect the flow to the default interface or the secure interface for different services based on information available in the network packets such as destination IP address and port. The tools utilized in the experiment were Open vSwitch (OvS) as the virtual OpenFlow switch to divide different services network traffic, OvS VXLAN plugin to create a virtual network interface for different slices, iPerf to emulate a service flow, and an ICMP client to follows a second service. The first VM plays the role of the access points where the traffic is routed to a gateway to reach its destination. The second VM is the gateway, which will decrypt the packets and route it to its target, which can be in the local area network or through the Internet.

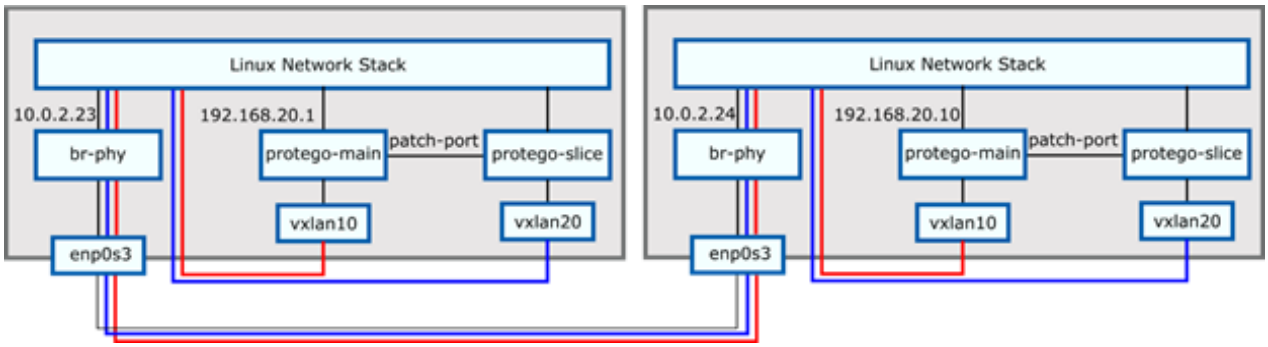

Figure III.5-3 Privacy Isolation experimental setup

For both VMs, we created three OvS bridges, one connected to the physical interface, ens0p3, and the other two, protego-main and protego-slice, with the VxLAN tunnel as can be seen in Figure III.5-4. One of the VxLAN tunnels is the default slice, and the other is the secure slice with encryption. The VxLAN tunnels were set up with different VNIs, so the traffic cannot be accessed from a different VxLAN. Moreover, the VNI tagging helps us to identify the traffic passing through the physical interface. The OvS Bridge for the physical interface is recommended to support the routing of the VxLAN encapsulated packets to the external network. In order to enable the L2 routing between the two VxLAN OvS Bridges, it is necessary to create a connection point called Patch Ports. With these ports, the OvS switch has a direct connection to the secure slice interface and can utilize OpenFlow rules to redirect the traffic to this port. In the Gateway VM, it is also

necessary to enable ipv4 forwarding and proxy ARP in the VxLAN OvS bridge to answer ARP requests from the AP VM. Moreover, it is necessary to create iptables rules to enable the forwarding among the network interfaces and NAT connectivity to external networks.

```
jackpot-protego:~$ sudo ovs-vsctl show
1981b982-d821-498a-8018-84d5212a7640
    Bridge protego-main
        fail_mode: standalone
        Port "vxlan10"
            Interface "vxlan10"
                type: vxlan
                options: {key="10", remote_ip="10.0.2.24"}
        Port "patch10-20"
            Interface "patch10-20"
                type: patch
                options: {peer="patch20-10"}
        Port protego-main
            Interface protego-main
                type: internal
    Bridge protego-slice
        fail_mode: standalone
        Port "patch20-10"
            Interface "patch20-10"
                type: patch
                options: {peer="patch10-20"}
        Port "vxlan20"
            Interface "vxlan20"
                type: vxlan
                options: {key="20", remote_ip="10.0.2.24"}
        Port protego-slice
            Interface protego-slice
                type: internal
    Bridge br-phy
        fail_mode: standalone
        Port br-phy
            Interface br-phy
                type: internal
        Port "enp0s3"
            Interface "enp0s3"
    ovs_version: "2.11.0"
```

Figure III.5-4 Configuration of the three Open vSwitch bridges

III.6. CONTINUOUS AUTHENTICATION

Continuous authentication is a technique used after a user is authenticated. This technology allows us to detect if a device, a mobile phone in this scenario, of a user is used by a different person. The ProTego Continuous Authentication (ProTego CA) system is a mobile application that works as Endpoint Detection and Response (EDR). This application implements continuous authentication systems and algorithms to detect the unauthorized use of a mobile phone.

The application monitors the activity and usage patterns of mobile phone users. If an anomalous usage is detected, the application can notify and response to such behaviour. The first release of the app monitors the keyboard usage. Based on how a user uses their keyboards, we can detect if a different user is using the mobile phone. After installing ProTego CA, the keyboard of the device will be replaced by a keyboard monitored by ProTego CA (see Figure III.6-1). This new keyboard will take different measures to train the system, like the typing speed or how long a user presses a key, to train the system.

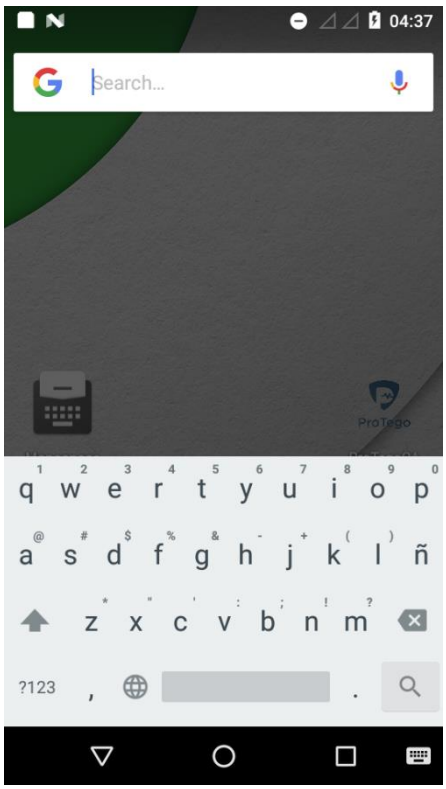

Figure III.6-1 ProTego CA app

In order to monitor the activity and usage patterns is necessary to give special permissions to the ProTego CA app (see Figure III.6-2). Once the user installed and accepted this permissions keyboard measurements training can take place. Using this information ProTego CA can detect if the mobile device is being used by another person.

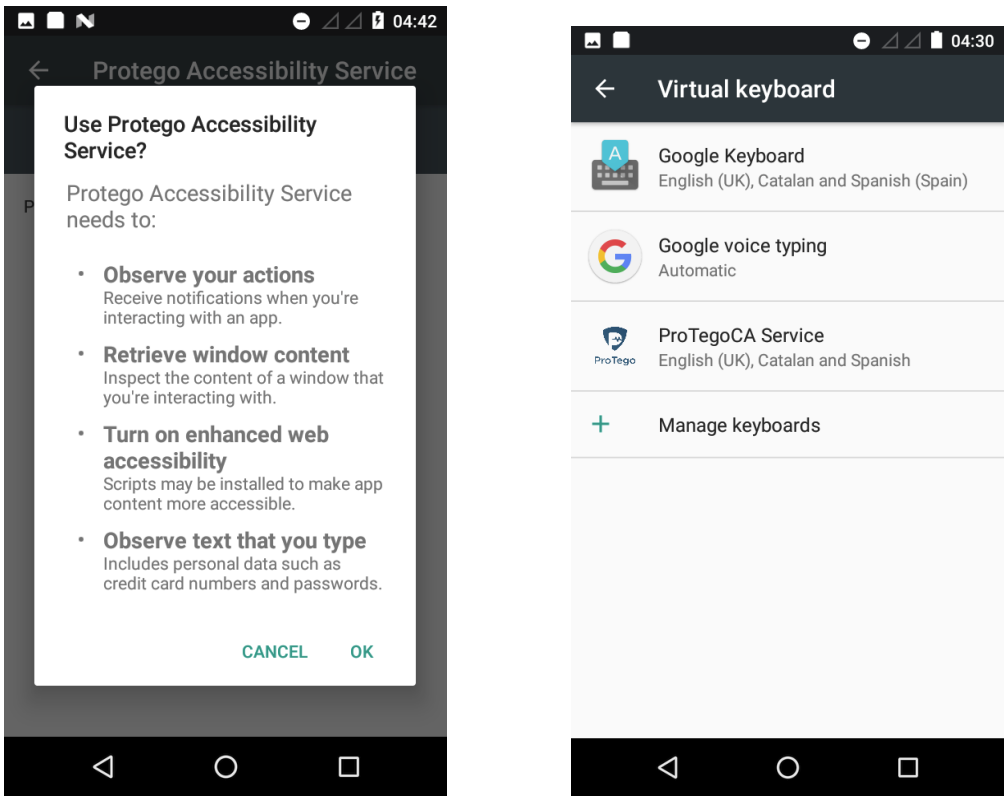

Figure III.6-2 ProTego CA permissions and service

After a period of training, the ProTego CA will be able to measure and monitor the “continuous authentication” value of the user. The app can show a status bar that showing and estimation of the matching for the current user. If the bar is less than a threshold value of 50%, the user would not be recognized, meaning that someone else could be using the phone app (see Figure III.6-3).

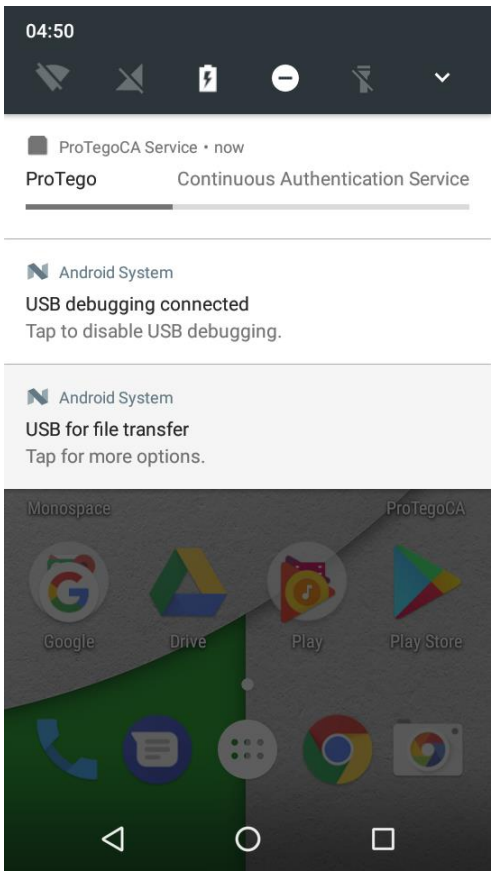

Figure III.6-3 ProTego CA app user recognition

One of the scenarios presented by the hospitals was Bring Your Own Device (BYOD). For validation partners, Marina Salud and San Raffaele mobile device security is a challenge. They are developing mobile apps and sensitive information is being stored in mobile phones of medical personnel.

To provide a working demo of the system, the ProTego CA app will be installed and trained in a test mobile phone. After that, this unlocked mobile phone can be used by other users showing the change in the status bar, which measures the detection of the original user.

The objective of this demo will be twofold: show the detection of an authorized user and detect the usage of a mobile phone by an unauthorized user. This application matches the problematic and use case of OSR and MS with bringing your device (BOY) policies. Hospital personnel is bringing personal mobile phones and using them to access medical and sensitive information. For example, installing ProTego CA in these mobile phones can allow the detection of phone that has been stolen to their staff, and revoke access to sensitive information.

IV. RESULTS

IV.1. SSM

The validation partners have successfully designed and performed a preliminary cybersecurity risk assessment of their system models using the SSM tool where the ProTego-specific domain model is deployed. Figure IV.1-1 and Figure IV.1-2 show the resulting risk models for the validation scenarios of Marina Salud and San Raffaele respectively.

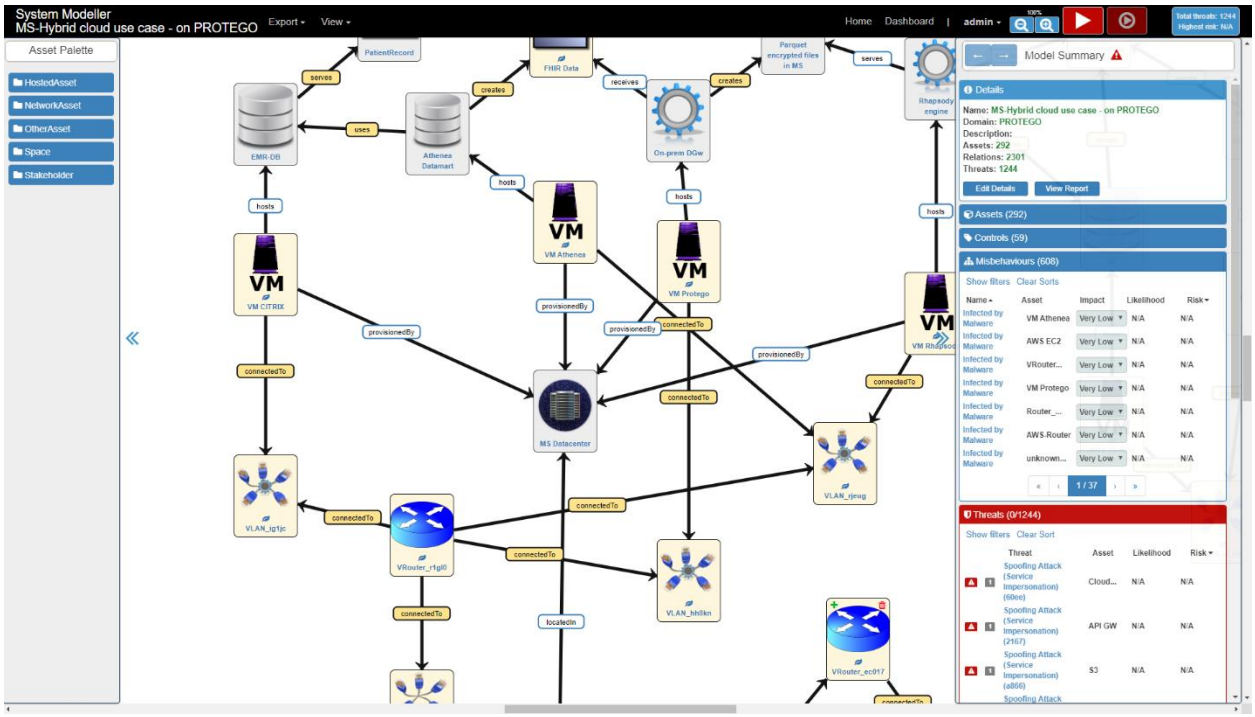

Figure IV.1-1 Snapshot of the initial risk model developed for the validation scenario of Marina Salud

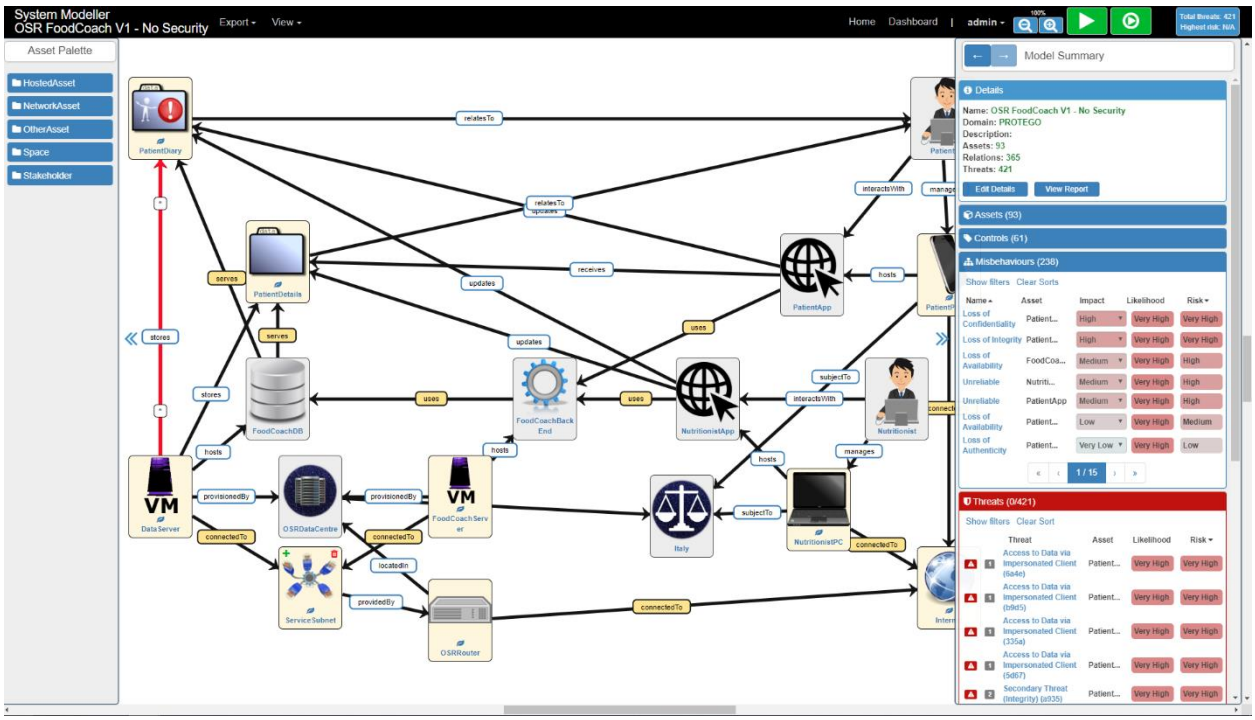

Figure IV.1-2 Snapshot of the initial risk model developed for the validation scenario of San Raffaele

Following the steps outlined above (see III.1.), the users first produced a graphical model of the overall architecture of their validation scenario in SSM. To do so, they defined the assets of the system by choosing the allowed asset types from the domain model and assigning meaningful labels, such as ‘PatientDetails’ for an asset of type data to indicate the kind of data that are stored and processed by the system (Step 1; see Figure IV.1-4).

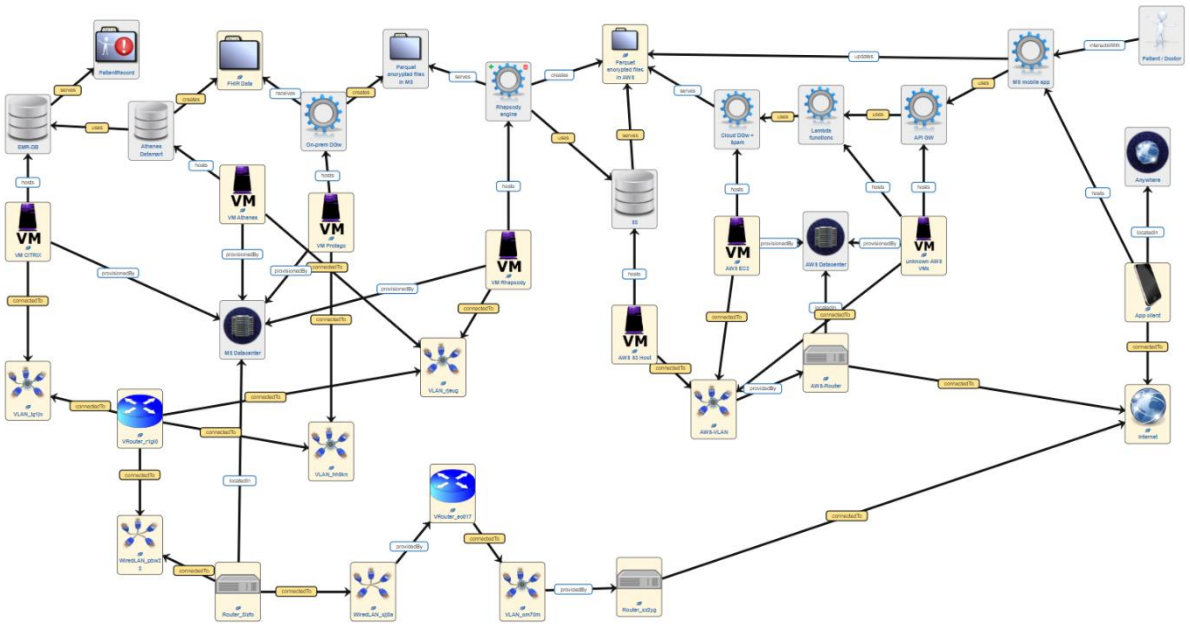

Figure IV.1-3 System model of the IT environment of Marina Salud

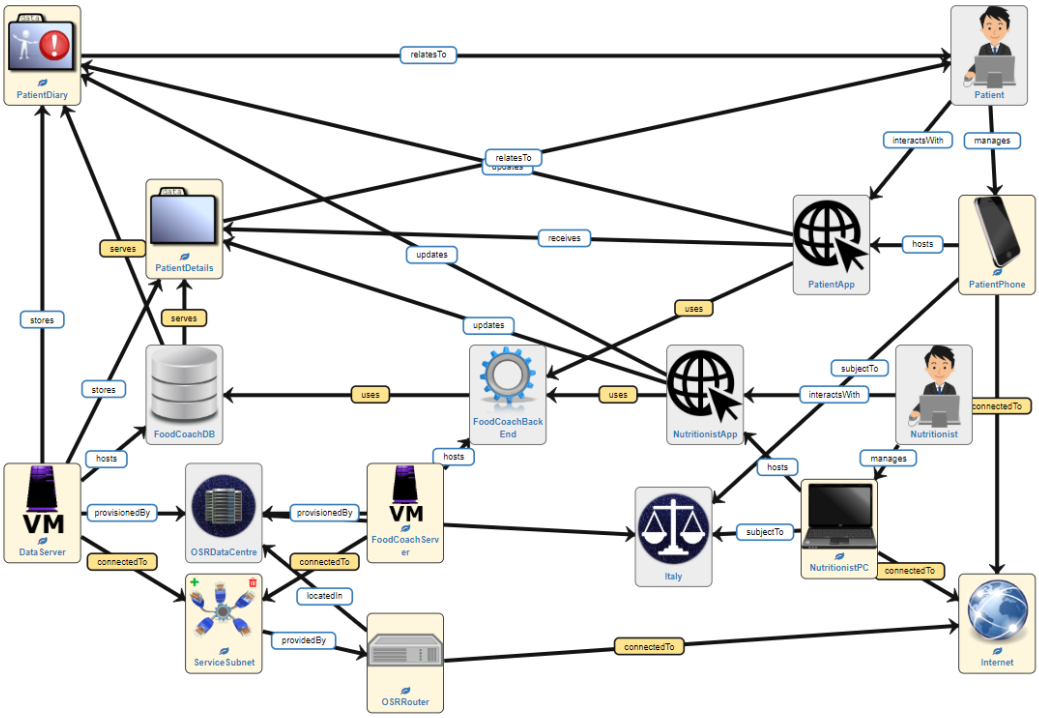

Figure IV.1-4 System model of the IT environment of San Raffaele

Once the assets had been put onto the model canvas, the users described the underlying relationships between pairs of assets in their system by connecting different components using unidirectional links (Step 2). Figure IV.1-3 and Figure IV.1-4 show the constructed system models for the validation scenarios of Marina Salud and San Raffaele respectively. The models included several simplifications to reduce complexity.

When the complete model was constructed, it was subsequently validated to ensure consistency among different components as well as inferring assets and relations that were not explicitly specified by the user (asserted) using semantic reasoning (Step 3). During the validation step the SSM generated a list of potential threats and misbehaviours that can be associated with the given system model. For the case of MS, the model is comprised of 292 assets, out of which 38 are asserted assets that were specified by the users corresponding to the observable square nodes of the graph. Similarly, the model of OSR is comprised of 93 assets, out of which 17 are asserted.

Once the model was complete and validated, the users determined whether to change some of the initial assumed trustworthiness levels for assets (Step 4). For example, this could be achieved by reducing the intrinsic trustworthiness level of patients, who have physical access to the ward and hence some physical access to its wireless network and devices; or by reducing the user trustworthiness of the hospital network, which may be appropriate if it has large numbers of users or users not employed directly by the hospital. In either case, we expect that a reduction in the level of trustworthiness leads to an increase in the likelihood of numerous threats.

At this stage, it was possible to calculate risk levels for all threats and misbehaviours (Step 5). The process of risk calculation can be re-run every time a trustworthiness level or misbehaviour impact level is changed, or when a security measure is selected (or deselected) to reduce the likelihood of a potential threat.

The panels on the right side of Figure IV.1-5 and Figure IV.1-6 show the resulting list of misbehaviours and potential threats for the two validation scenarios of MS and OSR respectively after the validation step where no security controls are considered. For the given models, we find 1244 potential threats that can lead to 608 misbehaviours for the case of MS, while we observe

421 potential threats and 238 misbehaviours for the case of OSR. The higher number of risks for the case of MS is partly explained by the size of their network; i.e., 292 assets with 2301 relations as opposed to 93 assets and 365 relations for the case of OSR. Note that usually there are far fewer high-risk misbehaviours than high-risk threats.

The misbehaviour explorer panel on the left side of Figure IV.1-5 and Figure IV.1-6 describes one of the possible misbehaviours that can directly or indirectly be caused by the associated threats listed below. For instance, for the case of OSR, Loss of Confidentiality on the patient diary data can be caused by 20 potential threats, such as 'Access to Data via Impersonated Client' or 'Unauthorised Remove Access'. When looking at causes of a specific misbehaviour, it is often helpful to rank based on the likelihood rather than the risk level. By checking the misbehaviours of each asset to find the ones that correspond to high risk, the cybersecurity analysts can explore and identify the threats that most need addressing.

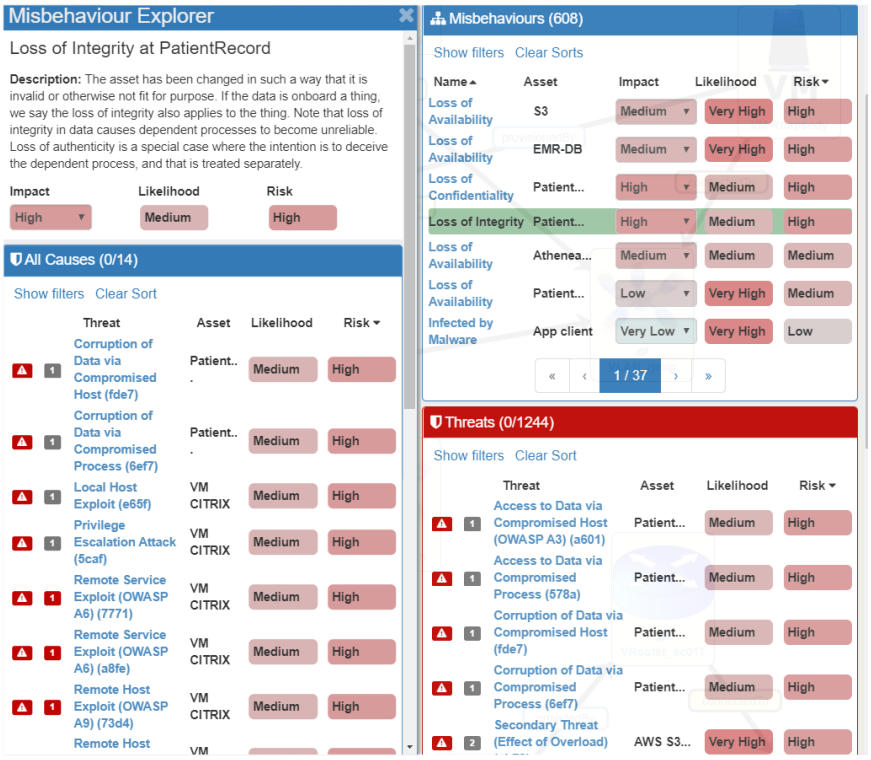

Figure IV.1-5 List of misbehaviours and threats for the validation scenario of MS where no security measures are considered.

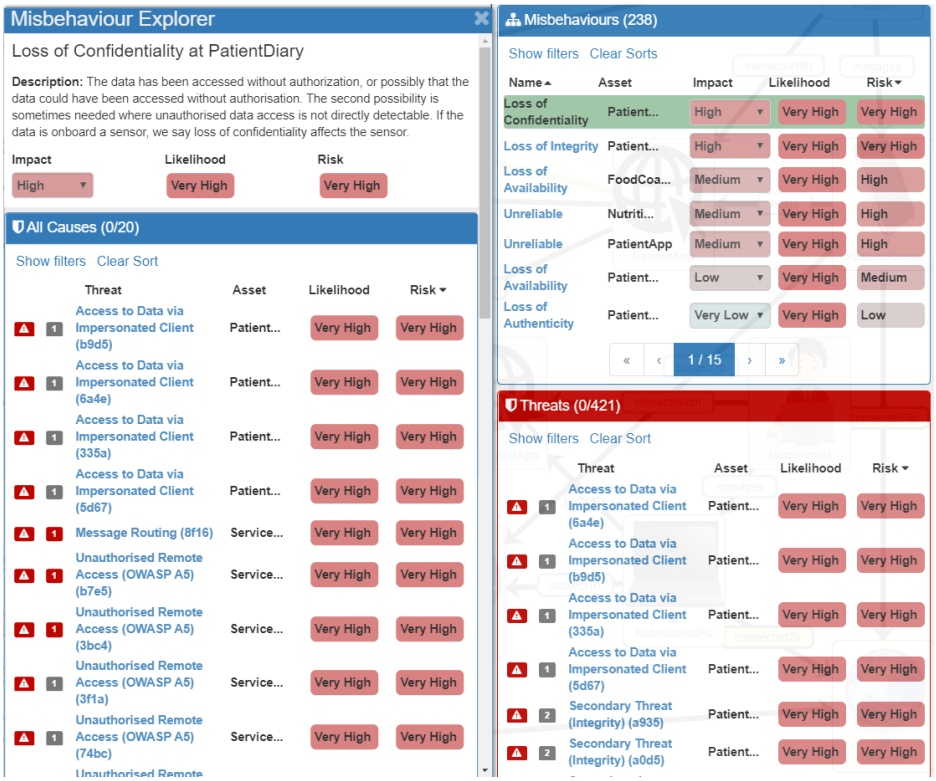

Figure IV.1-6 List of misbehaviours and threats for the validation scenario of OSR where no security measures are considered.

Following the detection and identification of the root causes of a misbehaviour, the users introduced security measures to address high risk threats by intervening in attack paths and/or containing secondary effects. Figure IV.1-7 shows the validation scenario of OSR where 'basic' security measures are introduced to address the majority of common threats. In this scenario, we see that a number of 'Very High' level risks involving loss of confidentiality or integrity in patients' data have been appropriately addressed by security controls lowering the Likelihood from 'Very High' to 'Low', and as a result reducing the risk level to 'Medium'. The process can be continued iteratively by identifying 'residual' risks and specifying extra security measures to address them. This progressively reduces the number of threats at each risk level until there are none above an acceptable risk level.

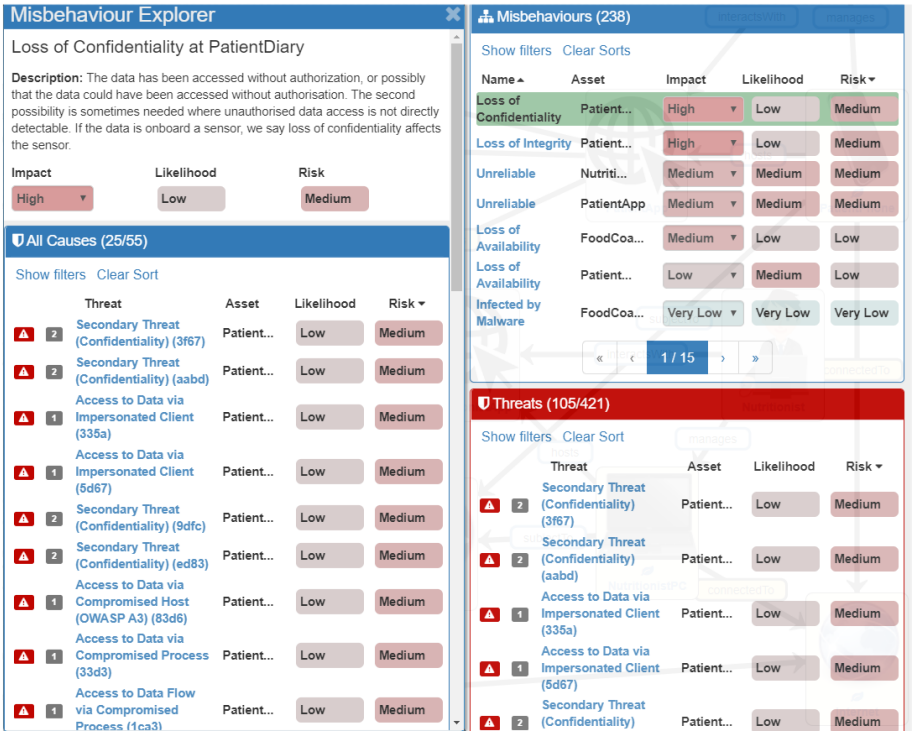

Figure IV.1-7 List of misbehaviours and threats for the validation scenario of OSR where basic security measures are considered

IV.2. SIEM

After the system has been collecting data for several weeks from the monitored machines, there is enough information to be analysed. As expected, log events are processed and stored as Elasticsearch documents:

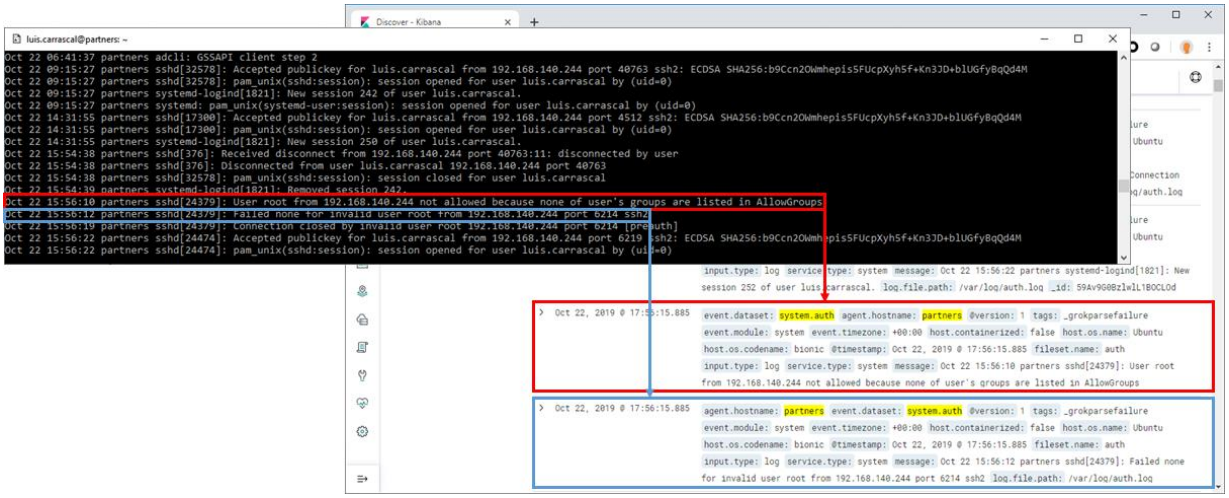

Figure IV.2-1 Log events stored as Elasticsearch documents

Nevertheless, there is a load of information that is stored and needs to be refined. For example, after logging in to the Windows operating system of the Alzheimer virtual machine, five different events are generated that contain the account name of the logged user:



Figure IV.2-2 Events generated for a Windows log in

But in addition, at least five other events are generated for the same logging in that do not contain the account name of the user.

When analysing the captured packets of network traffic for 24 hours in the Alzheimer virtual machine the statistics showed a high number of packets for a testing environment (258,325 packets):

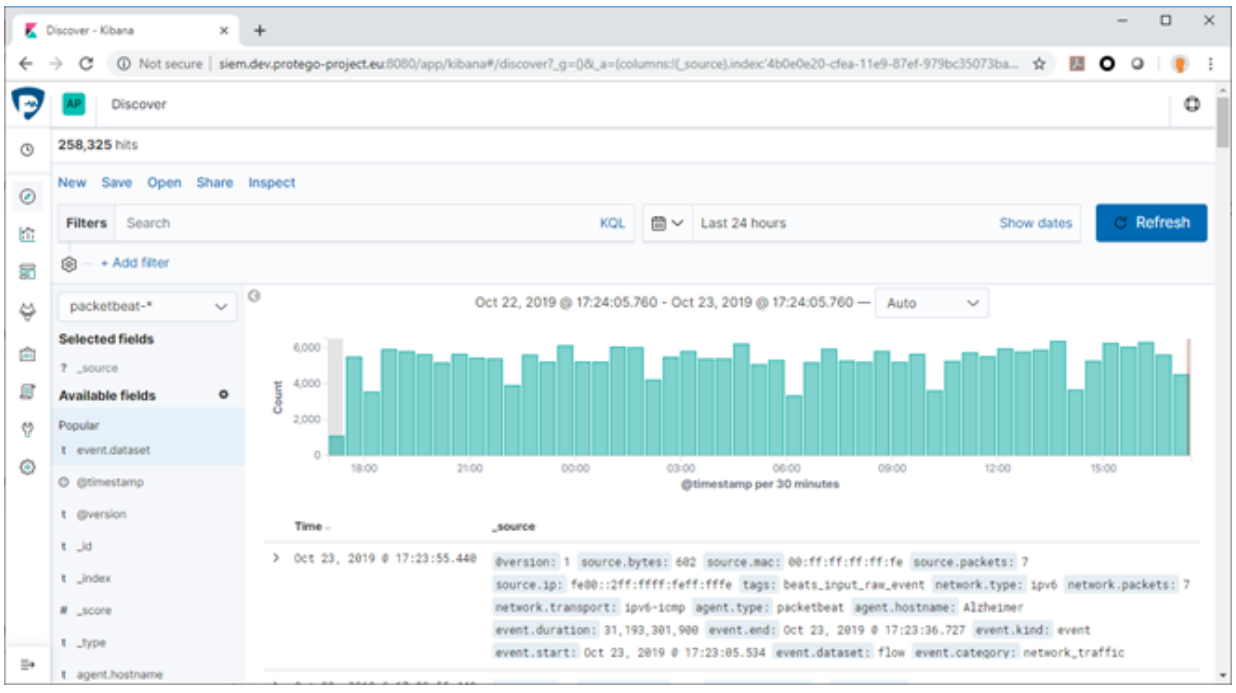

Figure IV.2-3 Packets captured for 24 hours on Alzheimer virtual machine

Once the packets were reviewed, it was easy to see a high load of IPv6 traffic: 244,330 packets, or nearly 95% of all the traffic:

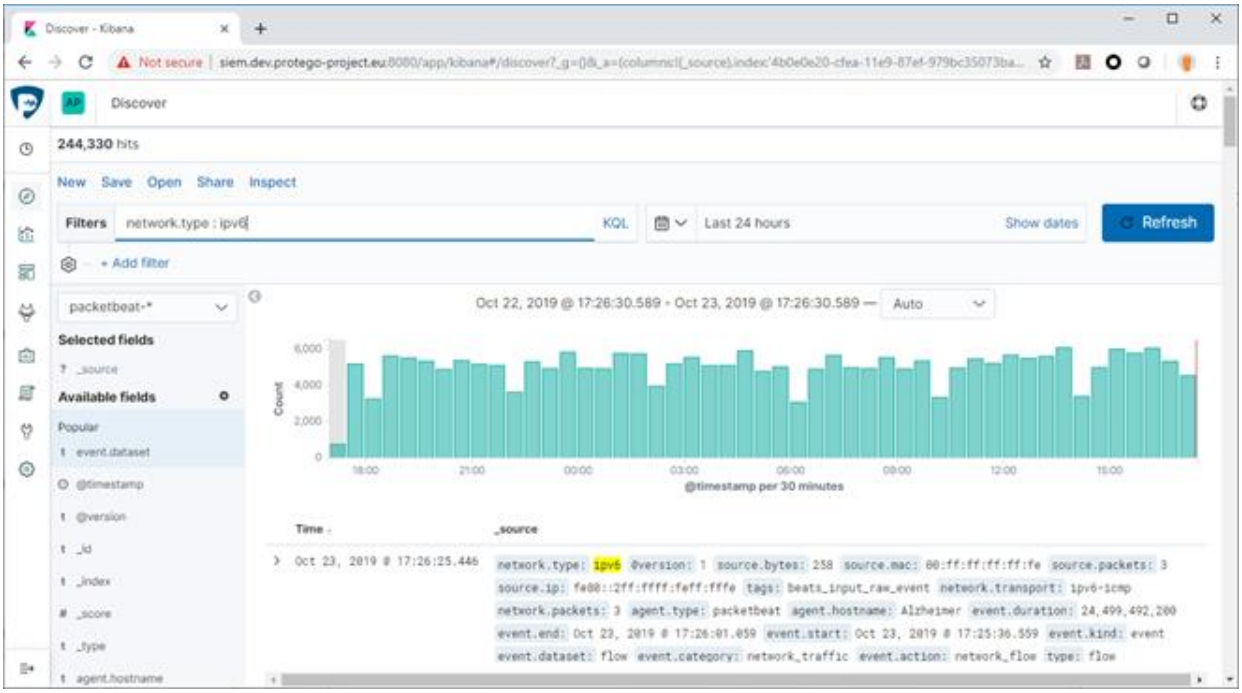

Figure IV.2-4 IPv6 packets

IPv6 is not used in the whole PRE-Production environment and therefore it could be a misconfiguration of the virtual machine. But after checking the settings, we could verify that they were correct:

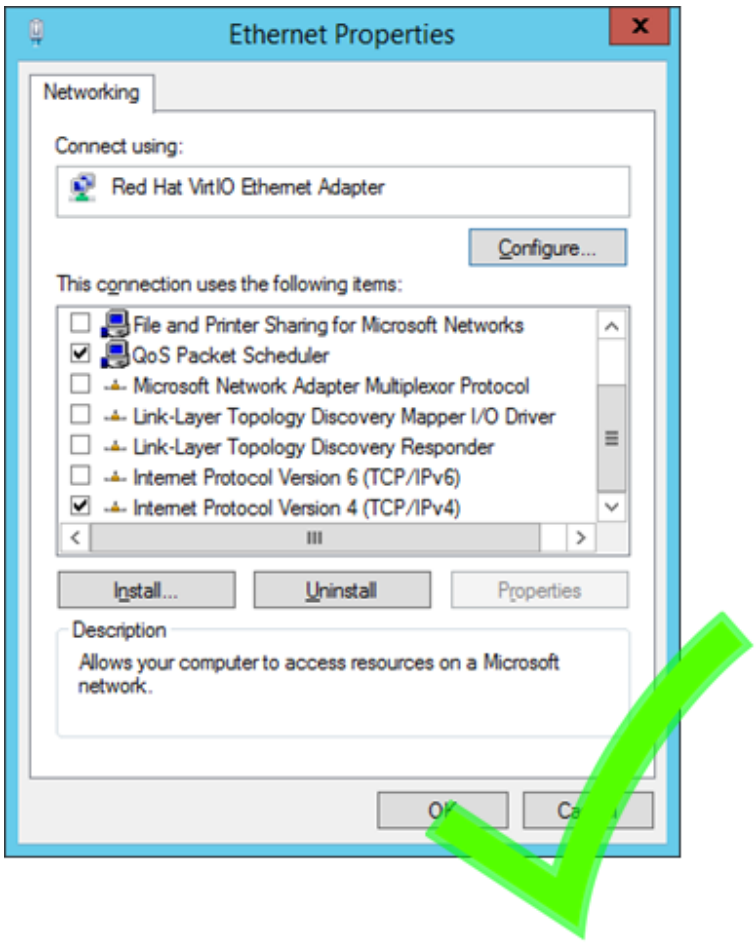

Figure IV.2-5 Alzheimer virtual machine network settings

It could also be a misconfiguration in the settings of the physical server that was hosting the virtual machine:

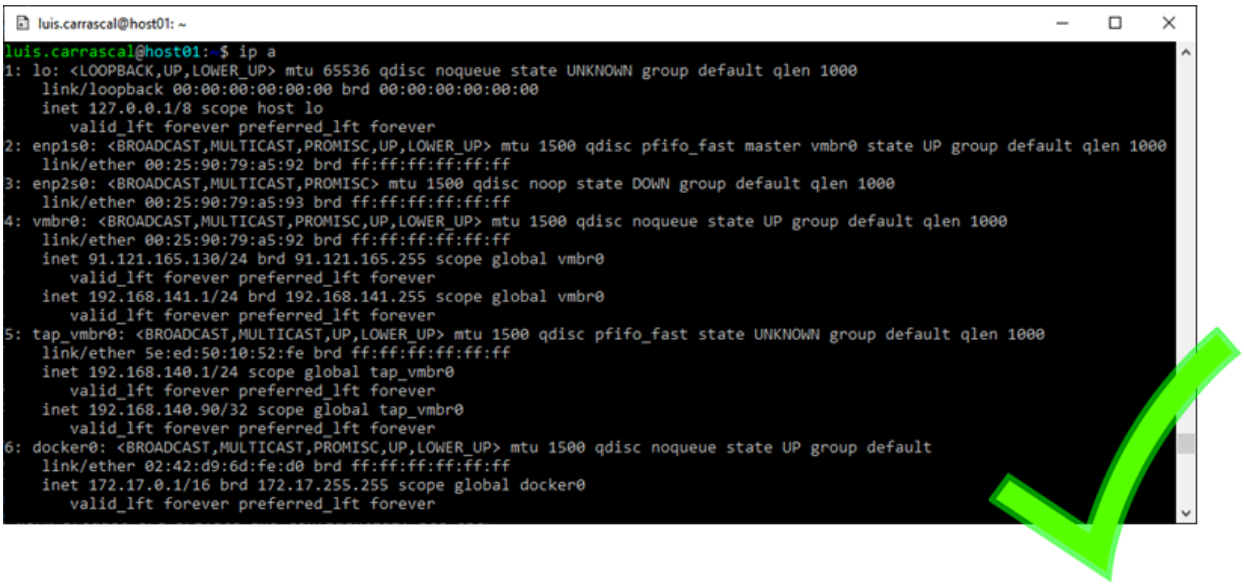

Figure IV.2-6 Host network settings

But again, once the settings for the physical host were checked, it was clear that it was not a misconfiguration in the environment.

A deeper analysis of the packets showed that IPv6 traffic was not part of the PRE-Production environment itself, but it was broadcast traffic in the OVH network, the Internet Service Provider where the physical servers are hosted, as it is showed in the following diagram:

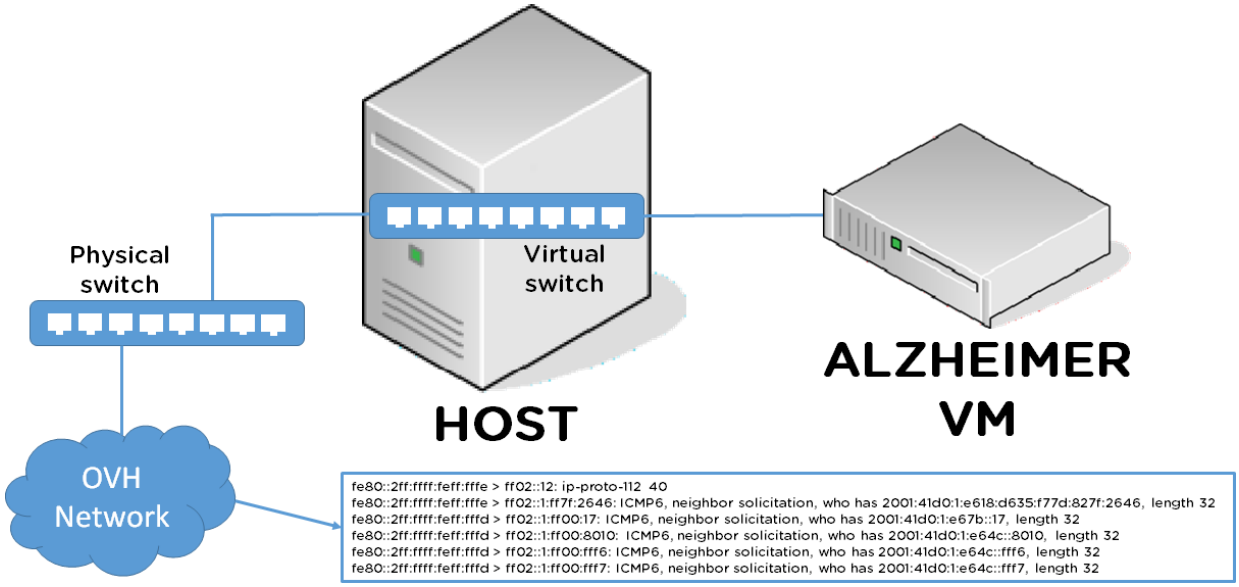

Figure IV.2-7 Why IPv6 packets are seen

The Alzheimer virtual machine is connected to the physical host through a virtual switch. That virtual switch is connected to a physical switch through the host network interface card. The physical switch is part of OVH network where that IPv6 traffic is generated.

These two examples show that when the system is deployed on a Production environment, it is necessary to perform a data ingest for some time that helps to establish a baseline of events. With that baseline, all collected data needs to be carefully analysed, and along with a knowledge of the infrastructures where the SIEM is deployed, it is essential to perform some fine tuning. Otherwise there will be a flood of events that will most likely be ignored by a human analyst.

This last example, IPv6 traffic, will be used to show how the tuning could be optimally performed. Since Logstash pipelines process the events, it might seem reasonable to think that this is where the filtering of IPv6 packets should take place. But that would generate a lot of unnecessary traffic and a processing overload that might impact the system. Packetbeat allows to establish a capture filter for the traffic that is analysed. Therefore, that is the optimal place, since the unnecessary traffic never leaves the monitored machine and does not overload a server component:

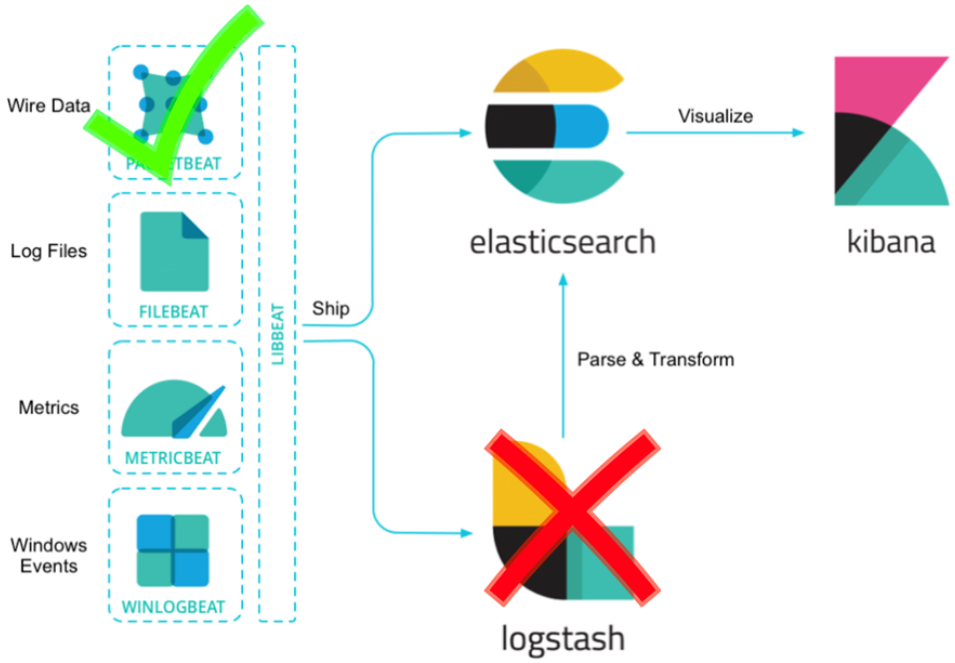

Figure IV.2-8 Example of Packetbeat tuning

This is just an example that shows how important it is to choose the right approach in fine tuning, not just for the data that is stored but also for performance reasons. It goes without saying that each case needs to be analysed and depending on what is found, different solutions will be tackled.

The access log generated by the Nginx reverse proxy was analysed as well. This analysis showed the reality of what it involves nowadays to expose a service on the Internet, even without openly publishing it or registering it in public search engines:

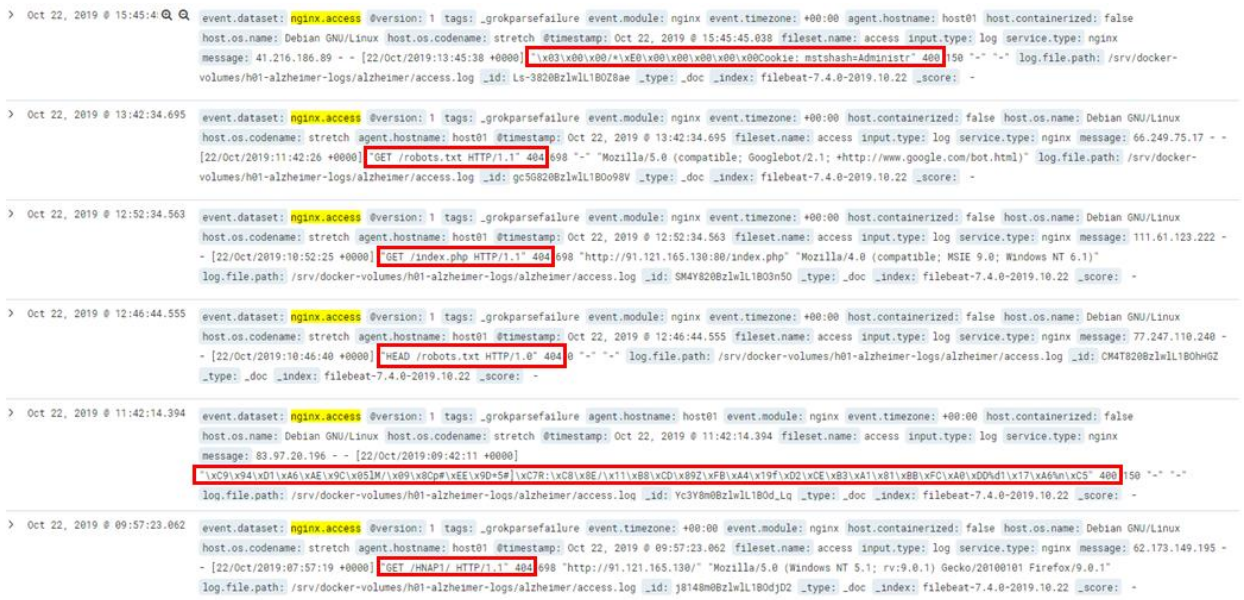

Figure IV.2-9 Nginx reverse proxy access log

Most of the events are illegitimate traffic. Some unauthorized users scan blocks of IP addresses to find open ports, and once they identify the service exposed, they try to hack into the system using different methods. The service exposed by Internet Information Server on the Alzheimer

virtual machine uses an ASP.NET web service. Therefore, the events that request the robots.txt file or any PHP file are illegitimate. The events where the requests are a sequence of binary codes expressed in hexadecimal notation are even a clearer example. Considering the service exposed, any request that returns a HTTP code 400 Bad Request, or 404 Not Found, are evidently illegitimate because the application in client devices uses a predefined API where those codes are not an option.

Something similar happens when syslog events for the Bind DNS service on the physical hosts are analysed:



Figure IV.2-10 Bind DNS service log

Requests that are refused, denied, or fail because of an unexpected end of input clearly show illegitimate events.

Beyond the obvious security risks that these last two examples represent, they also pose a problem for future progress in the development of the SIEM, when Machine Learning techniques are used to detect anomalies, because the majority of the events are an anomaly and there are only just a few legitimate events.

Therefore, for further progress in the development it is necessary to deploy the SIEM in hospitals to monitor a realistic scenario where enough data is collected that can be used for Machine Learning. Both Marina Salud and San Raffaele have been informed about this and collaborative plans are going to be studied soon to tackle this task.

Some dashboards and reports have been tested, that give an overview of the data that has been stored in the system:

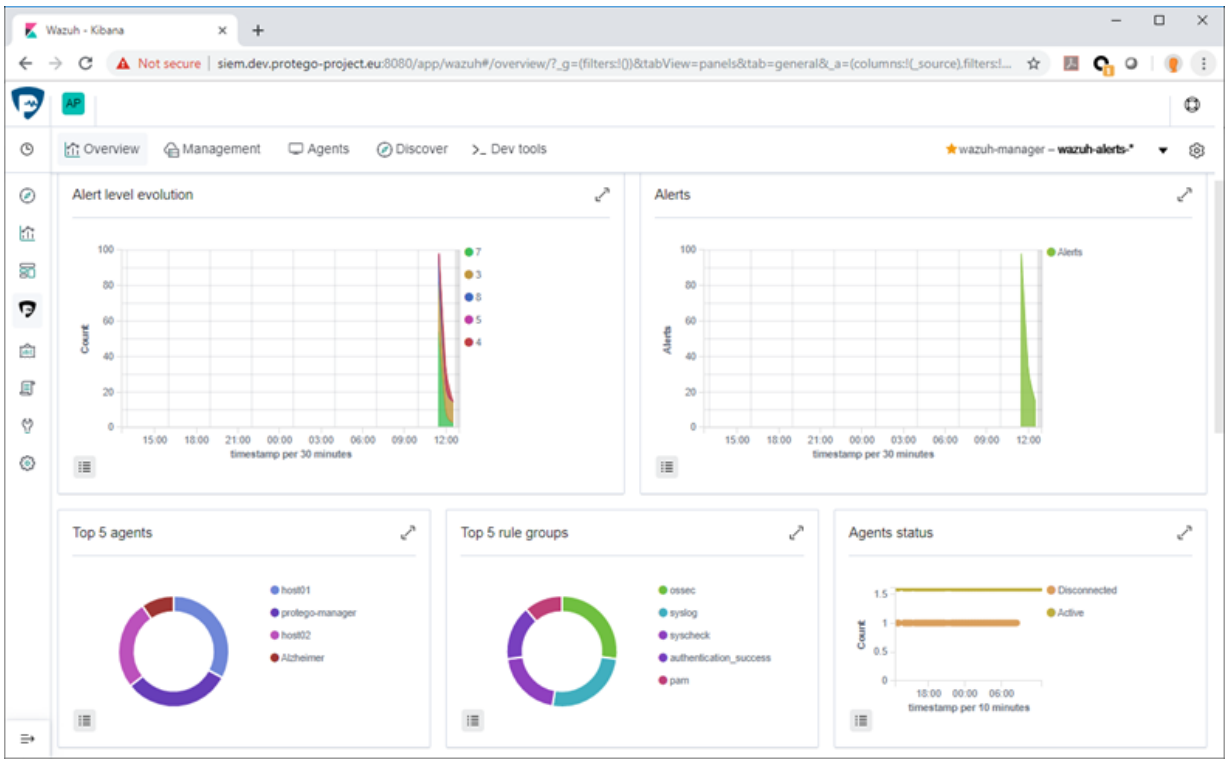

Figure IV.2-11 Example Dashboard

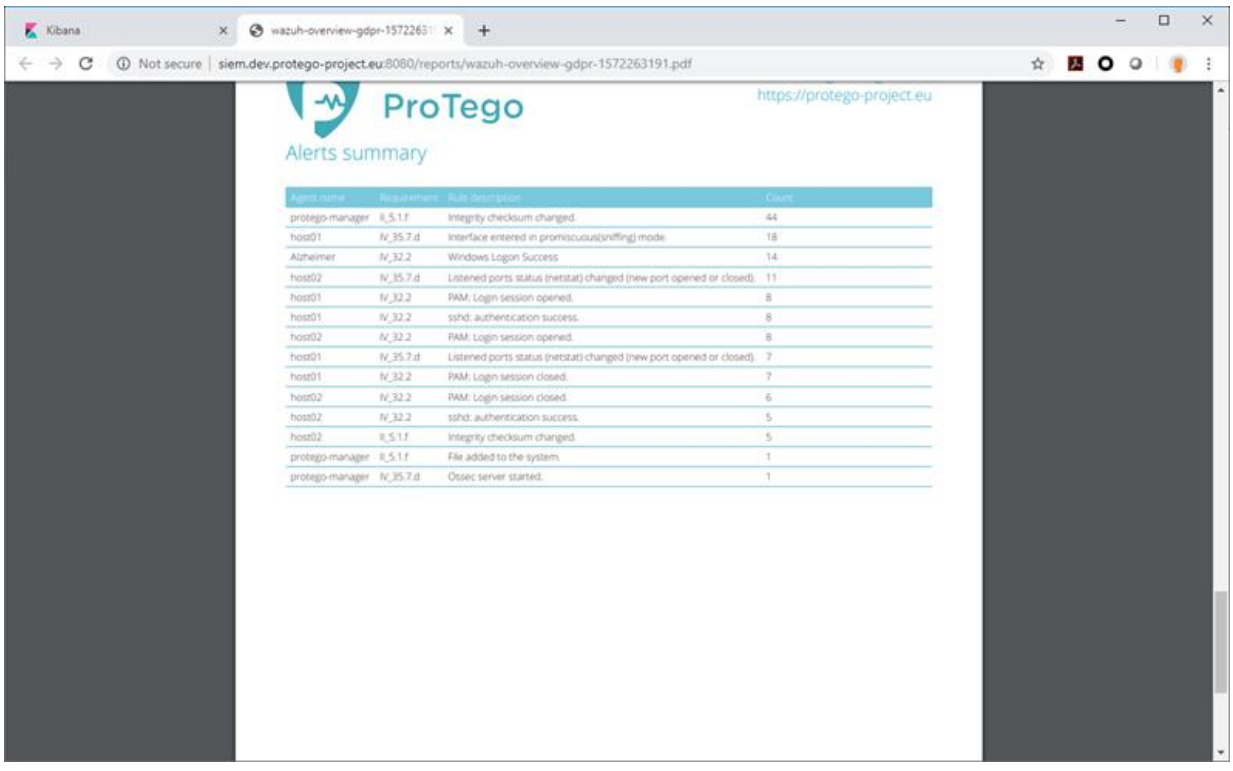

Figure IV.2-12 Example Report

Finally when testing OpenVAS, the vulnerability scanner, some reports were generated that can be exported in XML format, what facilitates the integration with the other components to pass vulnerability information to the SSM:

```
  <port>3306/tcp</port>
- <nvt oid="1.3.6.1.4.1.25623.1.0.142651">
    <type>nvt</type>
    <name>Oracle MySQL 5.7.x < 5.7.27, 8.0.x < 8.0.16 Security Update (2019-5072835) - Windows</name>
    <family>Databases</family>
    <cvss_base>7.5</cvss_base>
    <cve>CVE-2019-3822, CVE-2018-16890, CVE-2019-3823</cve>
    <bid>NOBID</bid>
    <xref>URL:https://www.oracle.com/technetwork/security-advisory/cpujul2019-5072835.html#AppendixMSQL</xref>
    <tags>cvss_base_vector=AV:N/AC:L/Au:N/C:P/I:P/A:P|qod_type=remote_banner|solution_type=VendorFix|summary=Oracle
      MySQL is prone to multiple vulnerabilities in libcurl.|vuldetect=Checks if a vulnerable version is present on the target
      host.|affected=MySQL 5.7.26 and prior and 8.0.15 and prior.|solution=Update to version 5.7.27, 8.0.16 or later.</tags>
  - <cert>
      <cert_ref id="CB-K19/0623" type="CERT-Bund"/>
      <cert_ref id="CB-K19/0620" type="CERT-Bund"/>
      <cert_ref id="CB-K19/0321" type="CERT-Bund"/>
      <cert_ref id="CB-K19/0320" type="CERT-Bund"/>
      <cert_ref id="CB-K19/0120" type="CERT-Bund"/>
      <cert_ref id="DFN-CERT-2019-2303" type="DFN-CERT"/>
      <cert_ref id="DFN-CERT-2019-1453" type="DFN-CERT"/>
      <cert_ref id="DFN-CERT-2019-0772" type="DFN-CERT"/>
      <cert_ref id="DFN-CERT-2019-0771" type="DFN-CERT"/>
      <cert_ref id="DFN-CERT-2019-0563" type="DFN-CERT"/>
      <cert_ref id="DFN-CERT-2019-0357" type="DFN-CERT"/>
      <cert_ref id="DFN-CERT-2019-0315" type="DFN-CERT"/>
      <cert_ref id="DFN-CERT-2019-0248" type="DFN-CERT"/>
    </cert>
  </nvt>
  <scan_nvt_version>2019-07-24T02:19:13+0000</scan_nvt_version>
  <threat>High</threat>
  <severity>7.5</severity>
- <qod>
    <value>80</value>
    <type>remote_banner</type>
  </qod>
  <original_threat>High</original_threat>
  <original_severity>7.5</original_severity>
```


Figure IV.2-13 Vulnerability scan XML report

The report includes for each vulnerability detected the CVE-IDs, that is the essential information that is to be passed to the SSM to recalculate risks.

IV.3. DATA GATEWAY

As shown in Figure IV.3-1, reported meal portions as Observation resources are captured by the FHIR interceptor and stored as Parquet format. The interceptor code allows to batch captured store (POST) messages in order to control the number of meal observations stored in each Parquet file.

```
salant@sgx-lnx:~/vault_0.11.5$ ls ~/fhir/gateway/observation_IBM_FHIR.parquet.encrypted/
part-00000-05cb4258-804b-4347-b523-7d4f3e052920-c000.snappy.parquet  part-00000-80ae478a-409a-4471-addb-b2cc86e877ba-c000.snappy.parquet
part-00000-0752a8fe-a1fc-472c-8d42-5f6b2287260e-c000.snappy.parquet  part-00000-862e7578-14d1-4ad8-bf8a-a7a307c94fb2-c000.snappy.parquet
part-00000-095fede4-adb8-4c6d-8ec7-e7b8b5f801d7-c000.snappy.parquet  part-00000-86ffd48d-83c4-42bb-a5dd-e6eeb2867ac8-c000.snappy.parquet
part-00000-0e2befad-b32a-4bf8-8cf7-1ed286dc928b-c000.snappy.parquet  part-00000-878baed5-f2fa-408d-b0db-99ed29c40256-c000.snappy.parquet
part-00000-0ee6e3fa-0862-448a-9439-4df14e2af9c1-c000.snappy.parquet  part-00000-8dabebd8-df9b-4482-8013-1e83c9c64d18-c000.snappy.parquet
part-00000-14f2eb3a-bead-4fe4-8e76-f068dcce6f11-c000.snappy.parquet  part-00000-912e3172-6c7d-4300-b66a-bf56fecf4743-c000.snappy.parquet
part-00000-15810d0f-b780-4a2e-b3d6-d391d338625c-c000.snappy.parquet  part-00000-94964ae3-f02a-43c8-974d-340448fe365d-c000.snappy.parquet
part-00000-16e995cf-fc7d-4409-9eed-8358ef7de062-c000.snappy.parquet  part-00000-95d0e349-8ba8-432c-ac04-e0907730fe51-c000.snappy.parquet
part-00000-17fdafaf-7f4b-44cc-bf02-99cc65d13c43-c000.snappy.parquet  part-00000-967c6810-6a3b-4d0f-bd9d-0d1626714042-c000.snappy.parquet
part-00000-1af64154-2b9d-47f2-b2bb-8ec9d378557d-c000.snappy.parquet  part-00000-96beba4c-eb93-4b70-a138-9ccaddfd447e-c000.snappy.parquet
part-00000-1b5fd94e-4222-464e-88ec-7482409dd135-c000.snappy.parquet  part-00000-96e9f08d-cc22-49a7-a54a-4b90224e2401-c000.snappy.parquet
part-00000-1e31d779-e314-4c51-93db-e6e7b957fcc7-c000.snappy.parquet  part-00000-983905d4-7f90-4e64-8c63-4fc0a9a015fa-c000.snappy.parquet
part-00000-1f663b32-7fdb-4a02-98ca-6fb561b916b9-c000.snappy.parquet  part-00000-9ac38684-f4c8-46cd-beb7-ed9caed7c86b-c000.snappy.parquet
part-00000-20c25f7e-c525-4806-84d9-d53d833786df-c000.snappy.parquet  part-00000-9ac7e010-0a54-475b-a2aa-01a0db9dbb3a-c000.snappy.parquet
part-00000-223dd5da-301f-4478-9d88-9f3b0b57c519-c000.snappy.parquet  part-00000-9ba98a5a-1540-4be1-9da8-8458e01ea7ba-c000.snappy.parquet
part-00000-23f7dadb-a196-47fc-9229-859395261023-c000.snappy.parquet  part-00000-9dc99f95-d79c-4ecd-b24b-9a150618f0fc-c000.snappy.parquet
part-00000-250b4963-69a4-446f-9cbe-5a5cf9b3cf5b-c000.snappy.parquet  part-00000-9fec0f4d-dc3e-4913-b642-9c426b72816d-c000.snappy.parquet
part-00000-274895d6-bec0-418a-be21-27d89da2a163-c000.snappy.parquet  part-00000-a03b2e8b-5aa9-4441-bb7a-e88d3e9be79e-c000.snappy.parquet
part-00000-27cd00b6-35b0-447c-908a-0c7261078004-c000.snappy.parquet  part-00000-a0f1fc78-9d9a-4386-a608-2697579416f5-c000.snappy.parquet
part-00000-27e18053-1a10-4659-809f-34c2b5e600ae-c000.snappy.parquet  part-00000-a24a9b75-b885-4343-9158-a1874fb20253-c000.snappy.parquet
part-00000-2f0d7fdd-23f0-4ab5-acbe-d48a8cf09811-c000.snappy.parquet  part-00000-a5d9d0d7-f094-4d9a-ba2c-c67379b7ab58-c000.snappy.parquet
part-00000-30487ba1-467c-4de2-979f-1a1dd1c7345e-c000.snappy.parquet  part-00000-a7eeebc1-bc4c-4399-9fe7-5ae1ab3cea37-c000.snappy.parquet
part-00000-3245e6e3-9dd8-4d71-b1c4-3c24063db3f2-c000.snappy.parquet  part-00000-aa6b07bf-9f21-4d19-beb0-058c8025419c-c000.snappy.parquet
part-00000-34fb317d-7d03-4162-bc41-a0e7fe36e19b-c000.snappy.parquet  part-00000-aa8ddc03-5857-4543-9e9d-a6413331fb7c-c000.snappy.parquet
part-00000-36343c22-8f2c-43f4-acc6-1d770c93d279-c000.snappy.parquet  part-00000-ac65b8ea-a10f-4576-b27a-741ad207648c-c000.snappy.parquet
part-00000-369a5774-3c80-412b-beaf-0cef855d6674-c000.snappy.parquet  part-00000-adcf8129-f00f-432f-95f9-b7e47ce53d24-c000.snappy.parquet
part-00000-36a97257-51fb-44e4-a287-a08f06b2e2de-c000.snappy.parquet  part-00000-af28211a-85e6-466b-8d0f-f0591f283377-c000.snappy.parquet
part-00000-3c3c03b0-a2d1-4c6d-b2aa-b4a2f061804d-c000.snappy.parquet  part-00000-b01e8108-4b51-430d-87ad-820a752245db-c000.snappy.parquet
part-00000-3ebab744-81ea-43dc-a7e9-84c48f1bb734-c000.snappy.parquet  part-00000-b16cfb1b-b6fe-405c-b94f-cdb532aeb5ca-c000.snappy.parquet
part-00000-3ee99ea2-fb75-40f4-a886-7cddf9bfe7d6-c000.snappy.parquet  part-00000-b1a4dd3a-bf9d-4cb5-855c-eef2799f12e5-c000.snappy.parquet
```

Figure IV.3-1 Reported meal portions stored in Parquet files

The Data Gateway harnesses the power of Spark SQL to efficiently analyse the stored data through the use of SQL queries. Some simple queries and their responses are shown in Figure IV.3-2.

```
select distinct name0_family, name0_given0 from patient, observation where patient.id=observation.subject
[["Bond", "James"]]
select extension0_valueString, valueQuantity_value from observation where valueQuantity_value>300
[["supper: Meat stuffed peppers", 333
 ["breakfast: Omelete", 301
 ["breakfast: Eggs", 343
 ["lunch: Fish", 314]]
```

Figure IV.3-2 Querying stored FHIR data

IV.4. ACCESS CONTROL FRAMEWORK

Recall that the access control framework is composed of three main components, all with the primary use of enhancing the functionality of the Data Gateway:

- Identity and Access Management (IAM): to manage users' identities and roles
- Key Management Service (KMS): to securely store the data encryption keys that are used by Modular Parquet encryption
- Access control: to evaluate access control requests and decide whether the Data Gateway should grant access to the data it has retrieved or not

To realize the IAM functionality, we will use a commercially available off-the-shelf component. We assume that the use cases, studied in WP2, already make use of an IAM to identity their users, for example using Amazon Cognito (https://aws.amazon.com/cognito/) or solutions like Keycloack (https://www.keycloak.org). These can then be used in the access control framework as well. This IAM component will be directly integrated into the overall integrated ProTego demo due by M18. Since the IAM component will not be developed by ProTego, it obviously is not part of a YearOne demo.

In the first stage of the project, the key management functionality will be realized by using Hashicorp Vault. If offers an encryption service by encrypting or decrypting 'data encryption keys' using its internal 'key encryption key'. Again, the ProTego efforts in the first half of the project are mainly targeted towards integrating this open source component with the Modular Parquet Encryption engine that is executed in the Data Gateway. Therefore, it does not make sense to demonstrate this KMS component now in an isolated way.

The access control component is used to decide whether the Data Gateway should forward the requested data (or the result of a query) to the application, or whether it should deny access. Therefore, it is obviously tightly coupled to the data access (i.e. the functionality of the Data Gateway) itself. Although most of the work is targeted towards integrating the overall access control architecture, the demo shows some limited functionality in an isolated way. A server instance has been developed that groups the different access control functionalities, discussed in Section III.4. , into a single software package. The installation scripts for the server software work on Ubuntu or Debian but could in theory work on other Unix platforms as well.

The different access control functions – i.e. policy enforcement point, policy decision point, policy access point and credential validation service – have been implemented in Java, based on the Common Open Policy Service (COPS) reference implementation for authorization components. Browser-based clients can login to this server using JavaScript and Ajax. Depending on the type of login, one can emulate a data request (i.e. trigger security permissions to be evaluated) or manage the access control system (update the users and their roles, or security permissions).

IV.5. NETWORK SLICING

In order to validate the actual outcome of slicing solutions, we performed three different experiments: two for performance isolation and one for privacy isolation. The experiments for network slicing were deployed in the IMEC facilities to enable easy recovery of the system in case of network misconfiguration - which could affect the remote access to the devices.

IV.5.1. Performance Isolation

The experiment for performance isolation has the objective to validate the expected software behaviour for different protocols such as TCP and UDP. The validation for these two protocols will deepen the understanding of each kind of hospital applications, which will avail from the WiFi network slicing. It is already known before our experiments that the performance isolation is restricted to the client downlink. The restriction is due to the distributed characteristics of WiFi. This causes every device to dispute the spectrum to send data, avoiding EmPOWER to have full control on the data transmissions. The performance isolation tests were deployed using iPerf3 as a traffic generator through the WiFi link, as presented in the demo setup section. The bitrate of the traffic generator was set to 20Mbps, and the traffic was generated from the server to the client (client downlink). The experiment starts with both slices having the same amount of airtime, distributing the traffic equally, and at the 90th second, we set the airtime of slice 1 to a third of the airtime of slice 2. The result of the experiments is presented in Figure IV.5-1 and Figure IV.5-2.

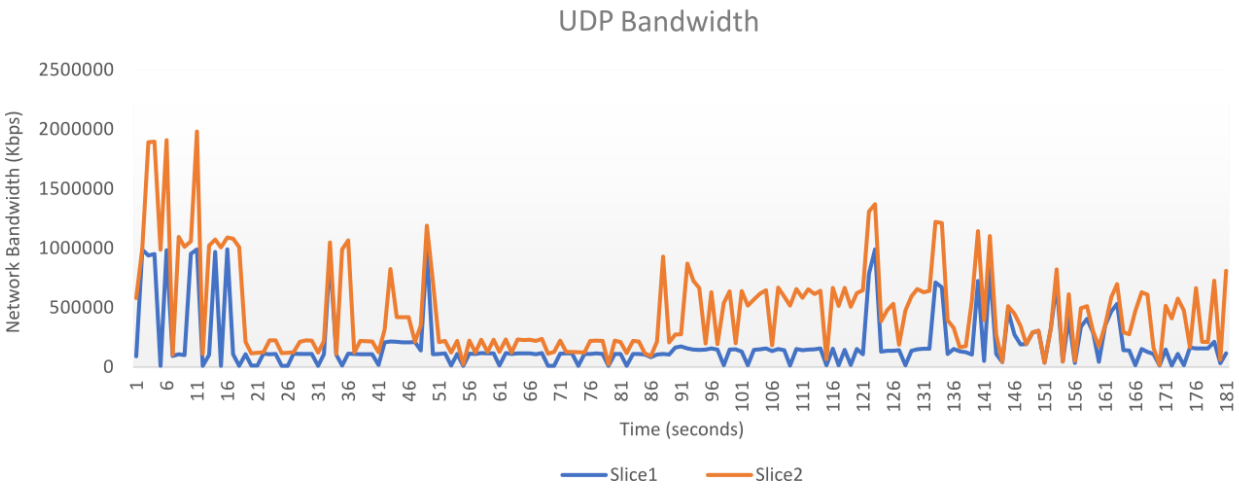

Figure IV.5-1 Performance isolation experiment with downlink UDP traffic through wireless link

As can be seen in Figure IV.5-1, in general, the traffic throughput pattern has a normal throughput variation due the interference of wireless networks. In the first half of the experiment, it is possible to see that the traffic for both slices maintain the same throughput, confirming the expected behaviour of the network given the airtime configuration. Furthermore, after the 90th second, slice 1 was changed to 1/3 of the airtime of slice 2, and the result of this change can be explicitly seen in the second half of the chart. Besides some network spikes due wireless connectivity variation, the throughput of slice 1 remains at 1/3 of the network throughput of the 2nd slice, demonstrating that the UDP downlink for application can be customized to prioritize hospital services.

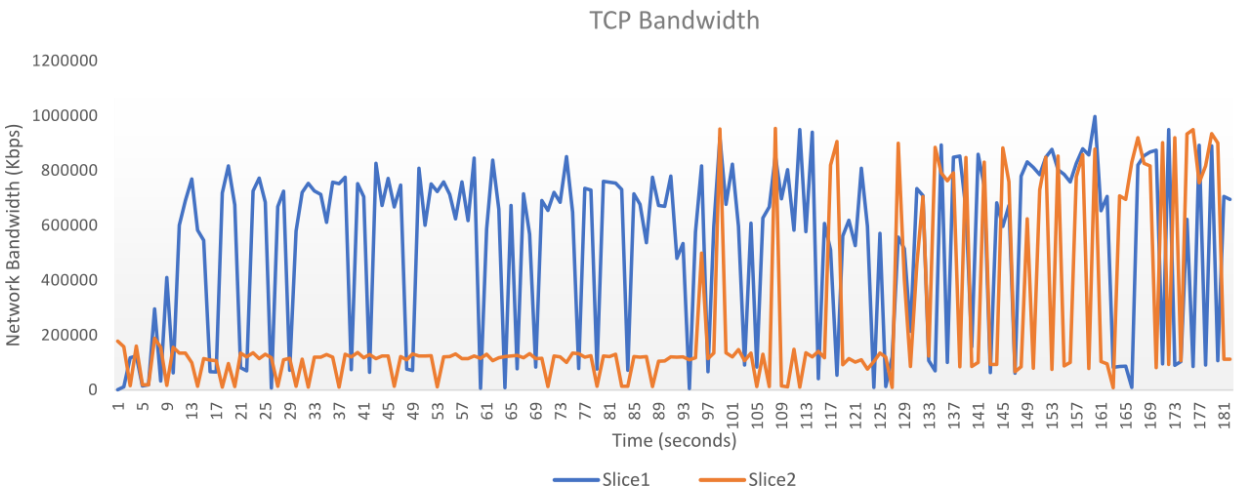

Figure IV.5-2 Performance isolation experiment with downlink TCP traffic through wireless link

The experiment for the performance isolation of TCP traffic begins with different airtimes for the slices, and after the 90th second, the airtimes were equally distributed between the slices, as can be seen in Figure IV.5-2. The distribution of airtime started with slice 2 having ¼ of the airtime of the slice 1. After we changed the airtime for slice 2 to the same value as slice 1, we can see that the throughput of the 2nd slice does not adapt immediately, having an adaptation time before consuming the same amount of bandwidth as slice 1. This delay is caused by the normal behaviour of the TCP protocol, which increases the amount of data sent gradually. After assessing the outcome of using performance isolation with downlink TCP traffic, we conclude that any web-based application where most of the time the downlink traffic is greater than the uplink traffic will avail from the performance isolation provided by 5G-EmPOWER and the ProTego network slicing tool.

IV.5.2. Privacy Isolation

Privacy isolation is characterized by the separation and encryption of a specific type of traffic. This separation can be done, for instance, by a client or service. In ProTego, we have the interest to apply the encryption in the hospital services to secure the patients and staff confidential data. The application data is encrypted when passing through a virtual secure network interface that utilizes VXLAN to create a virtual network and IPSEC as a secure protocol. Therefore, to assess the behaviour of the proposed privacy isolation tool, we deployed two iPerf3 clients and two iPerf3 servers. The first iPerf3 connection uses UDP protocol, and the second uses TCP. The traffic in this experiment is separated among the interfaces using the protocol as a service identifier, but later can be enhanced to be separated by destination port, for example. The default virtual network interface is tagged with the VXLAN Network Identifier (VNI) 10, and the secure network slice is tagged with the VNI 20. We would like to highlight that in this experiment, the traffic steering among different virtual interfaces was assessed. However, the encryption of the secure interface is still pending and will require further investigation.

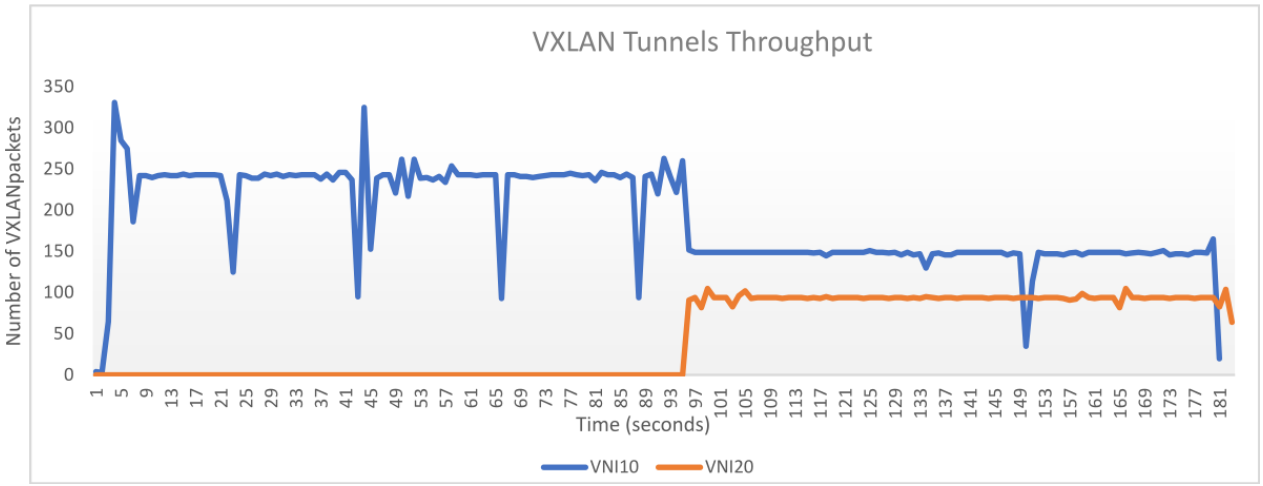

Figure IV.5-3 VXLAN tunnel throughput per VXLAN Network Identifier (VNI)

As can be seen in Figure IV.5-3, on second 0, we started the two traffic generators. The data were collected by sniffing the physical network interface where both traffic flows, secure and not secure, pass through. At first, the secure network slice was not created; therefore, all the traffic from both applications was passing through the same default virtual interface with the VNI 10. After the 90th second, the second slice was created, and the UDP traffic was redirected to the secure virtual interface, being tagged with the VNI 20.

The deployment of the network rule for traffic steering was proven to be effective for our purposes, having less than 5 seconds of delay to change the traffic from the default interface to the secure. Therefore, we will enhance this tool, so it will be possible to create several network slices with different security protocols and techniques to isolate attacks and enhance hospital network security.

IV.6. CONTINUOUS AUTHENTICATION

This Android app has been provided to the hospitals to be tested in real environments allowing to receive feedback. As mentioned before the installation process requires specific permissions like an accessibility service. The usage of an accessibility services offers us the new methods to monitor the continuous authentication of the user independent from the keyboard.

In order to talk about ProTego-CA results it is necessary to understand what information is collected. The current version identifies the user based on the keyboard typing behaviour. It relies on the results of preliminary research of continuous authentication, in which we have found that the time a person is pressing a key when typing a character is different from others'.

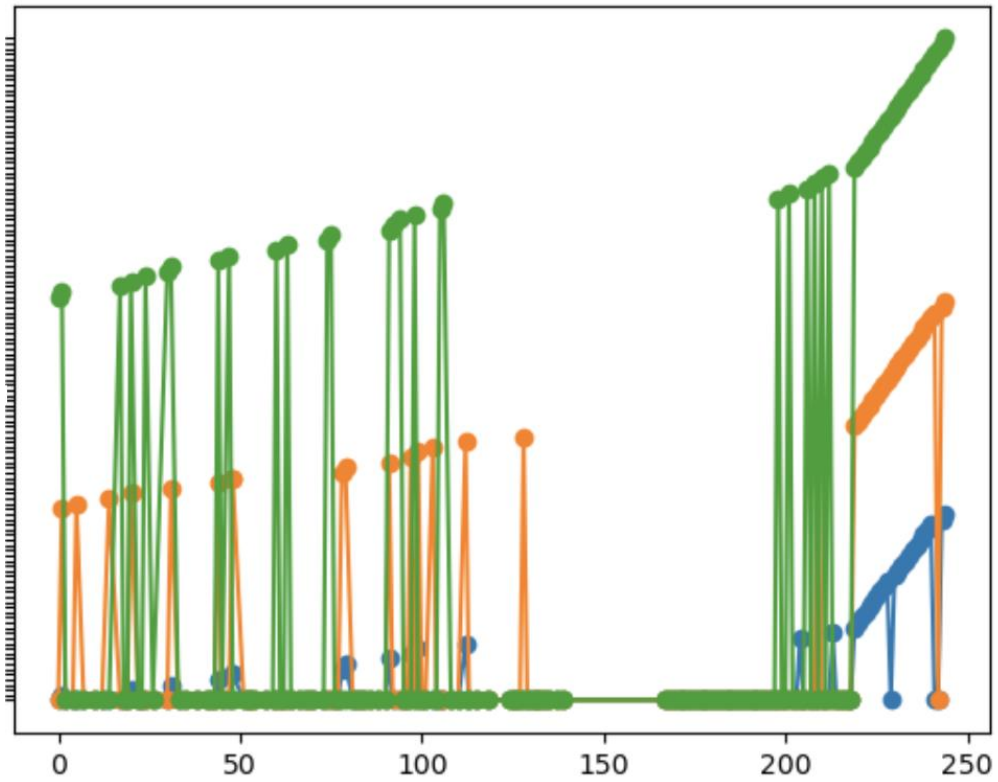

Figure IV.6-1 Three different user metrics obtained from keyboard usage

Also in this version of the application is it possible to show what values and metrics are being stored by the application. This information can be seen and deleted from the application (see Figure IV.6-2).

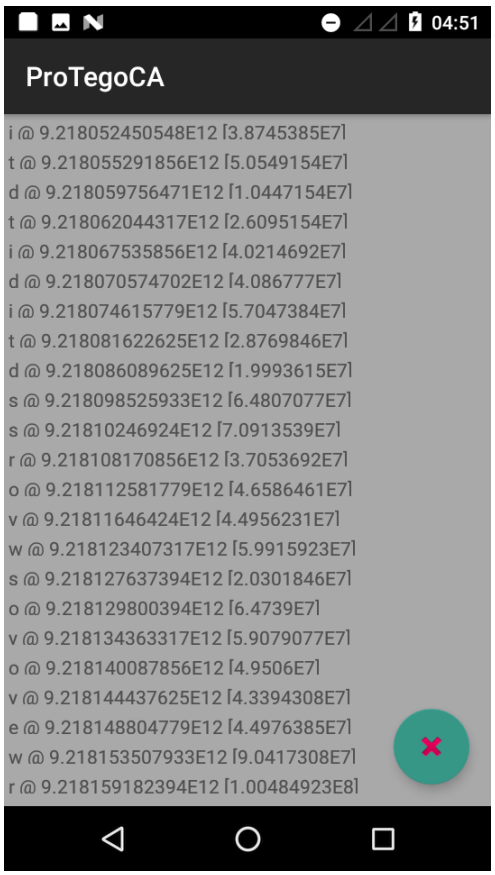

Figure IV.6-2 ProTego CA metrics

Finally, all these results and algorithms can be visually tested checking the ProTego CA status bar (see Figure III.6-3). During the demonstration it is possible to use the mobile phone from different users and check how the status bar change between the users.

V. CONCLUSION

The trials and results described in this document reveal that all the components are currently working even though the functionality is not yet complete. They will continue being developed to implement all the functionality, as well as the integration of the whole toolkit.

Integrating the demos is not planned until the Task 7.3 Experiments iteration and demos. However, this does not mean there has been little done in this area. The main aims of the integration are to utilise open source tools to integrate the other components, to form a platform. This platform is then deployed on a development server for access by the other technical partners.

This work will utilise Docker as a container engine. This allows code to run in separate environments and provides many additional benefits for deployment, maintenance, and development. Kubernetes is used as a container orchestrator. To manage and organise these containers with an extra layer of abstraction as Kubernetes services. Istio will act as a service mesh. Linking the Kubernetes services together and allowing the addition of sidecars to each service. Finally, this will all be managed by Rancher, a complete software stack for maintaining containers with multiple Kubernetes clusters.

The work has started in installing this open source tooling in the development environment. In fact, Rancher and its included Kubernetes instance are available on the PRE-Production environment. The next step will be to enable Kubernetes and install “vanilla” versions of the other components. This will allow a baseline platform that can be used for development. The main hurdle in this process is likely to be “Network Slicing” which will need to be integrated at a lower level within the stack. It will need to be integrated with the Flannel protocol used by Kubernetes to manage the network connections between clusters and pods. Knowing this up front allows us to focus efforts on this aspect but also work in parallel to achieve a baseline platform for the intermediary Demo of Task 7.3.

In addition to the platform that will support the toolkit, plans have already been made to work on the integration of the tools themselves, identifying dependencies and requirements, setting milestones, and the necessary meetings until the first iteration is complete. It is estimated that for M18 the following demos can be performed with the integrated toolkit:

- Legitimate data access: authorized user accessing data for which permission has been granted.
- User role unauthorized: authorized user trying to access data for which permission has not been granted.
- Stealing a device: an authorized device is stolen and device usage reveals the user is not the same as usual.
- Tampering encrypted data files: an unauthorized user tries to modify encrypted data and the platform detects the change.
- Eavesdropping encrypted data: an unauthorized user tries to read data that is encrypted.

For M15 all the requirements would be met and for M17 there would be a face-to-face meeting to review integration status. After this meeting final adjustments for the first integration iteration will be performed so that by M18 everything is ready.

VI. REFERENCES AND INTERNET LINKS

VI.1. Internet Links

[1] https://www.kimsufi.com/

[2] https://www.ovh.com/

[3] https://www.elastic.co/

[4] https://kafka.apache.org/

[5] https://spark.apache.org/

[6] https://wazuh.com/

[7] https://www.ossec.net/

[8] http://www.openvas.org/


ProTego