Horizon 2020 Programme

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

DATA-PROTECTION TOOLKIT REDUCING RISKS IN HOSPITALS AND CARE CENTERS

Project Nº 826284

ProTego

D4.3 Release of the final cybersecurity risk assessment tools

Responsible:	University of Southampton, IT Innovation Centre
Contributors:	IT Innovation, Inetum (GFI before January 2021)
Document Reference:	D4.3 Release of the final cybersecurity risk assessment tools
Dissemination Level:	Public
Version:	Version 1.0
Date:	30/06/2021

Executive summary

This deliverable is a report accompanying the final release of the ProTego cybersecurity risk assessment tools. This report describes advances from D4.2 in the final year of the ProTego project. The key advances are summarised as follows.

- We have complete risk models for both of the two end-user scenarios, Pocket EHR and FoodCoach. We have shown how the systems are modelled in the SSM toolkit, the application of initial controls and the resulting static risk level.
- We have investigated collaborative, cross-organisation risk modelling via an information hiding approach where different stakeholders can focus on their parts of the system – here a key example split is between application and infrastructure, and each can be hidden from the others.
- We have extended the dynamic risk assessment concept via extensions to the Run Time microservice, which provides a REST API for vulnerabilities detected at runtime, risk recalculation to determine the resulting risk level due to the vulnerabilities, and recommendations to reduce the risk level.
- We have provided additional support for web applications via support for OWASP ZAP (Zed Attack Proxy). The support is achieved through mappings between ZAP alerts and risk model updates.
- We have extended the domain model knowledge base to support relevant aspects such as contextualisation of e.g. mobile (BYOD) devices, so that their location is taken into account in the risk modelling, theft of devices and network localisation.
- We have extended the functionality of the SIEM to supply the SSM vulnerability information of web applications by integrating OWASP ZAP in the architecture.
- We have developed a Deep Learning mechanism in the SIEM to improve detection capabilities of possible attacks.

Overall, the key advance in WP4 from ProTego is the transition from static risk modelling to dynamic, runtime risk modelling, so that risks can be determined in response to security alerts and reports of vulnerabilities in real time. The key benefit of this to practitioners is that their system is constantly monitored and when risk events occur, they are detected and assessed in real time, together with recommendations regarding how to mitigate any raised risk.

Contributors Table

DOCUMENT SECTION	AUTHOR(S)	REVIEWER(S)
I.	Steve Taylor (UOS)	Eliot Salant (IBM), Antonio Gamito, Luis Carrascal, Fernando Rendon (Inetum)
II.	Steve Taylor, Mike Surridge, Panos Melas, Ken Meacham (UOS)	Eliot Salant (IBM), Antonio Gamito, Luis Carrascal, Fernando Rendon (Inetum)
III.	Steve Taylor, Panos Melas, Mike Surridge, Sam Senior (UOS)	Seyed Farhad Aghili (KU Leuven), Antonio Gamito, Luis Carrascal, Fernando Rendon (Inetum)
IV.	Mike Surridge (UOS)	Esteban Municio (IMEC), Antonio Gamito, Luis Carrascal, Fernando Rendon (Inetum)
V.	Luis Carrascal, Fernando Rendon (Inetum)	Carlos Cilleruelo (UAH)
VI.	Steve Taylor (UOS), Luis Carrascal (Inetum)	Carlos Cilleruelo (UAH)

Table of Contents

Contents

Table of Figures

List of Tables

Table of Acronyms and Definitions

Acronym / Definition	Explanation
AER	Average Error Rate
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Program Interface
APT	Advanced Persistent Threat
BYOD	Bring Your Own Device
CAPEC	Common Attack Pattern Enumeration and Classification
CRUD	Create, Read, Update, Delete
CWE	Common Weakness Enumeration
DDoS	Distributed Denial of Service
DMZ	Demilitarized Zone
DNS	Domain Name System
DoS	Denial of Service
EC	European Commission
GDPR	General Data Protection Regulation
IoC	Indicator of Compromise
IoT	Internet of Things
ISO	International Organization for Standardization
IT	Information Technology
KB	Knowledge Base(d)
LAN	Local Area Network
ML	Machine Learning
MLP	Multi-Layer Perceptron
MS	Marina Salud
OSR	Ospedale San Raffaele
OWASP	Open Web Application Security Project
PC	Personal Computer
RNN	Recurrent Neural Network
SIEM	Security Information and Event Management
SSM	System Security Modeller
UI	User Interface
UK	United Kingdom
URL	Uniform Resource Locator
WAN	Wide Area Network
WASC	Web Application Security Consortium
WLAN	Wireless Local Area Network
WP	Work Package
ZAP	Zed Attack Proxy

I. INTRODUCTION

This deliverable follows from D4.2 and describes updates in the final year of WP4, from June 2020 to June 2021. This deliverable is a report accompanying the final release of the ProTego cybersecurity risk assessment tools, and as such describes advances from D4.2 in the final year of the ProTego project.

This report is structured as follows. Next, is a section describing advances in static risk modelling that describes the risk model building for the two ProTego end-user cases, FoodCoach and PocketEHR, plus enhancements to support collaborative, cross-organisation risk modelling. This is followed by a section describing extensions to support dynamic risk assessments, where the updated concept of dynamic risk assessment is described, along with updates to the ProTego Run Time Microservice, mappings from OWASP ZAP to risk models and runtime recommendations for controls to mitigate risks identified by SIEM. Next, updates to the domain model are described covering local contexts, device theft and network localisation. Updates to the SIEM are then described, covering an AI & ML approach to threat detection. Finally, the deliverable is summarised in the conclusions.

II. STATIC RISK MODELLING

This section describes the work undertaken to support static risk modelling since D4.2. First the underpinning concepts are described to serve as a basis for subsequent discussion of recent work. This is followed by two sections describing the static risk modelling of the two ProTego end-user scenarios, FoodCoach and Pocket EHR. Finally, extensions to the System Security Modeller toolkit to support cross-organisation collaborative risk modelling are described.

II.1. Underpinning Work

The process for static risk modelling is described in [1] and follows a standardised risk assessment approach described in [2], where the operational system's assets and relationships between them are identified, and a system risk model is constructed based on them. Threats are automatically identified, and from these threats, risks are determined.

The risk modelling process is supported by a decision support toolkit named the "System Security Modeller" (SSM) toolkit, that has been developed at University of Southampton over 7+ years. The SSM has the following concepts.

- ***Asset***: something that has value within the system
    - it contributes to the purpose of the system, there is a cost if that contribution is lost
    - Assets have different types, e.g. ICT hardware, networking, software processes, data, people, physical spaces
- ***Relation***: connectors between Assets in models of operational systems
- ***Misbehaviour***: an adverse effect in Asset behaviour or status caused by a Threat attacking the Asset
    - Risk is determined for each Misbehaviour
- ***Threat***: a situation or event that could cause Misbehaviour
    - Threat effects: the Misbehaviours caused by a Threat
    - Secondary Threat: a Threat representing a mechanism for the propagation of effects
- ***Risk***: the severity of a Misbehaviour
    - Based on the ***likelihood*** of its occurrence (calculated by the SSM and determined by the likelihood of Threats causing the Misbehaviour) combined with the ***impact*** it would have (set by the user based on their use case, the asset's value etc)
    - Overall Risk level is highest of all Misbehaviour Risks in the model
    - Risk levels are measured using a Likert scale of: Very High, High, Medium, Low, Very Low.
- ***Matching Pattern***: Specification of Assets and Relations between them that triggers Threats
- ***Trustworthiness***: propensity of an Asset to avoid/resist Threats.
    - An Asset's Trustworthiness affects the likelihood of successful attack of that Asset by Threats – the higher the Trustworthiness on an Asset, the lower the likelihood of relevant Threats on that Asset.
    - Asset types have multiple measures of Trustworthiness, known as "Trustworthiness Attributes" (TWAs). These represent resistance capabilities to different types of threat.
- ***Control***: a security measure protecting an Asset that helps it resist Threats
- ***Control Strategy***: a combination of Controls that address one or more Threats

The concepts are related as shown in Figure II-1.

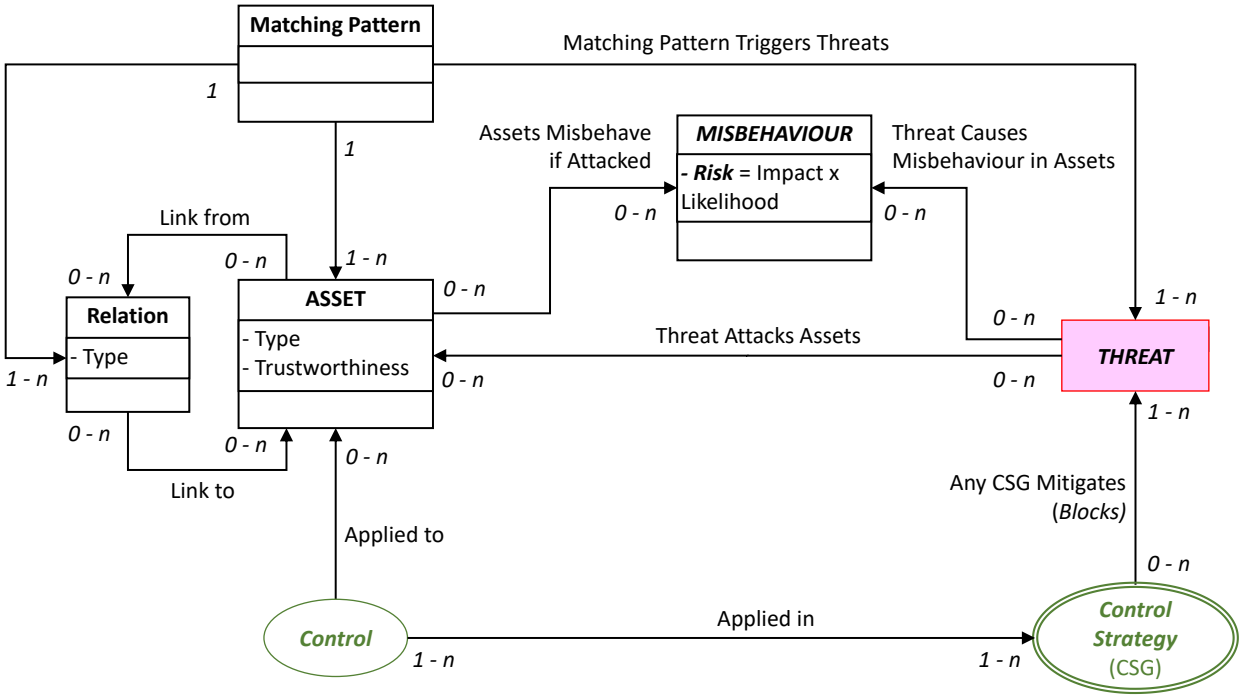


Figure II-1: SSM Concepts

An overview of the SSM toolkit in use is shown in Figure II-1. Expert risk management and threat knowledge is captured in a “Domain Model” that is bundled with the SSM toolkit. The Domain Model contains patterns of assets of different types (e.g. ICT components, processes, data, stakeholders and physical spaces) and relationships, threats that pertain to these patterns, misbehaviours (adverse effects) that occur on the assets if the threats are able to affect the assets, and controls that can block the threats and thus mitigate the risks. Users of the toolkit build “System Models” that describe their socio-technical system in terms of assets and relationships, using the same asset types as encoded in the Domain Model. The SSM matches asset patterns between domain and system models, and from this identifies applicable threats, which determine adverse effects on assets and from this it can determine risks to assets in the system. The SSM can also suggest controls to mitigate threats; which, if applied, will lower risk levels.

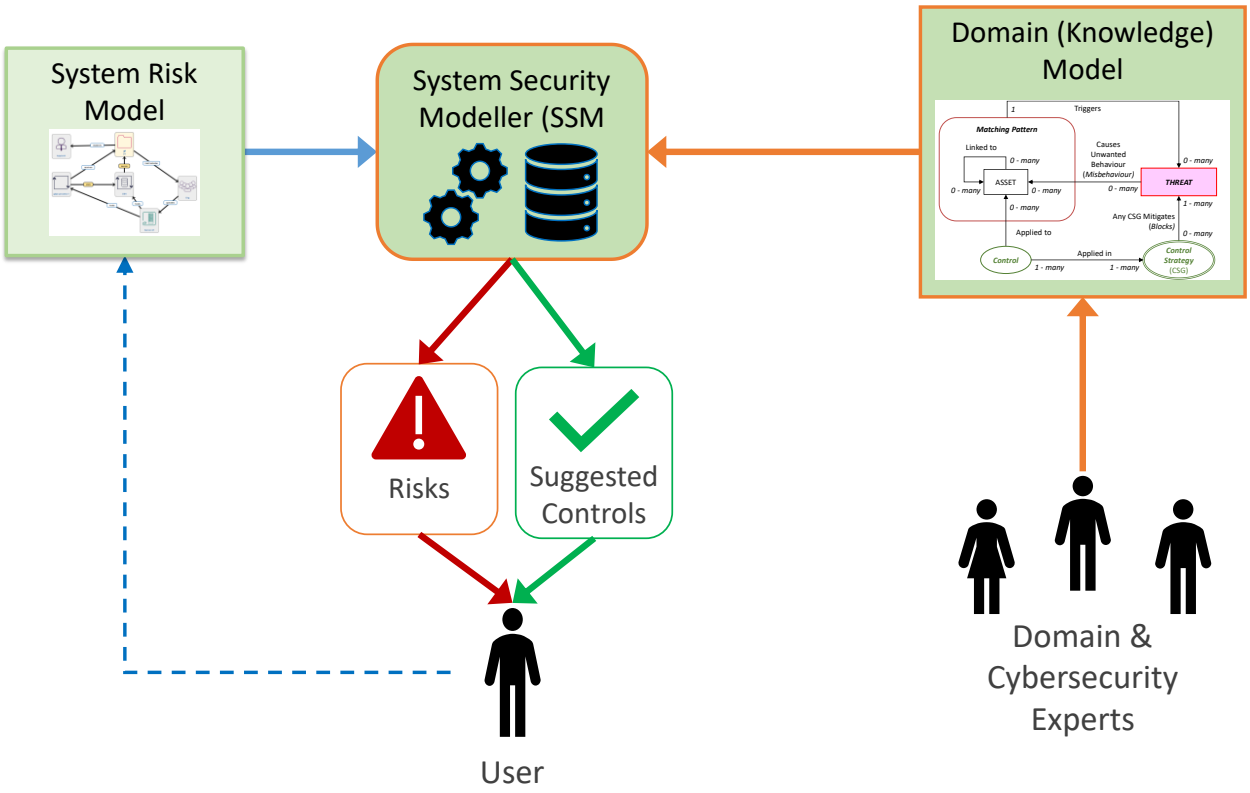


Figure II-2: Static Risk Modelling Approach

A screenshot of the toolkit's User Interface is shown in Figure II-3.

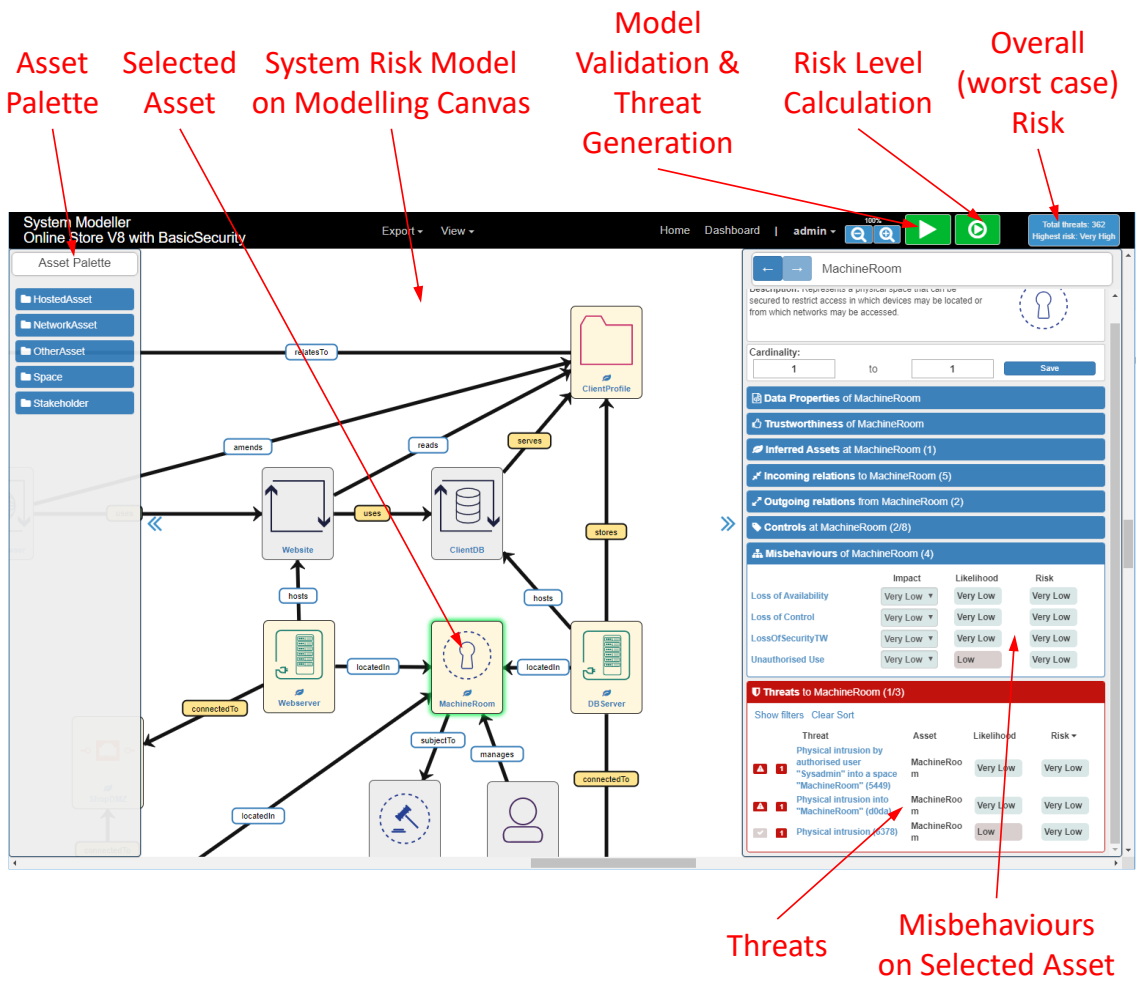


Figure II-3: System Security Modeller User Interface

The overall process for using the SSM for design-time risk modelling is as follows.

1. Identify system assets and relationships and build a risk model by dragging and dropping assets in the modelling canvas and connecting them up.
2. Press the “Model Validation” button (green button with an arrow top right of Figure II-3) to check the risk model and autogenerate hidden (inferred) assets, asset properties, and threats.
3. Specify input impact levels for different misbehaviours on assets in the misbehaviours panel at the centre right of the UI in Figure II-3.
4. Add security measures (controls) already specified or implemented.
5. Run risk calculation to determine risk levels (green button with a circle, top right of Figure II-3).
6. If the risk levels are not acceptable, add more security measures using recommendations provided by SSM and go to step (5), or go to back to step (1) and change the design (i.e. assets and relationships). The threats that cause the misbehaviours can be explored and control strategies are recommended to block the threats. These controls can be applied to assets within the system and the risk calculation rerun to determine the remaining misbehaviours and the threats that cause them, in an iterative way until the overall risk level is brought down to an acceptable level. The application of a control to an asset can contribute to addressing multiple threats, and in turn can prevent threat propagation. Hence, the number of high or very high-level threats can be reduced by the application of a moderate number of controls.

ProTego's two end-user scenarios, FoodCoach and Pocket EHR, are described next with commentary describing their construction, threats, misbehaviours and controls to illustrate the process of static risk modelling.

II.2. FoodCoach Scenario Modelling

II.2.1. Risk Model Description

The starting point is a model of the application. This was created using drag and drop construction in the canvas, to produce the diagram as shown in Figure II-4. For ease of reading, zoomed-in views of the left and right sides of the model are shown in *Figure II-5* and Figure II-6 respectively (with slight overlaps).

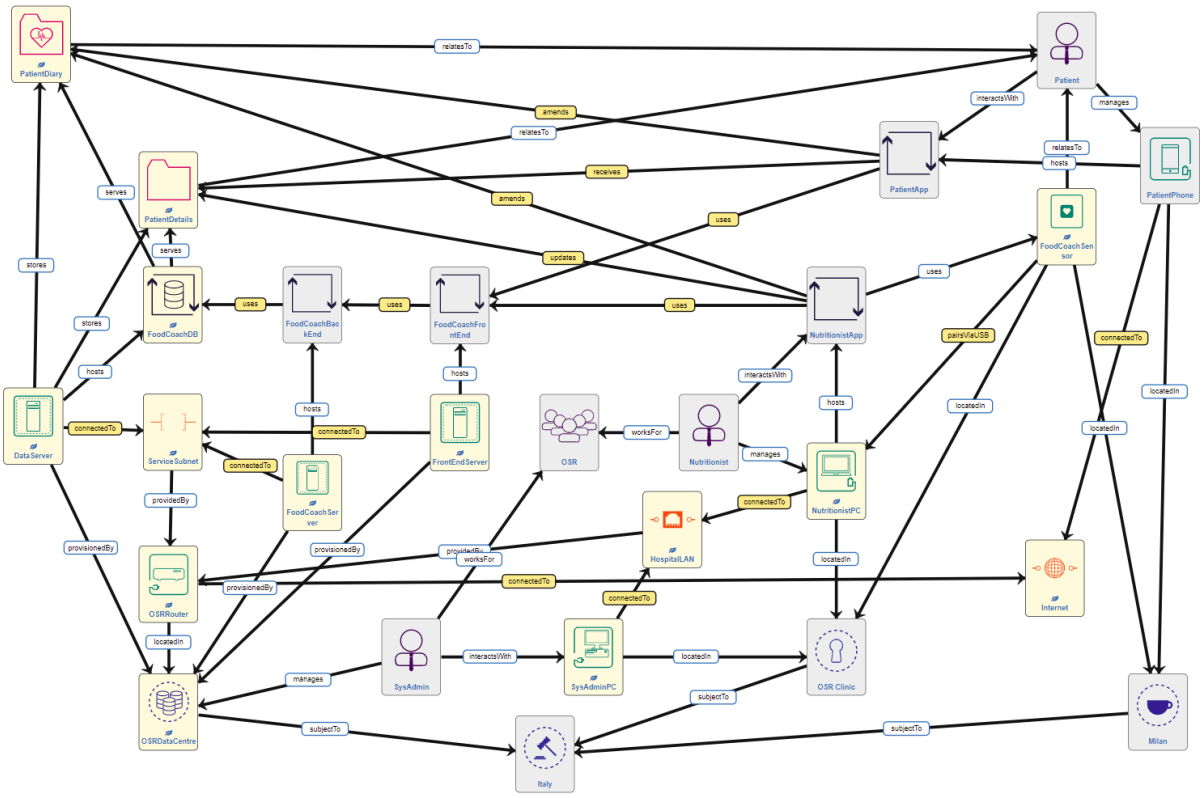


Figure II-4: FoodCoach Scenario

On the left is the OSR data centre, connected via a router to the Hospital LAN and the Internet. In the Data Centre we have three VMs hosting the FoodCoach front end, back end and database. They are managed by a system administrator who uses a PC connected to the hospital LAN. The network connecting the VMs is a software defined network slice provided by the router.

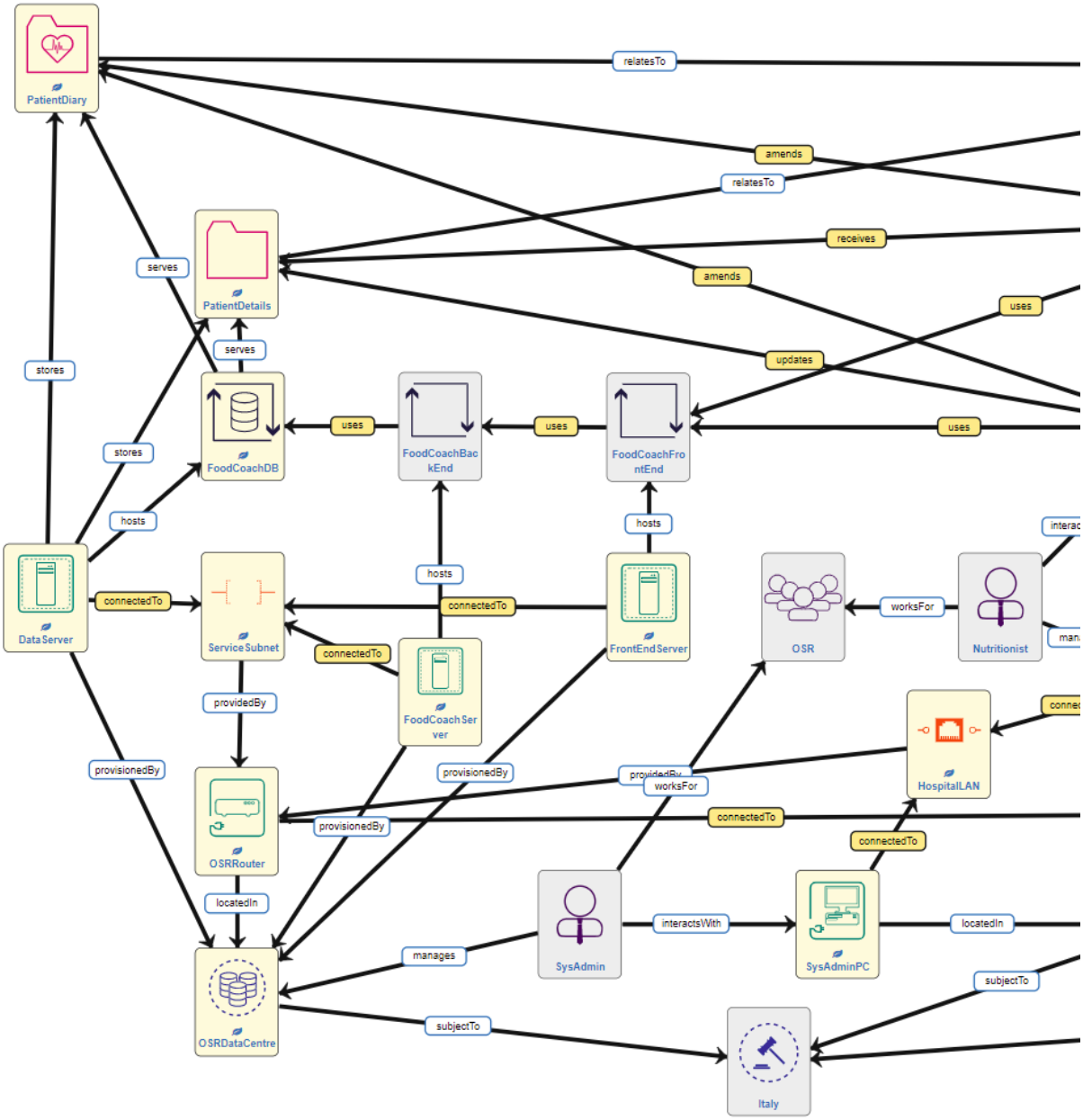


Figure II-5: FoodCoach Scenario (Left Side)

In the middle is the Nutritionist, using a PC connected to the Hospital LAN, running an application they use to connect to the FoodCoach front end service and access the patient data.

Top right is the Patient, who has a smartphone running an app they can use to access the FoodCoach front end to view and update their data and an IoT sensor. For security reasons, the IoT sensor is not connected to any public networks so data must be downloaded over a USB connection.

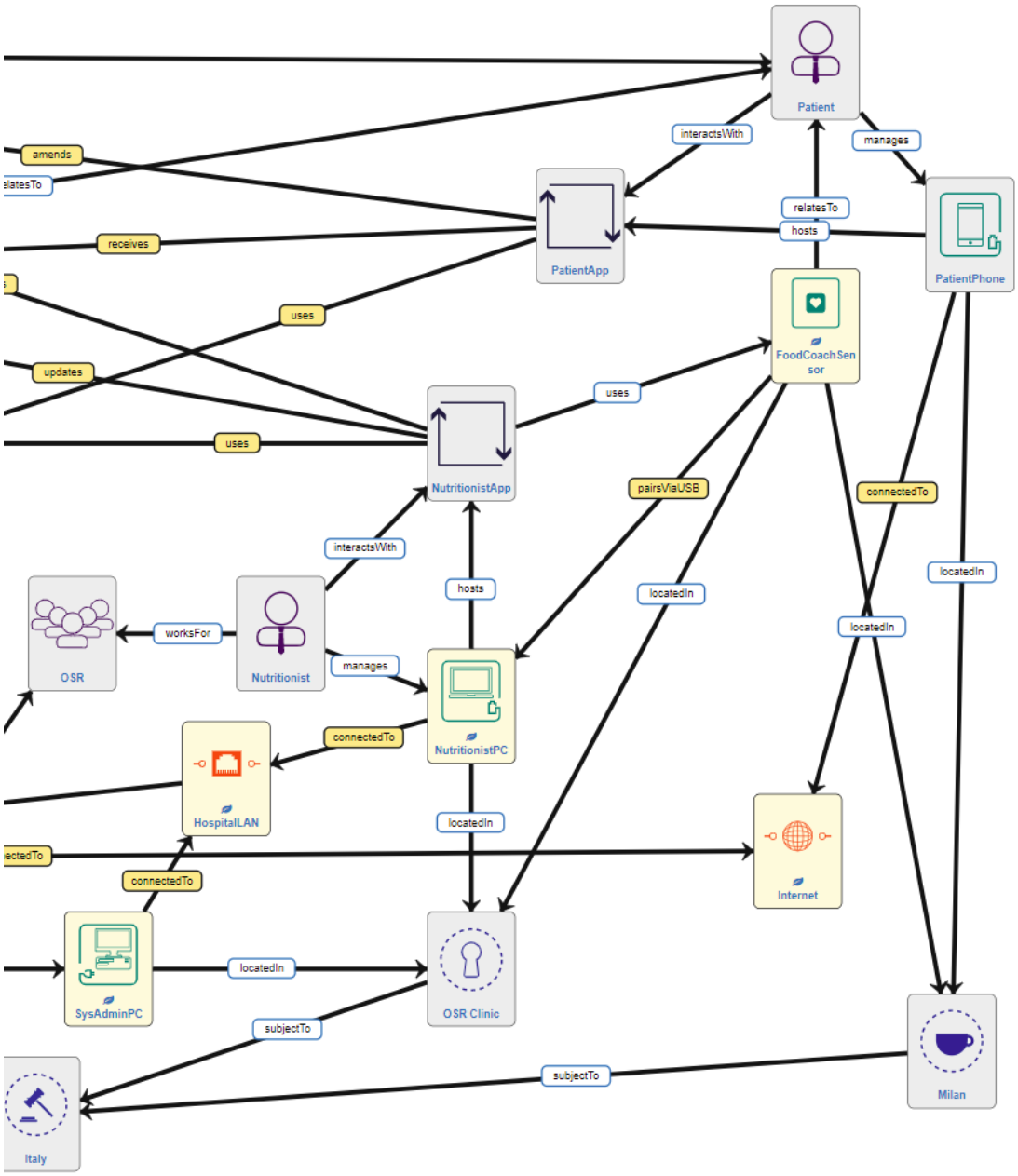


Figure II-6: FoodCoach Scenario (Right Side)

In a real network it is likely that several distinct routers would be used to provide the network slice and the connections to the Hospital LAN and Internet, but the demo uses a simpler setup.

II.2.2. Existing Known Controls

The security measures included in this model are:

- Firewall rules restricting communication via the router from more public networks to more private networks, or into devices (other than between the specified clients and services).
- Authentication using passwords for all authorised users of devices, including the sysadmin when remotely accessing the virtual servers.
- Software patching on all devices, and anti-virus on client devices.
- The FoodCoach front end also has X509 certification, access control based on authentication using a password, and TLS for communication with both clients.

II.2.3. Initial Misbehaviour Impact Levels

The impact of potential misbehaviours was then determined. For the patient diary, loss of confidentiality or integrity are rated as high impact, loss of availability medium impact, and loss of timeliness low impact (these are default values for health data). For the patient registration details, which include contact information and so forth but not health data, the default levels are all very low, so these were overridden: confidentiality and integrity breaches are still rated as high impact, but availability for this data is low impact, and for timeliness the impact is very low. Other impact levels above very low are for loss of availability or timeliness in the client apps, which are rated as medium and low impact respectively. The Loss of timeliness misbehaviour means the client doesn't fail but it may provide outdated information or ask the user to try again later.

II.2.4. Risk Assessment and Controls to Reduce Risks

The validation reveals the presence of 1658 potential security threats, each representing a possible step in an attack that may cause an unwanted effect, or an interdependency that could allow effects to propagate in a secondary effect cascade through the system. These threats are the result of the system topology and assets in the system. The number of threats does not change unless the system design changes, but the SSM provides decision support to reduce the likelihood of these threats and therefore the likelihood of misbehaviours that lead to risk.

There are 33 direct threats to the IoT sensor. We would get far more threats to the sensor if it were connected to a network – but because it is isolated, the potential threats all involve physical access. These pose little or no risk, mainly because the device is personal to the Patient, who can therefore be assumed to be taking care of it. The domain model covers these physical as well as digital threats, allowing SSM to analyse attacks using combinations of cyber and physical attack methods.

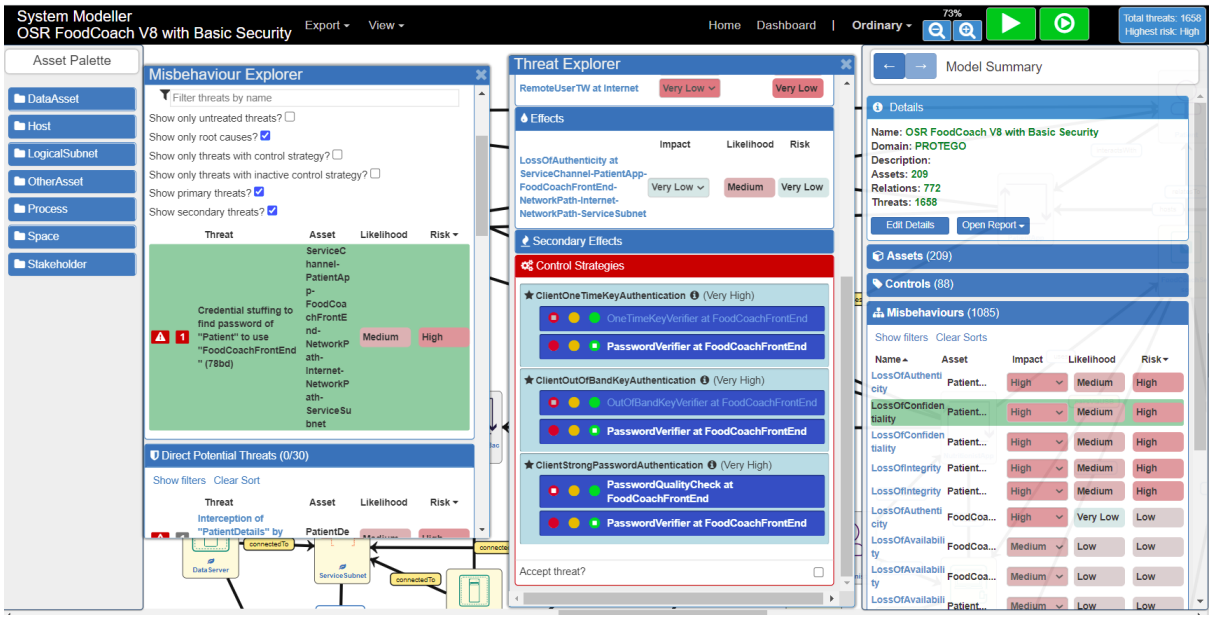


Figure II-7: SSM Display Window

The best way to check risk levels is in the list of misbehaviours on the right of the SSM display window, shown in Figure II-7. Initially, this showed 'high' risk levels for loss of integrity or confidentiality in the patient data. Clicking on a misbehaviour opens a window (the Misbehaviour Explorer at left in Figure II-7) giving more details on the threats that are causing the problem, and we can filter the list of 'all causes' to find just root causes. In this case, the same threat was the starting point for all the problems – a credential stuffing attack against the front end.

Clicking on the threat opens a second window (the Threat Explorer, in the middle of Figure II-7) giving details of the threat and its causes and direct effects (here a loss of authenticity in the

client-service interaction), and possible control strategies. The threat was caused by a simple oversight – the model did not include 'strong' password enforcement at the front-end service, so patients (and nutritionists) would be able to set a password that may be vulnerable to a brute force attack.

A password strength check was added to the spec as a feature 'to be implemented'. That way at the end of the analysis it will be easy to extract a list of new security features to be added, which can be sent back to the developers to be fixed before we deploy the application.

The risk levels were recalculated, and they were all reduced to 'Medium' level risks, and a similar process was followed – i.e. clicking on the misbehaviour again allows us to find the root causes, which was one remote attack exploiting a (yet to be discovered) vulnerability in the FoodCoach front end, and four attacks on the patient's phone using physical access.

The remote attack arises because this is a future risk calculation – i.e. the domain model sets the trustworthiness of software and devices based on the assumption that as-yet undiscovered vulnerabilities will at some stage be found by potential attackers. We can use mitigation strategies to increase the level of confidence, and this has already been done through the specification of a software patching strategy for the FoodCoach front end host device. This strategy means there will be a systematic process for applying security patches regularly and frequently, reducing the chance that an attacker will be able to exploit vulnerabilities before they are addressed by patching. However, it is possible the patching may not always be done before the attack, so there is a residual risk level.

One option would be to use a stringent 'up front' check like penetration testing to actively look for and remove vulnerabilities. This increases the confidence level, but it does not mean there can never be a vulnerability. It therefore makes sense to have a contingency plan for other actions that could be used should a software patch not be available in time.

There are two strategies of this type: to temporarily disable the service, or to leave it running but close the channel through the firewall whereby clients on the Internet (Patient App) can access it. In each case, there is a mitigation strategy representing the contingency plan, and a blocking strategy representing the state of having implemented the contingency plan. Pre-deployment, we will select a mitigation strategy since it is not the intention to pre-configure the system with either the service or the communication channel disabled. The increased risk level from loss of availability is in either case low, as the likelihood of the mitigation being required is itself low (there is a non-negligible but nevertheless low chance of a non-patchable vulnerability being found).

However, it makes sense to see how the functionality of the system is affected if the contingency measure had been applied. To do this we select the blocking control strategies, and find that:

- if the service is disabled, there is a high-level risk to the availability of the Nutritionist App: it would not be able to function.
- if the channel is disabled, this risk does not arise because the Nutritionist can still access the service from inside the hospital, and only medium risks appear due to the fact that the patient cannot keep their diary fully up to date.

On that basis we can select the mitigation strategy for firewall filtering in the event we have a Front-End service that is vulnerable and cannot be patched.

With this mitigation strategy in place to protect the Front-End service, it is time to address the four sources of risk originating at the patient phone. This is a personal device belonging to the patient, so we assume they will look after it, and that is why these attacks are relatively unlikely. However, relying on patients never to allow anyone else to use their phone is not the strongest possible defence, and because the phone is a BYOD controlled by the patient, the hospital cannot assume they will have more than basic digital security measures enabled. The hospital must therefore assume some level of compromise is possible in this BYOD device.

It is possible to trace the attack path from the root cause using the misbehaviour explorer. If we pick a root cause threat and look at its direct effects, we can find a list of threats triggered or enabled by that effect, and start exploring how threats lead from one to the next.

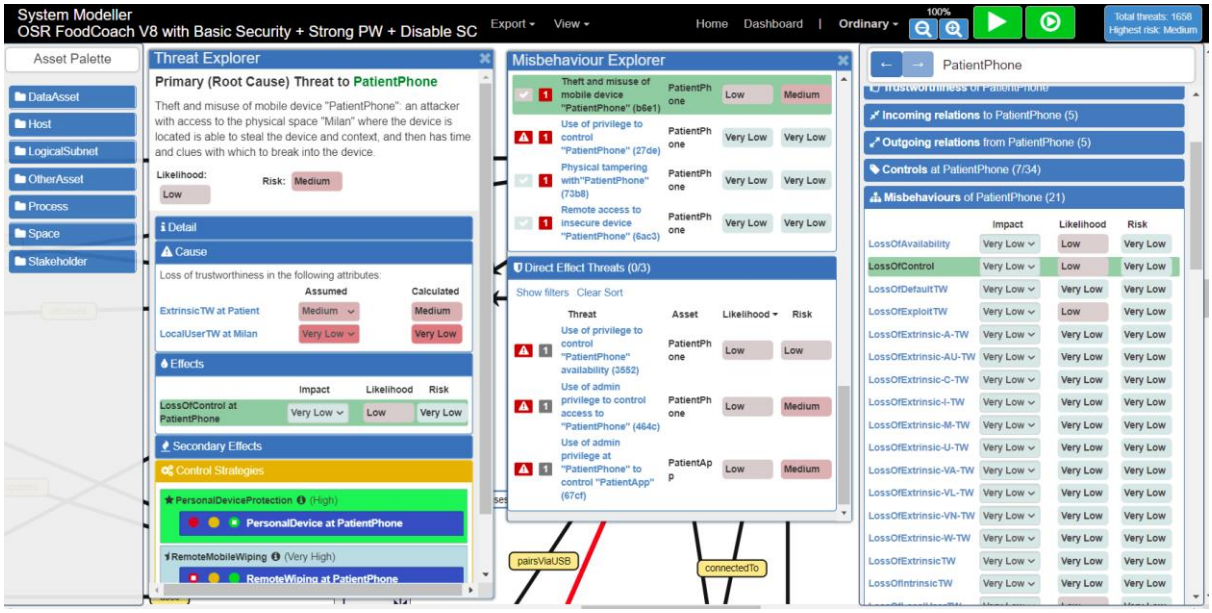


Figure II-8: Attack Path Tracing

Picking the root cause for theft and misuse of the phone, we see in the Threat Explorer that this leads to loss of control over the phone (see Figure II-8). Clicking on that effect, we see in the bottom panel of the Misbehaviour Explorer that this will allow two further abuses:

- allowing others to use the phone which doesn't pose any new risks.
- controlling the patient app which certainly does.

When the app is used to access the Food Coach service it has the rights of the patient, so nothing would prevent it being used to leak patient information or tamper with the Patient Diary. It would also be possible to extract the saved password and use it from other devices, etc.

A direct attack on the patient diary depends on the untrustworthy client App, but also on the authenticity in the channel from the client to the service. This represents authenticity of the user, and it is a factor because in this case, the attacker is effectively impersonating the patient. If we can't prevent an attacker getting their hands on the App, it may be possible to prevent it being used for impersonation. There are three threats for this, all ultimately enabled by the attacker's control of the phone. All can be blocked by using the ProTego security mechanism for continuous authentication. If the patient's client app or their password are compromised, an attacker could access the FoodCoach service with their identity. But if the patient's usage characteristics are registered, these can be used to provide a second verification check that the user really is the patient.

Recalculating risk levels we now see that with Continuous Authentication blocking threats to data from a compromise of the patient's BYOD, the risk levels are now all Low or Very Low.

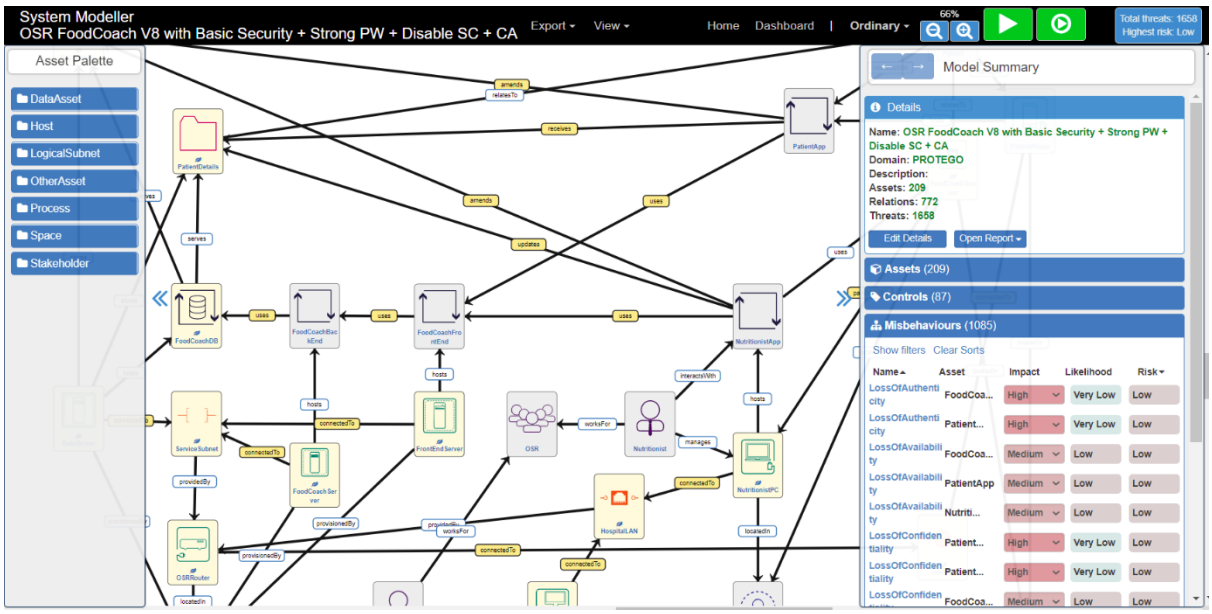


Figure II-9: Result of Continuous Authentication

II.2.5. Summary

This section has described the process of risk modelling for the OSR FoodCoach scenario. The three major controls derived during this analysis can be extracted and marked as 'to be implemented':

- adding a password strength test to the Front-End service
- adding ProTego continuous authentication to the Front-End service
- updating the threat response plan with a contingency action to block access to the Front-End service from the Internet should it become vulnerable.

The last of these includes deciding who is responsible for implementing the changes to the firewall rules, and briefing them on how they will be notified if this is necessary, etc.

II.3. Pocket EHR Scenario Modelling

The Pocket EHR (Pocket Electronic Health Record) Scenario has been developed by Marina Salud with the intention of assessing the security and robustness of an application that heart patients can use on their mobile phone to receive updates from their Cardiologist on their current health, as well as to provide feedback to their Cardiologist.

The scenario features three domains: the hospital, where the Cardiologist can provide input to the patient's EHR; the Patient who accesses the EHR via their mobile phone; and ProTego components hosted in Amazon Web Services (AWS) that serve the EHR to the patient's application and receive updates to it.

II.3.1. Risk Model Description

In order to assess the security and resilience of the ProTego components in the context of the Pocket EHR scenario, WP4 has built an SSM risk model of the scenario in consultation with Marina Salud, following the architecture described in Figure II-10.

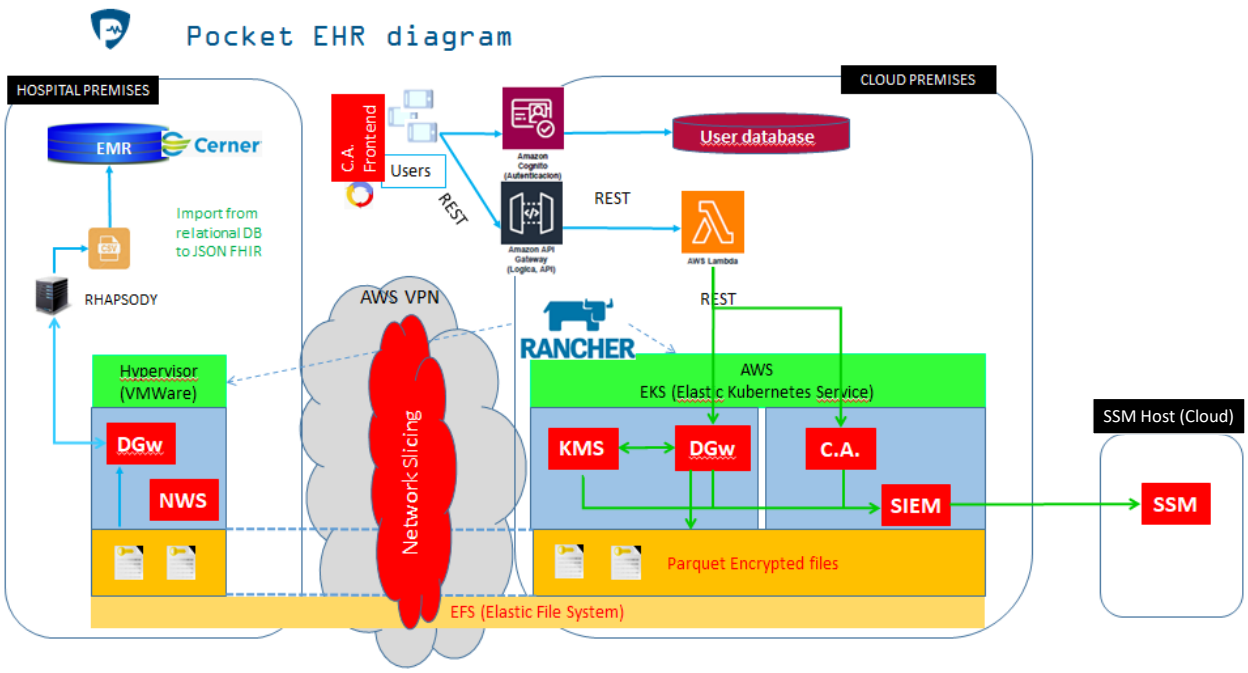


Figure II-10: Pocket EHR Scenario

The risk model follows the architecture and is shown in Figure II-11. The left side of the model covers the components inside the Marina Salud Hospital, the middle covers the Patient running the mobile phone application, and the right side covers the components hosted in AWS.

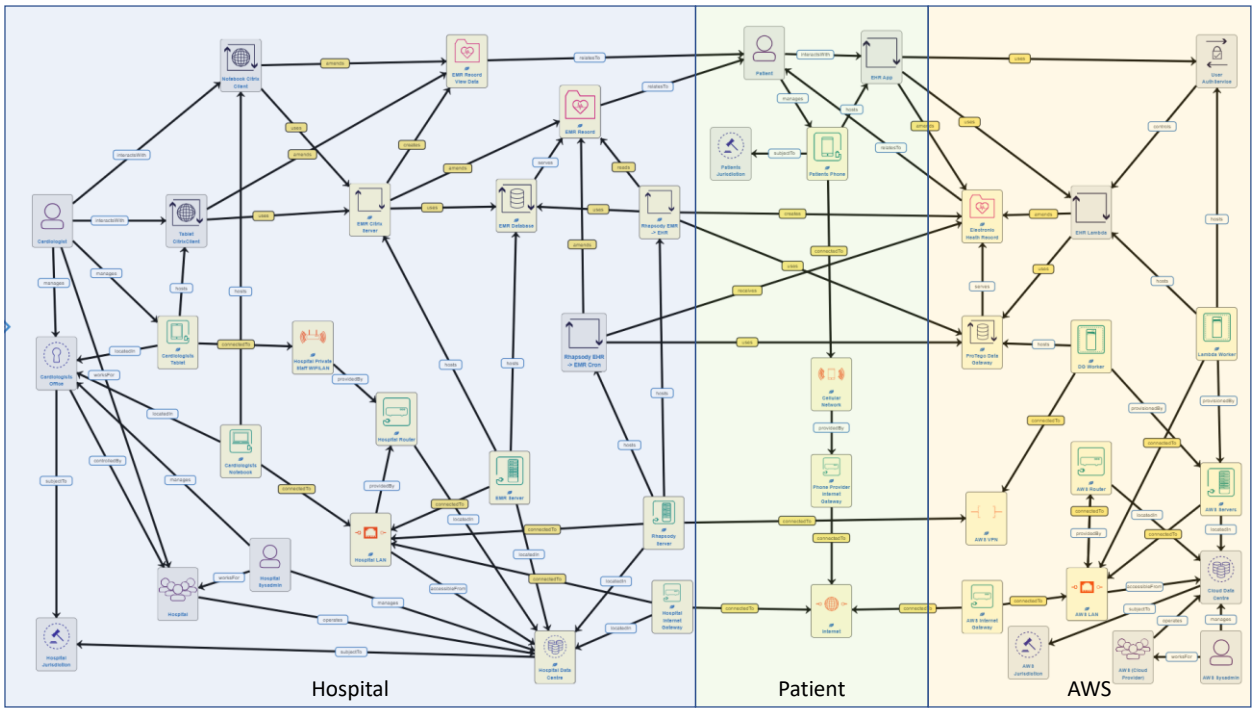


Figure II-11: Risk Model for Pocket EHR Scenario

The hospital components are shown in detail in *Figure II-12*.

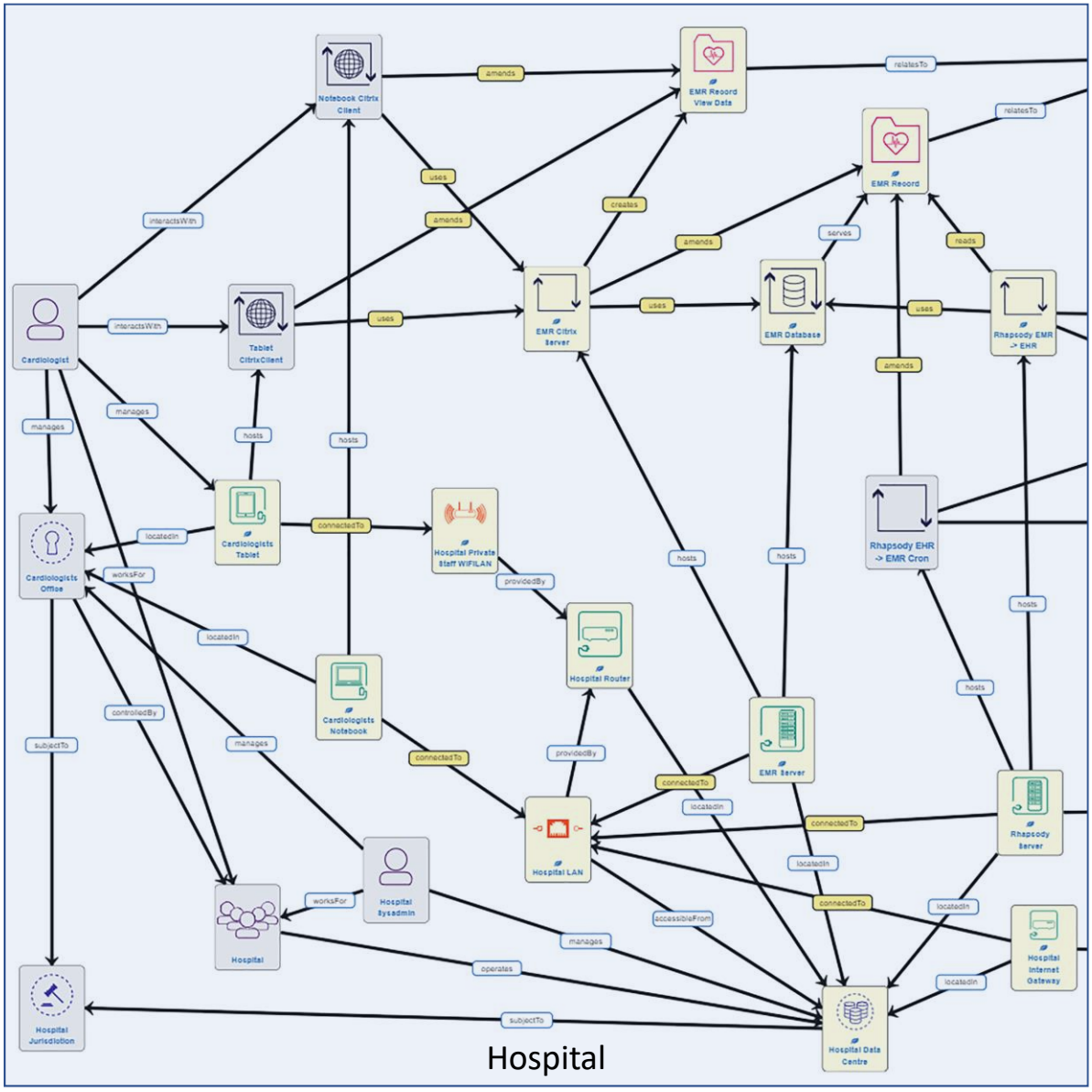


Figure II-12: Hospital Components Detail

The key data element is the EMR (Electronic Medical Record), to the top right. This is served by the EMR Database, and is transformed into EMR Record View Data, which in turn is used by the EMR Citrix Server to display to the clinical staff of the hospital. The EMR Citrix Server is accessed by the Cardiologist via Web Browser clients on either their Notebook or Tablet. The Cardiologist's notebook is located in their office and connected to the hospital's LAN, while their tablet is connected to the hospital's Private Wireless network. The Notebook is the property of the hospital, but hospital staff are permitted to bring their own devices, and this is represented here by the Cardiologist's tablet.

Two processes running in the hospital's Rhapsody server deal with the transformation of the EMR record to the EHR record that is hosted within AWS and back again. The first is Rhapsody EMR ⇨ EHR, which queries the relevant subset of the EMR record and update the EHR record. Patient feedback gathered via the mobile phone application are gathered by a scheduled process named Rhapsody EHR ⇨ EMR Cron.

The hardware supporting this functionality consists of the EMR Server and the Rhapsody Server. The EMR Server is connected to the hospital's wired LAN and runs the EMR Database and the EMR Citrix Server. The Rhapsody Server runs the two Rhapsody processes and is connected to

a VPN that represents a shared private network between the hospital and the processes running within AWS. The physical connection to external networks is via the hospital's Internet Gateway.

The Servers are located within the Hospital's Data Centre, which is managed by the Hospital Systems Manager (Sysman), and therefore the Hospital Sysman also manages the Servers located within it. The Hospital is the operator of the Data Centre, the Servers and the processes running on them, meaning that the legally responsible party for these assets is the Hospital. The Sysman works for the Hospital and has delegated rights to manage these assets. The Data Centre is located within a particular jurisdiction, which is important when considering aspects such as data protection legislation (e.g. local implementation of the GDPR.

Because the Cardiologist's Notebook is the property of the Hospital, it is managed by the Hospital Sysman. The Cardiologist's Tablet is the property of the Cardiologist, therefore it is managed by them.

The Patient and AWS-centric assets are shown in *Figure II-13* (with a small overlap to the hospital). Here the Patient is shown, and links to the EMR Record and the EMR Record View Data (hosted in the hospital); and the EHR Record (hosted in AWS) indicate that these data items "relate to" the patient, meaning that they are classed as personal data. Further, these data elements are classed as special category data under GDPR Article 9 because they contain details about the patient's health.

The Patient runs the EHR app on their mobile phone, which connects to the EHR Lambda service and accesses the EHR data, both hosted in AWS. The phone is subject to a jurisdiction due to its location, and is connected to a cellular network, which in turn connects to the Internet via the Phone Provider's gateway.

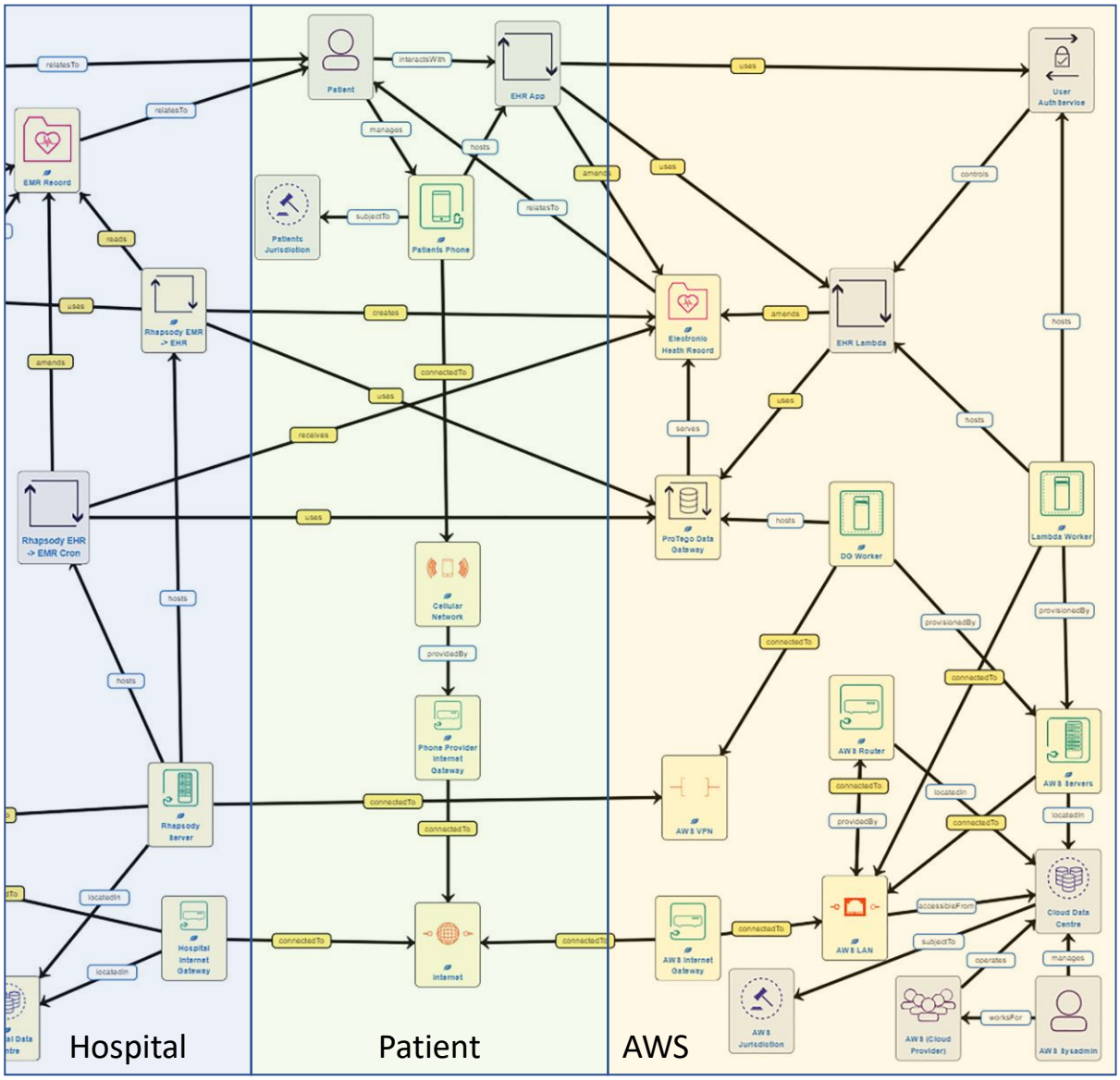


Figure II-13: Patient and AWS Detail

Inside AWS, the key data item is the Electronic Health Record (EHR). This is encrypted data hosted within the ProTego Data Gateway and served by the EHR Lambda process, to which the EHR app on the patient's phone connects, authenticating via a user auth service. The Data Gateway and the EHR Lambda processes are hosted in two Kubernetes worker nodes, the DG Worker and the Lambda Worker, which are in turn hosted on physical AWS servers. The AWS Servers are connected to the AWS LAN, which is provided by the AWS Router and accessible to the Internet via the AWS Gateway, and the DG Worker is connected to the VPN shared with the hospital. Similar to the hospital, the AWS Servers are located within the Cloud Data Centre, which is operated by AWS (as the Cloud Provider) which is subject to the AWS Jurisdiction and managed by the AWS System Manager.

II.3.2. Initial Misbehaviour Impact Levels

The initial impact levels were set in consultation with Marina Salud and depending on the criticality of the asset in question. The policy adopted in setting the impact levels of the misbehaviours was to consider the key data assets in the overall system as the determinator of impact. Once the impact levels of the key data items were set, similar impact levels were set on processes that serve and operate on them, as well as the devices that host the processes.

The most important data asset in the system is the *EMR Record* because it is the medical record of the patient, so any corruption, leakage or unavailability of this asset is intolerable. Therefore, the impact of the key misbehaviours (e.g. Loss of Authenticity, Loss of Availability, Loss of Confidentiality, Loss of Integrity and Loss of Timeliness) on the EMR record are all set to *Very High*. The associated processes and their host servers' corresponding attributes (e.g. Loss of Availability, Loss of Reliability, Overloaded, and Loss of Control) were also set to corresponding impact levels.

Both the *EMR Record View Data* and the *Electronic Heath Record* assets are important, but of slightly lesser importance to the EMR record, so they share similar, slightly lower impact levels when compared to the EMR record. For both types, the misbehaviour attributes concerning corruption and leakage impacts are set to *High*, meaning that they are highly important but not of critical importance. For both types, the availability misbehaviour is also set to *Medium*, meaning that it can be tolerated for a short time. For the view data, the timeliness misbehaviour is set to *High*, and for the EHR, it is set to *Medium*. This is because the slightly less up to date records can be tolerated at the EHR record, which is pulled back into the hospital via a daily scheduled process. Similar impact values are set for the processes and the host devices related to the view data.

II.3.3. Existing Known Controls

The existing known controls were applied to the model via direct consultation with Marina Salud (for the hospital portion of the model) or by using reasonable expectations of good practice for security (for the AWS portion of the model). Specific controls were applied as discussed individually below.

Access Control. The assets below have an enforced policy specifying who has access.

- EHR Lambda
- EMR Citrix Server
- EMR Database
- EMR Record
- EMR Record View Data
- Electronic Heath Record
- ProTego Data Gateway

Access Key. The process ProTego Data Gateway has a key for encrypting or decrypting data.

Anti-Malware. The hospital-managed devices (EMR and Rhapsody Servers and Cardiologist's Notebook), as well as the devices managed by AWS (AWS Servers) have anti-malware software installed on them. The devices managed by the Cardiologist and the Patient (tablet and phone respectively) are not assumed to have anti-malware on them.

Device Certification. The AWS Router and AWS Internet Gateway devices have been independently tested and certified as secure to a suitable evaluation assurance level.

Encryption. The Electronic Heath Record is encrypted both when stored on disk or when transmitted in the network.

Encrypted Processing. The processes EHR Lambda and ProTego Data Gateway are able to process inputs in the encrypted domain using homomorphic encryption.

Firewall blocks. The following assets have had a default policy applied to drop messages directed to a network address (interface) or via a network router (logical segment).

- Interface between the AWS Gateway and Internet
- Interface between the AWS Gateway and AWS LAN
- Interface between the Cardiologists Notebook and the Hospital LAN

- Interface between the Cardiologists Tablet and the Hospital Private Staff WLAN
- interface between the DG Worker and the AWS VPN
- Interface between the Hospital Internet Gateway and the Hospital LAN
- Interface between the Hospital Internet Gateway and the Internet
- Interface between the Hospital Router and the Hospital LAN
- Interface between the Hospital Router and the Hospital Private Staff WLAN
- Interface between the Lambda Worker and AWS LAN
- Interface between the Phone Provider Internet Gateway and the Cellular Network
- Interface between the Phone Provider Internet Gateway and the Internet
- Interface between the Rhapsody Server and the Hospital LAN

Health Monitoring. The AWS Servers, the DG and the Lambda Worker processes, as well as the User Auth Service are all monitored to detect any problems (errors or crashes).

Independent Instances. The AWS Servers are considered to be a set of instances whose provisioning is independent of their management.

Password. The AWS Sysadmin, Hospital Sysadmin, Cardiologist and Patient all have registered a password for identification purposes, which may be stored in a process (here the EHR app and the Tablet Citrix Client) acting on their behalf.

Password Quality Check. The following hosts or processes have a way to check the quality of a password when set or changed by a user. For example, it may use checks as specified in NIST-800-63.

- Cardiologists Notebook
- DG Worker
- EHR App
- EHR Lambda
- EMR Citrix Server
- EMR Server
- Lambda Worker
- Rhapsody Server
- User AuthService

Password Verifier. The host, process or space has a means to verify a password given by an authorised user. In the case of a physical space this may be verified by a security guard or a keypad, etc.

- Cardiologists Notebook
- Cloud Data Centre
- EHR App
- EHR Lambda
- EMR Citrix Server

Penetration Testing. The AWS Internet Gateway, the AWS Router and the AWS Servers have been tested to check they are not vulnerable to certain attacks.

Personal Device. The Cardiologists Tablet and the Patients Phone are each classed as a personal device, carried by and therefore monitored and protected by its only user.

Physical Checks. The Cloud Data Centre and the Hospital Data Centre are checked physically at suitable intervals to detect any physical alteration or removal of system assets.

Physical ID. The AWS Sysadmin, the Cardiologist and the Hospital Sysadmin have obtained documentation to prove their identity from a trusted source (e.g. an employer or a national authority).

Physical ID Verifier. The Cloud Data Centre and the Hospital Data Centre both has a means to check the physical ID of authorised users. This may be done by a human guard or by an automated system, depending on the documentation used.

Physical Lock. The Cloud Data Centre and the Hospital Data Centre each has a physical lock preventing access by users who do not possess a suitable key.

Secure BIOS. The following devices (all but the personal devices of the Cardiologist's Tablet and the Patient's Phone) have a secure BIOS and boot up sequence, ensuring its security cannot be bypassed by rebooting using an external (e.g. USB) boot device.

- AWS Internet Gateway
- AWS Router
- AWS Servers
- Cardiologists Notebook
- EMR Server
- Hospital Internet Gateway
- Hospital Router
- Phone Provider Internet Gateway
- Rhapsody Server

Security Training. The AWS Sysadmin and the Hospital Sysadmin have undergone security training before acting in the specified role.

Secure Configuration. The following devices employ removal of any built-in vulnerabilities in a device, coming from default configurations prior to entry of the affected device into the system.

- AWS Internet Gateway
- AWS Router
- AWS Servers
- Cardiologists Notebook
- EMR Server
- Hospital Internet Gateway
- Hospital Router
- Phone Provider Internet Gateway
- Rhapsody Server

Software Patching. The following devices (all but the personal devices) all employ a well-defined process for applying updates to the software for the associated host or process.

- AWS Internet Gateway
- AWS Router
- AWS Servers
- Cardiologists Notebook
- DG Worker

- EMR Server
- Hospital Internet Gateway
- Hospital Router
- Lambda Worker
- Phone Provider Internet Gateway
- Rhapsody Server

Software Testing. The process software has been tested to verify that it functions correctly.

1. EHR App
2. EHR Lambda
3. EMR Citrix Server
4. EMR Database
5. Notebook Citrix Client
6. ProTego Data Gateway
7. Rhapsody EHR -> EMR Cron
8. Rhapsody EMR -> EHR
9. Tablet CitrixClient
10. User AuthService

TLS. The following processes implement transport layer encryption for its communications.

- EHR App
- EHR Lambda
- EMR Citrix Server
- Notebook Citrix Client
- ProTego Data Gateway
- Rhapsody EHR -> EMR Cron
- Rhapsody EMR -> EHR
- Tablet CitrixClient
- User AuthService

X509 Verifier. The EMR Database has a means to verify that communication is with the holder of a private key corresponding to a trusted public key association such as an X509 certificate.

The model validator detected six modelling anomalies, but these relate to the system detecting an inaccessibility of the Hospital and AWS System Managers to the resources in their charge, but the model has declared that the Sysmans manage the physical spaces these resources are located in (i.e. the respective data centres), and therefore there is physical access to the hardware, so these anomalies are ignored.

Once these controls were applied and the impact levels set, the risk model is considered to be in a state that represents the real system. Next the risk assessment is run and the risks are determined.

II.3.4. Risk Assessment and Controls to Reduce Risks

The initial risk calculation shows that the worst-case risk in the whole system is Very High and that several misbehaviours on key assets have risks at this level. This is shown in Figure II-14

below, and there are approximately three pages of Very High-level risks in the misbehaviour table at the right.

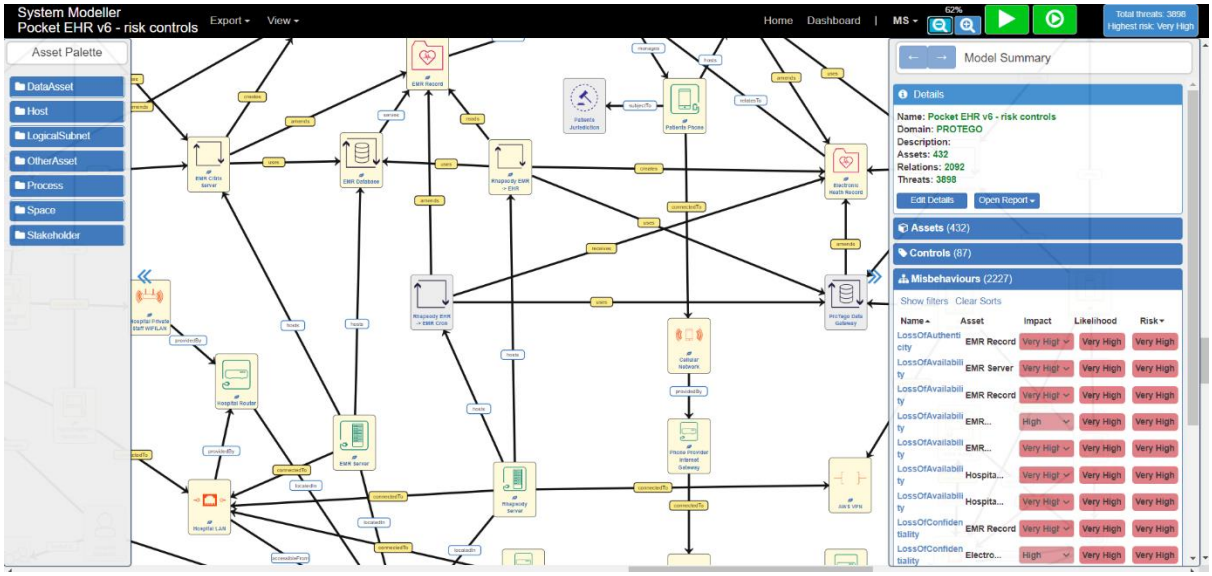


Figure II-14: Initial Risk Assessment Results

Given that the critical asset is the EMR Record, the risks to this will be addressed first. The first threat is Loss of Confidentiality at the EMR Record, illustrated in *Figure II-15*. This is Very High impact (determined by setting the impact levels above) and Very High likelihood (calculated by the SSM tool). Clicking on this threat produces the Misbehaviour Explorer window (left), and threats can be investigated. In the figure, the root cause threats of the misbehaviour are highlighted via an optional filter – this is usually good practice as in many cases addressing root cause threats can reduce the likelihood of many other secondary threats, so addressing root causes first can significantly reduce the number of likely threats. Here, the threats are caused by missing Firewall blocks between different logical subnets involving connections between the Hospital gateway and the internal LAN, and the Hospital Gateway and the internet. Similar FW blocks are missing at the AWS servers.

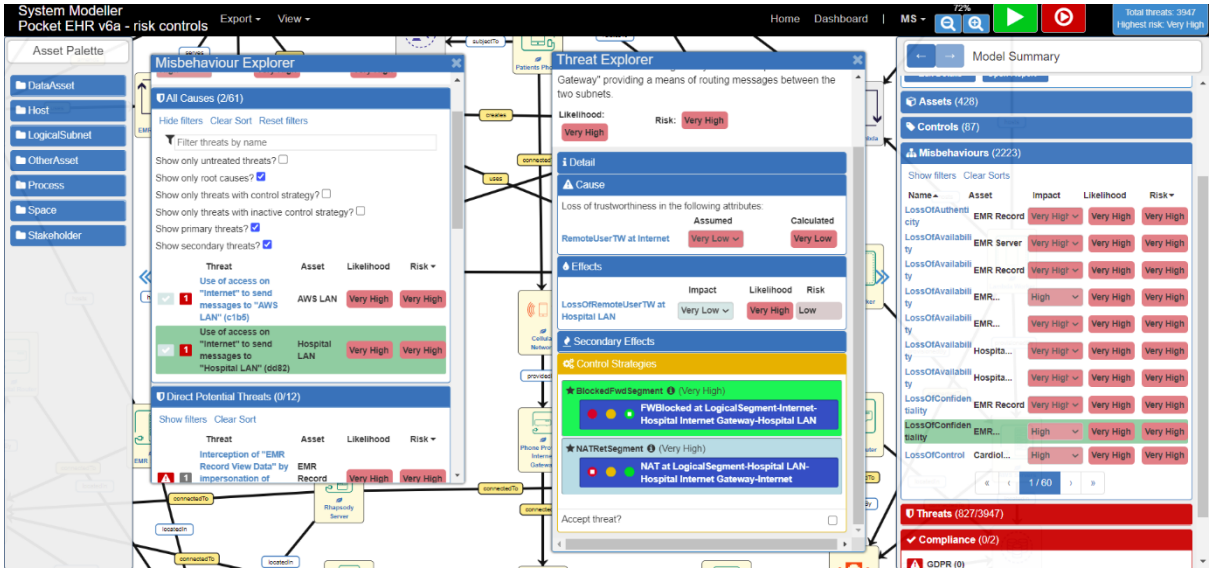


Figure II-15: Loss of Confidentiality at EMR Record

These controls are applied, and the result is that the Loss of Confidentiality risk to the EMR Record is reduced to Very Low, as shown in *Figure II-16*.

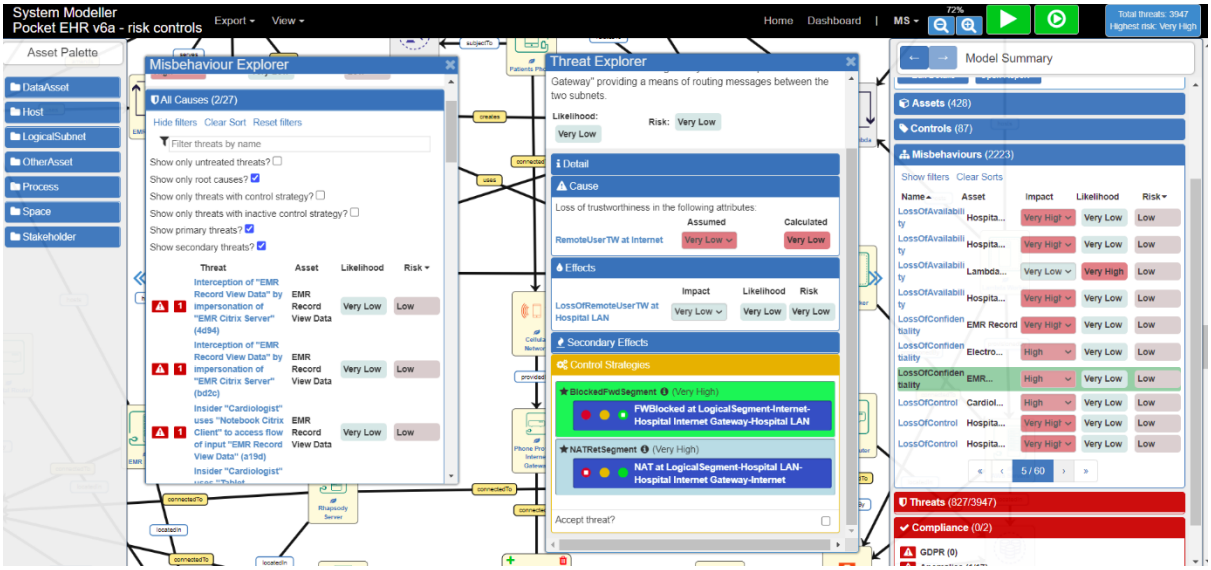


Figure II-16: Risk Reduction due to Additional FW Blocks

The next highest threat is a Loss of Timeliness on the Electronic Health Record (*Figure II-17*). There are three causes. The first is similar to the previous threat, and whose control is a similar FW block on the subnet between the phone provider's internet gateway and the cellular network (because the patient uses their phone connected to the cellular network).

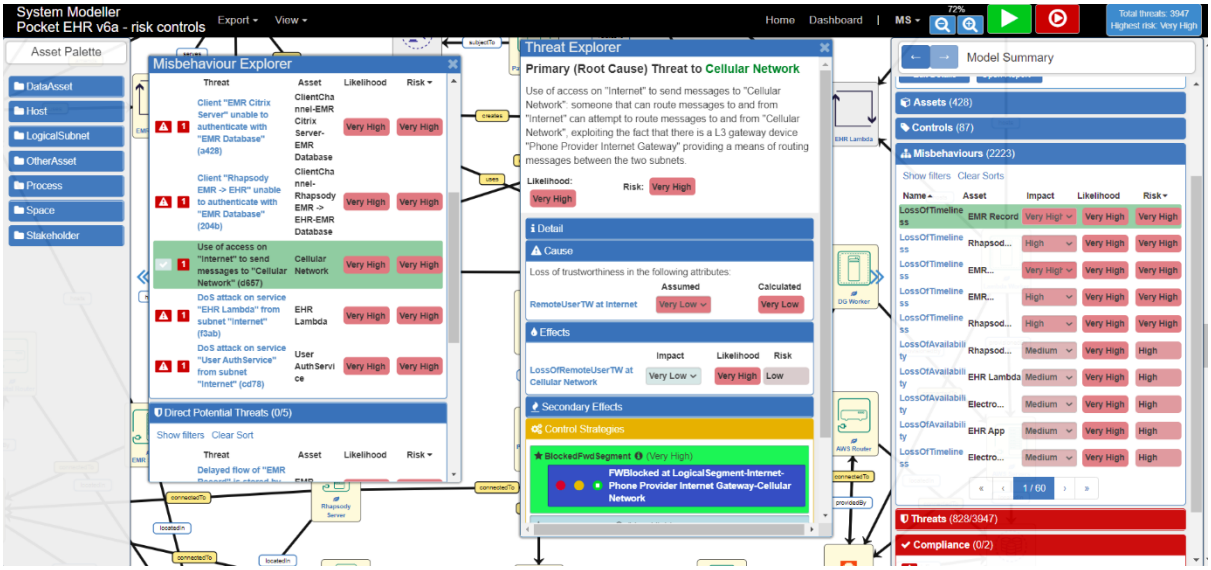


Figure II-17: Loss of Timeliness on EHR

The next cause is client processes being unable to authenticate with the EMR database (*Figure II-18*). The EMR database already has the X509 verifier control applied, so the recommended control is to apply X509 certification on the client end of the channel.

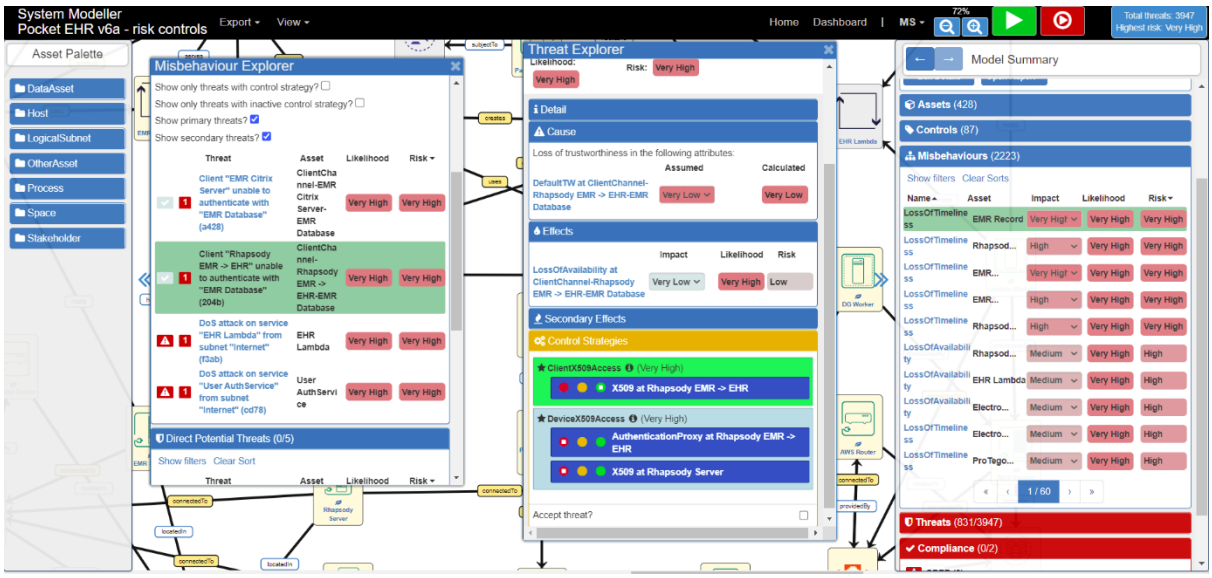


Figure II-18: Client Processes unable to Authenticate with EMR Database

Finally, there is a distributed denial of service attack at the EHR Lambda process and its associated authentication service (*Figure II-19*). This is addressed via bandwidth management at the interface between the EHR Lambda process and the AWS LAN.

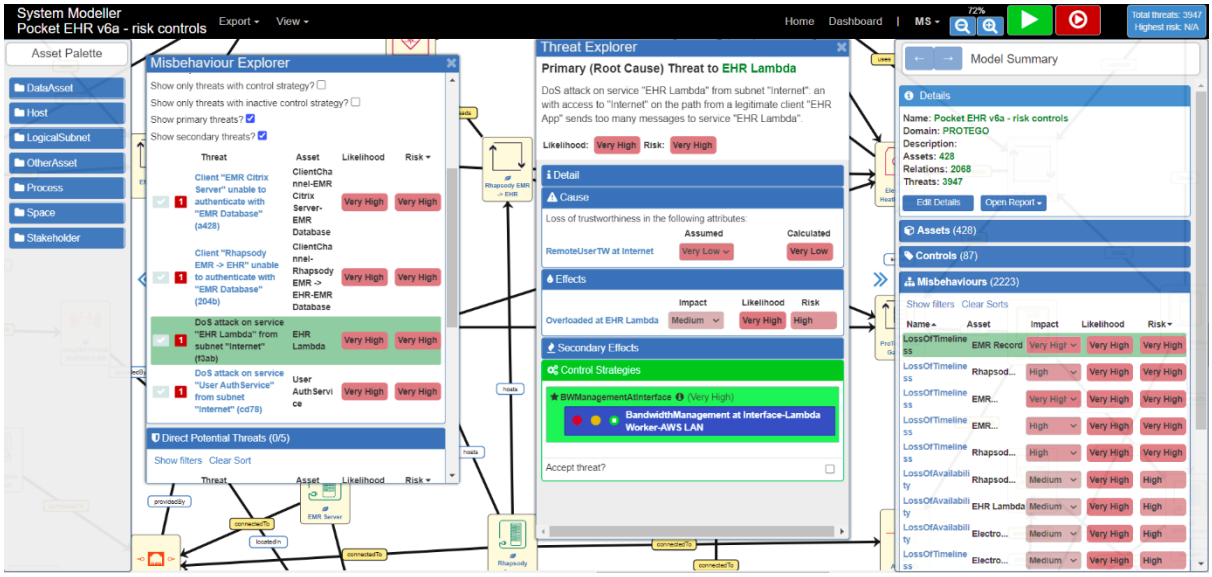


Figure II-19: DDoS Attack at EHR Lambda Process

Rerunning the risk calculation reduces the risk of the loss of timeliness on the EMR Record to high (*Figure II-20*):

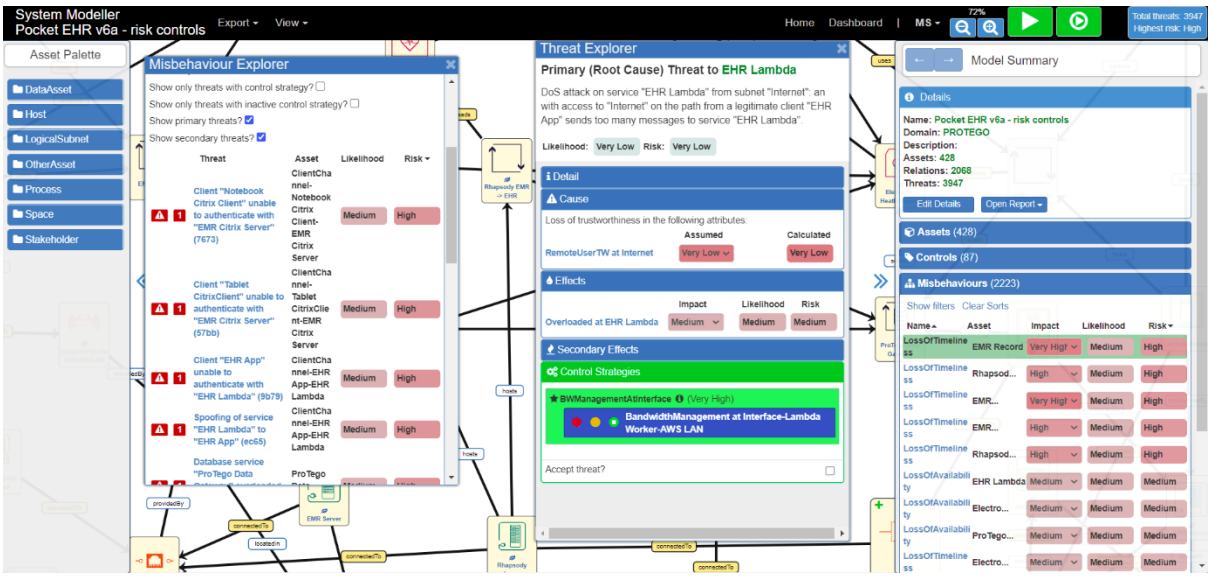


Figure II-20: Updated Risk Calculation

There are numerous causes to the remaining loss of timeliness for the EHR record, and are dealt with in groups. The first group concerns users forgetting their passwords, which can mean they cannot login to the services to provide updates. There are three options, as shown below. The preferred solution is to have a secure password store (e.g. KeePass - https://keepass.info/) co-located with the client application, so the user can securely access their password information. This can be applied for the Cardiologist's notebook and tablet, so is applied to these devices (*Figure II-21*), but is not available for the EHR app running on the patient's phone, so the second solution, password reset, is used for this device.

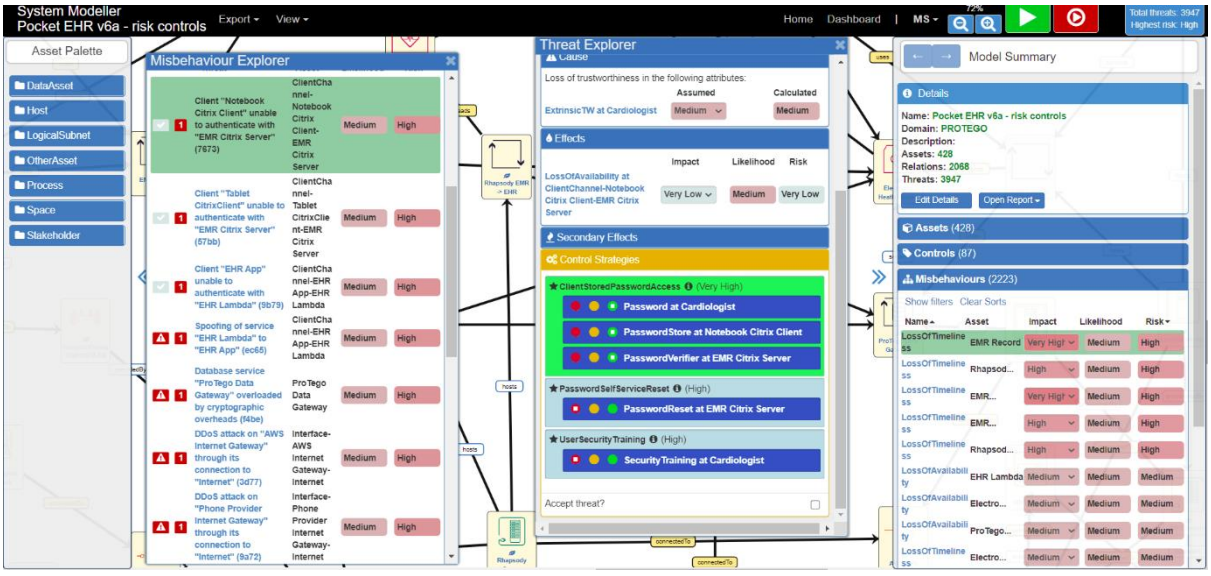


Figure II-21: Secure Password Storage

The next cause is a spoofing threat, where the EHR Lambda service could be spoofed to the EHR App (i.e. a malicious process could impersonate the Lambda process, *Figure II-22*). The control is to apply X509 certification at the sending end and X509 verification at the receiving end.

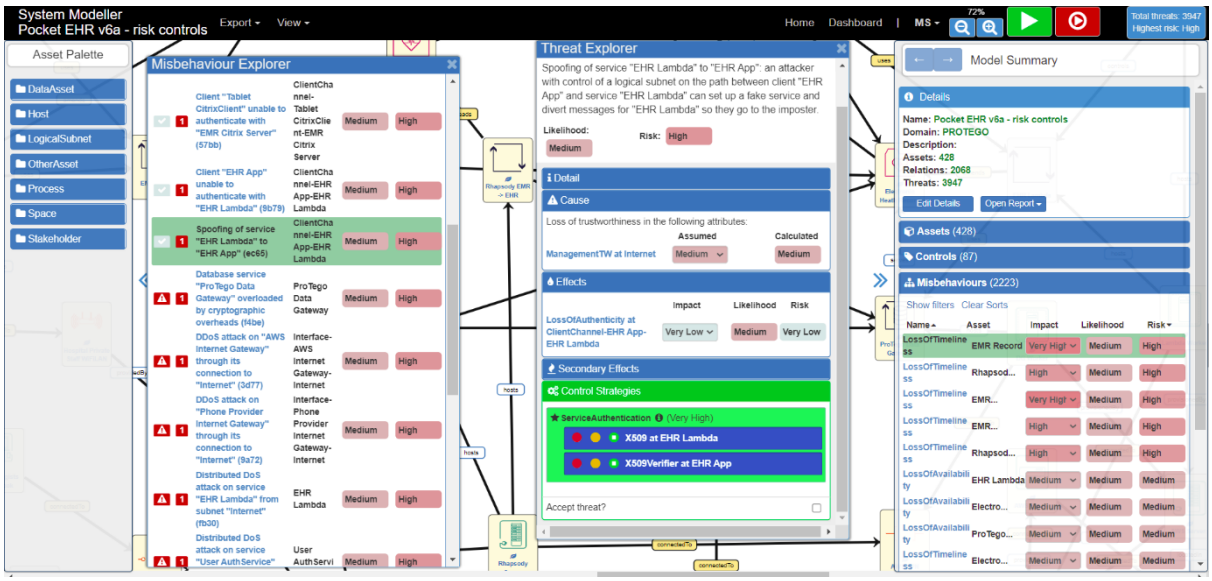


Figure II-22: EHR Lambda Spoofing

Recalculating the risks does not change the risk level as more threats need to be addressed. The remainder are in two groups. Firstly, there is a group of four further distributed denial of service attacks on the AWS gateway, the phone provider's gateway, the EHR Lambda and its authentication service, need to be addressed. Some DDoS attacks have already been addressed using controls before, but further controls are needed to lower the likelihood still further. The controls are applied as shown below in *Figure II-23*.

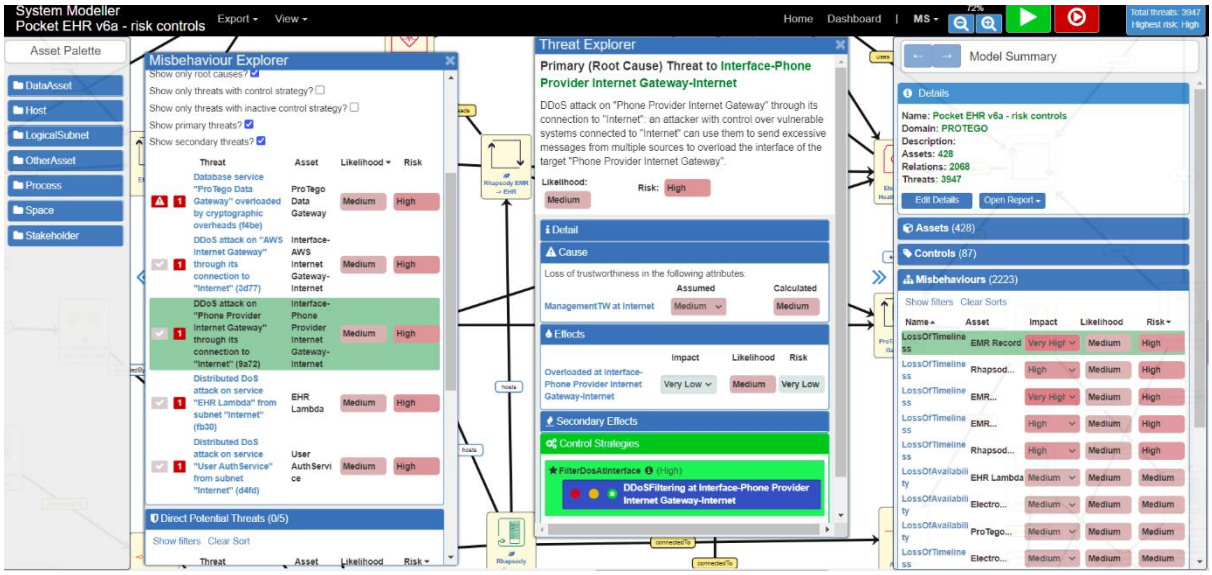


Figure II-23: Additional DDoS Controls

There remains one threat to be mitigated, the overload of the ProTego Data Gateway due to encryption processing (*Figure II-24*).

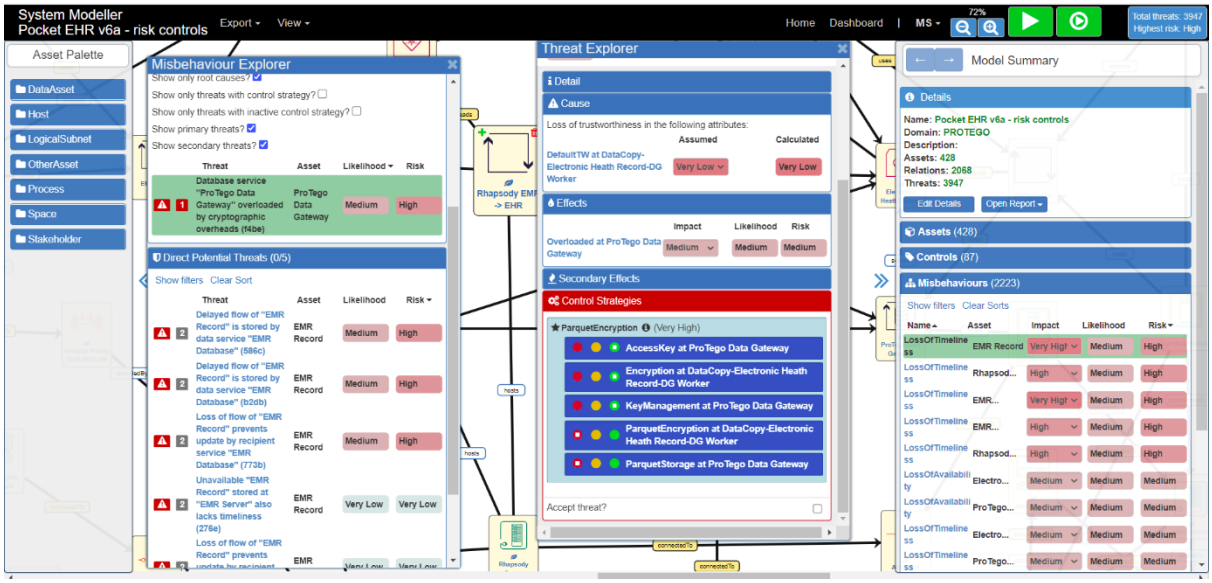


Figure II-24: Overload of ProTego Data Gateway due to Encryption Processing

The only control available is the ProTego Parquet Encryption, which represents an efficient encrypted database query processing scheme. The control strategy is partially applied already, as shown below, but the actual Parquet storage and encryption are also needed. These controls are applied to complete the control strategy, as shown in *Figure II-25*.

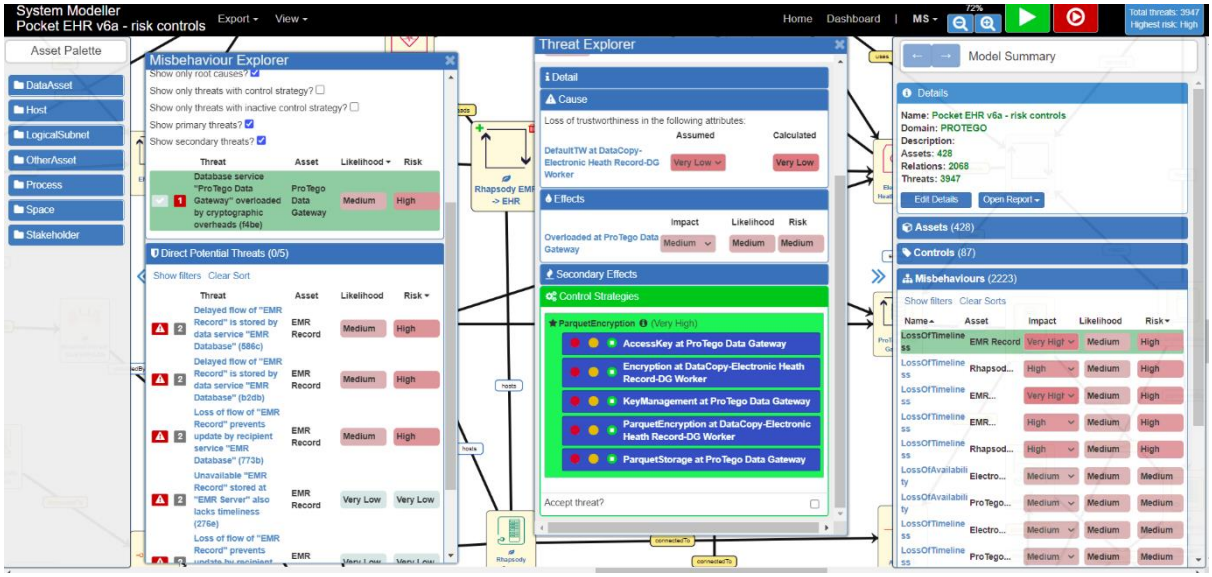


Figure II-25: Parquet Encryption Controls

The overall risk level is recalculated, resulting in an overall worst case risk level of medium, all for loss of timeliness misbehaviours, corresponding to a worst case misbehaviour likelihood of low, both of which are deemed acceptable given that small losses of timeliness are tolerable. The final result of the risk calculation is shown in Figure II-26.

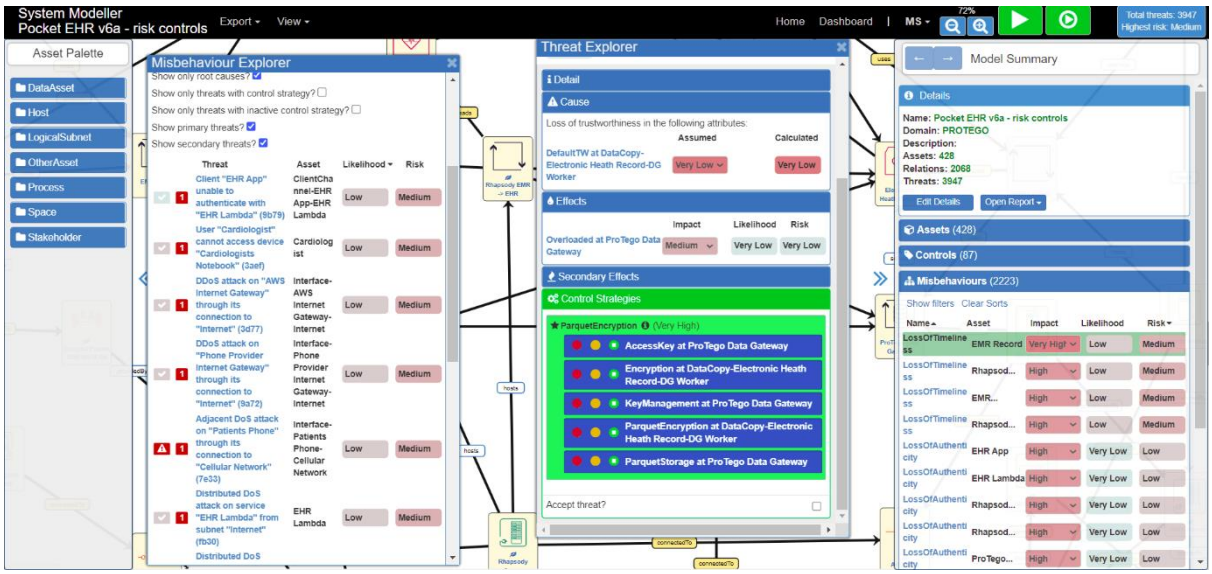


Figure II-26: Final Pocket EHR Risk Calculation

II.3.5. Summary

This section has described the Pocket EHR risk modelling process. A model was created of the real system, impacts on risks were determined and existing known controls were applied. This created the basis for the risk modelling. During the risk modelling process, misbehaviours were examined, worst first and controls applied to address them, gradually reducing the overall risk to an acceptable level using the decision support provided by the SSM tool.

A number of observations may be made.

- The initial security protection afforded by known existing controls was high. This is to be expected since the model is of a working hospital and a cloud data centre, both of which are necessarily highly competent in cyber security.
- Because major security challenges (e.g. loss of confidentiality, loss of integrity or loss of control) were already addressed by the existing known controls, the risk modelling tool therefore focused on timeliness and accessibility of important data. This is still important, as medical records and health data need to be kept available and up to date.
- Key areas identified by the tool to address the dominant loss of timeliness misbehaviour included:
    - Mutual authentication of processes to prevent spoofing.
    - Avoidance of delays to data timeliness caused by the loss of passwords.
    - Prevention of distributed denial of service attacks.
    - Prevention of overload of encrypted data query processing by using Parquet encryption.

II.4. Collaborative Risk Modelling

This section describes updates made to the System Security Modeller toolkit to enable different stakeholders to focus on the parts of the system they are responsible for – the aim is to enable cross-organisation, collaborative risk modelling. The approach is via encapsulation of areas of concern combined with selective information hiding, limiting and exposing access of different parts of the system respectively to different stakeholder types. The implementation of this is in the SSM as user-definable “concern groups”, into which assets can be placed. A group can correspond to an area of concern, and in the examples described below, the areas of concern are broadly split into application and infrastructure. Groups can be opened or closed – when a group is open, its assets are visible and are able to be manipulated; and when a group is closed, it is effectively a

"black box", hiding the assets within. Hence, different stakeholders can be exposed to the assets, risks and threats that concern them while other assets are hidden from view.

II.4.1. FoodCoach Scenario

The FoodCoach scenario has two groups – one for application-level assets and another for infrastructure-level assets. These groups correspond to the two stakeholder groups chosen for this illustration - the FoodCoach application developer and OSR's systems administrator – and the groups allow them to focus respectively on their areas of concern.

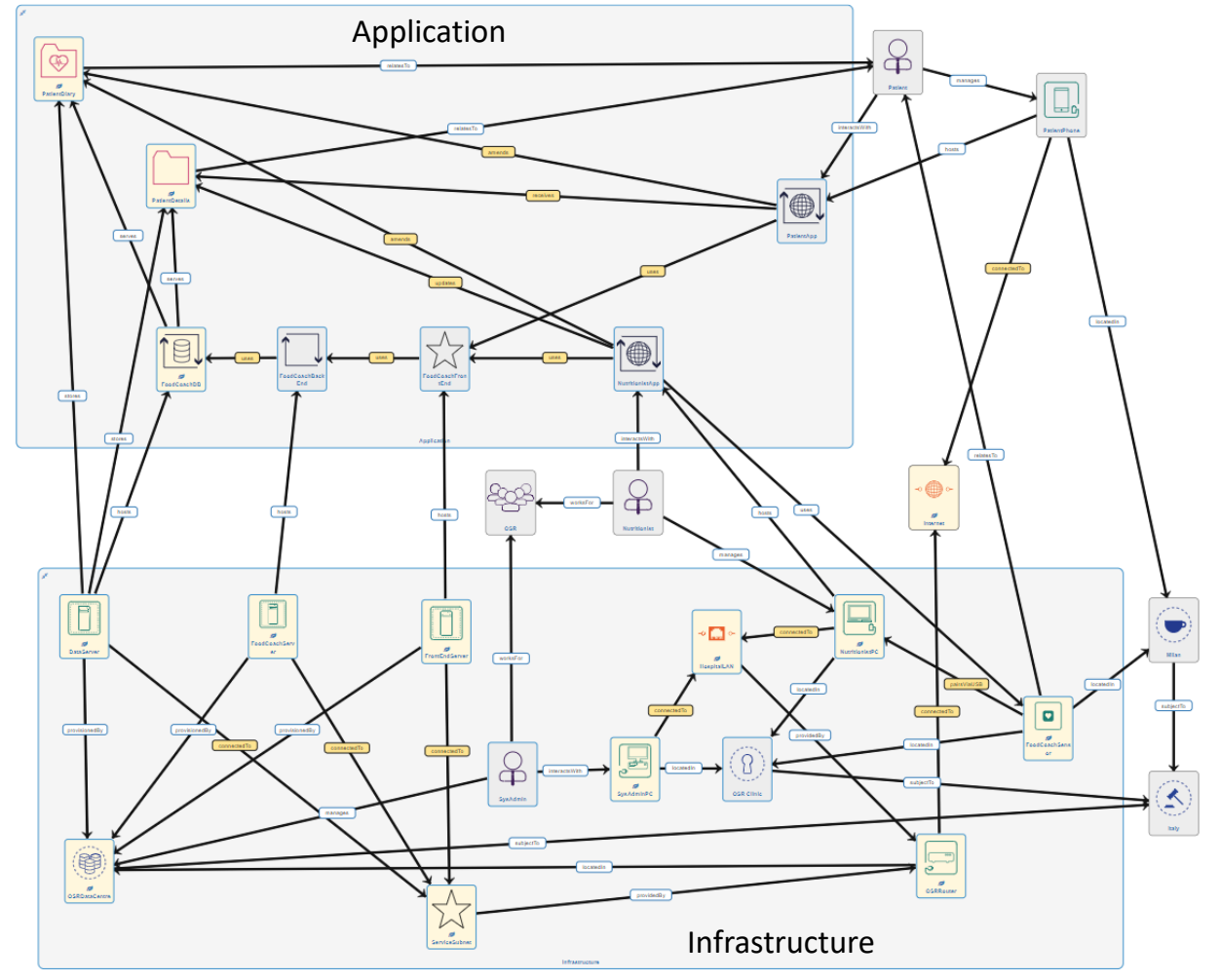


Figure II-27: FoodCoach Scenario Concern Groups - Overview

Figure II-27 shows a version of the FoodCoach risk model (compare with e.g. Figure II-4) with two groups – the top group is the application-level group and the bottom group is the infrastructure level group. Closer views of the same model are shown in Figure II-28 and Figure II-29 respectively.

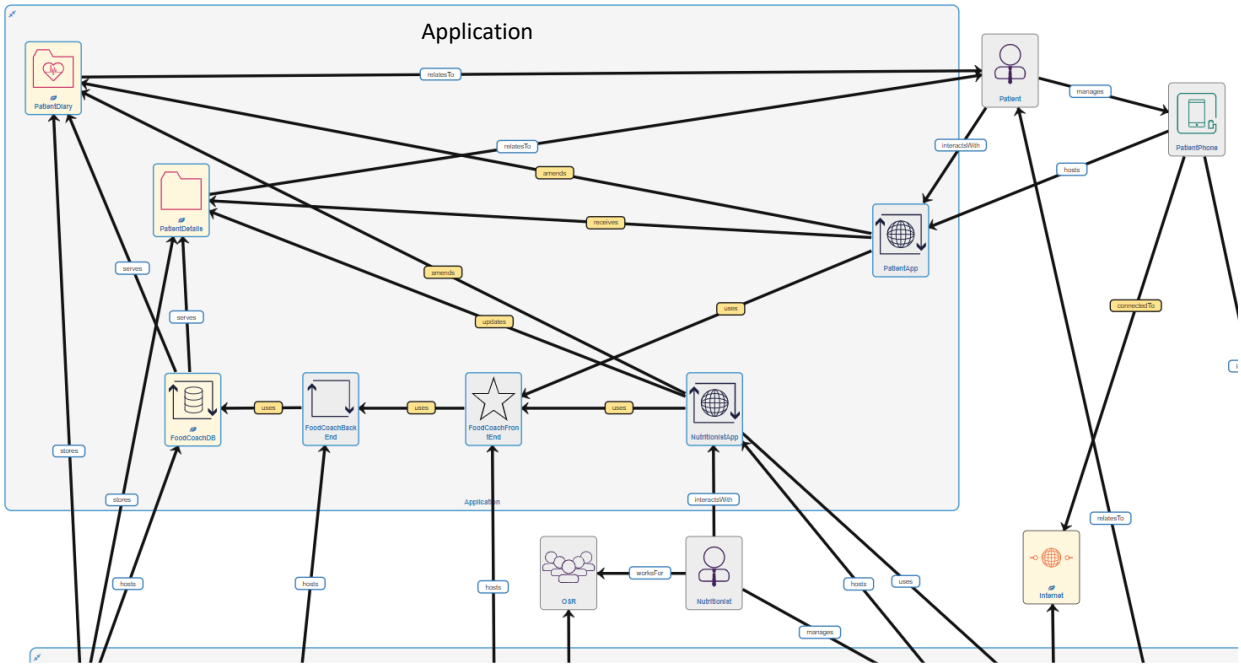


Figure II-28: FoodCoach Application-Level

The application level contains the data and process assets for the FoodCoach Scenario. These include the patient's data, the front end, the back end, the database and the two apps. All these are under the control of the application developer, so hence are all included in the group. Even though the patient's app is installed on their phone, it is developed by the application developer, so hence it is within their realm and included in the application-level group.

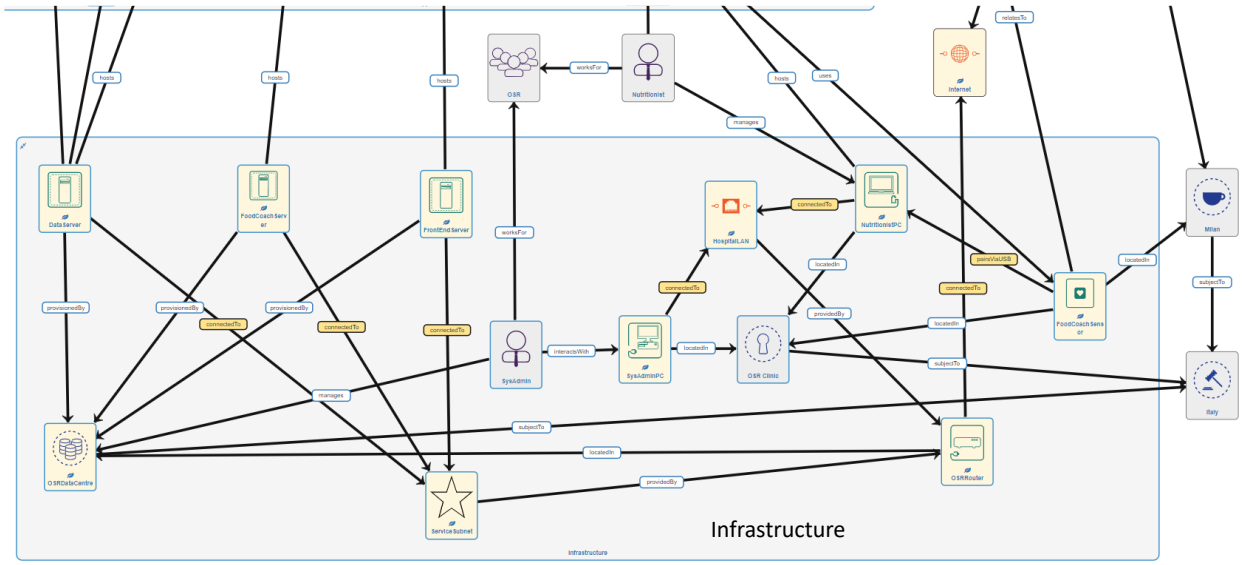


Figure II-29: FoodCoach Infrastructure Level

The infrastructure level contains all the infrastructure under the control of the hospital needed to support the application-level assets. Therefore, the infrastructure level contains servers, networks, the data centre in which the servers reside, routers and switches, the Nutritionist's desktop PC, the systems administrator and their PC. Of note is the FoodCoach Sensor, which is carried with the patient, but is supplied and administered by the hospital, so if is included within the infrastructure group as it is within their realm.

Some assets outside groups are common elements that are outside the area of concern of either stakeholder, and beyond their control. These are the patient, their phone, the Internet, Italy as the jurisdiction of OSR and Milan as a public space in which the patient's phone can be used, plus the OSR hospital and the Nutritionist. All of these are beyond the control of either the application developer or the system administrator.

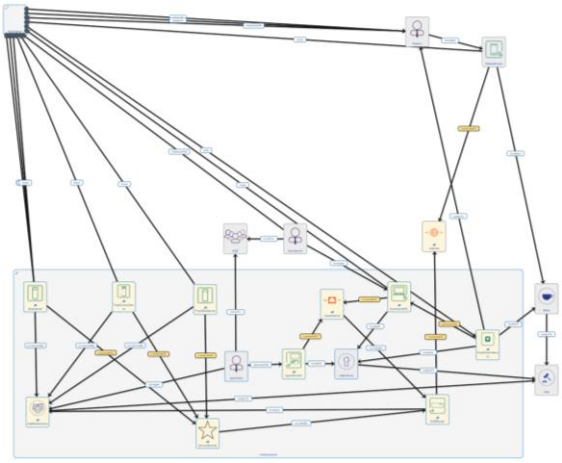


Figure II-30: FoodCoach - Infrastructure View

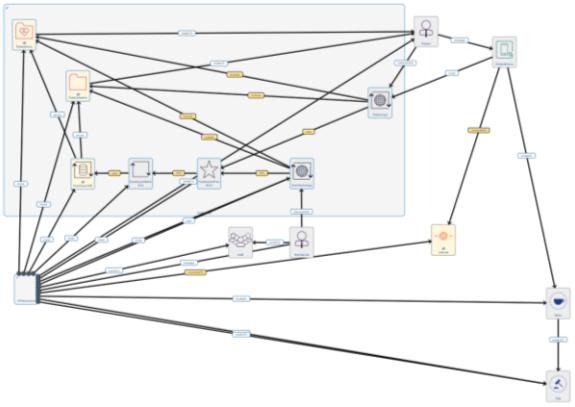


Figure II-31: FoodCoach - Application View

The figures above show two views corresponding to the areas of concern for each stakeholder – Figure II-30 shows the infrastructure view (where the application-level detail is hidden) and Figure II-31 shows the application view (where the infrastructure-level detail is hidden). It can be seen that assets outside either group (such as the Nutritionist, the patient etc) are shown in both views. Each group provides well-defined areas of concern that can be hidden from stakeholders who are not responsible for it.

II.4.2. Pocket EHR

A similar example for the Pocket EHR scenario is shown in Figure II-32 (which can be compared to Figure II-11 in Section II, which shows its risk model.

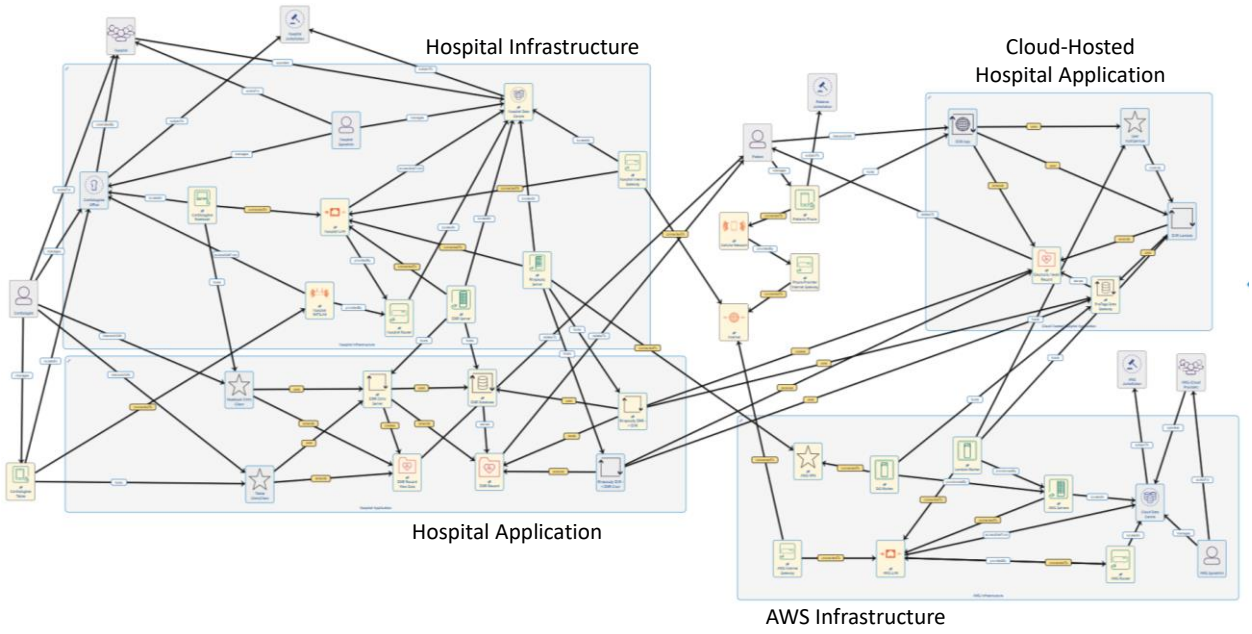


Figure II-32: Pocket EHR Concern Groups - Overview

A similar positioning of the key assets is adopted in both figures, but in Figure II-32 the key addition is the four groups representing different areas of concern. Here, there are four concern groups that correspond to three stakeholders. The mapping is given below in Table II-1.

Table II-1: Pocket EHR Stakeholder & Concern Groups

Stakeholder	*Concern Group*
Application Developer	Hospital Application, Cloud-Hosted Hospital Application
Hospital System Administrator	Hospital Infrastructure
AWS System Administrator	AWS Infrastructure

This illustrates that the concept and the implementation of concern groups is flexible enough to cover one-many relationships between stakeholders and groups. In this example, the decision to use two concern groups for the application is for logical separation between the application components that are hosted at the hospital and those that are hosted at AWS.

The Hospital Application concern group (Figure II-33) contains the application and data components that are located within the hospital. This includes the relevant existing systems of the hospital, e.g. the EMR Record and its View Data, the servers hosting this data (e.g. the EMR database) and the access components to this data (e.g. the Citrix server and client). It also includes specialist processes for the Pocket EHR application, e.g. the processes running in the Rhapsody engine that to extract the Electronic Health Record (EHR) from the EMR for display on the patient's phone, and to post results from the patient's feedback from the EHR back into the EMR.

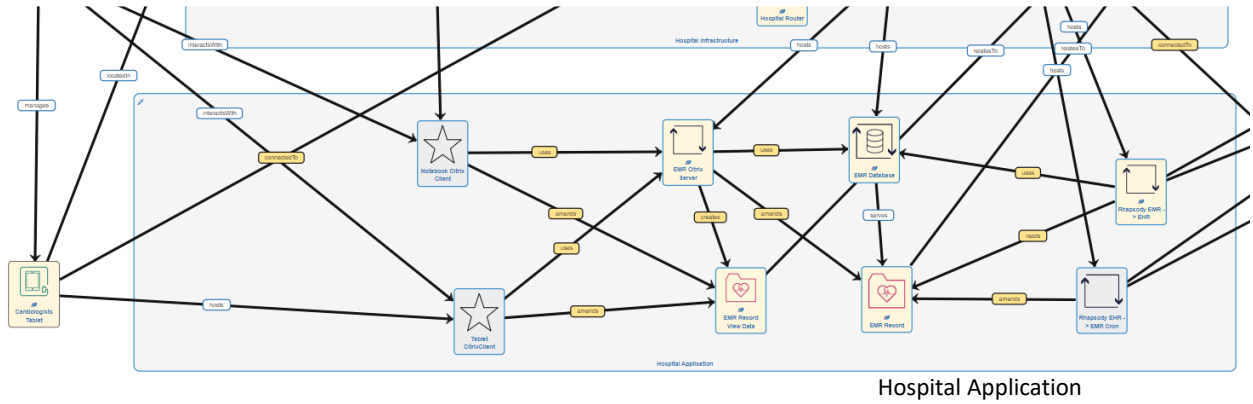


Figure II-33: Pocket EHR Hospital Application Detail

The Hospital Infrastructure concern group (Figure II-34) contains the infrastructure to support the Hospital Application components. This includes the EMR server, the Rhapsody server, the networks (both wired and wireless), routers and switches, the hospital's gateway to the Internet, the hospital's system administrator, the hospital's data centre and devices like notebooks used by staff but owned and operated by the hospital. Of note is that the cardiologist's tablet is outside the remit of the hospital (therefore outside the concern groups), because it is owned and managed by the cardiologist, and is therefore an exemplar of BYOD.

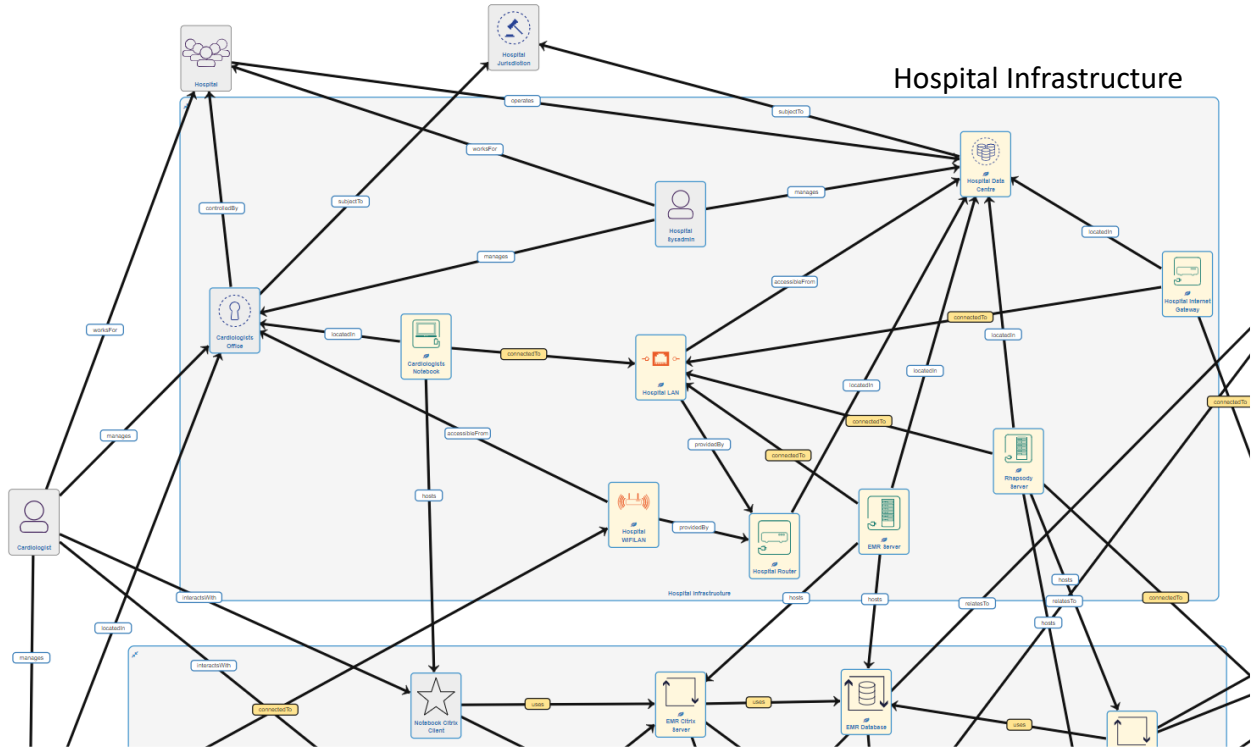


Figure II-34: Pocket EHR Hospital Infrastructure Detail

The Cloud-Hosted Hospital Application concern group (Figure II-35) contains the data and the processes that are hosted within AWS. This includes the Electronic Health Record, the ProTego Data Gateway, the EHR Lambda service process and the user authentication service. It also includes the EHR app, which is run on the patient's phone but is under the control of the hospital and uses data served by the AWS-Hosted processes.

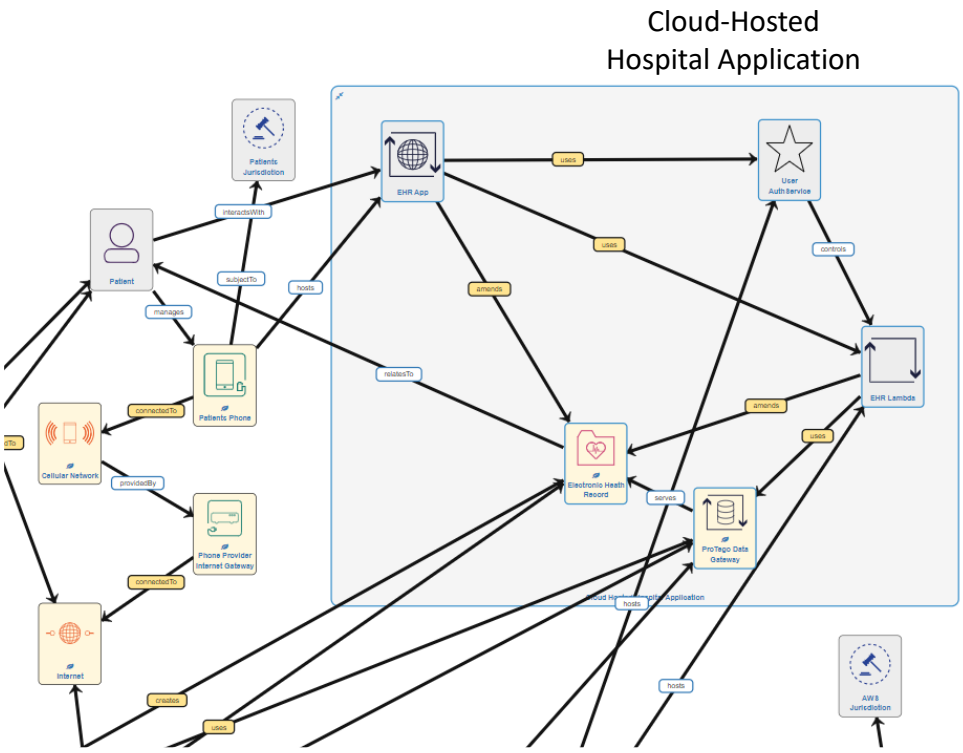
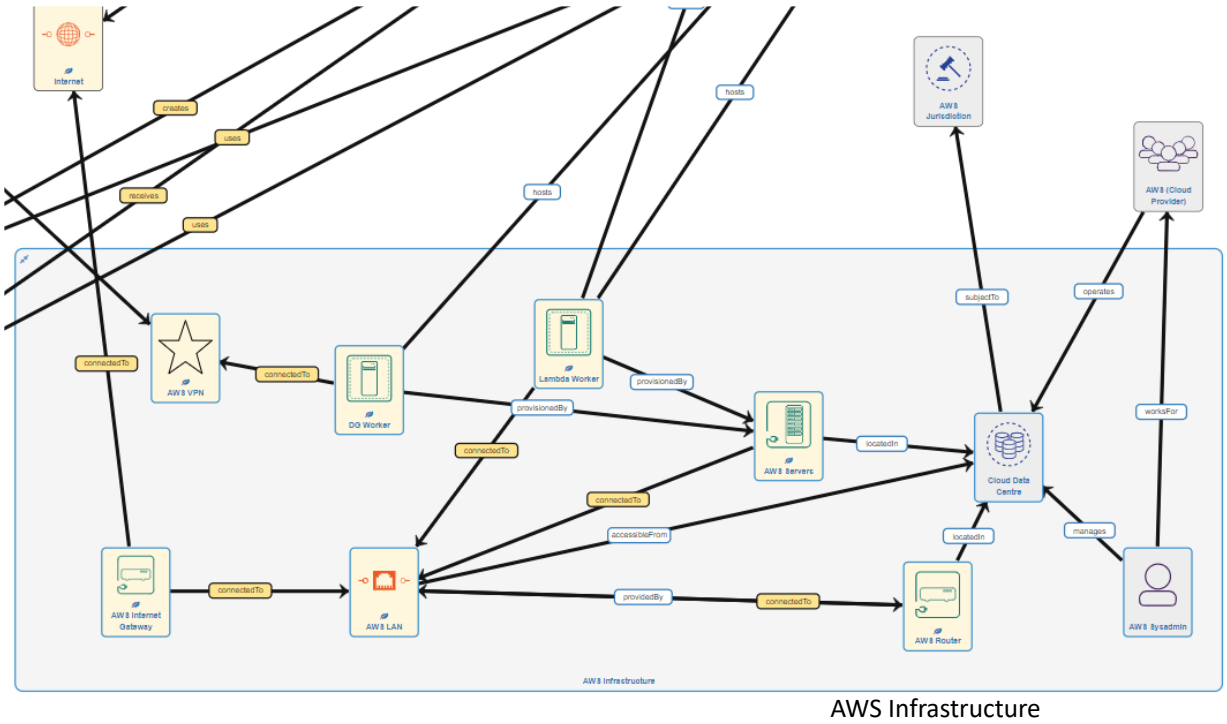


Figure II-35: Pocket EHR Cloud-Hosted Hospital Application Detail

The AWS infrastructure concern group (Figure II-36) contains the infrastructure necessary to host the Hospital Application. This includes the AWS data centre, the AWS sysadmin, routers, physical servers, virtual servers, physical and virtual networks. Even though the AWS system administrator is a recognised stakeholder, their activities and the AWS physical components are out of the control of any ProTego actor, so the main purpose of encapsulating the AWS components into a concern group is to hide them away from the other stakeholders (e.g. the Hospital system administrator and the Hospital application developer).



AWS Infrastructure

Figure II-36:Pocket EHR AWS Infrastructure Detail

There are some AWS components that may be under partial control of the hospital stakeholders are the virtualised hosts – i.e. the virtual servers that host the application at AWS and the virtual network providing the VPN, so it may be the case that these assets should be moved into a different concern group (e.g. the hospital infrastructure group), but the decision governing this is the extent to which they are under the control of the hospital. These are decisions made by the hospital stakeholders when developing and deploying the application.

As with the previous example, concern groups can be collapsed to hide their contents, and two examples are shown next. Figure II-37 shows the hospital infrastructure view, where all but the hospital infrastructure are collapsed, and Figure II-38 shows the application-level view, where the two concern groups for the application are open.

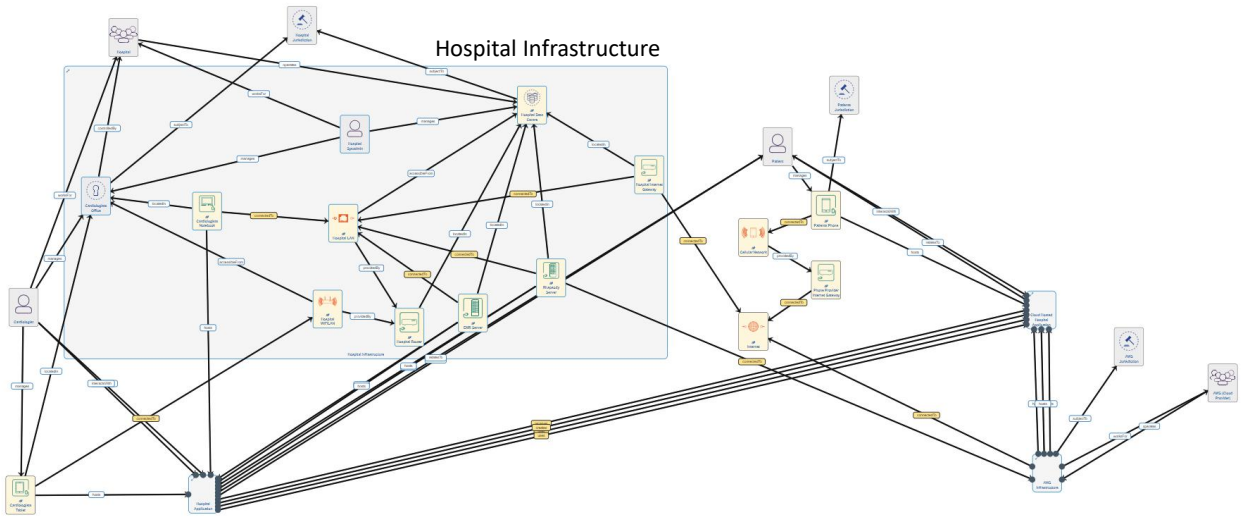


Figure II-37: Pocket EHR Hospital Infrastructure View

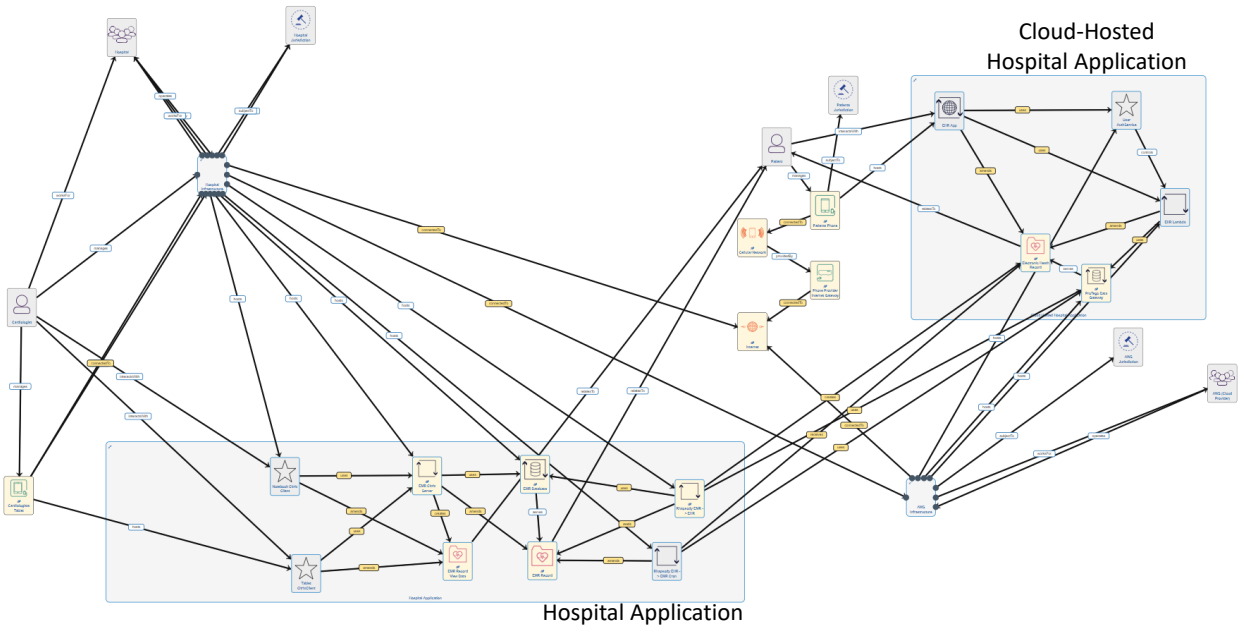


Figure II-38: Pocket EHR Application View

II.5. Summary & Outlook

This section has described recent work to support static risk modelling, both in actual form, where the risk models have been described for the FoodCoach and Pocket EHR end-user cases, as well as extensions to support collaborative cross-organisation risk modelling, enabling stakeholders responsible for different concerns to collaborate on risk modelling.

This section has also illustrated that the ProTego components reduce the risk of serious misbehaviours in both the FoodCoach and Pocket EHR scenarios. In particular, the ProTego Continuous Authentication provides ongoing protection for impersonation or vulnerabilities due to the theft of a BYOD device, and the ProTego Parquet encryption provides efficient encrypted data query processing.

Next steps are mainly focused on validation in WP7, where feedback will be factored into model updates as required.

III. DYNAMIC RISK ASSESSMENT

Risk Management provides automated decision support for human operators on risks to assets in socio-technical systems. H2020 ProTego takes a baseline of previous work on static, so-called "design time" risk assessment, where a system design is evaluated for cyber threats and the risks to the system that are caused by the threats, and has extended this work to enable a dynamic, runtime risk evaluation, where the risk assessment is automatically updated based on input from monitoring of the operational system. This extension provides continuous risk assessment, with close-to-real-time warnings of increased threat levels and includes associated recommendations for mitigating controls to reduce the threat levels.

III.1. Dynamic Risk Modelling Concept

Dynamic risk modelling is founded upon event and vulnerability detection via SIEM tools, with automated adjustment of system model parameters and recalculating the risk levels. The process is event-driven, i.e. when system monitoring tools detect vulnerabilities, the process of system model adjustment and risk recalculation is triggered automatically.

The SSM has a REST API, which forms the basis of its server-side processing of its web user interface, so that model changes from point and click events in the user's web browser are reflected in the system risk model stored at the server. To support dynamic risk evaluation, a new client has been created, named the Run Time Microservice. This component exposes its own REST interface, via which vulnerability reports can be input, and the microservice translates the contents of the vulnerability reports into updates to the system risk model.

The concept of dynamic risk assessment is shown in Figure III-1. In the figure, solid arrows refer to automated communication between different ICT processes, and dashed arrows refer to interactions between ICT processes and a human operator.

Static system setup is indicated by blue dashed arrows. Two major activities need to be undertaken: building a system risk model from the topology of the operational socio-technical system, and mapping the identifiers of assets in the operational socio-technical system to assets in the system risk model.

The event-driven dynamic risk assessment process begins with the red-coloured arrows in Figure III-1. Off-the-shelf vulnerability detection tools are used to monitor the ICT components of the operational social-technical system. Tools currently supported include OpenVAS [3] and OWASP ZAP [4]. These vulnerability detectors generate reports, which are passed to the ProTego SIEM, which collates them and delivers them to the Run Time Microservice by invoking its REST API. The events are mapped into the system risk model as adjustments in the risk model assets' trustworthiness levels (described later) by the Run Time Microservice and the adjustments are passed into the SSM, also by invoking its REST API. The Run Time Microservice also triggers a risk recalculation, which determines the new likelihood of threats given the new trustworthiness levels and the consequent likelihood of the misbehaviours that arise in assets as a result of the threats.

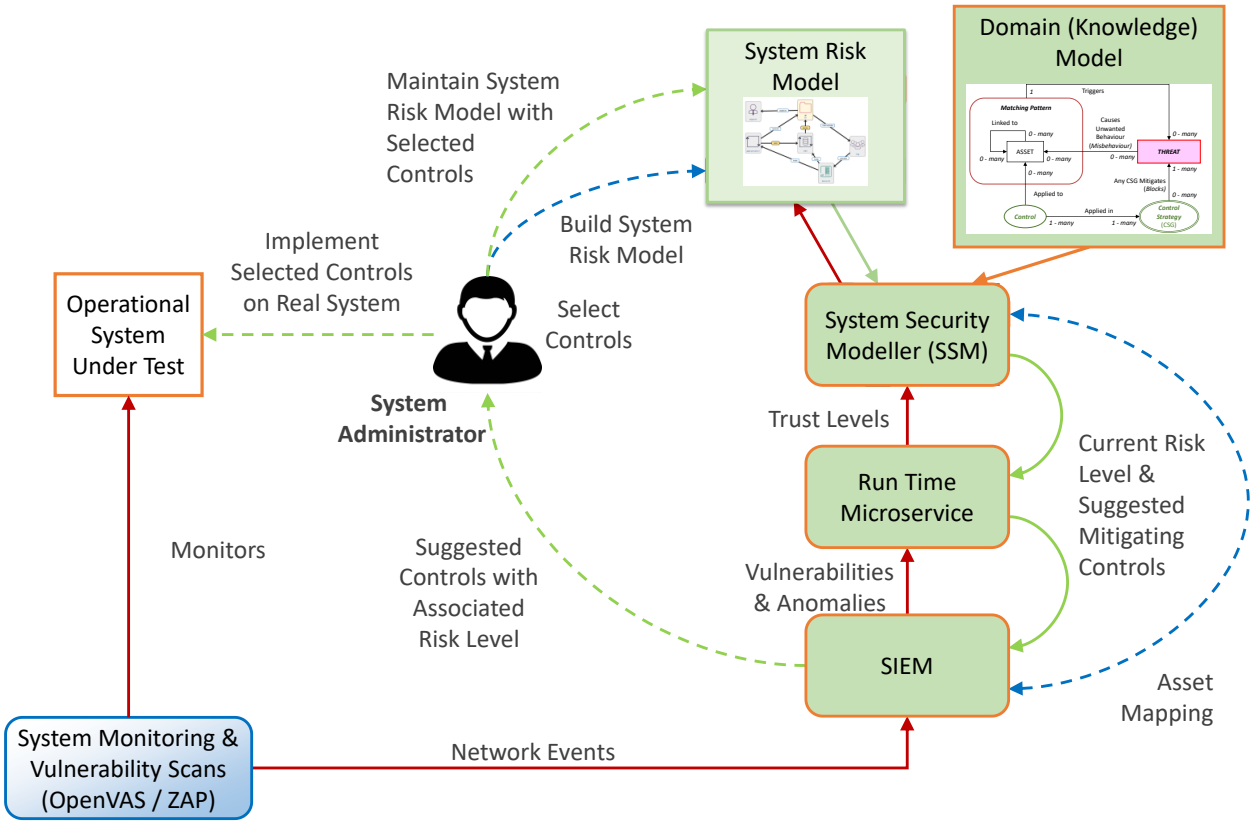


Figure III-1: Dynamic Risk Assessment

The response from the SSM is denoted by the green coloured arrows. When a risk level is recalculated, the SSM responds with an updated risk level (which is likely to be higher than before if new vulnerabilities have been detected and reported to the SSM). The risk level is reported as a ***risk vector***. This is a representation of each risk level (from "VeryLow" to "VeryHigh") with counts of how many misbehaviours are present at each risk level, for example:

```
"CurrentRisk": "High",
 "CurrentRiskVector": {
   "VeryHigh": 0,
   "High": 2,
   "Medium": 6,
   "Low": 140,
   "VeryLow": 932
  }
```

The current risk is the overall worst-case risk – i.e. the highest risk level where there is a non-zero misbehaviour count – in the previous example, the current risk level is High (because this is the highest non-zero risk level).

The SSM also returns recommendations for security controls, each with the resulting risk vector if those controls were applied to the real system. The risk vector and recommendations are passed from the SSM to the Run Time Microservice and then onto the SIEM, where they are made available to the human System Administrator (via the green dashed arrows). The recommendations are evaluated by the System Administrator, who selects the controls they wish to apply to the real system, and when they are applied to the operational system the System Administrator updates the risk model with them.

III.2. Run Time Microservice Updated Architecture & Functionality

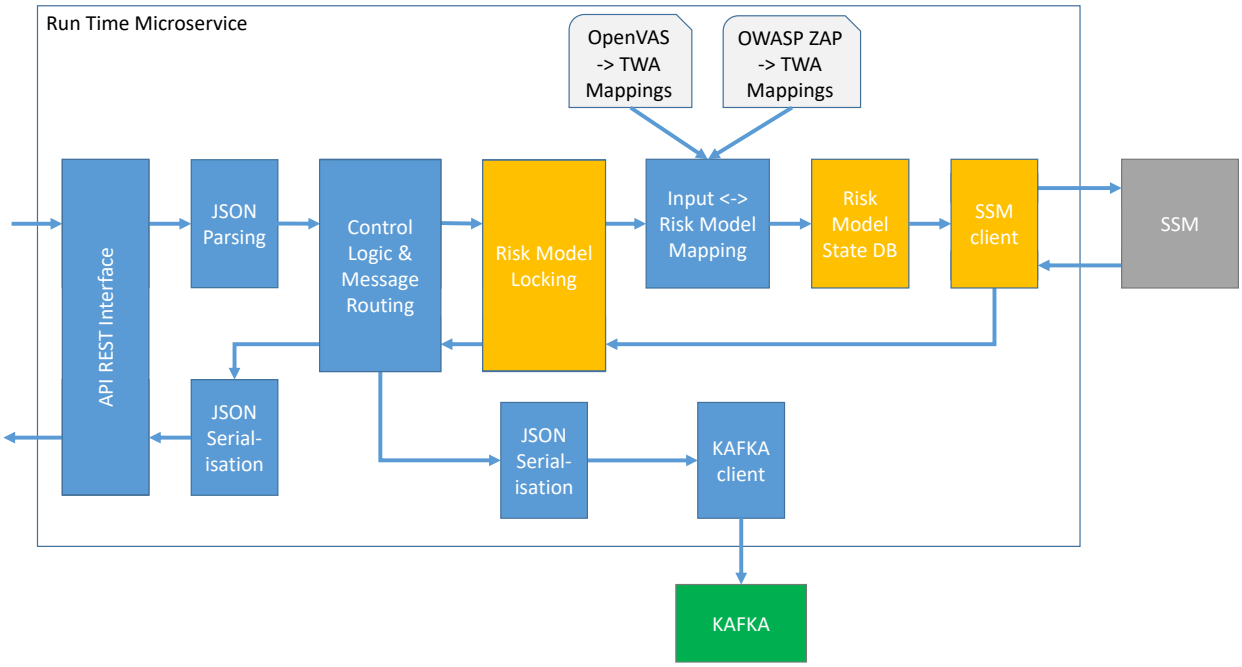


Figure III-2: Run Time Microservice Architecture

The architecture of the Run Time Microservice component is shown in Figure III-2. Development of this component is shared with H2020 FogProtect, and the shared subcomponents within the Microservice are indicated in orange. These are basic subcomponents that provide core functionality common to both projects. The blue subcomponents represent those that are dedicated to ProTego, with more advanced, project-specific functionality.

III.2.1. Dynamic Risk Assessment Process and Protocol

The overall protocol for dynamic risk assessment follows the form of a transaction for a batch of vulnerability reports.

The sequence comprises four main phases: begin the transaction, report the vulnerabilities, recalculate the risk and retrieve the results. These phases are executed via the usage of REST API calls on the Run Time Microservice (the API is described in the Appendix in Section VII.1).

1. Begin the transaction by executing the ***"reset-vulnerabilities"*** REST method. This resets any of the model's Trustworthiness Attributes that have been adjusted in a previous vulnerability reporting transaction. This method resets the attributes back to a known state, so that new vulnerability reports start from the same state each time.
2. Report vulnerabilities from OpenVAS and ZAP via execution of the ***"openvas-vulnerabilities"*** and ***"zap-vulnerabilities"*** REST methods respectively. These can be executed numerous times within one transaction if needed. These vulnerability reports are mapped into TWAs inside the Run Time Microservice and the adjustments to the risk model's TWAs are applied.
3. Run the risk calculation via execution of the ***"calc-risks"*** REST API method. This is a long-lived operation, so the REST API method returns immediately.
4. To retrieve the results of the risk calculation, there are two options. Firstly, the results (the risk vector and recommendations) can be pushed to a Kafka queue and subscribers to the relevant topics can receive them as push notifications. Secondly, the SIEM client can poll the status of the risk calculation (using the ***"task-status"*** REST method) and when it is complete, can call

additional methods to retrieve the risk vector (using the ***"risk"*** REST method) and the recommendations (using the ***"recommendations"*** REST method.

At this point, the human-in-the-loop takes over to assess the recommendations, apply controls to the operational system and to update the risk model.

5. The system administrator can examine the risk level and the recommendations and decide on a course of action. If the risk level is too high, the administrator can examine the recommendations to assess the recommended actions (sets of related controls) and the resulting risk level that would result if those controls were applied.
6. The system administrator applies the actions they select to the real system.
7. The system administrator updates the risk model (via the SSM User Interface) with the controls they have implemented in the real system.

III.3. Dynamic Web Application Support

This section describes an extension to the Run Time Microservice for dynamic assessment of web application. The approach chosen is to integrate the OWASP Zed Attack Proxy (ZAP), a vulnerability scanner for web applications. ZAP outputs alerts that describe detected vulnerabilities in web applications, and this section describes how these alerts are mapped into the SSM risk model.

As with the discussion of OpenVAS vulnerability support in D4.2, the target parameters in the SSM risk model are the Trustworthiness Attributes (TWAs) in risk model assets, as these represent propensities for assets to be affected by different types of threat. The target TWAs are described later in this subsection.

The key mapping tasks addressed by this section are:

- how to map the information in a ZAP alert report to which asset in the risk system model.
- which TWA to adjust in the selected asset.
- what level to set the TWA to.

How these tasks are accomplished is described in this subsection. This subsection begins with an example of a ZAP alert to illustrate the input to the mapping. It then describes the approach for selecting the assets in the risk model and how they map to the web applications referenced in the ZAP alert. This is followed by a description of the mapping between the information in the alert to the TWAs, including a summary of the target TWAs. The key source of information is from external vulnerability classifications that are referenced in the ZAP alert, and these are described in turn. The mapping algorithm is then described in detail, and the subsection concludes with some illustrative mapping examples.

III.3.1. ZAP Alert Example

Below is a truncated example of a ZAP alert report. Information that is utilised in the mappings work described here is highlighted in bold. This information will be discussed in later subsections illustrating how the mapping is achieved.


```
{
 "@version": "2.10.0",
 "@generated": "Tue, 9 Feb 2021 13:09:02",
 "site": [
  {
   "@name": "http://testasp.vulnweb.com",
   "@host": "testasp.vulnweb.com",
   "@port": "80",
   "@ssl": "false",
   "alerts": [
    {
     "pluginid": "10021",
```

```
      "alertRef": "10021",
      "alert": "X-Content-Type-Options Header Missing",
      "name": "X-Content-Type-Options Header Missing",
      "riskcode": "1",
      "confidence": "2",
      "riskdesc": "Low (Medium)",
      "desc": "<p>The Anti-MIME-Sniffing header X-Content-Type-Options
      was not set to 'nosniff'. This allows older versions of Internet
      Explorer and Chrome to perform MIME-sniffing on the response body,
      potentially causing the response body to be interpreted and
      displayed as a content type other than the declared content type.
      Current (early 2014) and legacy versions of Firefox will use the
      declared content type (if one is set), rather than performing MIME-
      sniffing.</p>",
      "instances": [
       {
        "uri": "http://testasp.vulnweb.com/showthread.asp?id=69",
        "method": "POST",
        "param": "X-Content-Type-Options"
       }
      ],
      "count": "408",
      "solution": "<p>Ensure that the application/web server sets the
      Content-Type header appropriately, and that it sets the X-Content-
      Type-Options header to 'nosniff' for all web pages.</p><p>If
      possible, ensure that the end user uses a standards-compliant and
      modern web browser that does not perform MIME-sniffing at all, or
      that can be directed by the web application/web server to not
      perform MIME-sniffing.</p>",
      "otherinfo": "<p>This issue still applies to error type pages (401,
      403, 500, etc.) as those pages are often still affected by injection
      issues, in which case there is still concern for browsers sniffing
      pages away from their actual content type.</p><p>At \"High\"
      threshold this scan rule will not alert on client or server error
      responses.</p>",
      "reference":"<p>http://msdn.microsoft.com/en-
      us/library/ie/gg622941%28v=vs.85%29.aspx</p><p>https://owasp.org/
      www-community/Security_Headers</p>",
      "cweid": "16",
      "wascid": "15",
      "sourceid": "3"
     }
    ]
   }
  ]
}
```


The *"site"* is the web application itself (e.g. website) and encapsulates the *"host"* and *"port"* information, which is used in the selection of assets in the risk model. The *"riskcode"* corresponds to the risk level of the current alert and the values are given in Table III-1 below. The *"confidence"* levels give the level of confidence in the alert and are given in Table III-2 below. The *"riskdesc"* is a description of the risk level and the confidence level, mapped from the following tables. The *"risk"* and *"confidence"* values are used in the mapping algorithm for filtration and to determine levels for TWAs.

Table III-1: Riskcode to Risk Level

Riskcode	Risk Level
0	Informational
1	Low
2	Medium
3	High

Table III-2: Confidence Value to Confidence Level

Confidence	Confidence Level
0	False Positive
1	Low
2	Medium
3	High

The two identifiers *"cweid"* and *"wascid"* provide important links to additional vulnerability classification schemes that provides useful information in the mapping task. How each of these attributes contributes to the mapping is described in the next few subsections.

III.3.2. Risk Model Asset Identification, Selection & Mapping

The risk model assets that map to the *"sites"* in the ZAP alert reports are processes that serve the web application. For example, in the Pocket EHR scenario, the key web application is the "EHR Lambda" process (see Figure **II-13**) and in the FoodCoach scenario the web application is the "FoodCoach Front End" (see Figure **II-5**). In both cases these assets represent the component that serves data to web clients (e.g. web browsers or apps on phones).

ZAP alert reports user "*host*" and "*port*" number to identify the web application site (see example ZAP report above), so the assets representing the web application need to be marked so they can be matched by the Run Time microservice. The mechanism for this is within the SSM User Interface – each asset has user-definable key-value pairs named "additional properties". The additional properties for the host and the port number need to be added for the relevant assets.

The example ZAP report given above cites the *"host"* as *"testasp.vulnweb.com"* and the *"port"* as *80*, so if this is a scan for the FoodCoach scenario, then the FoodCoach Front-End asset in the FoodCoach risk model needs its additional properties set with this information. This is illustrated in Figure III-3 via a screenshot of the SSM's user interface with the FoodCoach Front-End asset selected and the additional properties highlighted. Once this mapping is defined in the SSM UI, the Run Time microservice can direct all ZAP vulnerability reports for this site to this asset.

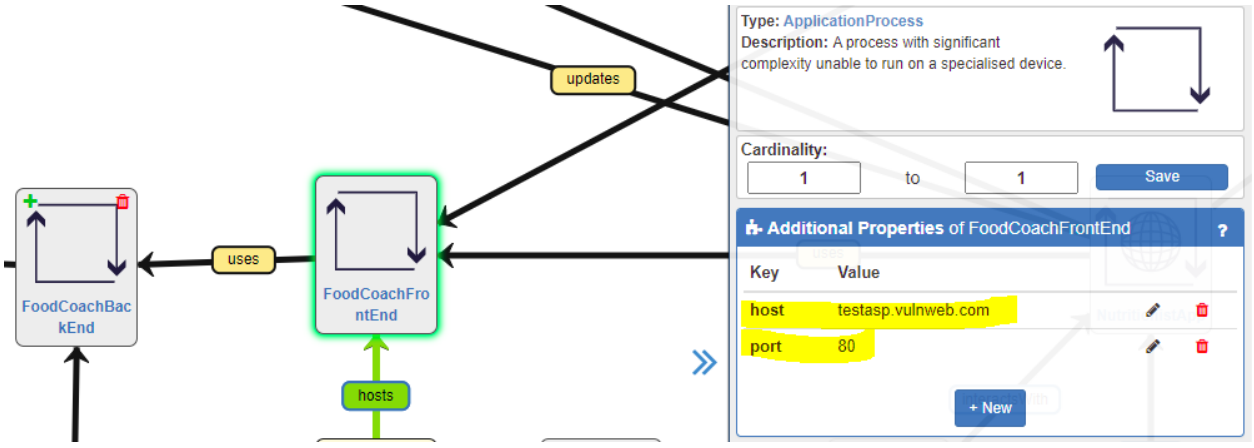


Figure III-3: Mapping ZAP Reports to Risk Modle Assets

There may be more than one web application in a scenario, in which case ZAP will identify more than one site with different *"host"* and *"port"* identifiers, and all relevant processes should be

identified and mapped as described above. Once all relevant web application process assets in the risk model have been mapped in this way, their TWAs will be adjusted according to the reported vulnerabilities. This is the only asset mapping required - the threat generation SSM analysis phase will propagate threats from the vulnerable web application assets to connected other assets.

III.3.3. Selection of Trustworthiness Attribute (TWA) & Level

This section describes how the appropriate TWA is selected and which level to set it to based on the ZAP report provided as input. The key information in the ZAP report are two ids, the “cweid” and the *“wascid”*, providing links to vulnerabilities in the *Common Weakness Enumeration (CWE)* and the *Web Application Security Consortium Threat Classification (WASC)* vulnerability classification schemes respectively. Both IDs are optional in ZAP reports, but they are provided in many cases and provide important information to guide the decision regarding which TWA to map and which level to set it to. Both CWE and WASC reports also provide links to other vulnerability reports that can provide mapping and level information, so the approach to map ZAP reports to TWAs and to find the levels is a depth-first search through the CWE or WASC reports (and the reports they link to) until the required information is found.

The algorithm firstly checks if the alert for a Common Weakness Enumeration (CWE) via the ZAP report’s *“cweid”* field, and then looks that up in the CWE mapping data to see if that CWE maps to the TWA information. If it does not, the algorithm next checks for the WASC report linked by the *“wascid”* in the ZAP alert and uses that with the WASC mapping data to search for corresponding *Common Attack Pattern Enumeration and Classification (CAPEC)*, *OWASP Top Ten 2010, SANS/CWE Top 25 2009 and CWE*. It searches each of these in turn for the necessary TWA information in the order just given. If that information is found in one of the mappings, then the searching process is stopped and if it is not then the next mapping in the order is checked. This process continues until either the required TWA information is found or all the mappings have been exhausted in the search.

III.3.4. Target TWAs

Several types of TWA are modelled in the SSM, as described in Table III-3.

Table III-3: SSM Trustworthiness Attributes

Extrinsic-A: ***(Availability)*** indicates the level of confidence that there are no vulnerabilities that can be exploited to crash or otherwise **degrade access to the target component**.
Extrinsic-AU: ***(Authentication Bypass)*** indicates the level of confidence that there are no vulnerabilities that allow **authentication to be bypassed**.
Extrinsic-C: ***(Confidentiality)*** indicates the level of confidence that there are no vulnerabilities that can be exploited to cause a **leak of data** from the target component.
Extrinsic-I: ***(Integrity)*** indicates the level of confidence that there are no vulnerabilities that can be exploited to **alter data** in the target component.
Extrinsic-M: ***(Management Rights)*** indicates the level of confidence that there are no vulnerabilities that can be exploited to give the **attacker management rights** on the target component.

Extrinsic-O: ***(Overload)*** indicates the level of confidence that there are no vulnerabilities that can be exploited to **overload or otherwise degrade performance of the target component.**
Extrinsic-QI: ***(Query Injection)*** indicates the level of confidence that there are no vulnerabilities that can be exploited to **inject queries into a back end database via the targeted component**.
Extrinsic-U: ***(Local User Access)*** indicates the level of confidence that there are no vulnerabilities that can be exploited to give the **attacker local user access** to the target component.
Extrinsic-VA: ***(Local Connection)*** indicates the level of confidence that there are no vulnerabilities that can be exploited by **connection from the local network**, below OSI Layer 3.
Extrinsic-VL: ***(Local Shell)*** indicates the level of confidence that there are no vulnerabilities that can be exploited given **local (physical or shell) access**.
Extrinsic-VN: ***(Remote Connection)*** indicates the level of confidence that there are no vulnerabilities that can be exploited from a **remote network connection**, i.e. at OSI Layer 3 or above.
Extrinsic-W: ***(Malware)***: freedom from vulnerabilities that would allow insertion of malware.
Extrinsic-XS: ***(Cross-Site Scripting)*** indicates the level of confidence that there are no vulnerabilities that can be exploited to **send malicious scripts to a browser used to access the targeted component**.

III.3.5. Vulnerability Classifications

III.3.5.a. Common Weakness Enumeration (CWE)

The CWE is a list of software and hardware weakness types, with weaknesses being flaws, faults, bugs, or other errors in the software or hardware.

CWEs can contain information on the *scope* and *likelihood* of exploit. It is not guaranteed that every CWE will have both scope and likelihood, but many provide this information. Scope provides information regarding which TWA and likelihood provides information on which level, e.g. a scope of “Confidentiality” and a likelihood of exploit of “High”.

Table III-4 below gives the mappings between CWE Scopes and the SSM TWAs. If the scopes of Confidentiality, Integrity, and availability are all present in the CWE then that maps to either Extrinsic-M or Extrinsic-U. Instead, if only one or two of these scopes are present then for each one present its corresponding TWA is mapped to. This results in combinations of Extrinsic-C, Extrinsic-I, and Extrinsic-A or Extrinsic-O being mapped to for these cases where these three scopes are not all present in the CWE. For the cases of Extrinsic-M or Extrinsic-U and Extrinsic-A or Extrinsic-O, which TWA to choose out of the two options is determined from the riskcode of

the ZAP alert, with riskcodes of 2 and higher (i.e. risks that are Medium or higher) giving the first and more severe TWA, and riskcodes less than two giving the second TWA.

Table III-4: Mappings of CWE Scopes to SSM TWAs

Scopes Present	***TWA***	***Reason (from D4.2)***	***Additional Selection Criteria***
(All present) Confidentiality, Integrity, Availability	Extrinsic-M (Management) or Extrinsic-U (User)	Exploitation leads to a complete loss of control of the affected component for Extrinsic-M, and a loss of access control, allowing user level access to the affected component, for Extrinsic-U.	Extrinsic M if ZAP Alert Riskcode >=2
			Extrinsic-U if ZAP Alert Riskcode <2
Confidentiality	Extrinsic-C (Confidentiality)	Attacker can read data stored in the affected component.	N/A
Integrity	Extrinsic-I (Integrity)	Attacker can make changes to data.	N/A
Availability	Extrinsic-A (Availability) or Extrinsic-O (Overload)	Attacker can shut down the affected component for Extrinsic-A, and can degrade the component availability for Extrinsic-O.	Extrinsic-A if ZAP Alert Riskcode >=2
			Extrinsic-O if Zap Alert Riskcode <2

The latest CWE list is available at [5]. This contains information for CWE weaknesses and categories in an xml format. This is processed to extract CWE Weakness Names and IDs, and TWAs and TWA levels which are obtained from scope and likelihood of exploit information within the list. The result is then exported to a CSV file that configures the Run Time Microservice to map CWE IDs to TWAs and TWA levels when processing ZAP alerts.

The list is also searched for CWE names and descriptions corresponding to cross-site scripting and query injection weaknesses and the resulting CWEs found are given TWAs of Extrinsic-XS and Extrinsic-QI respectively. This is done at the stage when the CSV file is generated, a step that occurs prior to deployment. During this stage, the scopes of each CWE are converted into their corresponding TWAs, the cross-site scripting and query injection TWAs are added in, and the CWE likelihoods of exploit are then converted into TWA levels. This results in the CWE-TWA mapping CSV file, which contains the CWE IDs, CWE names, affected TWAs, and new TWA levels.

A TWA level corresponds to the propensity of an asset to be affected by a threat of a certain type, so the higher the TWA level is, the less likely that it will be affected by threats of that type. Therefore, there is a *reflective relationship between any likelihood of a vulnerability reported by a CWE and a TWA's level* (i.e. low likelihood of the vulnerability corresponds to a high level of the mapped TWA), as shown in Table III-5. Hence, if the given CWE of a ZAP alert corresponds to a CWE with a known scope and likelihood of exploit, we can directly map that information to the TWA and TWA level respectively of an asset within the SSM. This is due to the CWE scope directly linking to a TWA of a given asset within the SSM and the likelihood relating to the TWA level through a reflective operation.

Table III-5: Relationship between likelihoods and TWA levels

Likelihood	***TWA Level***
Very Low	Very High
Low	High
Medium	Medium
High	Low
Very High	Very Low

If the likelihood of exploit of a CWE is missing but the scope is present, we can approximate the likelihood from the risk level of the ZAP alert. We do this by assuming a direct mapping between the risk and the likelihood. This relies on the assumption that there is a direct mapping between likelihood, impact, and risk (risk being dependent on both likelihood and impact). For the highest risk / likelihood levels this is a reasonable approximation as there are only a limited number of impact/likelihood combinations that can result in those highest risk levels. For risk levels that are lower there can be many impact/likelihood combinations that produce the given risk and so it can become less clear as to what the underlying likelihood is, though for these cases a direct mapping between the likelihood, impact, and risk is still assumed as it gives a balance between what the probable values of the likelihood and impact are.

Overall, there are 1292 CWE IDs (941 Weaknesses and 351 Categories). Out of those, Categories lack the information needed in the mapping and there are 718 Weaknesses that contain scope information. This results in a 56% coverage of supported CWE IDs, when considering all possible CWE IDs.

If the CWE scope information is missing for the given ZAP alert, or if the CWE is missing entirely from the ZAP alert, then we can instead look at the WASC ID within the ZAP alert, discussed next.

III.3.5.b. Web Application Security Consortium Threat Classification (WASC)

The Web Application Security Consortium Threat Classification Taxonomy Cross Reference View [6] provides mapping of WASC Attacks and Weaknesses to CWEs, CAPECs, SANS/CWE Top 25 2009, and OWASP Top Ten 2010, 2007, and 2004. An example snippet of this can be found in Table III-6 below.

Table III-6: Snippet of WASC ID mappings

WASC ID	*Name*	*CWE ID*	*CAPEC ID*	*SANS/CWE Top 25 2009*	*OWASP Top Ten 2010*	*OWASP Top Ten 2007*	*OWASP Top Ten 2004*
1	Insufficient Authentication	287	-	642	A3	A7	A3, A2
2	Insufficient Authorisation	284	-	285	A4	A10	A2
3	Integer Overflows	190	128	682	-	-	-
4	Insufficient Transport Layer Protection	311, 523	-	319	A9	A9	-

The WASC IDs provided within a ZAP alert do not give a direct way to map them to TWAs and TWA levels within the SSM. They can however be mapped to other metrics, which in turn can be mapped to TWAs and TWA levels. Using this, a WASC ID can be linked to these other metrics, which are each checked in turn for the required TWA mapping information. If none of the metrics searched contain the needed TWA information, then the WASC ID is not supported.

Not every WASC ID maps to each of metrics mentioned, though overall there is very good coverage. In total, there are 49 WASC IDs. Out of these 49 ID, 47 can be mapped to a metric that contains the necessary TWA information, giving a 96% coverage of supported WASC IDs.

III.3.5.c. Common Attack Pattern Enumeration and Classification (CAPEC)

The Common Attack Pattern Enumeration and Classification (CAPEC) (available at [7] provides a list of common attack patterns, which are descriptions of common attributes and approaches used to exploit known weaknesses. CAPEC can contain scopes and likelihoods of attack, and these can be mapped to TWAs and TWA levels. The scopes, TWAs, and reasons are the same as for the CWEs and can therefore be found in Table III-4.

Similar to the CWE list, the CAPEC list is an xml file format and is processed to extract CAPEC attack pattern Names and IDs, with the Scopes and Likelihoods of Attack then being used to determine their corresponding TWAs and TWA levels. This is also exported to a CSV file that contains the CAPEC IDs, names, affected TWAs, and new TWA levels. This file is read in and used for mapping CAPEC IDs to TWAs and TWA levels when processing ZAP alerts.

If scope is missing then the CAPEC cannot be linked to a TWA, though if the likelihood of attack is missing then it can instead be approximated to the risk level of the ZAP report.

III.3.5.d. 2009 CWE/SANS Top 25 Most Dangerous Programming Errors

The 2009 CWE/SANS Top 25 Most Dangerous Programming Errors [8] is a list of the most significant programming errors that can lead to serious software vulnerabilities.

This is a list of CWEs and so using the CWE ID linked to from the SANS/CWE-WASC mapping, the previously discussed CWE CSV file (and the mapping scheme in Table III-4) is used to obtain the TWA and TWA level information from a SANS/CWE Top 25 entry.

III.3.5.e. Open Web Application Security Project (OWSP) Top Ten 2010

The Open Web Application Security Project (OWASP) Top Ten 2010 [9] is a list of the top 10 web application security risks. Each OWASP entry contains a description of the security risk, an Attack Vector Exploitability value, amongst other things. The security risk description is used to infer the TWAs that would be affected by the given OWASP entry and the attack vector exploitability is used to determine the likelihood of the risk and so also the TWA level. The exploitability takes

one of the values Easy, Average, Difficult, and from this the TWA level becomes Low, Medium, High respectively. An snippet of the resulting CSV file that is formed from this is given in Table III-7 below.

Table III-7: Snippet of OWASP Top Ten 2010 entries to TWA and TWA level mappings

OWASP 2010 ID and Name	Attack Vector Exploitability	TWA	New TWA Level
A1 – Injection	Easy	Extrinsic-QI	Low
A2 – Cross-Site Scripting (XSS)	Average	Extrinsic-XS	Medium
A3 – Broken Authentication and Session Management	Average	Extrinsic-M or Extrinsic-U	Medium
A4 – Insecure Direct Object Reference	Easy	Extrinsic-C, Extrinsic-I	Low

It should be noted that the latest OWASP Top Ten is 2017 [10], but this version is not yet supported by WASC.

III.3.6. ZAP Mapping Algorithm

The ZAP mapping algorithm maps ZAP alerts to TWAs and TWA levels. Going through each alert contained within a ZAP report, it firstly checks if the alert for a CWE and then looks that up in the CWE mapping data to see if that CWE maps to the TWA information. If it does not map to the needed TWA information then the alert is checked next for a WASC ID and if present that is used with the WASC mapping data to search for corresponding CAPEC, OWASP Top Ten 2010, SANS/CWE Top 25 2009, and CWE entries. It then searches each of these in turn for the necessary TWA information. If that information is found in one of the mappings, then the searching process is stopped, and if it is not then further mappings are checked.

Inside the CWE mapping data, each CWE can map to none, one, or multiple TWAs. If the CWE maps to one or multiple TWAs then those are the TWAs that are to be affected, and if it maps to no TWAs then the other mappings need to be checked. The WASC mapping can map a single WASC ID to multiple CAPECS, SANS/CWE 2009s, or CWEs. If this is the case then all the mapped CAPECS, SANS/CWE 2009s, or CWEs are checked, and the one that results in the lowest TWA level is used (i.e. the “worst case”).

If TWAs have been found but TWA levels are missing, then the TWA levels are given from the riskcode using a reflective operation similar to that in Table III-5. After this, any TWA entries of “Extrinsic-M or Extrinsic-U" and “Extrinsic-A or Extrinsic-O" are reduced to one of their two options based on the riskcode level and the “Additional Selection Criteria” given in Table III-4.

After that the Extrinsic-VN TWA is set to the TWA level found for the ZAP alert. This TWA is set as it represents the level of confidence that there are no vulnerabilities that can be exploited through a remote network connection. Since ZAP scans are web vulnerability scans the vulnerabilities, they detect can all be exploited via remote network connections, hence Extrinsic-VN needs to be reduced to the found TWA level too.

ZAP scans can either be authenticated or unauthenticated scans. Authenticated scans detect vulnerabilities in which a malicious actor needs to authenticate to exploit the vulnerability, and unauthenticated scans detect vulnerabilities in which a malicious actor does not need to authenticate to exploit the vulnerability. If the ZAP alert comes from an unauthenticated ZAP scan, the Extrinsic-AU TWA is set to the TWA level found for the ZAP alert. This is because Extrinsic-

AU needing to be reduced if the given vulnerability can be exploited by an unauthenticated malicious actor.

After the TWAs have been found and set, they are stored and the process repeated for the next ZAP alert. At the end of processing a ZAP alert, the alert is only stored if the new TWA level is lower (and hence worse) than the currently stored level. The result of this is that once all the ZAP alerts in a report have been processed, a set of TWAs and TWA levels is obtained that contains all the affected TWAs from the ZAP report but only their lowest possible levels. This reduces the calls that need to be made to the SSM whilst producing the same result.

Once all the alerts in the ZAP report have been processed the affected TWAs of the given asset within the SSM are updated to their new, lower, values.

The main steps of the algorithm are as follows.

```
Loop through each alert in ZAP report site:
      IF alert confidence level is 0:
            Alert is false positive. SKIP ZAP alert.
      IF alert riskcode is 0 (i.e. informational):
            Alert is informational. SKIP ZAP alert.
      IF alert has a CWE ID:
            IF CWE-TWA mapping contains TWA information for CWE ID:
                SET TWA(s) to adjust to those in CWE mapping.
                IF CWE contains TWA level:
                      SET TWA level to this.
      IF alert has a WASC ID AND TWA not yet set:
            IF WASC ID maps to a CAPEC:
                IF CAPEC-TWA mapping contains TWA information for the
                CAPEC ID:
                      SET TWA(s) to adjust to those in CAPEC mapping.
                      IF CAPEC contains TWA level:
                            SET TWA level to this.
            IF WASC ID maps to an OWASP Top Ten 2010 AND TWA not set:
                IF OWASP 2010-TWA mapping contains TWA information for
                the OWASP 2010 ID:
                      SET TWA(s) to adjust to those in the OWASP 2010
                      mapping.
                      IF OWASP 2010 contains TWA level:
                            SET TWA level to this.
            IF WASC ID maps to a SANS/CWE Top 25 2009 AND TWA not set:
                IF CWE-TWA mapping contains TWA information for the
                SANS/CWE Top 25 2009 ID:
                      SET TWA(s) to adjust to those in the SANS/CWE 2009
                      mapping.
                      IF SANS/CWE 2009 contains TWA level:
                            SET TWA level to this.
            IF WASC ID maps to a CWE AND TWA not set:
                IF CWE-TWA mapping contains TWA information for the CWE
                ID:
                      SET TWA(s) to adjust to those in the CWE mapping.
                      IF CWE contains TWA level:
                            SET TWA level to this.
      IF TWA not yet set:
            ZAP alert not yet supported. SKIP ZAP alert.
      IF TWA level not yet set:
            SET TWA level from reflection of ZAP alert riskcode.
      IF TWA contains "Extrinsic-M or Extrinsic-U":
            IF riskcode >= 2:
                REPLACE with "Extrinsic-M".
            ELSE IF riskcode < 2:
```

```
                REPLACE with "Extrinsic-U".
        IF TWA contains "Extrinsic-A or Extrinsic-O":
                IF riskcode >= 2:
                    REPLACE with "Extrinsic-A".
                ELSE IF riskcode < 2:
                    REPLACE with "Extrinsic-O".
        SET "Extrinsic-VN" to the TWA level found.
        IF ZAP alert is from an unauthenticated ZAP scan:
                SET "Extrinsic-AU" to the TWA level found.
        IF TWA(s) have level(s) lower than the stored levels:
                UPDATE stored levels to these.
IF stored TWA level(s) lower than SSM asset level(s):
        UPDATE SSM asset TWA level(s) to stored alert level(s).
```

III.3.7. Mapping Examples

III.3.7.a. Example 1:

Here a simple SSM model with an application process being the asset under consideration is given. All TWAs but the Default-TW and Extrinsic-TW TWAs start off with values of "Very High", as seen in Figure III-4.

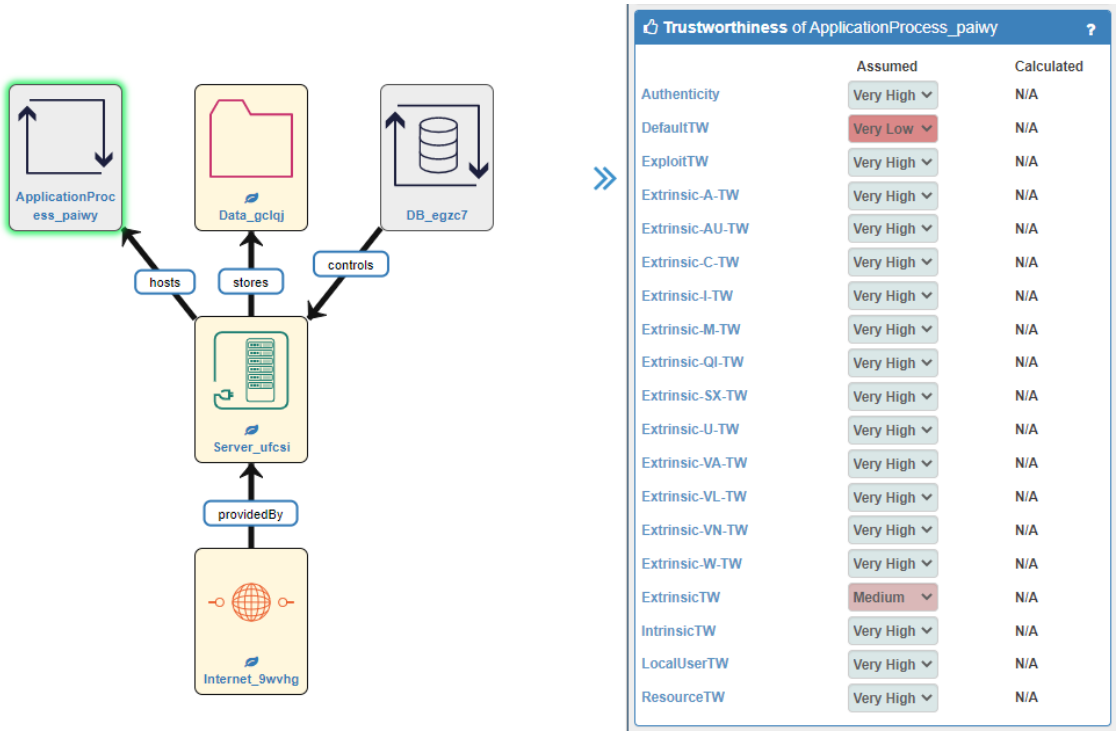


Figure III-4: Starting TWA values for ZAP mapping algorithm example 1.

An example authenticated ZAP report is passed in to the microservice that contains two ZAP alerts, the first with a CWE ID of 298 and the second a CWE of 311. The effects of these CWEs on the asset TWAs can be found in Table III-8.

Table III-8: TWA changes for ZAP mapping algorithm example 1

CWE ID	TWAs to Reduce	New TWA Levels
298	Extrinsic-I, Extrinsic-VN	High

311	Extrinsic-I, Extrinsic-C, Extrinsic-VN	Low

The microservice processes the alert containing CWE 298 first, reducing Extrinsic-I and Extrinsic-VN TWAs to values of “High”. The alert containing CWE 311 is processed next by the microservice, which overwrites the previous New TWA Levels with the value of “Low” whilst also changing Extrinsic-C to the value of “Low” too. The result is shown in Figure III-5.

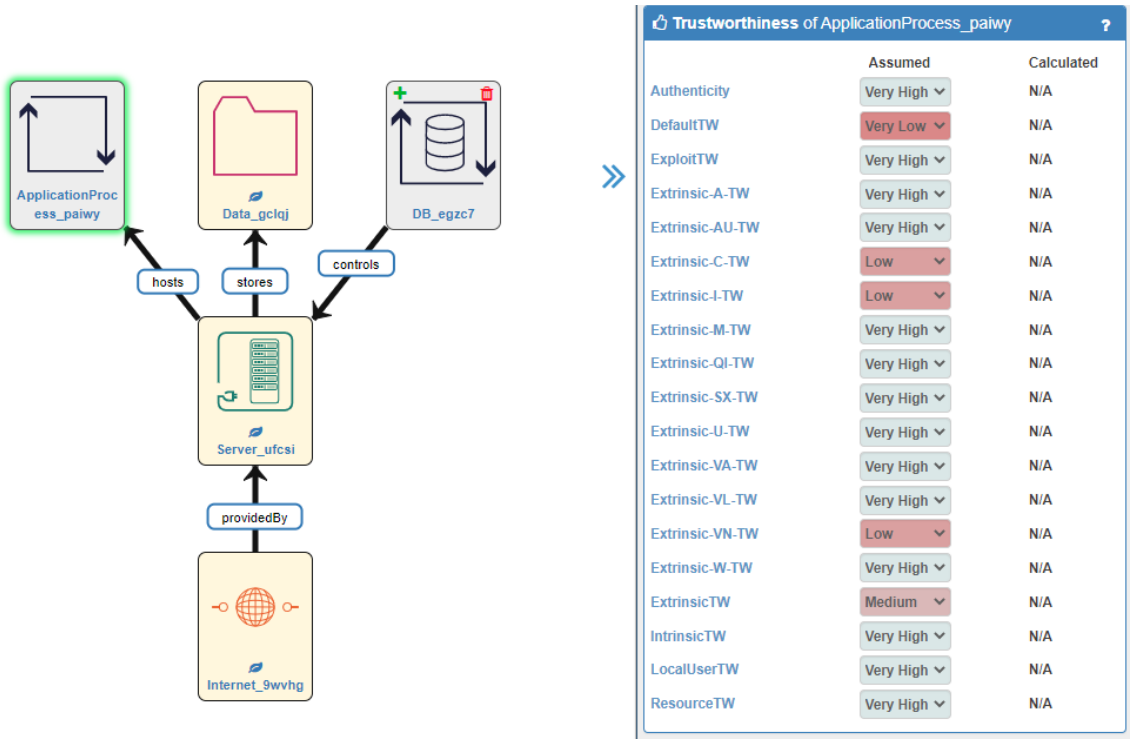


Figure III-5: Ending TWA values for ZAP mapping algorithm example 1.

III.3.7.b. Example 2:

In the next example, the starting point TWAs are as shown in Figure III-6.

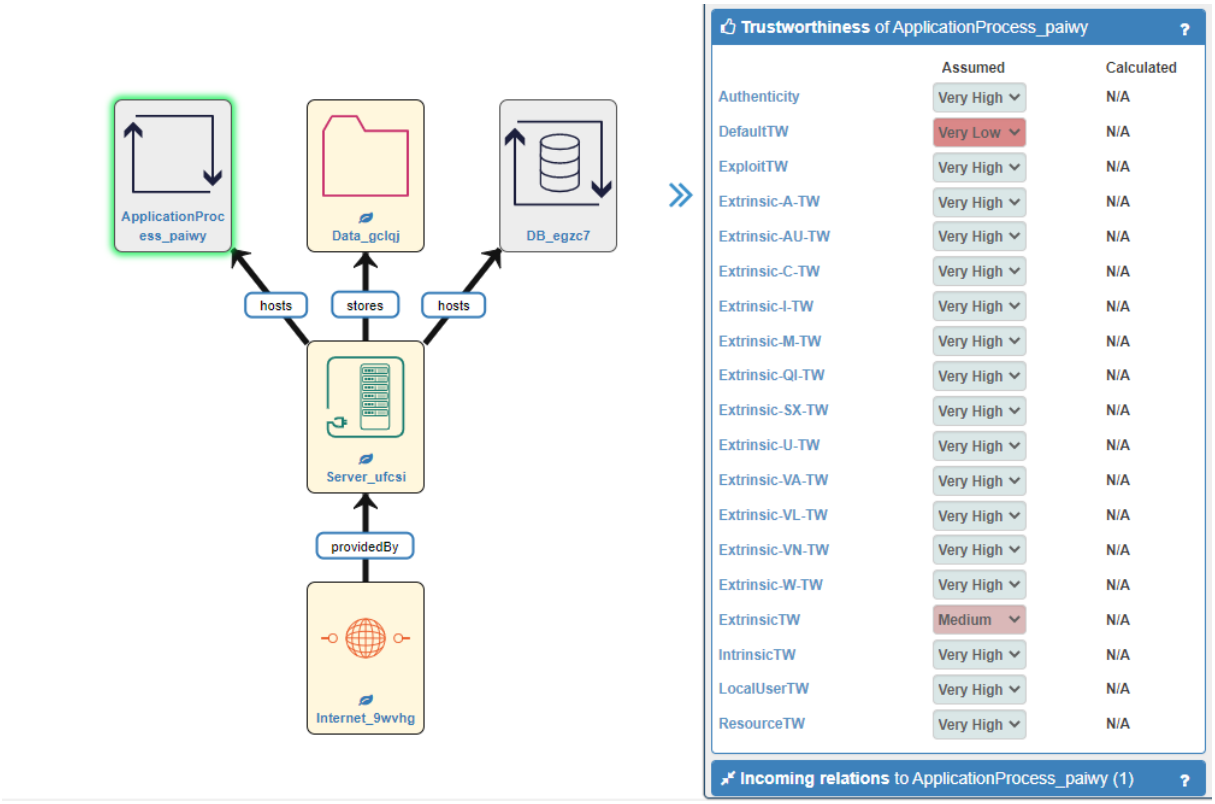


Figure III-6: Starting TWA values for ZAP mapping algorithm example 2.

An unauthenticated ZAP report is passed into the microservice with four ZAP alerts. The first three alerts either do not contain a CWE ID, or the CWE ID they do contain does not map to the needed TWA information. The final alert does map to the needed TWA information though it also lacks a value for the New TWA Level.

Table III-9 shows the different metrics mapped to for each of the ZAP alerts, as well as the ZAP riskcodes, the affected TWAs, and the TWA levels found for each of the alerts. In this table, each row gives a new ZAP alert and the resulting TWAs and TWA levels obtained for that alert. Following this table is a description of how the information in each alert is processed and how they build up to form the set of TWAs and TWA levels that the SSM asset TWAs are reduced to.

Table III-9: TWAs affect and TWA levels from the four ZAP alerts of the ZAP mapping algorithm in example 2

ZAP Alert #	*ZAP Risk code*	*CWE ID*	*WASC ID*	*CAPEC ID*	*OWASP 2010*	*SANS/ CWE 2009*	*TWA to Reduce*	*TWA Level From Alert*
1	1	16	15	-	A6	-	Extrinsic-U, Extrinsic-VN, Extrinsic-AU	Low
2	3	-	12	148	-	-	Extrinsic-I, Extrinsic-VN, Extrinsic-AU	Medium
3	2	16	13	118	-	209	Extrinsic-C, Extrinsic-VN, Extrinsic-AU	Low

4	2	829	-	-	-	-	Extrinsic-M, Extrinsic-VN, Extrinsic-AU	(From riskcode) Medium

Alert 1: For the first ZAP alert the CWE 16 is found. This CWE does not contain the needed TWA information and so the alert is searched for a WASC ID, which is found to be WASC ID 15. The WASC mapping is then checked for a CAPEC ID, which it cannot find, and then for an OWASP Top Ten 2010, which it does find. The OWASP 2010 found is A6, which gives the TWAs to reduce as Extrinsic-M or Extrinsic-U, Extrinsic-VN, and Extrinsic-AU. To decide whether to reduce Extrinsic-M or Extrinsic-U, the ZAP alert riskcode is used and, as it's less than 2, the TWA is determined to be Extrinsic-U. The result of this ZAP alert is that the new values for Extrinsic-U, Extrinsic-VN, and Extrinsic-AU become "Low".

Alert 2: For the next ZAP alert in the table there is no CWE ID but there is a WASC ID of 12 and so that is used. A CAPEC with the required information is first searched for and found, and this results in Extrinsic-I, Extrinsic-VN, and Extrinsic-AU TWAs being found with the level of "Medium". However, as "Medium" is not lower than "Low" only the Extrinsic-I TWA is reduced.

Alert 3: The third ZAP alert contains a CWE ID, though this CWE does not have the required information and so a WASC ID is looked for, and the WASC ID 13 is found. From this WASC ID and the WASC mapping table, a CAPEC is first found though this does not have the needed TWA information. An OWASP 2010 is then looked for, though there is none linked. A SANS/CWE Top 25 2009 is finally searched for and found. This SANS/CWE of 209 is a CWE that has the required TWA information and results in Extrinsic-C being lowered to "Low", with the Extrinsic-VN and Extrinsic-AU TWAs also being found but they are again not reduced as the TWA level found is "Low" which is not worse than their current values of "Low".

Alert 4: The final ZAP alert contains the CWE ID 829, which does have the needed TWA information and so the TWAs Extrinsic-M or Extrinsic-U, Extrinsic-VN, and Extrinsic-AU are found. However, it is missing the TWA level and so the riskcode is used to approximate the TWA level from the reflection of the riskcode. The riskcode has a value of 2 and so by following the "Additional Selection Criteria" of Table III-4 "Extrinsic-M or Extrinsic-U" is reduced to give the Extrinsic-M TWA. Overall, this alert results in a reduction to the Extrinsic-M TWA to the level of "Medium" as the Extrinsic-VN and Extrinsic-AU TWAs are not changed since their levels from this alert are not lower than their levers from the other alerts.

The resulting set of TWAs and TWA levels are shown in Table III-10 and are passed into the SSM to update the TWAs of the affected asset.

Table III-10: The overall TWAs and TWA levels from processing all four ZAP alerts of the ZAP mapping algorithm in example 2

TWA Changed	*New TWA Level*
Extrinsic-U	Low
Extrinsic-VN	Low
Extrinsic-AU	Low
Extrinsic-I	Medium
Extrinsic-C	Low
Extrinsic-M	Medium

The final result is shown in Figure III-7.

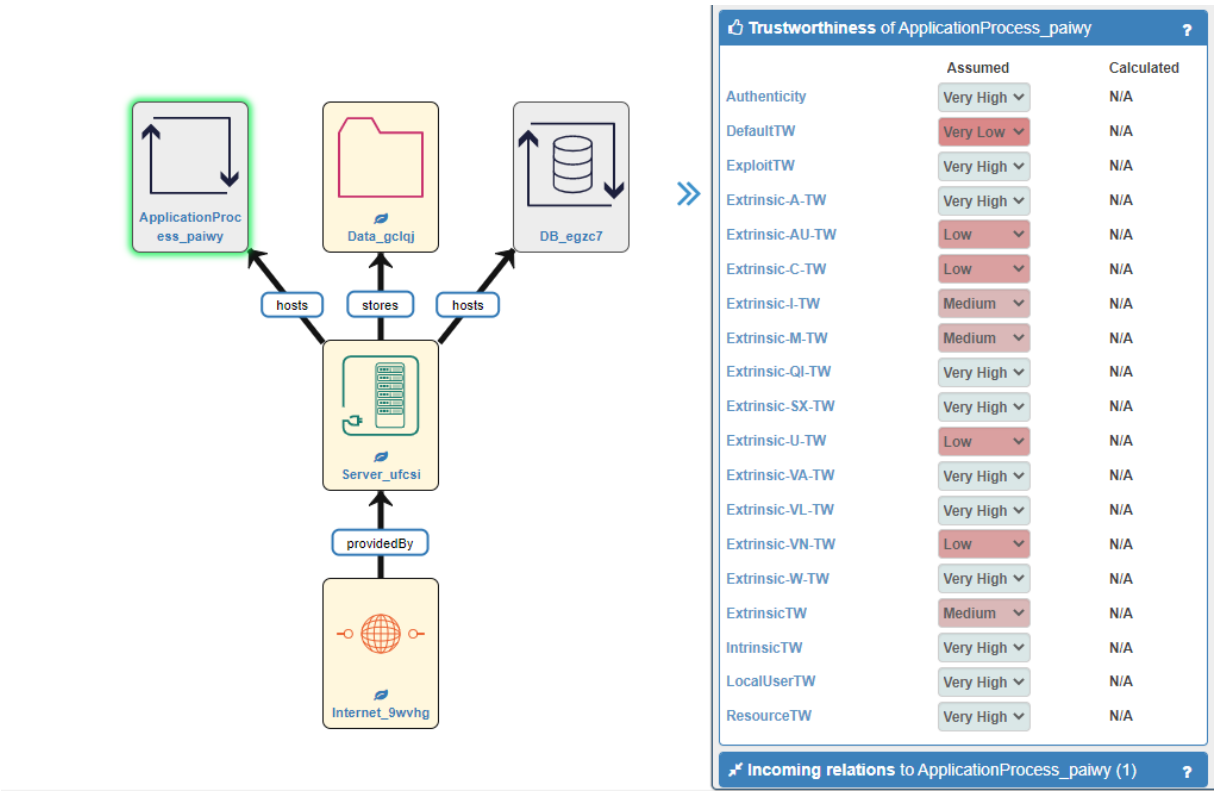


Figure III-7: Ending TWA values for ZAP mapping algorithm example 2.

III.4. Risk Mitigation Recommendations

III.4.1. Algorithm

The recommendations algorithm aims to find controls that can reduce the overall risk in the risk model. The SSM's knowledge base already suggests mitigating controls based on the threats identified, but the requirement is to determine effective combinations of these controls and the resulting risk level if they were all applied. The approach to address this requirement is to automatically conduct "what-if" experiments, where controls are applied, and the resulting risk level determined based on the application of these controls in the given risk models. These "what-if" experiments are conducted automatically and progressively by the Run Time microservice, where a control is kept if it reduces the overall risk and discarded if it does not. Through iteration over multiple experiments, a set of controls is built up, and these are reported to the system administrator, along with the expected risk level if those controls were applied to the operational system.

Part I: Identify which threats to explore

In order to provide effective recommendations (in terms of risk reduction) at reasonable cost in term of computation time, a subset of threats with maximum impact is chosen. The selection criteria initially are:

- Threats with valid *risk_level* attributes – i.e. threats with direct links to misbehaviours that determine risk. Some threats make the system more vulnerable to other threats, but the type of threat we are concerned with here are those that directly cause misbehaviours in assets, which lead to risk.
- Threats that have *root_cause* attribute – i.e. threats not caused by the effects of other threats. Mitigating root cause threats is typically much more effective than mitigating other

types of threats as an initial strategy because, as their name suggests, root cause threats can trigger many downstream threats and mitigating a root cause threat can therefore significantly reduce the number of downstream threats.

The resulting subset of threats is then sorted based on the risk level they cause (i.e. Very High, High, Medium, Low and Very Low), and within these bands, the risks are ordered by their impact (i.e. the user-defined severity of the misbehaviour). This ordering is used by the recommendations algorithm, with the highest level and most impactful risks used first in tests to find controls. In order to save computational time, an optional limit on the number of threats addressed in one recommendation cycle may be applied to limit the number of threats evaluated.

Part II: applying controls on threat control strategies, cumulative progressive algorithm

The key principle is to try control strategies that are recommended by the SSM's knowledge base in response to a threat and keep those controls that improve (i.e. reduce) the risk level. Each control strategy is tested by activating its controls in the risk model, and the resulting risk level is calculated. Controls that improve the risk level are kept in the model and those that do not improve the risk level are deactivated. This process iterates until all selected threats have been tested, and at this point, the recommendation report contains all controls that reduce the risk level. Finally, at the end of the algorithm, all controls applied to the model are deactivated because they are not activated in the operational system – the controls are contained in recommendations report, and it is the decision of the system administrator as to which controls to apply.

The main steps for the algorithm to determine recommendations are described in pseudocode as follows.

```
for each of the selected root threats {
  for each of the control strategies suggested for that threat {
    activate all controls in control strategy that are not already
    activated
    if there are newly activated controls {
      calculate the FUTURE risk of the model, i.e. the effect on
      risk level of enabling the controls
      if FUTURE risk < current risk {
        store newly activated controls in cache
        add a recommendation in report with:
        - all currently activated controls stored in cache
        - the FUTURE risk
        set current risk = FUTURE risk
      }
      else {
        deactivate newly activated controls
      }
    }
  }
}
deactivate all stored controls in cache and revert model to its
initial state
return recommendations report
```

III.4.2. Example

The following is an example of a recommendations output from the FoodCoach end-user case. This example shows three recommendations, each of which adds one control and successively improves the risk level.

```
{
```

```
"CurrentRisk": "Very High",
"CurrentRiskVector": {
  "VeryHigh": 5,
  "High": 2,
  "Medium": 6,
  "Low": 140,
  "VeryLow": 932
},
"ObjectToIdentify": null,
"Recommendations": [
  {
    "RecommendationID": 1,
    "Action": [
      {
        "Control": "FWBlocked",
        "ControlAsset": {
          "Action": "BLOCK",
          "AssetLabel": "LogicalSegment-Internet-OSRRouter-
          HospitalLAN",
          "AssetType": "LogicalSegment",
          "ControlLabel": "FWBlocked",
          "Uri": "calSegment_9602a801_63ee81f3_ee930cbb"
        }
      }
    ],
    "FutureRisk": "Very High",
    "FutureRiskVector": {
      "VeryHigh": 5,
      "High": 2,
      "Medium": 4,
      "Low": 129,
      "VeryLow": 945
    }
  },
  {
    "RecommendationID": 2,
    "Action": [
      {
        "Control": "FWBlocked",
        "ControlAsset": {
          "Action": "BLOCK",
          "AssetLabel": "LogicalSegment-Internet-OSRRouter-
          HospitalLAN",
          "AssetType": "LogicalSegment",
          "ControlLabel": "FWBlocked",
          "Uri": "calSegment_9602a801_63ee81f3_ee930cbb"
        }
      },
      {
        "Control": "NAT",
        "ControlAsset": {
          "Action": "BLOCK",
          "AssetLabel": "LogicalSegment-ServiceSubnet-OSRRouter-
          Internet",
          "AssetType": "LogicalSegment",
          "ControlLabel": "NAT",
          "Uri": "calSegment_a1ab69fc_63ee81f3_9602a801"
        }
```

```
        }
      ],
      "FutureRisk": "Medium",
      "FutureRiskVector": {
        "VeryHigh": 0,
        "High": 0,
        "Medium": 5,
        "Low": 10,
        "VeryLow": 1070
      }
    },
    {
      "RecommendationID": 3,
      "Action": [
        {
          "Control": "FWBlocked",
          "ControlAsset": {
            "Action": "BLOCK",
            "AssetLabel": "LogicalSegment-Internet-OSRRouter-
            HospitalLAN",
            "AssetType": "LogicalSegment",
            "ControlLabel": "FWBlocked",
            "Uri": "calSegment_9602a801_63ee81f3_ee930cbb"
          }
        },
        {
          "Control": "NAT",
          "ControlAsset": {
            "Action": "BLOCK",
            "AssetLabel": "LogicalSegment-ServiceSubnet-OSRRouter-
            Internet",
            "AssetType": "LogicalSegment",
            "ControlLabel": "NAT",
            "Uri": "calSegment_a1ab69fc_63ee81f3_9602a801"
          }
        },
        {
          "Control": "SoftwarePatched",
          "ControlAsset": {
          "Action": "BLOCK",
          "AssetLabel": "FoodCoachServer",
          "AssetType": "VM",
          "ControlLabel": "SoftwarePatched",
          "Uri": "ca0b"
          }
        }
      ],
      "FutureRisk": "Medium",
      "FutureRiskVector": {
        "VeryHigh": 0,
        "High": 0,
        "Medium": 5,
        "Low": 9,
        "VeryLow": 1071
      }
    }
  ]
```

}

The starting point (current risk, reported at the top of the example) shows a vector of {"VeryHigh": 5, "High": 2, "Medium": 6, "Low": 140, "VeryLow": 932}.

Recommendation 1 suggests an FWBlock Control applied to the logical segment across the path Internet-OSRRouter-HospitalLAN. This has marginal effect, as indicated by its FutureRiskVector: { "VeryHigh": 5, "High": 2, "Medium": 4, "Low": 129, "VeryLow": 945 }, where only the “medium” and “low” level risks are reduced. The worst case risk is still “VeryHigh”.

Recommendation 2 adds a NAT control on the LogicalSegment OSRRouter-Internet, additional to the FWBlock of Recommendation 1. This has a major effect, as illustrated by its FutureRiskVector: { "VeryHigh": 0, "High": 0, "Medium": 5, "Low": 10, "VeryLow": 1070}. Here, all the “VeryHigh” and “High” level risks have been mitigated, leaving the worst-case risk at the “Medium” level.

Recommendation 3 adds a Software Patching control on the FoodCoach server, in addition to the controls in Recommendations 1 and 2. This results in an improvement in the risk level, but it is marginal – the FutureRiskVector results in a reduction of a single “Low” level risk: { "VeryHigh": 0, "High": 0, "Medium": 5, "Low": 9, "VeryLow": 1071}.

From these recommendations, a system administrator can evaluate the relative benefit of progressively applying controls and then determine which recommendation to implement. Given this example, a reasonable choice may be Recommendation 2 – this has a significant risk reduction over Recommendation 1, but Recommendation 3 provides very marginal additional risk reduction.

IV. Domain Model Enhancements

IV.1. Overview

The starting point was the domain model described in ProTego D4.2, as demonstrated at the mid-point review meeting at the start of September 2020. As discussed in D4.2, this covered all the usual types of cyber and cyber physical attacks, with features developed in ProTego to model IoT devices as a combination of device + service + data, elaborated models for the presence and exploitation of software vulnerabilities, models of network security for protection of network slices, and side effects of encryption especially for query processing some of which could be overcome by using Parquet encryption.

As before, the model released as D4.3 contains enhancements developed in ProTego to address specific areas needed in the project, and work from other projects that happen to be of use in ProTego. In D4.2, several areas of further work were identified:

- The model of Parquet encryption was considered sufficient, and no further work was planned.
- Modelling the ProTego Data Gateway was covered, with some potential improvements possible: these have not been pursued, but work in other projects (notably H2020 FogProtect) make the existing approach slightly more effective.
- Key management by a service could be modelled following an EAP-style approach, but the models for Bluetooth SSP were primitive, and some improvements have now been implemented.
- Network slice isolation was covered by the generic access control models, but there was no model for interference between loads imposed on different slices: threats have been added to cover this.
- Continuous authentication was covered, but the parameters (which should be based on the sensitivity and accuracy of the algorithm) had not been calibrated. This is a trivial change which has not yet been made pending an assessment of the algorithm in a realistic setting.

In practice, much of the development up to D4.2 was on modelling IoT and ProTego-specific controls. Since D4.2, the priority has shifted towards BYOD aspects. The cyber-physical threat models in D4.2 do address BYOD, but attack paths constructed from these threats typically overestimated the potential risks. Fixing that became important and took priority over some of the enhancements previously identified, notably for slice isolation where the threat and control models are implemented, but in a less refined form than envisaged.

The focus of new development since D4.2 are in the following areas:

1. A model of attacker context has been added, capturing the means of (physical or network) access and the consequent limits to privileges gained, making attack paths more realistic.
2. An improved model of physical theft, capturing the ‘detachment’ of a stolen device from its system context, and taking account of potential ‘reconnection’ threats.
3. Inference rules for localisation of mobile/pervasive and virtual networks, maintaining the relationship to their underlying physical nature and infrastructure.
4. Inference rules for data centre resources and a Pod + Container management architecture. This was developed mainly in H2020 CyberKit4SME but relevant for cloud-based elements of ProTego.
5. A model of asset disablement: this was started in ProTego and reported in D4.2, but this was switched to H2020 FogProtect, elaborating the model so it is easier to use in Fog environments.

6. An improved data lifecycle model, partly so the new disablement model can be applied to data flows, also done in FogProtect but involving some changes to IoT models previously developed in ProTego.
7. Extra threat models, mostly related to physical threats needed in FogProtect, but some are relevant to ProTego scenarios.
8. Refined models of control strategies, including a naming convention such that the preparation (design time) and activation (run time) elements of contingency plans can easily be identified as related.
9. Inclusion of more inference rules to fill in details and reduce the number of direct assertions that SSM users need to include, especially in the physical layer of their system models.

The first three were developed specifically for (and in) ProTego, motivated by the need to better model risks from BYOD, and are discussed in some detail in the following sections. The last included some work done in ProTego related to BYOD, which is summarised. The other enhancements coming entirely from other projects are described briefly where relevant to the work done in ProTego.

IV.2. Host and Process Contextualisation

One of the drawbacks of the models from D4.2 was the fact that attacks were assumed to have 'system wide' scope and context. This has the advantage that SSM will overestimate rather than underestimate the risks, which is obviously desirable when using SSM to detect risks that may be overlooked in a manual analysis. However, it can produce unrealistic attack paths, e.g. if an attacker can access a mobile phone in a café well away from the hospital, the old model predicted they could exploit rights assigned to the user that are really only accessible inside the hospital. This is of most relevance to BYOD devices, where the device does not stay in a secure environment but may be used at home or in a public place.

To provide a more discriminating approach to modelling these effects, a new logical asset type was added that represents the location of a device. The domain model now contains rules to insert these assets as necessary through a construction pattern as shown in (Figure IV-1). The graphical notation is the same as that used in D4.2, so the double boundary indicates that the Host, Logical Subnet and Space assets are unique matches within the system, and the Host Context is constructed for each unique combination of matches.

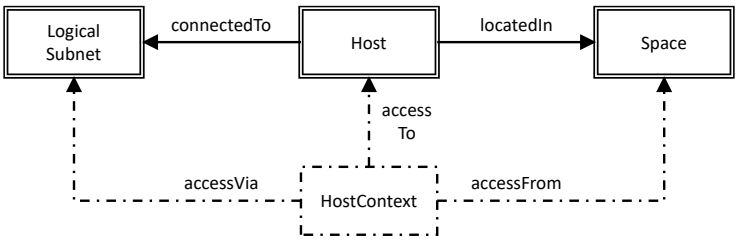


Figure IV-1. Device Contextualisation

The main drawback with this approach is that threat models become more complex, and the number of distinct threats increases as we need (in the worst cases) one threat per location and network connection. To mitigate this effect, network access context assets were introduced to aggregate information per network connection. Where a host context is related to one host in one location and potentially many networks, the network context is related to one host on one network and potentially many locations. This makes it possible to eliminate either factor (locations or networks) wherever possible to reduce the number of threats. Models are still significantly larger than they were before this model of context was added, but experiments indicate they should be workable. It is possible to optimise further, and this will be done during the final ProTego validation phase if performance causes serious problems for the case studies.

Threats that represent physical or network attacks to take control or access a host now refer to these context assets. For example, a local exploit of a vulnerability in a Process (CVSS access vector = 'Local') requires user level access to the Host. Previously the threat involved only the Host and Process, the causation term was User TW (Trustworthiness) at the Host, and the effect was only at the Process. Now, the cause is based on a host location context, and the effect arises in the process location context for the same host location, and all related network contexts related to that location, as shown in Figure IV-2:

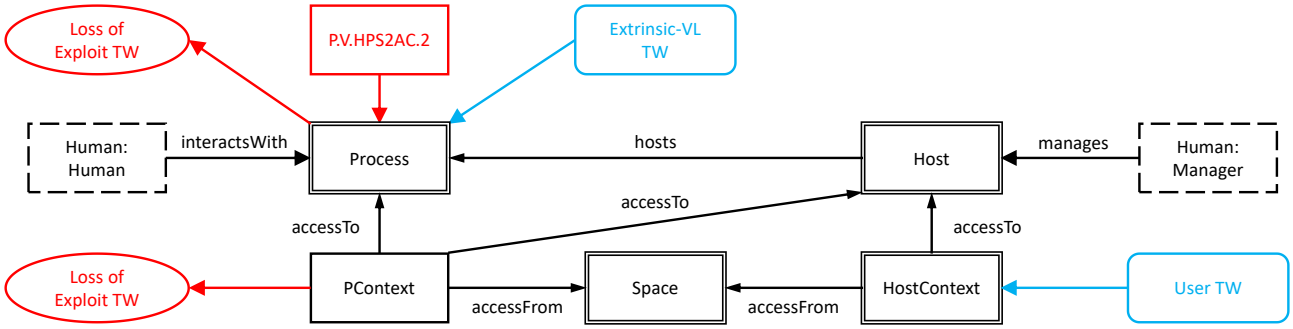


Figure IV-2. Local exploit of a Process vulnerability via user access to its Host

The effect in this case erodes 'Exploit TW', representing the attacker having access to the vulnerable code. The PContext class is the parent class for both location and network process context assets, and the single boundary indicates this is a non-unique node, so in this case the Process Context related to the space would be affected, along with every Process Network Context linked to that space. The optional (dotted) humans are included in the pattern because they can play a role in some control strategies against this threat.

If the host is mobile, what happens in one location is thus kept separate from what could happen in other locations. A PocketEHR app that accesses a service from a secure LAN in a hospital would not necessarily have access to the same service when its host was being used in an Internet café. That depends on whether there is a network path between the two locations that is not blocked by firewall policies at intervening devices. If there is no such path, then SSM would be able to determine that grabbing the phone in the Internet café does not give the attacker access to the hospital's service.

IV.3. Theft and device possession

Contextualisation raises the question of what happens if a device is stolen. If an attacker can control a device while it is performing system functions, they can corrupt its behaviour and/or misuse its privileges to cause extensive disruption. Stealing exploits insecure configurations, weak passwords, or vulnerabilities. If no such weaknesses can be found, an attacker can block any patching mechanisms and wait until a suitable vulnerability is discovered. Stealing a device removes it from the system, disconnecting it from its former context and limiting the opportunities to cause harm. For example, taking control of a device in either sense allows an attacker to read data stored on the device, but altering this data can only affect the system if the device is still connected. On the other hand, once an attacker has a device in their possession, they can dismantle it to access its data, gain control of the intact device by various means, and try to reconnect to the system, regaining some of the opportunities that were prevented by its removal.

To model these effects, new trustworthiness attributes were introduced as follows:

- Possession: an attribute reflecting the trustworthiness of whoever has the device, which is degraded by a threat representing theft of the device by an attacker who has physical access in its location.
- Local Control: like the existing Control attribute, this represents admin rights on a Host, but with a restriction to the device itself.

- Local User Trustworthiness: like the existing User Trustworthiness for a Process, this represents rights assigned to the process (i.e. to the user account running the process), but also limited to the device.

Once a device has been stolen, its (degraded) Possession attribute becomes a cause of other threats that model what the attacker can now do with the device. One threat covers dismantling the device to access its data storage media, which leads to no new privileges but destroys data confidentiality. Other actions depend on first gaining privileges, so several threats represent ways to do this, exploiting Possession to undermine Local Control on the device or Local User TW for hosted processes.

A successful attack undermines the Local Control or Local User TW attribute, enabling further threats that model reading (but not altering) data stored on the device directly or via a process (including decryption if the process has the necessary keys). Unlike the regular Control or User TW attributes, these privileges do not automatically propagate to associated location and network access contexts because those relate to a device in the system, which the stolen device is not. However, new threats have been added to model attacker actions to reconnect the device and use privileges assigned to it or to processes running on it.

IV.4. Network localisation

The context models make it possible to maintain a separation between privileges gained in one context and effects in unrelated contexts. However, where a network is accessible from multiple locations, it can form a bridge between those locations, which allows attack paths to form that may still be unrealistic. There are four cases where this arises:

1. The Internet: is accessible everywhere, creating connections between every pair of location contexts.
2. Cellular networks: are also accessible from multiple locations, creating connections between contexts.
3. Networks provided by mobile devices: may be accessible in each location where the device will go, e.g. a WLAN hotspot provided by a mobile phone, or a Bluetooth or USB pairing connection.
4. Networks internal to a host: also exist in each location where the device exists, e.g. the LAN provided by a virtual switch on a device that hosts VMs.

The Internet is a special case, because although it is publicly accessible, connection to it is a service that must be procured in each location where it is required. When we say a device connected to the Internet, what we mean is that it connected to a router that was connected via an Internet Service Provider (ISP). It is therefore reasonable to impose a restriction that only fixed devices can connect to the Internet, and location contexts are only linked to networks (at least in the first instance) for devices that are connected to the network. For the Internet, only (fixed) routers in each location do this, so no bridges between locations are created.

For cellular networks the same issue arises, and since the devices connected to cellular networks usually are mobile (at least in ProTego scenarios), they do have multiple location contexts. The pervasive cellular network thus forms a temporary bridge between locations. Of course, a cellular network (like the Internet) is not really one network accessible from every location. Devices in different locations connect to a distinct radio access networks, each provided by a router (a base station) connected to the core network. To emulate this, the approach taken in the new domain model is to construct (by inference) this core + access network topology, using construction patterns including the one from Figure IV-3:

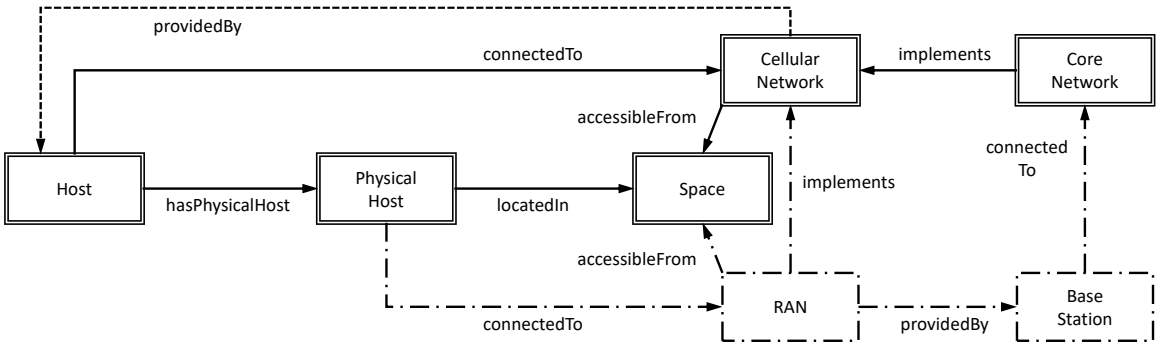


Figure IV-3. Construction of localised cellular network infrastructure

The dotted 'providedBy' link (dotted meaning that this link must not exist for a match) ensures this pattern is triggered by connections from supplicants, and not from the router providing the cellular network, which may exist in system models where the cellular network has a gateway linking it to some other network. The critical aspect of this pattern is that the IDs generated for the constructed Remote Access Network (RAN) and Base Station assets depends on the Space, so we get a distinct RAN per Space. In addition, the cellular network class has been reclassified as an 'abstract' network, so the connection of hosts to it are not interpreted as communication links – only the constructed links to the localised RAN have that status.

For mobile hotspots and pairing connections, a similar approach is used. For pairing connections this was quite simple, because the 'logical subnet' was already constructed based on the host-to-host pairing link to represent the Bluetooth or USB communication path between them. The construction pattern was altered so this 'subnet' is only constructed if both devices are in the same Space, and its ID depends on the Space.

Mobile hotspots present more challenges because a hotspot is represented as a WLAN provided by a mobile host. Such a WLAN is only a hotspot if the host may be in multiple locations, so we can't reclassify a WLAN as an abstract network and negate host connections to it. The mobile host may also have different connections to other networks in each location, so we need to generate different routes in and out of the hotspot. To cover all the alternatives, a sequence of construction patterns was needed:

- to detect if the WLAN provided by a mobile host is a hotspot, and if so mark it by adding a distinctive link.
- to generate a localised replica for each space served by each hotspot.
- to create connections to each replica from other devices present in each location.

Other patterns then had to be adapted to exclude the hotspot, including those used to generate logical routes through gateways (which now only create routes between subnets accessible in the same location), and for threats of attacks from or on WLANs (to exclude those that are hotspots).

The last type of network that creates bridges in this way are internal networks created within hosts to support VMs. This was addressed outside ProTego because creation of these networks forms part of the approach for modelling autonomic management of virtualised infrastructure using tools like Kubernetes. In ProTego, the only change is to impose the constraint that users should not define these internal networks explicitly in system models – they should leave this to the virtualisation inference patterns which will then handle the localisation issues.

IV.5. Other changes related to ProTego

IV.5.1. Slice isolation

The virtualised infrastructure patterns drawn from elsewhere also provided support for adding the ProTego model of network slice isolation. This is orthogonal to the issue of isolation between location contexts, but both involve finding the relationship between a virtual network and the underlying physical infrastructure.

The two situations pose opposite challenges: where before we had to eliminate bridges between locations to prevent spurious attack paths, here we need to insert bridges between network slices in secondary effect cascades where the slices share the same infrastructure. This is far easier to do, as it just requires a new threat (with suitable control strategies), which was added to the D4.3 version in ProTego.

IV.5.2. Data lifecycle model upgrades

The model of asset disablement added in ProTego D4.2 was substantially overhauled in H2020 FogProtect, to address dynamic threats arising within an automated Fog environment. Conceptually, the main change was to move away from using controls representing disablement of each type of asset, and towards a model of interdependency so a disablement control on one asset could affect other assets. The goal of this work in FogProtect was to support modelling of dynamic restrictions on data flows by changing policies for resource allocation and (virtual) network connectivity as well as directly disabling individual data flows.

The implications of these changes for ProTego fall into two areas:

- The new data lifecycle model includes new constructed asset and behaviour types representing access to data by processes, replacing those used in the encryption side effect threats introduced in ProTego D4.2.
- The changes in the model of the relationship between processes and data conflicts with the original model of IoT devices from ProTego D4.2.

The first issue was simple to fix. The threat models just needed changing to use the new asset types and express their effects in terms of the new behaviour types. These then feed into the dependency models from FogProtect, producing the correct outcomes in terms of the inability to access data if encryption measures are not consistently applied.

The second issue was also not difficult to fix but it required more significant changes to the original IoT model. In D4.2, an IoT device triggered a construction pattern that added onboard process and data assets. The Thing thus becomes represented by all three elements: the device represents a host with built in elements that can sense or affect the physical environment, the process for handling communication and mediating access to the data, allowing reading of sensor data or writing of control data. Relationships with a Thing were the same as any other Host, plus some extra Process-Process relationships which were transferred by construction rules from the Thing (which represents the device) to its onboard process.

The new data lifecycle model requires more precision over the relationships between processes and data, including relationships between the onboard data in an IoT Thing and both its onboard process and other processes that use the Thing. The original model from D4.2 provides enough clues from which to infer the relationships between other processes and the onboard process, but not the onboard data. To fix this, the external relationships of a Thing have been changed. Before they were based on Host and Process-Process relationships, and now they are based on Host and Process-Data relationships. The latter provide better discrimination between cases and can be transferred (by construction) directly to the onboard data. Where before an external process may ‘use’ a Thing, now it ‘reads’ or ‘polls’ an IoT Sensor, and ‘amends’ or ‘updates’ an IoT Controller.

IV.5.3. Bluetooth Secure Simple Pairing model

The original Secure Simple Pairing (SSP) model used in D4.2 was simply a derivation of the generic network authentication model, relying on an out of band key exchange between the paired Bluetooth devices. Some changes were needed in D4.3 for consistency with localisation of the inferred subnet(s) representing the Bluetooth connection. It is now possible to consider where and how this key is exchanged, and how this relates to security in locations where the connection is used. For example, a patient wearing a medical device paired with a smartphone may establish the pairing connection in a secure setting at home, then go out to a public place and use the device there.

Evidently the most important issue is when the user is involved in confirming the connection, whether they can do this on both devices, one device or neither, and whether the connection must be re-established later, in places where attackers would find it easier to eavesdrop on the key exchange and verification process. To this end, the original model (one control strategy for one threat) has been replaced. Now the SSP control represents the confirmation by a user, and the threat of connection spoofing is addressed by three strategies:

- a full (two-sided) verification which can only be used if both devices are interactive, and addresses the threat of spoofed communication in the space where the pair is used.
- two partial (one-sided) strategies representing verification at either end, which address communication spoofing but trigger a separate threat representing attacker interference in the pairing process.

The second threat is addressed by the first strategy only, but the cause term depends on the security of the most secure location shared by both devices. The model now says one cannot secure a Bluetooth connection by technical means using one-sided SSP, but if you can set it up in a secure location and confirm on one side to prevent reconnection attempts, the link should still be reasonably secure.

IV.5.4. Continuous authentication

Parameters in the model of the ProTego Continuous Authentication (CA) mechanisms from UAH were also not updated at this time. To recap, there are two parameters in the model:

- the blocking strength of the control strategy, denoting its ability to reject intruders.
- the intrinsic likelihood of the threat triggered by use of CA, representing the possibility of a false positive preventing access by a legitimate user.

The second parameter is related to the precision and the first to the recall (or sensitivity) of the AI algorithms used. Usually there is a trade-off between these metrics, so algorithms with high precision (very few false positives) tend to have lower recall (more false negatives) and vice versa.

The latest paper from UAH does not give concrete measurements of either parameter but states that recall should be very high (close to 100%), but relatively low precision may be acceptable[1]. For D4.2, we set the recall (blocking effect) to Very High (close to 100%), while the frequency of the triggered lock-out threat was set to Medium, which is consistent with this position.

However, experiments with the model from D4.2 suggest that CA could not reduce the overall level of risk in a system if recall or precision were below these levels, and precision must be significantly higher in some situations. The impact from loss of availability in health care systems is relatively high, and clinicians cannot afford to be locked out. Even with the assumed sensitivity level, the use of CA by clinicians increased rather than decreased the overall level of risk. For patients the balance of concerns leans more towards protecting their data (assuming clinicians can access data on their behalf), and our experiments suggested that the use of CA by patients could reduce the level of risks, at (but not below) the assumed sensitivity level.

Given reliable measurements of recall and sensitivity we can of course adjust the parameters very quickly. But depending on the results, it may make sense to combine CA with other authentication methods, in which a negative CA response triggers additional checks rather than simply locking out the user. With this approach, high recall would still be needed, but it would be possible to accept lower precision as a trade-off. To model this in the SSM domain model, we would need to replace the current CA model rather than simply adjusting its parameters. We therefore decided to wait for more validation results before making either change.

[1] Junquera-Sánchez, J., Cilleruelo-Rodríguez, C., de-Marcos, L. and Martínez-Herráiz, J.J., 2020, November. JBCA: Designing an Adaptative Continuous Authentication Architecture. In Workshop of Physical Agents (pp. 194-209). Springer, Cham.

IV.6. Outlook

Minimal model refinement and parameter tuning may be needed during the validation phase of ProTego and will be incorporated as necessary, for example finalising the model of Continuous Authentication as discussed above, and improvements to some other models if needed to support the validation case studies, e.g. in the slice isolation models.

In the longer term, our research on models for automated risk assessment is moving in three main directions:

1. improvements in scalability and automation levels, allowing risk assessment to be used in the run-time loop for bigger systems addressing populations of assets (including stakeholders) cost effectively.
2. improvements in usability, democratizing risk assessment and making it accessible to non-specialist users in SMEs and micro enterprises and in the public sector.
3. better support for collaborative, multi-stakeholder risk assessment, using models to support trust-building and communication, e.g. in supply chains and in public services including health care.

Domain model developments are relevant in all three areas, in conjunction with improvements in the tools used to process these models. For example, we have already (in H2020 CyberKit4SME) devised an extended set of inference patterns to fill in gaps in system models. This work will continue beyond the end of ProTego in CyberKit4SME, as will developments of the underlying ‘meta-model’ in H2020 FogProtect to better address large scale populations.

V. SIEM

As already expressed in section III of deliverable D4.2, the SIEM is a compendium of open-source tools integrated together, and some other tailor-made processes. This section describes the latest developments carried out in the SIEM, building on the architecture that can be found in the aforementioned deliverable. These changes can be divided into three main categories described in detail in the following sections.

V.1. Updates to existing components

Starting January 2021, the upstream versions of Elasticsearch and Kibana have changed to a non-open-source proprietary license [23]. The Elastic license limits how these components can be used. As a consequence of this, a new project called OpenSearch [24] has forked from the last open-source release of Elasticsearch and Kibana, that is maintained by the community and released under the Apache License. This project is fully reliable since it is supported by organizations such as Amazon, Red Hat, SAP, and others.

Therefore, the SIEM has been updated to use the components of the new OpenSearch project. This involved upgrading versions of the components that include new functionality that did not exist in the versions previously used. The main impact has to do with the alerting system. The new version of Kibana includes the functionality to create customized triggers that spring automated workflows into action. Thus, it was decided to remove the ElastAlert component to simplify the architecture.

V.2. Vulnerabilities Assessment

The vulnerability assessment performed by the SIEM is a key component to carry out the Dynamic Risk Assessment described in Section III, since the information obtained after scanning the target systems is passed on to the SSM to recalculate risk levels and provide recommendations.

V.2.1. Infrastructure Vulnerabilities

The functionality to scan for infrastructure vulnerabilities and supply this information to the SSM to recalculate risk dynamically was already described in deliverable D4.2. Not much has changed since then in this regard with the exception of distributed scanning.

All the trials performed in the lab environments and in the case studies for deliverable D7.3 were carried out with a single scanner deployed with the rest of the toolkit in the same platform. Nevertheless, chances are that for real-life scenarios where very strict network policies are applied, not all target systems to be scanned will be reached from that single point.

Therefore, a more realistic approach has been prepared for that type of environments. It is a lot easier for network or security operators to allow 1-on-1 communications between two specific hosts in different subnetworks than giving full access to all hosts in all subnetworks to the scanner deployed in the ProTego platform. Hence, the distributed scanning approach.

The distributed scanning includes a master scanner that will be deployed with the rest of the toolkit, and as many as necessary slave scanners placed strategically in different network segments that have access to the target systems. Figure V-1 represents the distributed scanning approach.

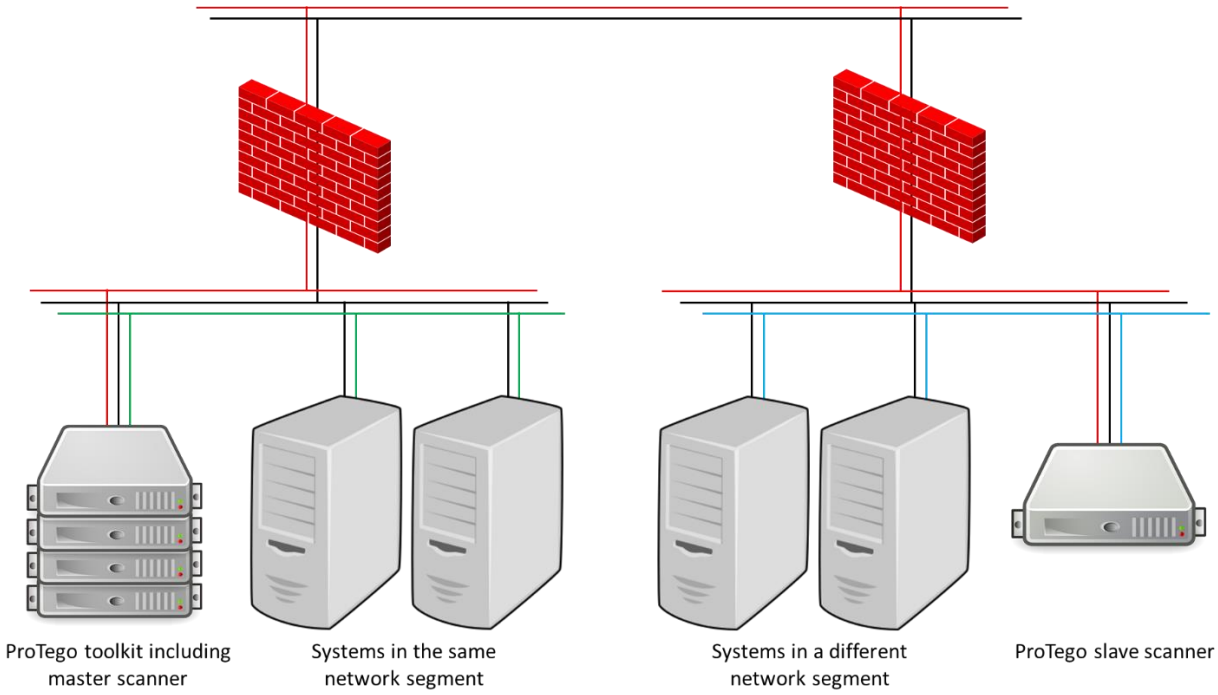


Figure V-1. Distributed vulnerability scanning

V.2.2. Web Application Vulnerabilities

To achieve the objective of cross-organisational risk modelling, besides infrastructure vulnerabilities, already addressed in deliverable D4.2, web application vulnerabilities needed to be addressed too.

Different open-source tools were tested (OWASP ZAP, w3af, and Wapiti). Although each tool has strengths and weaknesses, the final choice was OWASP ZAP since it is a very flexible tool, it allows automation, and gives more information than any other tool tested, which is essential to be able to recalculate risk.

Just as in the case of infrastructure vulnerabilities, the scheduler prepares the tasks to be launched. According to the schedule, the web application scans are launched and can be authenticated or unauthenticated. After the scan is done, a report is generated by ZAP. Just as in the case of infrastructure scans, the analyser parses this report and it is sent to the SSM using the microservice.

V.3. Machine Learning

V.3.1. Introduction

This subsection describes in detail the system that has been developed to detect potential cyber-attacks based on anomalies found in network packets by using machine learning techniques. There are three main contributions:

- A new neuronal network architecture based on a combination of multilayer perceptron (MLP) with attention mechanisms.
- A new embedding protection for cyber-attacks.
- Experimentation with two datasets in which the state of the art for attack detection is improved.

V.3.2. Datasets

Two different datasets have been used that cover a different type of attacks. Both have been obtained through the Canadian Institute of Cybersecurity. Each dataset targets a group of attacks. On the one hand, attacks based on malicious URLs, and on the other hand, intrusion and anomaly detection attacks. Models have developed using these datasets that allow to detect different kinds of attacks.

The first dataset (ISCX-URL 2016) [11] focuses on malicious URL attacks. It contains four classes of attacks:

- Defacement
- Malware
- Phishing
- Spam

This dataset divides the attacks into four subsets, each with a series of defined features. These subsets are made up of benign and malicious URLs, and the malicious URLs include each attack class. The data subsets are made up of 45,450 Defacement URLs, 11,500 Malware URLs, and 10,000 Phishing URLs, 12,000 Spam URLs.

The ISCX-URL 2016 [11] dataset provides different features for each attack class. Not all of them are common for all kind of attacks in the dataset. For Defacement URLs, it provides 12 features, 7 for Malware URLs, 14 for Phishing URLs, and 5 for Spam URLs [see Figure V-2]. Features are calculated using metrics based on URL lexical analysis. One of these features is the 'ratio of special characters found in the URL's domain. All features are numeric and the metrics used to calculate them are based on statistical functions.

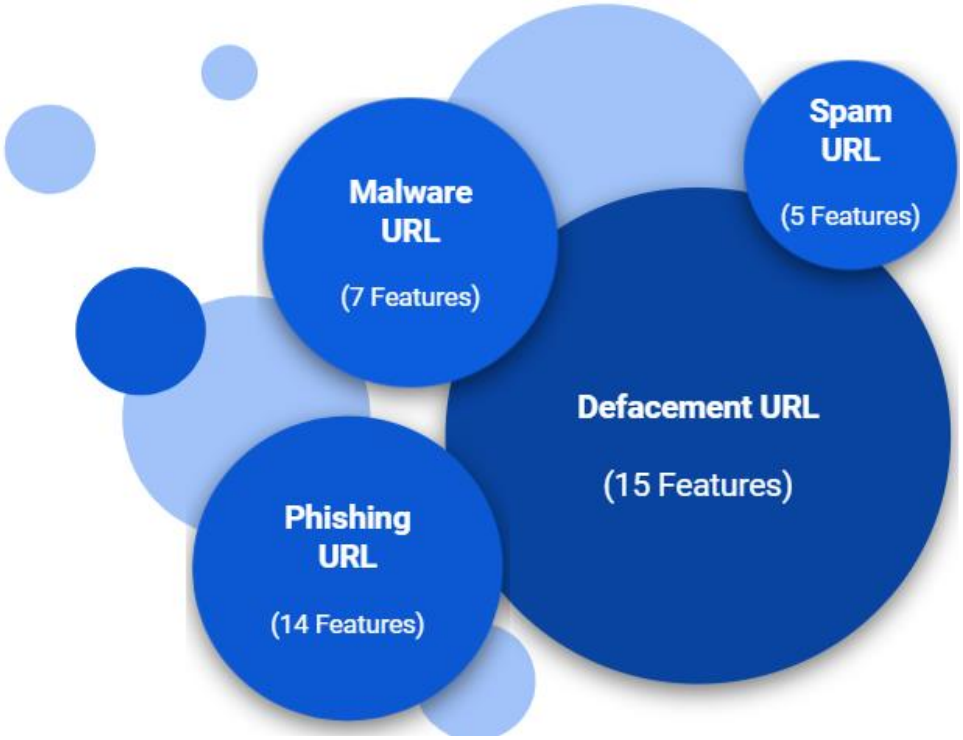


Figure V-2.Number of features of each attack kind in the ISCX-URL 2016 dataset

The second dataset (CSE-CIC-IDS 2018) [12] focuses on intrusion attacks. It includes seven different attack scenarios: Brute Force, DoS, Infiltration, Botnet, DDoS, PortScan and Web Attacks, the latter covering SQL Injection and XSS attacks. The attack infrastructure, with which the dataset was created, includes 50 machines and the victim organization has 5 departments

and includes 420 machines and 30 servers. The dataset includes network traffic captures and system logs for each machine, along with 78 functions extracted from the traffic captured using CICFlowMeter-V3.

The CSE-CIC-IDS 2018 [12] dataset provides 78 features that are common for each kind of attack. These features are numerical and most of them can be calculated by means of metrics from basic data that can be found in a network packet. Some of these data are the number of source and destination network packets and the size of the source and destination network packets. These basic data are part of the 78 features of the dataset. An example of a feature calculated from this data is the '*average source network packet size*'. Since all the features are common to all the attack classes covered by the dataset, all the attack classes have been grouped into one '*Intrusion Attack*', simplifying the dataset, [see Figure V-3].

Figure V-3. Attack type grouping

These datasets have been chosen because they are used for benchmarks and are recognized by the community in the area of cybersecurity, besides being up to date for each type of attack. It is vitally important that datasets are up to date so that detection of cyberattacks does not become obsolete too soon.

Another important reason for choosing these datasets has been the type of features it provides. Most of the features of datasets can be calculated using metrics from some basic data. Given a data series where features are missing, this data can be filled in using metrics. This makes the datasets robust enough to be applied to systems where there may be cases where incomplete data series have to be processed.

V.3.3. System Structure

The system is made up of five neural models with the same architecture. Each of the models has been trained and tested with one of the datasets described in the previous section.

On the one hand, for the ISCX-URL 2016 [11] dataset four neural network models have been developed, one for each kind of attack in the dataset, since each of them has different features. On the other hand, for the CSE-CIC-IDS 2018 [12] dataset there is only one neural model, since all the kind of attacks in the dataset has been grouped in one. This has been possible thanks to

the fact that all kind of attacks in the dataset has the same features, as already explained in the previous section.

Network traffic is captured with Packetbeat and stored on Elasticsearch, which provides the data events that are first pre-processed and then evaluated by the neural models. These data events are not part of any of the datasets described above.

From each data event, features that are required to be evaluated by each of the neural models are extracted. However, it may happen that the data events do not provide all the necessary features, or cannot be calculated. Another possibility is that all the necessary features are available for one neural model but not for another. Therefore, three cases are considered for each neural model:

- **Full Data:** It occurs when a data event provides all the necessary features to be evaluated by a neural model, or it has some and the rest can be calculated from what it provides.
- **Partial Data:** When the data event does not provide all the features required by the neural model or cannot be calculated from those it provides. A case of partial data is considered if at least half of the features required by the model can be obtained. For the missing features, a default value of -10.00 is assigned, which is not available in the datasets.
- **No Data:** When the features that the event provides plus the ones calculated, totals less than half of the features that the model requires. In this case, that neural model does not apply.

Neural models will only be applied in cases of full data and partial data. In the case of no data, that is, that a data event does not have enough features or cannot be calculated, for any of the neural models, the data event will be labelled as 'undefined'. If there is only one model that can be applied, the event will not be labelled as 'undefined' but will be classified as 'benign' or 'malicious' depending on what those models that can be applied determine.

A data event will be considered 'benign' if all applied models, that is, those for which there is enough data, have determined that it is. If a single data model classifies the event as 'malicious' then this is how the data event will be labelled. Depending on which neural model classifies the data event as 'malicious', the type of attack is determined. For example, if the model that classifies the event as 'malicious' is the SPAM URL model, the event will be malicious of the spam class.

Figure V-4 shows a complete diagram of the structure of the system. From the data events provided by the database or the network traffic tracker to determine whether it is a malicious event or not, determining what type of attack it is. In our case we extract the data events using Elasticsearch.

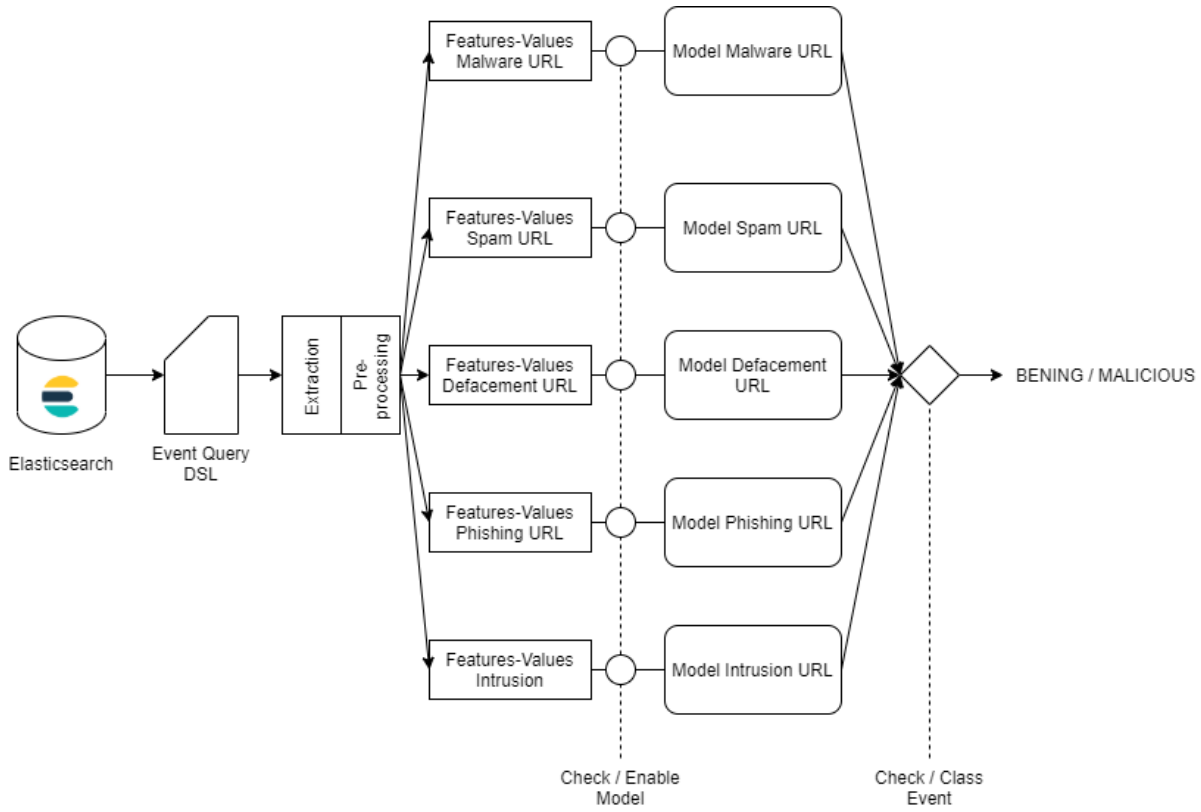


Figure V-4. SIEM Machine Learning structure

V.3.4. Model Architecture

The architecture developed for the neural models is a multilayer perceptron (MLP) with a self-attention mechanism, [see Figure V-5]. The model is made up of nine fully connected neural layers and a multi-headed self-attention layer. The multi-head self-care layer divides the fully connected layer assembly into two parts, two MLP modules. A first module that deals with the analysis of the features and a second module acts as a classifier.

The number of neurons in each fully connected layer before the multi-head attention layer is sequentially ascending, following the sequence: *(16, 32, 64, 128)*. After the attention layer the sequence is descending, *(128, 64, 32, 16)*. The last layer has as many neurons as there are output classes. The model has only two possible outputs, 'benign' and 'malicious'.

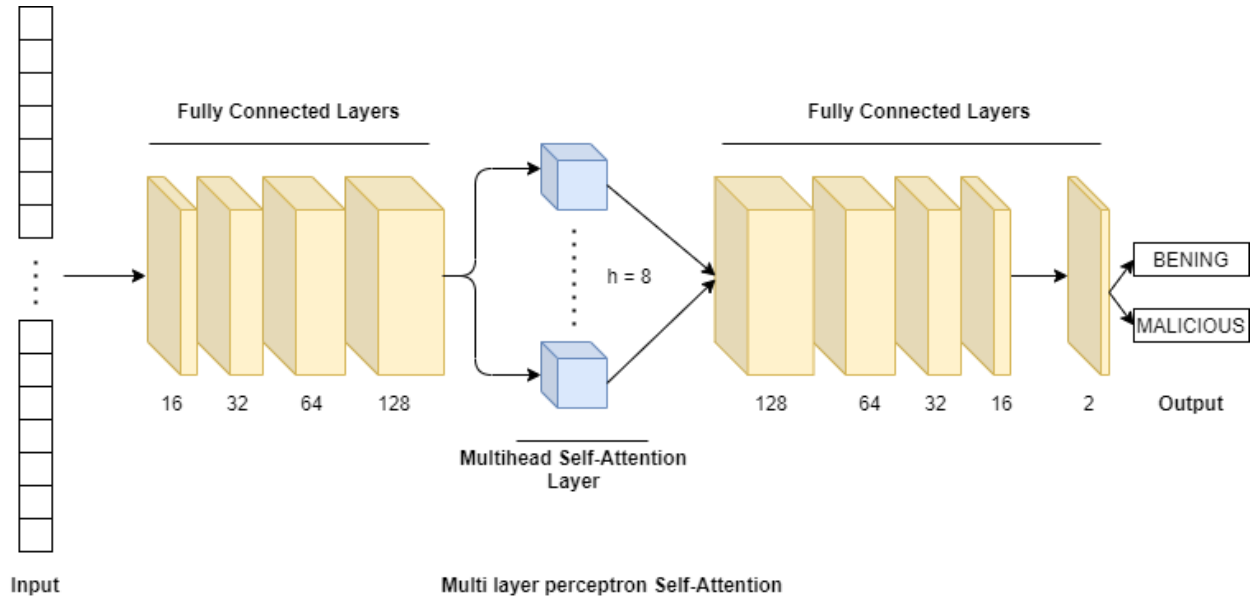


Figure V-5. Neural model architecture

The input to the model is the values of the features of each dataset in '*float32*' type and with a shape of *1 × k*, depending on the number of features. For the 'Intrusion' attack the input vector is *1 × 78*, for 'Defacement URL' it is *1 × 15*, for 'Phishing URL' it is *1 × 14*, for 'Malware URL' it is *1 × 7* and for 'Spam URL' is *1 × 5*. The values of the input vector $x_1 \ldots x_k$ are normalized using a data standardization based on the mean and standard deviation.

The join of the attention mechanisms with the first MLP module analyses and weights the input features based on their values $x_1 \ldots x_k$. MLP layers are made up of perceptron that generate outputs $\sum_{i=1}^{k} w_i \cdot x_i$ where w_i represents the synaptic weight of the neuron n_i.

The output vector $S = (s_1 \ldots s_r)$ of the first MLP module becomes the input of the multi-head self-attention layer. The self-attention function receives three input parameters, queries *(Q)*, keys *(K)*, and values *(V)*, where they all have the same value the vector *S*. The self-attention function in the model is:

$$Attention(S,S,S) = Softmax\left(\frac{S^{T-1}}{\sqrt{d_s}}\right) \cdot S$$

Where d_S is the dimension of the vector *S*. The Attention function weights each value of the vector *S* using the *Softmax* function. In our model we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{model} / h = 16$. Due to the reduced dimension of each head, the total computational cost is like that of single-head attention with full dimensionality. When the *Q, K* and *V* values are the same as they are in our case, the layer is called Multi-head Self-Attention. Attention mechanisms do not require sequential data to be sorted, as is the case with ours. Due to these features, the use of attention layers allows much more parallelization than RNN, thus reducing the training and testing times of the model. In fact, the order of complexity for each Multi-head Self-Attention layer is $O(n^2 \times d)$ being *n* is the sequence length and *d* is the representation dimension.

Indirectly the join of MLP with Attention mechanisms evaluates which features of the input vector are more relevant for the classifier after processing them by the first MLP module. Finally, after the weighting carried out by the Attention function, the second MLP module acts as a classifier and classify the input data, generating an output vector.

The output vector of the model has dimension *1 × 2*. The neural network associates a percentage of precision with each of the output classes, the sum of all of them gives 1. The class with the highest percentage of precision is selected.

The multilayer perceptron is an ANN formed by multiple layers, in such a way that it can solve problems that are not linearly separable. The multilayer perceptron is fully connected. Each output of a neuron *i* of layer *X* is an input of all neurons of layer *X + 1* in this way up to the output layer in which the activation function *f(x)* is applied, which is applied to an algorithm of back-propagation for the rest of layers thus achieving the objective of learning of the network, [see Figure V-6].

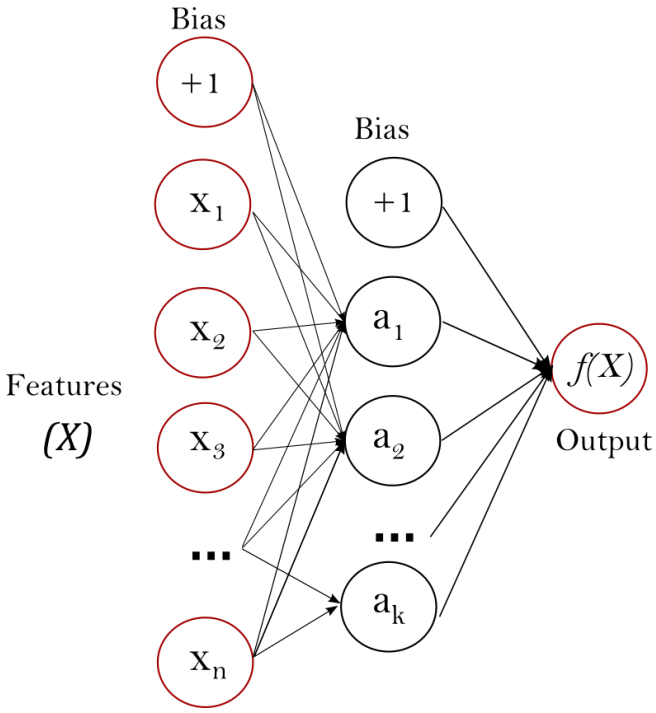


Figure V-6. Multilayer Perceptron Scheme

V.3.5. Methodology

For the models, the weights of all neurons were randomly initialized. A base learning rate of 10^{-4}, a batch size of 32 and 100 epochs were selected. Weight decay was initiated to 0.1. Furthermore, the default configuration of the Adam optimizer was used. The *Categorical Cross Entropy* function was chosen as the loss function and the *Softmax* function as the activation function for the last layer of the classifier. To perform the experiments, the CUDA toolbox was used to extract deep features on *Nvidia RTX 2070 Super GPU*. CUDA is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs. CUDA toolkit is widely used to perform the training process in machine learning models in less time and improve the inference time in the testing process. The operating system was *Windows 10* using *Intel Core i7*. The performances with the datasets have been carried out using a stratified permutation cross-validator method. A 5-fold cross validation has been chosen.

A 5-fold cross validation is a training and testing mechanism for learning models. This method divides the data set into four parts, stratified by classes, using four to train the model and one to test it. With this procedure, the training and testing set maintains a ratio of 80-20%. Validation is performed five times by varying the training and testing parts as shown in Figure V-7.

If all the results provided by this method converge close to the same value, it means that the learning of the model is correct. This method provides a robust way to confirm if the model has learned the concept we want.

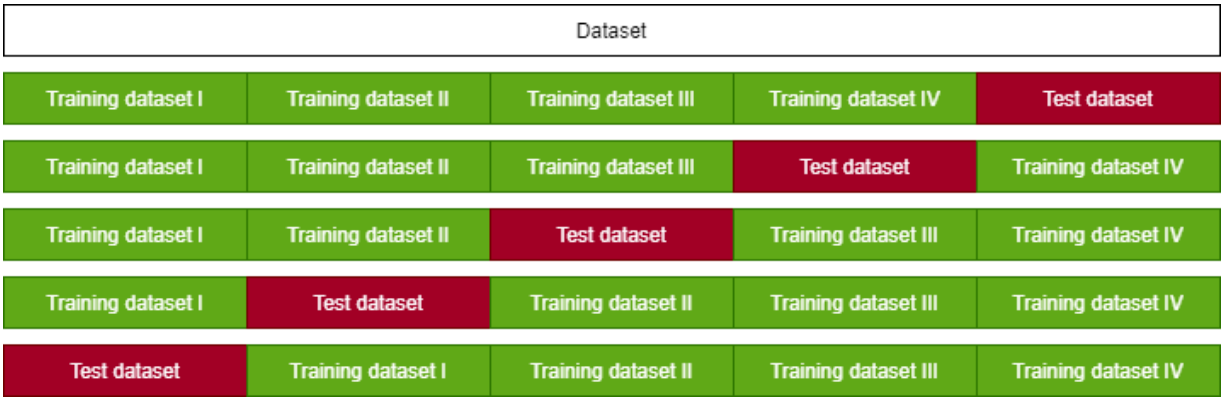


Figure V-7. Division of the dataset for a 5-fold cross validation

The whole system has been implemented with Python 3.8. For neural models, Tensorflow library has been used in its latest version, 2.4.1. Tensorflow version must be 2.4.1 since it is the only one that has the Multi-head Attention layers. To evaluate the models and process the data, the Scikit-learn API has been used.

V.3.6. Results

This section shows the results obtained for each dataset which are evaluated using the following metrics:

- **Accuracy Score**: In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of ground truth.
- **Precision Score**: The test accuracy or precision is the ratio $TP/(TP + FP)$ where TP is the number of true positives and FP the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.
- **Recall**: The recall is the ratio $TP/(TP + FN)$ where TP is the number of true positives and FN the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.
- **Prediction Latency**: This metric measures the latency one can expect when making predictions in bulk or atomic mode.

In a machine learning binary classification problem, as is our case, there are several criteria that it is important to define to better understand the metrics. These criteria are as follows:

- True Positives (TP): When the ground truth of a data is 1 (True) and its prediction is also 1 (True).
- True Negatives (TP): When the ground truth of a data is 0 (False) and its prediction is also 0 (False).
- False Positives (FP): When the ground truth of a data is 0 (False) and its prediction is 1 (True).
- False Negatives (FN): When the ground truth of a data is 1 (True) and its prediction is 0 (False).

Results show that the new neural model returns an accuracy score over 96% in all cases. Table V-1, Table V-2, Table V-3, Table V-4 and Table V-5 present the results for the 5-fold cross validation of each dataset.

Defacement URL Dataset

For the Defacement URL dataset, the average Precision Score is 99.54 % and the average Recall 99.35 %, the average percentage of accuracy score with standard deviation is 99.44 ± 0.15 %. The average prediction latency of the model is 0.000082 s.

Table V-1. Results with Defacement URL Dataset for 5-fold cross validation

Iteration	Accuracy Score	Precision Score	Recall	Prediction Latency
01	99.46 %	99.62 %	99.30 %	0.00008 s
02	99.37 %	99.06 %	99.68 %	0.00009 s
03	99.20 %	99.68 %	98.73 %	0.00008 s
04	**99.68 %**	**99.74 %**	**99.62 %**	**0.00009 s**
05	99.52 %	99.62 %	99.43 %	0.00007 s

Malware URL Dataset

For the Malware URL dataset, the average Precision Score is 96.83 % and the average Recall 98.23 %, the average percentage of accuracy score with standard deviation is 97.69 ± 0.32 %. The average prediction latency of the model is 0.000084 s.

Table V-2. Results with Malware URL Dataset for 5-fold cross validation

Iteration	Accuracy Score	Precision Score	Recall	Prediction Latency
01	97.51 %	95.95 %	98.80 %	0.00009 s
02	97.37 %	96.41 %	97.98 %	0.00008 s
03	**98.31 %**	**97.22 %**	**99.18 %**	**0.00009 s**
04	97.68 %	97.68 %	97.31 %	0.00008 s
05	97.58 %	96.90 %	97.91 %	0.00008 s

Spam URL Dataset

For the Spam URL dataset, the average Precision Score is 99.00 % and the average Recall 99.39 %, the average percentage of precision in the accuracy score with standard deviation is 99.25 ± 0.14 %. The average prediction latency of the model is 0.000084 s.

Table V-3. Results with Spam URL Dataset for 5-fold cross validation

Iteration	Accuracy Score	Precision Score	Recall	Prediction Latency
01	98.99 %	98.59 %	99.25 %	0.00009 s

02	**99.44 %**	**99.25 %**	**99.55 %**	**0.00009 s**
03	99.27 %	99.10 %	99.32 %	0.00008 s
04	99.27 %	98.96 %	99.47 %	0.00008 s
05	99.30 %	99.10 %	99.40 %	0.00008 s

Phishing URL Dataset

For the Phishing URL dataset, the average Precision Score is 97.45 % and the average Recall 97.25 %, the average percentage of accuracy score with standard deviation is 97.38 ± 0.35 %. The average prediction latency of the model is 0.00008 s.

Table V-4. Results with Phishing URL Dataset for 5-fold cross validation

Iteration	Accuracy Score	Precision Score	Recall	Prediction Latency
01	96.78 %	96.40 %	97.10 %	0.00008 s
02	97.33 %	97.67 %	96.90 %	0.00008 s
03	97.36 %	97.99 %	96.64 %	0.00008 s
04	**97.78 %**	**97.32 %**	**98.22 %**	**0.00008 s**
05	97.69 %	97.88 %	97.43 %	0.00008 s

Intrusion Detection Evaluation Dataset

For the Intrusion Detection Evaluation dataset, the average Precision Score is 98.60 % and the average Recall is 98.89 %. The average percentage of accuracy score with standard deviation is 99.41 ± 0.21 %. The average prediction latency of the model is 0.000086 s.

Table V-5. Results with Intrusion Detection Evaluation Dataset for 5-fold cross validation

Iteration	Accuracy Score	Precision Score	Recall	Prediction Latency
01	99.61 %	98.79 %	99.26 %	0.00009 s
02	**99.61 %**	**98.82 %**	**99.21 %**	**0.00008 s**
03	99.54 %	98.51 %	99.19 %	0.00009 s
04	99.13 %	98.94 %	98.92 %	0.00009 s
05	99.18 %	97.97 %	97.90 %	0.00008 s

The latency period is between 7×10^{-5} s and 9×10^{-5} s, which means that our models are capable of processing between 11.111 and 14.285 attack events per second, however, the processing speed of the models depends on the computational resources dedicated to them. The results

show that the accuracy score is between 97.38 ± 0.35 % and 99.41 ± 0.21 %, this means that the error rate of our models is between 0.59 % and 2.62 %. In the worst case, our system will erroneously classify 2 out of 100 data events.

The highest error rate corresponds to the model in charge of detecting Phishing URL events. However, the models are enabled or disabled depending on the amount of data we obtain from each data event as explained in the 'System Architecture' sub-section. Table V-6 shows the average error rate (AER) for each combination of models. The AER is calculated by adding the error of each of the enabled models and dividing it by the number of enabled models. In the case that there is only one model enabled, the average error rate is equal to the error rate of that model. Most of the average error rates are around 1 %.

Table V-6. Average error rate for each case

Defacement URL	Malware URL	Phishing URL	Spam URL	Intrusion Detection	Average Error Rate
Enable	Enable	Enable	Enable	Enable	1.03 %
Enable	Enable	Enable	Enable	Disable	1.19 %
Enable	Enable	Enable	Disable	Enable	1.15 %
Enable	Enable	Enable	Disable	Disable	1.41 %
Enable	Enable	Disable	Enable	Enable	0.74 %
Enable	Enable	Disable	Enable	Disable	0.85 %
Enable	Enable	Disable	Disable	Enable	0.80 %
Enable	Enable	Disable	Disable	Disable	1.00 %
Enable	Disable	Enable	Enable	Enable	0.87 %
Enable	Disable	Enable	Enable	Disable	1.03 %
Enable	Disable	Enable	Disable	Enable	0.97 %
Enable	Disable	Enable	Disable	Disable	1.27 %
Enable	Disable	Disable	Enable	Enable	0.42 %
Enable	Disable	Disable	Enable	Disable	0.44 %
Enable	Disable	Disable	Disable	Enable	0.35 %
Enable	Disable	Disable	Disable	Disable	0.32 %
Disable	Enable	Enable	Enable	Enable	1.21 %
Disable	Enable	Enable	Enable	Disable	1.49 %
Disable	Enable	Enable	Disable	Enable	1.43 %
Disable	Enable	Enable	Disable	Disable	1.95 %
Disable	Enable	Disable	Enable	Enable	0.88 %
Disable	Enable	Disable	Enable	Disable	1.12 %

Disable	Enable	Disable	Disable	Enable	1.04 %
Disable	Enable	Disable	Disable	Disable	1.69 %
Disable	Disable	Enable	Enable	Enable	1.05 %
Disable	Disable	Enable	Enable	Disable	1.39 %
Disable	Disable	Enable	Disable	Enable	1.30 %
Disable	Disable	Enable	Disable	Disable	2.22 %
Disable	Disable	Disable	Enable	Enable	0.47 %
Disable	Disable	Disable	Enable	Disable	0.56 %
Disable	Disable	Disable	Disable	Enable	0.39 %
Disable	Disable	Disable	Disable	Disable	0%

To verify how effective and accurate the neural models are, a study of the state of the art based on the datasets used has been carried out.

For the dataset (CSE-CIC-IDS 2018) [12], studies that propose different neural network architectures were found. Table V-7 shows the accuracy score of the different models. Despite using the same dataset, not all models have been trained and tested following the same methodology. In [18] the dataset is divided into 10 sub-datasets called scenarios. Some of these sub-datasets are multi-classes and others are single class, none of them covers all possible attacks on the dataset. The accuracy score result reported in Table V-5 is the arithmetic mean of the results obtained in [18] for each of the scenarios. The results show that the model proposed improves the state of the art with an accuracy score of 99.41 ± 0.21 %. The model developed is the most accurate, however we cannot determine that its performance is better compared to the other models, since previous research does not report the prediction latency values of the other models.

Table V-7. State of the art for the CSE-CIC-IDS 2018 dataset

Model	Acc. Score
Deep Neural Network [15]	97.28 %
Recurrent Neural Network [15]	97.31 %
Convolutional Neural Network [15]	97.37 %
Restricted Boltzmann Machine [15]	97.28 %
Deep Boltzmann Machine [15]	97.37 %
Deep Belief Networks [15]	97.30 %
Deep Auto-encoders [15]	97.37 %
Multi-Layer Perceptron [16]	93.27 %
Learning Vector Quantization [16]	65.61 %

Clonal Selection Algorithm [16]	65.64 %
2L-ZED-IDS (DoS) [17]	98.70 %
2L-ZED-IDS (Botnet) [17]	99.20 %
2L-ZED-IDS (Brute Force) [17]	95.80 %
2L-ZED-IDS (Web Attack) [17]	98.60 %
2L-ZED-IDS (DDoS) [17]	98.70 %
2L-ZED-IDS (Web Infiltration) [17]	98.50 %
Convolutional Neural Network [18]	95.14 %
Proposed Model	**99.41 ± 0.21 %**

Less studies report results for the dataset (ISCX-URL 2016) [11] than for the dataset (CSE-CIC-IDS 2018) [12]. There are two neural models and a Machine Learning algorithm that have used the dataset (ISCX-URL 2016) [11]. Table V-8 shows the accuracy score of the different algorithms. As the data indicates, our model for the dataset improves on the state of the art for Defacement and Spam attacks. However, it is outperformed by the Random Forest algorithm applied in [11] for Malware and Phishing attacks. The A^3 and DA^3D models of [19] are trained and tested using a multi-class methodology, giving accuracy score values much lower than those of the model that we propose. Our model improves the state of the art for two of the four attack classes. As prediction of latency values is not reported for previous studies, we cannot determine which model presents better performance.

Table V-8. State of the art for the ISCX-URL 2016 dataset

Model	**Acc. Score**
Random Forest (Defacement) [11]	99.00 %
Random Forest (Malware) [11]	99.00 %
Random Forest (Phishing) [11]	99.00 %
Random Forest (Spam) [11]	99.00 %
A^3 [19]	80.00 %
DA^3D [19]	80.00 %
Proposed Model (Defacement URL)	**99.44 ± 0.15 %**
Proposed Model (Malware URL)	97.69 ± 0.32 %
Proposed Model (Phishing URL)	97.38 ± 0.35 %
Proposed Model (Spam URL)	**99.25 ± 0.14 %**

V.3.7. SIEM Integration

For the correct integration in the SIEM, a series of files and scripts are necessary, which are detailed in this section.

In the first place some files that correspond to the training of the neural models. These files are the result of training the machine learning models with each of the datasets and each of them is prepared to detect a type of attack. In total, 25 pre-trained models were generated, since for each dataset, a 5-fold cross validation training was performed. Given that for each dataset the five models generated converge in precision values, only the first of each batch of five is taken to form the cyber-attack detection system. The files are saved in the *.joblib* format.

Second, there are the scalar files. Scalars are files that allow datasets to be normalized. A scalar file is generated from a training with all the data of the set in its entirety, which will be used by the model, both in training and in tests. Scalars normalize the data using the mean and standard deviation functions and store that information to be able to normalize new data that does not belong to the dataset from which the scalar was generated. In total 5 scalar files were generated, one for each dataset associated with each model. The files are saved in *.pkl* format.

The files corresponding to the pre-trained models and the scalars are stored in pairs in folders, which are named after the kind of attack for which they were generated. All pre-training and scalar model files have the same name, they differ by the folder in which they are located, [see Table V-9].

Table V-9. Folders for the pre-trained models and scalars

Folder	**Pre-trained Model**	**Scaler**
Pretrained_Models/Intrusion	model_v1_0.joblib	scaler.pkl
Pretrained_Models/Defacement	model_v1_0.joblib	scaler.pkl
Pretrained_Models/Malware	model_v1_0.joblib	scaler.pkl
Pretrained_Models/Phishing	model_v1_0.joblib	scaler.pkl
Pretrained_Models/Spam	model_v1_0.joblib	scaler.pkl

The metrics script (*metrics.py*) stores different functions with which, from a series of input data, it is possible to calculate those features that are not directly provided by a data event or network packet, that are necessary to fill in the missing features to run each of the models.

The data events come in the format shown in Figure V-8. There are a number of features that are common for all data events:

- @timestamp
- _id
- source_port
- destination_port
- network.bytes
- network.packets
- destination.bytes
- destination.packets
- source.bytes
- source.packets
- url.domain

The url.domain feature will always appear in those data events with a web connection. From these minimum features, functions of the metrics script are used to calculate the rest of the features.

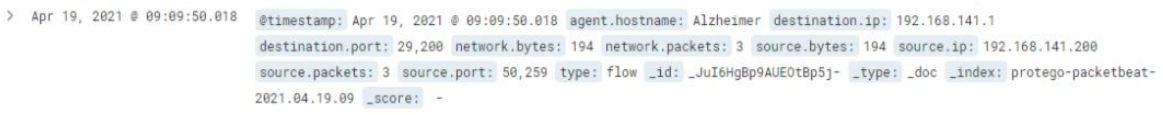


Figure V-8. Example of a data event obtained from the packetbeat using Elasticsearch

Finally, the last script is the main script (*system_main.py*) where the entire process is carried out and is stored together with the metrics script. This script generates a log file with the alerts that is read by the Wazuh Manager to decode it and trigger alerts is necessary.

V.3.8. Conclusions

The proposed Machine Learning System for the SIEM is based on neural models for the detection of cyberattacks that improves the state of the art. The main contributions are:

- An architecture that combines fully connected layers with attention mechanisms
- The detection against multiple cyberattacks
- An experimentation with two datasets that are benchmarks in the field of cybersecurity

In view of the results obtained a wide range of cyberattacks can be detected with more than 96% accuracy. The architecture is adaptable enough to add new models trained for other kind of attacks. However, this adaptability creates the drawback of not having a single model for all kind of attacks, since each one of them has a series of features that define and differentiate it from one another.

V.4. Final SIEM Architecture

This section shows the diagram of the final architecture for the SIEM after implementing all the changes described in the previous sections:

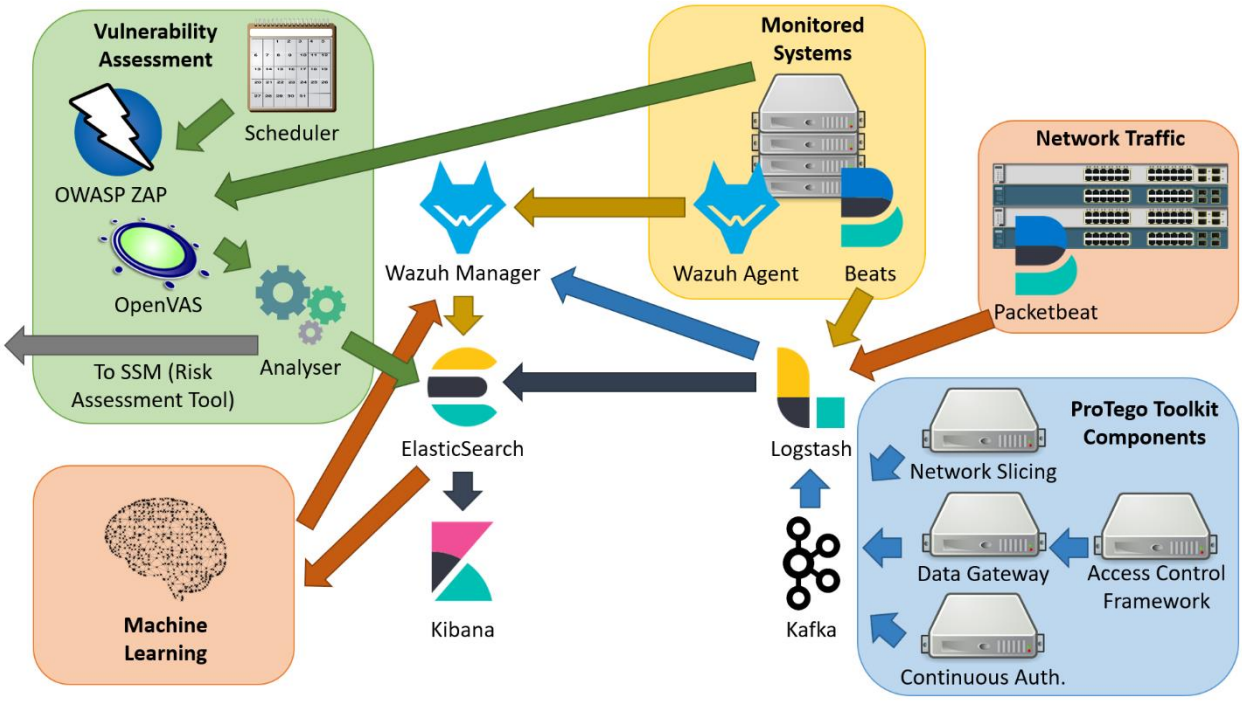


Figure V-9. SIEM Architecture

Four different types of information sources can be found:

1. **Monitored Systems**: collecting information from different log files. Two types of agents can be used:
    a. Wazuh, if the purpose is to analyse the logs to find possible threats. Information is sent to the Wazuh Manager to perform the analysis, raises alerts if necessary and stores it on ElasticSearch.
    b. Beats, if the purpose is just to store logs on a centralized repository. Information is sent to Logstash to perform some processing, if necessary, before storing it on ElasticSearch.
2. **Network Traffic**: collecting network packets that will later be analysed by the Machine Learning jobs, that raise alerts to the Wazuh Manager if any possible threat is found. Although the agent is a Beat in itself, it is mentioned as a separate data source because it will be used for a specific purpose, and the information is analysed by a different component. The flow of information is the same as other Beats, finally storing it on ElasticSearch, where the Machine Learning jobs read it from.
3. **ProTego Toolkit Components**: collecting specific alerts raised by other ProTego components, which may detect issues such as malicious network scans, data tampering, authentication issues, etc. Information is sent by these components to Kafka topics, where Logstash reads if from, processes it and sends it to the Wazuh Manager. In turn the Wazuh Manager analyses it, raises alerts, and stores it on ElasticSearch.
4. **Vulnerabilities**:
    a. The Scheduler creates the tasks for the different scanners (OpenVAS and OWASP ZAP) to run the scans on a regular basis.
    b. Scans are carried out according to the schedule where infrastructures and applications are tested.
    c. Vulnerability information is processed by the Analyser, stored on ElasticSearch and sent to the SSM.

d. The SSM recalculates risk levels and generates a list of recommendations that is sent back to the SIEM, as explained in section III - Dynamic Risk Assessment.

All the information is available at the Kibana User Interface, which can also be configured to notify alerts by some special means, such as email, Telegram, Microsoft Teams, etc.

VI. CONCLUSIONS

This deliverable has described advances from D4.2 in the final year of the ProTego project. The key advances are summarised as follows.

- We have complete risk models for both of the two end-user scenarios, Pocket EHR and FoodCoach. We have shown how the systems are modelled in the SSM toolkit, the application of initial controls and the resulting static risk level. We have also illustrated that the ProTego components reduce the risk of serious misbehaviours in both the FoodCoach and Pocket EHR scenarios. In particular, the ProTego Continuous Authentication provides ongoing protection for impersonation or vulnerabilities due to the theft of a BYOD device, and the ProTego Parquet encryption provides efficient encrypted data query processing.
- We have investigated collaborative, cross-organisation risk modelling via an information hiding approach where different stakeholders can focus on their parts of the system – here a key example split is between application and infrastructure, and each can be hidden from the others.
- We have extended the dynamic risk assessment concept via extensions to the Run Time microservice, which provides a REST API for vulnerabilities detected at runtime, risk recalculation to determine the resulting risk level due to the vulnerabilities, and recommendations to reduce the risk level.
- We have provided additional support for web applications via support for OWASP ZAP (Zed Attack Proxy). The support is achieved through mappings between ZAP alerts and risk model updates.
- We have extended the domain model knowledge base to support relevant aspects such as contextualisation of e.g. mobile (BYOD) devices, so that their location is taken into account in the risk modelling, theft of devices and network localisation.
- We have extended vulnerability detection capabilities of the SIEM to include web applications (besides infrastructure vulnerability detection).
- We have improved detection capabilities of the SIEM by using Deep Learning techniques.

Overall, the key advance in WP4 from ProTego is the transition from static risk modelling to dynamic, runtime risk modelling, so that risks can be determined in response to security alerts and reports of vulnerabilities in real time. The key benefit of this to practitioners is that their system is constantly monitored and when risk events occur, they are detected and assessed in real time, together with recommendations regarding how to mitigate any raised risk.

VII. APPENDICES

VII.1. Run Time Microservice API

VII.1.1. POST /api/models/{modelId}/reset-vulnerabilities

This method rolls back risk model TWAs that were changed by any previous vulnerabilities calls. This method should be used to mark the start of a new session followed by vulnerability updates and risk caclulation/recommendation calls. This call is BLOCKING. The key part

Parameters:

- str model_id: Model ID that can be used to access the model
- :param identification params

Request body:

- none

Return body:

- responses:return: None

HTTP return codes:

- 202 Successful Response
- 404 Item not found
- 422 Validation Error
- 423 Resource locked, by another process try again later.

VII.1.2. POST /api/models/{modelId}/asset/openvas-vulnerabilities

Add and update new vulnerabilities from OpenVAS. The SIEM sends a list of vulnerabilities retrieved from an OpenVAS scan report to the specified risk model mapped into trustworthiness levels of risk model assets in SSM.

Parse a list of vulnerabilities discovered by an OpenVAS system scan for a given asset and update its levels of trustworthiness attributes as appropriate. To do that, a look-up table is defined to map base cvss vectors to sets of trustworthiness attributes and their respective levels. This call is BLOCKING.

Parameters:

- :str model_id: Model ID that can be used to access the model

Request body:

- :param CVSS cvss: CVSS vulnerability POST body

Return body

- none:return:

HTTP return codes:

- 200 Successful Response
- 404 Item not found
- 422 Validation Error
- 423 Resource locked, by another process try again later.

VII.1.3. POST /api/models/{modelId}/asset/zap-vulnerability

Add and update new vulnerabilities from OWASP ZAP. The SIEM sends a list of vulnerabilities retrieved from an ZAP scan report to the specified risk model mapped into trustworthiness levels of risk model assets in SSM.

Parse a list of vulnerabilities discovered by a ZAP system scan for a given asset and update its levels of trustworthiness attributes as appropriate. To do that, a look-up table is defined to map base cvss vectors to sets of trustworthiness attributes and their respective levels. This call is BLOCKING.

Parameters:

- :str model_id: Model ID that can be used to access the model
- :Boolean authenticated: whether the scan uses authentication or not

Request body:

- :param ZAP vulnerability report POST body

Return body

- none

HTTP return codes:

- 200 Successful Response
- 404 Item not found
- 422 Validation Error
- 423 Resource locked, by another process try again later.

VII.1.4. POST /api/models/{modelId}/calc-risks

Run the risk calculation on the current state of the risk model. This is an asynchronous (NON_BLOCKING) call that instantiates the current model risk calculation and finds risk mitigation recommendations. This is because the risk calculation is a relatively long-lived operation, which is too long for a blocking call. Therefore a background task ID is returned so the calculation status can be polled. Additionally, the results, the risk vector and mitigation recommendations, are asynchronously pushed in to a Kafka queue.

Parameters:

- :str model_id: Model ID that can be used to access the model
- :Boolean authenticated: whether the scan uses authentication or not

Request body:

- :none

Return body

- :the background task ID of the requested risk calculation task

HTTP return codes:

- 202 Successful Response
- 404 Item not found
- 422 Validation Error
- 423 Resource locked, by another process try again later.

VII.1.5. GET /api/models/describe/task-status/{vid}

This is an auxilary call to support asynchronous mode calls. It allows to check the status of the background task.

Parameters:

- :param str vid: The ID of the risk calculation task. [From calc-risks return]

Request body:

- :none

Return body

- :Status of a background task.

HTTP return codes:

- 200 Successful Response
- 404 Item not found
- 422 Validation Error
- 423 Resource locked, by another process try again later.

VII.1.6. GET /api/models/download/risk/{vid}

Get risk calculation results in the form of a risk vector

Parameters:

- :param str vid: The ID of the risk calculation task. [From calc-risks return]

Request body:

- :none

Return body

- :risk vector in JSON format

HTTP return codes:

- 200 Successful Response
- 404 Item not found
- 422 Validation Error
- 423 Resource locked, by another process try again later.

VII.1.7. GET /api/models/download/recommendations/{vid}

Get risk mitigation recommendations

Parameters:

- :param str vid: The ID of the risk calculation task. [From calc-risks return]

Request body:

- :none

Return body

- :recommendations in JSON format

HTTP return codes:

- 200 Successful Response
- 404 Item not found
- 422 Validation Error
- 423 Resource locked, by another process try again later.

VIII. REFERENCES AND INTERNET LINKS

VIII.1. References and Internet Links

[1] Mohammadi, N., Goeke, L., Heisel, M. and Surridge, M. Systematic Risk Assessment of Cloud Computing Systems using a Combined Model-based Approach.In Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS 2020) - Volume 2, pages 53-66 DOI: http://doi.org/10.5220/0009342700530066. 2020.

[2] Information technology — Security techniques — Information security risk management. ISO/IEC 27005:2018. Available at https://www.iso.org/standard/75281.html, retrieved 2021-06-04.

[3] OpenVAS - Open Vulnerability Assessment Scanner. https://www.openvas.org/. Retrieved 2021-06-07.

[4] OWASP Zed Attack Proxy (ZAP). https://www.zaproxy.org/. Retrieved 2021-06-07.

[5] Common Weakness Enumeration (CWE), Latest Version (currently 4.4). Available at https://cwe.mitre.org/data/xml/cwec_latest.xml.zip. Accessed 2021-06-11.

[6] The Web Application Security Consortium Threat Classification Taxonomy Cross Reference View. http://projects.webappsec.org/w/page/13246975/Threat%20Classification%20Taxonomy%20Cross%20Reference%20View CC-BY 2.5 license. Accessed 2021-06-11.

[7] Common Attack Pattern Enumeration and Classification. https://capec.mitre.org/ Latest version (currently 3.4). Available at https://capec.mitre.org/data/xml/capec_latest.xml Last accessed 2021-06-14.

[8] CWE/SANS 2009 CWE/SANS Top 25 Most Dangerous Programming Errors. Available at https://cwe.mitre.org/top25/archive/2009/2009_cwe_sans_top25.html. Accessed 2021-06-14.

[9] QWASP Top Ten 2010. Available at https://github.com/OWASP/OWASP-Top-10/blob/master/2010/Documents/OWASP%20Top%2010%20-%202010.pdf Last accessed 2021-06-17.

[10] OWASP Top Ten, latest version, 2017. https://owasp.org/www-project-top-ten/ Last accessed 2021-06-17.

[11] Mamun, M. S. I., Rathore, M. A., Lashkari, A. H., Stakhanova, N., & Ghorbani, A. A. (2016, September). Detecting malicious urls using lexical analysis. In *International Conference on Network and System Security* (pp. 467-482). Springer, Cham.

[12] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018, January). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSp* (pp. 108-116).

[13] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

[15] Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, *50*, 102419.

[16] Ferreira, P., & Antunes123, M. Benchmarking bioinspired machine learning algorithms with CSE-CIC-IDS2018 network intrusions dataset.

[17] Catillo, M., Rak, M., & Villano, U. (2020, April). 2l-zed-ids: A two-level anomaly detector for multiple attack classes. In *Workshops of the International Conference on Advanced Information Networking and Applications* (pp. 687-696). Springer, Cham.

[18] Kim, J., Shin, Y., & Choi, E. (2019). An intrusion detection model based on a convolutional neural network. *Journal of Multimedia Information System*, *6*(4), 165-172.

[19] Schulze, J. P., Sperl, P., & Böttinger, K. (2021). Double-Adversarial Activation Anomaly Detection: Adversarial Autoencoders are Anomaly Generators. *arXiv preprint arXiv:2101.04645*.

[20] Lallie, H. S., Shepherd, L. A., Nurse, J. R., Erola, A., Epiphaniou, G., Maple, C., & Bellekens, X. (2021). Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. *Computers & Security*, *105*, 102248.

[21] Ahmad, T. (2020). Corona virus (covid-19) pandemic and work from home: Challenges of cybercrimes and cybersecurity. *Available at SSRN 3568830*.

[22] Muthuppalaniappan, M., & Stevenson, K. (2021). Healthcare cyber-attacks and the COVID-19 pandemic: an urgent threat to global health. *International Journal for Quality in Health Care*, *33*(1), mzaa117.

[23] Elastic Licensing changes https://www.elastic.co/blog/licensing-change.

[24] OpenSearch Project Introduction https://aws.amazon.com/es/blogs/opensource/introducing-opensearch/


ProTego